Vers une détection des attaques de phishing et pharming côté client

Sophie Gastellier-Prevost

Ecole Doctorale EDITE

Thèse présentée pour l'obtention du diplôme de
Docteur de Télécom & Management SudParis

Doctorat conjoint Télécom & Management SudParis et Université Pierre et Marie Curie

Spécialité : Informatique

Par Sophie GASTELLIER-PREVOST

Vers une Détection des Attaques de Phishing et Pharming Côté Client

Soutenue le 24 Novembre 2011 devant le jury composé de :

Ludovic MÉ	Professeur Supélec, Campus Rennes	Rapporteur
Radu STATE	Chercheur associé Université du Luxembourg, Campus Kirchberg	Rapporteur
Guy PUJOLLE	Professeur Université Pierre et Marie Curie, LIP6	Examinateur
Laurent TOUTAIN	Maître de Conférences Télécom Bretagne	Examinateur
Franck VEYSSET	Chef du CERTA ANSSI/COSSI	Examinateur
Maryline LAURENT	Professeur Télécom SudParis	Directrice de thèse

Thèse n° 2011TELE0027

à l'homme qui partage ma vie et à tous ceux qui veillent sur nous…

Remerciements

Cher lecteur, je m'excuse par avance de la longueur de ces remerciements, mais - cette thèse étant devenue pour moi le symbole d'une histoire - je me dois de remercier à la fois ceux qui ont contribué à la thèse, mais également ceux sans qui elle n'aurait jamais pu voir le jour.

Les 4 années de thèse : Pour commencer, je tiens à saluer et remercier pleinement tous les artisans (souvent de l'ombre) de cette thèse. Alors, MERCI :

à **Maryline Laurent**, ma Directrice de thèse, pour son soutien, sa confiance, ses conseils, bref son encadrement. Depuis mon entrée dans le monde de l'enseignement et de la recherche, j'ai pu m'apercevoir que la motivation d'un thésard ne peut que rarement se suffire à elle-même. L'encadrement est tout aussi primordial. Le choix de la thématique et du Directeur de thèse étaient donc, à mes yeux, tout aussi importants. Pour l'un tu m'as laissé une totale liberté et pour l'autre, tu m'a dit oui sans hésiter. Sans toi, je ne me serais certainement pas lancée dans l'aventure. Ton amitié m'est précieuse.

à **Gérard Hébuterne** - mon Directeur de Département au démarrage de cette thèse - pour son soutien.

à **Hervé Debar**, mon Directeur de Département actuel. Tu es arrivé à point nommé, au moment où je doutais le plus. Nos discussions techniques toujours riches, parfois philosophiques, sont incontestablement celles qui m'ont fait lever la tête et m'ont empêché de m'arrêter à l'aube de la 3ème année.

aux étudiants SSR ou stagiaires qui - même indirectement, même succinctement - ont contribué à cette thèse : **Cécilien, Mickaël, Franck, Dafé, Haïfa, Bassam, Afsaneh, Victor, Samantha, Kévin, Francis, Samer et** mention spéciale à **Gustavo** pour son implémentation de *Phishark*.

à tous ceux qui ont répondu à mes sollicitations pour la réalisation des tests de pharming - indispensables aux résultats exposés dans ce document - aux quatres coins du monde : **Gilbert Dérus, Eric Gaillard, Eric Thrierr, Gustavo Gonzalez Granadillo (et sa famille), Quentin Couturier, Tony Cheneau, Tony Le-Huidoux, Uciel Fragoso, Pars Mutaf, Houssem Jarraya, Alain Riou, Dafé Gnabaly, Annie Leclercq et Yves Yoseph**. Votre patience et votre disponibilité m'ont été infiniment précieuses.

à **M. Ludovic Mé** et **M. Radu State** qui, en tant que rapporteurs ont eu la lourde tâche de me relire, et me font l'honneur de participer à mon jury.

à **M. Guy Pujolle**, **M. Laurent Toutain**, et **M. Franck Veysset** qui me font également l'honneur de leur présence à mon jury.

à mes collègues et vaillants supporters : **Corinne, Mounia, Brigitte, Isabelle, Patrice, Céline, Jocelyne, Françoise, Tony C., Shirley, Dominique, Sébastien, Patrick, Laurent, Claude, Nunzio** (je sais que j'en oublie et je m'en excuse), que ce soit pour leurs encouragements, leur expérience, leur aide ou simplement nos moments de franche rigolade (il en fallait bien un peu pour décompresser !)

à **Alexiane L'Hostis** pour sa précieuse rééducation.

à **mes proches**, pour leur soutien et leur patience. Les sacrifices personnels pour la réalisation de cette thèse sont nombreux, et égoïstement je vous les ai imposés.

Les 5 années avant thèse : Je ne peux en effet conclure ces remerciements sans revenir brièvement sur une période marquante de ma vie : en 2002, rien ne me laissait présager que, ce qui avait débuté comme un banal coup de fatigue, se révélerait finalement être le début d'une douloureuse et interminable descente aux enfers...

Le cauchemar aura duré près de 5 ans. Cinq années de maladie interminables...

S'en suivent 2 années de reconstruction physique mais surtout, la chance extraordinaire d'une deuxième vie, certes différente. Probablement trop contraignante pour certains mais, pour moi, une véritable renaissance.

Je tiens donc à remercier tout particulièrement les équipes médicales de l'Hôtel Dieu, puis de la Pitié-Salpêtrière, qui ont su me sortir de cet enfer, et me redonner une nouvelle vie. Je pense en particulier au **Docteur Célia Lloret-Linarès**, au **Docteur Cécile Ciangura** et à **Leïla**, une infirmière inoubliable.

MERCI également infiniment à tous ceux qui m'ont aidée de leur présence et leur soutien infaillible durant ces années infernales :

une poignée de collègues qui sauront aisément se reconnaître,

mes amis véritables, pour leur faculté d'adaptation aux contraintes les plus folles avec le plus grand naturel,

le **Docteur Jean-Claude Antognarelli**, un homme et un médecin doté d'un coeur en or,

le **Docteur Yves Chestier**, trop tôt disparu. Il avait cet abord un peu rude qu'ont les gens qui ont compris que la vie est trop courte et trop fragile pour s'encombrer de détails inutiles. Il était en réalité d'un talent inouï, d'une persévérance incroyable, d'une générosité et d'une disponibilité tout aussi inimaginables. Trop rares sont ceux en effet qui ont compris que le talent - aussi immense soit-il - n'est rien sans confiance, persévérance, humilité, partage, exigence et générosité. Pour ma part, je n'imaginais pas qu'une rencontre aussi anodine puisse se révéler aussi bouleversante. Il demeure à jamais irremplaçable.

mes parents, mon frère : toujours là, toujours prêts, quelle que soit l'heure, quelle que soit l'épreuve,

et enfin, **Florent, mon si merveilleux mari** : les mots n'appartiennent qu'à nous, mais ils seraient de toute façon insuffisants...

Je conclurais par ces quelques lignes : à la fin du mois de Janvier 2007, au point culminent de la maladie, j'ai été hospitalisée. Durant cette hospitalisation - qui s'est avérée ultérieurement avoir été la clé de la délivrance -, je m'étais promis que si la vie m'offrait la chance d'une deuxième vie, je ferai une thèse.

En septembre 2007, je m'inscrivais en 1ère année de thèse...

Aujourd'hui grâce à vous tous, je peux enfin dire : Promesse tenue.

Une page se tourne...

Bonne lecture !

Abstract

The development of online transactions and "always-connected" broadband Internet access is a great improvement for Internet users, who can now benefit from easy access to many services, regardless of the time or their location. The main drawback of this new market place is to attract attackers looking for easy and rapid profits.

One major threat is known as a *phishing* attack. By using website forgery to spoof the identity of a company that proposes financial services, phishing attacks trick Internet users into revealing confidential information (e.g. login, password, credit card number). Because most of the end-users check the legitimacy of a login website by looking at the visual aspect of the webpage displayed by the web browser - with no consideration for the visited URL or the presence and positioning of security components -, attackers capitalize on this weakness and design near-perfect copies of legitimate websites, displayed through a fraudulent URL. To attract as many victims as possible, most of the time phishing attacks are carried out through spam campaigns. One popular method for detecting phishing attacks is to integrate an anti-phishing protection into the web browser of the user (i.e. anti-phishing toolbar), which makes use of two kinds of classification methods : blacklists and heuristic tests. The first part of this thesis consists of a study of the effectiveness and the value of heuristics tests in differentiating legitimate from fraudulent websites. We conclude by identifying the decisive heuristics as well as discussing about their life span.

In more sophisticated versions of phishing attacks - i.e. *pharming* attacks -, the threat is imperceptible to the user : the visited URL is the legitimate one and the visual aspect of the fake website is very similar to the original one. As a result, pharming attacks are particularly effective and difficult to detect. They are carried out by exploiting DNS vulnerabilities at the client-side, in the ISP *(Internet Service Provider)* network or at the server-side. While many efforts aim to address this problem in the ISP network and at the server-side, the client-side remains excessively exposed. In the second part of this thesis, we introduce two approaches - intended to be integrated into the client's web browser - to detect pharming attacks at the client-side. These approaches combine both an IP address check and a webpage content analysis, performed using the information provided by multiple DNS servers. Their main difference lies in the method of retrieving the webpage which is used for the comparison. By performing two sets of experimentations, we validate our concept.

Keywords : Phishing, Pharming, Client-side, Heuristics, Web browser, DNS, Edit distance, N-gram, HTML source code.

Résumé

Le développement de l'Internet à haut débit et l'expansion du commerce électronique ont entraîné dans leur sillage de nouvelles attaques qui connaissent un vif succès. L'une d'entre elles est particulièrement sensible dans l'esprit collectif : celle qui s'en prend directement aux portefeuilles des Internautes.

Sa version la plus répandue/connue est désignée sous le terme *phishing*. Majoritairement véhiculée par des campagnes de spam, cette attaque vise à voler des informations confidentielles (p.ex. identifiant, mot de passe, numéro de carte bancaire) aux utilisateurs en usurpant l'identité de sites marchands et/ou bancaires. Au fur et à mesure des années, ces attaques se sont perfectionnées jusqu'à proposer des sites webs contrefaits qui visuellement - hormis l'URL visitée - imitent à la perfection les sites originaux. Par manque de vigilance, bon nombre d'utilisateurs communiquent alors - en toute confiance - des données confidentielles. Dans une première partie de cette thèse, parmi les moyens de protection/détection existants face à ces attaques, nous nous intéressons à un mécanisme facile d'accès pour l'Internaute : les barres d'outils anti-phishing, à intégrer dans le navigateur web. La détection réalisée par ces barres d'outils s'appuie sur l'utilisation de listes noires et tests heuristiques. Parmi l'ensemble des tests heuristiques utilisés (qu'ils portent sur l'URL ou le contenu de la page web), nous cherchons à évaluer leur utilité et/ou efficacité à identifier/différencier les sites légitimes des sites de phishing. Ce travail permet notamment de distinguer les heuristiques décisifs, tout en discutant de leur pérennité.

Une deuxième variante moins connue de cette attaque - le *pharming* - peut être considérée comme une version sophistiquée du phishing. L'objectif de l'attaque reste identique, le site web visité est tout aussi ressemblant à l'original mais - a contrario du phishing - l'URL visitée est cette fois-ci elle aussi totalement identique à l'originale. Réalisées grâce à une corruption DNS amont, ces attaques ont l'avantage de ne nécessiter aucune action de communication de la part de l'attaquant : celui-ci n'a en effet qu'à attendre la visite de l'Internaute sur son site habituel. L'absence de signes "visibles" rend donc l'attaque perpétrée particulièrement efficace et redoutable, même pour un Internaute vigilant. Certes les efforts déployés côté réseau sont considérables pour répondre à cette problématique. Néanmoins, le côté client y reste encore trop exposé et vulnérable. Dans une deuxième partie de cette thèse, par le développement de deux propositions visant à s'intégrer dans le navigateur client, nous introduisons une technique de détection de ces attaques qui couple une analyse de réponses DNS à une comparaison de pages webs. Ces deux propositions s'appuient sur l'utilisation d'éléments de référence obtenus via un serveur DNS alternatif, leur principale différence résidant dans la technique de récupération de la page web de référence. Grâce à deux phases d'expérimentation, nous démontrons la viabilité du concept proposé.

Mots-clés : Phishing, Pharming, Côté client, Heuristiques, Navigateur web, DNS, Distance d'édition, N-gram, Code source HTML.

Table des matières

1 Introduction de la thèse

Sommaire

1.1 Problématique et contexte de la thèse

L'avènement de l'Internet à haut débit et l'expansion du commerce électronique ont conquis bon nombre d'utilisateurs. Aujourd'hui, rien de plus simple que de faire ses opérations bancaires ou ses achats en ligne, en quelques clics. A priori, l'Internaute n'y voit que des avantages : un meilleur choix de produits, une comparaison des prix, un gain de temps considérable, etc.

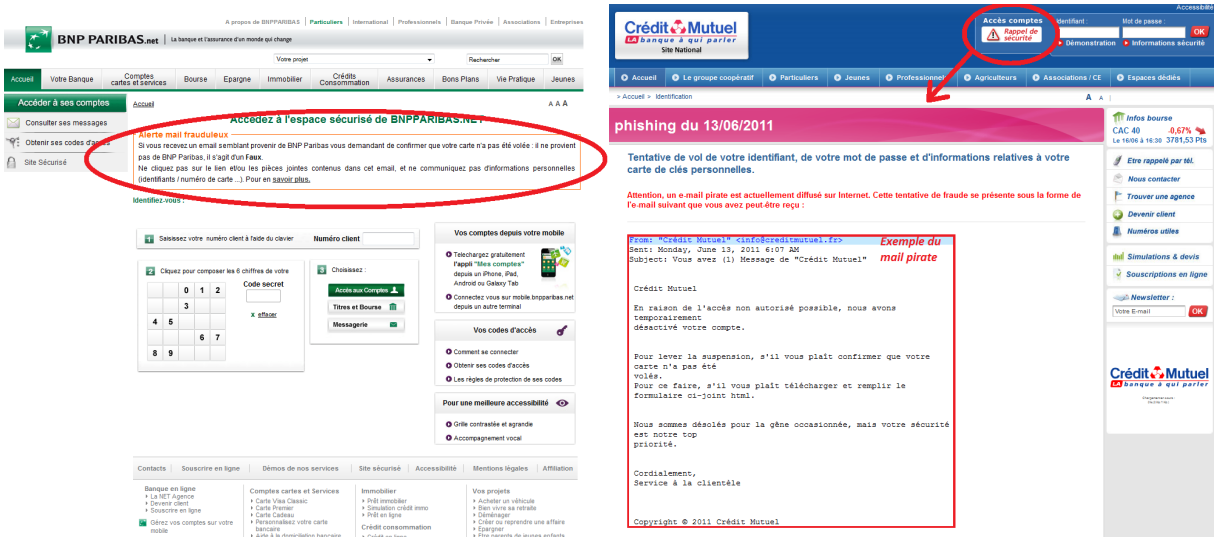
Pourtant, cette gigantesque plate-forme financière où se déroulent chaque jour des milliers de transactions bancaires, attire bon nombre d'attaquants à la recherche d'argent facilement gagné sur la crédulité des utilisateurs.

Dans ce contexte, au cours des dernières années, une nouvelle attaque a donc connu un essor extraordinaire : le phishing. Pour preuve bon nombre de banques diffusent désormais, de manière quasi-permanente, des messages d'alertes aux utilisateurs sur leurs pages d'authentification (cf. exemples en figure 1.1). La dernière étude publiée par l'APWG (pour *Anti-Phishing Working Group*), portant sur le deuxième semestre 2010, fait état de 26 000 à 31 000 nouveaux sites de phishing détectés chaque mois [APW10].

Très souvent véhiculées au travers d'emails frauduleux (également connus sous les termes *spam* ou *courrier indésirable*), les attaques de phishing tendent à voler des informations confidentielles aux utilisateurs (e.g. nom d'utilisateur, mot de passe, numéro de carte bancaire, etc.) en les redirigeant sur des sites webs contrefaits. Au fur et à mesure des années, ces attaques se sont perfectionnées. Désormais, on trouve des sites webs contrefaits qui visuellement - hormis l'URL visitée - imitent à la perfection les sites originaux.

Par manque de vigilance, l'utilisateur se retrouve alors piégé et communique, en toute confiance, des données confidentielles. En effet, qui prend le temps de vérifier l'exactitude de l'URL visitée ? Qui s'assure que l'envoi des informations confidentielles s'effectue de manière sécurisée, ou que le certificat du site visité est valide ? Pourtant, si l'utilisateur était attentif à tous ces indicateurs, il pourrait éviter bon nombre d'attaques de phishing.

Dans le même esprit, une version nettement plus sophistiquée du phishing a fait son apparition : le pharming. Dans ce type d'attaque, l'utilisateur - aussi vigilant soit-il - n'a qu'une chance infime de détecter la fraude (c.-à-d. la détection n'est envisageable que si la page web contrefaite est une mauvaise imitation visuelle du site original). En effet, l'objectif de l'attaque reste identique au phishing, le site web visité est tout aussi ressemblant à l'original mais - a contrario du phishing - l'URL visitée est cette fois-ci elle aussi totalement identique à l'originale. Impossible alors pour l'Internaute d'y échapper.



(a) BNP Paribas (b) Crédit Mutuel

FIGURE 1.1 – Exemples de messages d'alerte de deux banques françaises, pour sensibiliser les Internautes aux attaques de phishing

Cette thèse et les contributions associées s'inscrivent dans l'étude de ces deux problématiques que sont les attaques de phishing et de pharming, du point du vue côté client.

Techniquement, les attaques de phishing résident essentiellement dans la mise en ligne de sites webs contrefaits aux noms de domaines (plus ou moins) ressemblant à ceux utilisés par les sites usurpés (cf. exemple en figure 1.2). Pour une plus grande efficacité et une meilleure diffusion, ces attaques sont majoritairement propagées par des campagnes de spam (cf. exemple en figure 1.3).

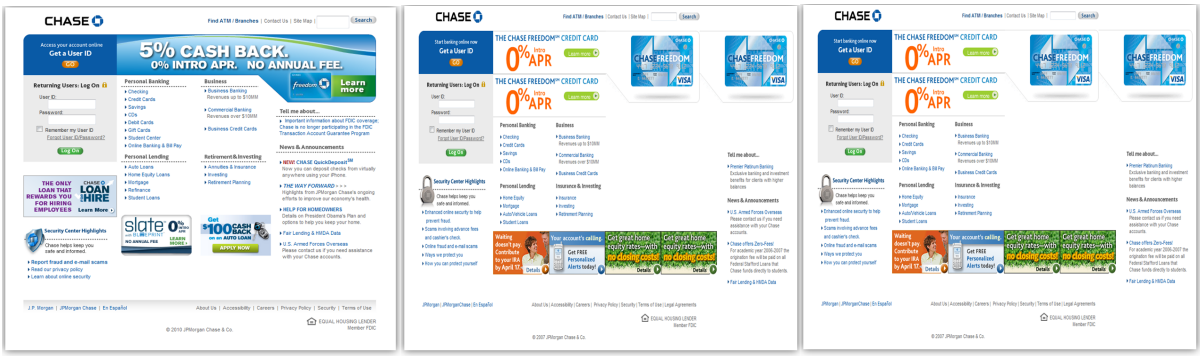
Les attaques de pharming reposent, quant à elles, essentiellement sur une corruption DNS (pour *Domain Name System*) qui redirige automatiquement les Internautes vers le site web contrefait mis en ligne par l'attaquant. Parce que l'URL visitée est l'URL légitime, impossible alors pour l'utilisateur de suspecter une éventuelle attaque, sauf si le rendu visuel du site web est de piètre qualité. A contrario du phishing, ces attaques de pharming ont donc un comportement plus passif - mais non moins efficace et redoutable - dans l'attente de la visite de l'Internaute sur son site habituel (cf. exemple en figure 1.2).

Notre étude s'est en premier lieu intéressée à l'étude du phishing et plus particulièrement à l'analyse de l'efficacité des tests heuristiques utilisés pour sa détection. Cette analyse s'appuie sur le développement d'une barre d'outils anti-phishing qui s'intègre dans le navigateur web du client. En marge, elle s'intéresse à comparer les performances de notre barre d'outils vis-à-vis de ses principales concurrentes.

Par la suite, nous nous intéressons aux attaques de pharming, et plus particulièrement aux attaques perpétrées sur le réseau client, pour lesquelles les efforts déployés côté réseau sont encore inefficaces. Par le développement de deux propositions, visant à s'intégrer dans le navigateur client, nous introduisons une technique de détection de ces attaques qui couple une analyse de réponses DNS à une comparaison de pages webs.

1.2 Contributions de la thèse et Organisation du mémoire

A travers cette thèse, nous avons en premier lieu analysé le phishing (cf. Chapitre 2). Parmi deux familles de techniques d'analyse - listes noires et tests heuristiques - qui s'intéressent à la détection des sites de phishing, nous nous sommes concentrés sur l'analyse des tests heuristiques (cf. Chapitre 3). Ensuite, basés sur cette première analyse, nous avons concentré nos efforts sur l'étude du pharming (cf. Chapitre 4) et l'élaboration d'une proposition de détection de cette attaque côté client (cf. Chapitres 5 et 6). Enfin, le Chapitre 7 rassemble nos conclusions ainsi que les futurs axes de recherche à explorer.



URL: https://www.chase.com/

URL: http://www.top-pharmacies.com/ePHARMACIES_languages/English/admin/help/chaseupdate/chaseupdate/chaseupdate/Signon.htm?section=signinpage&=&cookiecheck=yes&=nba/signin
ou
URL: http://www.chasebank-net.com

URL: https://www.chase.com/

(a) Site Légitime (b) Site Phishing (c) Site Pharming

FIGURE 1.2 – Exemple du site légitime de la banque CHASE
et d'éventuelles contrefaçons utilisant des techniques de phishing et pharming

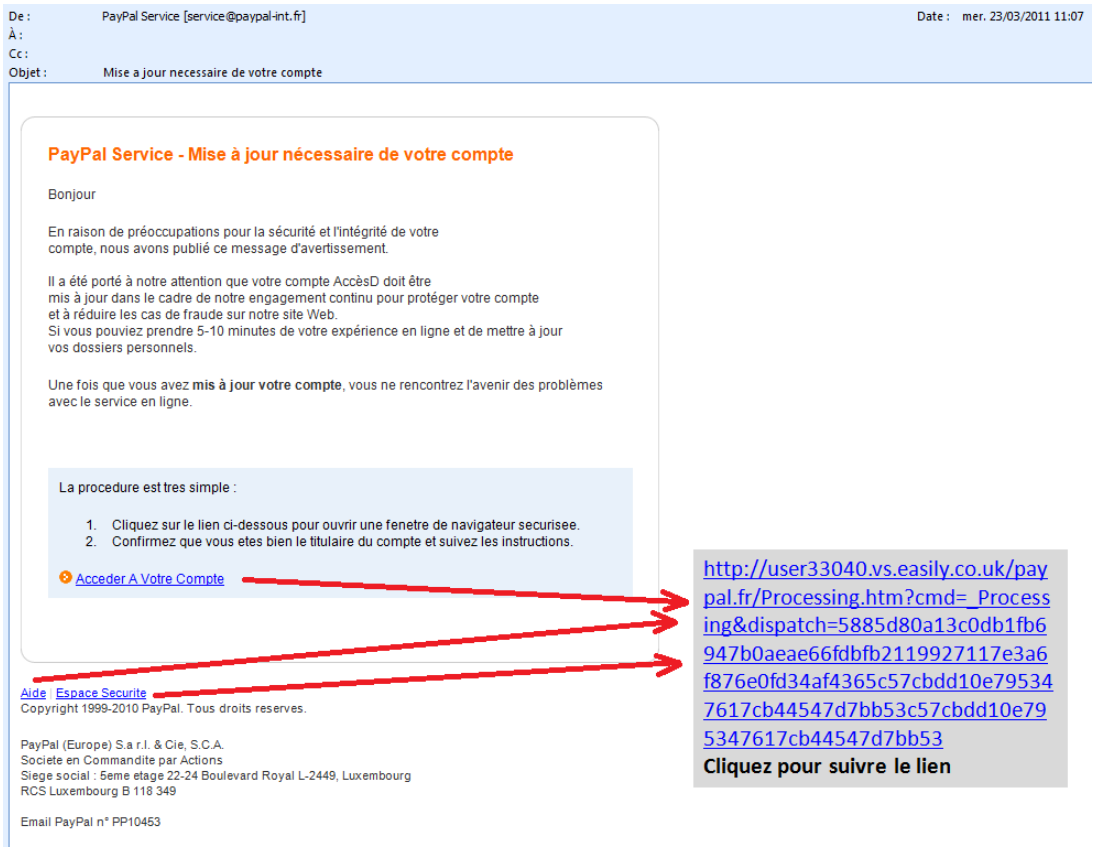


FIGURE 1.3 – Exemple de spam véhiculant une attaque de phishing

Précisons que l'organisation de ce mémoire est à l'image de notre étude, décomposé en deux grandes parties : la Première Partie est consacrée à l'étude du phishing, tandis que la Seconde Partie est dédiée à l'étude du pharming.

1.2.1 Analyse des heuristiques prédominants pour la détection des sites de phishing

Dans le Chapitre 2 nous avons étudié les attaques de phishing ainsi que leur principal vecteur de diffusion : le spam. Notre étude du spam et de ses diverses techniques de détection/prévention ont donné lieu à une première publication [Gas09].

A l'issue de cette première analyse - dans le Chapitre 3 -, nous nous sommes intéressés à un mécanisme de protection/détection des attaques de phishing facile d'accès pour l'Internaute : les barres d'outils anti-phishing, qui visent à s'intégrer dans le navigateur web. Ces barres d'outils - très nombreuses - reposent sur l'utilisation de l'une et/ou l'autre des techniques de détection suivantes : listes noires et tests heuristiques. Précisons que ces deux techniques sont toutes autant utilisées au sein des barres d'outils anti-phishing. Pourtant, l'utilisation des listes noires s'avère relativement contradictoire avec l'une des caractéristiques principales des sites de phishing, à savoir : leur durée de vie très courte. Cette dernière introduit en effet une très forte contrainte indispensable à une détection efficace : la nécessité d'une mise à jour en quasi-temps réel (c.-à-d. dès l'apparition du site de phishing) de la liste noire. De plus, l'utilisation d'une liste noire (qu'elle soit stockée côté client et/ou récupérée depuis Internet) introduit une zone de vulnérabilités de premier choix pour les attaquants (p.ex. via une attaque de type *Man-in-the-Middle*). A contrario, les tests heuristiques fonctionnent de manière autonome. Néanmoins, leur dimension plus "statique" (c.-à-d. si les critères de tests ne sont pas modifiés dans le temps) leur confère un degré de péremption non négligeable, d'où la forte association des deux familles de techniques. En effet, dès lors que les attaquants ont connaissance des tests heuristiques et/ou seuil de détection associés, ils peuvent essayer d'adapter leurs contrefaçons de sites webs afin qu'elles leurrent la détection.

Parmi l'ensemble des tests heuristiques utilisés (qu'ils portent sur l'URL ou le contenu de la page web), nous avons cherché à évaluer leur utilité et/ou efficacité à identifier/différencier les sites légitimes des sites de phishing. Pour ce faire nous nous sommes appuyés sur la conception de notre propre barre d'outils anti-phishing - nommée *Phishark* -, exclusivement basée sur les tests heuristiques. Au travers des tests effectués dans ce chapitre, nous avons pu mesurer l'efficacité des différents heuristiques évalués, définir les seuils de décision optimum pour les URLs testées et enfin, comparer ces résultats aux barres d'outils anti-phishing les plus courantes. Ces tests ont porté sur 20 heuristiques appliqués à 650 URLs légitimes et 950 URLs de phishing.

Cette étude a donné lieu à une deuxième publication [GGL11].

1.2.2 Proposition de méthodes de détection du pharming côté client

Notre contribution précédente a mis en lumière l'intérêt de l'étude du code source HTML pour l'identification des sites webs contrefaits.

En complément, dans le Chapitre 4 nous avons détaillé les différentes techniques de comparaisons applicables aux codes sources HTML. Nous y avons également explicité la problématique de corruption des informations DNS, véritable fer de lance des attaques de pharming. Cette analyse a par ailleurs démontré la difficulté de détection de ces attaques de pharming pour l'Internaute, ainsi que les nombreuses zones de vulnérabilités existantes côté client.

Dans les Chapitres 5 et 6, nous nous sommes donc intéressés à la portabilité de l'analyse du code source HTML à la détection des sites de pharming, auxquelles le client est encore trop exposé.

Nous y proposons deux approches visant à détecter le pharming côté client, basées sur l'étude du code source HTML du site web visité, combinée à des requêtes DNS.

1.2.2.1 Première proposition : vers un remplacement du nom de domaine

Dans un premier temps, dans le Chapitre 5, nous avons réalisé une étude préalable sur les hypothèses de travail afin de s'assurer que la vérification de l'adresse IP du domaine visité et l'analyse du contenu global de codes sources HTML sont des critères de décision pertinents de la légitimité d'un site.

Suite à cette étude, nous proposons une première approche visant à détecter les attaques de pharming côté client. Celle-ci s'appuie sur la vérification de l'adresse IP du domaine visité combinée à l'analyse du contenu de la page web visitée vs. des éléments de référence. Ces éléments de référence sont récupérés à partir des informations fournies par un serveur DNS, différent de celui auquel est connecté le client (c.-à-d. tel que proposé par son FAI - pour *Fournisseur d'Accès Internet* -). Ils comprennent une adresse IP (liée au domaine visité) ainsi que le code source d'une page web, tous deux dits de référence. La page web de référence est récupérée grâce à la génération d'une nouvelle requête HTTP créée à partir de l'URL initialement visitée par l'utilisateur, au sein de laquelle la zone de domaine est remplacée par l'adresse IP de référence. La validité de cette contribution a été vérifiée sur un panel de 108 URLs de login légitimes, évaluées depuis 11 localisations géographiques réparties sur 5 continents, et 37 couples de pages légitimes-contrefaites.

Cette première proposition a abouti à une troisième publication [GGL11b].

1.2.2.2 Seconde proposition : vers une redirection du GET HTTP

Dans le chapitre précédent, nous avons présenté une première solution de détection du pharming qui inclut la comparaison de la page web visitée auprès d'une page web dite de référence. Cette dernière est obtenue par le remplacement du nom de domaine par une adresse IP au sein de l'URL légitime. Bien que les résultats obtenus au travers de cette première proposition se soient avérés encourageants, ils ont également mis en exergue un certain nombre de faiblesses de l'approche proposée.

Dans le Chapitre 6, nous développons donc une seconde approche, amélioration de la proposition précédente. Nous y apportons notamment deux modifications majeures. Tout d'abord, la récupération de la page de référence se fait désormais grâce à une redirection de la requête HTTP vers l'adresse IP de référence, sans modification aucune de l'URL demandée. Puis, nous étudions et choisissons une méthode de comparaison des pages webs basées sur de multiples techniques d'analyse du code source HTML. Les tests réalisés dans cette seconde proposition portent sur 328 URLs de login légitimes, évaluées depuis 11 localisations géographiques réparties sur 5 continents, ainsi que sur 75 nouveaux couples de pages légitimes-contrefaites.

Nous terminons ce chapitre par une exposition des limitations associées aux deux approches proposées, ainsi que les verrous techniques demeurant à l'issue de notre étude.

La seconde proposition exposée dans ce chapitre a conduit à une quatrième publication [GL11].

Première partie :

Le phishing

2 Le phishing et son principal vecteur de diffusion : le spam

Sommaire

Difficile d'aborder la problématique du phishing sans évoquer son principal moyen de diffusion : le spam. En effet, les attaques de phishing sont majoritairement propagées/diffusées au travers de campagnes d'emails non sollicités. Les techniques de détection et de prévention associées s'en retrouvent donc souvent étroitement liées.

Dans ce chapitre, nous nous intéressons particulièrement au phishing. Nous démarrons en section 2.1 par une introduction à la problématique. En section 2.2, nous détaillons les propriétés d'un site de phishing. Puis, en section 2.3 nous discutons des vecteurs de propagations de ces attaques. Enfin, en section 2.4, nous traitons des différents travaux/méthodes de détection/protection qui s'y rapportent côté client. En complément, la section 2.5 discute des quelques mesures possibles côté réseau FAI/serveur web.

Ce chapitre fait partie de nos contributions : un Dossier Technique portant sur le spam a été publié dans la Collection *Sécurité des Systèmes d'Information* des *Editions Techniques de L'Ingénieur*, en Avril 2009 [Gas09].

2.1 Introduction au phishing

Avant de parler du phishing, il est nécessaire d'introduire le spam auquel il est étroitement lié. Un spam est un message électronique non sollicité, envoyé massivement à de nombreux destinataires, à des fins publicitaires ou malveillantes. L'ampleur du phénomène est conséquente puisque le spam représente aujourd'hui entre 88 et 91%[1] du volume total d'emails échangés [MAA11]. Les objectifs du spam sont divers et variés. Il sert notamment à propager[2] des publicités (p.ex. pour des produits contrefaits, pharmaceutiques), des canulars (c.-à-d. des fausses alertes à la population, des chaînes de solidarité censées prodiguer chance et bonheur, etc.), des logiciels malveillants : virus, chevaux de troie, etc. (p.ex. pour utiliser les postes d'Internautes comme membres d'un botnet[3]), des scams (qui visent à soutirer de l'argent grâce au leurre d'une forte rétribution financière à venir), ou encore des attaques de phishing.

Une attaque de phishing consiste à voler des informations confidentielles (p.ex. identifiant, mot de passe, numéro de carte bancaire, etc.) aux Internautes en usurpant l'identité de sites marchands et/ou bancaires. Pour ce faire, elle s'appuie sur deux maillons essentiels : la mise en ligne d'un site web contrefait qui usurpe l'identité d'un site légitime (p.ex. une banque, une plate-forme de jeux en ligne, etc.), et la mise en place d'une campagne de communication (typiquement basée sur du spam) afin d'attirer les Internautes.

2.1.1 Premier maillon : le site web contrefait

Techniquement, la mise en ligne du site contrefait s'appuie sur une page web d'accueil d'aspect (plus ou moins) ressemblant au site légitime. Meilleure sera la ressemblance et plus l'attaque sera efficace. Cette page web est accédée grâce à une URL également (plus ou moins) ressemblante à l'URL légitime. A nouveau, le degré de ressemblance de cette URL peut ajouter à l'efficacité de l'attaque. Il est toutefois indéniable que le rendu visuel global de la page sera plus décisif.

Pour illustrer ces propos, considérons deux exemples qui ciblent le site Facebook : les figures 2.1 et 2.2 montrent une première illustration de sites légitime/contrefait, tandis que les figures 2.3 et 2.4 en montrent une deuxième.

Le site contrefait développé dans le premier exemple est de prime abord, celui qui imite au mieux son pendant légitime. En effet, on ne détecte aucune erreur syntaxique, aucune modification de taille/type de polices de caractères, etc.). L'imitation apparaît parfaite. A contrario le deuxième exemple semble moins soigné, au sens visuel du terme. En effet, les zones 2 et 3 de la figure 2.4 démontrent des problèmes syntaxiques (c.-à-d. un manque de prise en charge des caractères spéciaux et accentués) ou des modifications de contenu (p.ex. on aperçoit que le site web contrefait est hébergé chez `T35 Hosting`, qui introduit des références publicitaires vers d'autres sites).

En ce qui concerne les liens accessibles depuis la page contrefaite, les zones 2 du premier exemple conduisent vers le site légitime. A contrario, les zones 3 et 4 de ce même exemple contiennent des liens contrefaits (cf. figure 2.2). Dans le deuxième exemple, tous les liens des zones 2 et 4 redirigent vers le site légitime (cf. figure 2.4). Seuls les liens amenés par l'hébergeur de site (cf. zone 3 de la figure 2.4) redirigent ailleurs (p.ex. `Free Domains` conduit à `http://www.domainsfree.org/`, `T35` redirige vers `http://www.t35.com`).

Enfin, si on s'intéresse à l'URL visitée (Zone 1 sur les figures 2.2 et 2.4), on voit que le deuxième exemple est cette fois-ci plus performant que le premier. En effet, l'URL contrefaite s'approche davantage de l'URL originale, en introduisant le nom de domaine légitime (c.-à-d. Facebook).

1. Ces chiffres - édités par le MAAWG (Message Anti-Abuse Working Group) - sont en provenance directe des FAIs. Ils portent sur plus de 500 millions de boîtes emails.
2. Liste non exhaustive.
3. ici, un botnet désigne un réseau de machines corrompues (appelées *bots* ou zombies) au travers d'Internet, utilisées comme rebond pour perpétrer des attaques.

FIGURE 2.1 – Zoom sur le site légitime italien de Facebook `http://it-it.facebook.com/`

FIGURE 2.2 – Zoom sur le site contrefait `http://genplus.altervista.org/` qui usurpe le site légitime Facebook illustré en figure 2.1

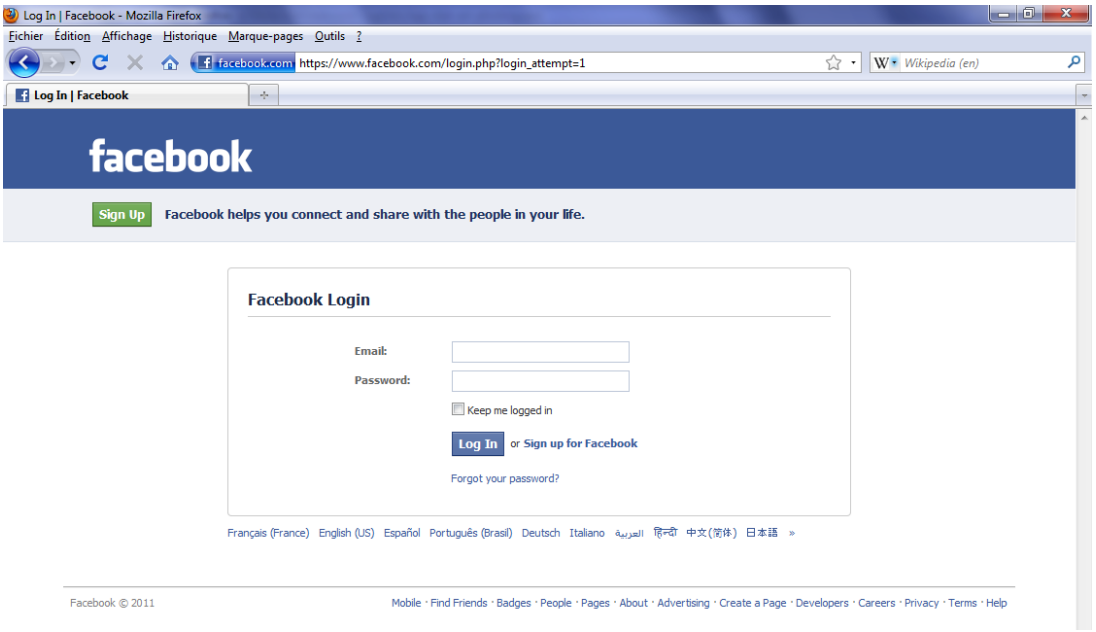


FIGURE 2.3 – Zoom sur le site légitime américain de Facebook `https://www.facebook.com/login.php?login_attempt=1`

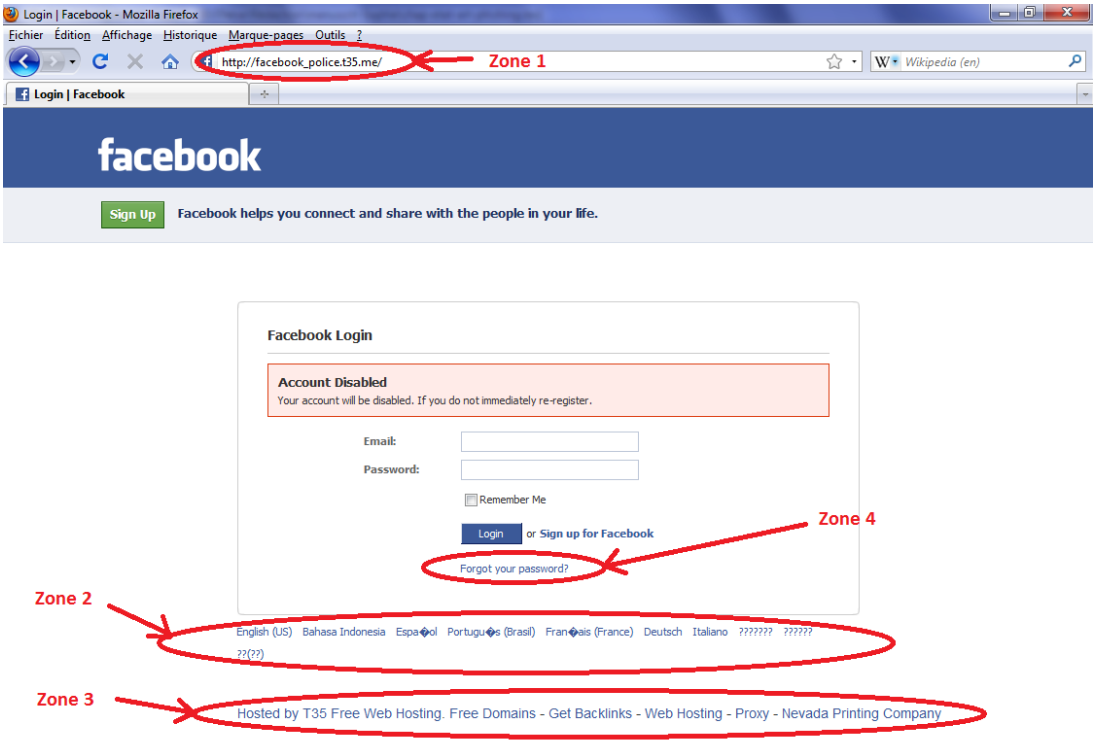


FIGURE 2.4 – Zoom sur le site contrefait `http://facebook_police.t35.me/` qui usurpe le site légitime Facebook illustré en figure 2.3

2.1.2 Second maillon : la campagne de communication

L'alliance d'un site web contrefait et d'une URL frauduleuse ne peut être suffisante pour attirer les victimes. En effet, les Internautes n'ont connaissance que de l'URL légitime. Il faut alors trouver un moyen de les amener à visiter l'URL frauduleuse. Pour ce faire, diverses techniques - exposées en section 2.3 - sont utilisables. Néanmoins l'une d'entre elles prédomine : la campagne de spam.

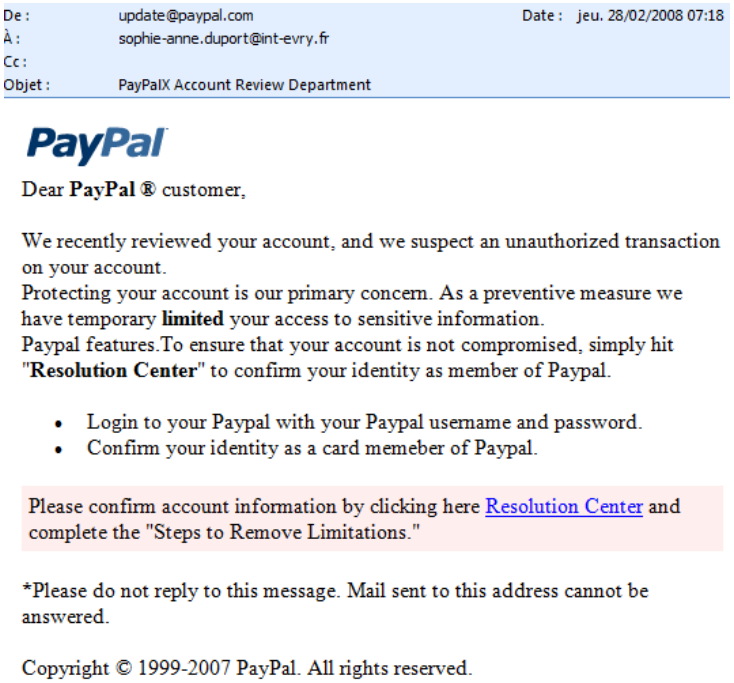
Ces spams qui servent à véhiculer les attaques de phishing sont eux aussi plus ou moins soignés. Prenons trois exemples :

- La figure 2.5 illustre un spam de haute qualité qui usurpe l'identité du site Paypal (`https://www.paypal.com`). Cet email ne comporte en effet que de minimes erreurs que bon nombre d'utilisateurs pourraient manquer. La majorité des indicateurs plaident en effet en faveur de la légitimité du site : le logo, l'émetteur affiché (c.-à-d. `update@paypal.com`), l'email destinataire qui apparaît valide, et l'URL frauduleuse utilisée qui - même si elle est masquée derrière l'élément `Resolution Center` - semblent faire référence au domaine légitime `http://paypal-secure-login.com/acc/login.php`. Seuls éléments d'alerte : un caractère en trop dans le titre du message (c.-à-d. le `X` après `Paypal`), ou l'URL de redirection qui utilise le protocole HTTP (et non HTTPS - pour *Hyper-Text Transfer Protocol Secure* -). De plus, l'email destinataire n'est pas celui du destinataire réel (bien qu'appartenant au même domaine) et enfin, en examinant le contenu de l'en-tête SMTP (pour *Simple Mail Transfer Protocol*) de l'email, on constate que l'émetteur réel est `akstcabilitamnsdgs@abilita.com`.

- La figure 2.6 illustre quant à elle un spam de qualité intermédiaire qui usurpe le Crédit Mutuel (`https://www.creditmutuel.fr/`). Les éléments qui plaident en faveur de la réussite de l'attaque sont : le logo légitime, des mentions de Copyright, un email émetteur affiché qui semble légitime même au sein de l'en-tête SMTP (c.-à-d. `service@creditmutuel.fr`), l'email destinataire ciblé et correct (c.-à-d. cohérent avec la boîte de réception où il est délivré), le lien de redirection masqué (derrière l'élément `Cliquez ici`) et enfin, un caractère d'urgence (grâce à l'email envoyé en priorité haute, et la menace explicite de résiliation de la carte bancaire sous 6 jours). A contrario, si on y regarde de plus près, le contenu du message est impersonnel et truffé de fautes. De plus, le lien vers lequel le client est emmené est `http://user33283.vs.easily.co.uk/credit/metuel/confirmation/compte/suspenstion/carte/bancaire/reconfirmation/informations/personelle/login.aspx/compte/`, sans rapport aucun avec le domaine légitime même s'il tente d'y faire vaguement référence dans le début de l'arborescence.

- Enfin, la figure 2.7 illustre un spam de phishing de basse qualité qui usurpe l'identité du site Paypal (`https://www.paypal.com`). En effet, l'email émetteur est imprécis et ne contient aucune référence au domaine légitime (c.-à-d. `carte de crXdit [systeme@security.net]`), l'email destinataire est impersonnel car masqué (`undisclosed-recipients`), le titre et le contenu du message sont truffés de fautes, ou encore l'URL de redirection est insuffisamment masquée : l'affichage indique en effet `definitefraudstart"www.artifizbox.com"definitefraudendhttps://www.paypal.com/fr/cgi-bin/webscr?cmd=_login-submit` pour une URL visitée `http://www.artifizbox.com/www.paypal.fr/cgi-bin/webscrcmd=_login-run/webscrcmd=_account-run/updates-paypal/confirm_paypal/`. Seul élément qui pourrait plaider en faveur de l'attaque : la mention d'un numéro de référence Paypal `PP-538-718-203`.

Pour terminer, notons également qu'un élément souvent utilisé par les attaques de phishing - afin d'accroître leur efficacité - est de limiter le temps de réflexion des Internautes. Pour ce faire, l'attaque introduit alors un caractère d'urgence avec lequel l'utilisateur doit réagir. Cette notion d'urgence peut aussi bien se trouver dans le spam (p.ex. dans la figure 2.6 où on voit la mention *Note :Si ce n'est pas achever le 10 Avril 2011, nous serons contraints de suspendre votre carte indéfiniment, car elle peut être utilisée pour des raisons frauduleuses...*) que dans le site web contrefait (p.ex. dans la figure 2.4 où on voit la mention *Account Disabled : Your account will be disabled. If you do not immediately re-register*).



De : update@paypal.com Date : jeu. 28/02/2008 07:18
À : sophie-anne.duport@int-evry.fr
Cc :
Objet : PayPalX Account Review Department

PayPal

Dear **PayPal ®** customer,

We recently reviewed your account, and we suspect an unauthorized transaction on your account.
Protecting your account is our primary concern. As a preventive measure we have temporary **limited** your access to sensitive information.
Paypal features.To ensure that your account is not compromised, simply hit "**Resolution Center**" to confirm your identity as member of Paypal.

- Login to your Paypal with your Paypal username and password.
- Confirm your identity as a card memeber of Paypal.

Please confirm account information by clicking here Resolution Center and complete the "Steps to Remove Limitations."

*Please do not reply to this message. Mail sent to this address cannot be answered.

Copyright © 1999-2007 PayPal. All rights reserved.

FIGURE 2.5 – Premier exemple de spam qui véhicule une attaque de phishing en usurpant Paypal

De : Crédit Mutuel [service@creditmutuel.fr] Date : lun. 04/04/2011 08:28
À : Sophie.Gastellier@it-sudparis.eu
Cc :
Objet : Votre Carte Bancaire a été suspendue

Crédit Mutuel
LA banque à qui parler

Bonjour client de Crédit Mutuel,

Votre Carte Bancaire a été suspendue .nous avons remarquer un problème sur votreCarte.

Nous avons déterminer que quelqu'un a peut-être utiliser Votre Carte sans votre autorisation. Pour votre protection, nous avons suspendue votre Carte de crédit.Pour lever cette suspension,**Cliquez ici**et suivez la procédure indiquerpourMettre a jour de votre Carte Crédit.

Note: Si ce n'est pas achever le 10 Avril 2011, nous serons contraints de suspendre votre carte indéfiniment, car elle peut être utilisée pour des raisons frauduleuses...

Nous vous remercions de votre coopération dans le cadre de ce dossier.

Merci,
Support Clients Service.

Copyright 1998-2011 Crédit Mutuel . Tous droits réserves.

FIGURE 2.6 – Deuxième exemple de spam qui véhicule une attaque de phishing en usurpant le Crédit Mutuel

Ce message a été envoyé avec une importance Haute.
De : carte de crXdit [systeme@security.net] Date : sam. 21/03/2009 13:38
À : undisclosed-recipients:
Cc :
Objet : Attention : Service-Veuillez retablire l'acess de votre compte PayPal .

Dans le cadre de nos mesures des securite, nous controlons regulierement les Activites
Encours dans le systeme PayPal. Au cours d'une recente verification,
Nous avons releve une probleme sur votre compte PayPal.

En etudiant votre compte, nous sommes rendu compte que nous avions besoin
D'informations supplementaires pour vous fournir un service securise.

definitefraudstart "www.artifizbox.com"definitefraudend https://www.paypal.com/fr/cgi-bin/webscr?cmd=_login-submit

Numero de reference : PP-538-718-203

PayPal

PayPal de sécurité et de la lutte anti-fraude Département.

FIGURE 2.7 – Troisième exemple de spam qui véhicule une attaque de phishing en usurpant Paypal

2.2 Anatomie d'un site de phishing

Un site web est un ensemble de pages webs, accédées au travers des URLs qui leur sont attachées. Analyser les caractéristiques d'un site de phishing - tel que développé au travers de plusieurs travaux précédents [PKKG10] [GPCR07] [PD06] [AHDT10] - revient donc à étudier les spécificités des URLs et/ou l'aspect/contenu de la page web contrefaite associée. Précisons en effet qu'un site web de phishing se résume en général à l'utilisation d'une unique page[1] contrefaite, qui conserve et fait appel à un maximum de redirections de liens du site légitime, typiquement pour l'accès à des informations complémentaires (p.ex. pour les pages d'aide).

2.2.1 Zoom sur l'URL

2.2.1.1 Structure d'une URL

Une URL est généralement composée des éléments suivants (cf. figure 2.8) :

1. Le protocole utilisé (p.ex. *http* pour une page web classique, *https* pour une page web de login, *ftp* pour un serveur de téléchargement, etc.), suivi des caractères *://* qui précèdent la désignation de l'emplacement de stockage de la ressource demandée.
2. Un nom d'hôte complet, appelé FQDN (pour *Fully Qualified Domain Name*) qui précise le nom de la machine qui héberge la ressource demandée. Ce FQDN est lui-même traditionnellement composé de 3 éléments : un nom d'hôte (p.ex. *www* pour un serveur web), un nom de domaine (p.ex. *yahoo*) et un TLD - pour *Top-Level Domain* - (p.ex. *fr*) qui est utilisé pour structurer/hiérarchiser la gestion des noms de domaines. Ce TLD est généralement représentatif d'une localisation géographique ou d'un type de site web (p.ex. EU pour Europe, .FR pour France, .COM pour des sites commerciaux, etc.). De plus amples détails sur la gestion des TLDs et des domaines sont disponibles en section 4.1.1.
3. Le chemin d'accès au sein de l'hôte spécifié, c.-à-d. l'arborescence de répertoires utilisés pour le stockage de la ressource, séparés par des /.
4. Le nom du fichier qui contient la ressource.

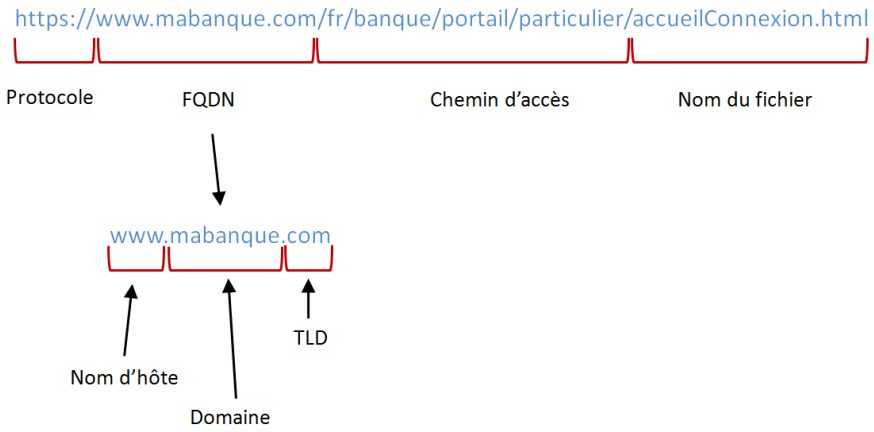


FIGURE 2.8 – Vue simplifiée de la décomposition d'une URL

Notons que le chemin d'accès et/ou le nom de fichier sont des éléments optionnels. Si l'hôte de destination a pré-configuré une page par défaut, il n'est pas nécessaire de spécifier ces éléments.

Précisons également que le schéma de décomposition d'une URL présenté ici est une version que l'on peut qualifier de "classique" et/ou simplifiée. Par exemple, la zone dite *nom d'hôte complet* est en réalité

1. ou un nombre très restreint de pages.

un peu plus complexe. En effet, sa syntaxe complète est : `<user>:<password>@<host>:<port>` [IET94], On y remarque la présence d'un caractère spécial : l'arobas(@)[1]. Cette syntaxe complète permet de spécifier des données d'authentification (c.-à-d. un nom d'utilisateur (`user`) et son mot de passe (`password`)) pour la connexion à un hôte (`host`) via un port destination choisi (`port`). Toutefois, seule la mention d'hôte est obligatoire. D'où la simplification courante de cette zone en *nom d'hôte complet*, puisqu'on y retrouve typiquement mention d'un FQDN. Néanmoins, d'autres variantes existent. On peut par exemple trouver une adresse IP en lieu et place du FQDN, ou une indication d'un numéro de port alternatif (c.-à-d. si le port destination est différent de 80 pour du HTTP, 443 pour du HTTPS, etc.).

URI, URL et URN : D'après le W3C (pour *World Wide Web Consortium*), URL, URI et URN sont trois notions différentes qu'il convient de distinguer [W3C01]. Un URI (pour *Uniform Resource Identifier*) est un terme global qui permet de désigner une ressource de façon unique. Un URI peut être une URL et/ou un URN.

Une URL (pour *Uniform Resource Locator*) désigne une ressource ainsi que sa localisation complète sur Internet et le moyen d'y accéder (c.-à-d. le protocole utilisé pour atteindre la ressource). Une URL est généralement utilisée pour la navigation web, l'accès aux mails, etc.

Un URN (pour *Uniform Resource Name*) est un identifiant permanent d'une ressource (p.ex. un numéro ISBN pour un livre), totalement décorrélé d'une quelconque notion de disponibilité.

2.2.1.2 Caractéristiques d'une URL de phishing

Plusieurs travaux précédents [PKKG10] [ZHC07] [GPCR07] [PD06] [CG06] [MG08] et notre étude approfondie des URLs de phishing font ressortir l'éventail des techniques utilisées par les attaquants afin de leurrer les utilisateurs. On peut notamment citer :

- **La substitution du FQDN par une adresse IP** : Certaines URLs de phishing utilisent une adresse IP en lieu et place d'un FQDN, ceci afin que le changement de FQDN - sans rapport aucun avec celui du site légitime - apparaisse moins visible. Par exemple, l'URL `http://74.220.215.65` est utilisée comme alternative au site contrefait `http://volleyballplayerz.com/` qui usurpe l'identité de la banque Natwest `http://www.natwest.com/`.

- **La déformation du domaine/FQDN légitime** : Il est très fréquent de rencontrer des URLs de phishing dont le FQDN est construit à partir d'une version déformée du nom de domaine ou FQDN (p.ex. via l'ajout, la modification ou le remplacement d'un - ou quelques - caractère(s)). Par exemple, l'URL de phishing `http://www.bhttle.net/` usurpe l'URL légitime `http://www.battle.net`, et l'URL contrefaite `http://faseboo.altervista.org/` usurpe l'URL légitime `http://www.facebook.com`.

- **L'utilisation de FQDNs/URLs longs** : Il est très fréquent de trouver des URLs de phishing constituées de FQDNs et/ou URLs à rallonge (p.ex. via l'utilisation de nombreux points (.)). Les URLs contrefaites `http://www.tsv1899benningen-ringen.de/chronik/update/alert/ibclogon.php` et `http://paypal.com.cg.ibin.webscr.cmd.login-submit.dispatch.5885d80a13c0db1f8e263663d3faee8d35d0e363192f28ea2.dgrrokpozefr.com/` en sont des exemples.

- **L'utilisation d'URLs très courtes** : A contrario du cas précédent, il est possible de rencontrer parfois des URLs très courtes. Pour ce faire, les attaquants ont recours à des services webs tels que celui de TinyURL [tin] qui crée des alias minimalistes redirigeant vers des URLs beaucoup plus longues. Par exemple, l'URL `http://www.amazon.com/Kindle-Wireless-Reading-Display-Globally/dp/B003FSUDM4/ref=amb_link_353259562_2?pf_rd_m=ATVPDKIKX0DER&pf_rd_s=center-10&pf_rd_r=11EYKTN682A79T370AM3&pf_rdt=201&pf_rd_p=1270985982&pf_rd_i=B002Y27P3M` qui se compose de 224 caractères peut être ramenée à l'URL `http://tinyurl.com/2enearw` de 26 caractères. Cette technique peut être utilisée afin de leurrer les moteurs de détection qui analysent les URLs à la recherche de comportements anormaux (c.-à-d. typiquement des tests heuristiques).

1. A ne pas confondre avec la présence d'un arobas (@) dans l'arborescence de l'URL qui permet d'apporter des indications complémentaires - appelées *attributs* - pour l'affichage de la page web (p.ex. pour spécifier un emplacement au sein de la page visitée, etc.)

- **L'utilisation de techniques de redirection** : Certains attaquants font appel à des techniques de redirection - plus ou moins masquées[1] - grâce à l'utilisation de caractères spéciaux (c.-à-d. @, //). Le caractère (@) doit alors être placé dans le FQDN, tandis que les caractères (//) doivent quant à eux être placés dans l'arborescence de l'URL, typiquement derrière la mention `http:`.
  L'URL contrefaite `http://usa.visa.com/track/dyredir.jsp?rDirl=http://200.251.251.10/.verified/` [CG06] est un exemple de redirection utilisant les caractères (//) : l'utilisateur croit visiter le site `http://usa.visa.com`, mais il est en réalité redirigé vers le serveur web possédant l'adresse IP 200.251.251.10.

- **L'utilisation de techniques d'encodage au sein de l'URL** : Tel qu'amorcé dans le point précédent, certaines URLs de phishing ont recours à des techniques d'encodage pour mieux masquer l'utilisation de caractères réservés, employés à des fins de redirections. Ces techniques sont utilisées tant pour éviter d'éveiller les soupçons de l'utilisateur, que pour leurrer d'éventuelles techniques de détection [Oll04]. Pour ce faire, les caractères réservés (p.ex. (://) et (@)) sont remplacés par leurs valeurs encodées. L'URL contrefaite `http://www.blizzard-accountlogin-security.com/login.asp?ref=https%3A%2F%2Fus.battle.net%2Faccount%2Fmanagement%2Fbeta-profil e.xml&app=bam&rhtml=y&rhtml=true` en est une illustration. On voit en effet apparaître la mention `https%3A%2F%2` dans l'arborescence de l'URL. En encodage ASCII, la valeur hexadécimale `%3A` correspond au (:), et la valeur `%2F` représente le (/).

- **La création de multiples dérivés d'une URL de phishing** : Les attaquants génèrent fréquemment de multiples URLs de phishing depuis une URL source, grâce à la modification/l'ajout de caractères tant au sein du FQDN que de l'arborescence (p.ex via le rajout de caractères aléatoires, de caractères d'indexation, etc.). Par exemple, les URLs contrefaites `http://fasemook.altervista.org/`, `http://faseboox.altervista.org/`, `http://fasebo.altervista.org/` et `http://faseboo.alterv ista.org/` usurpent le site Facebook (`http://www.facebook.com`) en altérant quelques caractères dans le FQDN. Un autre exemple : les URLs contrefaites `http://terabaap.hdfree.in/d.html`, `http://terabaap.hdfree.in/c.html` et `http://terabaap.hdfree.in/b.html` usurpent le site Orkut (`http ://www.orkut.com/`). Entre ces trois URLs, on voit que seul 1 caractère a été modifié, dans le nom de fichier.
  Précisons que des outils existent sur le web pour aider à la création de FQDN dérivés. On peut citer par exemple *Domain Typo Generator* [She].

- **La référence au FQDN/domaine légitime dans l'URL contrefaite** : Pour leurrer au mieux les utilisateurs, certaines URLs contrefaites mentionnent le FQDN/domaine légitime - dans le FQDN ou l'arborescence de la contrefaçon - sans pour autant effectuer une quelconque redirection. On peut citer par exemple l'URL de phishing `http://221.165.190.119/www.paypal.com/ws/www/x-us/webscr.html?cmd=x_login-run` qui utilise le FQDN de Paypal (`www.paypal.com`) - le site usurpé - en guise de nom de répertoire. Le site contrefait `http://verifymyfacebook.700megs.com/Index.html` mentionne quant à lui le nom de domaine légitime Facebook dans son FQDN.
  Précisons également que ce rappel du FQDN/domaine légitime est parfois réalisé via l'usage de l'arobas (@) dans l'arborescence de l'URL.

- **L'utilisation de multiples TLDs au sein du FQDN** : Il apparaît fréquent de rencontrer l'utilisation de multiples TLDs au sein des URLs de phishing, aussi bien dans le FQDN que dans l'arborescence de l'URL (typiquement lors de l'utilisation de techniques de redirection). On peut notamment citer les exemples d'URLs contrefaites : `http://www.ialp.org.br` qui utilise les TLD .ORG (pour *Organisation*) et .BR (pour *Brésil*), ou `http://www.click-here.us.ly/preview_login.htm.htm` qui utilise les TLD .US (pour *United States*) et .LY (pour la *Lybie*).

- **L'utilisation du protocole HTTP en lieu et place du protocole HTTPS** : Comme Ludl et al. [LMKK07] l'ont relevé dans leur étude, nous avons remarqué que la quasi-totalité des URLs de phishing rencontrées utilisent le protocole HTTP. Il est en effet plus rapide et moins dangereux pour un attaquant de ne pas utiliser de connexion sécurisée. En effet, il est ainsi moins exposé (vs. s'il

1. selon s'il y a encodage ou non.

devait obtenir un certificat valide) et évite toute alerte de sécurité émise par le navigateur et/ou le système d'exploitation en cas d'utilisation d'un certificat invalide. Les attaquants tirent ainsi avantage de la difficulté qu'ont les utilisateurs à distinguer un environnement sécurisé d'un environnement non-sécurisé [DT05] [FHH+02].

- **L'utilisation d'un numéro de port alternatif** : Tel que le rapporte régulièrement l'APWG[1] [APW10], certaines URLs de phishing font appel à une redirection de port dans la zone de FQDN (cf. section 2.2.1.1). Une URL contrefaite qui illustre cette redirection est : `http://186.97.10.96:8081/https/bancolombia.olb.todo1.com/olb/Init.php`, où l'on voit l'utilisation du port TCP 8081 pour accéder à la ressource demandée. Ce port 8081 - ou le port 8080 - sont des ports alternatifs au port 80, traditionnellement utilisé par le protocole HTTP.
  Cette redirection de port se justifie typiquement par le besoin de faire tourner un serveur web sur une machine corrompue (cf. section 3.4.1.1.4 pour plus de détails).

- **L'utilisation de mots-clés** : Il apparaît également que les URLs de phishing - qui usurpent des sites de login - utilisent de manière récurrente des mots-clés (p.ex. *login*, *signin*) en rapport avec la catégorie de sites usurpés au sein de leurs URLs. Par exemple, l'étude de Garera et al. [GPCR07] a retenu des mots-clés d'un minimum de 5 lettres (p.ex. *webscr*, *secure*, *banking*, *account*) pour détecter les URLs de phishing.

Notons que l'ensemble des techniques exposées ici peuvent être combinées entre elles pour construire une URL de phishing.

Précisons enfin que ces techniques ne sont pas l'apanage des URLs de phishing. Certaines d'entre elles sont également régulièrement rencontrées dans des URLs légitimes (p.ex. l'utilisation de mots-clés, de techniques de redirection ou d'encodage).

2.2.2 Zoom sur la page web

2.2.2.1 Synopsis d'une page web

Une page web est généralement constituée d'un document HTML (nommé *code source HTML*) auquel sont attachés un certains nombre de fichiers (p.ex. les images affichées, des scripts complémentaires pour l'ouverture d'encarts publicitaires en provenance d'un autre domaine, des feuilles de style, etc.). L'ensemble ainsi constitué est nommé *page web complète*.

Le *code source HTML* constitue le socle de la page web visitée. A ce titre, il est révélateur d'une grande partie de son contenu. Il permet par exemple de définir l'URL du site, donner des informations sur son propriétaire, définir les index de référencement pour les moteurs de recherche, ou encore il détaille le contenu visualisé par l'utilisateur (texte, images, etc.), que celui-ci soit intégré dans le corps du code source ou récupéré depuis d'autres ressources.

Un document HTML est constitué d'un assemblage de texte et de balises (ou *tags* en anglais). Les balises sont indiquées par des mots-clés (p.ex. TITLE, META SCRIPT, IMG, etc.) placés entre les caractères < et >[2]. On en dénombre plus de 90 sortes différentes.

Selon les standards définis par le W3C, un document HTML se décompose en 3 parties [W3Cb] (cf. figure 2.9) :

1. En préambule du code source, on trouve une déclaration de DTD (pour *Document Type Definition*). Celle-ci permet de préciser la version HTML utilisée pour le codage de la page. D'usage non-obligatoire, elle permet d'obtenir un meilleur rendu d'affichage de la page dans le navigateur web du client.
2. Un en-tête, nommé *HEAD* et identifié par une balise de même nom, qui contient des informations générales. On y retrouve typiquement :
   - le titre du document (balise TITLE),
   - des spécifications (p.ex. des mots clés, le nom d'auteur, une description générale de la page, des effets graphique associés au chargement de la page, etc.) indiquées dans les balises META.

1. cf. section 2.3 pour plus de détails sur cet organisme.
2. un début de balise est indiqué par <*mot-clé*>, tandis qu'une fin de balise est indiquée par</*mot-clé*>.

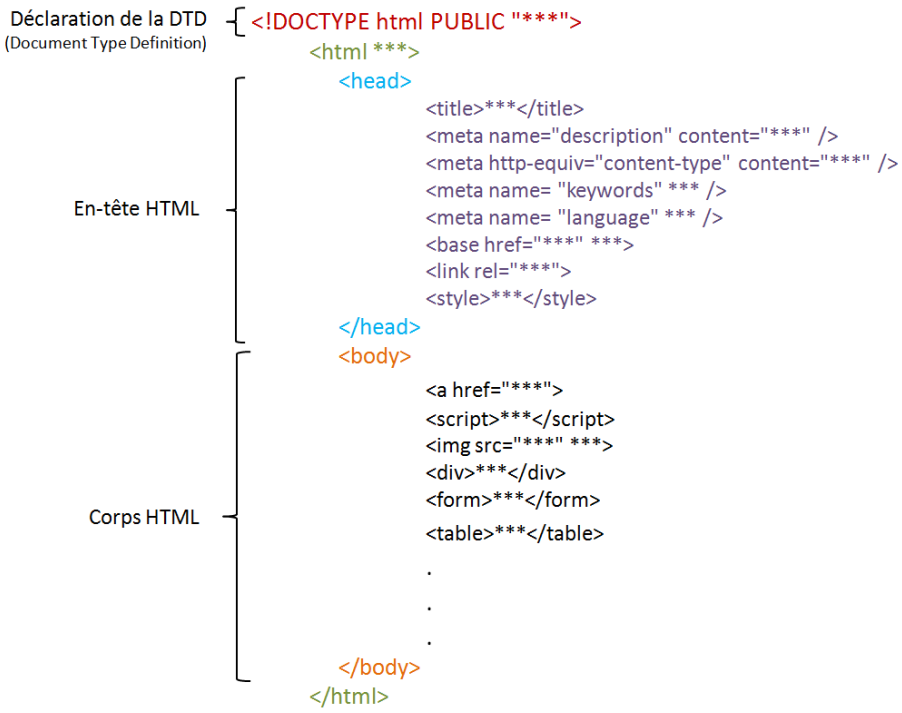


FIGURE 2.9 – Structure type du code source d'une page HTML

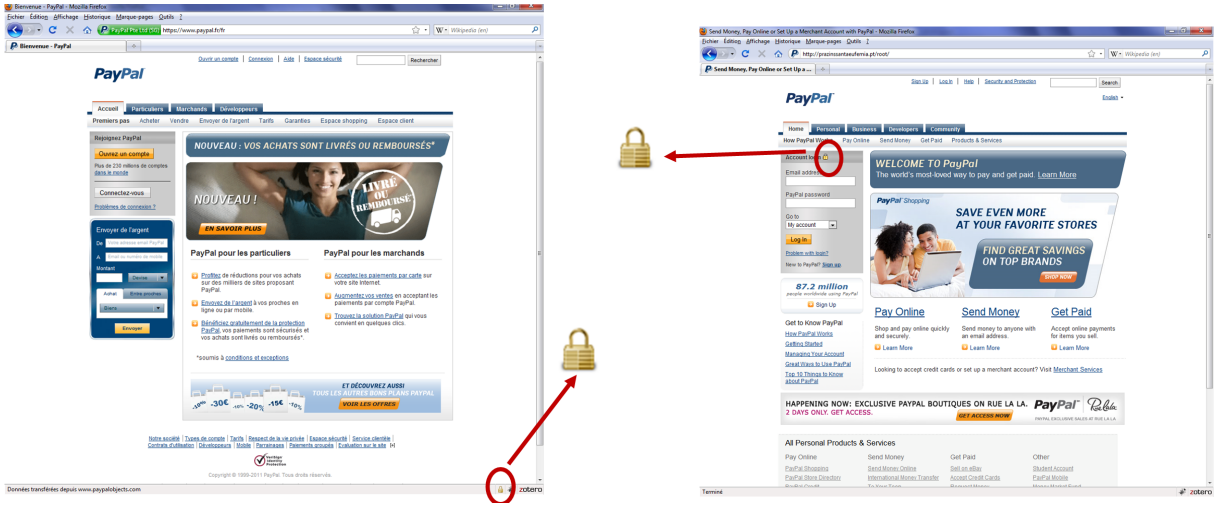
- une feuille de style (balise STYLE). Pour faciliter d'éventuelles modifications, on lui préfère plutôt une feuille de style spécifiée dans un fichier séparé, appelée par un lien hypertexte.
- le chemin d'accès absolu de tous les liens indiqués ultérieurement dans le document (balise BASE).
- des importations de documents externes (p.ex. pour des composants de feuille de style) via la balise LINK.

3. Le corps du document, nommé *BODY* et identifié par une balise de même nom, qui contient du texte, des tableaux, des images, des zones de saisie utilisateur, etc. Parmi les balises les plus utilisées dans cette partie du document, on peut citer notamment :
   - la balise A qui permet d'indiquer un lien hypertexte interne ou externe au document.
   - les balises FORM, INPUT utilisées pour définir des formulaires (typiquement des zones de saisie utilisateur).
   - la balise IMG pour insérer une image.
   - la balise DIV pour mettre en forme des données.
   - la balise SCRIPT pour insérer un bloc de code type Javascript.

L'arborescence constituée par l'assemblage et l'imbrication des différentes balises du code source HTML peuvent être désignées sous la terminologie structure/arbre DOM [W3Ca] (pour *Document Object Model*).

2.2.2.2 Caractéristiques d'une page de phishing

L'une des techniques préférées des attaquants - afin d'établir leurs contrefaçons - est d'utiliser des outils d'aspiration de sites webs dans l'objectif de s'approprier un maximum de contenu de la page légitime [Jam06]. On peut par exemple citer des outils comme Wget [Fou] ou WebWhacker [Squ] qui aident à la création de sites miroirs. Cette technique a le double avantage de simplifier l'élaboration du site contrefait, tout en rendant l'attaque moins détectable via l'utilisation d'un maximum de redirections légitimes et/ou l'utilisation de la structure originale. Notons néanmoins que quelques précautions peuvent être prises côté serveur web, afin de complexifier cette tâche (cf. section 2.5).



(a) Site légitime en HTTPS (b) Site contrefait en HTTP

FIGURE 2.10 – Utilisation de l'icône de cadenas dans Mozilla Firefox par deux sites Paypal

Des études [LMKK07] [PD06] [CG06] se sont attachées à distinguer les spécificités des pages webs de phishing. En complément, notre étude approfondie nous a laissés entrevoir que les pages contrefaites se perfectionnent, atteignant parfois une qualité d'imitation visuelle exceptionnelle, qui pourrait leurrer jusqu'aux utilisateurs les plus avertis.

Parmi les composants utilisés par les attaquants dans leurs contrefaçons de pages webs, nous pouvons commencer par citer ceux qui concernent l'aspect visuel :

- **L'intégration des logos et images du site légitime**.

- **L'utilisation du cadenas de sécurité au sein de la page web** : Toute page légitime affichée au travers d'une connexion sécurisée de type HTTPS, introduit automatiquement la présence d'un icône de cadenas dans la barre d'état du navigateur web client (typiquement en bas à droite du navigateur - cf. figure 2.10). Cet icône symbolise le certificat utilisé par le serveur web, élément indispensable au chiffrement des données client-serveur. A contrario, une page de phishing - qui utilise majoritairement une connexion non sécurisée - est dénuée de ce cadenas. Néanmoins, les attaquants ont trouvé une parade : pour leurrer les Internautes, ils insèrent l'icône au sein de la page web visitée, telle une image (cf. figure 2.10).

- **L'intégration de logos de sécurité** : Il est très fréquent de trouver au sein des pages webs de login légitimes, bon nombres de logos de sécurité utilisés comme gage d'une sécurité renforcée. La figure 2.11 en illustre quelques exemples. Les attaquants ont donc également recours à cette même technique.

- **L'utilisation de la structure globale de la page légitime** en conservant tailles et positionnements des images, textes, tableaux, formulaires, etc. du site original.

- **La conservation d'un maximum de redirections du site original**. Nous avons en effet remarqué que les sites de phishing conservent une - plus ou moins grande - proportion de liens originaux (cf. exemples détaillés en section 2.1). Ceci permet de minimiser le travail de l'attaquant et limiter les soupçons de l'utilisateur. En effet, nous remarquons sinon que bon nombre de liens additionnels renvoient vers une page d'erreur.

En regardant de plus près le contenu de la page web contrefaite (c.-à-d. son code source HTML), on peut également noter des caractéristiques typiques d'une contrefaçon :

FIGURE 2.11 – Exemples de logos de sécurité présents dans les pages webs

– **Les balises <TITLE> et <FORM> ne correspondent pas au domaine visité** : Par exemple le site contrefait (`http://www.top-pharmacies.com/ePHARMACIES_languages/English/admin/help/chaseupdate/chaseupdate/chaseupdate/Signon.htm?section=signinpage&=&cookiecheck=yes&=nba/signin`) qui usurpe le site légitime de la banque Chase contient *Chase Personal Banking Investments Credit Cards Home Auto Commercial Small Business Insurance* en guise de balise <TITLE>, ce qui est décorrélé du FQDN visité `www.top-pharmacies.com`.

– **Une large majorité des liens ne correspondent pas au domaine visité** : Typiquement des liens d'image (balise <IMG>), de redirection (p.ex. <A HREF>), etc. font appel au domaine légitime. En utilisant l'exemple du point précédent, on constate notamment que bon nombre de liens débutent par `http://www.chase.com/`.

Précisons toutefois que ces deux derniers points doivent faire l'objet de précautions particulières puisque nous avons également rencontré des sites légitimes qui répondaient à ces mêmes caractéristiques, faute de contenus adéquats ou explicites dans leurs balises. Par exemple, la balise <TITLE> de la page légitime de login Hotmail (`http://www.hotmail.com`) contient *Sign In*.

Notons enfin que l'ensemble des techniques exposées ici peuvent être combinées entre elles, et/ou avec les caractéristiques énoncées en section 2.2.1.2, pour élaborer le site de phishing.

2.3 Mise en œuvre du phishing

Diffusion des attaques : Le principal moyen de diffusion des attaques de phishing est le spam. Néanmoins, d'autres techniques existent pour attirer les utilisateurs sur des sites contrefaits ou détourner leurs données confidentielles [Oll04] [Emi05] [Jam06]. On peut citer par exemple l'utilisation :

– de bannières publicitaires insérées dans des sites webs,

– de publicités véhiculées sur les réseaux de messagerie instantanée,

– des moteurs de référencement pour indexer au mieux le site contrefait,

– des réseaux de Voix sur IP (VoIP) pour véhiculer des faux appels téléphoniques automatisés. Typiquement, ceux-ci informent l'utilisateur d'un blocage/d'une utilisation frauduleuse de son compte ou de sa carte bancaire, et l'invitent à rappeler un numéro qui leur demandera de saisir les informations confidentielles recherchées. Cette attaque est plus souvent désignée sous le terme de *vishing*, pour la contraction de VoIP et phishing.

– des réseaux mobiles via l'envoi de SMS/MMS (pour *Short Message Service*/*Multimedia Messaging Service*) de phishing, une technique connue sous la dénomination *smishing*, pour la contraction de SMS et phishing. Là encore, les messages véhiculés sont de nature similaire au point précédent : le danger d'une utilisation frauduleuse d'un moyen de paiement / d'un compte bancaire en l'absence de réaction de l'utilisateur. Celui-ci est alors invité à se rendre sur un site contrefait ou appeler une plate-forme frauduleuse pour résoudre le problème.

– d'outils malveillants tels que des *keylogger* ou *screenlogger* - récupérés typiquement par les Internautes lors de leur navigation web - qui espionnent les frappes clavier ou effectuent des captures écrans. On peut également mentionner des scripts malveillants qui visent à superposer de fausses zones de saisie de login sur les zones de saisie originales des pages légitimes, lors de la navigation web de l'Internaute [Oll04].

Localisation des sites de phishing : L'étude de McGrath et al. [MG08] met en exergue le fait que les sites de phishing ne sont généralement pas hébergés dans le pays annoncé par leur TLD. Leurs travaux démontrent en effet que près de 20% des domaines de phishing testés dans leur étude sont hébergés sur de multiples machines dispersées à travers le monde. Ils indiquent par ailleurs qu'un pourcentage significatif des adresses IP associées à ces sites de phishing sont utilisées par des particuliers (p.ex. 14% des sites contrefaits évalués sont hébergés par des particuliers situés aux États-Unis). Ceci s'explique notamment par la corruption des machines de ces utilisateurs, détournées afin d'être utilisées comme membres d'un réseau de botnet.

Sensibilisation au phishing et recensement d'URLs contrefaites : Face à la prolifération des attaques de phishing, les acteurs d'Internet (p.ex. FAI, banques) se mobilisent et diffusent très régulièrement des campagnes d'informations/de sensibilisation à destination des utilisateurs. La figure 1.1 - présentée dans le Chapitre 1 - et la figure 2.12 présentée ci-après en sont des exemples. En effet, la première figure illustre deux messages d'alerte diffusés par des banques sur leurs pages de login, tandis que la seconde figure illustre un email de sensibilisation émis par un FAI (c.-à-d. Orange) à destination de ses clients.

Au-delà de ces messages d'alerte, plusieurs organismes recensent et publient des listes d'URLs de sites contrefaits. Ces listes, plus connues sous la terminologie *liste noire* - ou *blacklist* en anglais - sont typiquement utilisées pour la détection des attaques de phishing (cf. section 2.4). Deux organismes particulièrement connus sont l'APWG [apw] et Phishtank [phi].

L'APWG est un consortium d'industriels ouvert - moyennant adhésion - aux banques, FAIs, éditeurs/constructeurs du domaine de la sécurité, ou organismes de recherche et/ou gouvernementaux. Pour la réalisation des études présentées dans ce mémoire, l'Institut Télécom/Télécom SudParis a donc signé un accord avec l'APWG, ce dernier s'autorisant à contrôler toute divulgation des données recueillies.

A contrario, Phishtank est une plate-forme d'accès libre où les utilisateurs soumettent et vérifient eux-même des URLs dites de phishing. Cette plate-forme est gérée par OpenDNS [ope].

Une durée de vie éphémère : Dans leurs travaux, McGrath et al. [MG08] ont calculé que la durée de vie moyenne d'une campagne de phishing était de 3 jours, 31 minutes et 8 secondes. Ils mentionnent par ailleurs que certaines campagnes ont une durée de vie extrêmement courte (33% d'entre elles n'excèdent pas 55 minutes), tandis 25% d'entre elles peuvent subsister durant 12 jours. De leur côté, Sheng et al [SWW+09] ont constaté que 63% des sites de phishing qu'ils ont testés ont vécu moins de 2 heures.

Globalement, on peut donc considérer que les sites de phishing ont une durée de vie très éphémère - et donc les listes noires associées également - d'où la multiplication d'URLs/domaines contrefaits.

2.4 Méthodes de détection/protection existantes côté client

Les techniques de détection/protection anti-phishing côté client découlent des différentes caractéristiques exposées en section 2.2. Précisons que ces techniques sont relativement similaires à celles rencontrées pour le spam, du fait que les deux attaques sont étroitement liées.

De manière générale, il est vivement recommandé aux utilisateurs d'équiper leurs postes clients avec les traditionnels anti-virus, anti-spyware, anti-spam, etc. en association avec la mise en œuvre de quelques règles de sécurité minimales (p.ex. restriction des paramètres d'acceptation des cookies et des contrôles ActiveX pour la navigation web, mise à jour régulière du système d'exploitation et

de :	"Orange votre service clients internet" <noreply@mailforge.orange.fr>
à :	[redacted]@orange.fr
date :	12/08/11 00:16
objet :	Emails frauduleux ? Ayez les bons réflexes !

voir l'en-tête complet

orange

Orange vous informe

Tentatives de phishing

Chère cliente, Cher client,

Nous attirons votre attention sur le fait que **des tentatives d'escroqueries par e-mail, aussi appelées "phishing", sont régulièrement menées.**

Le "phishing" est une technique consistant à usurper l'identité d'un tiers de confiance (banques, administrations, fournisseurs d'accès internet, etc.) pour soutirer des renseignements personnels à des fins malveillantes.

Ces emails reprennent généralement l'identité visuelle de différentes sociétés et vous invitent à modifier vos informations (coordonnées postales ou bancaires, identifiants et mots de passe) à partir d'un lien hypertexte.

Orange ne vous demandera en aucun cas par e-mail vos coordonnées bancaires, identifiants ou mots de passe.

Pour en savoir plus sur le phishing, l'assistance orange.fr met à votre disposition des compléments d'information accessibles depuis orange.fr > assistance > internet > sécurité > phishing.

Nous vous recommandons d'être vigilant lors de la consultation de vos e-mails et vous invitons à transférer tout message suspect à notre cellule spécialisée, à l'adresse suivante : abuse@orange.fr.

Adoptez les bons réflexes :

- **Traiter tout mail suspect comme indésirable,** sans y répondre, ni cliquer sur les liens.
- Par précaution, nous recommandons de **protéger votre ordinateur des risques liés à internet** : orange.fr > assistance > internet > sécurité > risques et prévention > protéger mon ordinateur

Nous vous remercions de votre confiance.

Votre service clients internet

FIGURE 2.12 – Le FAI Orange diffuse un email de sensibilisation aux attaques de phishing, à ses clients

des applications utilisés, etc.) [Oll04]. En complément d'autres techniques de détection/protection plus ciblées sur le phishing peuvent être utilisées. Elles sont exposées ci-après.

2.4.1 Le filtre idéal

Avant de détailler ces techniques, il est nécessaire d'introduire la notion de filtre anti-phishing idéal. Le doux rêve de tout(e) technique/outil qui cherche à s'attaquer à la problématique du phishing est de pouvoir détecter 100% des sites contrefaits, tout en autorisant 100% des sites légitimes.

Techniquement, cela se traduit par : 100% de sites contrefaits détectés, 0% de faux-positifs et 0% de faux-négatifs. Les faux-positifs (FPR pour *False Positive Rate*) correspondent au nombre/taux de sites légitimes classifiés à tort, comme sites contrefaits. Par similitude, les faux-négatifs (FNR pour *False Negative Rate*) correspondent au nombre de/taux de sites contrefaits classifiés à tort, comme sites légitimes.

Aucune technique de détection/protection existante à l'heure actuelle n'est capable d'atteindre cet objectif idéal. D'où la multitude d'approches proposées - qu'elles s'appliquent à l'URL, à la page web ou au moyen de diffusion de ces attaques - pour tenter de s'en approcher.

2.4.2 Au niveau des emails

Une première approche vise à bloquer/détecter les attaques de phishing au moment de leur diffusion en s'attaquant à leur principal vecteur de propagation : le spam.

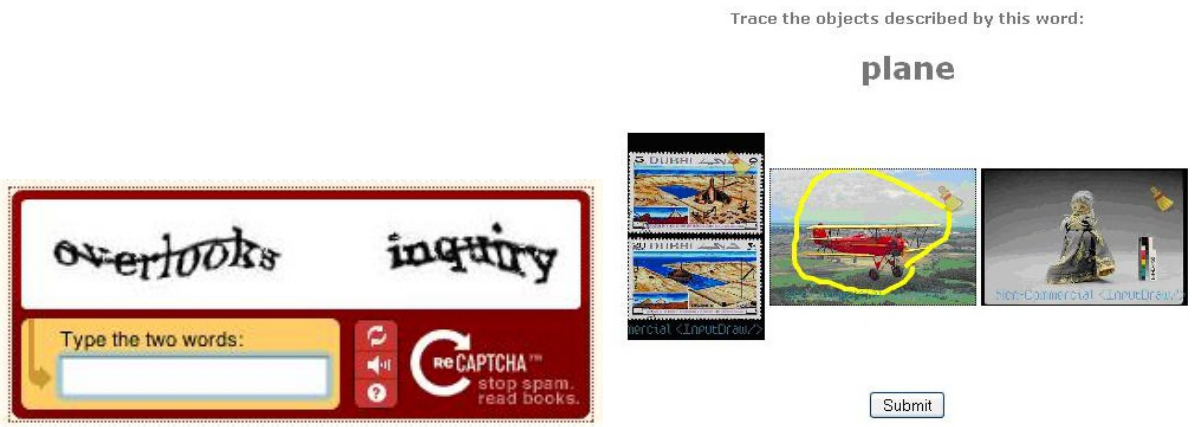
Pour ce faire, de nombreuses pistes sont proposées côté client. On peut notamment citer [Gas09] :

- un filtrage des emails - qui s'applique au corps du message et/ou à son en-tête SMTP - qui effectue :
  - la recherche de mots-clés interdits considérés comme caractéristiques d'une attaque, contenus dans un dictionnaire pré-établi.
  - un filtrage bayésien [Gra03] qui, après un apprentissage préliminaires sur *bons* et *mauvais* emails, classifie un email entrant selon les types de mots trouvés (c.-à-d. *bons* ou *mauvais*) et leur quantité.
  - une recherche de comportements dits anormaux - cette technique est également plus connue sous le terme de *tests heuristiques* - tels que : un email écrit en MAJUSCULES, un nom de domaine expéditeur anormalement long, un champ expéditeur vide, le couplage de plusieurs langues au sein du texte, etc. L'ensemble de ces comportements anormaux sont additionnés (selon les règles de pondération préalablement définies) et, si l'ensemble dépasse le seuil choisi, l'email est estampillé spam.
  - un filtrage selon le type de caractères/langues utilisés.
  - une analyse des images contenues dans le message [DGE07]. En effet, pour mettre en échec les techniques usuelles de filtrage par analyse du texte, certains attaquants encapsulent celui-ci dans des images.
  - une recherche de signatures. A l'image de la détection anti-virale, des signatures sont générées et pré-enregistrées à partir d'emails de spams avérés.
  - un filtrage à partir de listes noires (pour bloquer) et/ou blanches (pour autoriser) portant sur l'expéditeur (p.ex. nom de domaine, adresses IP, adresse email) ou les URLs contenues dans les emails. Ces listes peuvent être personnelles (c.-à-d. définie par l'utilisateur et/ou l'administrateur réseau auquel il appartient), ou émises par la communauté Internet.

- des techniques de vérification du domaine expéditeur de l'email qui s'assurent que la machine qui a expédié le message, est bien autorisée à le faire par le domaine pour lequel elle prétend agir. Ces techniques s'appuient typiquement sur des requêtes DNS[1].

- un filtrage dit par détection humaine, via l'utilisation des travaux de Turing [Tur50], pour essayer de détecter si le message a été expédié par un robot (p.ex. un attaquant) ou un être humain. En

1. cf. section 4.1.1 pour plus de détails.



(a) (b)

FIGURE 2.13 – Exemples de CAPTCHA utilisables pour une protection anti-spam [cap]

effet, pour leurs envois massifs de messages, les attaquants s'appuient sur des outils de génération automatique d'emails. Le filtrage mentionné ici est une technique amont qui vise à bloquer temporairement un expéditeur, jusqu'à ce qu'il réussisse un test que les robots ne peuvent théoriquement pas accomplir (p.ex. via un CAPTCHA - pour *Completely Automated Public Turing test to tell Computers and Humans Apart* -, qui est un test relativement simple et principalement visuel. Des exemples sont illustrés en figure 2.13). Ce test doit être réalisé par chaque expéditeur, uniquement lors d'un premier envoi d'email à destination d'un utilisateur protégé par cette technique. Si le test est réussi, l'expéditeur est ajouté à la liste blanche du destinataire.

– la signature des emails via l'utilisation de protocoles type S/MIME (pour *Secure/Multipurpose Internet Mail Extensions*) [Mai03] qui permettent d'authentifier l'expéditeur d'un message.

Ces pistes qui s'intéressent à bloquer les spams - et donc les attaques de phishing véhiculées par ce biais - sont certes intéressantes, mais insuffisantes. En effet la mise en œuvre de protocoles visant à signer les emails au niveau du poste client est peu facile d'accès pour des utilisateurs non-avertis. Ces protocoles s'en retrouvent donc insuffisamment déployés et, par conséquent, peu efficaces face à la problématique du phishing. Par ailleurs, les autres pistes évoquées ici ne sont ni infaillibles ni dépourvues de FPR/FNR. Enfin, les attaques de phishing pouvant être véhiculées par d'autres biais, cette approche ne peut être suffisante.

2.4.3 Au niveau du navigateur web

D'autres approches se focalisent quant à elles plus autour du navigateur web du client.

2.4.3.1 Sécurisation de la connexion client-serveur avec le protocole HTTPS

L'une d'entre elles repose sur l'établissement d'une connexion sécurisée avec le serveur web visité pour l'échange des données. Il s'agit ici des pages affichées via la mention HTTPS, accédées au travers du port TCP 443. Le protocole HTTPS n'est ni plus ni moins qu'une association des protocoles HTTP, et SSL ou TLS (pour *Secure Socket Layer* et *Transport Layer Security*). Il permet à la fois le chiffrement des données échangées entre client-serveur et l'authentification de ce dernier[1], grâce au(x) certificat(s) généré(s) par un tiers de confiance. Ce protocole HTTPS est typiquement utilisé pour le e-commerce.

Malheureusement, plusieurs études ont démontré le manque d'efficacité des indicateurs de sécurité affichés dans le navigateur web en cas de connexion HTTPS (cf. section 2.2.1.2). Le succès des attaques de phishing - qui usurpent en grande majorité des sites HTTPS grâce à des URLs contrefaites en HTTP - en est d'ailleurs la plus belle preuve.

1. au minimum. En effet, dans la majorité des cas, seul le serveur est authentifié. Pour certains usages il arrive que le client le soit également.

2.4.3.2 Alternatives pour la saisie des login/mot de passe

Pour pallier les problèmes des *keylogger* ou *screenlogger* (cf. section 2.3), certains sites d'e-commerce - typiquement les sites bancaires pour l'accès à la gestion des comptes sur Internet - proposent des zones de saisie des mots de passe à base de claviers virtuels (cf. exemple du site de login BNP Paribas, en figure 1.1 dans le Chapitre 1). D'autres sites ont quant à eux recours à l'utilisation de mots de passe à usage unique (ou OTP pour *One-Time Password* en anglais).

D'autres pistes [KK05] [RKKF07] [CHL08] proposent de stocker une base de données d'authentification côté client (c.-à-d. comprenant un couple login/mot de passe associé à une URL de site web légitime), en vue d'émettre des alertes dès lors que ces données sont saisies sur un serveur différent de celui qui est pré-enregistré. Dans la même veine, Dhamija et al. [DT05] ont proposé l'utilisation d'une fenêtre séparée pour la saisie des données d'authentification. Cette fenêtre dite de confiance repose sur l'utilisation d'une image de fond, préalablement choisie par le client et affichée par le serveur lors de la visite de ce dernier. Ross et al. [RJM+05] proposent quant à eux une extension pour navigateur web qui améliore la sécurité des mots de passe choisis par l'utilisateur. En effet, le plug-in s'interpose de façon transparente entre le client et les serveurs web visités, afin que chaque mot de passe utilisé soit différent et renforcé via une fonction de hash. D'autres alternatives similaires existent [WML06] [YS06]. Les défauts majeurs de ces solutions résident dans le besoin de stockage d'une base de login/mots de passe ou d'images choisies par l'utilisateur, qui introduisent des zones de vulnérabilités supplémentaires côté client ou côté serveur.

Enfin, une alternative proposée par Yue et al. [YW08] vise à fondre le mot de passe saisi par l'utilisateur sur un site de phishing, dans une multitude de mots de passe fantômes générés par l'outil et envoyés au site suspect. Cette méthode s'appuie sur une détection amont de la présence d'un site de phishing (p.ex. via une barre d'outils anti-phishing). Elle s'en retrouve donc exposée aux mêmes vulnérabilités.

2.4.3.3 Détection des signatures de pages webs contrefaites

Un autre style d'approche vise à comparer les pages de phishing à des bases de données de pages légitimes, en s'intéressant à leur aspect visuel (p.ex. structure générale, zones de texte, images, etc.). Par exemple, l'approche de Medvet et al. [MKK08] génère une base de signatures à partir des images de différents sites légitimes, auxquelles sont rattachées un certain nombre de mots-clés caractéristiques de leurs contenus/propriétaires. Ensuite, dès lors qu'une URL de phishing est suspectée, le moteur de recherche intégré à l'outil entre en action afin de retrouver la page légitime associée (c.-à-d. grâce aux mot-clés pré-enregistrés). Une signature de la page suspecte est alors générée pour comparaison avec la signature légitime sélectionnée. Dès lors que ces deux signatures sont trop similaires, une alerte est notifiée à l'utilisateur. Dans la même lignée, on peut également citer les approches de Hara et al. [HYM09], Wenyin et al [WHX+05] ou Chen et al. [CDM10].

Les inconvénients majeurs de ces approches résident à la fois dans le maintien de bases de données d'images et/ou de signatures d'images (dont découlent des zones de corruptions potentielles), et dans la quantité de faux-positifs générés dès lors que les pages webs ont recours à des contenus dynamiques.

2.4.3.4 Barres d'outils anti-phishing

Une dernière approche de détection se présente sous la forme d'une barre d'outils anti-phishing qui s'intègre dans le navigateur web. Dès lors que l'Internaute visite un site suspect, une alerte lui est notifiée. Le type d'alarme émise est très variable selon les éditeurs, allant d'un simple changement de couleur/texte affiché dans la barre d'outils (on parle alors de notification passive) jusqu'à un blocage de la navigation de l'Internaute dans l'attente d'une action de sa part (on parle alors de notification active). Ces barres d'outils anti-phishing - dont quelques exemples sont visibles en figure 2.14 - reposent sur l'utilisation de listes noires (pour vérifier l'URL visitée) et/ou de tests heuristiques (qui s'appliquent à l'URL et/ou au contenu de la page web). Diverses études ont été menées pour évaluer l'efficacité de ces barres anti-phishing [ZECH07] [WMG06] [ECH08] et/ou définir/évaluer les méthodes de détection qu'elles utilisent [LMKK07] [SWW+09] [GPCR07] [MSSV09]. Au global, il en ressort que cette approche - comme toutes les autres - n'est pas infaillible ni dépourvue de FPR/FNR. Néanmoins, il apparaît que les notifications actives sont incontournables et que la combinaison des deux types de techniques de

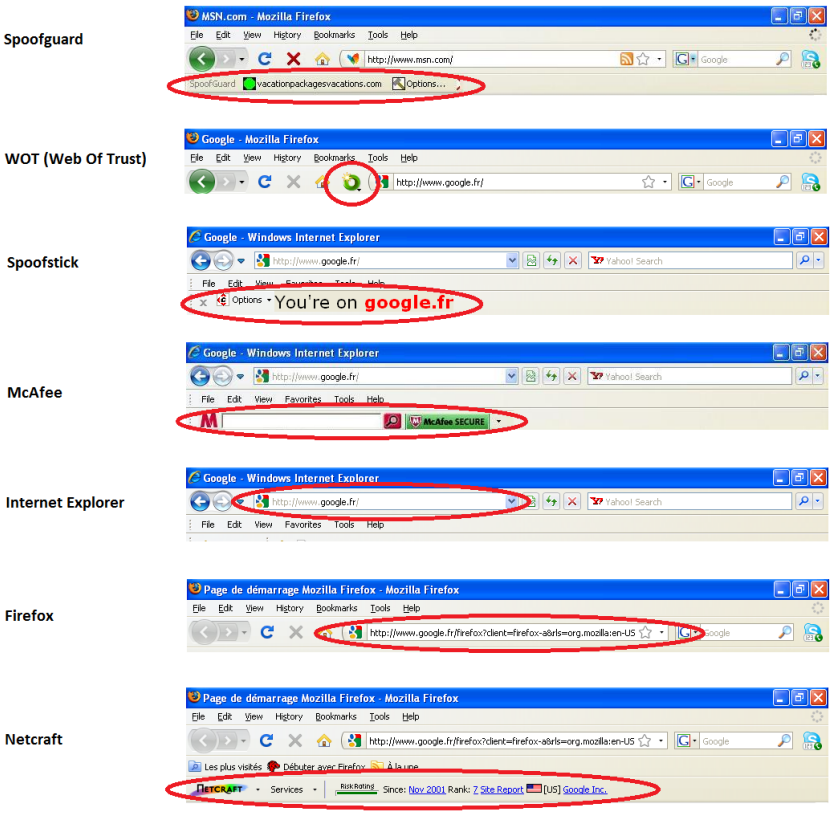


FIGURE 2.14 – Aperçu de plusieurs barres anti-phishing

détection (c-à-d. listes noires et tests heuristiques) sont un gage d'une meilleure détection (c.-à-d. qui limite les FPR/FNR). Une analyse plus détaillée de ces techniques de détection est menée dans le Chapitre 3.

Notons enfin que d'autres outils/mesures s'intéressent à la sécurité du navigateur web en général. Certains d'entre eux, qui peuvent répondre à la problématique des attaques de phishing diffusées via des outils/scripts malveillants (cf. section 2.3), sont évoqués en section 4.1.3.3.

2.5 Quelques alternatives côté réseau FAI/serveur web

Côté réseau FAI/serveur web, quelques mesures alternatives sont disponibles pour contrer les attaques de phishing. Néanmoins, celles-ci s'articulent majoritairement autour du spam.

L'une de ces alternatives concerne la sensibilisation des utilisateurs aux dangers de ces attaques, telle que nous l'avons déjà évoquée en sections 1.1 et 2.3.

Une deuxième alternative réside dans la mise en œuvre de mesures réseaux qui ciblent le spam. Sur ce point, de nombreuses approches sont évoquées et/ou utilisées par les FAIs ou entreprises [Gas09]. On peut par exemple citer la mise en place de *greylisting* pour rejeter temporairement les emails entrants, dans l'attente d'une réémission par le serveur de messagerie expéditeur (ce que les serveurs emails attaquants savent rarement faire). On peut également parler de mesures d'authentification des emails entre serveurs de messagerie (pour éviter cette tâche au niveau du poste client). Enfin, on peut également évoquer des règles de gestion de trafic telles que : A/ une limitation par les FAIs de l'utilisation du port 25 (qui correspond au protocole SMTP). Cette mesure cible principalement les particuliers, afin de réduire l'émission de spam via des machines corrompues, B/ une limitation du nombre/volume des

emails par expéditeur, C/ une limitation du nombre de destinataires d'un email, etc.

Néanmoins, cette alternative a ses limites puisque comme nous l'avons déjà évoqué, le spam n'est pas l'unique vecteur de propagation du phishing.

Enfin, une troisième alternative s'articule autour du serveur web visité. Au delà des mesures d'authentification vues en section 2.4.3, certaines approches consistent à compliquer la tâche des attaquants qui aspirent les sites webs légitimes. On peut par exemple citer l'utilisation de techniques d'encodage dynamiques - p.ex. en remplaçant des caractères ASCII par leur valeur hexadécimale - pour compliquer l'interprétation des liens contenus dans le code source de la page web [Fac05]. On peut également parler de l'utilisation de techniques qui visent à limiter la collecte d'adresses emails sur les pages webs, dans l'objectif de réduire le spam (p.ex. masquer/modifier les adresses emails en leur ajoutant des caractères vides ou en remplaçant le caractère @ par les caractères *(at)*, piéger les robots des spammeurs qui scrutent les pages webs dans des boucles dynamiques infinies, etc.) [Gas09].

2.6 Synthèse du chapitre

Ce chapitre a présenté les attaques de phishing ainsi que les mesures qui aident à s'en prémunir et/ou les détecter. Il apparaît évident qu'aucune des mesures exposées ici n'est suffisamment efficace ou infaillible pour éradiquer le problème. Néanmoins, parmi les techniques qui ciblent le phishing côté client, les barres d'outils anti-phishing se révèlent être un(e) outil/mesure particulièrement facile d'accès aux Internautes.

Comme pour toute mesure de détection, la véracité de la décision qu'elles délivrent est capitale. Les faux-négatifs sont évidemment bien plus dangereux que les faux-positifs. Néanmoins ces derniers auront tendance à lasser les utilisateurs, qui risquent alors de se détourner de leur solution anti-phishing dans les plus brefs délais.

La décision de légitimité établie par les barres d'outils anti-phishing s'appuie sur des listes noires et/ou des tests heuristiques. A l'image de la durée de vie des sites contrefaits, les listes noires semblent relativement éphémères. De plus, leur forte dépendance à une base de données - personnelle ou en provenance d'une tierce partie - fiable (c.-à-d. non corrompue) et suffisamment réactive, nous amène à penser qu'il serait dangereux de se limiter à leur simple utilisation (car trop de faux-négatifs potentiels). A contrario, les tests heuristiques semblent moins exposés aux vulnérabilités car plus autonomes. Mais la grande diversité des caractéristiques des sites webs (tant au niveau de l'URL que du contenu de la page web) peut les amener à délivrer des décisions plus enclines aux erreurs (c.-à-d. quelques faux positifs et faux-négatifs). Néanmoins, on peut imaginer qu'il y a une multitude de tests heuristiques à étudier : au moins autant que de signes caractéristiques d'un site de phishing. Quelle est leur efficacité ? Que vaut-il mieux analyser : l'URL ou le contenu de la page web ? Les tests heuristiques sont-ils tous égaux dans la détection des sites légitimes et contrefaits ? Quelle peut être leur pérennité ? La suite de notre étude s'essaie à y répondre.

3 Analyse des heuristiques prédominants pour la détection des sites de phishing

Sommaire

Le Chapitre 2 a introduit les attaques de phishing et leur principal vecteur de diffusion : le spam. Nous y avons notamment détaillé les principales caractéristiques d'un site de phishing ainsi que les moyens de détection/prévention associés. Dans notre volonté de nous focaliser sur le poste client de l'Internaute, nous avons vu qu'un moyen de détection facile d'accès qui leur est proposé s'installe/se configure dans le navigateur web. Il s'agit des barres d'outils anti-phishing qui se basent sur des listes noires (et éventuellement blanches) et/ou des tests heuristiques.

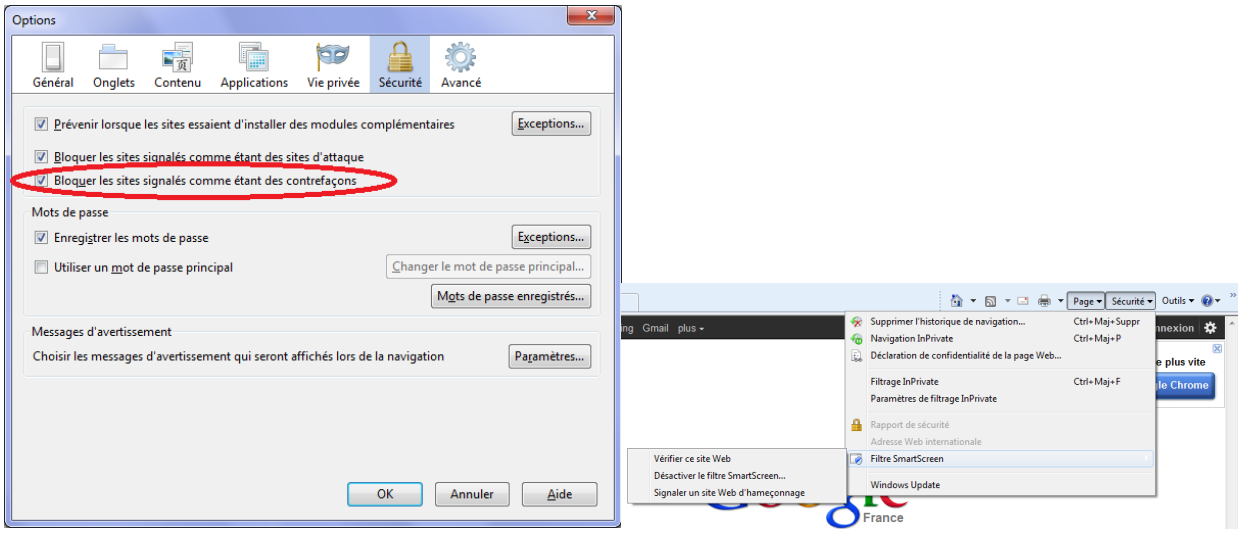
Dans ce chapitre, nous avons souhaité examiner plus avant ces barres anti-phishing. En particulier, nous nous sommes intéressés aux tests heuristiques qu'elles utilisent, afin d'en évaluer l'efficacité/la pérennité à distinguer les sites légitimes des sites contrefaits. Pour ce faire, nous nous sommes appuyés sur la conception de notre propre barre d'outils anti-phishing - nommée *Phishark* -, exclusivement basée sur les tests heuristiques.

En section 3.1, nous démarrons par une introduction qui place le contexte de notre étude et la situe par rapport aux travaux similaires existants. La section 3.2 explique ensuite la manière dont notre moteur de détection prend sa décision de légitimité/contrefaçon. Puis, la section 3.3 détaille les conditions d'expérimentation des différentes phases de tests. Dans la continuité - basés sur notre analyse des caractéristiques des sites de phishing vus dans le Chapitre 2 - nous détaillons en section 3.4 les 20 tests heuristiques étudiés ainsi que les seuils de classification légitime/contrefait associés. Nous y abordons également la barre d'outils anti-phishing développée pour évaluer l'efficacité des heuristiques. La section 3.5 s'attache ensuite à vérifier l'efficacité des heuristiques et seuils de décision choisis. En marge, nous comparons les performances de la barre développée aux barres d'outils anti-phishing les plus courantes. Puis, en section 3.6 nous déduisons de ces tests les heuristiques les plus pertinents pour l'identification des sites légitimes et contrefaits. Nous discutons également de leur pérennité en section 3.7. Enfin, nous terminons par un examen des problèmes rencontrés en section 3.8. Les tests effectués dans ce chapitre portent sur 650 URLs légitimes et 950 URLs de phishing.

Ce chapitre fait partie de nos contributions : les résultats de notre étude ont été publiés et présentés à la conférence *Sécurité des Architectures Réseaux - Sécurité des Systèmes d'information* (SAR-SSI) en Mai 2011 [GGL11].

3.1 Introduction à l'approche développée

Parmi les moyens de détection/protection du phishing, il y en a un qui est particulièrement facile d'accès aux Internautes : la barre d'outils anti-phishing. A l'origine, celle-ci se présentait sous forme d'un plug-in additionnel à installer au sein du navigateur. Au cours des dernières années, devant la prolifération des sites de phishing, les éditeurs ont intégré ces fonctionnalités nativement au sein de leurs navigateurs (p.ex. depuis la version 3 pour Mozilla Firefox, ou depuis la version 7 pour Microsoft Internet Explorer - cf. figure 3.1). Néanmoins, d'autres barres alternatives que celles proposées par les navigateurs - plus ou moins maintenues - peuvent être utilisées (p.ex. Spoofguard [CLT+04], Spoofstick, Netcraft, WOT (pour *Web of Trust*), etc. - cf. aperçus visuels en figure 2.14 de la section 2.4.3.4). Les logiciels anti-virus les plus courants (p.ex. BitDefender, Symantec Norton, McAfee Site Advisor, etc.) proposent également nativement des barres anti-phishing qui s'intègrent automatiquement (c.-à-d. au moment de l'installation de l'anti-virus) dans le navigateur. Précisons enfin que des attaquants explorent également ce filon, en diffusant régulièrement de fausses barres anti-phishing sur le web.



(a) Firefox 3.5

(b) Internet Explorer 8

FIGURE 3.1 – Configuration des fonctionnalités anti-phishing dans deux navigateurs webs courants : Mozilla Firefox et Microsoft Internet Explorer

Les barres d'outils anti-phishing basent leur détection sur l'utilisation de listes noires et/ou tests heuristiques. Précisons que ces deux techniques sont toutes autant utilisées. Elles ont d'ailleurs donné lieu à de nombreux articles et, bien que les avis divergent, il apparaît difficile de trancher définitivement en faveur de l'une ou l'autre de ces techniques.

L'utilisation des listes noires s'avère relativement contradictoire avec l'une des caractéristiques principales des sites de phishing, à savoir : leur durée de vie très courte (cf. section 2.3). En effet, bien que les listes noires s'avèrent plus exactes dans leur détection (c.-à-d. à priori, elles ne présentent pas de faux-négatifs), elles n'en sont pas moins incomplètes. De plus, elles nécessitent une mise à jour en temps réel ou presque (c.-à-d. dès l'apparition du site de phishing). Enfin, que la liste noire soit stockée côté client et/ou récupérée/testée depuis Internet, elle est une cible providentielle pour les attaquants (p.ex. via une attaque de type *Man-in-the-Middle*).

A contrario, les tests heuristiques semblent moins vulnérables car ils fonctionnent de manière autonome. De plus, ils ne nécessitent pas de mises à jour fréquentes. Néanmoins, croire que définir des tests heuristiques à un instant t peut s'avérer suffisant est un leurre. En effet, leur dimension plus "statique" leur confère un degré de péremption non négligeable. Dès lors que les attaquants ont connaissance des tests heuristiques et/ou seuils de détection associés, ils peuvent essayer d'adapter leurs contrefaçons de sites webs afin qu'elles leurrent la détection.

D'où la forte association des deux familles de techniques (listes noires et tests heuristiques) dans les barres anti-phishing. L'étude menée par Sheng et al. [SWW+09] a d'ailleurs démontré que leur utilisation combinée était un facteur indispensable à une détection efficace et optimisée.

Cette même étude montre notamment que 63% des 191 campagnes de phishing qu'ils ont étudiées étaient terminées au bout de 2 heures, alors que les listes noires testées (utilisées par McAfee Site Advisor, Symantec Norton, Netcraft, Internet Explorer, Chrome ou Firefox) n'étaient capables d'en détecter que 20% à l'instant *t=0*. Ces mêmes listes noires sont également mises à jour dans des délais très variables, aboutissant à une détection de 43 à 87% des URLs de phishing au bout de 12 heures. En complément, une étude menée par Kumaraguru et al. [KCA+09] a démontré qu'après réception d'un email de phishing, au moins 50% des victimes potentielles accédaient au site contrefait dans les 2 premières heures, et 90% dans les 8 heures.

Diverses suggestions/solutions sont proposées pour améliorer l'efficacité des listes noires. On peut notamment citer que Sheng et al. [SWW+09] proposent de générer celles-ci à partir des filtres anti-spam, partant du principe que les attaques de phishing sont majoritairement véhiculées par les emails. De leur côté, Prakash et al [PKKG10] ont développé un outil visant à prédire des listes noires d'URLs depuis une liste noire source, grâce à l'utilisation d'heuristiques (p.ex. changement de TLD, dérivation de nom de fichier, etc.). Leur théorie s'appuie sur le fait que les attaquants génèrent souvent de multiples URLs contrefaites très ressemblantes (p.ex. `http://g00gle.hdfree.in/1.html` et `http://g00gle.hdfree.in/2.html`, ou `http://e-baltikums-mailbox.com/different.files/geoga.html` et `http://e-baltikums.info/different.files/arta.html`).

Toutefois, il n'en demeure pas moins que les listes noires semblent majoritairement inefficaces au moment où la probabilité d'accéder au site contrefait est la plus forte. Nous avons donc choisi de nous intéresser aux heuristiques sur lesquels repose la détection dans cette phase clé.

Plus particulièrement, nous nous sommes intéressés à évaluer l'efficacité des tests heuristiques, qu'ils portent sur l'analyse de l'URL et/ou sur l'étude du contenu de la page web visitée. Pour ce faire, nous nous sommes appuyés sur le développement de notre propre barre anti-phishing - *Phishark* - conçue tel un plug-in pour Mozilla Firefox, premier navigateur web open source [web] au niveau mondial, et N° 1 en Europe devant Internet Explorer [Reu11].

Nous y avons intégré 20 tests heuristiques répartis en 6 catégories. Trois catégories s'intéressent à l'analyse de l'URL visitée, tandis que les trois autres se focalisent sur l'étude du contenu de la page web.

Le moteur de détection a été étalonné, puis vérifié, sur des jeux d'URLs légitimes/contrefaites différents. A partir des résultats obtenus lors de la vérification, nous avons déterminé les catégories de tests heuristiques les plus déterminantes (c.-à-d. celles conduisant à des scores négatifs pour les listes noires, et celles conduisant à un score positif pour les listes blanches) pour la différenciation des sites légitimes et contrefaits.

Positionnement par rapport aux travaux similaires : Les travaux précédents qui s'apparentent le plus à notre étude ont été amorcés dans le Chapitre 2. Nous les détaillons plus avant ci-après afin de nous positionner :

- L'étude de Zhang et al. [ZECH07] s'est intéressée à évaluer/comparer l'efficacité de 10 barres d'outils anti-phishing, dont la décision de légitimité s'articule principalement autour de blacklists. Sur les échantillons d'URLs testées, un manque de réactivité dans la mise à jour des blacklists utilisées est apparu. Sur ces points, nos études/avis se rejoignent. Néanmoins, leur étude n'apporte aucune mesure d'efficacité des tests heuristiques (les uns par rapport aux autres, ou vs. les blacklists).

- Les travaux de Ma et al. [MSSV09] se sont focalisés sur une étude de l'URL pour en déduire une méthode de classification des sites légitimes/contrefaits. Plusieurs des critères pertinents qu'ils utilisent ont été retrouvés dans une partie amont de notre étude (cf. section 3.4.1.2). Cette dernière nous a permis d'aboutir à une liste de triplets déterminants que nous utilisons lors de notre analyse de l'URL visitée. A contrario de notre étude, l'utilisation de ces critères est le socle de leur détection/classification. Un autre point de divergence de nos travaux réside dans le fait qu'ils ne s'intéressent pas à l'analyse du code source HTML.

- L'étude menée par Ludl et al [LMKK07] est probablement celle qui s'apparente le plus à nos travaux. Elle s'est intéressée à évaluer l'efficacité de deux blacklists - utilisées par Mozilla Firefox et Internet Explorer - pour la détection des sites de phishing, ainsi qu'à distinguer les caractéristiques prédominantes pouvant amener à une décision de légitimité/contrefaçon d'un site. Leurs travaux aboutissent à l'élaboration d'un arbre de décision qui s'articule aussi bien autour de l'analyse de l'URL que du contenu de la page web. Néanmoins il apparaît difficile de vraiment comparer nos deux études puisque leurs tests heuristiques sont peu détaillés (en terme de contenus étudiés et/ou scores décisionnels associés). On peut néanmoins dire que pour les sites légitimes, leurs résultats portent sur davantage d'URLs que nous en avons testés. Leur taux de faux-positifs obtenu est de 0.43%. Sur les pages de phishing, nos bases d'URLs sont de tailles comparables, mais leur taux de faux-négatifs est relativement important : 16.97%. Il semble également que nos tests heuristiques qui s'intéressent à l'URL sont plus nombreux. Enfin, leur étude ne discute pas de l'éventuelle pérennité des tests utilisés.

3.2 Décision de légitimité/contrefaçon

La décision de légitimité/contrefaçon prise par le moteur de détection de notre barre d'outils baptisée *Phishark*, repose sur le calcul d'un score global établi en additionnant - avec un poids équivalent - l'ensemble des scores obtenus pour chaque heuristique :

$$\text{Score global (p)} = \sum_{i=1}^{20} \text{Score(i)}$$

où *(p)* représente la page analysée/visitée dans le navigateur
et *(i)* le numéro d'heuristique

Si le moteur de détection ne peut aboutir à une décision (c.-à-d. le score global est = 0), le site visité est considéré comme *Risqué* (notre barre d'outils affiche *Risky Site*). Si le score global est > 0, le site visité est indiqué *Légitime* (notre barre d'outils affiche *Legitimate Site*). Sinon (c.-à-d. si le score global est < 0), le site visité est décidé *Contrefait* (notre barre d'outils affiche *Phishing Site*).

Les seuils de décision définitifs associés à chaque heuristique sont déterminés au travers de la combinaison de 3 facteurs : 1/ d'éventuels seuils de décision relevés au travers de précédents travaux qui ont servi de base à notre étalonnage, 2/ de nos études complémentaires approfondies sur les pages légitimes/contrefaites, qui ont abouti à des critères supplémentaires et/ou des seuils affinés, 3/ ou encore de notre phase d'étalonnage du moteur de détection qui a permis d'arriver au meilleur compromis (c.-à-d. à un minimum de décisions erronées) dans la détection des sites légitimes/contrefaits.

3.3 Conditions d'expérimentation

Afin d'évaluer la pertinence des tests heuristiques utilisés par les barres d'outils anti-phishing, nous avons réalisé deux types de tests : 1/ des tests portant sur des sites légitimes, et 2/ des tests portant sur des sites de phishing.

Ces deux types de tests ont été réalisés par 3 phases différentes : 1/ une première phase dite d'étalonnage qui a permis de définir des seuils de détection optimisés, 2/ une deuxième phase de vérification des performances du moteur de détection *Phishark*, en comparaison avec d'autres barres d'outils anti-phishing courantes, et 3/ une troisième phase d'identification des heuristiques déterminants[1].

Précisons que pour améliorer la lisibilité des explications données dans ce chapitre, les listes d'URLs légitimes sont ultérieurement nommées *whitelist*, tandis que les listes d'URLs contrefaites (c.-à-d. de phishing) sont désormais nommées *blacklist*.

3.3.1 Établissement des listes d'URLs

Une des difficultés premières de notre étude réside dans la possibilité de tester des sites de phishing. En effet, de par leur durée de vie très courte, les sites de phishing doivent être testés dès leur apparition. Pour sélectionner ces URLs, nous nous sommes appuyés sur les sites de l'APWG [apw] et de Phishtank [phi] qui délivrent des bases d'URLs de sites contrefaits. Nous avons choisi de sélectionner des sites de phishing "validés" à 100%. En effet, les bases de données communiquées par l'APWG et Phishtank indiquent un niveau de confiance (c.-à-d. il est avéré et vérifié que l'URL présentée est un site de phishing), variant de 50 à 100% pour les URLs mises à disposition. Nous avons pris soin de sélectionner - pour chaque phase de tests - des URLs de phishing d'aspects très variables, répondant à l'ensemble des caractéristiques exposées en section 2.2. Précisons toutefois que les URLs de phishing de type HTTPS sont rares.

De plus, ces URLs de phishing ont été récupérées et aussitôt testées par blocs de 50 à 100 URLs maximum, pour limiter la quantité d'URLs périmées. Même en procédant ainsi, nous avons eu des taux de pertes atteignant jusqu'à 7% par jeu d'URLs récupérées.

Les URLs de phishing collectées - récupérées entre Janvier et Septembre 2010 - usurpent majoritairement des sites bancaires (p.ex. Bank of America, Paypal, Chase, etc.), des sites d'e-commerce (p.ex. eBay, etc.), des sites d'email (p.ex. Hotmail), des sites de réseaux sociaux (p.ex. Facebook) ou autres (p.ex. jeux en ligne avec RuneScape).

Les whitelists ont quant à elles été générées à partir de diverses sources. On peut notamment citer le Top 1000 des sites les plus visités délivré par Google [Gooc], le Top 500 des sites les plus visités publié par Alexa [Al], le Top 100 des sites les plus populaires édité par Netcraft [Net], la whitelist proposée par Google [Goob], des URLs de sites bancaires récupérés grâce au site Levoyageur [lev], etc.

Nous avons pris soin - pour chaque phase de tests - de multiplier les secteurs d'activité, le type de site visité (HTTP et HTTPS), le langage des pages webs, les types d'URLs sélectionnées (FQDN simple ou suivi de multiples niveaux d'arborescence), ou encore les TLDs (Top-Level Domain).

L'ensemble des URLs légitimes utilisées pour cette étude ont été collectées entre Avril 2009 et Septembre 2010.

Bien que les sites de phishing visent à usurper en grande majorité des sites de login, notre sélection d'URLs légitimes ne s'est pas restreint à cette catégorie de sites. En effet, notre moteur de détection s'exécute à chaque URL visitée. Il se doit donc de délivrer les bonnes décisions quel que soit le type de site web visité, d'où la diversité d'URLs sélectionnées.

3.3.2 Phase d'étalonnage

Une des phases les plus délicates de l'étude développée dans ce chapitre réside dans l'ajustement des seuils de décision associés à chaque heuristique.

Pour ce faire, la phase d'étalonnage s'est déroulée en deux temps : 1/ nous avons cherché à définir les seuils de décision optimum pour les sites de phishing testés (c.-à-d. un maximum de décisions

1. Notons que cette troisième phase a pour vocation d'aider à affiner les heuristiques et seuils de détection associés, lors d'une étude ultérieure (cf. Chapitre 7).

négatives), puis 2/ nous avons cherché à définir les seuils de décision optimum pour les sites légitimes testés (c.-à-d. un maximum de décisions positives).

A partir de ces résultats obtenus en 2 temps, nous avons ensuite réalisé un compromis pour aboutir à des seuils de détection optimum pour les deux types d'URLs : légitimes et contrefaites. En effet, chaque modification de seuil peut affecter une liste positivement et l'autre négativement. Il a donc fallu procéder à plusieurs séries de tests successifs avant d'aboutir aux tableaux optimum présentés pour chaque liste (whitelist et blacklist) en section 3.4.

Cette phase d'étalonnage a porté sur une whitelist de 150 URLs, et une blacklist de 200 URLs issues du site de Phishtank.

3.3.3 Phase de vérification et de comparaison aux autres barres d'outils

Dans cette deuxième phase, nous avons voulu vérifier l'efficacité de notre moteur de détection. En marge, nous en avons profité pour le confronter aux performances des barres d'outils anti-phishing les plus courantes.

Les tests réalisés dans cette phase ont porté sur une whitelist de 500 nouvelles URLs, et une blacklist de 520 nouvelles URLs issues des sites de Phishtank (à hauteur de 69.81%) et de l'APWG (à hauteur de 30.19%).

Barres anti-phishing sélectionnées : A partir de travaux précédents [ZHC07] [WMG06] et d'une étude que nous avons menée sur les barres anti-phishing les plus courantes, nous avons sélectionné 4 barres d'outils qui apparaissent comme les mieux classées en terme de performance (c.-à-d. qui délivrent les meilleurs taux de détection avec un minimum de FPR/FNR) et les mieux maintenues. Ainsi, nous avons comparé les performances de *Phishark* avec :

- Le navigateur web ***Mozilla Firefox v.3.6.7*** qui intègre un moteur de détection anti-phishing basé sur des listes blanches/noires (parmi lesquelles celles de Google et de Phishtank), la liste noire étant rafraîchie toutes les 30 minutes [Moz]. D'après leurs travaux, Zhang et al. [ZECH07] présument également que le moteur de détection de Mozilla Firefox utilise des tests heuristiques.
- La barre anti-phishing ***Netcraft v.1.4.1.5*** qui s'installe telle un plug-in dans le navigateur web Mozilla Firefox. Elle appuie sa détection sur l'utilisation des listes blanches/noires propres à Netcraft (auxquelles l'Internaute peut contribuer) : à chaque page web visitée, la barre d'outils envoie l'URL de la page à contrôler aux serveurs Netcraft pour vérification. Cet envoi est non sécurisé [AN08]. En complément, la barre anti-phishing utilise des tests heuristiques tels que la vérification de l'adresse IP du site, le pays hébergeur, le nom du serveur web, l'âge du site, etc.
- Le navigateur web ***Internet Explorer v.8.0.601.18702*** qui intègre un moteur de détection anti-phishing nommé *SmartScreen Filter*. Celui-ci utilise des listes blanches/noires maintenues par Microsoft. La liste noire provient du service en ligne *Microsoft URL Reputation Service* qui s'appuie sur des listes noires externes à Microsoft, auxquelles les Internautes contribuent. En complément, elle utilise des tests heuristiques [Mic].
- La barre ***Web of Trust (WOT) v.20100503***, qui s'installe telle une extension dans le navigateur web (au choix : Mozilla Firefox, Internet Explorer, Safari, Opera ou Google Chrome). Elle utilise des tests heuristiques (p.ex. la popularité du site web, la localisation du serveur web et sa réputation, etc.), ainsi qu'une liste noire qui inclut des URLs délivrées par Phishtank [Ser06].

3.3.4 Phase d'identification des heuristiques déterminants

Dans cette troisième phase, nous avons cherché à identifier les heuristiques prédominants qui permettent de distinguer un site légitime d'un site contrefait, et réciproquement.

Pour ce faire, nous avons regardé les scores attribués à chaque catégorie d'heuristiques (les catégories sont détaillées en section 3.4), pour l'ensemble des URLs testées. Le score de chaque catégorie d'heuristiques (H) - exprimé en pourcentage - est ainsi déterminé en calculant la proportion d'URLs qui présentent un score négatif (pour les blacklists (BL)) ou positif (pour les whitelists (WL)), parmi l'ensemble des URLs testées :

Tableau 3.1 – Heuristiques implémentés

Catégorie	Nº	Heuristique
Points et caractères spéciaux	1	Nombre de points (.) dans l'URL
	2	Nombre d'arobas (@) dans l'URL
	3	Nombre de double slash (//) dans l'URL
	4	Présence d'une adresse IP dans le FQDN
	5	Présence d'un numéro de port dans le FQDN
Triplets et mots-clés (dits de phishing)	6	Nombre de triplets dans le FQDN
	7	Nombre de triplets dans l'arborescence de l'URL (hors FQDN)
	8	Nombre de mots-clés dans l'arborescence de l'URL (hors FQDN)
TLD	9	Présence d'un TLD "sensible" dans le FQDN
	10	Présence d'un TLD "sensible" dans l'arborescence de l'URL (hors FQDN)
	11	Comparaison entre TLD et pays hébergeur du site
Code source HTML	12	Evaluation de la balise de Titre
	13	Evaluation de balise Formulaire
	14	Evaluation de lien Image
	15	Evaluation d'autre lien
Page de Login	16	HTTPS et zones de login
Autres balises HTML	17	Evaluation de balise de Description Meta
	18	Evaluation de balise de mots-clés Meta
	19	Evaluation de balise Script
	20	Evaluation de balise Link

$$\text{Score } H(x)_{BL} = \frac{\sum_{i=1}^{n}\left(H(i)<0\right)}{n}\cdot 100 \quad \text{et} \quad \text{Score } H(x)_{WL} = \frac{\sum_{i=1}^{n}\left(H(i)>0\right)}{n}\cdot 100$$

où *(x)* représente la catégorie d'heuristique étudiée
et *(n)* le nombre d'URLs testées

Les tests réalisés ici ont porté sur une whitelist de 230 URLs (extraites des 500 URLs utilisées lors de la phase de vérification), et sur une blacklist de 230 nouvelles URLs.

3.3.5 Environnement

Pour *Phishark*, les 3 phases ont été réalisées depuis une seule et même machine ayant pour caractéristiques : double processeur Intel® Core™2, 0,99 Go. de RAM, système d'exploitation Microsoft Windows XP Professional, version 2002, Service Pack 3, et navigateur web Mozilla Firefox v.3.6.7.

Pour la phase de comparaison des performances, afin de garantir les mêmes conditions d'expérimentation à toutes les barres anti-phishing, chaque barre sélectionnée a été installée sur une machine de mêmes caractéristiques (exceptée une machine où le navigateur web est différent, afin de tester la fonctionnalité anti-phishing d'*Internet Explorer*).

Ainsi la phase de comparaison a été réalisée en simultané pour chaque barre d'outils (c.-à-d. 1 URL testée au même moment par les 5 barres).

3.4 Phase d'étalonnage : Heuristiques étudiés et seuils de détection

Basés sur l'analyse des caractéristiques des pages de phishing vue en section 2.2, et des travaux précédents [ZECH07] [ZHC07] [HYM09], nous avons défini et implémenté 20 tests heuristiques qui se décomposent en 6 catégories (cf. tableau 3.1) :

- **Points et caractères spéciaux**
- **Triplets et mots-clés (dits de phishing)**
- **TLD**
- **Code source HTML**
- **Page de login**
- **Autres balises HTML**

Ces heuristiques ont été choisis à partir :
- D'études précédentes qui ont fait apparaître les principales caractéristiques à analyser et/ou seuils de décision associés. Ces travaux - qui portent en grande majorité sur l'URL - sont cités au fur et à mesure des détails donnés ci-après à propos de chaque heuristique.
- Et d'études complémentaires approfondies que nous avons menées sur l'analyse des caractéristiques des pages légitimes/contrefaites, tant au niveau de l'URL que du contenu du code source des pages HTML.

Précisons que quel que soit le seuil de décision retenu pour chaque heuristique, il existe toujours de nombreux cas de faux-positifs et/ou faux-négatifs résiduels associés. Seul l'ensemble des heuristiques cumulés - pour arriver au score global de décision - peut limiter ces cas. En effet, plus il y a de scores heuristiques individuels négatifs, plus la probabilité d'être en présence d'un site contrefait est forte. Et inversement.

3.4.1 URL

Les sites de phishing reposent en partie sur l'utilisation d'une URL alternative. Une piste de détection évidente est donc l'analyse de cette URL.

Dans notre étude, les heuristiques qui s'y rapportent sont regroupés dans les 3 catégories suivantes : *Points et caractères spéciaux*, *Triplets et mots-clés (dits de phishing)*, et *TLD* (cf. tableau 3.1).

3.4.1.1 Catégorie Points et caractères spéciaux

3.4.1.1.1 Heuristique N° 1 : Nombre de points (.) dans l'URL L'étude de Zhang et al. [ZHC07] indique que les URLs de phishing sont généralement constituées d'un minimum de 5 points. En effet, pour leurrer les utilisateurs, les attaquants ont tendance a faire référence au FQDN légitime dans l'arborescence de l'URL. Par exemple, l'URL `http://www.neural-net.ca/store/images/webscr/1/www.paypal.co.uk/` qui usurpe le site Paypal et a pour FQDN `www.neural-net.ca`, rappelle le FQDN légitime `www.paypal.co.uk` dans son arborescence, ce qui introduit des points supplémentaires (vs. une URL "classique").

Une URL légitime contient quant à elle généralement un minimum de 2 ou 3 points (p.ex. `www.mondomaine.com` ou `www.mondomaine.com/dossier1/dossier2/fichier.html`). Il est en effet plus rare de voir des URLs de phishing uniquement constituées d'un FQDN.

Par conséquent, nous avons choisi la valeur de 3 points comme valeur charnière de la décision de légitimité d'un site (cf. tableau 3.3). Jusqu'à cette valeur, le score de l'heuristique est positif ou nul. Au-delà, il est négatif. Précisons que cet heuristique s'applique à l'intégralité de l'URL.

La phase d'étalonnage démontre que les seuils optimum sont identiques sur blacklist et whitelist (cf. tableau 3.4).

Des exemples de faux-positifs / faux-négatifs résiduels :
- L'URL légitime du site de login Hotmail `http://login.live.com/login.srf?wa=wsignin1.0&rpsnv=11&ct=1281707336&rver=6.0.5285.0&wp=MBI&wreply=http:%2F%2Fmail.live.com%2Fdefault.aspx&lc=2057&id=64855&mkt=en-gb` comporte 10 points.
- L'URL légitime du site de login SFR `https://www.sfr.fr/cas/login?service=https%3A%2F%2Fsfr-messagerie.services.sfr.fr%2Fwebmail%2Fj_spring_cas_security_check` comprend 5 points.
- L'URL contrefaite `http://gomezcomunidade.t35.com/` ne contient que 2 points.

3.4.1.1.2 Heuristiques N° 2 et 3 : Nombre d'arobas (@) et de double slash (//) dans l'URL Les caractères spéciaux tels que l'arobas (@) - situé dans le FQDN -, ou le double slash (//) - situé dans l'arborescence de l'URL - sont parfois utilisés par les attaquants, soit pour procéder à d'éventuelles redirections, soit pour leurrer l'utilisateur en faisant apparaître le FQDN légitime (cf. section 2.2).

Par conséquent, dès lors qu'un caractère de ce type est détecté lors de l'analyse, le score de l'heuristique correspondant est négatif. Sinon, il est positif. Ces deux heuristiques s'appliquent à l'intégralité de l'URL, au-delà de l'indication de protocole - c.-à-d. au delà du `http://` - pour la recherche de double

slash (cf. tableau 3.3).

La phase d'étalonnage sur blacklist et whitelist démontre que les seuils optimum sont identiques pour la présence d'arobas .

A contrario, il arrive de trouver le double slash à de multiples reprises dans les URLs de phishing (cf. tableau 3.4), souvent afin de rappeler le FQDN légitime.

On trouve également des double slash dans les URLs légitimes, typiquement pour des redirections de pages de login (parfois ces redirections sont masquées via des techniques d'encodage, p.ex. `https://` peut être remplacé par `https%3A%2F%2F` avec un encodage ASCII). Toutefois, la majorité des URLs légitimes qui font appel à ces techniques de redirection renvoient généralement vers un FQDN appartenant au même domaine. Nous avons donc pris cette notion en compte pour déterminer le seuil de décision final (cf. tableau 3.5).

Des exemples d'URLs faisant appel à ces techniques de redirection :
- L'URL légitime du site de login Orange `http://id.orange.fr/auth_user/bin/auth0user.cgi?date=1276854069&skey=0fe469df5d9227a79b29390183013939&service=communiquer&url=http://webmail1j.orange.fr/webmail/fr_FR/welcome_freeUrl.html` utilise le (//) pour rediriger vers le FQDN `webmail1j.orange.fr` appartenant au même domaine.
- L'URL légitime du site de login Wachovia `https://www.wachovia.com/stateselector?referring_page=https://www.wachovia.com/foundation/v/index.jsp?vgnextoid=06ce9e05d1674210VgnVCM200000627d6fa2RCRD&product_code=CHK` utilise le (//) pour rediriger vers le même FQDN.
- L'URL contrefaite `http://desoskiz.org/personalbankingalertnotification@hsbc.co.uk/index.php` utilise l'(@) pour passer des attributs et faire apparaître le FQDN légitime.
- L'URL contrefaite `http://www.amazon.com:fvthsgbljhfcs83infoupdate@69.10.142.34` [CG06] semblerait indiquer pour un utilisateur non-averti qu'il est dirigé vers le FQDN `www.amazon.com`. Néanmoins, l'utilisation du (@) le redirige vers le serveur web de l'attaquant, qui a pour adresse IP : `69.10.142.34`.

3.4.1.1.3 Heuristique N° 4 : Présence d'une adresse IP dans le FQDN Pour éviter qu'un FQDN différent du FQDN légitime n'attire l'attention de l'Internaute, les attaquants préfèrent parfois le remplacer par une adresse IP [CG06]. Nous constatons que ce remplacement est d'ailleurs souvent couplé à un rappel du domaine ou FQDN légitime dans l'arborescence de l'URL, afin de maximiser les chances de leurrer l'utilisateur.

A contrario, ce cas de figure se présente très rarement sur URLs légitimes (une adresse IP est parfois utilisée dans les URLs qui utilisent le protocole FTP, mais celles-ci sont rarement la cible d'attaques de phishing).

Cet heuristique s'applique à l'analyse du FQDN exclusivement. A l'examen de l'URL, dès lors qu'une adresse IP est détectée en lieu et place d'un FQDN (cf. tableau 3.3), nous attribuons un score négatif à l'heuristique correspondant. Dans le cas contraire, le score assigné est nul. Précisons toutefois que rien n'empêche un utilisateur d'accéder à un site légitime via son adresse IP. Le score négatif attribué ici se doit donc d'être limité pour pallier ce cas de figure.

La phase d'étalonnage démontre que les seuils optimum de ce critère sont identiques sur blacklist et whitelist (cf. tableau 3.4).

Des exemples d'URLs contrefaites qui utilisent une adresse IP en lieu et place d'un FQDN :
- L'URL contrefaite `http://216.139.111.51/~hsbc/hsbc/`, qui usurpe le site HSBC, prend soin de rappeler le domaine légitime dans l'arborescence de l'URL.
- L'URL contrefaite `http://221.5.225.151/www.paypal.com.it.cgi-bin.webscr.cmd.login.run/index.htm`, qui usurpe le site Paypal, prend soin de rappeler le FQDN légitime dans l'arborescence de l'URL.

3.4.1.1.4 Heuristique N° 5 : Présence d'un numéro de port dans le FQDN L'APWG [APW10] indique que les URLs de phishing utilisent parfois des ports alternatifs - en association avec le FQDN - à ceux traditionnellement utilisés par les protocoles (exemples de ports standards : 80 pour le HTTP, 443 pour

Tableau 3.2 – Liste des 240 triplets

".at"	".au"	".br"	".ch"	".cn"	".co"	".de"	".eb"	".ed"	".es"	".eu"	".go"	".il"	".in"	".iv"	".mo"	".ms"	".ne"	".nl"	".nz"
".or"	".rr"	".ru"	".tk"	".to"	".us"	".ya"	"0.n"	"10."	"1nc"	"a.g"	"a.o"	"a.u"	"adf"	"adu"	"aeo"	"aho"	"ail"	"an."	"ank"
"arc"	"art"	"asi"	"at."	"au."	"aud"	"aue"	"b.c"	"ban"	"bay"	"bes"	"bmw"	"br."	"c.u"	"cas"	"ch."	"cit"	"cn."	"co."	"com"
"cor"	"cou"	"cro"	"cs."	"d.o"	"d.u"	"dco"	"dcr"	"de."	"deb"	"dru"	"du."	"e.i"	"e.o"	"e.u"	"eat"	"ear"	"eba"	"ebt"	"eca"
"ech"	"eco"	"edu"	"ej."	"eo."	"eof"	"er."	"erc"	"ers"	"es."	"et."	"eu."	"ews"	"exf"	"ez."	"f.c"	"fan"	"fil"	"fo."	"gen"
"gir"	"gov"	"h.u"	"hoo"	"iae"	"iau"	"ics"	"ieo"	"if."	"ij."	"ik."	"il."	"ilm"	"inc"	"inf"	"ino"	"int"	"iq."	"irl"	"ity"
"iw."	"j.c"	"jou"	"l.c"	"l.o"	"l.u"	"lan"	"lms"	"loa"	"m.b"	"m.c"	"m.e"	"mai"	"mer"	"mon"	"msn"	"n.c"	"n.u"	"nal"	"nc."
"nc0"	"nco"	"net"	"new"	"nfo"	"nk."	"nl."	"nlo"	"no."	"ns."	"nst"	"nte"	"nty"	"nux"	"nz."	"o.c"	"o.n"	"o.u"	"oan"	"ob."
"of."	"ofc"	"ofm"	"ofn"	"ofs"	"ogl"	"oj."	"om."	"omm"	"on."	"ons"	"oog"	"org"	"orp"	"oun"	"ov."	"oz."	"q.c"	"r.u"	"rch"
"rg."	"rls"	"rp."	"rs."	"rth"	"ru."	"rug"	"s.c"	"s.o"	"s.u"	"sci"	"sco"	"sea"	"sec"	"sex"	"sfa"	"sin"	"s-l"	"sn."	"sp-"
"ss-"	"sta"	"ste"	"sys"	"t.u"	"tat"	"tec"	"teo"	"ter"	"tk."	"tla"	"to."	"ty."	"tyn"	"tyo"	"u.o"	"ud."	"uen"	"uh."	"uj."
"uni"	"unt"	"us."	"uv."	"uw."	"v.c"	"vwb"	"w.c"	"web"	"wes"	"wnl"	"ws."	"ww."	"www"	"y.u"	"yah"	"yof"	"you"	"z.c"	"-lo"

le HTTPS, etc.). Nous avons donc choisi de vérifier que l'URL affichée ne présente pas d'indication de port autre que le port standard.

Si on détecte l'indication d'un port alternatif pour le protocole affiché, nous attribuons un score négatif à l'heuristique correspondant. En l'absence de numéro de port, le score est nul. Cet heuristique s'applique à l'analyse du FQDN exclusivement (cf. tableau 3.3).

La phase d'étalonnage démontre que ce cas de figure n'a jamais été rencontré sur whitelist (cf. tableau 3.4).

Des exemples d'URL contrefaites qui utilisent un numéro de port non-standard : `http://raceobject.ru:8080/index.php?pid=14/` et `http://hillchart.com:8080/index.php?pid=14/` emploient le port 8080 en association avec le protocole HTTP. Ce port 8080 est souvent utilisé pour faire fonctionner un serveur web alternatif, lorsque l'utilisateur ne possède pas les droits administrateurs[1] (c.-à-d. qu'il n'a pas l'autorisation d'utiliser les ports *well-known* 0 à 1023) ou que le port 80 de la machine est déjà utilisé par un autre serveur web.

3.4.1.2 Catégorie Triplets et mots-clés (dits de phishing)

3.4.1.2.1 Heuristiques N° 6 et 7 : Nombre de triplets dans l'URL Une étude préalable, réalisée conjointement avec le FAI Orange, menée sur des whitelists et des blacklists (comprenant au cumulé jusqu'à 56 227 URLs ou composants d'URLs) nous a permis d'identifier un certain nombre de triplets de caractères souvent présents dans les URLs de phishing.

Cette étude a été réalisée à partir du logiciel *Khiops* proposé par Boullé [Bou], qui offre un outil de préparation et de modélisation des données, utilisable à des fins d'exploration de bases de données (p.ex. pour des études d'usage) [Bou08]. Il permet d'aboutir à une analyse prédictive des variables d'entrées (c.-à-d. les composants d'URLs) via une approche statistique de type Bayésienne.

Des résultats de cette étude, nous avons extrait un ensemble de 240 triplets de caractères - considérés par l'outil comme les plus présents dans les URLs de phishing -, que nous utilisons comme base de notre analyse de l'URL visitée (cf. tableau 3.2).

Parmi ces triplets, nous avons constaté que 4 d'entre eux étaient très fréquents dans tous FQDNs (y compris ceux des URLs légitimes) : `www`, `ww.`, `.co` et `com`. Il peut alors apparaître surprenant que l'outil nous les énonce comme caractéristiques d'une URL de phishing. Ceci s'explique néanmoins par le fait que ces triplets - habituellement réservés à une utilisation au sein du FQDN (par les URLs légitimes) - se retrouvent fréquemment utilisés tant dans le FQDN que dans l'arborescence des URLs de phishing (via le rappel du FQDN légitime, cf. section 2.2.1).

Nous avons donc assigné les scores des heuristiques en conséquence :

– *Nombre de triplets dans le FQDN :* la présence de 4 triplets est choisie comme valeur charnière de la décision de légitimité d'un site, afin de correspondre aux 4 triplets courants cités précédemment (cf. tableau 3.3). Jusqu'à cette valeur, le score de l'heuristique est positif ou nul. Au-delà, il est négatif.

  L'étalonnage réalisé fait apparaître que les triplets sont nombreux, particulièrement dans les FQDNs de phishing. Toutefois, il semble important de préserver les URLs légitimes en deçà de 3

1. C'est le cas typique d'un site de phishing hébergé et mis en ligne au travers d'un réseau de botnets, cf. section 2.3.

Tableau 3.3 – Valeurs charnières pour les seuils de décision des heuristiques implémentés

N°	Heuristique	Valeur charnière
1	Nombre de points (.) dans l'URL	3
2	Nombre d'arobas (@) dans l'URL	1
3	Nombre de double slash (//) dans l'URL	1 au delà des (//) énoncés après le protocole (p.ex. `http://`)
4	Présence d'une adresse IP dans le FQDN	1
5	Présence d'un numéro de port dans le FQDN	1, si non standard
6	Nombre de triplets dans le FQDN	4
7	Nombre de triplets dans l'arborescence de l'URL (hors FQDN)	1
8	Nombre de mots-clés dans l'arborescence de l'URL (hors FQDN)	1
9	Présence d'un TLD "sensible" dans le FQDN	si appartenance au Groupe 1 ou au Groupe 2
10	Présence d'un TLD "sensible" dans l'arborescence de l'URL (hors FQDN)	si appartenance au Groupe 1 ou au Groupe 2
11	Comparaison entre TLD et pays hébergeur du site	si non correspondance
12	Evaluation de la balise de Titre	si absence du domaine/FQDN
13	Evaluation de balise Formulaire	si absence du domaine/FQDN
14	Evaluation de lien Image	si absence du domaine/FQDN
15	Evaluation d'autre lien	si absence du domaine/FQDN
16	HTTPS et zones de login	si présence de champs *PASSWORD* ou *LOGIN* hors connexion HTTPS
17	Evaluation de balise de Description Meta	si absence du domaine/FQDN
18	Evaluation de balise de mots-clés Meta	si absence du domaine/FQDN
19	Evaluation de balise Script	si absence du domaine/FQDN
20	Evaluation de balise Link	si absence du domaine/FQDN

triplets (cf. tableau 3.4).
- *Nombre de triplets dans l'arborescence de l'URL (hors FQDN) :* au-delà d'1 triplet (cf. tableau 3.3), le score assigné est négatif. Sinon, le score attribué est positif.
  L'étalonnage démontre qu'un nombre conséquent d'URLs de phishing utilise les triplets à de multiples reprises dans l'arborescence de l'URL (cf. tableau 3.4).

Des exemples d'occurrences de triplets :
- L'URL légitime `http://bluewin.ch/` utilise 2 triplets dans son FQDN et 0 triplet dans le reste de l'URL.
- L'URL légitime `http://www.cashfiesta.com/php/login.php` utilise 7 triplets dans son FQDN et 0 triplet dans le reste de l'URL.
- L'URL contrefaite `http://www.agb.com.ve/maps/160703/survey/login.htm?paypal.com/us/cgi-bin/webscr?cmd=_login-submit&dispatch=5885d80a13c0db1f1ff80d546411d7f8a8350c132bc41e0934cfc023d4e8f9e5` utilise 6 triplets dans son FQDN et 4 triplets dans le reste de l'URL.
- L'URL contrefaite `http://www.jacuzzifilms.com/plugins/editors-xtd/modulesanywhere/css/wellsfargo-online.php` utilise 8 triplets dans son FQDN et 1 triplet dans le reste de l'URL.

3.4.1.2.2 Heuristique N° 8 : Nombre de mots-clés dans l'arborescence de l'URL (hors FQDN) Après une étude approfondie des URLs de phishing, à l'image des travaux menés par Garera et al. [GPCR07], nous avons sélectionné 5 mot-clés (c.-à-d. `www`, `http`, `login`, `logon`, `paypal`) qui apparaissent régulièrement dans les URLs de phishing. Le choix des 4 premiers mots-clés s'explique par le fait que les URLs de phishing tendent à usurper des pages de login et/ou mentionner le FQDN légitime dans l'arborescence de l'URL pour leurrer les Internautes. Le choix du 5ème mot-clé se justifie quant à lui par le fait que Paypal apparaît comme une cible privilégiée des attaquants.

Toutefois, nous notons que ces mots-clés se retrouvent également fréquemment dans les FQDNs des

URLs légitimes. Nous choisissons donc de limiter l'analyse de cet heuristique à l'arborescence de l'URL (hors FQDN).

Ainsi, dès l'apparition d'un mot-clé (cf. tableau 3.3), le score attribué est négatif. En leur absence, le score attribué est positif.

La phase d'étalonnage laisse apparaître qu'il est assez fréquent de trouver l'un des mots-clés (typiquement `login` ou `logon`) dans le nom de fichier ou les données complémentaires précisées dans les URLs de login légitimes (cf. tableau 3.4).

Des exemples d'occurrences de mots-clés dans les URLs :
- L'URL légitime `https://cas.it-sudparis.eu/cas/login?service=http://gaspar.it-sudparis.eu/uPortal/Login` utilise 2 fois le mot-clé `login` dans son arborescence.
- L'URL légitime `https://login.yahoo.com/config/mail?&.src=ym&.intl=fr` utilise le mot-clé `login` dans son FQDN uniquement.
- L'URL contrefaite `http://www.neural-net.ca/store/images/webscr/1/www.paypal.co.uk/` utilise les mots-clés `www` et `paypal` dans son arborescence.

3.4.1.3 Catégorie TLD

3.4.1.3.1 Heuristiques N° 9 et 10 : Présence d'un TLD "sensible" dans l'URL Parmi les triplets retournés par notre analyse (cf. section 3.4.1.2), nous avons remarqué que certains d'entre eux correspondent à des TLDs. Par ailleurs, chaque trimestre, l'APWG publie un Top 10 des pays hébergeurs des sites de phishing (en indiquant le proportion de sites de phishing qu'ils hébergent).

Nous avons donc décider d'utiliser cette information, pour pénaliser les URLs dont le TLD est issu d'un pays connu pour sa proportion à héberger les sites de phishing. Pour ce faire, nous avons créé 2 groupes de TLDs : le Groupe 1 contient les TLDs des pays qui hébergent en moyenne plus de 40% de sites de phishing, tandis que le Groupe 2 inclut le reste des TLDs connus pour l'hébergement de sites contrefaits. Plus précisément, nous avons recensé l'ensemble des TLDs signalés par l'APWG sur une période de 12 mois. Pour chacun d'entre eux, nous avons calculé la moyenne d'hébergement sur la période étudiée. Ainsi, nous pouvons mieux faire face à l'éventuelle mouvance des pays hébergeurs.

Nous attribuons alors un score négatif aux URLs qui utilisent un TLD des Groupes 1 et 2 (ce score est doublé pour les URLs dont le TLD appartient au Groupe 1), tandis que le score assigné est nul pour les URLs en provenance d'autres TLDs (cf. tableau 3.3). Cet heuristique s'applique à l'intégralité de l'URL, afin de tenir compte d'éventuelles redirections de sites au sein des URLs de phishing.

La phase d'étalonnage démontre que les seuils optimum sont identiques sur blacklist et whitelist (cf. tableau 3.4).

A ce jour, les pays hébergeurs de sites de phishing pénalisés par notre moteur de détection sont :
- Groupe 1 : US et UM (tous deux correspondent aux États-Unis).
- Groupe 2 : SE, CN, CA, UK, GB, DE, KP, FR, PM, RE, TF, WF, GT, RU, SU, AN, NL, TW, RO, PL, ES, HU, HK et BR.

3.4.1.3.2 Heuristique N° 11 : Comparaison entre TLD et pays hébergeur du site A partir de l'étude de McGrath et al. [MG08] qui indique que la plupart des sites de phishing ne sont pas hébergés dans le pays annoncé par leur TLD, nous avons cherché à évaluer la corrélation entre TLD annoncé par l'URL et pays hébergeur du site.

Pour ce faire, nous nous sommes appuyés sur un plug-in additionnel existant pour Mozilla Firefox : *World IP* [wor]. Celui-ci permet de donner diverses informations sur le site web visité (p.ex. adresse IP, pays hébergeur, etc. - cf. figure 3.2 - grâce à des requêtes reverse DNS, ping, traceroute, etc.)[1].

Néanmoins, parce que de nombreux sites légitimes ne sont pas non plus hébergés dans le pays indiqué par le TLD - typiquement dans le cas de multinationales et/ou de TLD génériques -, nous avons choisi d'utiliser ce critère de manière favorable exclusivement (cf. tableau 3.3).

1. D'autres plug-ins de même type sont disponibles pour Mozilla Firefox, mais *World IP* est celui qui nous est apparu le plus complet/moins restrictif en terme d'informations délivrées.

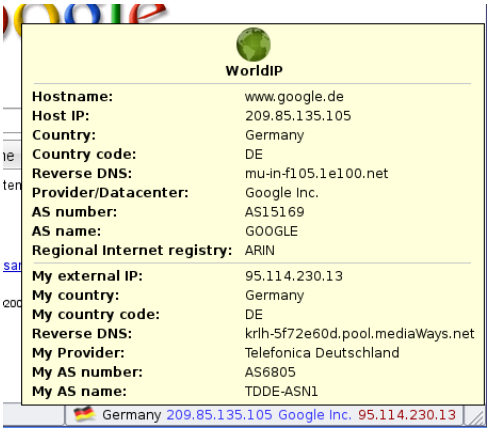


FIGURE 3.2 – Aperçu des informations délivrées par World IP [wor]

Ainsi, si le TLD du FQDN contenu dans l'URL correspond au code pays (champ *Country-Code*) annoncé par *World IP*, le score assigné est positif. Sinon, il est nul (cf. tableau 3.4).

Des exemples d'URLs et pays hébergeurs associés :
- Les sites légitimes `https://particuliers.societegenerale.fr/` et `http://www.013netvision.net.il/` sont hébergés dans le pays indiqué par leur TLD.
- Les URLs légitimes `http://www.dell.fr` et contrefaites `http://estiloforrozeiro.com.br/paypal.co.uk/_cmd-check.php?cmd_check=logged&SESSION-ID=8859-1&uid=F654557d45Af4t` et `http://secure.paypal.fr.freezonesuae.com/login/`, sont toutes trois hébergées aux États-Unis, bien que cela ne transparaisse pas au niveau de leurs TLDs.

3.4.2 Code source HTML

Au-delà de l'étude de l'URL, il peut être intéressant d'examiner le contenu de la page web (autrement dit, le code source HTML de la page) pour détecter les sites de phishing.

En effet, notre étude approfondie des contenus des pages webs laisse apparaître que les sites de phishing présentent un certain nombre d'incohérences, entre code source de la page et URL/FQDN (cf. section 2.2.2). Pour cette raison, nous nous intéressons ici à définir/utiliser des heuristiques qui visent à comparer ces informations, via l'étude de la cohérence des contenus de balises HTML vs. le FQDN/domaine visité.

Ces heuristiques sont regroupés en 3 catégories : *Code source HTML*, *Page de Login* et *Autres balises HTML* (cf. tableau 3.1).

3.4.2.1 Catégorie Code source HTML

3.4.2.1.1 Heuristique N° 12 : Evaluation de la balise de Titre

La balise <TITLE> présente dans le document HTML est généralement utilisée pour spécifier le titre du document.

Dans les URLs légitimes, on y retrouve principalement une information en lien avec le domaine/FQDN visité. Des exemples de contenu de balises <TITLE> de sites légitimes :
- Dans la page de login `http://www.facebook.com/`, on trouve le titre `Bienvenue sur Facebook. Connectez-vous, inscrivez-vous ou découvrez !` qui contient le nom de domaine.
- Dans la page d'accueil `http://www.google.fr/`, on trouve le titre `Google` qui est le nom de domaine et du service.
- Dans la page de login Yahoo ! `https://login.yahoo.com/config/mail?&.src=ym&.intl=fr`, on trouve le titre `Ouverture de session Yahoo! - Création de compte` qui contient le nom de domaine.

TABLEAU 3.4 – Seuils de décision optimum des heuristiques pour la Whitelist et la Blacklist

N°	Heuristique	Seuils optimum pour la Whitelist	Seuils optimum pour la Blacklist
1	Nombre de points (.) dans l'URL	-2 si quantité (.) > 10 -1 si quantité (.) > 5 0 si quantité (.) ≤ 3 1 si quantité (.) = 1	-2 si quantité (.) > 10 -1 si quantité (.) > 5 0 si quantité (.) ≤ 3 1 si quantité (.) = 1
2	Nombre d'arobas (@) dans l'URL	-1 si quantité (@) ≥ 1 1 si quantité (@) = 0	-1 si quantité (@) ≥ 1 1 si quantité (@) = 0
3	Nombre de double slash (//) dans l'URL	-1 si quantité (//) ≥ 1 1 si quantité (//) = 0	-2 si quantité (//) ≥ 2 -1 si quantité (//) ≥ 1 1 si quantité (//) = 0
4	Présence d'une adresse IP dans le FQDN	-2 si adresse IP 0 si pas adresse IP	-2 si adresse IP 0 si pas adresse IP
5	Présence d'un numéro de port dans le FQDN	*Aucune URL concernée*	-1 si non standard 0 si pas de numéro de port
6	Nombre de triplets dans le FQDN	-2 si quantité triplets > 10 -1 si quantité triplets ≤ 10 0 si quantité triplets ≤ 4, ou pas de FQDN[1] 1 si quantité triplets ≤ 2 2 si quantité triplets = 0	-2 si quantité triplets > 10 -1 si quantité triplets ≤ 10 0 si quantité triplets ≤ 4, ou pas de FQDN[1] 1 si quantité triplets = 0
7	Nombre de triplets dans l'arborescence de l'URL (hors FQDN)	-1 si quantité triplets ≥ 1 1 si quantité triplets = 0	-2 si quantité triplets > 4 -1 si quantité triplets ≥ 2 0 si quantité triplets = 1 1 si quantité triplets = 0
8	Nombre de mots-clés dans l'arborescence de l'URL (hors FQDN)	-3 si quantité mots-clés > 4 -2 si quantité mots-clés > 2 -1 si quantité mots-clés = 2 0 si quantité mots-clés = 1 1 si quantité mots-clés = 0	-3 si quantité mots-clés > 4 -2 si quantité mots-clés ≥ 2 -1 si quantité mots-clés = 1 1 si quantité mots-clés = 0
9	Présence d'un TLD "sensible" dans le FQDN	-2 si TLD ∈ Groupe 1[2]	-2 si TLD ∈ Groupe 1[2]
10	Présence d'un TLD "sensible" dans l'arborescence de l'URL (hors FQDN)	-1 si TLD ∈ Groupe 2[2] 0 si TLD ∈ aucun Groupe	-1 si TLD ∈ Groupe 2[2] 0 si TLD ∈ aucun Groupe
11	Comparaison entre TLD et pays hébergeur du site	0 si pas correspondance 1 si correspondance	*Pas nécessaire*[3]
12	Evaluation de la balise de Titre	-1 si ∅ ou pas correspondance avec FQDN/domaine 2 si correspondance avec FQDN/domaine	-2 si pas correspondance avec FQDN/domaine -1 si ∅ 1 si correspondance avec FQDN/domaine
13	Evaluation de balise Formulaire		
14	Evaluation de lien Image		
15	Evaluation d'autre lien		
16	HTTPS et zones de login	-1 si zone de login hors HTTPS 0 si pas de zone de login 3 si zone de login en HTTPS	-3 si zone de login hors HTTPS 0 si pas de zone de login 2 si zone de login en HTTPS
17	Evaluation de balise de Description Meta	0 si ∅ ou pas de correspondance avec FQDN/domaine 1 si correspondance avec FQDN/domaine	-1 si ∅ ou pas de correspondance avec FQDN/domaine 1 si correspondance avec FQDN/domaine
18	Evaluation de balise de mots-clés Meta		
19	Evaluation de balise Script		
20	Evaluation de balise Link		

[1] c.-à-d. remplacé par une adresse IP.
[2] cf. section 3.4.1.
[3] car pas de pénalité négative possible, cf. section 3.4.1.

A contrario dans les URLs de phishing, on trouve rarement cette corrélation. En effet de par les techniques d'aspiration de sites webs utilisées par les attaquants pour créer leur contrefaçon et/ou leur volonté de minimiser les modifications apportées, on retrouve très souvent le contenu de la balise <TITLE> du site légitime dans la page web contrefaite. Des exemples de contenus de balises <TITLE> de sites contrefaits :

- Dans la page contrefaite de login `http://61.146.118.23:8088/a/`, on trouve le titre `Welcome to eBay - Sign in - Error` qui contient le nom de domaine usurpé `eBay`.
- Dans la page contrefaite de login `http://buyacaravan.net/webscr.php`, on trouve le titre `Send Money, Pay Online, and Receive Money - all with PayPal` qui contient le nom de domaine usurpé `Paypal`.
- Dans la page contrefaite de login `http://service.eshost.es/index3.htm`, on retrouve le titre `¡Bienvenido a Facebook en Español!` qui contient le nom de domaine usurpé `Facebook`.

Parce qu'une URL de phishing - bien qu'elle soit parfois très ressemblante - n'utilise pas (ou qu'exceptionnellement) le domaine légitime dans son FQDN, nous avons considéré qu'il pouvait être intéressant de vérifier la cohérence de ces informations (c.-à-d. FQDN/domaine visité et contenu de la balise <TITLE>).

Ainsi, s'il y a correspondance, le score assigné est positif. A contrario, s'il n'y a pas correspondance ou si la balise est vide, le score attribué est négatif (cf. tableau 3.3).

Notre phase d'étalonnage démontre que l'étude de cet heuristique est particulièrement déterminante pour les sites de phishing existants actuellement. Le score positif profite quasi-exclusivement aux sites légitimes. Une balise vide est souvent (mais pas forcément) révélatrice d'une contrefaçon. Enfin, il est encore plus fréquent qu'une incohérence entre FQDN et <TITLE> soit révélatrice d'une contrefaçon (cf. tableau 3.4).

3.4.2.1.2 Heuristiques N° 13, 14 et 15 : Evaluation de balises de formulaire (FORM) et de liens (IMG et A HREF) Une contrefaçon de site web se doit de conserver un maximum de contenu du site légitime, y compris les liens accessibles depuis la page usurpée. En effet, si un utilisateur souhaite consulter une information d'aide (avant de s'authentifier par exemple), et qu'il se trouve face à une page d'erreur ou une page approximative, son attention risque d'être éveillée. Or la qualité première d'une contrefaçon est de s'approprier le décor original au maximum. Quelle meilleure technique que de conserver toutes les redirections d'origine (autres que celles de login par exemple) pour leurrer l'utilisateur ? Non seulement, cela permet de rendre l'attaque plus transparente (visuellement parlant dans l'affichage rendu par le navigateur) mais en plus, cela minimise les efforts de l'attaquant qui peut se contenter de contrefaire la première page visitée par l'Internaute.

Le revers de la médaille, c'est que ce type de technique n'est pas du tout transparente dans le code source HTML. En effet, quasiment tous les liens spécifiés font référence au FQDN/domaine légitime et non au FQDN visité, comme Chen et al. [CG06] l'ont déjà remarqué.

De cet état de fait, nous avons choisi de déduire des tests heuristiques. Ainsi, nous nous intéressons aux balises les plus révélatrices de ces incohérences et/ou les plus soumises à contrefaçon : <IMG>, <A HREF> et <FORM>. En effet, toutes 3 contiennent généralement des URLs et/ou des paramètres additionnels qui utilisent le domaine légitime (cf. section 2.2.2 pour plus de détails sur ces balises).

Pour ces trois heuristiques, si le domaine visité apparaît dans le contenu de la balise, le score assigné est positif. A contrario, s'il n'y a pas correspondance ou si la balise est vide, le score attribué est négatif (cf. tableau 3.3).

Précisons que dans la version actuelle de *Phishark*, notre analyse se cantonne à l'étude de la première balise de chaque type trouvée. Dans une version ultérieure nous envisageons d'améliorer la robustesse de notre solution en étendant cet heuristique à une analyse de l'ensemble des balises de chaque type (p.ex. chaque score d'heuristique serait alors déterminé par l'intermédiaire d'une moyenne).

Notre phase d'étalonnage démontre que le score positif profite quasi-exclusivement aux sites légitimes, qu'une balise vide est souvent (mais non forcément) révélatrice d'une contrefaçon, et qu'il est encore plus fréquent qu'une incohérence entre FQDN et une balise soit révélatrice d'une contrefaçon (cf. tableau 3.4).

Des exemples des contenus de ces 3 balises :
- Le site légitime de météo italien Yahoo !`http://it.meteo.yahoo.com/Europa/italia/` présente les contenus suivants :
  - La balise <FORM> contient `action="http://it.search.yahoo.com/search"`,
  - La balise <IMG> contient `src="http://row.bc.yahoo.com/b?P=0WwM8Ff4erSlvqiQTchZWhi"`
  - La balise <A HREF> contient `"http://it.notizie.yahoo.com/"`

  Dans chacune des trois balises, nous retrouvons le domaine légitime.
- Le site contrefait `http://permitds.com/system/` qui usurpe le site web légitime USAA `https://www.usaa.com/`...présente les contenus suivants :
  - La balise <FORM> contient `action="done4.php"method="post"onSubmit="returnvg(this,'id,pw,q1,a1,q2,a2,q3,a3,fn,cc,sc,pin,em','Pleasenoteallfieldsarerequired.');"`,
  - La balise <IMG> contient `src="usaa_log.png"width="53"height="55"`
  - La balise <A HREF> contient `"https://www.usaa.com/inet/ent_home/CpHome"class="logo""`

  Aucune balise ne désigne explicitement le domaine visité. A contrario, deux d'entre elles font référence au domaine légitime.

3.4.2.2 Catégorie Page de Login

3.4.2.2.1 Heuristique N° 16 : HTTPS et zones de login Les sites de phishing visent à usurper des sites de login légitimes. Ces derniers sont accédés - en grande majorité - au travers d'une URL qui utilise le protocole HTTPS afin d'établir une connexion sécurisée. Précisons toutefois que des exceptions existent : on peut notamment citer le cas de `Facebook` qui sécurise l'envoi des données de façon totalement transparente pour l'utilisateur, ou de sites d'abonnement divers et variés (p.ex. des abonnements à des journaux, etc.). Néanmoins, à ce jour, nous n'avons pas rencontré de site de login qui donne un accès direct à des sources financières (carte ou compte bancaire) sans un affichage explicite en HTTPS.

A contrario les contrefaçons des sites légitimes sont en grande majorité accédées via le protocole HTTP. Il est en effet plus difficile et/ou moins souhaitable pour l'attaquant de disposer d'un certificat valide pour son site (cf. section 2.2.1). Néanmoins, le manque de vigilance des utilisateurs combinée à l'utilisation de subterfuges (tels que l'affichage du cadenas habituellement affiché par une connexion HTTPS, l'insertion de logos de sécurité, etc.) préservent l'efficacité de leurs attaques (cf. section 2.2.2).

De ce constat, nous avons imaginé qu'il serait intéressant de développer un heuristique qui s'attache à vérifier qu'une page web qui présente une zone de login, le fasse sous couvert de la mention explicite d'une connexion sécurisée en HTTPS. Tout en ayant conscience que de nombreux sites de login (moins sensibles que les sites d'accès direct aux finances) risquent de générer des faux-positifs.

Pour ce faire, nous parcourons le code source HTML de la page, à la recherche d'une zone d'authentification mentionnée par les champs *TYPE=PASSWORD* ou *TYPE=LOGIN*, typiquement trouvés au sein des balises <FORM> et <INPUT> pour désigner des zones de saisie utilisateur. Puis, nous examinons l'URL afin de voir si elle utilise le protocole HTTPS (cf. tableau 3.3). Si c'est le cas, le score assigné est positif. Sinon, il est négatif. Enfin, en l'absence de champs *TYPE=PASSWORD* ou *TYPE=LOGIN*, le score est nul.

Notre phase d'étalonnage démontre qu'il faut fortement privilégier les sites qui utilisent une connexion HTTPS. A contrario, de par le nombre de sites légitimes qui présentent des zones de login sous un affichage HTTP, il apparaît plus problématique de pondérer aussi fortement l'absence de HTTPS (cf. tableau 3.4).

Des exemples de sites légitimes qui utilisent - ou non - une connexion HTTPS :
- Mention explicite d'une connexion HTTPS : les pages de login des sites bancaires `https://www.paypal.com/`, `https://particuliers.societegenerale.fr/`, `https://www4.bankofamerica.com/hub/index.action?template=signin`, des sites marchands `https://www.amazon.fr/gp/css/ho mepage.html?ie=UTF8&ref_=topnav_ya`, etc.
- Zone d'authentification, hors HTTPS : les pages d'accueil des réseaux sociaux `http://www.facebook.com/`, `http://www.linkedin.com/`, des sites d'informations `http://www.lemonde.fr/`, des sites marchands `http://www.pixmania.com/fr/fr/c_action/mon_compte/index.html`, etc.

3.4.2.3 Catégorie Autres balises HTML

Nous avons constaté que les analyses des balises <IMG>, <FORM> et <A HREF> comparées au FQDN/domaine visité sont particulièrement efficaces pour identifier les sites de phishing. Nous avons donc souhaité étendre ces heuristiques à d'autres balises. Nous nous sommes alors intéressés aux balises <META DESCRIPTION>, <META KEYWORDS>, <SCRIPT> et <LINK>.

Précisons que les balises <META> sont nombreuses (description, keywords, author, date, etc.) mais que leur utilisation n'est pas obligatoire. Comme tous champs optionnels, il apparaît hasardeux de les utiliser en tant que critères de décision. Néanmoins, nous en avons retenu deux : la balise <META DESCRIPTION> qui est présente dans la quasi-totalité des sites que nous avons étudiés, et la balise <META KEYWORDS> que nous trouvons particulièrement intéressante puisqu'elle est principalement utilisée par les moteurs de référencement (un attaquant aura donc peu d'intérêt à la modifier, mieux vaudrait encore la supprimer que d'y mettre des informations erronées).

3.4.2.3.1 Heuristique N° 17 : Evaluation de balise de Description META La balise <META DESCRIPTION> permet de donner une description du site. Dans les pages légitimes, on y retrouve donc souvent la mention du domaine visité. Dans les pages de phishing, le contenu de cette balise est souvent inchangé par rapport au site original (c.-à-d. il contient le domaine légitime), ou la balise est tout simplement supprimée.

De par sa présence quasi-permanente dans les sites légitimes, nous lui accordons un traitement particulier (vs. les balises des heuristiques N° 18, 19 et 20 détaillés ci-après). Ainsi, si nous retrouvons le domaine visité dans la balise, le score attribué est positif. Si la balise est inexistante, le score est nul. Enfin, si la balise est incohérente avec le domaine visité, le score assigné est négatif (cf. tableau 3.3).

Notre phase d'étalonnage démontre qu'il est également possible de trouver des incohérences sur sites légitimes. Néanmoins, ces cas sont surtout l'apanage des sites contrefaits, de même que l'inexistence de la balise (cf. tableau 3.4).

Des exemples de balises <META DESCRIPTION> :

- La page d'accueil légitime Facebook `http://www.facebook.com/` contient la description `"Facebook est un réseau social qui vous relie à des amis, des collègues de travail, des camarades de classe..."`. On y retrouve aisément la mention du domaine.
- La page de login légitime de la banque BNP PARIBAS `https://www.secure.bnpparibas.net/banque/portail/particulier/HomeConnexion?`...contient la description `"Tous les produits et services de votre banque en France Accéder à vos comptes, titres et Bourse."`. On n'y retrouve aucune mention du domaine.
- La page contrefaite `http://permitds.com/system/` qui usurpe le site web légitime USAA `https://www.usaa.com/`...ne comporte pas de description.
- La page contrefaite `http://www.lospws2.co/ca/` qui usurpe la page web légitime Paypal `https://www.paypal.com/us/cgi-bin/webscr?cmd=_home&locale.x=en_US`...contient la description `"PayPal lets you send money to anyone with email. PayPal is free for consumers"`.... On y retrouve mention du domaine légitime et non du domaine visité.

3.4.2.3.2 Heuristiques N° 18, 19 et 20 : Evaluation de balises de mots-clés META, SCRIPT et LINK Les balises <META KEYWORDS>, <SCRIPT> et <LINK> (cf. section 2.2.2 pour plus de détails) sont d'usage moins courant que les balises étudiées dans les heuristiques vus précédemment. De plus, elles ne contiennent pas forcément une référence au domaine en se limitant parfois à une arborescence ou un nom de fichier appelé.

Il apparaît donc difficile de pénaliser d'éventuelles incohérences entre domaine visité et contenu de la balise. Nous réservons donc ces heuristiques à un usage bénéfique exclusivement, ceci afin de privilégier les sites légitimes.

Ainsi, si nous retrouvons le domaine visité dans la balise, le score attribué est positif. Si la balise est inexistante ou incohérente avec le domaine, le score est nul (cf. tableau 3.3).

Notre phase d'étalonnage démontre qu'il est important de ne pas pénaliser les incohérences pour les sites légitimes, contrairement au cas des sites contrefaits (cf. tableau 3.4).

TABLEAU 3.5 – Seuils de décision retenus après fusion des seuils optimum sur Whitelist et Blacklist

N°	Heuristique	Seuils de décision retenus
1	Nombre de points (.) dans l'URL	-2 si quantité (.) > 10 -1 si quantité (.) > 5 0 si quantité (.) ≤ 3 1 si quantité (.) = 1
2	Nombre d'arobas (@) dans l'URL	-1 si quantité (@) ≥ 1 1 si quantité (@) = 0
3	Nombre de double slash (//) dans l'URL	-1 si quantité (//) ≥ 1 et redirection vers un domaine différent 1 si quantité (//) ≥ 1 et redirection vers le même domaine 1 si quantité (//) = 0
4	Présence d'une adresse IP dans le FQDN	-2 si adresse IP 0 si pas adresse IP
5	Présence d'un numéro de port dans le FQDN	-1 si non standard 0 si pas de numéro de port
6	Nombre de triplets dans le FQDN	-2 si quantité triplets > 10 -1 si quantité triplets ≤ 10 0 si quantité triplets ≤ 4, ou pas de FQDN[1] 1 si quantité triplets ≤ 2 2 si quantité triplets = 0
7	Nombre de triplets dans l'arborescence de l'URL (hors FQDN)	-2 si quantité triplets > 4 -1 si quantité triplets ≥ 1 1 si quantité triplets = 0
8	Nombre de mots-clés dans l'arborescence de l'URL (hors FQDN)	-3 si quantité mots-clés > 4 -2 si quantité mots-clés ≥ 2 -1 si quantité mots-clés = 1 1 si quantité mots-clés = 0
9	Présence d'un TLD "sensible" dans le FQDN	-2 si TLD ∈ Groupe 1[2]
10	Présence d'un TLD "sensible" dans l'arborescence de l'URL (hors FQDN)	-1 si TLD ∈ Groupe 2[2] 0 si TLD ∈ aucun Groupe
11	Comparaison entre TLD et pays hébergeur du site	0 si pas correspondance 1 si correspondance
12	Evaluation de la balise de Titre	-2 si pas correspondance avec FQDN/domaine -1 si ∅ 2 si correspondance avec FQDN/domaine
13	Evaluation de balise de Formulaire	-1 si ∅ ou pas correspondance avec FQDN/domaine
14	Evaluation de lien Image	1 si correspondance avec FQDN/domaine
15	Evaluation d'autre lien	
16	HTTPS et zones de login	-2 si zone de login hors HTTPS 0 si pas de zone de login 3 si zone de login en HTTPS
17	Evaluation de balise de description Meta	-1 si pas de correspondance avec FQDN/domaine 0 si ∅ 1 si correspondance avec FQDN/domaine
18	Evaluation de balise de mots-clés Meta	0 si ∅ ou pas de correspondance avec FQDN/domaine
19	Evaluation de balise Script	1 si correspondance avec FQDN/domaine
20	Evaluation de balise Link	

[1] c.-à-d. remplacé par une adresse IP.
[2] cf. section 3.4.1.

Des exemples de contenus de ces 3 balises :

- La page de login légitime de la banque BNP PARIBAS `https://www.secure.bnpparibas.net/banque/portail/particulier/HomeConnexion?`...contient `"BNPPARIBAS.NET, connexion, comptes,titres,bourse"` pour les <META KEYWORDS>. Les balises <SCRIPT> et <LINK> se limitent quant à elles à des arborescences et noms de fichiers (p.ex. `src="/banque/PA_CanalnetApp/scripts/canalnetForms.js"`).
- La page contrefaite `http://www.lospws2.co/ca/` qui usurpe la page web légitime Paypal `https://www.paypal.com/us/cgi-bin/webscr?cmd=_home&locale.x=en_US`...ne contient pas <META-KEYWORDS>. Les balises <SCRIPT> et <LINK> se limitent à des arborescences et noms de fichiers.
- La page légitime MSN `http://www.msn.com/?st=1` contient le lien `src="http://col.stj.s-msn.com/br/sc/js/jquery/jquery-1.4.2.min.js"` où figure le nom de domaine.

Pour conclure notre description des heuristiques implémentés, le tableau 3.5 expose les seuils décisionnels actuellement implémentés par *Phishark*. Ils ont été choisis pour les raisons exposées précédemment à partir des seuils optimum relevés sur blacklist et whitelist.

3.4.3 Description de *Phishark*

Le moteur de détection de notre barre d'outils a été développé en Javascript. L'interface graphique a quant à elle été développée en langage XUL (pour *Xml-based User interface Language*) - basé sur le XML (pour *eXtensible Markup Language*) -. Ainsi, chaque action/événement de l'interface visuelle fait appel à une fonction Javascript. Le choix de ces langages orientés client réside essentiellement dans leur légèreté, simplicité, compatibilité et l'absence de besoin de compilation. L'interface graphique de la barre d'outils (p.ex. les indications de localisation des images, sa taille, etc.) est définie et structurée par une feuille de style CSS (pour *Cascading Style Sheet*).

Basés sur des travaux précédents [ECH08] [WMG06] menés sur diverses barres d'outils anti-phishing et notre analyse de leurs différentes interfaces visuelles, nous avons choisi de développer une barre d'outils relativement simple, visuellement aussi explicite que possible (via des jeux de couleurs ou de changements d'icônes), principalement axée autour de messages de notifications actifs (p.ex. via des pop-up). En effet, ces travaux précédents laissent apparaître que les notifications actives sont plus efficaces auprès des utilisateurs, car elles nécessitent obligatoirement une action de l'Internaute en cas de site suspect.

L'intégration générale de la barre *Phishark* au sein du navigateur est illustrée en figure 3.3. Par défaut, la barre d'outils se positionne tel qu'indiqué sur la figure. On remarque également une zone d'informations associée à l'utilisation de *World IP*[1] en bas à droite du navigateur. En amenant la souris sur une des adresses IP (représentatives du côté client ou du côté serveur web) affichée par *World IP*, une vue plus détaillée des informations délivrées par ce plug-in est dévoilée.

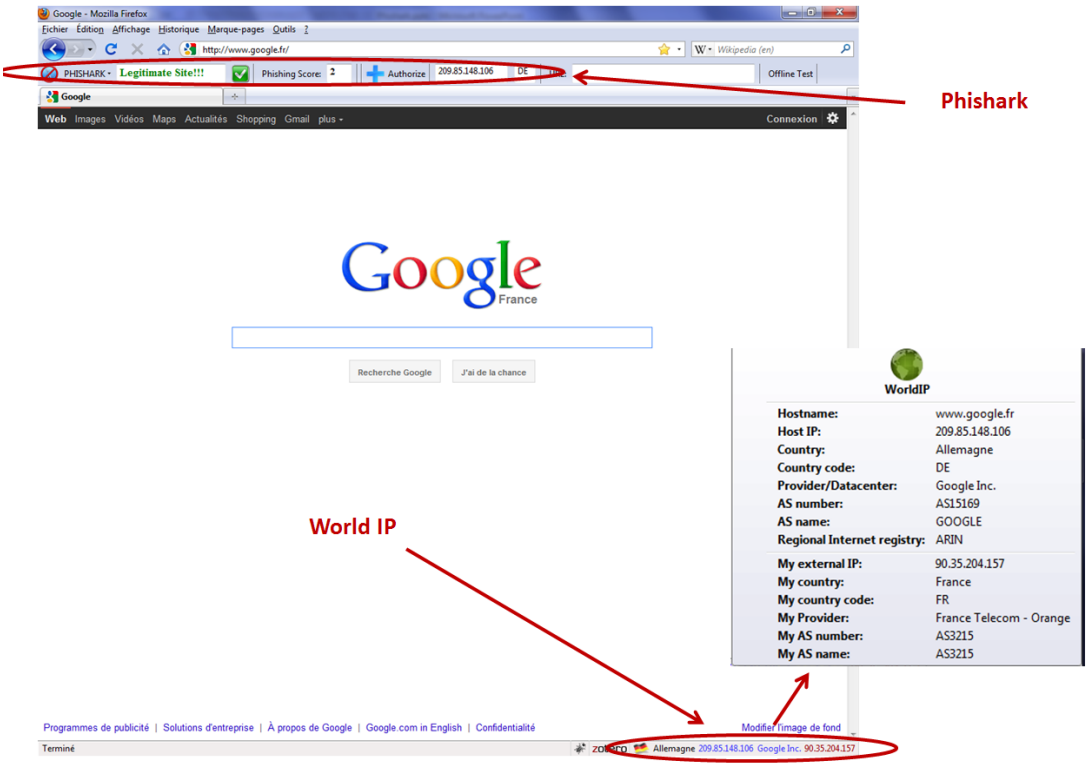


FIGURE 3.3 – Schéma d'intégration de Phishark dans le navigateur web Mozilla Firefox, incluant le plug-in World IP [wor]

La figure 3.4 zoome sur *Phishark*. On y voit que la barre est constituée de 5 zones :

– la zone 1 donne accès au menu de configuration de la barre pour : définir/modifier une white-list personnelle, redéfinir le seuil de notification *Risqué* (par défaut établi à 0, mais une plage

1. Pour plus d'informations, cf. section 3.4.1.3.2.

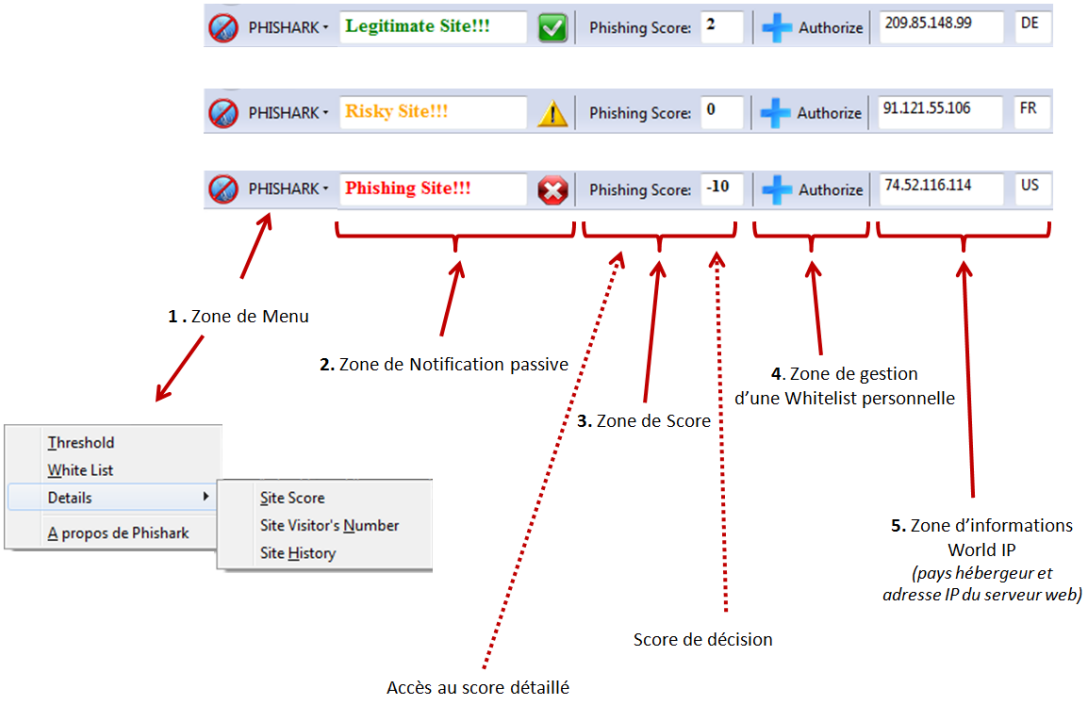


FIGURE 3.4 – Aperçu général de *Phishark*

min./max. peut être définie), obtenir un affichage plus détaillé du score obtenu pour l'URL visitée (cf. figure 3.5), obtenir des informations additionnelles concernant le site visité (p.ex. sa date de mise en ligne) ou encore connaître les informations d'auteur de la barre.
- la zone 2 contient la notification passive de décision de la barre. On aperçoit notamment les changements d'icônes/couleurs selon les 3 décisions possibles : *Légitime*, *Risqué* et *Contrefait*.
- la zone 3 délivre le score de décision. En cliquant sur le bouton *Phishing Score*, une vue plus détaillée est donnée (c.-à-d. on voit apparaître les scores de chaque heuristique/catégorie d'heuristique, tel qu'exposé en figure 3.5). Il s'agit ici d'accéder par un moyen plus rapide aux informations normalement obtenues via le menu de configuration de la barre.
- la zone 4 est un moyen rapide d'ajouter le site visité à la whitelist personnelle de l'Internaute (autrement accessible via le menu de configuration de la barre).
- enfin, la zone 5 indique l'adresse IP du serveur web et le pays hébergeur du site, tels que délivrés par *World IP*.

Enfin, la figure 3.6 montre les messages de notification active délivrés par *Phishark* en cas de sites suspects. Si le site est évalué *Risqué*, un message de type pop-up apparaît pour avertir l'utilisateur. Ce dernier doit alors cliquer sur le bouton *OK* pour continuer sa navigation. Si le site est évalué *Contrefait*, un message pop-up est également affiché pour proposer deux actions à l'utilisateur : soit continuer sa navigation (en cliquant sur *OK*), soit l'arrêter (en cliquant sur *Cancel*). Dans ce dernier cas, un message[1] exposant les dangers du phishing est alors affiché. L'utilisateur peut à nouveau choisir d'ignorer l'avertissement pour continuer sa navigation (bouton *Ignore*), avoir accès à une définition plus détaillée du phishing (bouton *More Information* qui redirige par exemple sur la page `Wikipédia` associée au phishing), ou retourner à sa page d'accueil (bouton *Exit*).

3.5 Phase de vérification et de comparaison aux autres barres d'outils

Après la phase d'étalonnage, nous avons cherché à vérifier l'efficacité de notre moteur de détection. Pour ce faire nous avons testé ses performances sur une whitelist de 500 nouvelles URLs et une blacklist

1. à l'image de celui proposé dans la fonctionnalité anti-phishing de Mozilla Firefox.

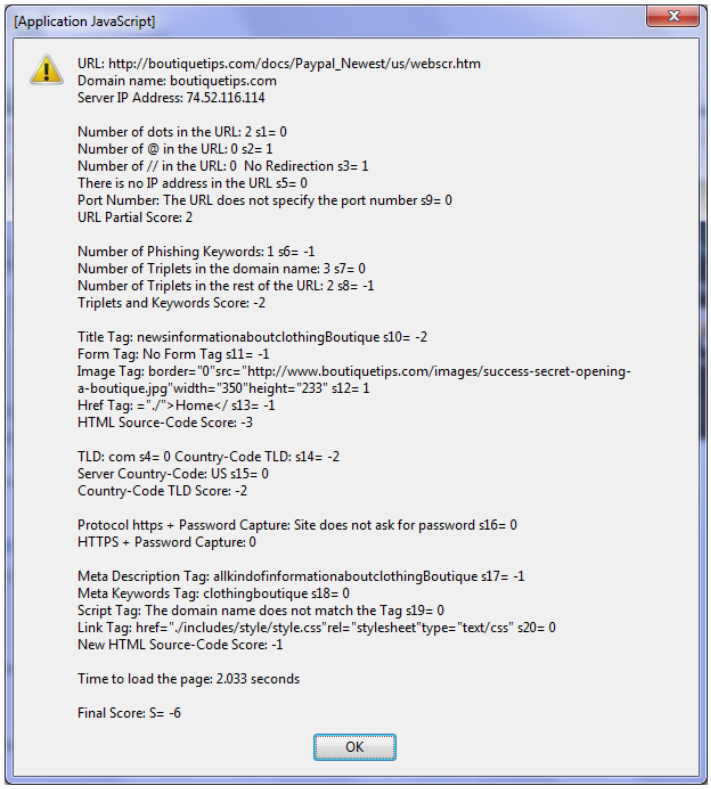
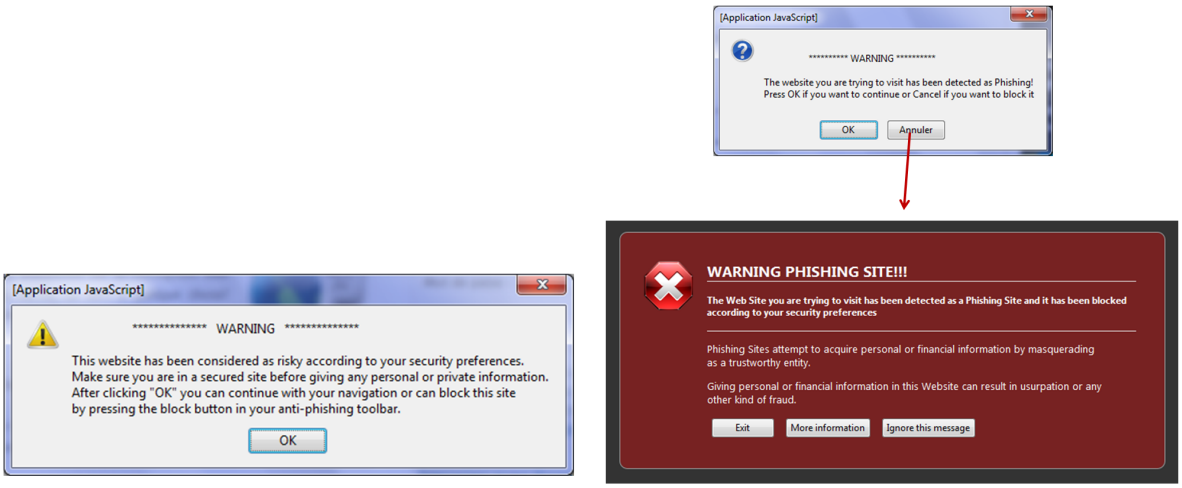


FIGURE 3.5 – *Phishark* : Exemple de scores détaillés des heuristiques



(a) Site Risqué (b) Site Contrefait

FIGURE 3.6 – Messages d'alertes *Phishark* qui requièrent une action de l'utilisateur

de 520 nouvelles URLs. En complément, nous en avons profité pour comparer ses performances aux 4 barres d'outils sélectionnées qui combinent tests heuristiques et blacklist/whitelist (cf. section 3.3).

3.5.1 Performances sur Whitelist

Le tableau 3.6 indique les résultats obtenus lors des tests de performances sur whitelist. En complément, le tableau 3.8 précise l'échelle des scores relevés pour *Phishark*.

Nous constatons que *Netcraft* a réalisé une détection parfaite. Notons toutefois que la barre a certainement été légèrement avantagée, par le fait que le Top 100 des sites les plus visités publié par l'éditeur a contribué à l'élaboration de notre whitelist de test.

Elle est ensuite secondée par *Internet Explorer* qui n'a manqué qu'un seul site : le navigateur nous a en effet fait part d'un message d'erreur, alors qu'au même instant l'ensemble des autres barres réussissaient à accéder au site concerné sans aucun problème.

En troisième position, nous retrouvons *Mozilla Firefox* qui réalise une très bonne détection, malgré 4 sites légitimes indiqués contrefaits à tort. En effet, le navigateur a bloqué ces sites pour cause de certificats problématiques (p.ex. pour un manque de cohérence entre le domaine précisé dans le certificat et le FQDN visité). Précisons que ces problèmes n'ont pas été relevés par *Internet Explorer*.

En quatrième position, nous trouvons *Phishark* qui réalise une bonne performance avec 97.60% de sites détectés légitimes, malgré le non-recours à une liste blanche pour réaliser la détection (à contrario des autres barres mieux classées). Les 12 sites légitimes détectés contrefaits à tort sont majoritairement dus aux heuristiques de la catégorie *Code source HTML*. En effet, sur ces sites légitimes les balises HTML sont incohérentes avec le domaine visité. On peut notamment citer le cas du site Mozilla `http://pv-mirror01.mozilla.org/` - typiquement mentionné dans les listes blanches - qui permet de télécharger des versions logicielles d'outils développés par l'éditeur (p.ex. le client de messagerie Thunderbird). Ce site obtient le score global de -1 pour cause de contenus de balises non soignés (p.ex. la balise <TITLE> contient `indexof`).

On trouve également quelques rares cas qui amènent à des scores négatifs importants, à l'image du site de login Hotmail `https://login.live.com/login.srf?wa=wsignin1.0&rpsnv=11&ct=1314453909&rver=6.1.6206.0&wp=MBI&wreply=http://mail.live.com/default.aspx?rru=inbox&lc=1036&id=64855&mkt=fr-FR&cbcxt=mai&snsc=1` qui obtient un score de -7. Celui-ci cumule en effet un ensemble de comportements suspects : une URL complexe qui utilise de nombreux points, l'utilisation de triplets et mots-clés dits de phishing, l'utilisation de plusieurs TLDs, ou encore des balises sans rapport avec le FQDN (p.ex. la balise <TITLE> contient : `connexion`). De plus, ce site est hébergé sur un nom de domaine global et commun à l'ensemble des services Windows Live de Microsoft, alors que les quelques balises qui pourraient rappeler le FQDN portent uniquement mention du service accédé (p.ex. la balise <META DESCRIPTION> indique `Hotmail nouvelle generation est arrive...`, alors que le FQDN mentionne exclusivement la famille de service Live).

Notons toutefois que le score moyen relevé sur Whitelist pour *Phishark* est de 5.92, avec un écart-type de 3.82. L'intervalle de confiance à 95% (c.-à-d. l'incertitude d'estimation) associé oscille quant à lui entre 5.58 et 6.25 (cf. tableau 3.8). Ces résultats laissent donc présager d'une classification globalement correcte des sites légitimes sur un plus grand nombre d'URLs.

Enfin, en dernière position, nous trouvons *WOT* qui obtient un niveau de détection relativement satisfaisant à 96.20%. Nous notons toutefois que la barre émet une alerte erronée pour 9 sites et, plus surprenant encore, qu'elle ne délivre aucune information pour 10 autres sites.

3.5.2 Performances sur Blacklist

Le tableau 3.7 indique les résultats obtenus lors des tests de performances sur blacklist. En complément, le tableau 3.8 précise l'échelle des scores relevés pour *Phishark*.

En première position, nous trouvons *Phishark* qui réalise la meilleure performance - en comparaison avec les 4 autres barres testées -, en détectant 97.12% des sites de phishing. Ceci s'explique principalement par les nombreux tests heuristiques utilisés par le moteur de détection, dont l'usage semble

Tableau 3.6 – Résultats des tests de performance des 5 barres d'outils sur une Whitelist de 500 URLs

Classement	Barre anti-phishing	Détections positive ou neutre		Faux-positif		Aucune information[1]	
		Quantité	%	Quantité	%	Quantité	%
1	Netcraft	500	100	-	-	-	-
2	Internet Explorer	499	99.80	-	-	1	0.20
3	Mozilla Firefox	496	99.20	4	0.80	-	-
4	**Phishark**	488	97.60	12	2.40	-	-
5	Web of Trust	481	96.20	9	1.80	10	2.00

[1] site déclaré indisponible ou absence de décision de la barre.

moindre dans les autres barres (au profit des blacklists).

Néanmoins, *Phishark* délivre une décision erronée pour 15 sites. Après investigation sur ces mauvaises détections, il apparaît que les URLs concernées sont principalement celles qui utilisent une URL relativement simple quasi-exclusivement constituée d'un FQDN ou d'une arborescence standard de type `http://www.mondomaine.com/dossier1/dossier2/fichier.html`. On peut par exemple citer les cas des sites `http://olegya.fr/` qui obtient un score de 3, `http://www.worldofwarcrajft-account.com/` qui obtient un score de 2, ou encore `http://www.linsenmeier-naturwein.de/naturweine/irs/irsb/SearchTAXERR.php` qui obtient un score de 3. Pour ces cas, nous relevons que les deux premières catégories d'heuristiques (c.-à-d. *Points et caractères spéciaux* et *Triplets et mots-clés dits de phishing*) sont les principales causes de ces détections erronées. En effet, elles sont insuffisamment compensées par les mauvais scores obtenus sur les heuristiques des catégories *Code source HTML* et *Page de login* qui détectent respectivement les balises incohérentes avec le FQDN visité (c.-à-d. toujours liées au FQDN légitime usurpé) et la présence de zones de login dans un site qui ne propose pas de connexion sécurisée.

Malgré ces mauvaises détections, nous notons que le score moyen relevé sur Blacklist pour *Phishark* est de -5.93, avec un écart-type de 3.37, l'intervalle de confiance à 95% associé oscillant entre -6.22 et -5.64 (cf. tableau 3.8). A nouveau, ces résultats rendent assez confiants sur une classification correcte des sites de phishing sur un plus grand nombre d'URLs. Ceci sous réserve bien sûr de l'évolution des techniques de phishing.

Concernant les autres barres d'outils, nous avons remarqué que leurs performances étaient fortement impactées à la baisse dès lors que nous tentions de tester les URLs de phishing dès leur récupération. Nous avons donc réalisé les tests de performances par mini-blocs de 50 à 100 URLs maximum (cf. section 3.3), tant pour minimiser le taux de pertes des URLs (dus à la courte durée de vie des sites de phishing), que pour laisser un délai de synchronisation/mise à jour raisonnable aux blacklists utilisées par ces barres. Ainsi, en moyenne, chaque URL n'était testée que 2 à 3 heures après sa récupération (c.-à-d. le temps nécessaire à la récupération des mini-blocs d'URLs de phishing), en débutant bien évidemment par les premières URLs récupérées.

Nous notons alors que *Netcraft* délivre le deuxième meilleur taux de détection des sites de phishing, avec 90.38% d'alertes. Elle est ensuite suivie par *Mozilla Firefox* qui retourne un taux de détection de 87.31%. En quatrième position arrive *Internet Explorer* avec un taux de 78.08% de détections correctes. Enfin, *WOT* ferme à nouveau les rangs avec une détection de 77.12% des sites contrefaits.

Nous remarquons également que *Mozilla Firefox* et *Internet Explorer* semblent avoir recours à des blacklists différentes. En effet, pour bon nombre de sites de phishing, nous avons vu que certains d'entre eux étaient détectés par une barre et non par l'autre, puis inversement. Néanmoins, au vu des résultats obtenus, il semblerait que celle de *Mozilla Firefox* soit plus performante et/ou plus rapidement mise à jour, de même que le navigateur semble plus rapide pour le chargement des sites.

Aucune barre n'est épargnée par les taux de faux-négatifs qui sont les plus élevés pour *Mozilla Firefox* et *Internet Explorer*, alors respectivement de 12.31 et 18.08%. A contrario, *WOT* délivre moins de faux-négatifs (c.-à-d. 5.19%), mais ceux-ci sont remplacés par une absence totale d'information pour 17.69% des sites testés.

Pour conclure cette phase, précisons que le temps demandé par *Phishark* pour l'évaluation d'un site est en moyenne de l'ordre de 2 à 3 secondes. Ce qui permet d'alerter l'utilisateur dans un délai

Tableau 3.7 – Résultats des tests de performance des 5 barres d'outils sur une Blacklist de 520 URLs

Classement	Barre anti-phishing	Détection négative ou neutre		Faux-négatif		Aucune information[1]	
		Quantité	%	Quantité	%	Quantité	%
1	**Phishark**	505	97.12	15	2.88	-	-
2	Netcraft	470	90.38	44	8.46	6	1.15
3	Mozilla Firefox	454	87.31	64	12.31	2	0.38
4	Internet Explorer	406	78.08	94	18.08	20	3.85
5	Web of Trust	401	77.12	27	5.19	92	17.69

[1] site déclaré indisponible ou absence de décision de la barre.

Tableau 3.8 – Échelle des scores finaux *Phishark* sur Whitelist de 500 URLs et Blacklist 520 URLs

	Échelle des scores relevés (min $\leq$ moyenne $\leq$ max)	Écart-Type	Intervalle de confiance à 95%
WHITELIST	-7 $\leq$ **5.92** $\leq$ 16	3.82	[5.58 ; 6.25]
BLACKLIST	-15 $\leq$ **-5.93** $\leq$ 6	3.37	[-6.22 ; -5.64]

raisonnable (c.-à-d. avant qu'il n'ait pu saisir ses données confidentielles) en cas de site suspect.

3.6 Phase d'identification des heuristiques déterminants

Après avoir apprécié les performances de *Phishark*, comparables à celles des autres barres d'outils, nous avons utilisé notre barre de détection afin d'identifier les heuristiques prédominants pour la décision de légitimité/contrefaçon d'un site.

Pour ce faire, nous avons appliqué notre moteur de détection à 230 URLs de whitelist (extraites des 500 utilisées lors de la vérification) et 230 nouvelles URLs de blacklist. En effet, de par la durée de vie très courte des sites de phishing et le temps nécessité pour tester les performances des 5 barres, il ne nous a pas été possible de réutiliser les URLs précédentes de blacklist ou de procéder à ces tests au cours de la phase de vérification.

Ainsi dans cette nouvelle phase, pour chaque URL testée, nous avons noté les scores intermédiaires de chacune des 6 catégories d'heuristiques, en complément du score global. Les échelles de scores intermédiaires relevés sont détaillées dans les tableaux 3.9 et 3.10, tandis que les échelles de scores finaux sont détaillées dans le tableau 3.11.

Puis, nous en avons déduit les heuristiques prédominants sur whitelist et blacklist, selon la technique énoncée en section 3.3. Les résultats obtenus sont présentés en figure 3.7.

3.6.1 Heuristiques prédominants pour la Whitelist

D'après la figure 3.7, nous constatons que pour les sites légitimes, les catégories *Points et caractères spéciaux* (à hauteur de 95.65%[1]) et *Triplets et mots-clés dits de phishing* (à hauteur de 71.30%[1]) qui portent sur l'étude de l'URL, ainsi que les catégories *Code source HTML* (à hauteur de 66.09%[1]) et *Autres balises HTML* (à hauteur de 63.04%[1]) qui s'intéressent à l'analyse du code source de la page web, sont déterminantes pour l'identification d'un site légitime.

A contrario, les catégories *TLD* et *Page de login* ne semblent pas ici déterminantes, de par leurs taux relevés respectivement à 11.74%[1] et 25.65%[1].

Nous pouvons alors en déduire que les facteurs influençant pour une décision de légitimité sont donc la simplicité de l'URL, et un remplissage adéquat des balises HTML via l'inclusion d'une référence au FQDN visité. Les échelles des scores relevés pour les catégories d'heuristiques associées y sont en effet les plus élevées (cf. tableau 3.9).

1. c-à-d. ces catégories d'heuristiques délivrent un score positif pour le pourcentage d'URL testées mentionné.

Tableau 3.9 – Échelles des scores intermédiaires *Phishark* sur Whitelist de 230 URLs

	Échelle des scores relevés (min ≤ moyenne ≤ max)	Écart-Type	Intervalle de confiance à 95%
Points et caractères spéciaux	-2 ≤ **1.80** ≤ 3	0.77	[1.71 ; 1.90]
Triplets et mots-clés (dits de phishing)	-4 ≤ **0.86** ≤ 4	1.53	[0.66 ; 1.06]
TLD	-2 ≤ **0.05** ≤ 3	0.55	[-0.02 ; 0.12]
Code source HTML	-5 ≤ **0.95** ≤ 5	2.93	[0.57 ; 1.33]
Page de login	-2 ≤ **0.33** ≤ 3	1.59	[0.13 ; 0.54]
Autres balises HTML	-1 ≤ **1.11** ≤ 4	1.16	[0.96 ; 1.26]

Tableau 3.10 – Échelles des scores intermédiaires *Phishark* sur Blacklist de 230 URLs

	Échelle des scores relevés (min ≤ moyenne ≤ max)	Écart-Type	Intervalle de confiance à 95%
Points et caractères spéciaux	-3 ≤ **1.13** ≤ 3	1.20	[0.97 ; 1.28]
Triplets et mots-clés (dits de phishing)	-5 ≤ **-1.16** ≤ 4	2.01	[-1.42 ; -0.90]
TLD	-2 ≤ **-0.26** ≤ 1	0.64	[-0.34 ; -0.17]
Code source HTML	-5 ≤ **-4.60** ≤ 3	1.46	[-4.79 ; -4.41]
Page de login	-2 ≤ **-1.57** ≤ 0	0.83	[-1.67 ; -1.46]
Autres balises HTML	-1 ≤ **-0.42** ≤ 2	0.61	[-0.50 ; -0.34]

A contrario, la notion de connexion sécurisée ne semble pas ici déterminante, ce qui peut aisément s'expliquer par la grande diversité des URLs testées (cf. section 3.3). Appliquée à des URLs de login exclusivement, l'échelle des scores relevés s'en trouverait certainement rehaussée. Il en va de même pour la vérification des informations de localisation du serveur web (via l'analyse du TLD et du pays hébergeur du site). Là aussi, cette faible importance peut s'expliquer par la forte proportion d'URLs utilisant des TLDs génériques[1] et/ou le grand nombre de sites webs gérés par des multinationales. En effet, ces cas de figures rendent difficile une quelconque bonification engendrée par la cohérence des informations TLD / pays hébergeur.

3.6.2 Heuristiques prédominants pour la Blacklist

Pour les sites de phishing, nous constatons que les catégories *Code source HTML* (à hauteur de 96.52%[2]) et *Page de login* (à hauteur de 78.26%[2]) qui s'intéressent à l'analyse du code source de la page web, ainsi que la catégorie *Triplets et mots-clés dits de phishing* (à hauteur de 63.48%[2]) qui porte sur l'étude de l'URL, sont déterminantes pour l'identification d'un site contrefait (cf. figure 3.7).

Une autre catégorie semble d'importance moindre, quoique non négligeable : *Autres balises HTML* avec un taux de 46.96%[2] (cf. figure 3.7).

Enfin, les 2 autres catégories *Points et caractères spéciaux* et *TLD* ne semblent pas ici déterminantes, par leurs taux relevés respectivement à 15.22%[2] et 24.35%[2] (cf. figure 3.7).

Nous pouvons alors en déduire que les facteurs influençant pour une décision de contrefaçon sont donc les incohérences présentes dans les balises HTML (principalement les liens <A HREF> et <IMG>, le titre, ou le contenu des formulaires), l'absence de connexion sécurisée pour les pages de login (toutes les pages de phishing présentent des zones de login), ou la présence des triplets/mots-clés caractéristiques de phishing dans l'URL visitée. Les échelles des scores relevés pour les heuristiques associés indiquent en effet qu'ils présentent les valeurs les plus basses (cf. tableau 3.10).

A contrario des sites légitimes, la complexité des URLs ne semble pas ici déterminante. On peut donc en déduire que cet heuristique n'est indispensable qu'à l'identification des sites légitimes. Par ailleurs, les incohérences entre TLD et pays hébergeur du site ne semblent pas non plus décisives. En effet - de même que vu pour les sites légitimes -, la forte proportion d'URLs utilisant des TLDs génériques explique à nouveau le manque d'efficacité de la catégorie d'heuristiques associée. Notons toutefois que cette catégorie permet tout de même d'aboutir à des scores majoritairement négatifs, ce qui n'était pas le cas pour les sites légitimes.

1. cf. section 4.1.1.1 pour plus d'informations sur ces TLDs.
2. c-à-d. ces catégories d'heuristiques délivrent un score négatif pour le pourcentage d'URL testées mentionné.

Tableau 3.11 – Taux de détections et échelles des scores finaux *Phishark* sur Whitelist et Blacklist de 230 URLs

	Taux de détection correcte ou neutre		Taux de Faux-négatif ou Faux-positif		Échelle des scores relevés	Écart-Type	Intervalle de confiance à
	Quantité	%	Quantité	%	(min ≤ moyenne ≤ max)		95%
WHITELIST	218	94.78	12	5.24	-7 ≤ **5.12** ≤ 16	4.20	[4.57 ; 5.66]
BLACKLIST	227	98.70	3	1.30	-14 ≤ **-6.89** ≤ 3	3.35	[-7.32 ; -6.46]

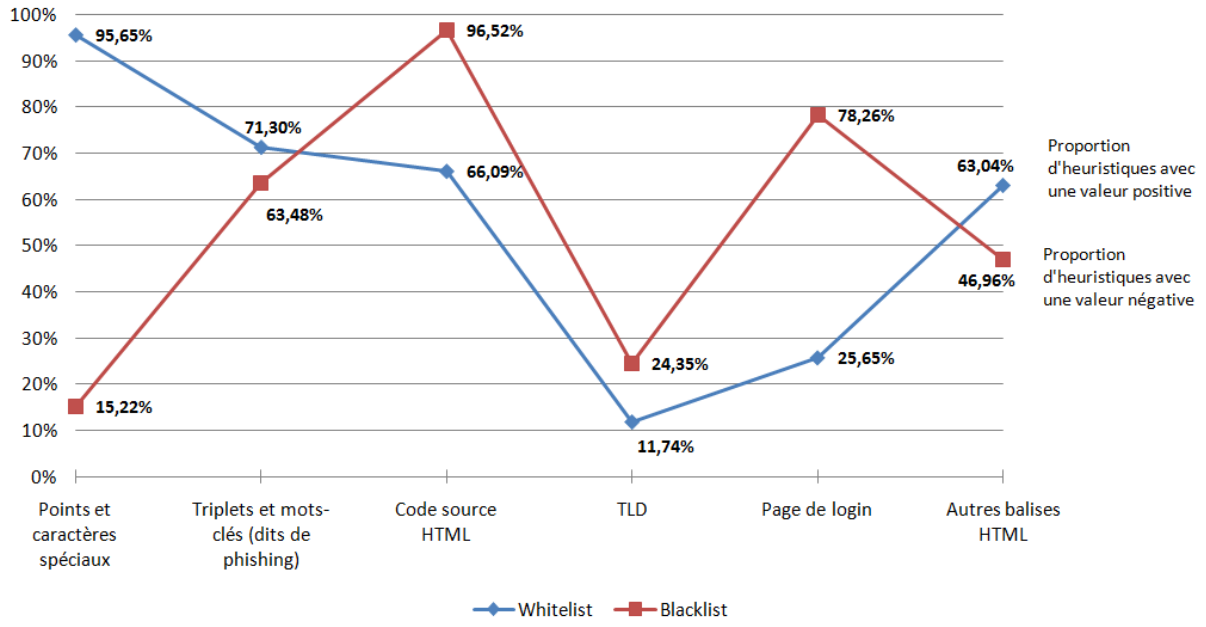


Figure 3.7 – Heuristiques déterminants pour la différenciation des sites légitimes et contrefaits

Enfin, pour conclure notre analyse des heuristiques déterminants, nous pouvons remarquer que les taux de détections obtenus dans cette phase sont d'envergure comparable à ceux relevés lors de la phase de vérification : 94.78% (contre 97.60% précédemment) pour la whitelist, et 98.70% (contre 97.12% précédemment). On note toutefois une légère amélioration pour la détection des sites de phishing et une légère diminution pour la détection des sites légitimes. Cette dernière peut être éventuellement compensée - à l'usage - par la création d'une whitelist personnelle, telle que *Phishark* le propose.

Par ailleurs, le but ultime de cette phase d'analyse est d'aider à affiner les heuristiques et seuils de détection associés, lors d'une étude ultérieure (cf. propositions effectuées en Chapitre 7).

3.7 Discussion sur la pérennité des heuristiques

Au-delà des tests de performances réalisés dans ce chapitre, se pose la question de la pérennité des tests heuristiques utilisés. Parmi les 20 heuristiques utilisés, nous pouvons aisément imaginer qu'il peut être assez facile pour un attaquant de modifier son site contrefait afin qu'il réponde aux exigences de 10 critères (c.-à-d. les heuristiques N° 1, 2, 3, 4, 8, 9, 10, 12, 17 et 18 - cf. tableau 3.5). En effet, si celui-ci a connaissance des tests effectués, il peut modifier son URL afin qu'elle réponde à un schéma plus standard/classique (c.-à-d. peu de points, pas de (@), pas de (//), pas d'adresse IP en guise de FQDN, pas de mots-clés et pas de TLDs "sensibles"), ainsi que le code source de sa page web (c.-à-d. des balises <TITLE>, <META KEYWORDS> et <META DESCRIPTION> cohérentes avec son FQDN), sans que cela lui soit trop coûteux.

Sous réserve que notre moteur de détection soit amélioré afin d'analyser l'ensemble des balises HTML d'un même type (cf. Chapitre 7), il peut ensuite apparaître plus contraignant/problématique de répondre aux exigences de 2 autres critères (c.-à-d. les heuristiques N° 19 et 20 - cf. tableau 3.5). En

effet, il est plus astreignant d'être cohérent au niveau des balises <SCRIPT> et <LINK>, puisque les premières nécessitent un hébergement de l'ensemble des scripts sur le serveur web malveillant, tandis que les secondes nécessitent un hébergement local de la feuille de style par exemple.

Enfin, il est nettement plus compliqué car trop contraignant de répondre aux exigences des 8 critères restants (c.-à-d. les heuristiques N° 5, 6, 7, 11, 13, 14, 15 et 16 - cf. tableau 3.5), voire même quasi-impossible pour certains. En effet, de par les techniques rapides, peu coûteuses et plus transparentes qui sont actuellement utilisées par les attaquants pour confectionner leurs sites contrefaits (c.-à-d. via l'aspiration de sites webs et le maintien d'un maximum de redirections de liens vers le site légitime), l'effort demandé serait considérable. Au-delà de la nécessité de dupliquer l'intégralité du site web usurpé (pour les formulaires, tous les liens <IMG>, <A HREF>), il faudrait également avoir connaissance des triplets de phishing. De plus, les attaquants ont fortement recours aux réseaux de botnets pour héberger leurs sites contrefaits (cf. section 2.3), ceci afin d'être moins facilement identifiables. Cela implique donc souvent l'utilisation d'une redirection de port et/ou une difficulté à être hébergé dans le pays associé au TLD utilisé dans l'URL de phishing. Pour terminer, le critère le plus compliqué à atteindre est certainement celui qui concerne l'utilisation d'une connexion HTTPS. En effet, celle-ci nécessite l'utilisation d'un certificat côté serveur. Si celui-ci est invalide ou incohérent avec le FQDN visité, le navigateur web du client émet aussitôt un message d'alerte (indépendamment d'une quelconque détection du phishing). A contrario, si l'attaquant cherche à détenir un certificat valide, il en devient plus facilement identifiable.

3.8 Problèmes rencontrés

L'une des premières difficultés rencontrées dans cette étude a déjà été en partie exposée en sections 3.3 et 3.5.2. En effet, la durée de vie très courte des sites de phishing introduit un degré de péremption des blacklist élevé. Cet état de fait est à la fois rassurant sur le niveau de réactivité des FAI/hébergeurs qui suspendent les sites émetteurs, mais également fortement contraignant si l'on souhaite utiliser ces blacklists à des fins d'analyses. Par conséquent, le délai entre la récupération des URLs et leurs tests doit être aussi court que possible. De plus, bien que les blacklists proposées par Phishtank et l'APWG soient très régulièrement rafraîchies - p.ex. toutes les 5 minutes sur le site de l'APWG - afin d'informer des derniers sites de phishing détectés, elles n'en comportent pas moins une forte proportion d'URLs périmées. Ainsi, il faut donc effectuer un premier écumage à la récupération des URLs de phishing, pourtant déclarées valides et encore en ligne. En moyenne, le temps de récupération des mini-blocs de 50 à 100 URLs nécessite donc facilement 2 à 3 heures, ce qui n'empêche aucunement que les premières URLs récupérées soient périmées durant ce laps de temps.

Les autres problèmes rencontrés les plus notables et/ou les précautions qu'il a fallu prendre ont principalement concerné la mise en œuvre des heuristiques au sein du moteur de détection. En effet, en premier lieu il faut s'assurer que la page web récupérée est bien valide. En effet, en cas d'indisponibilité de site, il ne sert à rien de lancer la détection de phishing, tant pour ne pas délivrer de décision erronée que ne pas fatiguer l'utilisateur avec de fausses alarmes intempestives. Pour ce faire, nous analysons l'en-tête HTTP récupéré avec la page afin de s'assurer que le code de récupération associé est bien "200" - ce qui signifie que la page a été récupérée avec succès[1] -, avant toute exécution du moteur de détection. Néanmoins, cette technique n'est pas infaillible puisqu'il nous est arrivé de rencontrer des pages d'erreur (p.ex. liées à une indisponibilité temporaire de site) improprement codées au niveau de l'en-tête HTTP (c.-à-d. avec un code d'erreur "200").

Les heuristiques associés aux balises HTML posent également parfois problèmes, puisque les spécificités de certaines langues introduisent des incohérences entre contenu de certaines balises et FQDN visité. En effet, par exemple les FQDN ne contiennent un caractère accentué. Ainsi, à titre d'illustration, une banale vérification de cohérence entre la balise <TITLE> (qui contient `Banque et Assurances - Société Générale`) et son FQDN associé `https://particuliers.societegenerale.fr/index.html` pour le site légitime de login de la Société Générale, nous retournerait un score négatif, à tort. Pour pallier ce problème, nous avons été amenés à implémenter des fonctions spécifiques qui visent à remplacer tous les caractères accentués[2] des balises par leurs pendants sans accent (p.ex. é est remplacé

1. cf. section 6.1.5.2 pour plus de détails sur l'en-tête HTTP et ses codes d'erreurs.
2. aujourd'hui notre implémentation porte sur les caractères "a", "e", "i", "o", "u", "y", "æ", "c" et "n".

par `e`, `ñ` est remplacé par `n`, etc.). Néanmoins, la liste utilisée aujourd'hui n'est pas exhaustive. Elle se doit donc d'être enrichie pour éviter ce travers.

Enfin, un des soucis rencontrés pour lequel nous n'avons pas de solution actuellement concerne la redirection de sites webs. En effet, certaines URLs légitimes procèdent à une redirection automatique vers d'autres URLs. C'est le cas par exemple du site de login Hotmail, où `www.hotmail.com` n'est qu'un alias qui redirige vers `https://login.live.com/login.srf?wa=wsignin1.0&rpsnv=11&ct=1314530844&rver=6.1.6206.0&wp=MBI&wreply=http:%2F%2Fmail.live.com%2Fdefault.aspx&lc=1036&id=64855&mkt=fr-FR&cbcxt=mai&snsc=1`. Ainsi l'analyse est effectuée uniquement à partir de cette dernière URL et de la page web associée. Tels qu'exposés précédemment (cf. section 3.4), la complexité et le manque de lien entre contenu de la page et FQDN de l'URL de redirection entraîne alors un score négatif pour ce site (de -7). Ceci n'aurait pas été le cas, si nous avions utilisé l'URL originale saisie par l'utilisateur (c.-à-d. `www.hotmail.com`). Néanmoins, parce que des sites de phishing pourraient également utiliser cette technique en contournement (c.-à-d. en présentant systématiquement une URL simple avant une redirection vers une URL plus complexe), nous ne pensons pas qu'il soit vraiment souhaitable de chercher à remédier à ce problème.

3.9 Synthèse du chapitre

Dans ce chapitre, nous avons évalué l'efficacité et la prédominance des tests heuristiques utilisés au sein de barres anti-phishing, pour la détection des sites contrefaits visités par l'Internaute. Pour ce faire nous avons implémenté notre propre barre anti-phishing, dont le moteur de détection est exclusivement basé sur 20 tests heuristiques qui analysent aussi bien l'URL que le contenu du code source HTML de la page web.

Nous avons ainsi prouvé que l'utilisation des heuristiques - qu'ils se rapportent à l'étude de l'URL ou l'analyse du code source HTML -, est primordiale pour une détection efficace des sites de phishing dès leur apparition. Toutefois, il apparaît clairement que tous ces heuristiques ne sont pas égaux dans la détection. En particulier, la simplicité des URLs légitimes et l'analyse du code source HTML associé sont essentielles pour une décision de légitimité. Cette même analyse du code source HTML et la vérification d'une connexion sécurisée sur une page de login sont à leur tour vitales pour une décision de contrefaçon. Les résultats que nous obtenons ici nous permettent également de distinguer plusieurs pistes d'amélioration des tests heuristiques utilisés. Celles-ci sont exposées dans le Chapitre 7.

Pour terminer, nous mesurons également la principale faiblesse des heuristiques, à savoir : leurs taux de FPR/FNR. Cette faiblesse peut à elle seule aisément expliquer l'utilisation de whitelist/blacklist additionnelles, dans le but d'obtenir une détection moins encline aux erreurs. Toutefois, l'un des inconvénients majeurs introduits par ces listes blanches/noires est l'opportunité supplémentaire qu'elles offrent pour la corruption du moteur de détection.

Dans la même veine, quelque soit l'efficacité de leur détection, les barres anti-phishing n'en demeurent pas moins assujetties aux faiblesses du système sur lequel elles reposent, à savoir : le navigateur web, le poste client et le réseau auquel il appartient. La Seconde Partie de ce mémoire essaie de se pencher sur une fraction de cet épineux problème.

Seconde partie :

Le pharming

4 Le pharming et la comparaison de pages webs

Sommaire

Au cours des dix dernières années, la prolifération de sites webs contrefaits a conduit à explorer diverses pistes de prévention/détection. La plupart des approches développées s'intéressent aux attaques de phishing - grâce à des techniques d'analyses (p.ex. listes noires, tests heuristiques) appliquées sur l'URL et/ou le contenu de la page web contrefaite (cf. Chapitre 2) -, ou aux attaques de pharming réalisées au travers d'une corruption DNS. Dans le cas de ces dernières, il y a deux manières d'aborder le problème : explorer du côté des informations DNS utilisées pour atteindre le site web, ou investiguer au niveau du contenu de la page web contrefaite.

Dans ce chapitre, nous présentons le pharming (au travers des différentes corruptions DNS qui peuvent être effectuées) et les techniques de comparaisons de pages webs pouvant aider à la détection des sites webs contrefaits. Précisons que les explications délivrées dans ce chapitre sont volontairement simplifiées et limitées, pour ne définir que les notions utilisées ultérieurement dans ce document.

En section 4.1.1, nous débutons par quelques rappels DNS nécessaires à la bonne compréhension des attaques et mécanismes de prévention/détection présentés. Puis en section 4.1.2, nous détaillons les différents niveaux d'attaques de pharming. Enfin, en section 4.1.3, nous évoquons les techniques actuellement disponibles pour s'en prémunir, ainsi que les travaux qui s'y rapportent côté client.

Dans un deuxième temps, nous détaillons les deux grandes catégories de méthodes qui s'avèrent pertinentes pour la comparaison de documents HTML : les méthodes de comparaison de texte (cf. section 4.2.1) et les méthodes de comparaisons de structures (cf. section 4.2.2). Pour terminer, la section 4.2.3 discute plus spécifiquement des travaux qui appliquent ces méthodes au cadre de la comparaison de pages webs.

4.1 Le pharming

Les attaques de pharming sont une version sophistiquée des attaques de phishing. L'objectif est le même, à savoir : voler des informations confidentielles aux Internautes telles que des login, mots de passe, numéros de carte bancaire, etc. D'un point de vue technique, la mise en œuvre de ce type d'attaque diffère quelque peu : il faut certes la mise en ligne d'un site web contrefait mais, basée sur une corruption DNS réalisée en amont, l'attaque est cette fois-ci totalement imperceptible à l'utilisateur (sous réserve que l'imitation visuelle de la page web affichée soit de bonne qualité). En effet, grâce à une corruption/redirection effectuée au niveau DNS, l'utilisateur est automatiquement redirigé vers le site web contrefait dès lors qu'il tente d'accéder à son site habituel. Pourtant l'URL visitée est bien l'URL légitime.

Ce type d'attaque peut être considérée comme plus passive que le phishing, car elle ne nécessite pas de campagne de diffusion incitant les Internautes à se rendre sur le site contrefait. Néanmoins elle n'en est que plus redoutable car difficilement détectable.

Avant de s'intéresser aux attaques de pharming et aux moyens de prévention/détection associés, revenons à quelques rappels/considérations DNS nécessaires à la bonne compréhension de la suite de ce chapitre.

4.1.1 Rappels DNS

Dans le monde de l'Internet, les adresses IP sont utilisées afin d'échanger des messages entre machines. Sachant qu'une adresse IP est constituée de 4 à 8 blocs d'octets séparés par des caractères spéciaux[1], leur manipulation et leur mémorisation n'est pas tâche aisée. Le protocole DNS est donc un élément crucial de l'architecture réseaux d'Internet. A l'image de l'annuaire téléphonique qui permet de lier nom d'abonné et numéro de téléphone, le protocole DNS permet de faire la corrélation entre une adresse IP et le nom de domaine (ou plus exactement le FQDN[2]) associé. Afin de simplifier les échanges au travers d'un réseau, il est en effet plus simple d'utiliser un nom de domaine (p.ex. `www.it-sudparis.eu`) plutôt qu'une adresse IP (p.ex. 157.159.11.8). Par la même occasion, l'administration réseaux qui inclut des besoins de partage de charge ou une migration de serveurs s'en retrouve simplifiée.

Le protocole DNS utilise le port 53, généralement en UDP pour les requêtes et TCP pour les transferts de zone[2]. Il s'appuie sur les RFC fondatrices 1034 (concepts) et 1035 (implémentation). Celles-ci sont complétées par les RFC 2136, 2181, 4033 et 6195[3] qui apportent des spécifications complémentaires sur les enregistrements RR[2] et introduisent les notions de sécurité.

4.1.1.1 Architecture DNS

Le DNS [AFNa] [Oll05] s'appuie sur une architecture client/serveur reposant sur 3 dispositifs :
- Un espace de noms hiérarchique permettant de garantir l'unicité des noms de domaine,
- Un système de serveurs distribués permettant la diffusion de ces noms de domaine,
- Et un système client permettant de résoudre les noms de domaine.

4.1.1.1.1 Espace de noms hiérarchique : La structure hiérarchique des noms de domaine est conçue comme une arborescence, ayant pour origine une racine représentée par un ".", afin que chaque nom d'hôte (c.-à-d. machine) soit unique. L'espace de nommage, c.-à-d. le chemin dans l'arbre inversé pour

1. 4 blocs d'octets séparés par des "." pour les adresses IPv4, et 8 blocs de 2 octets séparés par des " :" pour les adresses IPv6. Notons que l'ensemble de notre étude se focalise exclusivement sur les adresses IPv4.
2. cf. section 4.1.1.1.
3. Liste non exhaustive.

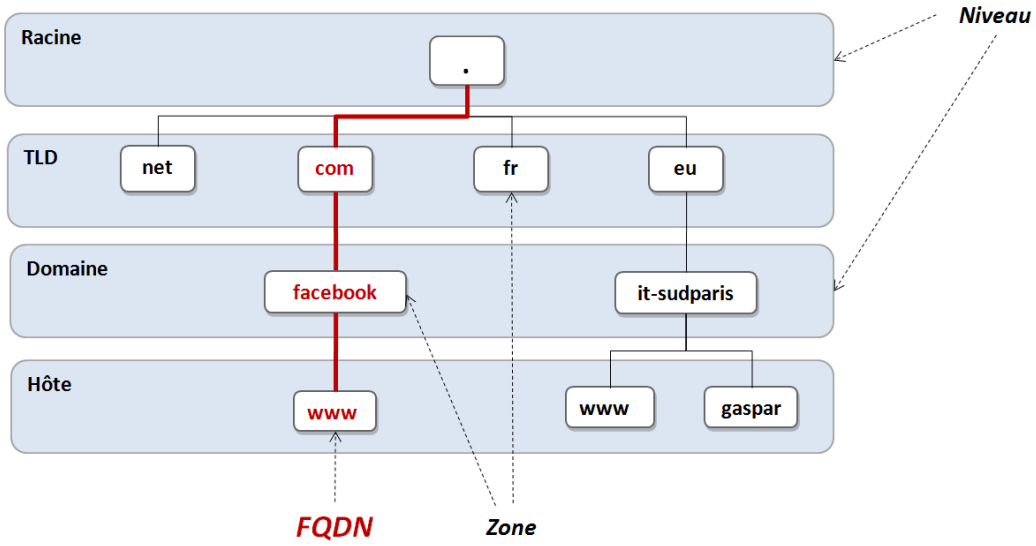


FIGURE 4.1 – Hiérarchie DNS simplifiée

un nom d'hôte donné, comporte typiquement 4 niveaux : racine, TLD, domaine et hôte (cf. figure 4.1). Chaque nom d'hôte (sur 63 caractères maximum) doit être unique dans le domaine concerné. Un domaine peut être éventuellement re-découpé en sous-domaines intermédiaires, tant que le nom d'hôte complet (appelé FQDN et constitué des différents niveaux d'arborescence séparés par des ".") n'excède par 255 caractères et 127 niveaux d'arborescence. Chaque responsable de domaine dispose d'une totale liberté de nommage de ses sous-domaines. Pour un minimum de structure, les TLD - également appelés domaines de premier niveau - suivent une codification spécifique établie par l'ICANN (pour *Internet Corporation for Assigned Names and Numbers*). Ils sont par ailleurs gérés soit directement par l'ICANN, soit par des organismes délégués appelés registres. Enfin, les noms de domaine sont achetés auprès d'un bureau d'enregistrement des domaines, appelé registrar. A noter qu'il est possible de consulter les données contenues par les registres via l'utilisation du protocole WHOIS (RFC 3912, port TCP 43).

Les TLDs se décomposent en plusieurs groupes. Les plus connus sont les g-TLD pour *generic TLD* (p.ex. .COM, .ORG, .GOV, etc.), et les cc-TLD pour *country-code TLD* représentatifs d'un pays ou d'une zone géographique (p.ex. .FR pour la France, .EU pour l'Europe, etc.).

4.1.1.1.2 Système de serveurs distribués : Chaque domaine dispose d'un serveur DNS (au minimum), afin d'annoncer sur Internet les combinaisons FQDN/adresses IPs qu'il détient. Ainsi, il existe des centaines de milliers de serveurs DNS à travers le monde, chacun d'entre eux ne contenant que peu d'informations. A chaque requête DNS effectuée, ce sont en fait de nombreux serveurs DNS qui sont sollicités. On dit de l'architecture DNS qu'elle est un système décentralisé, reposant sur la délégation de domaine. En effet pour que l'ensemble de l'arborescence DNS soit gérable, à chaque niveau (non-terminal) on associe une zone par entité (cf. figure 4.1). Chaque zone est responsable, par délégation de la zone supérieure, de répondre aux requêtes DNS concernant les entités qu'elle détient (c.-à-d. de niveau inférieur).

Traditionnellement, chaque zone contient (jusqu'à) 3 serveurs DNS (cf. figure 4.2) : un serveur primaire contenant les informations de zone (on parle alors de serveur "autoritaire" pour la zone), un serveur de noms secondaire généralement utilisé pour la continuité de services, et un serveur cache qui mémorise les réponses aux précédentes requêtes DNS avec une durée de vie limitée. Les transferts d'informations depuis le serveur DNS primaire vers le serveur DNS secondaire sont appelés transferts de zone.

Le rôle d'un serveur DNS est de répondre à toute requête demandant des informations contenues

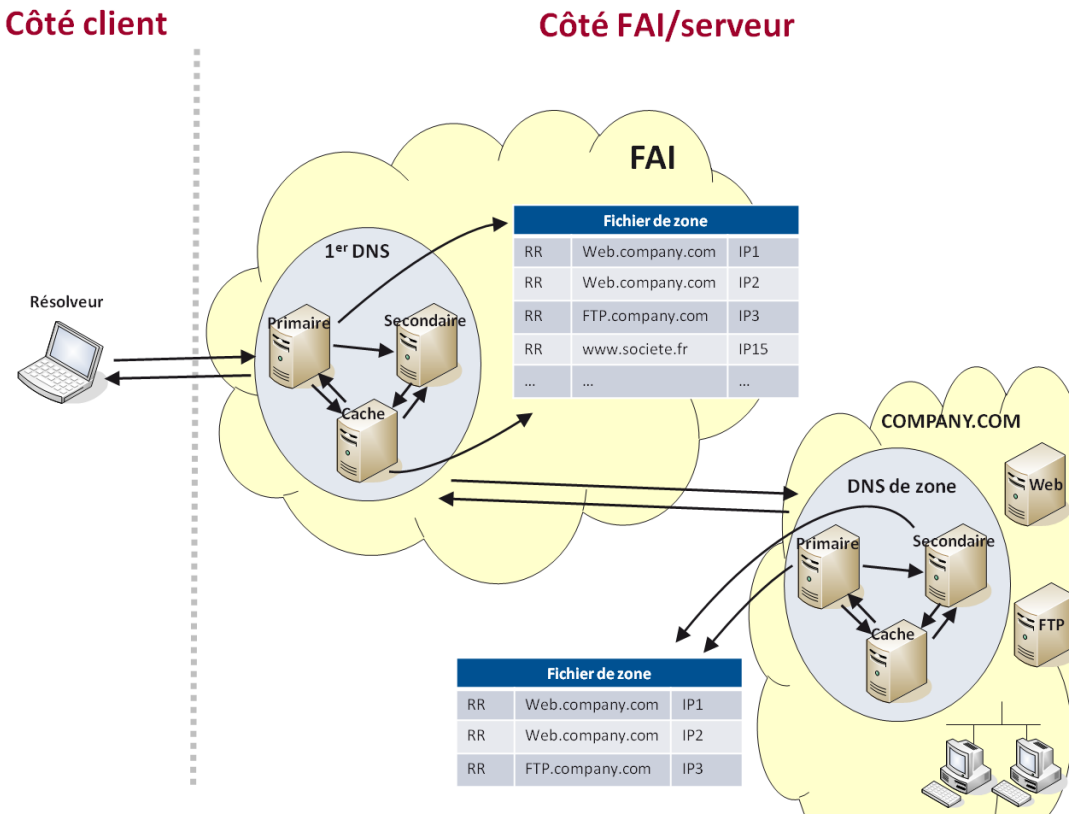


FIGURE 4.2 – Vue simplifiée des zones DNS

TABLEAU 4.1 – Exemples d'enregistrements DNS Resource Record (RR)

RR	FQDN	Type	Classe	TTL	Adresse IP
1	www.google.fr	A	IN	86400	209.85.148.103
2	www.google.fr	A	IN	86400	209.85.148.99
3	www.google.fr	A	IN	86400	209.85.148.105
4	www.it-sudparis.eu	A	IN	43200	157.159.11.8

dans sa base, qu'elles concernent ou non sa zone. Les informations de sa propre zone sont concentrées dans un fichier (nommé fichier de zone) qui contient plusieurs enregistrements appelés *Resource Record* (RR), comme le présente la figure 4.2.

Chaque enregistrement RR contient 5 informations (cf. tableau 4.1) : le FQDN, le Type d'enregistrement, la Classe, le TTL (pour *Time To Live*), ainsi que l'adresse IP associée.

Les types d'enregistrements servent à donner une indication sur la teneur de l'hôte. Les plus connus sont : MX (pour *Mail eXchange*) utilisé pour décrire un serveur mail, NS (pour *Name Server*) utilisé pour décrire le serveur "autoritaire" de la zone, ou A (pour *Address*) utilisé pour décrire une machine/un hôte IPv4 (p.ex. un serveur web, un serveur ftp, etc.). Le TTL permet, quant à lui, d'associer une durée de vie (exprimée en secondes) à chaque enregistrement. Enfin, la Classe permet de décrire le système associé à l'enregistrement. On utilise ici IN pour indiquer *Internet*.

4.1.1.1.3 Système client : Le système client DNS, permettant de réaliser l'opération de résolution de domaine, est appelé résolveur (ou *resolver* en anglais). Installé côté client, il est appelé par les applications du système en vue d'effectuer les requêtes DNS. Ainsi, par l'intermédiaire du résolveur, une application souhaitant contacter un hôte par son nom de domaine envoie une requête au serveur DNS dont il détient l'adresse (typiquement celui du FAI, automatiquement configuré via DHCP, dans le cas d'un réseau personnel).

Trois modes de fonctionnement se présentent alors :

1. Le serveur DNS interrogé possède les informations dans son serveur cache, il répond directement à la requête.
2. Le serveur DNS interrogé n'a pas les informations en cache, mais il connaît un des serveurs de zone du domaine demandé. Il interroge alors directement ce serveur pour obtenir l'adresse IP demandée et ainsi répondre à la requête.
3. Le serveur n'a ni les informations en cache, ni connaissance de la zone demandée. Il génère alors une requête DNS vers un serveur racine pour obtenir une réponse.

A noter qu'un client peut également effectuer une requête DNS à partir d'une adresse IP, en vue de connaître le FQDN associé. On parle alors de requête inverse ou *reverse DNS*.

4.1.1.2 Processus de résolution des requêtes DNS

Les requêtes DNS peuvent être de 2 types : récursives ou itératives. Dans le cas d'une requête récursive, le serveur DNS interrogé doit obligatoirement répondre à la requête (avec une adresse IP/FQDN ou un message d'erreur). Il s'agit là du cas typique de requête générée par un résolveur.

Dans le cas d'une requête itérative, le serveur DNS interrogé retourne tous les éléments en sa possession au demandeur, afin que celui-ci se charge lui-même de poursuivre les requêtes vers d'autres serveurs DNS. Ce type de requête est typiquement utilisée (sauf demande contraire) entre serveurs DNS. Elle est privilégiée ici afin de réduire la charge des serveurs DNS.

4.1.1.3 Exploitation du DNS

De par la quantité colossale de machines qui nécessitent des enregistrements DNS (p.ex. les serveurs webs, emails, FTP, etc.) et la quantité encore plus importante de demandeurs potentiels (c.-à-d. toute machine disposant d'une adresse IP peut émettre des requêtes DNS), l'architecture DNS constitue l'un des socles d'Internet. A ce titre, elle se doit de répondre à des exigences considérables :

- Un service ininterrompu, c.-à-d. les informations DNS doivent être disponibles à tout instant, depuis n'importe quel coin du globe et ce, même en cas de panne d'un serveur DNS.
- Une rapidité pour la réponse au demandeur, indépendamment de la distance géographique entre le client et le service recherché.
- Une répartition de charge optimisée pour ne saturer ni les serveurs DNS interrogés, ni les services accédés.

Pour parvenir à satisfaire ces exigences, plusieurs mécanismes sont mis en œuvre au sein des réseaux des FAI et des entreprises :

- L'utilisation d'un trio de serveurs DNS (c.-à-d. serveur primaire, serveur secondaire et serveur de cache), tant pour assurer la continuité de services que la rapidité des réponses.
- L'utilisation de techniques de répartition de charge pour ne pas saturer les serveurs ciblés. En effet, dès lors que plusieurs adresses IP sont associées à un même hôte au sein des enregistrements DNS contenus dans sa base, le serveur DNS qui répond permute l'ordre des adresses IP retournées, de manière automatique à chaque requête. Cette permutation est réalisée grâce à des techniques telles que le *round-robin* ou le GSLB (pour *Global Server Load Balancing*).

Ces mécanismes présentent toutefois des inconvénients. En effet, l'utilisation de plusieurs serveurs DNS (c.-à-d. primaire et secondaire) entraîne des besoins de réplication de bases et mises à jour, réalisés via les transferts de zones. Ces derniers sont des cibles supplémentaires pour les attaques DNS [Gue06].

Par ailleurs, les techniques de répartition de charge augmentent la quantité d'informations présentes dans les caches DNS. De plus, le *round-robin* effectue une répartition "basique" qui ne tient pas compte de la disponibilité de service associée à chaque adresse IP contenue dans les enregistrements DNS d'un hôte. Les solutions de type GSLB [Net09], qui nécessitent la duplication des services sur un minimum de 2 zones distinctes, permettent de pallier ce problème via une répartition de charge DNS "intelligente", effectuée en fonction de la localisation de l'utilisateur et de la disponibilité du service demandé (c.-à-d.

la capacité des serveurs et leur disponibilité/charge). Cependant, le développement de l'utilisation de ces techniques rend difficilement exploitable le recours aux enregistrements DNS pour identifier d'un domaine (cf. section 5.2.1.4).

4.1.2 Les attaques de pharming

Les attaques ciblant le DNS, typiquement vectrices de pharming, visent à altérer le processus normal de résolution des requêtes DNS. Elles sont classifiées en sous-ensembles qui diffèrent selon les articles et études considérés. Pour notre part, nous avons choisi de les scinder en deux grandes familles, représentatives de la zone attaquée : côté client ou côté réseau FAI/serveur web (cf. figure 4.3).

A noter que dans cette section nous mentionnons également quelques attaques visant le serveur web visité, afin d'offrir une meilleure vue d'ensemble de la chaîne de liaison client - serveur web, ciblée par le vol d'informations confidentielles.

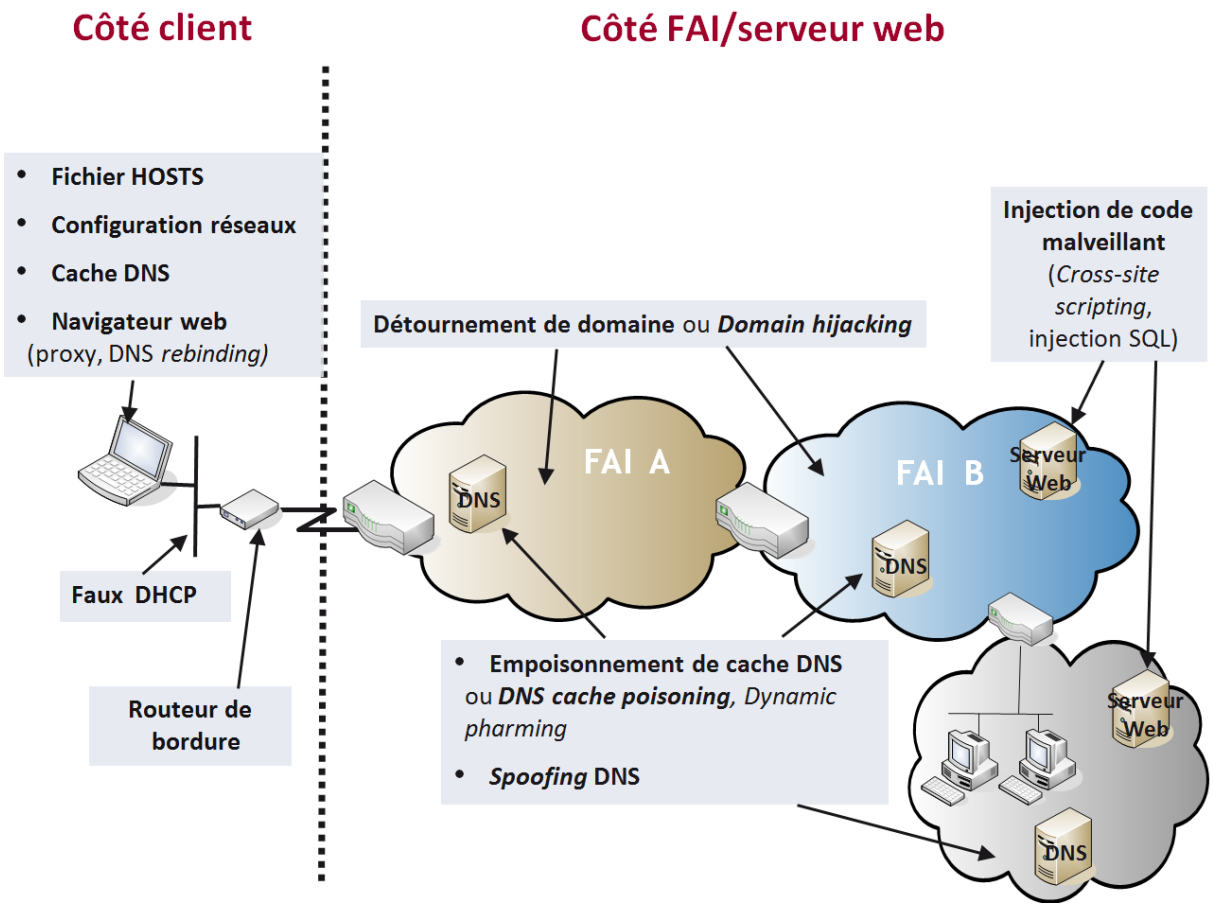


FIGURE 4.3 – Les attaques de pharming réparties par zones cibles

4.1.2.1 Côté client

Les attaques DNS perpétrées côté client [Oll05] peuvent être conduites à 3 niveaux :
- Directement sur le poste client : au niveau du système d'exploitation ou au niveau du navigateur web,
- Sur le réseau local,
- Ou à la frontière entre le réseau local et le FAI, c.-à-d. sur le routeur personnel/de bordure.

Au niveau du poste client, on recense 4 zones de vulnérabilités et/ou cibles d'attaques :

Une première cible peut être le **fichier HOSTS**, présent dans tout système d'exploitation (p.ex. accessible sous Windows dans le répertoire C :\windows\system32\drivers\etc\). Ce fichier sert en effet à définir des associations adresses IP / FQDN de manière statique. Il est consulté par le système, en amont de toute requête DNS envoyée à un serveur.

Une deuxième cible peut être la **configuration réseaux** paramétrée au niveau de la carte d'interfaces. En effet, il est possible de spécifier de manière statique les adresses IP des serveurs DNS interrogés (c.-à-d. primaires et secondaires) au niveau de la carte réseaux. Ces paramètres remplacent alors ceux traditionnellement assignés lors de la configuration dynamique réalisée par DHCP. Dans le cadre d'une utilisation normale, cette zone de configuration est modifiée par les utilisateurs qui souhaitent utiliser un serveur DNS alternatif à celui proposé par leur FAI.

Une troisième cible est le **cache DNS** du poste client, manipulé soit au niveau du système d'exploitation, soit au niveau du navigateur web. Par insertion de fausses entrées (adresse IP/FQDN) dans le cache, qui est consulté avant chaque émission d'une requête DNS, l'utilisateur est alors automatiquement redirigé sur le(s) site(s) web(s) frauduleux.

Ces trois premières cibles peuvent être typiquement corrompues par des scripts malveillants, récupérés au travers d'un spam ou de la navigation web de l'Internaute. Ces scripts se basent aussi bien sur le manque de vigilance/la crédulité des utilisateurs (p.ex. *Wareout* [Mic08] - récupéré par un simple clic lors de la navigation web -, ou le cheval de troie Zcodec - caché à l'intérieur d'un lecteur vidéo dit gratuit [DPLL08] -, configurent statiquement des adresses DNS pirates), que l'exploitation des vulnérabilités du système client (p.ex. une vulnérabilité Microsoft [Mic09] permettait d'ajouter de fausses entrées DNS dans le cache).

Une quatrième zone de vulnérabilités est le **navigateur web** du client. Il peut être corrompu au niveau du cache, comme énoncé précédemment.

Sinon, un **proxy web** peut également être défini au sein du navigateur afin d'entraîner une redirection automatique de toutes les requêtes HTTP vers un serveur web frauduleux.

Enfin, le navigateur web peut être ciblé par une attaque de type ***DNS rebinding*** qui vise à le convertir en proxy ouvert, via l'exploitation des interactions entre plug-ins (p.ex. Flash, Java) et navigateur web [JBB+09]. L'objectif final de cette attaque est d'atteindre et corrompre une machine du réseau local (p.ex. pour ajouter de fausses entrées DNS dans un PC ou le routeur personnel).

Le réseau local, filaire ou sans-fil est également une zone de vulnérabilités. Par l'installation d'un **faux** serveur **DHCP** (p.ex. le cheval de troie Flush.M [Sym08]), un attaquant peut alors assigner de faux paramètres DNS aux utilisateurs. Cette vulnérabilité est d'ailleurs exacerbée par la forte utilisation des réseaux sans-fils déployés à domicile, pour l'accès à Internet. L'étude menée par Abu-Nimeh et al. est un exemple flagrant de la mise en œuvre de ce scénario [AN08].

Enfin, le **routeur personnel** situé à la frontière avec le réseau Internet - traditionnellement fourni par le FAI -, est lui aussi une cible privilégiée des attaquants. Une étude menée par Stamm et al. [SRM07] a démontré la vulnérabilité de ces routeurs personnels, basée sur la double exploitation des lacunes de configuration de ces routeurs et du support du Javascript et des applets Java dans le navigateur client.

En effet, bon nombre d'utilisateurs utilisent ces boîtiers prêts-à-installer dans leur pré-configuration d'origine, sans autre paramétrage de sécurité que la définition d'un mot de passe sécurisant la connexion sans-fil PC(s) - routeur. Les utilisateurs omettent alors généralement de modifier les paramètres d'accès à la configuration du routeur, laissant ainsi activés les login/mot de passe par défaut. En quelques clics sur Internet, il est rapide et aisé de trouver des listes de paramètres d'accès par défaut (c.-à-d. login/mot de passe) à ces équipements [Phe10]. Après une identification de l'adresse IP utilisée par le routeur (p.ex. grâce à une écoute discrète effectuée à partir d'applets Java), et l'utilisation de requêtes HTTP (intégrées à l'intérieur de balises <script> dans le code source d'une page web), l'accès à la configuration DNS du routeur personnel peut être réalisé. On peut alors spécifier - de manière statique - l'adresse IP d'un faux serveur DNS, ou créer des règles de routage qui redirigent le trafic.

Bien évidemment, les cas d'attaques exposés dans cette étude répondent à des conditions spécifiques (p.ex. la nécessité d'un mot de passe par défaut vide pour certaines attaques, l'acceptation de messages d'alertes par l'utilisateur pour d'autres, etc.). Néanmoins, elles sont tout à fait viables.

Il est important de souligner que cet article [SRM07], que nous avons considéré comme marquant, a été l'un des points d'origine de notre réflexion dans la recherche d'une méthode de détection des attaques de pharming côté client.

4.1.2.2 Côté réseau FAI/serveur web

Côté réseau FAI/serveur web, les attaques peuvent se situer à différents niveaux [Oll05] :

- Détournement du nom de domaine (ou *Domain hijacking* en anglais) par modification de son enregistrement,
- Altération du fonctionnement normal d'un serveur DNS par *spoofing* DNS,
- Corruption de cache DNS (ou *DNS cache poisoning* en anglais) au niveau d'un serveur DNS,
- Ou directement au niveau du serveur web visité, par l'injection de code malveillant.

Le **détournement du nom de domaine** (ou *Domain hijacking* en anglais) consiste à s'approprier un nom de domaine légitime, basé sur des failles liées au processus d'enregistrement de celui-ci. Le cas du domaine `Panix.com`, détenu par un FAI qui délivre des accès Internet sur New-York, en est un exemple. En effet, à cause d'un oubli dans le processus d'enregistrement, le domaine du FAI a été détourné (c.-à-d. les adresses IP ont été modifiées) au profit d'une société Australienne, durant quelques dizaines d'heures en 2005 [spe05]. Ce type d'attaque, relativement rare, ne fonctionne généralement que sur une courte durée.

Le ***spoofing* DNS** consiste à altérer le fonctionnement normal d'un serveur DNS légitime afin d'émettre de fausses réponses DNS. Ces dernières sont alors délivrées soit par le serveur DNS légitime lui-même (p.ex. grâce à l'utilisation de techniques d'empoisonnement de cache), soit par le serveur DNS de l'attaquant, en lieu et place du serveur légitime (p.ex. via des techniques de dénis de service conduisant à un arrêt ou un ralentissement des réponses du serveur DNS légitime). Ces attaques peuvent cibler le serveur DNS d'un FAI, ou le serveur DNS autoritaire d'une zone (p.ex. par corruption des transferts de zone).

Les attaques de type ***DNS cache poisoning*** visent à corrompre le cache d'un serveur DNS (localisé chez un FAI ou dans une zone) par insertion de faux couples d'adresses IP/FQDN. Elles sont souvent utilisées en association avec une attaque de pharming basée sur du *spoofing* DNS.

Une variante de ce type d'attaque - appelée *Dynamic pharming* - combine la délivrance au client d'un code Javascript malveillant et l'utilisation de techniques de *DNS rebinding* (cf. section 4.1.2.1). L'association de ces deux techniques permet d'incorporer la page web légitime au sein de la page web malveillante visualisée par le client.

La figure 4.4, basée sur celle contenue dans l'article de Olzak [Olz06], montre un exemple d'attaque de pharming combinant des techniques de *DNS cache poisoning* pour corrompre le cache DNS du FAI (via l'étape 7b de la figure), et de *spoofing* DNS pour effectuer un déni de service sur le serveur DNS de la zone *company.com*.

Enfin, nous pouvons mentionner les **injections de code malveillant** (par *Cross Site Scripting* (XSS) ou injection de code SQL) qui ciblent le serveur web visité.

Ces techniques consistent à insérer des scripts malveillants et/ou des redirections de contenu au sein d'une page légitime (p.ex. insertion d'une fausse zone de saisie de login/mot de passe), via l'exploitation de failles inhérentes aux serveurs et/ou navigateurs webs utilisés. A titre d'exemple, on peut citer le cheval de troie Sinowal [RSA08] qui était exécuté sur le poste client. Actif sur plus de 2700 URLs de services financiers, il a permis de dérober les données de plus de 300 000 comptes bancaires sur une période de 3 ans.

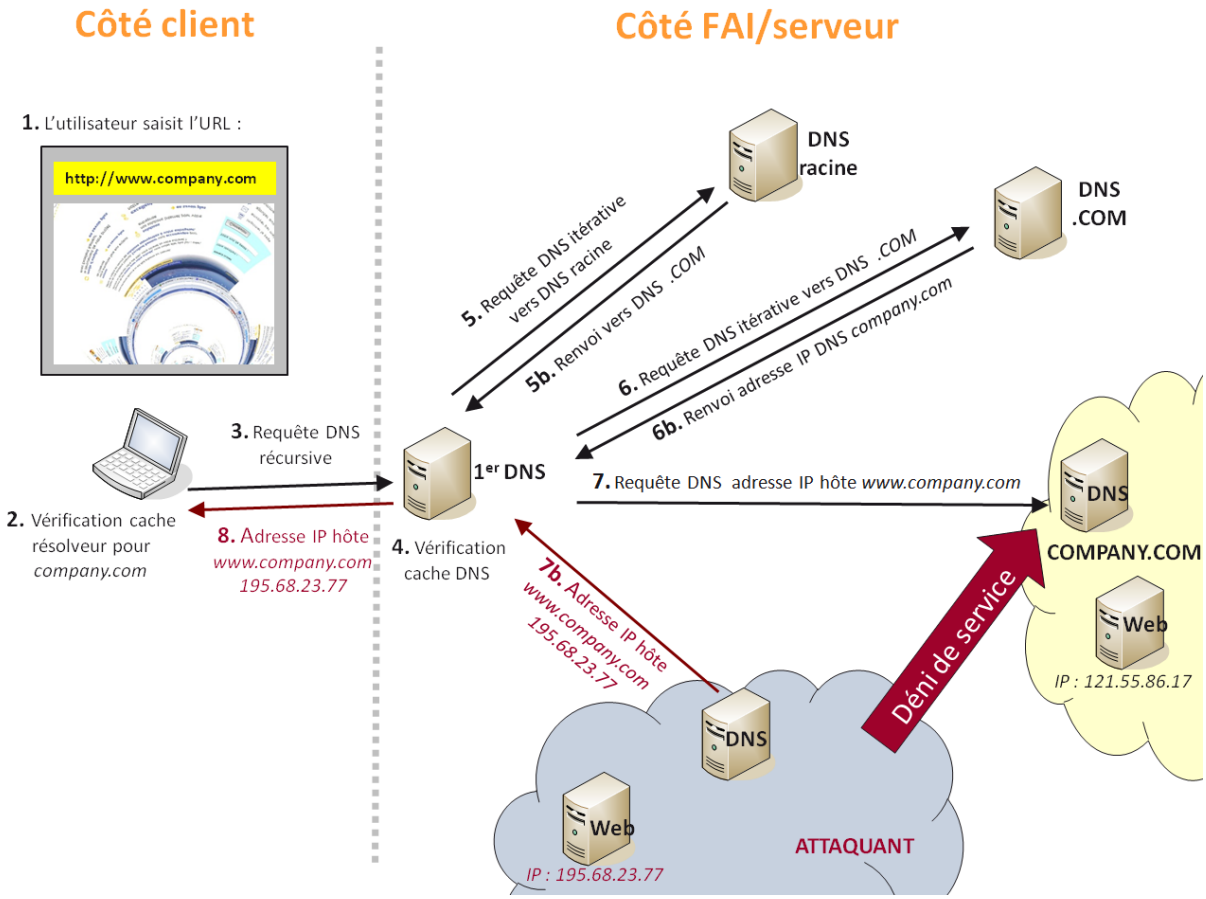


FIGURE 4.4 – Exemple d'attaque de pharming combinant des techniques de *DNS cache poisoning* et de *spoofing* DNS

4.1.3 Méthodes de prévention/détection du pharming

Les attaques de pharming s'appuient sur les faiblesses du protocole DNS et/ou les zones de configuration associées, qu'elles se situent côté client ou côté réseau FAI/serveur web.

A son origine - à l'image des protocoles associés à IPv4 -, le protocole DNS a été conçu sans aucune notion de sécurisation. En effet, durant les années 80, la priorité était au développement de protocoles efficaces et performants pouvant répondre aux besoins de croissance des réseaux [AFN10].

Depuis, de par le développement de l'utilisation du DNS - désormais inéluctable aux échanges sur Internet - ainsi que les nombreuses zones de vulnérabilités associées, diverses solutions de sécurité visant à améliorer le DNS et/ou protéger des attaques de pharming ont été proposées.

4.1.3.1 DNSSEC : une solution côté réseau FAI/serveur web

La proposition majeure associée à la sécurité du DNS concerne le développement des extensions de sécurité DNSSEC (pour *Domain Name System SECurity*) dont les bases fondatrices sont définies dans les RFC 4033, 4034 et 4035[1].

4.1.3.1.1 Fonctionnement général : DNSSEC [AFNb] [Gue06] vise à sécuriser l'architecture de nommage DNS dans sa globalité, tant au niveau des échanges effectués entre serveurs DNS, que des différents niveaux d'arborescence non-terminaux (cf. figure 4.1).

Ces extensions de sécurité reposent sur des mécanismes de signatures (générées à l'aide de chiffrement asymétrique) pour assurer l'authenticité et l'intégrité (c.-à-d. les informations n'ont pas été

1. Liste non exhaustive.

Tableau 4.2 – Exemples d'enregistrements DNS
Resource Record set (RRset)

RRset	FQDN	Type	Classe	TTL	Adresse IP
1	www.google.fr	A	IN	86400	209.85.148.103
		A	IN	86400	209.85.148.99
		A	IN	86400	209.85.148.105
2	www.it-sudparis.eu	A	IN	43200	157.159.11.8

altérées durant leur transfert) des enregistrements DNS [AFN10].

Supposons le cas d'un client A (ou plus exactement son résolveur) qui souhaite connaître l'adresse IP associée au serveur web B, afin de répondre à la demande d'une application cliente. Le mode de fonctionnement à minima des échanges DNS en utilisant DNSSEC vise à être le suivant :

1. Au préalable, le serveur DNS autoritaire de la zone B a publié sa clé publique.
2. Au préalable, le serveur DNS autoritaire de la zone B a publié ses informations DNS (c.-à-d. ses couples d'adresses IP/FQDN) accompagnées d'une signature. Cette signature a été générée localement à partir de la clé privée du serveur DNS B et des informations à publier.
3. En retour de la requête DNS émise par le résolveur du client, une réponse DNS est reçue. Celle-ci contient soit l'information demandée (c.-à-d. un couple d'adresses IP/FQDN) accompagnée de sa signature, soit un message d'erreur.
4. Si la réponse DNS contient l'information demandée : la machine DNSSEC validante côté client (c.-à-d. la dernière machine qui implémente DNSSEC : soit le serveur DNS du FAI auquel est rattaché le client, soit le résolveur demandeur) se charge alors de vérifier que l'association adresses IP/FQDN retournée a bien été émise par le serveur DNS B[1]. En effet, par l'utilisation de la clé publique publiée par B (et diffusée dans l'arborescence DNS), la signature accompagnant les informations est vérifiée.
5. Si la réponse DNS contient un message d'erreur : la machine DNSSEC validante côté client peut vérifier que ce message a bien été émis par le serveur DNS B (via la vérification de la signature). Si c'est le cas, il peut en supplément s'assurer que le domaine interrogé référence ce type d'enregistrement.

A noter que, comme précédemment (c.-à-d. avec l'utilisation du DNS seul, sans extensions de sécurité), les informations DNS d'une zone sont diffusées à l'ensemble des serveurs DNS d'Internet, au travers des échanges réalisés entre serveurs DNS parents.

Avec l'apport de DNSSEC, les échanges d'informations DNS entre ces serveurs DNS parents se font désormais de manière sécurisée.

En supplément, afin d'éviter la création/utilisation de fausses clés DNSSEC par un attaquant, un serveur DNS parent signe (à l'aide de sa propre clé privée) chacune des clés publiques des zones qui lui sont rattachées (selon le même procédé qu'expliqué précédemment). Par conséquent, pour une sécurité optimale, chaque machine validante DNSSEC recevant des enregistrements DNS se doit de vérifier leur appartenance à la zone annoncée, via l'utilisation de la clé publique du serveur DNS parent associé.

4.1.3.1.2 Précisions techniques : L'implémentation de ces extensions de sécurité DNSSEC entraîne l'apparition de notions/enregistrements DNS additionnels [Gue06]. Parmi les plus connus, on peut citer notamment les enregistrements RRset, RRSIG, DNSKEY, DS, NSEC (ou NSEC3) et les clés KSK, ZSK :

Pour limiter la quantité d'informations nouvellement stockées par l'apport de DNSSEC, les enregistrements RR de même type, même nom d'hôte et même classe sont regroupés au sein d'un ensemble d'enregistrements nommé RRset (pour *Resource Record set*, cf. tableau 4.2).

Ainsi le serveur DNS autoritaire d'une zone signe un RRset, en lieu et place de signer chaque RR. Les signatures résultantes sont alors stockées dans des enregistrements nommés RRSIG (pour *Resource*

1. Cette vérification est effectuée avant transmission de la réponse au résolveur, si la dernière machine DNSSEC validante est le DNS du FAI.

Record SIGnature). Il y a autant d'enregistrements RRSIG que de RRset signés avec une clé privée donnée. Les signatures sont générées à partir d'un algorithme de chiffrement asymétrique de type RSA ou DSA, combiné à une fonction de hachage de type MD5, SHA.

La clé publique du serveur DNS autoritaire d'une zone est nommée ZSK (pour *Zone Signing Key*). Elle est publiée, dans le serveur DNS autoritaire de la zone, via un enregistrement DNSKEY (pour *Domain Name System KEY*). Précisons qu'un serveur DNS autoritaire de zone peut utiliser plusieurs couples de clés privées/clés publiques. Dans ce cas, chaque clé publique fait l'objet d'un enregistrement DNSKEY et chaque RRset doit être signé par chacune des clés privées de la zone.

Le(s) enregistrement(s) DNSKEY sont également référencés dans le serveur DNS de la zone parente, via un(des) enregistrement(s) DS (pour *Delegation Signer*). Le serveur DNS de la zone parente signe les enregistrements DNSKEY de ses zones filles avec une clé nommée KSK (pour *Key Signing Key*).

Enfin, dans le cas où la requête DNS est un succès, le demandeur reçoit le RRset de l'hôte recherché, accompagné de la signature RRSIG associée. Dans le cas idéal, la machine validante DNSSEC (c.-à-d. le demandeur ou le serveur DNS du FAI) effectue une vérification en 3 étapes : 1/ elle récupère la clé DNSKEY de la zone émettrice, 2/ elle s'assure de la légitimité de la DNSKEY récupérée en contactant le serveur DNS parent de la zone, et 3/ si DNSKEY est légitime, elle l'utilise afin de vérifier l'authenticité et l'intégrité du RRset reçu.

A contrario, en cas d'échec de la requête DNS, le demandeur reçoit un enregistrement NSEC ou NSEC3 (pour *Next-SECure record*) en guise de message d'erreur. Contrairement au simple code d'erreur reçu avec l'utilisation du DNS sans extension de sécurité, le message NSEC ou NSEC3 donne des indications sur la légitimité du répondant et l'existence de l'hôte demandé.

La figure 4.5 montre une vue d'ensemble des liens entre enregistrements DNSSEC (en pointillés sur le schéma), ainsi qu'un exemple de requête DNS/DNSSEC (en traits continus sur le schéma).

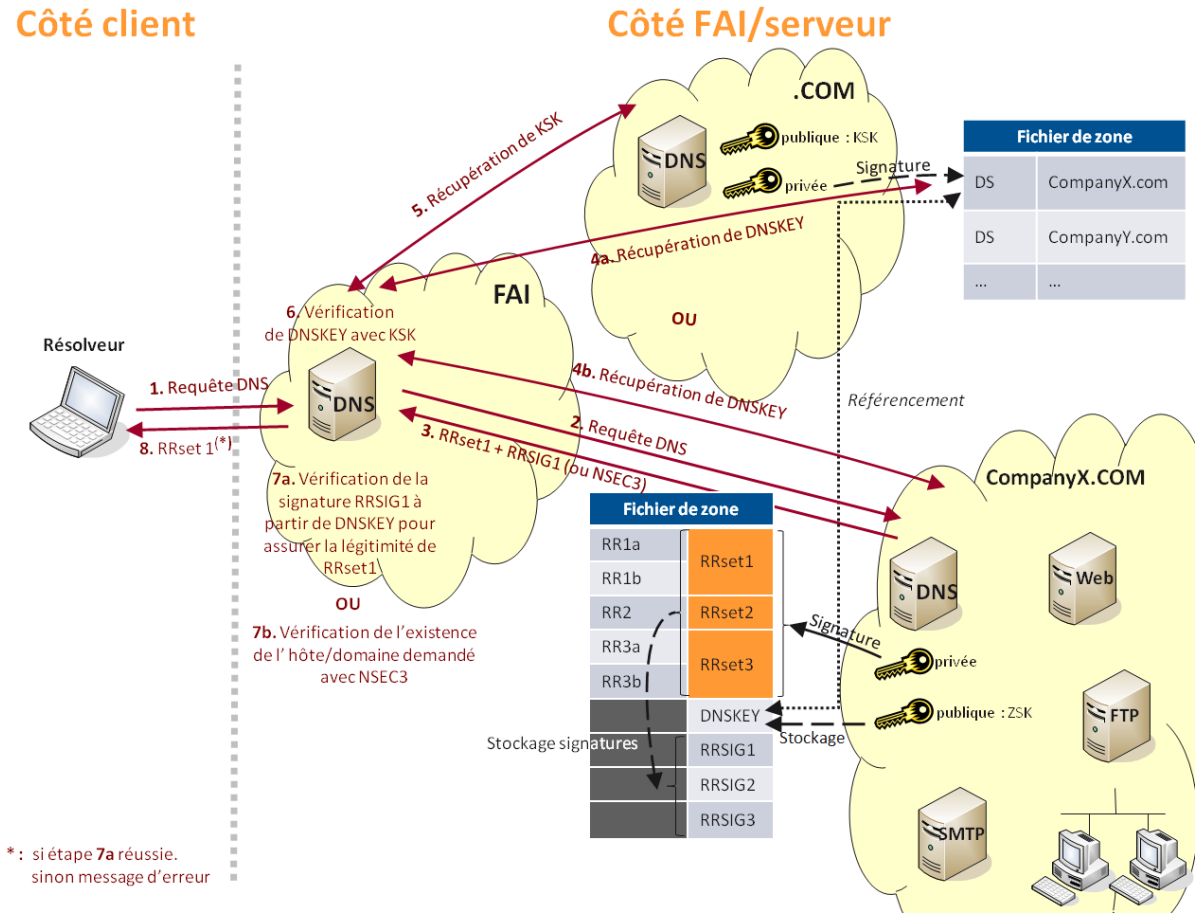


FIGURE 4.5 – Exemple de requête DNS avec extensions de sécurité DNSSEC

4.1.3.1.3 Point sur le déploiement DNSSEC : La vulnérabilité découverte par Dan Kaminsky [Sch08] durant l'été 2008, a été le point d'orgue au déploiement des extensions de sécurité DNSSEC, aux plus hauts niveaux de l'arborescence DNS. La faille découverte - assimilée à de l'empoisonnement de cache DNS - a révélé à tous le caractère d'urgence associé à la mise en œuvre de DNSSEC. Depuis, les serveurs TLD ont donc progressivement commencé à déployer ces extensions de sécurité. Parmi les plus connus, on peut notamment citer les TLD .ORG, .GOV, .UK, .EDU, .FR, .COM et .NET.

Au 15 Juillet 2010, une avancée majeure a été réalisée : le déploiement de DNSSEC a été finalisé au niveau des 13 serveurs racines [IV10].

Néanmoins, malgré ces efforts, l'ensemble de l'arborescence DNS est loin d'avoir migré. Il faut en effet que l'ensemble des acteurs d'Internet (c.-à-d. FAI, bureaux d'enregistrements, entreprises, clients, etc.) s'y mettent à leur tour.

Côté client/entreprise, Microsoft a par ailleurs annoncé la mise en œuvre de fonctionnalités DNSSEC dans les systèmes d'exploitation Windows 7 et Windows Server 2008 R2 [Mic10]. Toutefois, la mise en œuvre des extensions de sécurité côté client passe d'abord par la mise en œuvre côté réseau.

Par conséquent, aujourd'hui on ne peut que trop rarement s'attendre à voir DNSSEC déployé jusqu'au résolveur du particulier. A défaut, la vérification des enregistrements DNSSEC se voit donc assurée par le serveur DNS le plus proche, à savoir : celui du FAI.

4.1.3.1.4 Limitations : Il est important de préciser ce que DNSSEC ne fait pas [AFN10] : on peut notamment citer qu'il n'assure pas la confidentialité des enregistrements DNS, puisque ceux-ci sont uniquement signés. Il ne vise pas non plus à se substituer aux mécanismes d'authentification actuellement utilisés pour sécuriser les transactions électroniques (p.ex. via HTTPS). Il ne peut rien contre le phishing, une grande partie des attaques effectuées côté client ou les attaques réalisées sur le serveur web. Il ne peut non plus garantir d'une éventuelle compromission des enregistrements DNS en amont de leur insertion dans le serveur DNS autoritaire.

De plus, au travers de l'ajout d'extensions qui visent à introduire davantage de sécurité (p.ex. la délivrance d'un message d'erreur consistant et explicite qui permet d'appréhender l'existence de l'hôte demandé), DNSSEC a également introduit d'autres vulnérabilités et/ou problèmes [BM10].

On peut notamment citer le cas de l'enregistrement NSEC. Celui-ci peut permettre à un attaquant de prendre connaissance des hôtes du domaine légitime. Pour contrer ce problème, un nouvel enregistrement NSEC3 a donc été développé, afin de masquer les noms des hôtes.

On peut également parler des problèmes associés au chiffrement introduit, p.ex. la lourdeur de gestion des clés dans une arborescence aussi conséquente (c.-à-d. assurer une cohérence dans les clés utilisées, gérer leur renouvellement), la bonne gestion des interactions serveur autoritaire - serveur parent lors des mises à jour de clés (de par la forte relation entre les enregistrements DS et DNSKEY), ou encore l'absolue nécessité de diffuser rapidement les mises à jour de clés/signatures dans l'arborescence DNS.

4.1.3.2 Le réseau client reste une cible privilégiée

Auparavant les principales techniques de sécurisation associées au DNS résidaient davantage en des techniques de supervision (p.ex. remontées d'alertes spécifiques, mises à jour logicielles) et des mesures de continuité de service [AFN09]. L'ensemble des zones de vulnérabilités, et attaques associées, exposées en section 4.1.2 démontre bien les lacunes sécuritaires du protocole DNS.

De par la structure DNS, il apparaît que la compromission des informations DNS côté client est certainement moins visible que côté réseau. En effet, les serveurs DNS des FAI et des plus hauts niveaux d'arborescence sont sous haute surveillance. Une attaque sur le service recherché (c.-à-d. le serveur web et/ou son domaine) peut alors être préférée mais, là encore, elle deviendra rapidement visible. Par conséquent, le réseau client est donc une cible privilégiée, même si plus contraignante car la compromission à large échelle doit être effectuée en une multitude de points.

L'apport de DNSSEC est donc bien une nécessité incontournable visant à éradiquer/minimiser les compromissions des enregistrements DNS au sein d'Internet. Toutefois DNSSEC ne deviendra réellement efficace que lorsque l'ensemble de la chaîne de liaison DNS (du client jusqu'au serveur web visité) l'aura déployé. De plus, il est indispensable que les informations DNSSEC soient correctement

publiées/renouvelées par les serveurs autoritaires, et que les mises à jour soient rapidement diffusées à l'ensemble de la chaîne de liaison. Dans le cas contraire, des indisponibilités des services recherchés sont à prévoir.

Côté client, tant que le résolveur n'aura pas mis en œuvre DNSSEC, aucune protection supplémentaire ne sera apportée si ce n'est la garantie que les données DNS n'auront pas été corrompues entre le DNS du FAI et celui de la zone du serveur web visité. Par ailleurs, même en cas de déploiement sur le résolveur client, DNSSEC ne protège pas de l'ensemble des attaques perpétrées côte client (p.ex. les ajouts de fausses entrées IP/FQDN statiques sur le poste client, telles qu'exposées en section 4.1.2.1).

Enfin, précisons également que l'utilisation du protocole HTTPS est insuffisante pour se prémunir des attaques de pharming. En effet, l'attaquant peut également corrompre les paramétrages de sécurité effectués au niveau du poste client, pour altérer la vérification et/ou l'alerte faite à l'utilisateur en cas de certificat non valide. Certaines propositions exposées en section 4.1.3.3 s'intéressent d'ailleurs à cette problématique.

4.1.3.3 Des propositions côté client

Les propositions qui ciblent le côté client indiquent plusieurs vecteurs de sécurisation des informations DNS et/ou de protection envers les sites contrefaits. Elles sont étayées par deux types d'approches : des techniques amont qui visent à restreindre le champs d'accès des attaquants, ou des techniques de détection utilisées lors de la navigation web. Ces dernières peuvent aussi bien travailler sur les informations de connexion (adresse IP, certificat SSL, etc.) que sur le contenu de la page web, à la recherche de comportements anormaux.

4.1.3.3.1 Restreindre le champ d'accès des attaquants :

Poste client : De manière générale, l'ensemble des propositions qui s'intéressent au poste client visent à mieux contrôler le contenu des pages webs visitées (c.-à-d. mieux protéger le navigateur web) ou les vecteurs qui y conduisent. En effet, de par la forte exploitation des applets Java (pour obtenir les adresses IP du client) et des scripts malveillants (incorporés au code HTML des pages via des scripts Javascript) pour diffuser les compromissions DNS côté client, les études se rejoignent sur le besoin de mieux contrôler l'accès et l'exécution de ces contenus. Pour exemple, Barth et al. [BFSB10] ont démontré que 88% des 25 extensions couramment utilisées par les utilisateurs de Firefox et testées dans leur étude, donnent davantage de points d'accès à la configuration du système qu'elles n'en nécessitent pour fonctionner. Par conséquent, elles introduisent autant de brèches potentielles pour une éventuelle compromission par un attaquant. Les auteurs recommandent donc de revoir la plateforme utilisée pour la création des extensions en accordant une meilleure place à la sécurité (c.-à-d. en n'accordant à une extension que les droits d'accès strictement nécessaires à son bon fonctionnement).

Pour contrer et/ou détecter les corruptions DNS effectuées côté client, Ollmann [Oll05] propose de désactiver les fonctionnalités qui autorisent l'affichage de contenu dynamique et/ou personnalisé (p.ex. ActiveX, fenêtres pop-up, cookies, etc.). Plus spécifiquement, Stamm et al. [SRM07] recommandent de restreindre l'accès aux fichiers de configuration système (c.-à-d. qui gouvernent la configuration IP/DNS) aux applets qui sont signées. Dans la même lignée, des extensions de sécurité pour navigateurs sont désormais disponibles afin de mieux contrôler les exécutions automatiques de scripts incorporés dans les pages webs. On peut par exemple citer des extensions comme NoScript [Inf] et Sabre [DG09], disponibles pour Firefox, qui aident à mieux contrôler les Javascripts exécutés au travers de listes blanches.

Une autre proposition de Stamm et al. [SRM07] réside en l'exclusion des scripts, issus d'un site web tiers (p.ex. des publicités), dans l'affichage de la page web visitée par l'utilisateur. L'étude de Karlof et al. [KSTW07] qui porte sur les connexions SSL, rejoint cette idée. Elle propose le renforcement des SOP (pour *Same Origin Policy*) par l'insertion d'indicateurs, révélateurs de la cohérence des informations de sécurité obtenues lors de l'établissement de la connexion sécurisée. Les SOP, généralement utilisés par les codes Javascript, permettent d'autoriser sans restriction l'exécution de scripts issus d'un domaine donné au sein d'une page web de même origine (c.-à-d. de même domaine). Ainsi dans la proposition évoquée, un script ne pourra être exécuté sans restriction que s'il y a concordance du nom de domaine et

de l'indicateur inséré. Ce dernier peut refléter, par exemple, une incohérence entre les noms de domaine présents dans l'URL et le champ CN (pour *Common Name*) du certificat SSL délivré par le serveur web.

Réseau client : De manière globale, les études considérées ([JBB+09], [SRM07] et [Oll05]) préconisent d'utiliser des outils de diagnostic réseau disponibles en ligne ou installés dans le réseau client, afin de détecter d'éventuelles compromissions (p.ex. des outils de scan pour détecter d'éventuels malwares installés sur le poste client, analyser régulièrement les paramètres DNS, etc.).

Routeur de bordure : L'étude de Stamm et al. [SRM07] propose d'abandonner les mots de passe par défaut actuellement pré-configurés (p.ex. *(vide)*, admin, etc.) pour le compte administrateur qui sert à la configuration IP/DNS du routeur. Les auteurs proposent de le remplacer, par exemple, par le numéro de série de l'équipement afin de rendre son identification moins facile.

4.1.3.3.2 Détecter l'attaque lors de la navigation web :

Des extensions de sécurité pour navigateurs peuvent être utilisées pour valider les informations de sécurité reçues par Internet, lors de la navigation web. On peut par exemple citer Perspectives [WAP08] pour Firefox, ou HTTPSLock [FC10] qui visent à vérifier les certificats SSL auprès d'une liste blanche (p.ex. via un réseau notaire).

Cao et al. [CHL08] proposent, quant à eux, une méthode de détection des sites de pharming/phishing basée sur une comparaison de l'adresse IP du domaine visité auprès d'une liste blanche d'adresses IP légitimes. Cette méthode s'appuie sur un principe de staticité des adresses IP associées aux pages de login.

Basés sur ce même principe, Bin et al. [BQX10] préfèrent se focaliser sur la détection de la saisie des numéros de cartes bancaires au sein des pages webs. Leur système s'appuie sur une base de connaissances préalable des noms de banques, adresses IP et plages de numéros de cartes bancaires associés. Ainsi, dès lors qu'un utilisateur commence à saisir un numéro de carte bancaire dans une page web, le système en déduit le nom d'une banque. Il émet alors automatiquement une requête DNS inverse, afin de s'assurer que l'adresse IP du domaine actuellement visité est bien incluse dans les adresses IP connues pour la banque interrogée.

Un système d'analyse passive des informations DNS est également proposé par Bilge et al [BKKB11] afin de détecter les domaines contrefaits. Davantage ciblé sur la détection de sites de phishing, certains critères étudiés peuvent toutefois également s'appliquer à la détection du pharming. Par exemple, le nombre d'adresses IP retournées par une requête DNS sur le domaine visité, les résultats de requêtes reverse DNS, etc.

Enfin de manière plus générale, Ollman [Oll04] préconise d'étendre l'utilisation des moyens habituels de détection proposés pour le poste client (p.ex. pare-feu, anti-spyware).

4.2 Analyse et comparaison des pages webs : deux grandes catégories de méthodes pour la comparaison de documents HTML

Le Chapitre 3 a mis en évidence l'intérêt de l'analyse du code source HTML des pages webs, dans la différenciation des sites légitimes et contrefaits. Basé sur ce constat et notre recherche d'une technique de détection des sites webs contrefaits de pharming, nous nous sommes naturellement intéressés à l'analyse et la comparaison du contenu des pages webs.

Pour les raisons évoquées ultérieurement dans ce document (cf. section 5.2.2), nous avons choisi de focaliser notre comparaison de pages webs sur la comparaison de codes sources HTML. En effet, notre piste de recherche vise à confronter deux documents HTML afin de déterminer un degré de similitude/divergence des deux contenus étudiés. Dans notre étude, cette comparaison a pour vocation à être utilisée pour l'identification et/ou la différenciation de pages légitimes vs. des pages contrefaites.

Un document HTML (c.-à-d. le code source d'une page web) est avant tout un fichier texte, organisé selon une structure spécifique (cf. section 2.2.2.1). Deux grandes catégories de méthodes de comparaisons s'offrent alors : les méthodes de comparaison de textes qui travaillent sur le contenu du document de manière globale, ou les méthodes de comparaison qui se basent sur la structure du document étudié (nommée DOM).

Dans la suite de ce chapitre, nous nous intéressons donc à décrire ces deux grandes catégories de méthodes de comparaison, avec une attention particulière sur les méthodes de comparaisons de texte. Celles-ci se sont en effet révélées être mieux adaptées au contexte de notre étude (cf. explications détaillées en section 5.2.2.3).

4.2.1 Les méthodes de comparaison de textes

Les méthodes de comparaison de textes se décomposent entre différentes catégories/familles d'algorithmes (cf. figure 4.6) :

- Les algorithmes d'alignement de chaînes
- Les algorithmes de recherche de sous-chaînes
- Les algorithmes de mesure de similarité

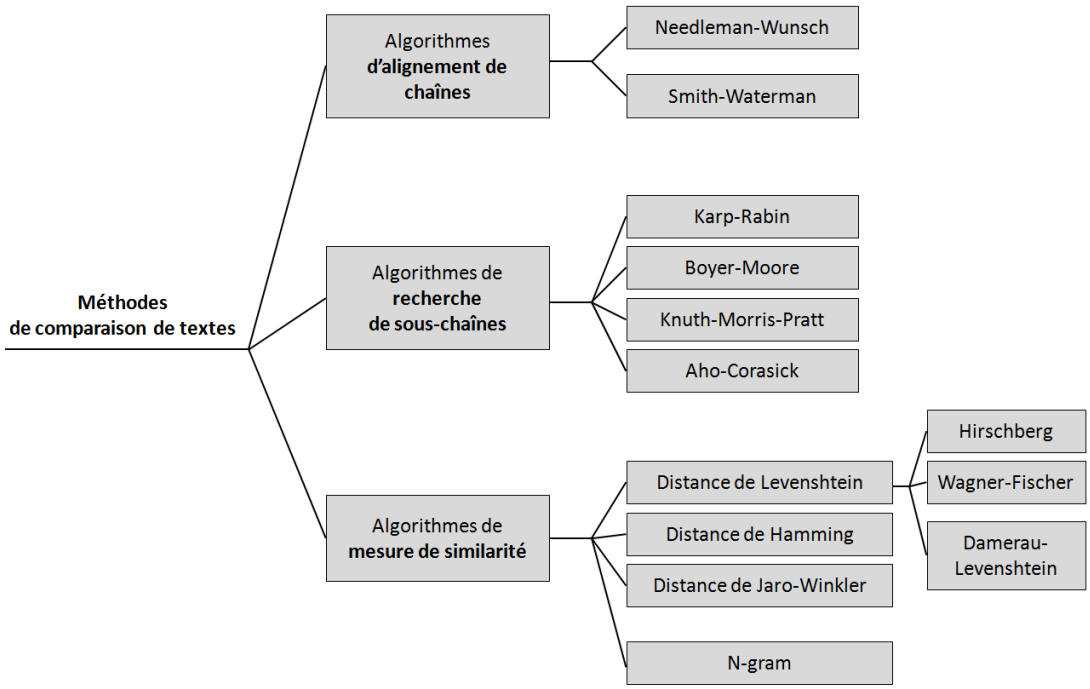


FIGURE 4.6 – Taxonomie des méthodes de comparaison de textes

Les algorithmes de comparaisons de textes étant particulièrement nombreux, chaque catégorie exposée ci-après ne sera illustrée qu'au travers du détail d'un à deux algorithmes maximum.

4.2.1.1 Les algorithmes d'alignement de chaînes

Cette famille d'algorithmes est typiquement utilisée en génétique, afin de comparer deux séquences (p.ex. ADN). Les algorithmes Needleman-Wunsch [NW70] et Smith-Waterman [SW81] en sont les exemples les plus connus. Ils font tous deux partie des techniques de programmation dynamique, dont le mode de fonctionnement vise à découper un problème en sous-problèmes, afin d'appliquer un traitement optimal à chacun d'entre-eux. La programmation dynamique s'appuie sur le principe que des sous-problèmes traités de façon optimale permettront d'aboutir à une solution optimale pour le problème global [Lik05].

Prenons l'exemple de l'**algorithme Needleman-Wunsch**, moins restrictif que l'algorithme Smith-Waterman. Il fonctionne sur le principe suivant : pour deux chaînes de caractères à comparer (x, y), on définit une matrice de similarité (à chaque colonne on associe un caractère de la chaîne x, et à chaque ligne on associe un caractère de la chaîne y). A cette matrice de similarité, on associe une pénalité de trou (p) ≤ 0. Cette dernière est utilisée pour initialiser les valeurs des premières ligne et colonne de la matrice (nommées *Coût*), selon la règle suivante :

$$\begin{aligned} Co\hat{u}t(i) &= Co\hat{u}t(i-1) + p \\ Co\hat{u}t(j) &= Co\hat{u}t(j-1) + p \end{aligned} \tag{4.1}$$

où *(i)* est le numéro de ligne et *(j)* est le numéro de colonne.

Le score de chaque cellule (C) de la matrice est ensuite déterminé selon les scores de ses cellules immédiatement adjacentes : (Cdiag(i-1,j-1)) pour cellule diagonale supérieure gauche, (Cup(i-1, j)) pour cellule supérieure et (Cleft(i,j-1)) pour cellule gauche, selon la formule suivante :

$$\text{Score C(i,j)} = max \begin{cases} \text{Score C}diag\text{(i-1,j-1)} + S(x(i), y(i)) \\ \text{Score C}up\text{(i-1,j)} + p \\ \text{Score C}left\text{(i,j-1)} + p \end{cases} \tag{4.2}$$

où *(i)* est le numéro de ligne, *(j)* est le numéro de colonne,
et S est la valeur de substitution des deux caractères étudiés selon une matrice pré-définie (p.ex. la matrice BLOSUM62 [oK]).

Puis, en partant de la fin de la matrice (c.-à-d. depuis la dernière cellule en bas à droite), on détermine le meilleur alignement (c.-à-d. qui permet d'aboutir à l'alignement d'un maximum de caractères communs) en remontant les cellules jusqu'au point d'origine de la matrice. Pour remonter les cellules, on choisit toujours la cellule de score le plus élevé parmi les 3 cellules immédiatement adjacentes (c.-à-d. cellules gauche, droite, ou diagonale supérieure gauche). A chaque déplacement latéral correspondra l'insertion d'un trou dans la chaîne y, tandis qu'à chaque déplacement vertical correspondra l'insertion d'un trou dans la chaîne x.

La figure 4.7 illustre un exemple de l'alignement des chaînes de caractères *CHAINE* et *CRAIE* en utilisant la matrice de substitution BLOSUM62 (pour plus de détails, cf. section 5.2.2.3.1), et une pénalité de trou à 0.

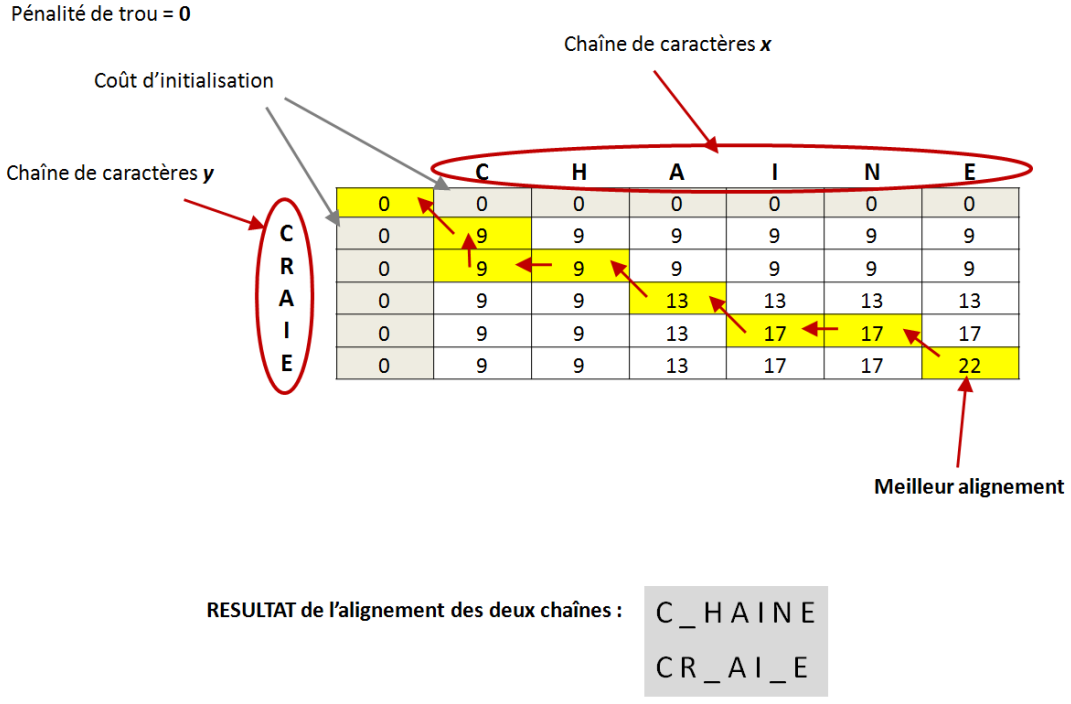


FIGURE 4.7 – Exemple de l'alignement de 2 chaînes de caractères, avec l'algorithme Needleman-Wunsch

L'**algorithme Smith-Waterman** présente un fonctionnement très similaire à l'algorithme Needleman-Wunsch. Néanmoins, il vise à s'appliquer plutôt sur des sous-parties d'une chaîne de caractères vs. une chaîne de caractères complète.

Le projet B.A.B.A (pour *Basic Algorithms of Bioinformatics Applet*) [Sou]) est un exemple d'implémentation de ces deux algorithmes. Il permet de tester en ligne les deux algorithmes d'alignement de chaînes énoncés dans cette section.

4.2.1.2 Les algorithmes de recherche de sous-chaînes

Les algorithmes de recherche de sous-chaînes ont pour but de rechercher une chaîne de caractères à l'intérieur d'une autre chaîne de caractères, dite de référence. Bien connues dans le domaine de la sécurité informatique, les attaques de type *brute force* en sont une application typique : par le test successif de toutes les combinaisons possibles, elles visent à retrouver un mot de passe, une clé de chiffrement, etc.

Les algorithmes de recherche de sous-chaînes sont particulièrement nombreux [CL97]. Parmi les plus connus, on peut citer les algorithmes Karp-Rabin [KR87], Boyer-Moore [BM77], Knuth-Morris-Pratt [KMP77], Aho-Corasick [AC75], etc. Ils traitent la chaîne de caractères à trouver de gauche à droite, ou de droite à gauche, en s'appuyant sur des dictionnaires, des fonctions de hachage (représentative de la base de caractères possibles), etc.

Prenons l'exemple de l'**algorithme Boyer-Moore**, considéré comme l'un des plus efficaces de sa catégorie. Il a la particularité de traiter les chaînes de caractères de droite à gauche. Si l'on considère un texte de référence (nommé T) constitué de 25 caractères, au sein duquel on doit rechercher une chaîne de 5 caractères (nommée C, dont la longueur exprimée en caractères est nommée X), l'algorithme se déroule de la manière suivante : si la dernière lettre de C ne correspond pas à la dernière lettre des X premiers caractères de T, on fait un saut de X caractères dans T. On regarde alors la correspondance entre la dernière lettre de C et la dernière lettre du mot actuellement étudié dans T. Si cela ne correspond pas, on fait un nouveau saut de X caractères dans T, et ainsi de suite jusqu'à atteindre la fin de T. Dès lors qu'on obtient une correspondance d'un dernier caractère (entre C et le mot actuellement étudié dans T), on remonte au caractère précédent et ainsi de suite. Dès qu'on détecte une non-concordance au sein des deux mots étudiés, on passe au mot suivant de T.

A noter que la taille du saut à effectuer dans le texte de référence (indiquée par X dans le déroulement simplifié ci-dessus) est en réalité déterminée à l'issue d'une phase de pré-apprentissage de l'algorithme, effectuée sur les deux chaînes à comparer.

La figure 4.8 illustre un exemple de résultat obtenu avec l'algorithme Boyer-Moore. Elle a été établie à partir de l'implémentation proposée par Charras [CL97].

4.2.1.3 Les algorithmes de mesure de similarité

Les algorithmes de mesure de similarité ont pour but de calculer la distance entre deux éléments de même type (typiquement deux chaînes de caractères, deux documents, etc.). Cette distance est représentative du nombre d'opérations minimum nécessaires pour passer d'une chaîne de caractères à l'autre.

Les principaux algorithmes de mesure de similarité sont : la Distance de Levenshtein [Lev66] et ses dérivés (p.ex. Wagner-Fischer [WF74], Hirschberg [Hir75], Damerau [Dam64]-Levenshtein) - plus connus sous la terminologie de Distance d'édition -, la Distance de Hamming [Ham50] ou encore la Distance de Jaro-Winkler [Win06].

Prenons l'exemple de l'**algorithme de Distance de Levenshtein**, qui adresse davantage d'opérations que ses pairs. Il fait partie de la famille des techniques de programmation dites dynamiques (cf. section 4.2.1.1). Cet algorithme prend en compte trois types d'opérations : l'ajout, la suppression et la modification (également appelée substitution) de texte.

La base utilisée est similaire à celle vue pour l'algorithme Needleman-Wunsch (cf. section 4.2.1.1), à savoir la définition d'une matrice de similarité. Soit deux chaînes de caractères à comparer x, y. A chaque colonne, on associe un caractère de la chaîne x, et à chaque ligne on associe un caractère de

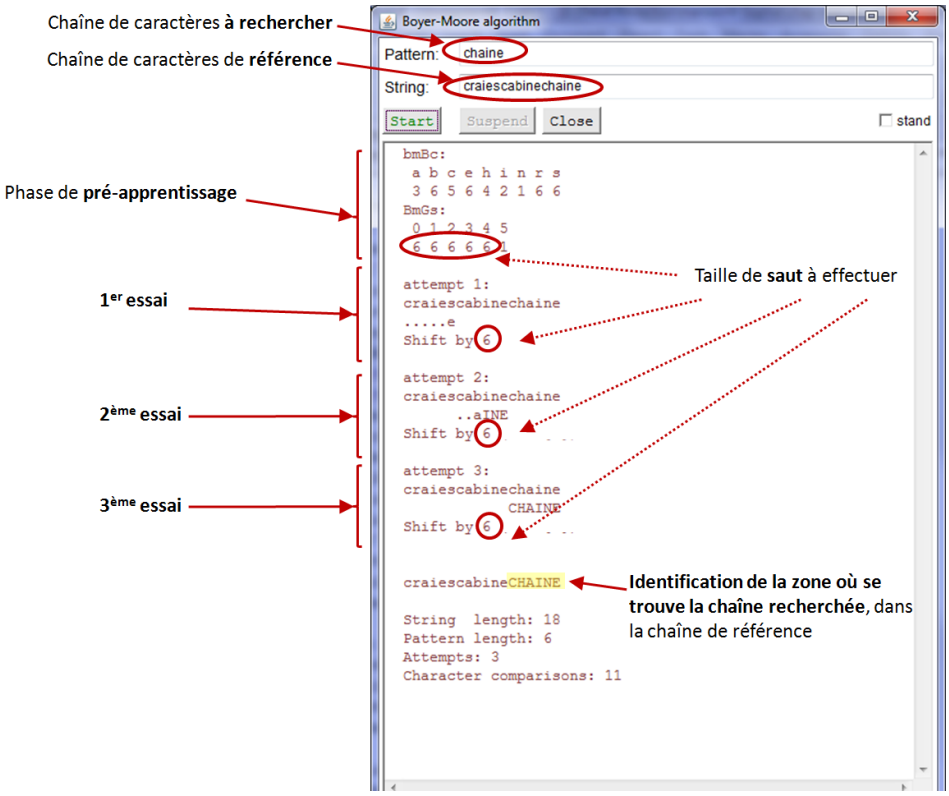


FIGURE 4.8 – Exemple de recherche de sous-chaîne, établi avec l'implémentation [CL97] de l'algorithme Boyer-Moore

la chaîne *y*. Puis, on définit les 3 pénalités choisies pour l'ajout *Pajout*, la modification *Pmodif* et la suppression *Psuppr*. La matrice de coût est ensuite initialisée avec les règles suivantes :

- Le coût d'ajout est indiqué sur la 1ère ligne horizontale. Il est établi à partir de *Pajout* (p.ex. si *Pajout*=1, le coût d'ajout correspondant est : 1,2,3,4 etc.)
- Le coût de suppression est indiqué sur la 1ère ligne verticale. Il est établi à partir de *Psuppr* (p.ex. si *Psuppr*=1, le coût de suppression correspondant est : 1,2,3,4 etc.)
- Chaque cellule restante est initialisée en fonction de la correspondance des lettres de sa colonne et de sa ligne. Cette valeur ainsi attribuée est appelée : Coût de modification (*Coût modif*) de la cellule. S'il y a correspondance entre les lettres, le coût de modification est de 0. Sinon il est égal à *Pmodif*.

Le score de chaque cellule (C) de la matrice est ensuite déterminé selon les scores de ses cellules immédiatement adjacentes : (Cdiag(i-1,j-1)) pour cellule diagonale supérieure gauche, (Cup(i-1, j)) pour cellule supérieure et (Cleft(i,j-1)) pour cellule gauche, selon la formule suivante :

$$\text{Score C(i,j)} = min \begin{cases} \text{Score C}diag\text{(i-1,j-1)} + \text{Coût modif(i-1,j-1)} \\ \text{Score C}up\text{(i-1,j)} + Psuppr \\ \text{Score C}left\text{(i,j-1)} + Pajout \end{cases} \tag{4.3}$$

où *(i)* est le numéro de ligne et *(j)* est le numéro de colonne.

Puis, en partant de la fin de la matrice (c.-à-d. depuis la dernière cellule en bas à droite), on détermine le chemin des opérations (ajouts, suppressions, modifications) en remontant les cellules jusqu'au point d'origine de la matrice. Pour remonter les cellules, on choisit toujours la cellule de score le plus faible parmi les 3 cellules immédiatement adjacentes (c.-à-d. cellules gauche, droite, ou diagonale supérieure gauche). A chaque changement de valeur au cours du chemin des opérations, correspond l'ajout, la modification ou la suppression d'un caractère.

La figure 4.9 illustre un exemple de calcul de Distance de Levenshtein pour les mots *CHAINE* et *CRAIE*, réalisé à partir de l'implémentation de Kleiweg [Kle], avec des pénalités d'opération établies à 1.

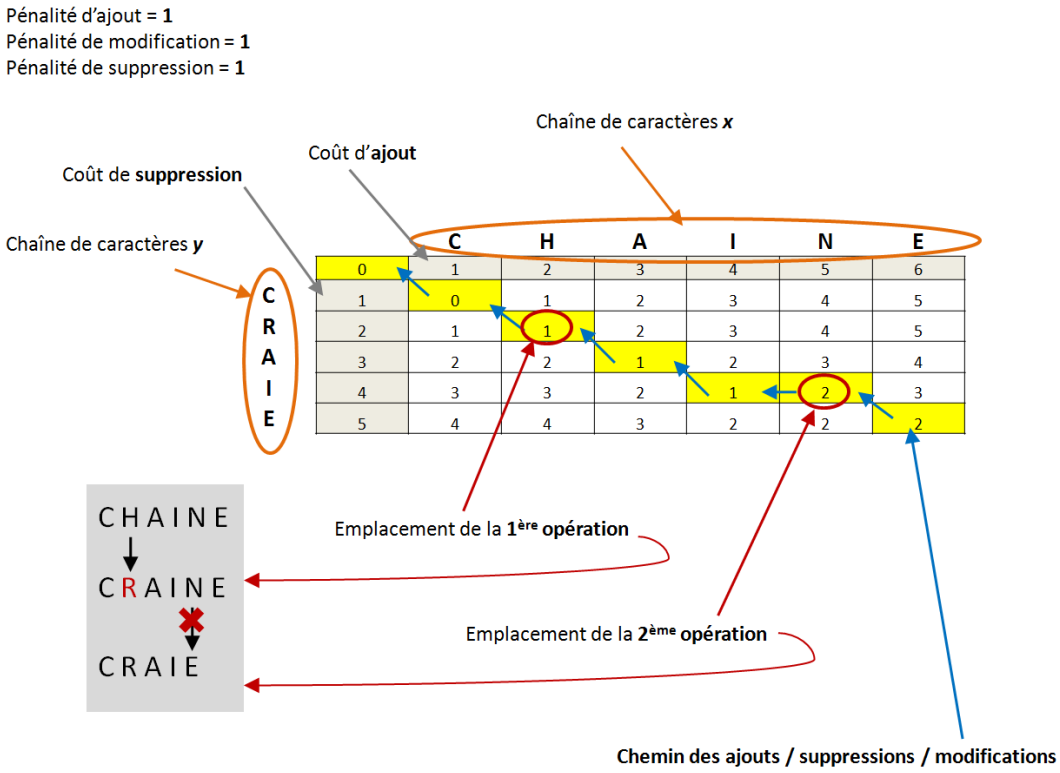


FIGURE 4.9 – Exemple de calcul de Distance de Levenshtein sur 2 chaînes de caractères, avec l'implémentation de Kleiweg [Kle]

Un cas particulier du calcul de la Distance d'édition consiste à déterminer la plus longue sous-séquence commune (plus connue sous la terminologie *LCS - Longest Common Subsequence - problem* en anglais) à deux chaînes de caractères. Celle-ci est obtenue en établissant une matrice des deux chaînes à comparer *x, y* dans laquelle les pénalités d'ajout et de suppression sont fixées à 1, tandis que le coût de suppression est établi à 0. Autrement dit, on ne prend en compte que les ajouts ou suppressions de caractères, et les modifications de caractères sont ignorées [BHR00].

Ce cas particulier du calcul de la Distance d'édition a servi de base à l'élaboration du programme *diff* [Mye86] [WMMM90] utilisé, sous Unix, pour la comparaison de textes.

L'algorithme de **Distance de Hamming** est fortement utilisé par les codes correcteurs d'erreurs [Sym10]. Il ne s'intéresse qu'à la substitution de caractères. A ce titre, l'algorithme ne peut être appliqué que sur deux chaînes de longueur identique. Le résultat qu'il délivre correspond au nombre de caractères modifiés. Par exemple, la Distance de Hamming entre les chaînes "phishing" et "pharming" est de 3.

L'algorithme de **Distance de Jaro-Winkler** est utilisé de préférence sur des chaînes de caractères de faible longueur [Win06], telles que les mots de passe. Le score délivré, compris entre 0 et 1, est représentatif du degré de similarité des deux chaînes comparées. Plus il est élevé, plus les chaînes sont ressemblantes. Dans son calcul de distance, cet algorithme ne prend en compte que les caractères identiques et déplacés. Son déroulement s'effectue en trois étapes :

1. Au sens de l'algorithme Jaro-Winkler, les deux chaînes à comparer sont considérées comme ressemblantes si le nombre de caractères transposés (transp) est inférieur ou égal à :

$$\left(\frac{max(|c1|,|c2|)}{2}\right)-1 \tag{4.4}$$

où *c1* et *c2* représentent le nombre de caractères des deux chaînes à comparer

Dans ce cas, la variable (t) - utilisée ci après pour calculer la distance de Jaro - prend la valeur de $(\frac{transp}{2})$. Sinon (t) = 0.

2. La distance de Jaro est alors calculée, selon la formule suivante :

$$\text{Distance de Jaro} = \frac{1}{3}\left(\frac{m}{|c1|} + \frac{m}{|c2|} + \frac{m-t}{m}\right) \tag{4.5}$$

où m est le nombre de caractères identiques (et au même emplacement) entre les 2 chaînes comparées,
où $c1$ et $c2$ représentent le nombre de caractères des deux chaînes à comparer,
et t est représentatif du nombre de caractères transposés/déplacés. Il est calculé selon la méthode expliquée ci-dessus.

3. Enfin, la distance de Winkler est calculée ainsi :

$$\text{Distance de Winkler} = \text{Distance de Jaro} + \left(l \cdot p(1 - \text{Distance de Jaro})\right) \tag{4.6}$$

où l est le nombre de caractères identiques entre les 2 chaînes comparées (sa valeur maximale est 4).
et p est un coefficient ayant pour but de favoriser les chaînes avec un préfixe commun. Sa valeur recommandée est 0.1.

Dans la catégorie des algorithmes de mesure de similarité, on peut également parler des approches de type ***N-gram***. Celles-ci sont généralement utilisées en traitement du signal pour corriger des erreurs de transmission (p.ex. l'algorithme de Viterbi), ou en traitement automatique du langage naturel pour effectuer de la reconnaissance vocale, de la reconnaissance de textes, etc.

Une étude menée par Cavnar [CT94] a notamment démontré l'efficacité de ce type d'approche pour la classification de texte/document. A un *N-gram* correspond un ensemble de N caractères consécutifs. Le principe de fonctionnement simplifié est alors le suivant :

1. On démarre par une phase de pré-apprentissage, c.-à-d. on prend des documents issus des différentes catégories de classifications souhaitées. Pour chacun de ces documents dits de référence, on détermine l'ensemble des *N-gram* contenus ainsi que leurs fréquences d'apparition. Ainsi, on peut en déduire un profil type associé à chaque catégorie de classification.
2. Chaque document à classifier est ensuite analysé selon le même principe, afin de déterminer les *N-gram* qu'il contient et leurs fréquences d'apparition.
3. Reste alors à effectuer un calcul de distance entre le document à référencer et les profils types pré-établis, la distance la plus courte permettant de définir la catégorie du document.

La figure 4.10 illustre un exemple de classification d'un document basée sur une approche *N-gram*, à partir de profils types créés.

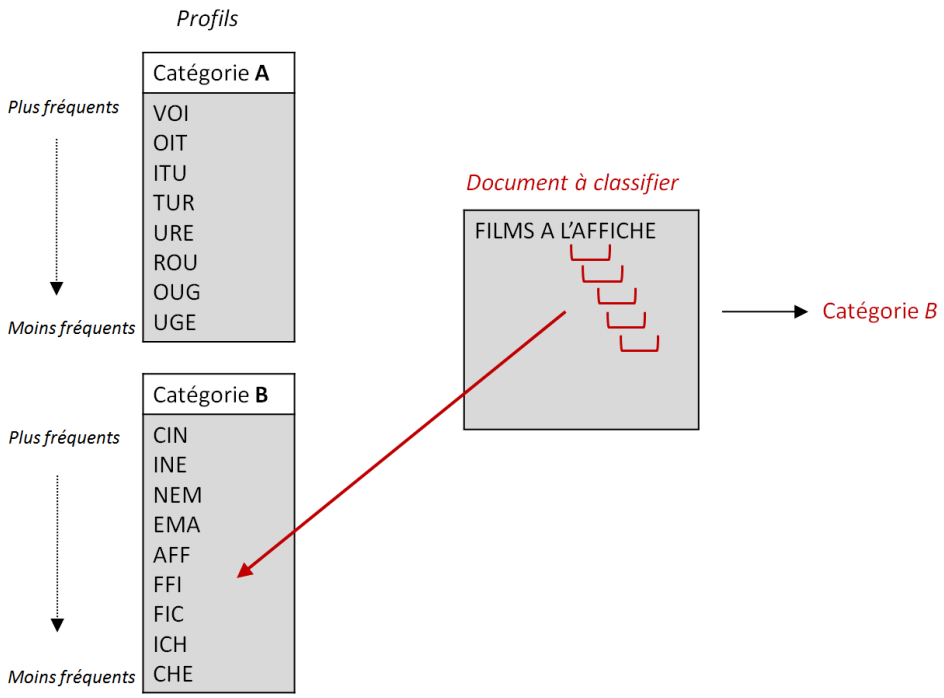


FIGURE 4.10 – Exemple de classification d'un document, à partir d'une approche *N-gram* à base de triplets

4.2.2 Les méthodes de comparaison de structures

D'autres méthodes de comparaison peuvent s'appliquer aux codes sources HTML. En effet de par leur contenu, ces fichiers texte peuvent également être examinés en fonction de leur structure bien particulière et ordonnée (c.-à-d. via les balises).

La différence essentielle avec les méthodes de comparaison de textes vues en section 4.2.1, réside dans le fait que cette fois-ci la comparaison se focalise sur la structure des documents étudiés. En effet, les algorithmes détaillés précédemment font abstraction de la structure, qu'ils considèrent comme du texte au même titre que le contenu réel de la page.

Ces méthodes de comparaison de structures sont dites en arbres (on parle notamment de structure DOM[1]). Elles s'appliquent sur des documents de même type présentant une arborescence ordonnée. On y distingue deux familles d'algorithmes (cf. figure 4.11) : les algorithmes d'alignement et les algorithmes de mesure de similarité.

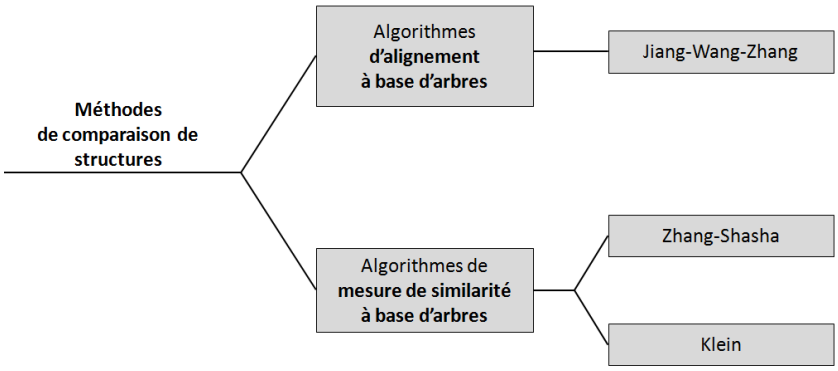


FIGURE 4.11 – Taxonomie des méthodes de comparaison de structures

Les bases communes à ces deux familles d'algorithmes sont les suivantes (cf. figure 4.12) : chaque document à comparer est vu comme un ensemble de nœuds (p.ex. les balises), reliés les uns aux autres selon un ordre déterminé. L'ensemble constitué étant nommé arbre complet et noté T. Le nœud d'origine où chaque nœud intermédiaire est vu comme la racine r d'un arbre F constitué des nœuds inférieurs qui lui sont raccordés. La notation de chaque sous-arbre (compris d'une racine et de l'arbre inférieur qui lui est associé) au sein de l'arbre complet est la suivante : $r(F)$.

A l'image des algorithmes présentés en section 4.2.1, les méthodes de comparaison de structures détaillées ici s'appuient sur l'établissement de matrices de similarité. Au sein de ces matrices, les colonnes et les lignes symbolisent les nœuds des arbres complets à comparer.

4.2.2.1 Les algorithmes d'alignement

Comme nous l'avons vu en section 4.2.1.1, cette famille d'algorithmes est typiquement utilisée en génétique, afin de comparer deux séquences (p.ex. ADN). L'algorithme Jiang et al. [JWZ94] est l'exemple le plus connu des algorithmes d'alignement basés sur les arbres.

Le principe général est le suivant : deux séquences à comparer, représentées par des arbres complets, sont alignées grâce à l'application des opérations d'ajout, suivies par les opérations de suppression. Là encore, le calcul du meilleur alignement est effectué grâce à l'utilisation d'une matrice de substitution (pour plus de détails sur la notion de matrice de substitution, cf. section 4.2.1.1).

La figure 4.13 - extraite de la présentation de Touzet [Tou05] - illustre les opérations réalisées pour passer de l'*arbre complet* T à l'*arbre complet* T' : on commence par ajouter les nouveaux nœuds f et d dans l'arbre de droite. Puis, on supprime les nœuds b et d dans l'arbre de gauche.

1. cf. section 2.2.2.1 pour plus de détails.

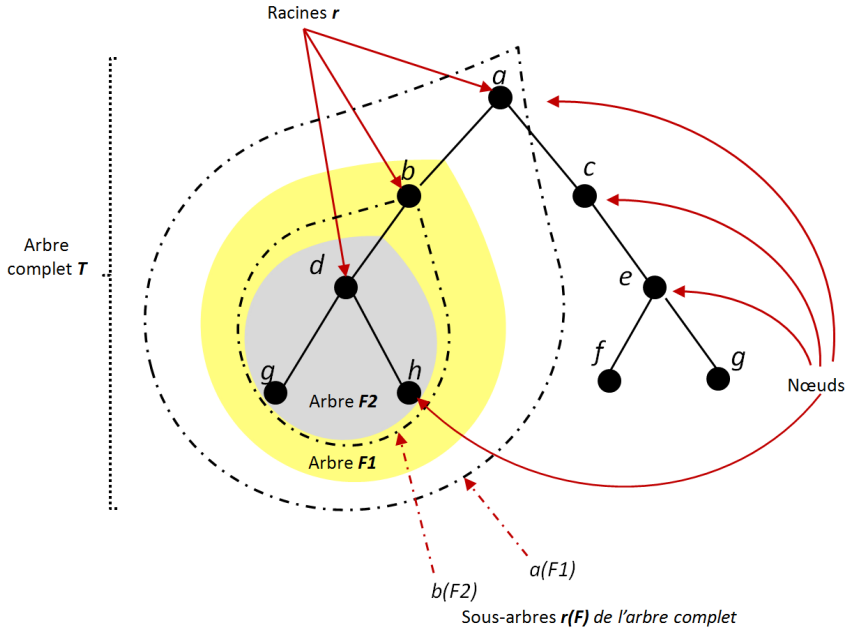


FIGURE 4.12 – Vue simplifiée des composants d'un arbre

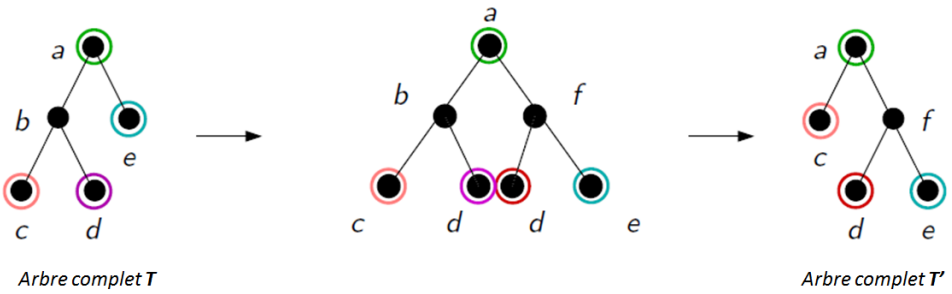


FIGURE 4.13 – Exemple d'opérations réalisées pour l'alignement de deux arbres complets T et T' [Tou05]

4.2.2.2 Les algorithmes de mesure de similarité

Les algorithmes de mesure de similarité basés sur les arbres ont le même but que ceux vus dans les méthodes de comparaison de texte (cf. section 4.2.1.3), à savoir : calculer une distance dite d'édition, représentative du nombre d'opérations minimum nécessaires (ajout, suppression ou modification) pour passer d'un document à l'autre.

Les algorithmes les plus connus dans ce domaine sont ceux de Zhang-Shasha [ZS89] et Klein [Kle98].

Le calcul de la Distance d'édition basée sur les arbres se décompose en deux temps principaux :

1. La distance de chaque sous-arbre est déterminée selon le calcul suivant :

$$Distance\Big(r(F), r'(F')\Big) = min\begin{cases} Distance\Big(F, r'(F')\Big) + Csuppr(r) \\ Distance\Big(r(F), F'\Big) + Cajout(r') \\ Distance\Big(F, F'\Big) + Cmodif(r, r') \end{cases} \tag{4.7}$$

où *Csuppr*, *Cajout* et *Cmodif* sont respectivement les coûts de suppression, ajout et modification d'un noeud.

2. Puis la distance d'édition de l'arbre complet est déterminée suivant deux approches : gauche ou droite. A titre d'exemple, l'algorithme Zhang-Shasha utilise la décomposition gauche, tandis que l'algorithme Klein peut utiliser l'une ou l'autre selon la structure de l'arbre. Nous détaillons ci-après la version gauche du calcul de la distance d'édition de l'arbre complet (La version droite et de plus amples détails sur les méthodes de comparaison à base d'arbres sont disponibles en [DT03] et [Bil05]).

$$Distance\Big(r(F)\circ T, r'(F')\circ T'\Big) = min\begin{cases} Distance\Big(F\circ T, r'(F')\circ T'\Big)+Csuppr(r) \\ Distance\Big(r(F)\circ T, F'\circ T'\Big)+Cajout(r') \\ Distance\Big(r(F), r(F')\Big)+Distance\Big(T, T'\Big) \end{cases} \tag{4.8}$$

où $\circ$ symbolise le rattachement d'un arbre (ou sous-arbre) à son arbre complet.

Au final, la matrice de similarité établie permet d'indiquer l'ensemble des opérations à effectuer pour passer d'un arbre complet à l'autre. Les ajouts de nœuds sont symbolisés par des tracés verticaux, tandis que les suppressions de nœuds sont symbolisées par des tracés horizontaux. Les tracés en diagonale indiquent une modification de noeud ou sa conservation.

La figure 4.14 - extraite de la présentation de Touzet [Tou05] - illustre les opérations menées pour passer de l'*arbre complet T* à l'*arbre complet T'* : on supprime le nœud *b*, puis on ajoute le nœud *f* en amont des nœuds *d* et *e*.

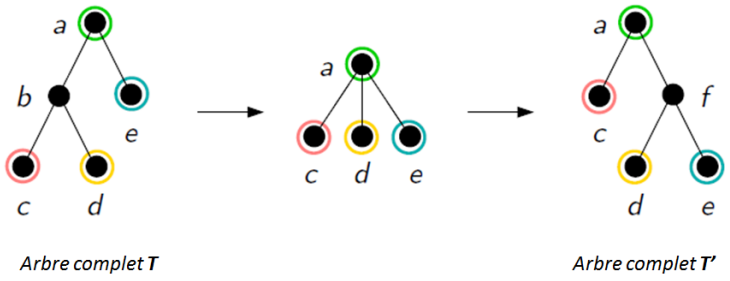


FIGURE 4.14 – Exemple d'opérations considérées par le calcul de Distance d'édition pour passer de l'arbre complet T à l'arbre complet T' [Tou05]

4.2.3 Application des méthodes de comparaison aux pages webs

Dans cette section, nous discutons exclusivement des algorithmes qui se prêtent à la comparaison de pages webs, aux travers de travaux existants.

4.2.3.1 Les méthodes de comparaison de textes

Les algorithmes de recherche de sous-chaînes sont typiquement utilisés par les moteurs de recherches et autres robots qui scrutent les pages webs en vue de les référencer, les classifier, etc. [GJ09] [ACC+09].

Les algorithmes de mesure de similarité sont quant à eux couramment utilisés pour la comparaison de pages webs, tant pour la mise en exergue des zones de changements dans le cadre de l'administration d'un site web, que pour la détection d'éventuels sites frauduleux. On peut citer par exemples les travaux de Medvet et al. [MKK08] qui font appel à la Distance de Levenshtein pour comparer deux pages webs (ou plus exactement des portions de texte contenues dans celles-ci), afin de détecter d'éventuelles contrefaçons. De leur côté, Reddy et al. [RRJ11] appuient leur détection des sites contrefaits sur l'utilisation d'une liste blanche d'URLs/adresses IP associées, afin de déterminer la légitimité du site visité par l'utilisateur. Leur concept fait appel à la Distance de Levenshtein pour comparer les URLs. Fu et al. [FDWL06] proposent quant à eux d'utiliser la Distance de Levenshtein pour détecter les attaques Unicode. Celles-ci sont typiquement utilisées dans des URLs afin de passer des commandes visant à

corrompre la machine de l'utilisateur. De leur côté, Simon et al. [SL05] utilisent l'algorithme Jaro-Winkler pour améliorer les extractions automatisées de contenus webs.

Enfin, l'approche N-gram est couramment utilisée sur le web [WTV+10] [AS] [LCJ+10] aussi bien pour constituer des corpus de mots, qui ont vocation à être utilisés par des outils qui doivent réaliser un découpage de mots/phrases intelligent (p.ex. pour appliquer les césures au bon endroit), que pour permettre la comparaison de documents (p.ex. des pages webs).

4.2.3.2 Les méthodes de comparaison de structures

Seuls les algorithmes de mesure de similarité basés sur les arbres se prêtent à la comparaison de pages webs (vs. les algorithmes d'alignement basés sur les arbres).

Dans leur étude, Mikhael et al. [MS05] ont implémenté un outil de comparaison des pages HTML - nommé *VDiff* -, qui se base sur l'algorithme Zhang-Shasha. Leur outil permet de détecter les changements de structures, c.-à-d. les changements de balises (ajout, suppression, modification). Son usage est typiquement destiné aux webmasters qui veulent apprécier les changements apportés aux pages webs par les différents contributeurs, avant publication.

De leur côté, Kim et al. [KPKC07] attribuent des poids différents aux balises du document HTML (c.-à-d. aux nœuds des arbres), selon la taille d'affichage des informations dans le navigateur web. L'idée est de faciliter la collecte/recherche des informations essentielles contenues dans les pages webs, via une comparaison de contenus effectuée grâce à des algorithmes de Distance d'édition basés sur les arbres.

Précisons que le postulat de départ de ces deux études est de considérer que la page HTML est écrite selon la structure XML .

Les travaux de Garofalakis et al. [GK05] et ceux de Li et al. [LLL+07] s'appuient également sur l'algorithme Zhang-Shasha pour la comparaison de documents XML.

4.3 Synthèse du chapitre

Ce chapitre a présenté le pharming - qui associe corruption DNS et site web frauduleux - ainsi que les différentes zones de vulnérabilité et/ou cibles d'attaques, depuis le réseau client jusqu'au site web visité.

Il est aujourd'hui de notoriété publique que le manque de protection du protocole DNS - première cible de ces attaques - ne peut éviter d'éventuelles compromissions des informations DNS légitimes, ce qui facilite l'exposition de l'Internaute à des sites webs frauduleux sans signe apparent de l'attaque perpétrée. Des efforts notables sont actuellement en cours de déploiement côté FAI/serveur web pour pallier cette lacune, via l'implémentation des extensions de sécurité DNSSEC. Néanmoins, nous avons vu que le réseau client demeure une cible privilégiée et insuffisamment protégée.

Côté client, l'ensemble des propositions de détection/prévention du pharming se rejoignent sur une volonté de mieux contrôler le contenu des pages webs visitées. Parmi les différentes pistes d'analyses de contenu des pages webs, nous avons choisi de détailler les méthodes de comparaison pouvant s'appliquer à l'étude des codes sources HTML (au niveau texte et structure), amorçant ainsi un des vecteurs de détection utilisé dans notre contribution.

5 Détection du pharming côté client : vers un remplacement du nom de domaine

Sommaire

Le Chapitre 3 montre l'efficacité des tests heuristiques pour la détection des sites de phishing, ainsi que leurs limitations. Il fait notamment apparaître la prédominance de certains heuristiques dans la différenciation des sites légitimes et contrefaits. Plus particulièrement, il met en évidence l'intérêt de l'analyse du code source HTML des pages webs.

En complément, le Chapitre 4 détaille les différentes techniques de comparaisons de contenus pouvant s'appliquer aux codes sources HTML. Par ailleurs, alors que les zones de vulnérabilités sont nombreuses côté client, ce même chapitre démontre la difficulté de détection des attaques de pharming pour l'Internaute.

Dans les Chapitres 5 et 6, nous nous intéressons à la portabilité de cette analyse du code source HTML à l'identification de sites webs, et plus particulièrement au cadre de la détection des attaques de pharming auxquelles le client est encore trop vulnérable.

Notre proposition est étayée via deux approches - visant à détecter le pharming réalisé côté client - basées sur l'étude du code source HTML du site web visité, combinée à des requêtes DNS. Par souci de clarté et afin d'apporter un maximum de détails sur chacune de ces deux approches, nous les détaillons de manière successive. Il est important de noter que la seconde approche (développée dans le Chapitre 6) est issue de l'évolution de la première (détaillée ci-après), avec pour principaux objectifs la correction et l'amélioration des défauts majeurs rencontrés.

Nous démarrons donc ce chapitre par une introduction à l'approche développée (cf. section 5.1). Nous y expliquons notamment l'origine de notre proposition et nous en profitons pour la positionner par rapport aux travaux existants.

Puis, nous détaillons une étude préalable que nous avons réalisée sur les hypothèses de travail (cf. section 5.2) avec le double objectif : 1/ de tester si la vérification de l'adresse IP du domaine visité peut s'avérer un critère de détection pertinent, et 2/ d'évaluer s'il est possible de différencier des pages légitimes de pages contrefaites au travers de l'analyse du contenu global des codes sources HTML.

Enfin, nous développons la première approche proposée (cf. section 5.3) qui combine à la fois une vérification de l'adresse IP du domaine visité et l'analyse du contenu de la page web visitée vs. des éléments de référence. Ces éléments sont récupérés à partir des informations fournies par un serveur DNS, différent de celui auquel est connecté le client (c.-à-d. tel que proposé par son FAI). Ils sont constitués d'une adresse IP (liée au domaine visité) ainsi que du code source d'une page web, tous deux dits de référence. La page web de référence est récupérée grâce à la génération d'une nouvelle requête HTTP. Cette requête est basée sur l'URL initialement visitée par l'utilisateur, au sein de laquelle la zone de domaine est remplacée par l'adresse IP de référence. Les tests réalisés dans cette proposition portent sur 108 URLs de login légitimes, évaluées depuis 11 localisations géographiques réparties sur 5 continents, ainsi que sur 37 couples de pages légitimes-contrefaites.

Ce chapitre fait partie de nos contributions : cette première proposition a été publiée et présentée à la conférence *New Technologies, Mobility and Security* (NTMS) en Février 2011 [GGL11b].

5.1 Introduction à l'approche développée

Notre proposition tire son origine de 2 éléments :

- La volonté de se focaliser côté client, afin de détecter les attaques de pharming pour lesquelles les efforts déployés côté réseau sont encore inefficaces.
- Une étude menée par Stamm et al. [SRM07], que nous avons considérée comme déterminante, puisqu'elle met en avant la vulnérabilité essentielle des routeurs personnels déployés à large échelle, à savoir : leurs utilisateurs. En effet, la méconnaissance des Internautes, qui utilisent les routeurs personnels dans leur pré-configuration d'origine, induit des points d'entrée providentiels pour les attaquants (cf. section 4.1.2.1).

A l'image des mécanismes de protection couramment utilisés contre les attaques de phishing (ie. les barres d'outils intégrées dans les navigateurs), nous avons élaboré une proposition de détection des attaques visant à s'intégrer dans le navigateur de l'Internaute. D'où les développements menés en langage Java.

Notre proposition vise à s'intégrer de façon complémentaire dans une barre d'outils anti-phishing. Un indicateur visuel très simple et binaire a pour rôle d'indiquer le niveau de confiance envers le site visité. En complément, en cas de site suspicieux, l'Internaute est alerté via une notification active (p.ex. un message d'alerte affiché sous forme de pop-up) (cf. section 6.2.3).

Le cœur de notre proposition repose sur la combinaison de deux analyses menées grâce aux informations fournies par plusieurs serveurs DNS :
- La vérification de l'adresse IP du domaine visité
- L'analyse et la comparaison du code source de la page web

Les attaques ciblées : Dans l'absolu, notre proposition de détection du pharming côté client vise les attaques mentionnées en figure 5.1 (cf. section 4.1.2.1 pour plus de détails sur ces attaques). Ainsi, elle cible aussi bien les attaques qui visent à éviter l'interrogation du serveur DNS configuré (par ajout de fausses entrées IP/FQDN), que les attaques qui visent à rediriger le trafic (par ajout de routes statiques[1] ou de proxy HTTP qui orientent le trafic vers les machines de l'attaquant), ou encore les attaques qui modifient l'adresse IP du serveur DNS habituellement interrogé.
Précisons toutefois que seule la seconde approche développée dans le Chapitre 6 permet de cibler l'ensemble de ces attaques.

Cible / Zone de vulnérabilités		Ajout de fausse association	Modification de l'adresse IP du serveur DNS interrogé	Autre
Poste client	Fichier HOSTS	**X** (IP/FQDN)		
	Configuration Réseaux		**X**	
	Cache DNS	**X** (IP/FQDN)		
	Navigateur web	**X** (IP/FQDN)		Proxy HTTP *DNS rebinding*
Réseau local			**X**	
Routeur personnel		**X** (routage)	**X**	

FIGURE 5.1 – Les attaques ciblées par notre proposition de détection du pharming côté client

Positionnement par rapport aux travaux similaires : Les travaux précédents, qui s'apparentent aux deux approches proposées dans notre étude, sont détaillés en sections 4.1.3.3.2 et 4.2.3. Les quatre études les plus proches sont discutées ci-après :
- La méthode développée par Cao et al. [CHL08] base sa détection du pharming sur une comparaison de l'adresse IP du domaine visité auprès d'une liste blanche d'adresses IP légitimes. Elle repose sur un principe de staticité des adresses IP associées aux pages de login. L'étude que nous menons dans ce chapitre va démontrer que la prise en compte de l'adresse IP du domaine visité s'avère un critère parfois insuffisant pour déterminer la légitimité d'un site. A contrario de cette étude, nous avons également exclu ici toute méthode de détection nécessitant le maintien d'une liste blanche ou liste noire (pour les faiblesses associées à cette approche, cf. explications détaillées dans les Chapitres 2 et 3).

- La méthode de détection proposée par Bin et al. [BQX10] est certainement celle qui s'approche le plus de notre travail de par l'étude combinée de deux types de contenus (c.-à-d. une donnée saisie dans une page web et l'adresse IP associée au domaine visité). Leur technique s'appuie sur une liste blanche pré-établie de noms de banques, adresses IP et plages de numéros de cartes bancaires associés. De notre point de vue, elle présente deux inconvénients majeurs : 1/ Elle nécessite un temps de latence proche de zéro. En effet, leur méthode implique obligatoirement un début de

1. Cette attaque n'est détectée que si l'adresse IP retournée par le serveur de référence - interrogé dans notre proposition - est différente de celle configurée dans la route statique.

saisie d'une information critique de l'utilisateur, à savoir son numéro de carte bancaire, en amont du lancement de la détection. 2/ De plus - tel qu'évoqué précédemment -, nous considérons que la nécessité du maintien d'une liste blanche est à éviter, de par les vulnérabilités supplémentaires qu'elle introduit.

- L'approche développée par Bilge et al. [BKKB11], publiée très récemment, est celle qui s'apparente le plus à notre étude en terme de techniques de détection utilisées, à savoir les tests heuristiques basés sur les informations DNS. Nos travaux et les leurs se basent sur une vérification de l'adresse IP du domaine visité. Néanmoins, leur étude se destine plutôt au phishing, de par le contenu de la majorité des critères étudiés : âge du site visité, nombre de caractères alphanumériques du nom de domaine, etc.

- L'étude de Reddy et al. [RRJ11], récemment publiée, propose deux scénarios de détection des sites contrefaits côté client. L'un des deux scénarios exposé se rapproche de notre proposition, bien qu'il se destine plutôt à la détection des sites de phishing. Celui-ci s'appuie sur une vérification de l'adresse IP du site visité, combinée à une analyse de l'URL en utilisant la Distance de Levenshtein. Les éléments de référence utilisés pour leurs vérification/comparaison sont issus d'une liste blanche pré-établie. Comme exposé précédemment, nous pensons que l'utilisation d'une liste blanche est un point de vulnérabilité important de la solution. De plus, ce scénario ne peut s'appliquer aux sites de pharming puisqu'il ne s'intéresse qu'à l'étude de l'URL visitée.

5.2 Etude préalable sur les hypothèses de travail

Plusieurs pistes ont été étudiées avant d'aboutir aux deux approches décrites en sections 5.3 et 6.1, l'idée essentielle étant de trouver un moyen d'utiliser un serveur DNS et une page web dits de référence.

Divers scénarios intermédiaires ont été envisagés pour la définition d'un serveur de référence. Toutefois, parce que ces scénarios impliquaient l'échange d'une adresse de référence entre le client et le FAI (ou une tierce partie), nous avons jugé préférable de les abandonner. De plus, ils restaient vulnérables à une corruption de l'adresse de référence en amont, dans la chaîne de résolution DNS.

Nous avons également un temps envisagé une comparaison "délocalisée", réalisée en dehors du réseau client, tant pour la vérification de l'adresse IP que pour la comparaison de pages webs. Néanmoins, les résultats de notre étude préalable nous ont indiqué que cela s'avérait difficilement exploitable (cf. sections 5.2.1 et 5.2.2).

Cette section s'intéresse donc à expliquer les études préalables réalisées, ainsi que les choix auxquels elles ont abouti dans la conception de notre approche.

5.2.1 Variabilité de l'adresse IP du domaine visité

Un des fondements de notre approche repose sur la possibilité de vérifier l'adresse IP du site visité.

Nous avons donc conduit un premier set d'expérimentations depuis 9 localisations géographiques réparties sur 5 continents (Amérique du Nord, Europe, Afrique, Asie et Australie). Pour chaque localisation, nous avons récupéré les adresses IP retournées par le serveur DNS par défaut ainsi que 2 serveurs DNS de référence (OpenDNS [ope] et GoogleDNS [gooa]), afin d'analyser les variations d'adresses IP des domaines visités.

5.2.1.1 Échantillon d'URLs

Nous avons testé 226 domaines (plus exactement des FQDN) de sites webs HTTP, sélectionnés de la manière suivante :

- 100 domaines issus des sites webs les plus populaires au niveau mondial[1],
- 100 domaines issus des sites webs les plus populaires en France[1],
- et 26 domaines issus de sites bancaires

1. Les domaines ont été récupérés depuis le Top 1000 Google des sites les plus visités [Gooc], le Top 500 Alexa des sites webs [Al] ainsi que la base de sites publiée par NetCraft [Net].

Tableau 5.1 – Quelques résultats OpenDNS pour des FQDN génériques, sur 4 localisations géographiques

	France	Tunisie	Mexique	Turquie
www.facebook.com	66.220.146.25	66.220.153.19	66.220.146.11	66.220.153.11
www.apple.com	92.123.129.15	92.123.193.15	184.50.237.15	92.123.233.15
www.ask.com	62.41.85.83	62.41.85.49	63.97.94.41	194.221.37.139
	62.41.85.49	62.41.85.83	63.97.94.80	194.221.37.163
www.amazon.com	72.21.210.250	72.21.210.250	72.21.207.65	207.171.166.252
www.commentcamarche.net	62.41.85.40	62.41.85.40	63.97.94.35	194.221.37.163
	62.41.85.48	62.41.85.48	63.97.94.49	194.221.37.176
www.comcast.net	67.215.65.132	213.155.157.49	63.97.94.10	93.158.110.107
		213.155.157.16	63.97.94.58	93.158.110.144

Tableau 5.2 – Quelques résultats OpenDNS pour des FQDN plus précis, sur 4 localisations géographiques

	France	Tunisie	Mexique	Turquie
webmail.laposte.net	193.251.214.117	193.251.214.117	193.251.214.117	193.251.214.117
portail.free.fr	212.27.48.10	212.27.48.10	212.27.48.10	212.27.48.10
images.google.fr	66.102.9.99	66.102.9.105	209.85.225.106	74.125.39.105
	66.102.9.103	66.102.9.99	209.85.225.105	74.125.39.147
	66.102.9.147	66.102.9.106	209.85.225.103	74.125.39.104
	66.102.9.106	66.102.9.103	209.85.225.147	74.125.39.99
	66.102.9.105	66.102.9.147	209.85.225.99	74.125.39.106
	66.102.9.104	66.102.9.104	209.85.225.104	74.125.39.103
news.bbc.co.uk	212.58.226.77	212.58.226.138	212.58.246.83	212.58.226.140
login.yahoo.com	69.147.112.160	209.191.92.114	217.146.187.123	69.147.112.160
			217.12.8.76	
particuliers.societegenerale.fr	193.178.154.165	193.178.154.167	193.178.154.165	193.178.154.166
	193.178.154.166	193.178.154.164	193.178.154.166	193.178.154.167
	193.178.154.167	193.178.154.165	193.178.154.167	193.178.154.164
	193.178.154.164	193.178.154.166	193.178.154.164	193.178.154.165

Nous avons pris soin de multiplier les secteurs d'activité (p.ex. e-commerce, réseaux sociaux, annuaires téléphoniques, banques, forums de discussion, jeux en ligne, etc.), les langages des pages webs ainsi que leurs TLDs.

5.2.1.2 Variabilité IP entre localisations géographiques

Les résultats de tests démontrent que, pour les domaines évalués, les adresses IP varient notablement selon la localisation, et ce quel que soit le serveur DNS interrogé. A titre d'exemple, le tableau 5.1 reprend quelques résultats retournés par OpenDNS depuis 4 localisations différentes.

Nous constatons par exemple que pour Facebook, Apple et Comcast - malgré quelques similitudes -, il n'y a aucun recoupement d'adresses entre les 4 localisations. Tandis que pour Ask, Amazon et Commentcamarche, nous avons quelques recoupements d'adresses, entre la France et la Tunisie.

En utilisant des FQDN plus précis (p.ex. images.google.fr, portail.free.fr, webmail.laposte.net, etc.), nous obtenons davantage de résultats partiellement convergents, entre les différentes localisations (cf. tableau 5.2). Toutefois, nous notons que des exceptions subsistent (p.ex. login.yahoo.com, news.bbc.co.uk).

5.2.1.3 Variabilité IP depuis une même localisation géographique

Nous avons également évalué la staticité des adresses IP sur 4 localisations. Pour chacune d'entre elles, nous avons comparé - en local - les adresses IP retournées par le serveur DNS par défaut ainsi que celles retournées par nos 2 serveurs de référence (GoogleDNS et OpenDNS).

Pour les FQDN génériques, selon la localisation étudiée et pour une même liste d'URLs, nous constatons une grande variabilité des résultats (cf. tableau 5.3) entre les adresses IP retournées par le DNS par défaut et les adresses IP de référence.

Néanmoins, lorsque nous analysons les résultats portant sur les FQDN plus précis, nous constatons à nouveau une nette amélioration (c.-à-d. les adresses IP des serveurs DNS Défaut et Référence sont

Tableau 5.3 – Convergence des adresses IP Défaut et Référence pour 4 localisations géographiques

	Taux de convergence avec les adresses IP retournées par le serveur DNS par défaut (min ≤ moyenne ≤ max)	Écart-Type[1]
OpenDNS	62.83% ≤ **71.24%** ≤ 92.92%	14.55%
GoogleDNS	62.39% ≤ **73.34%** ≤ 84.51%	10.82%

[1] entre localisations

plus convergentes).

5.2.1.4 Problématique des informations WHOIS

Nous avons envisagé d'exploiter les données WHOIS en vue de vérifier l'identité des domaines visités. Néanmoins, plusieurs problèmes se posent :

- La RFC 3912 qui définit le protocole WHOIS ne spécifie pas de norme concernant les données stockées. Elle n'indique pas non plus de spécification sur le formatage/contenu des réponses WHOIS, envoyées en mode texte. Chaque registrar fournit donc ce service dans le format qu'il désire, ce qui rend les données difficilement exploitables de manière automatisée.
- On peut alors imaginer se focaliser sur les cc-TLD, qui sont généralement gérés par une même autorité gouvernementale imposant un format standard pour l'ensemble de ses domaines. Néanmoins, le service WHOIS n'étant pas dimensionné pour le traitement automatique, l'interrogation automatique s'avère difficile. En effet, lorsque nous avons essayé d'automatiser des requêtes WHOIS, nous constatons qu'après une dizaine de requêtes depuis une même machine, l'adresse IP source est bloquée.
- Il n'y a pas non plus d'obligation de maintenir des informations WHOIS à jour. D'ailleurs, ces informations accessibles au public, posent des problèmes liés à la protection (au sens protection de la vie privée). De nombreuses données stockées sont donc obsolètes et peu fiables (En 2010, une étude de l'ICANN - menée sur 5 g-TLD et 1419 enregistrements - a montré que seulement 23% des domaines interrogés ont des données WHOIS à jour [atUoCfI10]).
- Enfin, lorsqu'une entreprise utilise des services de GSLB (Global Server Load Balancing) pour une meilleure disponibilité, les informations retournées pour une adresse IP peuvent s'avérer fausses ou inexploitables pour l'identification d'un domaine. Considérons un exemple : Une interrogation DNS sur le FQDN `www.bouyguestelecom.fr` retourne les adresses IP 62.41.70.12 et 62.41.70.138. Une interrogation de la base WHOIS pour l'adresse IP 62.41.70.12 retourne des informations sur la société Akamai International. Il n'y alors aucun moyen de faire le lien entre l'entreprise BouyguesTelecom et la société Akamai, spécialisée dans la mise en cache de contenus web.

5.2.1.5 Synthèse de l'étude préalable sur la variabilité des adresses IP : choix des pages de login

Les données WHOIS peuvent générer un taux de faux-positif important, tant par leur obsolescence que par l'externalisation de la gestion des sites webs. Elles s'avèrent donc inexploitables pour la vérification de l'adresse IP d'un FQDN.

L'interrogation DNS sur des FQDN génériques démontre une trop grande variabilité des réponses (c.-à-d. adresses IP retournées) selon la localisation géographique. Notre approche visant à s'adresser à tout client, quelle que soit sa localisation, il s'avère donc impossible de délocaliser la vérification d'adresse IP sur un serveur déporté.

Enfin, des interrogations DNS sur des FQDN "plus précis" semblent être davantage propices à des résultats convergents entre serveur par défaut et serveur de référence, depuis une même localisation. Une étude menée par Cao et al. [CHL08] semble d'ailleurs indiquer une stabilité des adresses IP associées aux sites de login.

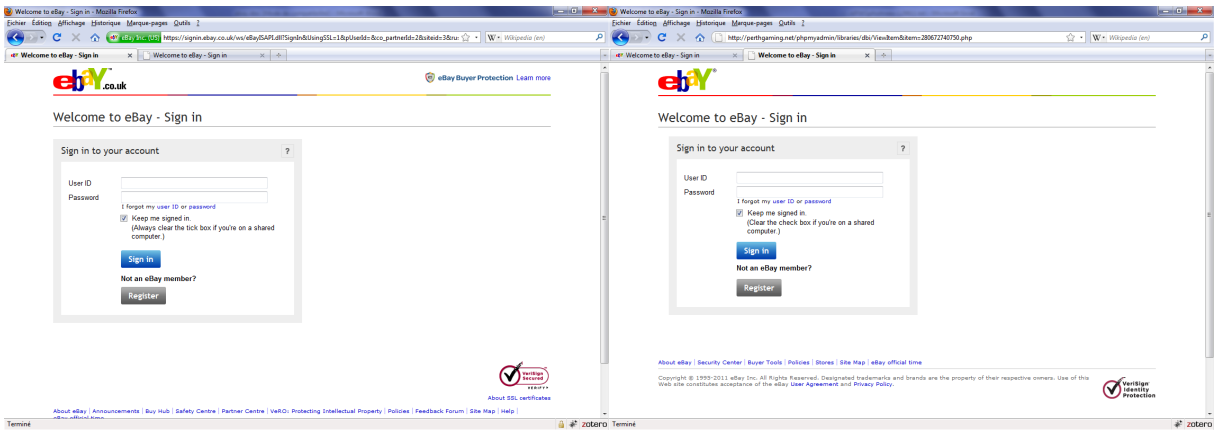
Pour les raisons évoquées précédemment, **nous focalisons donc la suite de notre étude sur des FQDN "plus précis", et plus spécifiquement sur des pages de login, cibles des attaques de pharming**.

5.2.2 Contenu des pages webs

5.2.2.1 Analyse des caractéristiques des pages webs légitimes et contrefaites

Nos travaux menés sur l'étude de pages webs légitimes et contrefaites, détaillés dans le Chapitre 2, ainsi que des analyses de pages récupérées depuis différents navigateurs webs et différentes localisations, font apparaître les caractéristiques et difficultés suivantes :

- **Le contenu des pages webs est de plus en plus dynamique**, incluant des flux RSS, des images renouvelées fréquemment, des publicités, etc.
- **Les attaquants créent des sites contrefaits qui, visuellement, sont de plus en plus ressemblants aux sites légitimes** (cf. exemple en figure 5.2). En effet, l'utilisation d'aspirateurs de sites webs et le choix délibéré de conserver un maximum de liens du site web légitime, permettent de minimiser les possibilités de détection de la contrefaçon et de leurrer un plus grand nombre d'utilisateurs.
- **Les sites légitimes et contrefaits utilisent tous deux des chemins absolus et relatifs** pour les références aux liens, images, etc.
- **Le code source des pages webs peut être modifié selon le navigateur utilisé par le client**. En effet, des morceaux de script peuvent être ajoutés aux codes sources des pages webs, selon le navigateur utilisé (p.ex. Internet Explorer, Firefox, Opera, etc.)
- **La structure HTML d'une page web est parfois modifiée d'une localisation à une autre**. Nous avons en effet constaté que l'ordonnancement du code source d'une même page web récupérée - à partir de la même URL - depuis différentes localisations, se retrouve parfois totalement bouleversé, bien que les deux pages apparaissent totalement identiques à l'affichage dans le navigateur.
- **Le contenu d'une page HTML est adapté aux préférences et/ou à la localisation de l'utilisateur**. En effet, l'affichage se fait selon la langue choisie, ou la localisation depuis laquelle la requête est effectuée.



(a) Site légitime (b) Site contrefait

FIGURE 5.2 – Captures d'écran du site légitime eBay (`https://signin.ebay.co.uk/ws/eBayISAPI.dll?SignIn&UsingSSL...`) et d'une contrefaçon (`http://perthgaming.net/phpmyadmin/libraries/dbi/ViewItem&item=280672740750.php`), récupérées le 12 Juin 2011

Nous en déduisons alors que baser l'analyse et la comparaison des pages webs uniquement sur la structure, la page complète (c.-à-d. incluant les contenus dynamiques) ou le type de liens, peut générer des taux de faux-positifs élevés.

La variabilité du contenu des pages, selon le navigateur utilisé, les paramètres de l'utilisateur et la localisation géographique, nous amène également à exclure toute possibilité de comparaison déportée. En effet, le contenu de la page web étant assujetti au contexte client (c.-à-d. le type de navigateur, sa version, les éventuels plug-in associés, etc.), il semble difficile de disposer d'une page web de référence personnalisée avec ces mêmes paramètres depuis une autre machine. En complément, de par la haute qualité de bon nombres de pages contrefaites qui répliquent (quasi-)parfaitement les sites légitimes usurpés et le développement des contenus dynamiques (qui peuvent conduire à des détections erronées [MKK08]), nous choisissons également d'exclure toutes techniques de comparaison par "images" de la page web.

Pour l'ensemble de ces raisons, **nous décidons donc de concentrer notre comparaison de pages webs sur l'étude du code source HTML**.

5.2.2.2 Décision de légitimité

Notre décision de légitimité d'une page web va s'appuyer sur un seuil de décision pré-défini, auquel sera comparé le taux de similitude obtenu pour les deux pages confrontées. Pour évaluer l'efficacité de notre proposition et déterminer ce seuil de décision, nous basons donc nos analyses sur deux types de comparaisons :

- des comparaisons de pages légitimes. L'idée est de déterminer si deux pages légitimes, potentiellement identiques - puisqu'elles correspondent à une même URL - mais récupérées par deux biais différents, apparaissent effectivement - du point de vue de notre comparaison - comme totalement ou très similaires.
- des comparaisons entre pages légitimes et contrefaites. L'idée est de déterminer si deux pages, visuellement très ressemblantes mais générées par deux personnes/organismes totalement différents, sont effectivement - du point de vue de notre comparaison - très différentes.

Plus la frontière basse des comparaisons de pages légitimes sera distincte et éloignée de la frontière haute des pages légitimes-contrefaites, plus il sera facile de déterminer un seuil de décision valable, amenant à des décisions fiables.

5.2.2.3 Application des méthodes de comparaison de documents HTML à la problématique des pages légitimes et contrefaites

Alors que le Chapitre 4 a détaillé les différentes techniques de comparaisons de contenus pouvant s'appliquer aux documents HTML, nous examinons à présent quelles sont celles qui pourraient s'appliquer à notre étude (c.-à-d. au cas des comparaisons de pages légitimes et contrefaites).

5.2.2.3.1 Pertinence de l'application des différentes méthodes de comparaison de textes à notre étude : Les algorithmes d'alignement de chaînes (cf. section 4.2.1.1) s'appliquent généralement au domaine de la bio-informatique où il est possible de définir une distance entre deux protéines. Dans ce contexte, les algorithmes d'alignement de chaînes reposent sur l'utilisation d'une matrice de substitution (p.ex. PAM, BLOSUM), dont le rôle est le suivant : pour une protéine donnée, elle indique sa probabilité de substitution par une autre protéine, suite à une mutation dans le temps [Jas].

Une application de ce type d'algorithmes à notre étude s'avère plus compliquée. En effet, il est difficile de prédire le caractère de remplacement pour un caractère donné, en cas de modification du code source de la page web (p.ex. un changement dans un lien hypertexte). Nous pourrions éventuellement nous contenter d'utiliser la pénalité de trou sans matrice de substitution. Ainsi un trou serait inséré jusqu'à ce que la correspondance reprenne. Cependant, au regard de la taille de l'alphabet et de la diversité des caractères pouvant être utilisés (c.-à-d. selon les langages), il se peut que la correspondance tarde à reprendre. Nous pourrions alors nous retrouver avec des zones de trous conséquentes, ce qui impacterait le degré de similitude des pages de manières conséquente (si on comptabilise le nombre de trous), pour un simple changement d'arborescence dans des liens par exemple. Par conséquent, notre choix ne s'est pas porté pas sur ce type d'algorithmes.

Les algorithmes de recherche de sous-chaînes (cf. section 4.2.1.2), qui permettent de recherche une chaîne de caractères au sein d'un texte, ne sont pas adaptés à notre problématique. Ils ne sont donc pas exploités dans notre étude.

Dans les algorithmes de mesure de similarité (cf. section 4.2.1.3), **seuls deux types d'algorithmes/approches retiennent notre attention : l'algorithme de Distance de Levenshtein et l'approche *N-gram***.

En effet, la Distance de Levenshtein prend en compte les 3 types d'opérations qui nous intéressent : la modification, l'ajout et la suppression. Elle nous semble donc pertinente pour l'étude développée.

La Distance de Hamming, quant à elle, ne s'applique que sur des chaînes de longueur identique. De plus, elle ne prend en compte que les substitutions de caractères. Elle ne peut donc être révélatrice des éventuelles insertions de script malveillants, ce qui ne répond pas à nos attentes.

L'algorithme de Distance de Jaro-Winkler délivre un score de type taux de similitude, représentatif des caractères identiques et déplacés. Partant du principe que le code source HTML d'une page contrefaite est fortement similaire à son pendant légitime, nous pourrions imaginer que cet algorithme puisse être intéressant pour notre problématique. Il faudrait pour cela, au préalable, découper les codes HTML en mots (c.-à-d. à raison d'un mot par ligne par exemple). Toutefois, dès l'apparition d'un décalage de mots (principalement en première moitié de code), le score délivré serait fortement impacté. En effet, cette Distance ne permet pas de distinguer l'ajout et/ou la suppression de mots. A titre d'exemple, si on prend le cas des 2 pages webs suivantes : la page légitime contient 500 mots, la page contrefaite les mêmes 500 mots auxquels ont été rajoutés 50 mots (p.ex. correspondant à l'insertion d'un script malveillant) au milieu du code. Ainsi, seuls les 250 premiers mots apparaissent strictement identiques et dans le même ordre. En conséquence, le taux de similitude retourné ne serait que de 69% alors que seulement 10% de mots ont été insérés, et que le reste du code est identique en terme de contenu.

Enfin, l'approche *N-gram*, qui est fortement recommandée pour le filtrage et le routage de documents textes [CT94], nous semble elle aussi intéressante pour notre étude. En effet, si on considère que le code source HTML légitime est notre document de référence, nous pouvons utiliser l'approche *N-gram* pour lui associer un score. Le code source HTML suspect (c.-à-d. le site visité par l'utilisateur) peut alors être traité selon le même processus. Puis, selon l'écart mesuré entre les deux scores obtenus, nous pourrions déterminer la légitimité du site visité.

5.2.2.3.2 Pertinence de l'application des différentes méthodes de comparaison de structures à notre étude : De par leur contenu, les pages HTML peuvent également être considérées selon leur structure. La condition préalable à l'utilisation des algorithmes de calcul d'édition basés sur les arbres est que les deux documents comparés doivent être de structure identique. Si on part de l'hypothèse que la page contrefaite est fortement ressemblante à son pendant légitime, c'est en grande partie parce que les attaquants utilisent des outils/techniques d'aspiration du site légitime comme base à l'élaboration de la contrefaçon. Ainsi, on peut donc supposer que bon nombre de sites contrefaits doivent avoir une structure très similaire à la page usurpée[1].

Les algorithmes de mesure de similarité sont typiquement indiqués pour la comparaison de documents XML [DT03] - un langage parallèle au HTML - qui utilisent un format de balisage sous forme d'arborescence. En comparaison avec le HTML qui est utilisé pour l'affichage des pages webs, le XML est dit "extensible" car il permet de définir ses propres balises en fonction des données traitées. Il est donc utilisable sur différents types de plateformes (p.ex. les mobiles, etc). Cependant, le XML est un langage strict dont l'écriture se doit d'être rigoureuse. A contrario, le HTML permet plus d'erreurs syntaxiques automatiquement corrigées par les navigateurs webs. Il est ainsi désormais devenu fréquent de rencontrer des codes source de pages webs truffés d'erreurs syntaxiques (p.ex. oublis de fermeture de balises, guillemets manquants, changements de l'ordre de balises imbriquées lors de leur fermeture, etc.), particulièrement en ce qui concerne les sites contrefaits (cf. section 6.1.5.6). Certes l'utilisation du XHTML - un HTML 4.0 amélioré selon les règles de syntaxe du XML - pour l'élaboration des pages webs limite ces dérives syntaxiques. Toutefois, force est de constater qu'une grande majorité des pages webs analysées dans notre étude sont encore en HTML. Pour cette raison, il est par conséquent difficilement envisageable d'utiliser une méthode de comparaison utilisant des algorithmes de mesure de similarité basés sur les arbres. Nous n'avons donc pas retenu cette piste, ni celle des algorithmes d'alignement basés sur les arbres qui ne répondent pas à nos attentes (pour les mêmes raisons que celles évoquées en section 5.2.2.3.1). Reste à voir si l'arrivée du HTML5 - actuellement en cours de développement [W3Cc] -, confirmera ou infirmera cette tendance.

5.2.2.4 Les méthodes de comparaison retenues

Nous choisissons donc de focaliser notre comparaison de pages webs sur l'étude du code source HTML en utilisant deux approches : l'une par caractères, l'autre par mots. L'idée première est de refléter les modifications globales apportées à l'intégralité du code source. Les méthodes d'analyse visent ainsi à détecter les changements de structure (balises) et de contenu (texte affiché, liens, etc.) de manière globale, sans pour autant préciser les zones affectées.

1. Notons ici que nous considérons le cas le plus défavorable. En effet, les travaux de Medvet et al. [MKK08] indiquent que la structure DOM d'une page web n'est pas nécessairement un critère de comparaison fiable, puisqu'un attaquant peut utiliser une structure différente et aboutir malgré tout à une apparence visuelle très similaire à celle de la page légitime.

TABLEAU 5.4 – Échelle des valeurs Ascii attribuées aux caractères alphanumériques

Échelle des caractères alphanumériques	**Échelle des valeurs Ascii associées** en décimal
[a ;z]	[97 ;122]
[A ;Z]	[65 ;90]
[0 ;9]	[48 ;57]

TABLEAU 5.5 – Taux de similitude de 10 à 23 couples de pages de login légitimes-contrefaites, avec l'approche par caractères

	Taux de similitude entre page légitime et page contrefaite (min ≤ moyenne ≤ max)	Écart-Type[2]
10 couples de pages[1]	59.09% ≤ **80.54%** ≤ 93.12%	15.93%
23 couples de pages[1]	19.57% ≤ **77.41.%** ≤ 99.93%	26.05%

[1] Les deux séries de tests portent sur des couples de pages distincts
[2] entre couples de pages

5.2.2.4.1 Approche par caractères : Cette méthode de comparaison est basée sur la technique des *N-gram* (cf. section 4.2.1.3). Dans notre utilisation de cette approche, un score est assigné à chaque page web (p), fonction de l'occurrence (occ) de chaque caractère (i) :

$$Score(p) = \sum_{i=1}^{n} occ(i) \cdot valAscii(i)$$

où *valAscii(i)* représente la valeur Ascii de *(i)*

A noter que l'échelle des valeurs attribuées à chaque caractère alphanumérique dans la table Ascii, permet d'utiliser un score différent pour chacun d'entre eux, tout en limitant les écarts créés au sein d'une même classe (p.ex lettres minuscules, lettres majuscules, etc.). Ceci permet donc d'accorder un poids de niveau sensiblement similaire aux différents caractères (cf. tableau 5.4).

Puis, nous calculons le taux de similitude (exprimé en pourcentage) des deux pages (page défaut[1] vs. page de référence[2], ou page légitime vs. page contrefaite) en comparant leurs scores :

$$\text{Taux de similitude} = \frac{\min\{Score(p1), Score(p2)\}}{\max\{Score(p1), Score(p2)\}} \cdot 100$$

où *p1* représente la première page comparée
et *p2* la deuxième page comparée

Nous avons implémenté et testé cette méthode sur des couples de pages de login légitimes-contrefaits (p.ex. HSBC, Paypal, Bank of America, etc.) récupérés grâce au site Phishtank [phi] (pour plus de détails sur la méthode de récupération des pages légitimes-contrefaits, cf. section 5.3.2). Les premiers résultats (cf. tableau 5.5) indiquent un taux de similitude moyen entre 77 et 80% avec un écart-type variant de 15 à 26%.

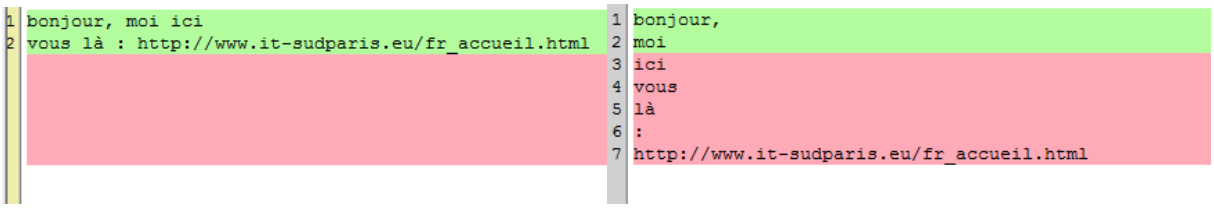
5.2.2.4.2 Approche par mots : Cette méthode de comparaison est basée sur l'approche *diff* qui appartient à la famille des algorithmes de mesure de similarité, traditionnellement utilisés pour la comparaison de textes (cf. section 4.2.1). La méthode de comparaison que nous utilisons ici - nommée *Diff* pour faciliter la lisibilité - implémente la Distance de Levenshtein.

Cette méthode *Diff* permet de calculer un taux de similitude via la comparaison des lignes contenues dans chaque page web. Elle délivre 4 indicateurs : le nombre de lignes ajoutées, modifiées, inchangées ou supprimées.

1. récupérée à partir des adresses IP retournées par le serveur DNS configuré par défaut, c.-à-d. typiquement celui du FAI auquel le client est connecté.
2. récupérée à partir des adresses IP retournées par le serveur DNS de référence utilisé dans notre solution (cf. section 5.3.1 pour plus de détails).

Dans notre approche, suite à nos premiers tests, nous avons appliqué un traitement de découpage des pages en mots (où les délimiteurs de mots sont les espaces), en amont de l'application de la méthode *Diff*. Ainsi, nous obtenons une meilleure précision du calcul de similitude. Nous pallions également tout problème de lisibilité du code source de la page (c.-à-d. le contenu de la page ne risque pas d'être interprété comme écrit sur 1 seule et même ligne par *Diff*).

A noter toutefois que ce découpage par mots ne reflète pas le nombre réel de mots contenus dans la page web (c.-à-d. tel qu'interprété par l'esprit humain). En effet, par exemple, un lien hypertexte n'est vu que comme un seul mot par la méthode de comparaison (cf. figure 5.3).



(a) Texte vu par Diff

(b) Texte vu par Diff, après découpage par mots

FIGURE 5.3 – Exemple de textes sous Diff

Dans notre utilisation de *Diff* incluant le découpage par mots, le score (ou taux de similitude) est exprimé en pourcentage. Il est calculé à partir du nombre de mots ajoutés (ajout), modifiés (modif) et supprimés (suppr). Un poids (c.-à-d. une pénalité) identique est attribué à chaque changement apporté.

$$\text{Taux de similitude (ou Score)} = \frac{(longueur(ref) - modif - suppr - ajout) \cdot 100}{longueur(ref)}$$

où *ref* représente la page de référence,

et *longueur* le nombre de mots total de l'objet spécifié

Pour être précis, il est important de noter que l'appellation "taux de similitude" que nous utilisons pour désigner le score obtenu ici est une appellation erronée. Nous l'utilisons uniquement afin d'améliorer la lisibilité et la comparaison des résultats obtenus avec l'approche par caractères.

En effet, dans le calcul de score obtenu établi ici, nous sommes en mesure d'identifier les zones de changements (ce qui n'est pas le cas avec l'approche par caractères) de manière globale. Nous avons donc délibérément choisi de privilégier le degré de divergence des deux pages comparées, plutôt que leur degré de convergence. Considérons l'exemple montré en figure 5.4 : La partie gauche affiche la page légitime (c.-à-d. la page de référence), tandis que la partie droite affiche la page contrefaite. On constate que par rapport à un total de seulement 8 mots dans le fichier légitime, il y a : 4 mots ajoutés, 1 mot modifié et 1 mot supprimé dans le fichier contrefait. Si on basait notre calcul de score sur une simple mesure de similitude, on obtiendrait un score de 75% (c.-à-d. 6 mots inchangés sur un total de 8 mots). Or, nous considérons que ce calcul ne reflète pas réellement le danger auquel est exposé l'utilisateur. En effet, 50% de code supplémentaire, potentiellement malveillant, ont été ajoutés (c.-à-d. les 4 lignes "script"). Nous préférons donc profiter de la connaissance des changements réalisés et ainsi, grâce au calcul de score que nous avons défini, le résultat de la comparaison est de seulement 25%.

Il est également important de noter que nous avons borné le taux de similitude minimum à 0%, afin d'éviter des scores négatifs aberrants.

Les premiers résultats obtenus, avec cette approche par mots appliquée aux pages légitimes-contrefaites (cf. tableau 5.6), indiquent un taux de similitude moyen variant de 30 à 39 %, avec un écart-type de l'ordre de 30%, sur pages légitimes-contrefaites.

5.2.2.5 Synthèse de l'étude préalable sur le contenu des pages webs

Les caractéristiques des pages webs légitimes et contrefaites, ainsi que les contraintes imposées par une comparaison effectuée depuis le poste client, nous conduisent à une analyse de contenu des codes sources HTML, basée sur des méthodes de comparaison de texte de type mesure de similarité.

Les premiers résultats obtenus sur des couples de pages légitimes-contrefaites semblent indiquer que l'approche par mots donne des résultats plus intéressants concernant le taux de similitude (c.-à-d. les scores les plus bas), tandis que l'approche par caractères donne des écart-types jusqu'à deux fois

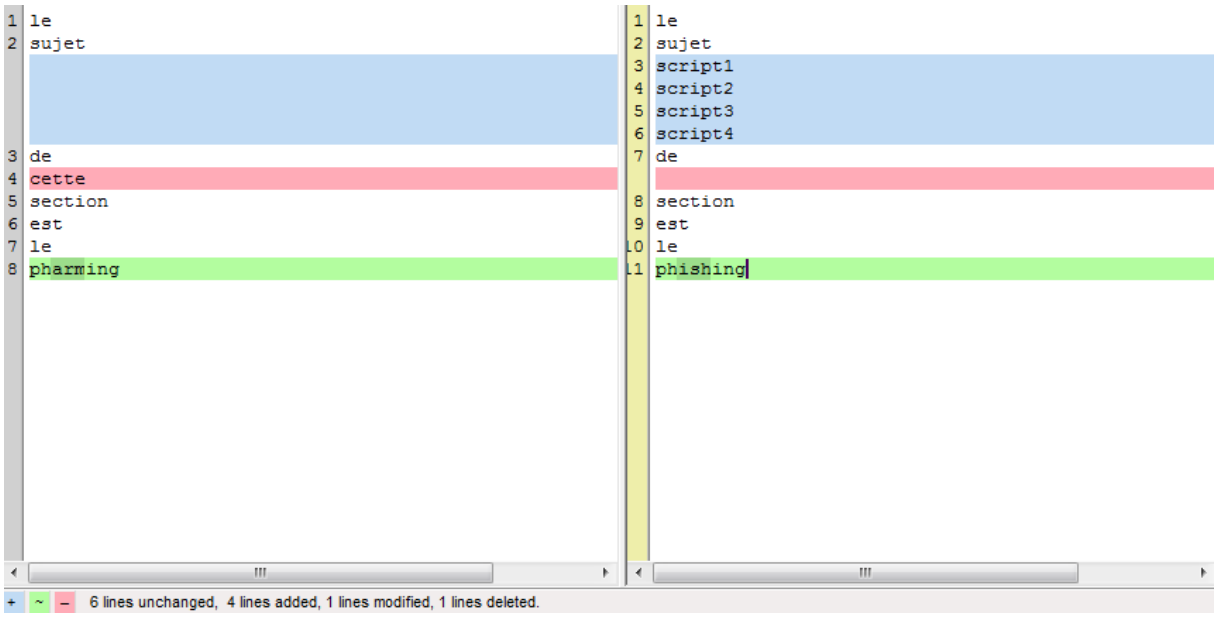


FIGURE 5.4 – Impact des modifications apportées dans le calcul de score, avec l'approche par mots

TABLEAU 5.6 – Taux de similitude de 14 à 23 couples de pages de login légitimes-contrefaites avec l'approche par mots

	Taux de similitude entre page légitime et page contrefaite (min ≤ moyenne ≤ max)	Écart-Type[2]
14 couples de pages[1]	3.16% ≤ **30.78%** ≤ 89.41%	30.20%
23 couples de pages[1]	0% ≤ **39.35%** ≤ 84%	31.63%

[1] Les deux séries de tests portent sur des couples de pages distincts
[2] entre couples de pages

moins élevés. A ce stade, il s'avère impossible de privilégier l'une des deux approches par rapport à l'autre : les valeurs minimales et maximales sont quasiment aussi extrêmes.

Les prochains tests doivent donc vérifier les premiers résultats sur une plus grande base de pages légitimes-contrefaites. Ils doivent également s'intéresser aux performances des deux méthodes de comparaison sur des couples de pages légitimes récupérées depuis une même localisation géographique.

5.3 Première proposition : vers un remplacement du nom de domaine

5.3.1 Fonctionnement général

La première étape de notre proposition (représentée par les points (1) à (4b) sur la figure 5.5) consiste à comparer l'adresse IP du domaine visité à une adresse IP (ou un pool d'adresses IP) dite de référence.

A chaque fois que l'utilisateur accède à une page web de login, le FQDN est extrait de l'URL visitée. Une requête DNS est alors envoyée à deux serveurs DNS : le serveur par défaut (nommé *DNSdef*) et un serveur de référence (nommé *DNSref*), en vue de comparer les adresses IP retournées pour le FQDN concerné. Le serveur DNS par défaut retourne plusieurs adresses IP, dont celle utilisée pour l'affichage de la page web dans le navigateur (nommée(s) *IPdef*). DNSref retourne, quant à lui, une ou plusieurs adresses IP (nommée(s) *IPref*), incluant ou excluant IPdef.

Dans le cas où l'adresse IPdef est incluse dans la réponse IPref, le site est considéré comme légitime. Dans le cas contraire, nous passons à la seconde étape de notre proposition : l'analyse de la page web.

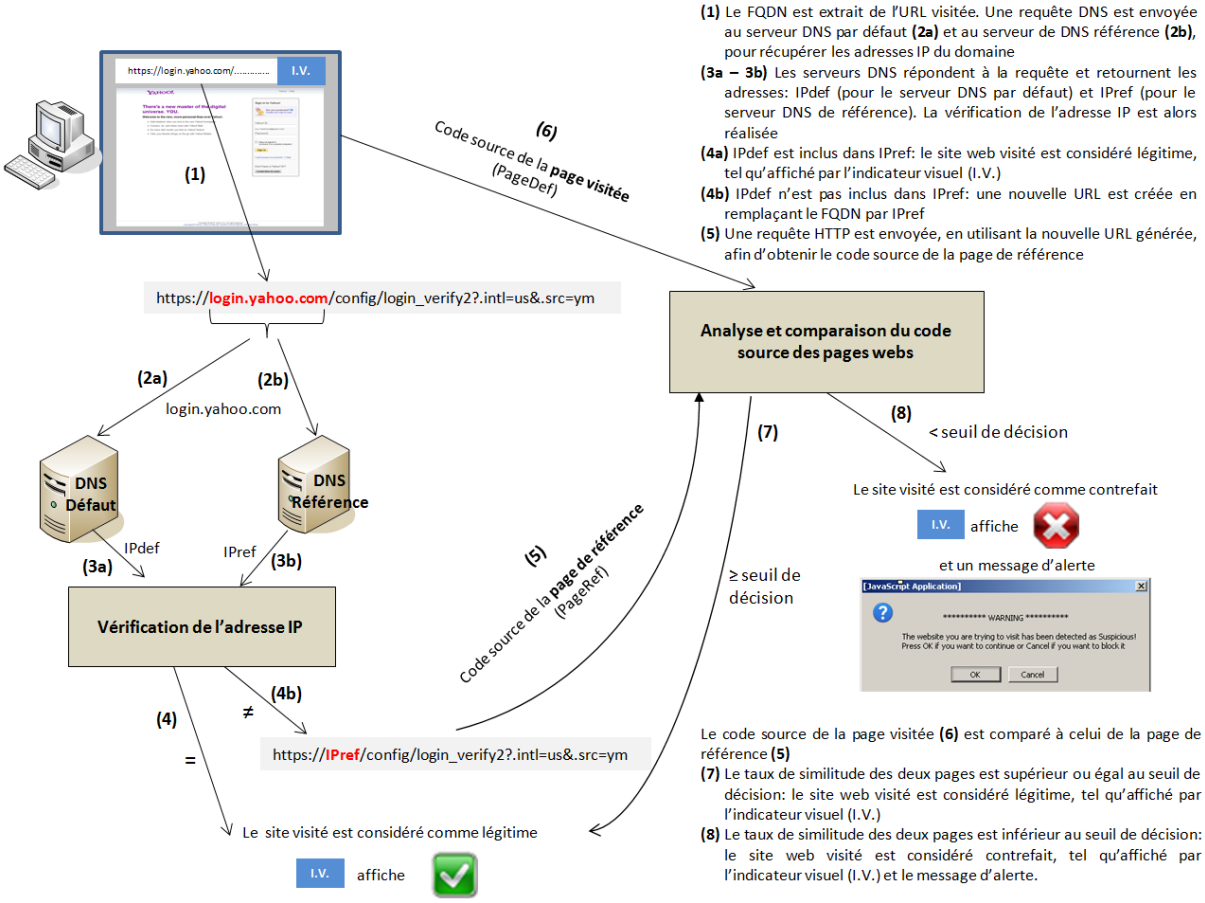


FIGURE 5.5 – Première approche : fonctionnement général

Définition du serveur de référence : Dans le schéma d'intégration envisagé pour notre proposition, nous prévoyons d'inclure une étape de configuration du serveur DNS de référence. Ce choix, laissé à ce stade à l'appréciation de l'utilisateur, se fait parmi une liste de serveurs DNS ouverts alternatifs proposés (p.ex. OpenDNS, GoogleDNS, DNSAdvantage [dns]).

La seconde étape de notre proposition se concentre sur l'analyse et la comparaison de la page web visitée.

L'étude de travaux précédents sur la comparaison de pages webs dans le cadre du phishing [WHX+05] [RKKF07] [MKK08] [HYM09], nous a montré que ces comparaisons sont essentiellement basées sur des bases de données de référence. Les inconvénients majeurs de ce type d'approche sont à la fois le maintien d'une base de données de référence à jour, ainsi que la préservation de sa confidentialité et de son intégrité.

Dans notre approche - de la même manière que nous avions écarté les listes noires dans l'analyse du phishing -, nous avons donc considéré comme essentiel que, bien qu'utilisant une comparaison de page web envers une page dite de référence, il ne devait y avoir aucun stockage de ces pages considérées comme légitimes.

La seconde étape de notre proposition se déroule donc de la manière suivante (représentée par les points (5) à (8) sur la figure 5.5) :

- la page web visitée (nommée *PageDef*), telle qu'affichée dans le navigateur, est récupérée à partir de IPdef.
- La page de référence (nommée *PageRef*) est, quant à elle, récupérée grâce à une nouvelle URL générée en utilisant IPref. En effet, le FQDN de l'URL d'origine est alors remplacé par IPref. Considérons l'exemple suivant : l'URL visitée par l'Internaute via son navigateur est `https://www.amazon.com/gp/yourstore?ie=UTF8&ref_=pd_irl_gw&signIn=1`. Le serveur DNS par défaut de

l'utilisateur retourne l'adresse IP : 72.21.210.250 (IPdef), tandis que le serveur DNS de référence interrogé retourne l'adresse IP : 207.171.166.252. La nouvelle URL utilisée pour la récupération de la page référence (PageRef) est alors : `https://207.171.166.252/gp/yourstore?ie=UTF8&ref_=pd_irl_gw&signIn=1`.

Les codes sources HTML des deux pages webs (PageDef et PageRef) sont ensuite comparés (cf. figure 5.5) en utilisant une des techniques énoncées en section 5.2.2. Enfin, la légitimité de la page web visitée est déterminée en comparant le pourcentage de similitude obtenu entre les deux pages (PageDef et PageRef) et un seuil de décision pré-défini.

Les sections suivantes de ce Chapitre 5.3 visent à détailler les expérimentations réalisées, ainsi qu'à exposer les résultats obtenus.

5.3.2 Conditions d'expérimentation

Pour évaluer l'efficacité de notre proposition, nous avons réalisé deux types de comparaison de pages webs :

- des comparaisons de couples de pages de login[1] légitimes récupérées, à partir des informations émises par plusieurs serveurs DNS, depuis différentes localisations géographiques.
- des comparaisons de couples de pages de login[1] légitimes-contrefaites, visuellement très similaires.

L'ensemble des pages utilisées pour ces tests ont été collectées entre Décembre 2010 et Janvier 2011.

5.3.2.1 Couples de pages légitimes

Les couples de pages de login légitimes ont été récupérées depuis 11 localisations géographiques différentes, réparties sur 5 continents.

Pour chaque localisation, nous avons récupéré les pages webs obtenues grâce aux adresses IP retournées par le serveur par défaut ainsi que par 3 serveurs de référence (OpenDNS, GoogleDNS et DNSAdvantage).

Sur chaque site géographique, nous avons testé un set de 108 URLs de login, dont 104 sont issues de FQDN distincts. Les principaux critères de sélection de ces URLs ont été : diversifier les secteurs d'activités, multiplier les TLD et les langues dans lesquelles les pages sont écrites (anglais, français, espagnol, arabe, russe, etc.). La majorité des URLs de sites bancaires ont été récupérées à partir du site *Levoyageur* [lev].

Classification des 108 URLs de login : Nous avons classifié les URLs utilisées en 5 secteurs d'activités : banques, réseaux sociaux, e-commerce, email, et autres (cf. Table 5.7). La catégorie *autres* comprend des sites provenant des domaines de l'assurance, l'administration, les jeux en ligne, les universités, les plateformes de partage vidéos et photos, des sites d'information, des sites de support produits logiciels ou divers sites issus de l'industrie (p.ex. constructeur automobile, panneaux solaires, etc.).

A noter que nous avons constaté que plusieurs sites, tels que celui de Facebook (réseaux sociaux) semblent utiliser une connexion non sécurisée pour leur page d'authentification (celle-ci s'affiche en *http*, p.ex. `http://fr-fr.facebook.com/`). Néanmoins, à l'aide d'un analyseur de protocoles, nous avons constaté que l'envoi des données d'authentification de l'abonné s'effectue pourtant bien de manière sécurisée. Cet envoi se fait de façon totalement transparente pour l'utilisateur, via l'URL `https://login.facebook.com/`. Dans ce type de cas, nous avons donc préféré intégrer l'URL réellement utilisée pour l'envoi des données (c.-à-d. celle en HTTPS) dans notre set de 108 URLs.

Les URLs sélectionnées sont issues de 20 TLDs différents, divisables en 2 catégories : les cc-TLD et les g-TLD. Pour une meilleure lisibilité, les TLDs sélectionnés ont été regroupés par continent (cf. Table 5.8).

Une large majorité des sites sélectionnés utilisent le TLD générique COM (pour *Commercial*), suivis par les TLDs d'Europe, d'Asie, d'Océanie et le TLD générique NET (pour *Network*). Enfin, quelques TLDs sont issus d'Amérique du Nord, d'Amérique du Sud, d'Afrique et du TLD générique ORG (pour *Organization*).

1. Les raisons de la focalisation de notre étude sur les pages de login sont expliquées en section 5.2.1.5

Tableau 5.7 – Répartition des 108 URLs de login légitimes par secteur d'activités

Catégories	Quantité	Pourcentage
Banques	53	49%
Autres (administration, assurances, logiciels, jeux, FAI, industrie, vidéos, photos, informations)	37	34%
e-commerce	8	7%
Réseaux sociaux	5	5%
email	5	5%

Tableau 5.8 – Répartition des 20 TLDs des 108 URLs de login légitimes

	Pourcentage	TLD
Europe	37.0%	AD, AT, BE, DE, DK, FR, IT, LV, UK
Asie	5.6%	IN, MV, TR
Océanie	3.7%	AU, NZ
Amérique du Nord	0.9%	CA
Amérique du Sud	0.9%	MX
Afrique	0.9%	ZA
Commercial	46.3%	COM
Network	3.7%	NET
Organization	1.0%	ORG

Répartition géographique des tests : Les tests ont été effectués depuis 11 localisations géographiques, réparties sur 5 continents (cf. figure 5.6) :

- Europe : France (4 tests en Île de France et 1 test dans le Sud de la France)
- Amérique du Nord : Mexique
- Amérique du Sud : Vénézuela
- Asie : Chine, Émirats Arabes Unis
- Afrique : Tunisie

Pour chaque localisation, nous nous sommes assurés que la configuration DNS par défaut était différente des 3 DNS de référence que nous avons sélectionnés.

La récupération des pages webs auprès des 4 serveurs DNS (c.-à-d. le serveur DNS par défaut et les 3 serveurs de référence) a été automatisée grâce au programme que nous avons développé en Java (cf. section 5.3.3). Il est important de noter que lors des tests, sur certaines localisations géographiques (p.ex. Sénégal, Vénézuela), nous avons rencontré des pertes de connectivité (rupture de courant électrique et/ou Internet) assez fréquentes. Par conséquent, les taux d'erreur de récupération des pages sur ces localisations sont notablement plus conséquents que pour les autres localisations testées.

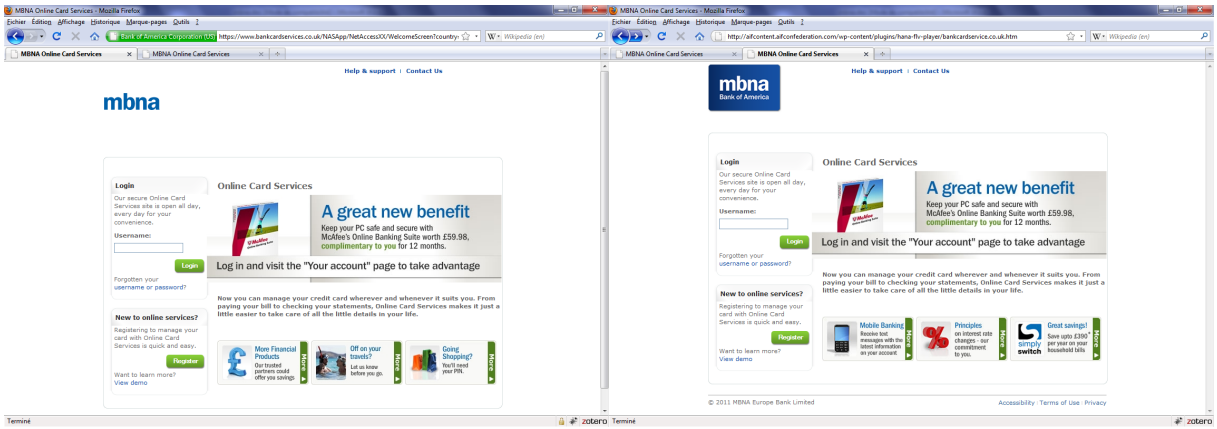
5.3.2.2 Couples de pages légitimes-contrefaites

Une des difficultés essentielles de notre expérimentation est de trouver des pages contrefaites réelles et en ligne (en effet leur durée de vie est très courte, cf. section 2.3) pouvant servir pour des attaques de pharming, ainsi que leur pendant légitime. Nous nous sommes donc appuyés sur les sites de l'APWG [apw] et de Phishtank [phi] qui délivrent des bases d'URLs de phishing pour récupérer nos pages contrefaites. Nous avons choisi de nous placer dans le cas le plus défavorable, en basant notre sélection sur ces 2 critères :

- Sélectionner des sites de phishing "validés" à 100%. En effet, les bases de données communiquées par l'APWG et Phishtank indiquent un niveau de confiance (c.-à-d. il est avéré et vérifié que l'URL présentée est un site de phishing), variant de 50 à 100% pour les URLs mises à disposition.
- Ne sélectionner que des sites de phishing pour lesquels nous trouvons le site légitime associé, visuellement très similaire (c.-à-d. même structure visuelle, cf. exemple en figure 5.7).

Nous avons ainsi collecté 37 couples de pages légitimes-contrefaites, chaque couple de page ayant été récupéré depuis une même machine et un même navigateur web.

FIGURE 5.6 – Première approche : répartition géographique des tests



(a) Site légitime (b) Site contrefait

FIGURE 5.7 – Captures d'écran du site légitime MBNA (`https://www.bankcardservices.co.uk/NASApp/NetAccessXX/...`) et d'une contrefaçon (`http://aifcontent.aifconfederation.com/wp-content/plugins/hana-flv-player/bankcardservice.co.uk.htm`), récupérées le 11 Juin 2011

Algorithme 1 Première approche : DNS par défaut

Entrées: le fichier *.txt* contenant la liste des n URLs de login.
Sorties: 1 fichier contenant les n réponses du serveur par défaut, 1 fichier contenant les scores des n pages par défaut, et n fichiers contenant les codes sources des pages webs par défaut.

1: **pour** $i = 0$ **à** n **faire**
2: Requête DNS auprès du serveur par défaut pour le FQDN(i).
3: Sauvegarde des adresses IP (*IPdef*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
4: Récupération et sauvegarde du code source de la page web par défaut en utilisant l'URL `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html` (1 fichier par URL, nommé *date_heure_def_FQDN*(i)*.txt*).
5: Calcul du score de la page(i) récupérée avec l'approche par caractères, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
6: **fin pour**

L'ensemble des pages contrefaites ainsi collectées nous ramène à nos 5 catégories de secteurs d'activités (vus pour les pages légitimes) : majoritairement des banques (p.ex. Bank of America, Paypal, Chase, etc.), des sites d'e-commerce (p.ex. eBay, etc.), d'email (p.ex. Hotmail), de réseaux sociaux (p.ex. Facebook) ou autres (p.ex. jeux en ligne avec RuneScape).

5.3.3 Implémentation : aperçu général

L'implémentation de notre approche, visant à récupérer les couples de pages légitimes, a été réalisée en Java. Un certain nombre de fonctions pré-existantes, assez facilement utilisables, sont disponibles dans les librairies Java (p.ex. *java.net*), tant pour générer une requête DNS vers le serveur DNS par défaut que pour récupérer une page web en HTTP.

Une des premières difficultés rencontrées dans l'implémentation a été de pouvoir générer une requête DNS vers un serveur dit de référence. Pour ce faire, nous avons développé une fonction spécifique (cf. section 5.3.4.1). En outre, plusieurs mois d'essais ont été nécessaires pour éliminer toute erreur (ou presque) liée à cette seconde requête DNS de référence (cf. section 5.3.5.3).

Une deuxième étape importante a été de savoir quelle adresse avait été utilisée pour récupérer la page web. En effet, le DNS retourne les adresses connues pour le domaine interrogé, à un instant t. Mais au moment de la requête HTTP, ce n'est pas forcément la première adresse IP retournée par le serveur DNS qui est utilisée. Dans la seconde approche (développée en section 6.1), nous avons pu répondre avec certitude à cette question.

Une troisième difficulté rencontrée a été de pouvoir récupérer une page de login en HTTPS. Il fallait en effet pouvoir passer les étapes d'établissement de la connexion SSL/TLS avant de récupérer la page web requise.

Enfin, une quatrième étape a concerné la réécriture de l'URL d'origine, dans laquelle le FQDN est remplacé par l'adresse IP *IPref*.

Dans cette première approche, seul le calcul de score par caractères a été calculé automatiquement, au téléchargement des pages webs. L'approche par mots, ajoutée plus tardivement, a été appliquée aux pages séparément, après téléchargement.

La liste des URLs à récupérer est placée dans un fichier *.txt*, ordonnancé par secteur d'activités des URLs.

5.3.3.1 Traitement effectué durant les tests

Cette étape consiste en la récupération des adresses IP *IPdef* et *IPref*, des pages webs associées, et au calcul de score utilisant l'approche par caractères.

Suite à des problèmes d'implémentation, nous avons été obligés de scinder les traitements associés à DNSdef et DNSref en deux programmes séparés (cf. algorithmes 1 et 2), pour les raisons évoquées ultérieurement dans ce document (cf. section 5.3.4.3).

Algorithme 2 Première approche : DNS de référence

Entrées: le fichier *.txt* contenant la liste des n URLs de login.
Sorties: 1 fichier contenant les n réponses du serveur de référence, 1 fichier contenant les scores des n pages de référence, et n fichiers contenant les codes sources des pages webs de référence.

1: **pour** $i = 0$ **à** n **faire**
2: Requête DNS auprès du serveur de référence pour le FQDN(i).
3: Sauvegarde des adresses IP (*IPref*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
4: Récupération et sauvegarde du code source de la page web de référence en utilisant l'URL `https://IPref/arborescence1/.../arborescenceX/fichier.html` (1 fichier par URL, nommé *date_heure_nomDNSref_FQDN*(i)*.txt*). A noter que l'adresse IPref utilisée est la première adresse IP retournée par le serveur DNS de référence.
5: Calcul du score de la page(i) récupérée avec l'approche par caractères, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
6: **fin pour**

Ensuite, le taux de similitude des deux pages est calculé sous Excel, à l'aide des deux fichiers de score générés précédemment (selon la méthode expliquée en section 5.2.2.4.1).

5.3.3.2 Traitement effectué après les tests

Cette étape consiste au traitement des pages webs récupérées précédemment (*PageDef* et *PageRef*) avec l'approche par mots.

Pour ce faire, nous avons été obligés de procéder au réordonnancement des fichiers de pages webs, afin de pouvoir procéder à des comparaisons justes et comparables aux scores obtenus avec l'approche par caractères. En effet, lors de la sauvegarde du code HTML, nous avions choisi d'horodater les fichiers, en incluant la date et l'heure de récupération, afin de faciliter l'archivage des pages récupérées (p.ex. *20110126_15.01.18_def_login.live.com.txt* est le nom de fichier pour l'URL contenant le FQDN *login.live.com* récupéré auprès du serveur par défaut *Def*, le 26/01/2011 à 15h01min18s.). Or, dans notre implémentation, nous récupérons l'ensemble des URLs auprès d'un même serveur DNS, avant de passer au DNS suivant. Les temps de réponse des serveurs webs interrogés pouvant être très variables, nous avons sauvegardé les pages webs dans un ordre parfois très différent de celui dans lequel les requêtes avaient été émises. Nous avons pallié ce problème en supprimant l'indication horaire dans les noms de fichiers sauvegardés. Ainsi l'ensemble des fichiers - quel que soit le serveur DNS interrogé - ont été classés, à l'identique, par ordre alphabétique des FQDNs. A noter que les développements menés lors de la seconde approche ont corrigé ce problème.

L'approche par mots telle que nous l'avons envisagée (cf. section 5.2.2.4.2) a été implémentée en trois parties :

- La première partie consiste en une méthode de découpage par mots des pages webs, dans laquelle les délimiteurs sont des espaces. Nous avons développé cette méthode à l'aide de la classe Java *TextIO*.
- La deuxième partie consiste en la comparaison des pages et l'identification des zones de changements (ajouts, modifications, suppressions). Pour ce faire, nous avons utilisé une implémentation Java de *Diff* pré-existante [Gat].
- Enfin, la troisième partie est le calcul de score réalisé grâce à l'extraction et à la réutilisation des zones de changements identifiées précédemment.

Contrairement à l'approche par caractères, un seul fichier de score est généré : celui contenant le taux de similitude des deux pages.

A noter que les analyses par mots et par caractères ont toutes deux été appliquées aux couples de pages légitimes-contrefaites selon les mêmes principes que décrits ici.

5.3.4 Vérification de l'adresse IP du domaine visité

Les résultats de tests de la vérification de l'adresse IP se concentrent sur les FQDN uniques, à savoir : 104 des 108 URLs légitimes sélectionnées (cf. section 5.3.2.1).

5.3.4.1 Implémentation : points spécifiques

Les requêtes vers le serveur DNS par défaut sont faites grâce à la fonction Java *getAllByName*, présente nativement dans la classe *InetAdress*.

Les requêtes vers le serveur DNS de référence sont quant à elles réalisées grâce à l'utilisation d'une librairie particulière, DNSJava [lib]. Cette bibliothèque, qui est une implémentation de DNS en Java, propose des fonctions similaires à la classe InetAddress. En supplément, elle apporte des fonctions additionnelles telles que le transfert de zone, DNSSEC, etc. Pour forger une requête DNS, on utilise un objet *SimpleResolver* qui représente le serveur DNS à contacter (il contient le nom du serveur, son adresse IP et le port destination (53)). Ensuite, on créé un *Lookup* qui correspond aux caractéristiques de la requête à envoyer. Elle contient le domaine interrogé, le type d'enregistrement demandé ("A" pour nom d'hôte/adresse IPv4) et sa classe ("IN" pour protocole Internet). Reste alors à affecter le *Lookup* au *SimpleResolver*, puis lancer la requête.

5.3.4.2 Résultats

5.3.4.2.1 Variabilité du nombre de réponses DNS : L'exploitation des résultats DNS s'est avéré difficile. En effet, selon les sites testés, les DNS interrogés, les heures et dates des tests, etc., une à plusieurs adresses IPs peuvent être retournées par chacun des serveurs DNS.

Difficile donc d'automatiser une comparaison, puisqu'aucune indication préalable ou certitude n'était acquise sur les tailles des espaces d'adresses IP récupérés. Néanmoins, dans la seconde approche développée en section 6.1, nous avons réussi à optimiser ce traitement.

5.3.4.2.2 Variabilité des adresses IP utilisées par les pages de login : Les résultats IP énoncés dans le tableau 5.9 sont obtenus en vérifiant si au moins 1 adresse *IPdef* figure dans les résultats *IPref* du serveur de référence. Prenons trois exemples, sur une même localisation donnée (France) :

- Pour le FQDN `login.yahoo.com`, DNSdef retourne les adresses IP 217.12.8.76 et 217.146.187.123, tandis que DNSref retourne l'adresse IP 69.147.112.160. Dans ce cas, nous considérons une convergence de 0.
- Pour le FQDN `www.paypal.com`, DNSdef retourne les adresses IP 66.211.169.65, 64.4.241.33, 64.4.241.49 et 66.211.169.2, tandis que DNSref retourne les adresses IP 64.4.241.49, 66.211.169.2, 66.211.169.65 et 64.4.241.33. Ici, la convergence est de 4.
- Pour le FQDN `www.halifax-online.co.uk`, DNSdef retourne l'adresse IP 212.140.245.71, tandis que DNSref retourne les adresses IP 62.172.43.199, 212.140.245.11, 62.172.43.131 et 212.140.245.71. Dans ce cas, la convergence est de 1.

Le taux de convergence global est ensuite obtenu de la manière suivante : pour l'ensemble des FQDN interrogés, nous additionnons l'ensemble des convergences ≥ 1, que nous divisons par le nombre total de FQDN pour lesquels nous avons des réponses DNS (c.-à-d. absence d'erreurs DNS).

Les résultats obtenus tendent à indiquer que, comme supposé en section 5.2.1.5, les adresses IP utilisées par des sites de login sont plus convergentes que pour les FQDN génériques. Pour rappel, la moyenne de convergence des adresses IP sur les FQDN génériques était de l'ordre de 71 à 73% avec des écart-types (entre localisations) oscillants entre 10 et 14% (cf. section 5.2.1). Ici, l'écart-type (entre localisations) est considérablement réduit (3 à 5%), les moyennes des taux de convergence oscillent autour de 81 à 82%. Enfin, l'intervalle de confiance à 95% (c.-à-d. l'incertitude d'estimation, entre localisations) varie entre 79 à 85%. Nous constatons que les 3 DNS de référence donnent le même type de résultats, avec un léger avantage de convergence pour OpenDNS.

Il est important de noter que ces résultats ne signifient par pour autant, qu'en moyenne, 81 à 82% des pages HTML ont été récupérées auprès du même serveur web (c.-à-d. même adresse IP). En effet, à ce stade, nous n'avons aucune certitude sur l'adresse IP utilisée pour la récupération de PageDef, de par la fonction utilisée (cf. section 5.3.5.3).

Tableau 5.9 – Comparaison des réponses du DNS par défaut vs. 3 serveurs DNS de référence, sur 11 localisations géographiques

	Taux de convergence avec les adresses IP retournées par le serveur DNS par défaut (min ≤ moyenne ≤ max)	Écart-Type [1]	Intervalle de confiance à 95% [1]
OpenDNS	76.47% ≤ **82.90%** ≤ 95.19%	5.00%	[79.95% ; 85.85%]
GoogleDNS	76.92% ≤ **81.61%** ≤ 91.35%	3.62%	[79.47% ; 83.75%]
DNSAdvantage	74.51% ≤ **81.22%** ≤ 89.42%	4.10%	[78.80% ; 83.64%]

[1] entre localisations

Tableau 5.10 – Taux d'échec des requêtes DNS auprès des 4 serveurs DNS, sur 11 localisations géographiques

	Taux d'échec des requêtes DNS (min ≤ moyenne ≤ max)	Écart-Type [1]
Défaut	0% ≤ **0.08%** ≤ 0.96%	0.29%
OpenDNS	0% ≤ **0.16%** ≤ 1.92%	0.58%
GoogleDNS	0% ≤ **0.80%** ≤ 9.62%	2.90%
DNSAdvantage	0% ≤ **0.24%** ≤ 0.96%	0.45%

[1] entre localisations

5.3.4.2.3 Taux d'échec des requêtes DNS : Sur les 11 localisations testées, nous obtenons des taux d'erreur des requêtes DNS très faibles (cf. tableau 5.10). En effet, nous obtenons au maximum 1 à 2 erreurs (sur les 104 URLs testées) par site géographique. Une exception est à noter toutefois : sur le site du Vénézuela, nous atteignons un taux d'échec de 9.62% avec GoogleDNS, ce qui représente 10 erreurs. Ceci s'explique par les pertes de connectivité rencontrées lors de la réalisation des tests (cf. explications en section 5.3.2.1).

5.3.4.3 Problème majeur rencontré à l'implémentation : la réinitialisation du cache DNS

Le problème majeur que nous avons rencontré à l'implémentation a concerné la possibilité de faire des requêtes vers différents serveurs DNS, au sein d'un même programme Java. En effet, les requêtes vers le DNS par défaut se déroulaient sans aucun problème mais, dès lors que nous faisions appel à un second serveur DNS, nous étions confrontés à des comportements aléatoires. Grâce à une analyse du trafic, nous nous sommes aperçus que les nouvelles requêtes DNS étaient aléatoirement générées. Ainsi, nous utilisions parfois les réponses DNS du serveur par défaut en tant qu'adresse *IPref*, ce qui faussait une partie de nos résultats.

Confrontés à un problème de réinitialisation du cache DNS, nous avons essayé de trouver son origine, à savoir : problème de configuration du système d'exploitation ou problème Java.

Dans un premier temps, nous avons identifié un paramètre susceptible de permettre la réinitialisation du cache DNS Java : *networkaddress.cache.ttl*.

Trois méthodes ont donc été testées pour permettre la réinitialisation du cache DNS Java :

Méthode N° 1 - Ajout d'une commande de réinitialisation du cache au sein du programme Java : Le cache *networkaddress.cache.ttl* peut être configuré selon trois types de valeur : 0 = absence de cache, -1 = cache infini, >0 = valeur de cache en secondes. Nous avons donc forcé la valeur du cache à 0. A l'exécution du programme nous avons cependant constaté que la valeur de cache restait inchangée à sa valeur par défaut : 30 s.

Méthode N° 2 - Modification du paramètre de cache dans le logiciel Java : La modification du cache *network.cache.ttl* est possible par manipulation du fichier *java.security*, accessible dans le réper-

Algorithme 3 Programme pour l'identification du Bug DNS Java

1: Récupération de la page web par défaut en utilisant l'URL `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html`.
2: Requête vers le DNS Référence afin de récupérer *IPref*.
3: Récupération et sauvegarde du code source de la page web de référence en utilisant l'URL `https://IPref/arborescence1/.../arborescenceX/fichier.html`.

toire d'installation %JRE_HOME%/lib/security/. A nouveau, malgré un cache paramétré à 0, nous avons constaté que celui demeurait toujours bloqué à 30 s.

Méthode N° 3 - Positionnement de la valeur de cache, en ligne de commande, à l'exécution du programme : Le cache peut être forcé à 0 via la commande *java -jar -Dsun.net.inetaddr.ttl=0 nom_programme.jar*. Là encore, aucune amélioration à l'exécution du programme.

Après investigation du côté des problèmes connus, nous avons identifié un bug Java susceptible de correspondre à notre problème [Ora05]. En effet, il met en avant l'inefficacité de la réinitialisation du cache DNS. Toutefois ce bug est mentionné comme "State 11-Closed, Not a Defect, bug". Il est ramené à un problème de paramétrage du cache DNS au niveau du système d'exploitation.

Toutefois, via une implémentation alternative, nous avons bel et bien constaté que ce problème est inhérent au Java.

En effet, théoriquement, une implémentation logique serait d'intégrer la totalité de notre programme en un seul et unique exécutable Java, constitué de 4 étapes successives : 1/ Requête DNSdef, 2/ Récupération de PageDef, 3/ Requête DNSref et 4/ Récupération de PageRef. Néanmoins, pour les raisons évoquées précédemment, cela s'avère impossible. Afin de faire la distinction entre bug Java ou paramétrage du cache DNS au sein du système d'exploitation, nous avons créé un programme alternatif (cf. algorithme 3). Dans celui-ci, la requête DNSdef n'est pas explicitement demandée par le programme Java, mais automatiquement générée par le fonctionnement normal de la pile TCP/IP induit par l'étape 1 de cet algorithme. Nous constatons alors que la requête DNSref, lancée à l'étape 2 de l'algorithme, fonctionne parfaitement.

Le problème rencontré est donc bien lié à un problème de réinitialisation du cache associé à la fonction *java.net.InetAddress.getByName()* utilisée pour la requête au DNS par défaut.

Il aurait alors paru simple d'appliquer la librairie DNSJava, utilisée pour la requête DNSref, à la requête DNSdef (cf. section 5.3.4.1). Toutefois, cela s'avère difficile. En effet, les fonctions proposées impliquent de spécifier l'adresse IP du DNSdef. Ceci parait difficilement utilisable en environnement réel, c.-à-d. demander à un utilisateur de spécifier l'adresse de son DNSdef. De plus, cela devient problématique en cas de mise à jour automatique de l'adresse DNSdef par le FAI.

Pour les deux approches développées dans notre contribution, nous avons donc été amenés à scinder le programme en plusieurs morceaux. Nous avons ainsi créé un exécutable par serveur DNS interrogé, afin d'éliminer tout problème de cache DNS Java.

5.3.4.4 Synthèse sur la vérification de l'adresse IP

Au travers des résultats obtenus dans cette première proposition, nous confortons les hypothèses de travail initiales, à savoir : une certaine staticité des adresses IP associées aux pages de login. Ceci peut notamment s'expliquer par la gestion de certificat associée à un FQDN, impliquant la pré-définition d'adresses IP réservées à cet usage.

Les trois serveurs DNS de référence interrogés donnent sensiblement le même type de résultats.

Les taux d'échec des requêtes DNS sont minimes et ce, quel que soit le serveur utilisé (les résultats GoogleDNS sont impactés par les pertes de connectivité rencontrées au Vénézuela. En effet, les 10 autres localisations ne présentent aucun échec).

Nous constatons également que le nombre de réponses DNS n'est pas forcément directement lié à l'idée que nous nous faisons de la taille du domaine interrogé.

Tableau 5.11 – Taux de similitude des 108 pages légitimes avec l'approche par caractères, sur 11 localisations géographiques

	APPROCHE PAR CARACTÈRES		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	98.14% ≤ **98.63%** ≤ 98.96%	0.26%	[98,48% ; 98.78%]
GoogleDNS	98.14% ≤ **98.68%** ≤ 98.96%	0.27%	[98,52% ; 98.84%]
DNSAdvantage	98.22% ≤ **98.71%** ≤ 99.26%	0.32%	[98.52% ; 98.90%]

[1] entre localisations

Nous en déduisons donc que la vérification de l'adresse IP semble une étape valable dans notre processus d'identification des attaques de pharming. Néanmoins, les résultats obtenus ici doivent être vérifiés sur davantage d'URLs. De plus, cette étape ne peut être le seul indicateur de légitimité du site visité.

Enfin, dans l'intégration envisagée pour notre solution, le problème identifié pour la réinitialisation du cache DNS Java peut s'avérer problématique. Ce point devra donc être étudié avec une attention toute particulière.

5.3.5 Analyse et comparaison du code source des pages webs

Les tests de cette partie portent sur la totalité des 108 URLs légitimes sélectionnées (cf. section 5.3.2.1).

5.3.5.1 Implémentation : points spécifiques

Contrairement à l'étude préalable effectuée, qui portait sur des URLs HTTP, nous nous sommes focalisés ici sur la comparaison d'URLs de login en HTTPS. Ceci implique de passer avec succès les étapes d'établissement de la connexion SSL/TLS, avant de pouvoir récupérer la page web souhaitée.

En préambule du GET HTTP (qui récupère la page web), nous avons donc implémenté une fonction spécifique *https3.gethttps* qui réalise l'établissement de la connexion sécurisée grâce à la librairie Java *javax.net.ssl*. Les deux éléments essentiels de cette fonction sont : l'acceptation de tout certificat proposé, ainsi que la non-vérification du "hostname" (c.-à-d. le FQDN).

Une autre étape importante est la génération de la nouvelle URL utilisant IPref. Pour ce faire, nous avons utilisé la première adresse IP retournée par le serveur DNS de référence.

5.3.5.2 Résultats

5.3.5.2.1 Taux de similitude des pages légitimes : Les tableaux 5.11 et 5.12 indiquent les scores obtenus avec les deux approches, pour la comparaison des pages issues de IPdef vis-à-vis de celles fournies par IPref. Il apparaît que l'approche par caractères semble donner les meilleurs résultats : le taux de similitude y est le plus élevé (en moyenne 98 %) et les écart-types entre localisations les plus minimes (autour de 0.30%). Néanmoins, l'approche par mots donne également de très bon résultats avec un taux de similitude autour de 91% et des écart-types entre localisations assez faibles (autour de 3%).

Nous constatons également peu de variabilité des résultats associés à la provenance de l'adresse IPref. Autrement dit, les 3 serveurs DNS conduisent à des résultats très similaires.

5.3.5.2.2 Taux de similitude des pages légitimes-contrefaites : Concernant la différenciation des pages légitimes-contrefaites, il apparaît très clairement que la méthode par mots est la plus intéressante (cf. tableau 5.13). En effet, c'est l'approche qui donne les scores les plus intéressants (c.-à-d. les plus bas) avec un taux de similitude moyen de 56%, contre 80% avec l'approche par caractères. En comparaison avec les résultats de l'étude préalable, on constate également que l'approche par caractères perd en avantage sur l'écart-type : nous obtenons ici 23% pour cette dernière et 27% pour l'approche par mots.

Tableau 5.12 – Taux de similitude des 108 pages légitimes avec l'approche par mots, sur 11 localisations géographiques

	APPROCHE PAR MOTS		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	81.96% ≤ **91.56%** ≤ 93.57%	3.24%	[89.65% ; 93,51%]
GoogleDNS	81.61% ≤ **91.50%** ≤ 94.53%	3.46%	[89.46% ; 91.50%]
DNSAdvantage	82.20% ≤ **91.38%** ≤ 92.76%	3.13%	[89.53% ; 93.23%]

[1] entre localisations

Tableau 5.13 – Taux de similitude des 37 couples de pages légitimes-contrefaites

	Taux de similitude entre pages légitimes et contrefaites (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]	Taux de faux-négatifs si le seuil de décision est 80%	90%	Approche déterminante pour % couples de sites
CARACTÈRES	19.57% ≤ **80.37%** ≤ 99.93%	23%	[72.95% ; 87.79%]	59.46%	24.32%	2.70%
MOTS	2% ≤ **56.76%** ≤ 92%	27.42%	[29.34% ; 65.59%]	2.70%	-	97.30%

[1] entre couples de pages

Les taux de similitude maximum obtenus avec les deux approches semblent assez voisins (92 ou 99%). Néanmoins, dans l'objectif de définir un seuil de décision, nous avons commencé à nous intéresser au taux de faux-négatifs (FNR - False Negative Rate) potentiel. Il apparaît alors qu'avec un seuil fixé à 80%, un fossé se creuse entre les deux méthodes. Avec l'approche par caractères, 59.46% des pages contrefaites seraient déclarées légitimes à tort, contre seulement 2.70% avec l'approche par mots. En positionnant le seuil à 90%, il n'y aurait plus aucun faux-négatif avec l'approche par mots, contre encore 24.32% avec l'approche par caractères.

Enfin, nous avons cherché à savoir quelle méthode donnait le score le plus bas, pour chaque couple de page légitime-contrefaite. Il apparaît alors que l'approche par mots est déterminante pour 97.30% des sites (soit 36 couples de sites sur 37), contre seulement 2.70% des sites avec l'approche par caractères (soit 1 couple de sites sur 37).

Ces résultats de meilleure qualité, obtenus avec l'approche par mots, s'expliquent notamment par la possibilité d'identifier les zones de modifications entre pages légitimes et contrefaites, ainsi que par le poids accordé aux changements.

A contrario, avec l'approche par caractères, il faut qu'une partie conséquente de script soit ajoutée - par rapport à la taille de la page légitime - afin qu'elle soit détectée. De plus, les modifications de script sont moins détectables si les changements apportés tournent toujours autour des mêmes caractères.

5.3.5.2.3 Taux d'erreur de récupération des pages Défaut et Référence : Au-delà de l'efficacité de la méthode de comparaison utilisée sur la comparaison de contenu des pages, nous avons voulu évaluer son efficacité en terme de récupération des pages. En effet, si nous trouvons une méthode de comparaison de contenu efficace mais que notre technique de récupération des pages ne fonctionne que pour une faible proportion des URLs, l'analyse perd tout son intérêt.

Nous avons donc cherché à évaluer le taux d'échec de récupération des pages, auprès de chaque adresse IP utilisée : IPdef ou les 3 adresses IPref (cf. tableau 5.14).

Avec l'adresse IPdef, le taux d'échec moyen est de 2.19%, variant de 0 à 13.68% selon la localisation. A noter que sur les 11 localisations, 9 d'entre elles présentent un taux d'échec inférieur à 2%. Seules deux localisations situées en Asie (Chine et Émirats Arabes Unis) présentent des taux d'échec plus élevés, respectivement de 3.85% et 13.68%.

Avec les adresses IPref, le taux d'échec moyen est de l'ordre de 22%, oscillant entre 20 et 31%. Pour 10 des 11 localisations testées, les résultats sont de même calibre (taux d'échec entre 20 et 24%). Une localisation sort du lot, avec les adresses IPref issues de OpenDNS et GoogleDNS, le Vénézuela. Sur

Tableau 5.14 – Taux d'échec de récupération des pages webs légitimes en utilisant IPdef ou IPref, sur 11 localisations géographiques

URL utilisant l'adresse IP fournie par le DNS	**Taux d'échec** de récupération des pages webs (min ≤ moyenne ≤ max)	Écart-Type[1]
Défaut	0% ≤ **2.19%** ≤ 13.68%	3.99%
OpenDNS	20.37% ≤ **22.73%** ≤ 28.70%	2.09%
GoogleDNS	20.37% ≤ **22.56%** ≤ 31.48%	3.11%
DNSAdvantage	21.30% ≤ **22.39%** ≤ 24.07%	0.81%

[1] entre localisations

ce site géographique, nous avons deux pics d'échec à 28 et 31%.

Nous notons donc ici une nette différence de résultats sur les taux d'échec obtenus, selon l'adresse IP utilisée : IPdef ou IPref (cf. explications sur les causes probables en section 5.3.5.3).

5.3.5.3 Problèmes rencontrés

5.3.5.3.1 Causes probables des échecs de récupération de la page PageRef : Nous avons constaté un taux d'échec de récupération des pages 10 fois plus élevés pour PageRef (vs. PageDef). La principale différence, entre les 2 techniques de récupération de page web, réside dans la zone "domaine" de l'URL demandée. Dans un cas il s'agit d'un FQDN, dans l'autre d'une adresse IP. Au travers de l'analyse des pages récupérées ou des erreurs rencontrées à l'exécution du programme, nous supposons que les causes probables de ces échecs sont : soit un manque de configuration (ou une configuration inappropriée) du reverse DNS, soit la virtualisation de plusieurs domaines derrière une seule et même adresse IP. En effet dans ce dernier cas, puisque nous n'apportons aucune précision sur le FQDN recherché dans l'URL demandée, nous ne pouvons obtenir de réponse.

Les causes probables d'échec rencontrées sont illustrées au travers de 4 exemples, pour lesquels nous n'avons pu récupérer la page PageRef :

- Une interrogation DNS sur le FQDN `www.lsn.com.au` retourne l'adresse IP 20.134.224.83. Néanmoins, une interrogation Reverse DNS sur l'adresse IP 20.134.224.83 retourne un message d'erreur (*"unable to resolve, Country IP Address : UNITED STATES"*). Nous sommes confrontés ici à une absence de configuration du reverse DNS.
- Une requête DNS sur le FQDN `www.allianz.fr` retourne l'adresse IP 194.98.39.11. Une interrogation Reverse DNS sur l'adresse IP 194.98.39.11 retourne le FQDN `www.agf.fr`. A savoir que AGF et Allianz ont fusionné. Via un navigateur web, nous constatons d'ailleurs qu'une redirection automatique est effectuée depuis `www.agf.fr` vers `www.allianz.fr`. Nous sommes donc ici en présence d'un double hébergement de FQDN derrière une même adresse IP.
- Une interrogation DNS sur le FQDN `moncompte.numericable.fr` retourne l'adresse IP 85.68.0.39. Une requête DNS inverse sur l'adresse IP 85.68.0.39 retourne le FQDN `ip-39.net-85-68-0.static.numericable.fr`. Dans ce cas, la configuration Reverse DNS n'est pas appropriée pour la récupération de la page web. En effet, elle est représentative d'un découpage en sous-domaines.
- Une requête DNS sur le FQDN `www.cisco.com` retourne les informations suivantes : 88.221.8.170 pour l'adresse IP et 4 alias : `www.cisco.com.akadns.net`, `geoprod.cisco.com.akadns.net`, `www.cisco.com.edgekey.net` et `www.cisco.com.edgekey.net.globalredir.akadns.net`. Une requête DNS inverse sur l'adresse IP 88.221.8.170 retourne le FQDN `a88-221-8-170.deploy.akamaitechnologies.com`. Nous sommes ici en présence d'une virtualisation de plusieurs domaines derrière une même adresse IP, effectuée par une société tierce : Akamai Technologies.

A contrario, un exemple pour lequel notre récupération de PageRef fonctionne est le FQDN `ib.swedbank.lv` dont l'adresse IP retournée est 193.203.196.143. Derrière cette adresse IP, il semblerait qu'il n'y ait pas de découpage, ni de virtualisation du domaine. En effet, une résolution DNS inverse sur l'adresse IP 193.203. 196.143 retourne le FQDN `ib.swedbank.lv`.

Un des objectifs essentiels de notre seconde approche, développée en section 6.1, sera donc de pallier ce taux d'échec anormalement élevé pour la récupération de la page PageRef.

5.3.5.3.2 Elimination des comparaisons de pages faussées : L'approche par mots, telle que nous l'avons conçue dans le calcul de score, nous retourne directement un taux de similitude (et non un score par page). Il a donc fallu être particulièrement attentif aux analyses de ce résultat. En effet, par exemple, il nous a fallu éliminer tous les cas où nous obtenions un score de 100% alors que les deux pages comparées étaient vides ou inappropriées (c.-à-d. deux pages d'erreur).

Autre cas qui concerne les deux approches (par caractères et par mots) : certaines pages récupérées ne sont pas vides mais erronées. En effet, il arrive qu'une des deux pages récupérées (voire les deux) soit une page d'erreur (p.ex. indisponibilité temporaire du site, déplacement de page, etc.). Le résultat de la comparaison étant faussé, le score obtenu doit également être éliminé des statistiques.

Dans cette première proposition, l'essentiel des éliminations de résultats erronés a été faite manuellement. Dans la seconde proposition portant sur davantage d'URLs, ce traitement a été automatisé.

5.3.5.3.3 Identification de l'adresse IPdef : De par l'implémentation réalisée, à partir de fonctions pré-existantes Java, il s'est avéré impossible d'identifier l'adresse IPdef utilisée pour la récupération de la page par défaut. En effet, rien n'indique que l'adresse IPdef utilisée est la première adresse IP retournée par DNSdef. Les fonctions utilisées pour la requête HTTP (p.ex. *url.openConnection()*) ne nous permettent pas d'identifier cette adresse. Ce verrou a été levé dans la seconde proposition.

5.3.5.3.4 Phase d'établissement de la connexion sécurisée : Une des difficultés rencontrées pour la récupération des pages webs a été la phase d'établissement SSL/TLS, préambule nécessaire à la récupération des pages webs de login. Divers essais ont été nécessaires avant d'aboutir à la solution exposée en section 5.3.5.1. Cette solution semble satisfaisante pour une très large majorité des URLs. Néanmoins, nous avons constaté quelques erreurs de certificats résiduelles et aléatoires. A ce jour, nous n'avons pas eu le temps d'investiguer plus avant ces erreurs ponctuelles.

5.3.5.4 Synthèse sur l'analyse du code source des pages webs

Les résultats de comparaison de pages webs obtenus dans cette première proposition sont assez convergents avec les résultats de l'étude préalable. Ils semblent toutefois indiquer que l'approche par mots doit être privilégiée, principalement de par les résultats obtenus sur les pages légitimes-contrefaites. Néanmoins, les résultats de la seconde proposition - portant sur davantage d'URLs - doivent confirmer ces résultats.

Un des freins essentiels exposés dans cette première proposition concerne le taux d'échec - trop élevé - relevé pour la récupération des pages de référence. Les développements menés dans la seconde approche viseront donc à tenter de remédier à ce problème, ainsi qu'à corroborer nos hypothèses sur les causes probables d'échec.

A l'issue de cette première proposition, deux autres verrous demeurent :

- la difficulté d'identification et d'élimination des comparaisons de pages erronées (c.-à-d. les pages d'erreur).
- l'identification de l'adresse IPdef utilisée pour le chargement de la page par défaut.

Enfin, même si l'approche par mots semble la plus appropriée pour déterminer la légitimité d'un site, les seuils minimum relevés sur les taux de similitude ne permettent pas - à ce stade - de dégager un seuil de décision. En effet, concernant les comparaisons de pages légitimes (cf. tableau 5.12), le seuil minimal moyen relevé est de 81.61% (tous DNSref et localisations confondus). Concernant les pages légitimes-contrefaites, le seuil maximal relevé est de 92%. La fenêtre de recouvrement est donc d'environ 11%. Certes, d'après les résultats des pages légitimes-contrefaites, ce chiffre peut être tempéré par deux éléments (cf. tableau 5.13) :

- la quantité de taux de similitude dépassant la barre des 80% n'est que de 2.70%.
- la fourchette haute de l'intervalle de confiance (à 95%) se situe à 65.59%.

Néanmoins, même si de futurs résultats confirment ces tendances, il semble peu probable que l'approche par mots telle qu'envisagée ici (c.-à-d. appliquée à l'ensemble du code source de la page) puisse être un critère suffisant dans la détermination de la légitimité de la page web visitée.

Notre seconde proposition visera donc également à explorer d'autres pistes pouvant aider à la décision de légitimité (p.ex. application de la méthode de score à des sous-parties de la page web, analyse focalisée sur les balises, etc.).

5.4 Synthèse du chapitre

Ce chapitre a présenté une première proposition visant à détecter les attaques de pharming réalisées côté client, focalisée sur les pages de login. Cette proposition est constituée de deux étapes : une vérification de l'adresse IP du domaine visité et une analyse/comparaison de la page web visitée, vs. des éléments de référence obtenus en utilisant un serveur DNS alternatif.

Les tests effectués sur la première solution proposée ont démontré que la vérification de l'adresse IP est une étape valable - mais pas toujours suffisante - dans la décision de légitimité d'un site de login. Ils ont également indiqué que les résultats issus de différents serveurs DNS de référence sont globalement convergents, ce qui laisse présager d'une certaine liberté dans le choix du serveur alternatif.

Ces résultats ont par ailleurs démontré que les méthodes actuellement utilisées pour la comparaison des pages webs - bien qu'intéressantes - se doivent d'être améliorées pour aboutir à un seuil de décision. En effet, à ce stade, aucune des deux approches ne se distingue avec certitude.

Par ailleurs, à l'issue de cette première proposition des doutes subsistent. Tout d'abord, la base d'URLs testées se doit d'être augmentée pour conforter les résultats obtenus et ce, pour les deux étapes de notre processus de décision.

Ensuite, certains éléments sont insuffisamment précis et/ou nécessitent une meilleure optimisation de traitement. On peut par exemple citer le problème d'identification de l'adresse IPdef ou le problème de l'élimination des pages d'erreur récupérées.

Enfin, le verrou majeur subsistant concerne les taux d'échec de récupération des pages webs de référence trop importants.

L'ensemble de ces éléments nous amène donc à envisager un nouveau scénario, qui fait l'objet de la seconde proposition développée dans le chapitre suivant.

6 Détection du pharming côté client : vers une redirection du GET HTTP

Sommaire

Notre proposition de détection du pharming côté client se base sur une étude du code source HTML de la page web, combinée à des requêtes DNS.

Dans le Chapitre 5, nous avons présenté une première solution de détection du pharming au sein de laquelle la page web visitée est comparée à une page web dite de référence, grâce à la substitution du nom de domaine par une adresse IP au sein de l'URL légitime. Bien que les résultats obtenus au travers de cette première proposition se soient avérés encourageants, ils ont également mis en exergue un certain nombre de faiblesses de l'approche proposée.

Dans ce chapitre, nous développons donc une seconde approche qui améliore la proposition précédente. Nous y expliquons notamment les modifications majeures apportées pour la récupération de la page de référence, désormais basée sur une redirection de la requête HTTP vers une adresse IP déterminée (c.-à-d. l'adresse IP de référence). Nous y détaillons également la combinaison de multiples techniques d'analyse du code source de la page web, afin de déterminer un seuil de décision. Les tests réalisés dans cette seconde proposition portent sur 328 URLs de login légitimes, évaluées depuis 11 localisations géographiques réparties sur 5 continents, ainsi que sur 75 nouveaux couples de pages légitimes-contrefaites.

Ensuite, nous explicitons les limitations associées aux deux approches proposées ainsi que les verrous techniques demeurant à l'issue de notre étude (cf. section 6.2).

Ce chapitre fait partie de nos contributions : cette seconde proposition a été publiée et présentée à la conférence *Network and System Security* (NSS) en Septembre 2011 [GL11].

6.1 Seconde proposition : vers une redirection du GET HTTP

Notre seconde proposition a été élaborée avec le quadruple objectif : de pallier les défauts et problèmes majeurs rencontrés dans la première approche, d'augmenter la base de résultats, d'améliorer les techniques de comparaison et enfin, de définir un seuil de décision.

Les verrous demeurant à l'issue de notre première proposition sont :

- un taux d'échec trop élevé pour la récupération de la page de référence PageRef,
- la difficulté d'identification et d'élimination des pages d'erreur,
- et enfin, l'absence d'identification de l'adresse IPdef, utilisée pour la récupération de la page par défaut PageDef.

6.1.1 Fonctionnement général

En premier lieu, nous nous sommes attachés à répondre au problème majeur du taux d'échec de récupération de PageRef.

Si nos hypothèses sur les causes probables de ce taux d'erreur (cf. section 5.3.5.3) s'avèrent exactes, nous devons trouver un moyen de transmettre l'information du FQDN demandé, lors de la requête de l'URL de référence.

La piste idéale se révèle être la réécriture du paquet afin de conserver l'URL d'origine intacte (et donc transmettre le FQDN), tout en imposant l'adresse du serveur destination.

En conséquence, le fonctionnement de notre seconde proposition se déroule de la manière suivante (cf. figure 6.1) :

Pour chaque page de login, le FQDN de l'URL visitée est extrait afin de générer deux requêtes DNS : l'une est envoyée au serveur DNS par défaut (DNSdef), l'autre est adressée au serveur DNS de référence (DNSref). DNSdef retourne plusieurs adresses IP, dont celle utilisée pour l'affichage de la page web dans le navigateur (IPdef), tandis que DNSref retourne une ou plusieurs adresses IP (IPref), incluant ou excluant IPdef.

Dans le cas où l'adresse IPdef est incluse dans la réponse IPref, le site est considéré comme légitime. Dans le cas contraire, nous procédons à l'analyse de la page web :

- la page web visitée (PageDef), telle qu'affichée dans le navigateur, est récupérée à partir de IPdef.

– La page de référence (PageRef) est récupérée grâce à l'envoi d'une requête GET HTTP, utilisant l'URL originale, à destination de IPref.
Considérons l'exemple suivant : l'URL visitée par l'Internaute est `https://www.amazon.com/gp/yourstore?ie=UTF8&ref_=pd_irl_gw&signIn=1`. La page affichée a été récupérée depuis le serveur web possédant l'adresse IP : 72.21.210.250 (IPdef), adresse connue grâce à la technique de récupération de PageDef utilisée dans cette approche (cf. section 6.1.4). Le serveur DNS de référence interrogé retourne l'adresse IP : 207.171.166.252 (IPref). La page de référence est alors récupérée en envoyant un GET HTTP, contenant l'URL `https://www.amazon.com/gp/yourstore?ie=UTF8&ref_=pd_irl_gw&signIn=1`, à l'adresse IP destination IPref : 207.171.166.252.

Les codes sources HTML des deux pages webs (PageDef et PageRef) sont ensuite comparés (cf. figure 6.1), grâce aux techniques énoncées en sections 5.2.2 et 6.1.5. Enfin, la légitimité de la page web visitée est déterminée en comparant le pourcentage de similitude obtenu entre les deux pages (PageDef et PageRef) à un seuil de décision pré-défini.

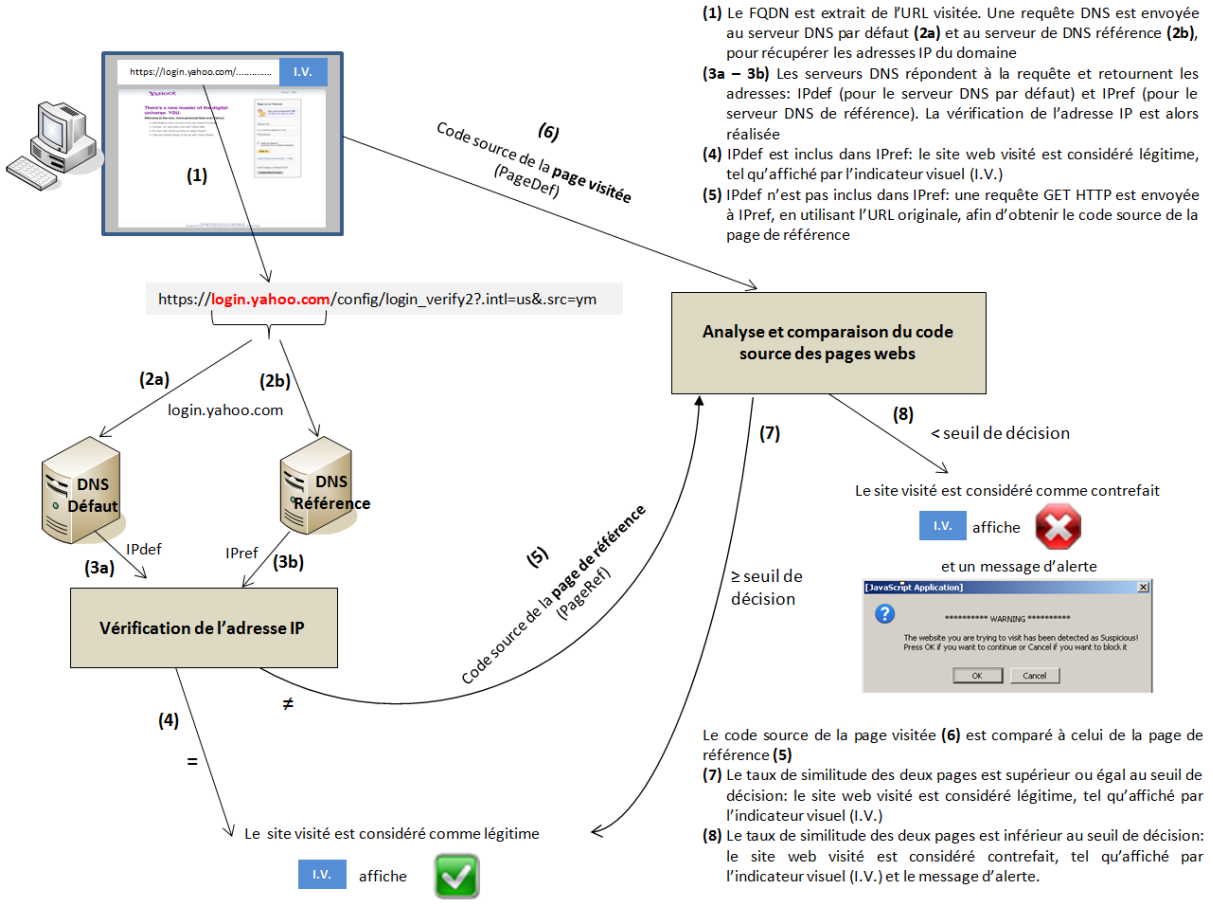


FIGURE 6.1 – Seconde approche : fonctionnement général

Définition du serveur de référence : Au vu des résultats obtenus précédemment, nous avons élaboré un nouveau scénario de définition du serveur de référence. L'utilisateur aurait alors le choix entre deux propositions : 1/ le choix du DNSref est laissé à son appréciation parmi une liste de serveurs pré-définis (= choix exposé dans la première proposition), ou 2/ le programme sélectionne et utilise - de manière automatique et aléatoire - un serveur DNSref, parmi une liste de serveurs pré-définis. Cette sélection, qui serait renouvelée à chaque vérification d'une page de login, est proposée grâce aux résultats très similaires obtenus auprès des différents serveurs DNS de référence. L'avantage de ce deuxième scénario - qui serait celui recommandé - serait de proposer une meilleure résistance aux attaques (cf. section 4.1.2.1), grâce au choix aléatoire effectué pour DNSref, renouvelé à chaque vérification.

Tableau 6.1 – Répartition des 328 URLs de login légitimes par secteur d'activités

Catégories	Quantité	Pourcentage
Banques	204	62%
Autres (administration, assurances, logiciels, jeux, FAI, industrie, vidéos, photos, informations)	71	22%
e-commerce	44	13%
Réseaux sociaux	5	2%
email	4	1%

6.1.2 Conditions d'expérimentation

Sur le même principe que celui utilisé lors de notre première proposition (cf. section 5.3.2), nous avons évalué l'efficacité de notre proposition à partir de deux types de comparaisons :

- des comparaisons de couples de pages de login légitimes récupérées depuis plusieurs localisations géographiques et plusieurs serveurs DNS (c.-à-d. le serveur DNS par défaut de l'utilisateur ainsi que nos trois serveurs DNS de référence : OpenDNS, GoogleDNS et DNSAdvantage). L'ensemble des couples de pages légitimes ont été collectées entre Mars et Juin 2011.
- des comparaisons de couples de pages de login légitimes-contrefaites, visuellement très similaires. L'ensemble des pages contrefaites (et légitimes associées) utilisées ici ont été récupérées entre Décembre 2010 et Juin 2011.

Les tests effectués se sont ensuite déroulés en deux temps :

- Une première analyse qui a deux objectifs : 1/ Mesurer l'efficacité de cette seconde proposition vis-à-vis de la précédente, tant au niveau des résultats IP que de l'analyse sur le code source HTML dans son intégralité, et 2/ Explorer de nouvelles techniques de comparaison pour n'en retenir que les plus pertinentes. Pour la suite du chapitre, cette première analyse sera nommée *Analyses pour la définition des techniques de comparaison les plus pertinentes*.
- Une deuxième analyse visant à définir le seuil de décision ainsi que la méthode de comparaison finale des pages webs (à partir des techniques de comparaison retenues en première analyse). Pour la suite du chapitre, cette deuxième analyse sera nommée *Analyses pour l'élaboration de la méthode de comparaison finale*.

6.1.2.1 Couples de pages légitimes

Nous avons établi un panel de 328 URLs de login décomposé de la manière suivante :

- **224 nouvelles URLs (dont 2 à FQDN commun)**. Ces 224 URLs ont été sélectionnées en utilisant les mêmes critères que précédemment, à savoir : diversifier les secteurs d'activités, multiplier les TLDs ainsi que les langages des pages webs.
- **et les 104 URLs à FQDN uniques utilisées dans la première proposition**. L'intégration de ces 104 URLs a pour objectif d'avoir un premier indicateur de la variabilité temporelle des résultats obtenus précédemment (cf. sections 5.3.4.2 et 5.3.5.2).

Classification des 328 URLs de login : Nous avons classifié les 328 URLs sélectionnées selon les 5 secteurs d'activités identifiés dans la première proposition, à savoir : banques, réseaux sociaux, e-commerce, email et autres (cf. section 5.3.2.1 pour plus de détails sur les classifications). La répartition des catégories, par ordre d'importance, reste identique à l'étude précédente, avec une nette prédominance des sites bancaires (cf. tableau 6.1).

Les 328 URLs sélectionnées sont issues de 48 TLDs différents, divisables en 2 catégories : les cc-TLD et les g-TLD. Pour une meilleure lisibilité, les TLDs sélectionnés ont été regroupés par continent (cf. tableau 6.2). A nouveau, la répartition des TLDs reste à peu près similaire à l'étude précédente, même si leur nombre a plus que doublé. On constate également l'apparition d'un nouveau g-TLD : *COOP* pour *Cooperative*.

Tableau 6.2 – Répartition des 48 TLDs des 328 URLs de login légitimes

	Pourcentage	TLD
Europe	38.1%	AD, AT, BA, BE, BG, CY, CZ, DE, DK, EE, FI, FR, GR, IT, LU, LV, NL, NO, PL, RO, SE, UK
Asie	5.8%	GE, ID, IL, IN, JO, JP, MV, MY, PH, PK, SG, TR
Océanie	4.3%	AU, NZ
Amérique du Sud	3.4%	AR, BR, CL, CO, MX
Afrique	0.9%	ZA
Amérique du Nord	0.6%	CA, HN
Commercial	44.8%	COM
Network	1.2%	NET
Organization	0.6%	ORG
Cooperative	0.3%	COOP

Analyses pour la définition des techniques de comparaison les plus pertinentes : Pour comparer nos deux propositions et définir les techniques de comparaison les plus pertinentes, nous avons testé nos 328 URLs de login sur 11 sites géographiques, répartis sur 5 continents. Nous avons ainsi collecté les résultats issus de 4 serveurs DNS (c.-à-d. le serveur DNS par défaut de l'utilisateur - vérifié différent des serveurs DNS de référence -, et nos 3 serveurs de référence).

La distribution géographique des tests effectués est la suivante (cf. figure 6.2) :

- Europe : France (3 tests en Île de France et 1 test dans le Sud de la France), Belgique
- Amérique du Nord : États-Unis, Mexique
- Amérique du Sud : Vénézuela
- Asie : Chine, Turquie
- Afrique : Sénégal

A noter que pour certaines comparaisons effectuées (cf. section 6.1.5), nous avons parfois restreint nos analyses à 6 localisations, principalement par manque de temps.

Dans un second temps, depuis une même localisation (située en Île de France), nous avons testé les 328 URLs à 10 reprises, sur une période de 3 mois (entre Avril et Juin 2011). Ceci afin d'avoir une seconde mesure de la variabilité temporelle des résultats.

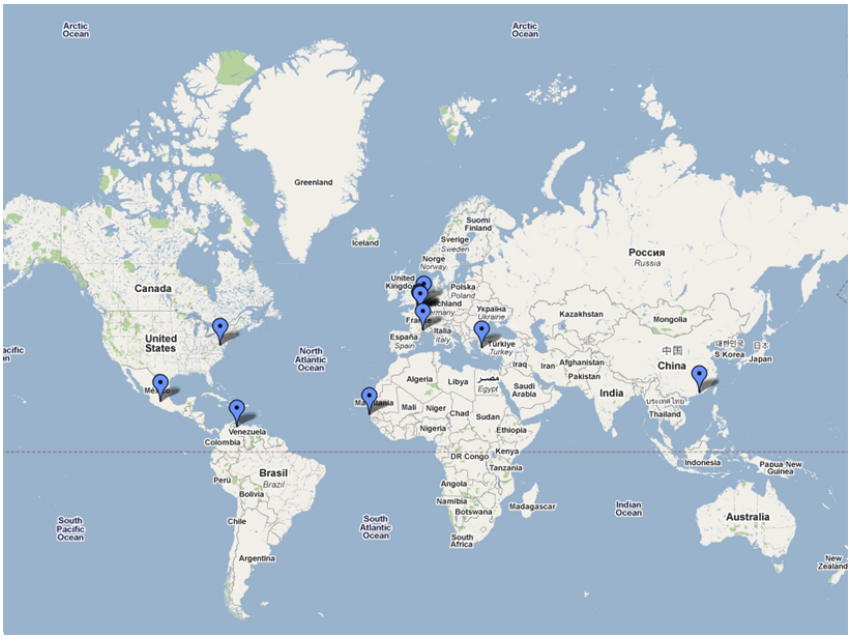


Figure 6.2 – Seconde approche : répartition géographique des tests

A noter que, comme dans la première proposition, nous avons rencontré des pertes de connectivité importantes (rupture de courant électrique et ou Internet) sur les localisations du Vénézuela et du Sénégal. Ceci peut expliquer des taux d'erreur sensiblement plus élevés sur ces localisations.

Analyses pour l'élaboration de la méthode de comparaison finale : Enfin, pour élaborer une méthode de comparaison basée sur les techniques les plus pertinentes et déterminer un seuil de décision pour la comparaison des pages webs, nous avons choisi d'étalonner notre solution sur les résultats issus d'une seule localisation et d'un seul couple DNSdef-DNSref : Bruxelles(Belgique) / DNSdef-GoogleDNS. Puis, une fois défini le seuil de décision, nous avons vérifié l'efficacité de notre méthode de détection sur 5 autres localisations : Mexico(Mexique), Montpellier(France-Sud), Samoreau(France-IdF), Dakar(Sénégal) et Shenzhen(Chine). Pour chaque localisation, nous n'avons utilisé qu'un seul couple DNSdef-DNSref, respectivement DNSdef- : GoogleDNS, OpenDNS, DNSAdvantage, OpenDNS et GoogleDNS.

6.1.2.2 Couples de pages légitimes-contrefaites

Les couples de pages légitimes contrefaites, visuellement très similaires, ont été récupérées selon la méthode exposée en section 5.3.2.2.

L'ensemble des pages collectées nous ramène aux 5 secteurs d'activité exposés précédemment.

Analyses pour la définition des techniques de comparaison les plus pertinentes : Nous avons étudié la variabilité des résultats, par rapport à la première proposition, sur 75 nouveaux couples de pages légitimes-contrefaites.

Analyses pour l'élaboration de la méthode de comparaison finale : Puis, pour la définition de notre seuil de décision, nous avons étalonné notre méthode de détection sur 55 couples de pages légitimes-contrefaites (extraits des 75 couples précédents). Nous avons ensuite vérifié notre méthode de détection sur un deuxième jeu de 58 couples de pages légitimes-contrefaites, répartis de la façon suivante : 20 couples de pages - non utilisés dans l'étalonnage - sont issues des 75 couples précédents, auxquels ont été rajoutés 38 nouveaux couples de pages.

6.1.3 Implémentation : aperçu général

La nouveauté - et la difficulté - majeure de cette seconde proposition réside dans la récupération de la page de référence, effectuée via une redirection du GET HTTP vers une adresse IP définie.

A cet effet, nous avons exploré deux pistes/scénarios d'implémentation visant à intervenir à un niveau inférieur du modèle OSI (en comparaison des fonctions de récupération de pages utilisées dans la première approche) :

- Une première piste a consisté en l'exploration de l'utilisation d'une technique d'écoute des paquets. L'objectif visé étant d'écouter la requête de la PageDef, afin de générer la requête de la PageRef.
- Une seconde piste a consisté en la réécriture complète de la requête GET HTTP, au travers de l'écriture de sockets.

Ces deux pistes sont détaillées en section 6.1.4.

L'implémentation de la première piste a automatiquement induit des modifications majeures dans l'ordonnancement initial de nos programmes. En effet, la nécessité d'un modèle de paquet et la génération de la nouvelle requête GET HTTP qui devaient être regroupés au sein d'un même programme, ont fortement impacté notre technique - initialement très séparée - de récupération des informations Défaut (adresse IP et page web) et Référence (cf. section 6.1.4.1).

Au travers de la seconde piste, l'implémentation est revenue à son organisation initiale, présentant ainsi le même type de découpage que dans la première proposition.

Au passage, on peut noter que l'intérêt de connaître l'adresse IPdef s'est révélé ici devenir une véritable nécessité, et ce quelle que soit la piste retenue.

Concernant l'analyse du contenu du code source de la page web, nous avons également exploré de nouvelles pistes visant à une meilleure différenciation des pages légitimes et contrefaites. L'objectif est triple : la définition d'un seuil de décision, l'éventuelle optimisation/diminution du temps de traitement, et l'éventuelle détection des zones de codes majoritairement impactées par la contrefaçon (pour en déduire les techniques les plus pertinentes à utiliser).

En complément des techniques de détection étudiées dans notre première proposition (c.-à-d. approches par caractères et par mots, appliquées au code source complet), nous avons donc exploré deux nouvelles familles de pistes (détaillées en section 6.1.5) : 1/ application de la méthode de calcul sur des sous-parties du code source complet (c.-à-d. dans son intégralité), et 2/ analyse des balises contenues dans le code source.

A noter que pour une meilleure lisibilité, le *code source complet* sera désormais nommé *code complet* dans la suite de ce chapitre. A ne pas confondre avec la page web complète (qui inclut des images, des fichiers de scripts complémentaires, etc.) que nous n'avons pas explorée dans notre étude.

6.1.4 Vérification de l'adresse IP du domaine visité

L'objectif de cette section est de déterminer si la relative staticité des adresses IP associées aux FQDN des pages de login, se confirme sur davantage d'URLs (vs. les résultats obtenus lors de la première proposition).

En complément, cette section détaille les modifications majeures apportées à notre implémentation afin de répondre aux deux problèmes intrinsèquement liés que sont : l'identification de l'adresse IPdef utilisée pour récupérer PageDef, et la mise en œuvre de l'implémentation de la redirection du GET HTTP.

Les résultats de tests obtenus ici se concentrent sur l'étude des URLs à FQDN unique, c.-à-d. 327 des 328 URLs sélectionnées.

6.1.4.1 Implémentation : points spécifiques

Premier scénario d'écoute des paquets : Une première piste étudiée pour la réécriture du GET HTTP vers une adresse IP choisie, a été l'utilisation de l'API *pcap* (pour *packet capture*), couramment utilisée par les outils de supervision du trafic réseau. La piste évoquée consiste alors à utiliser un modèle de paquet (c.-à-d. le premier GET HTTP envoyé pour la récupération de PageDef) pour générer le GET HTTP de PageRef.

Le fonctionnement envisagé pour notre programme devient alors le suivant : 1/ Un premier programme contenant la requête DNS vers DNSdef (cf. algorithme 4), et 2/ Un second programme contenant : la requête DNSref, la récupération des pages webs et le calcul des scores en utilisant les approches par caractères et par mots (cf. algorithme 5).

Algorithme 4 Seconde approche avec scénario *pcap* : DNS par défaut

Entrées: le fichier *.txt* contenant la liste des n URLs de login.
Sorties: 1 fichier *DNSqueries_default.txt* contenant les n réponses du serveur par défaut.

1: **pour** $i = 0$ **à** n **faire**
2: Requête DNS auprès du serveur par défaut pour le FQDN(i).
3: Sauvegarde des adresses IP (*IPdef*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
4: **fin pour**

Pour notre implémentation, nous avons testé deux librairies Java *jNetCap* [Tec] et *Jpcap* [UoC]. Les difficultés majeures associées à cette approche et implémentation résident dans les étapes 2, 3 et 9 de l'algorithme 5 :

- Dans l'étape N° 2, si DNSdef a retourné plusieurs adresses IP, nous n'avons pas connaissance de l'adresse qui est utilisée pour la récupération de PageDef en étape N° 4.
- L'écoute et l'interception du paquet à l'étape N° 3 doit se faire grâce à l'utilisation d'un filtre ciblé sur l'adresse IP destination IPdef. En effet, un filtrage par port applicatif ne peut être suffisant (à moins d'interdire toute autre navigation Internet simultanée, ce qui est inenvisageable en dehors d'un environnement expérimental). Or, l'utilisation de la fonction habituelle *url.openConnection()*

pour la récupération de PageRef ne nous donne pas connaissance de cette adresse IP. Il faudrait alors envisager un filtrage sur l'ensemble des adresses IP retournées par DNSdef pour le FQDN concerné, espace dont la taille est variable et changeante à chaque requête DNS.

– Enfin l'étape 9 - facilement réalisable pour une page HTTP classique - s'est avérée bloquante pour une page de login HTTPS, puisque nos différents essais à ce propos se sont révélés insatisfaisants. En effet, en préambule du GET HTTP, l'ensemble des échanges liés à l'établissement de la connexion sécurisée doivent être réalisés. Ceci revient à effectuer une implémentation complète de SSL, ce qui représente un codage relativement lourd et complexe.

Algorithme 5 Seconde approche avec scénario *pcap* : DNS de référence

Entrées: le fichier *.txt* contenant la liste des n URLs de login, et le fichier *DNSqueries_default.txt* créé par le programme précédent (cf. algorithme 4).

Sorties: 1 fichier contenant les n réponses du serveur de référence, 3 fichiers contenant les scores des n pages (2 pour l'approche par caractères et 1 pour l'approche par mots), et n fichiers contenant les codes sources des pages webs de référence.

1: **pour** $i = 0$ **à** n **faire**
2: Récupération de l'adresse IPdef dans le fichier *DNSqueries_default.txt*.
3: Écoute du paquet à suivre.
4: Récupération et sauvegarde du code source de la page web par défaut en utilisant l'URL `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html`. (1 fichier par URL, nommé *date_heure_def_FQDN*(i)*.txt*)
5: Calcul du score de PageDef(i) avec l'approche par caractères, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
6: Requête DNS auprès du serveur de référence pour le FQDN(i).
7: Sauvegarde des adresses IP (*IPref*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
8: Récupération de la première adresse IP *IPref*, différente de *IPdef*.
9: Création du nouveau paquet pour la récupération et la sauvegarde du code source de la page web de référence, en utilisant l'URL `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html`. Le nouveau paquet créé est envoyé à destination de *IPref* (1 fichier par URL, nommé *date_heure_nomDNSref_FQDN*(i)*.txt*).
10: Calcul du score de PageRef(i) avec l'approche par caractères, et sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
11: Calcul du taux de similitude PageDef(i)/PageRef(i) avec l'approche par mots, et sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
12: **fin pour**

Second scénario d'écriture de socket : Une seconde piste étudiée a été la réécriture de la socket qui contient le GET HTTP. En effet, la fonction *url.openConnection()*, classiquement utilisée pour la récupération d'une page web, est une version simplifiée d'une socket présentant les arguments *host* (contenant le FQDN de l'URL demandée) et *port* (contenant le port destination, c.-à-d. 443 pour le HTTPS). Une fois la socket ouverte, les paramètres du GET HTTP peuvent alors être spécifiés.

Notre implémentation repose sur 4 librairies Java : *java.net.Socket* pour l'écriture des sockets, ainsi que *java.security.**, *javax.net.ssl.** et *javax.net.SocketFactory* pour l'établissement de la connexion HTTPS (c.-à-d. l'utilisation d'un modèle de socket SSL pré-établi).

Pour générer la requête de PageDef nous utilisons la version basique de la fonction *socket*, tandis qu'une deuxième version plus élaborée (permettant de spécifier l'adresse IP du serveur destination) est utilisée pour récupérer PageRef.

Pour illustrer nos propos, considérons deux exemples d'implémentation de la récupération de PageDef et PageRef, pour l'URL `https://login.yahoo.com/config/mail?&.src=ym&.intl=fr`, basés sur l'utilisation de socket :

– L'algorithme 6 illustre un exemple de récupération de PageDef : les étapes 1 à 3 permettent la définition de l'URL et du port destination associé, utilisés pour la requête. Puis, les étapes 4 et 5 créent et ouvrent la socket HTTPS pour le FQDN requis. Enfin, les étapes 6 à 10 spécifient le contenu du GET HTTP, à savoir : le chemin d'accès du fichier demandé.

Algorithme 6 Exemple de requête HTTP à base de socket, pour la récupération de PageDef

```
1: String FQDN = "login.yahoo.com";
2: String file = "/config/mail?&.src=ym&.intl=fr";
3: int portdest = 443;
4: SocketFactory socketFactory = SSLSocketFactory.getDefault();
5: s = socketFactory.createSocket(FQDN, portdest);
6: OutputStream out = s.getOutputStream();
7: PrintWriter outw = new PrintWriter(out, false);
8: outw.print("GET " + file + " HTTP/1.0\r\n");
9: outw.print("Accept : text/plain, text/html,text/*\r\n");
10: outw.print("\r\n");
```

– L'algorithme 7 illustre un exemple de récupération de PageRef : les étapes 1 à 4 permettent la définition de l'URL, de l'adresse IP et du port destination, utilisés pour la requête. Puis, les étapes 5 et 6 créent et ouvrent la socket HTTPS avec le serveur possédant l'adresse IP destination spécifiée précédemment. Enfin, les étapes 7 à 12 spécifient le contenu du GET HTTP, à savoir : le chemin d'accès du fichier ainsi que le FQDN demandé.

Algorithme 7 Exemple de requête HTTP à base de socket, pour la récupération de PageRef

```
1: String FQDN = "login.yahoo.com";
2: String IPref = "217.146.187.123";
3: String file = "/config/mail?&.src=ym&.intl=fr";
4: int portdest = 443;
5: SocketFactory socketFactory = SSLSocketFactory.getDefault();
6: s = socketFactory.createSocket(IPref, portdest);
7: OutputStream out = s.getOutputStream();
8: PrintWriter outw = new PrintWriter(out, false);
9: outw.print("GET " + file + " HTTP/1.0\r\n");
10: outw.print("Accept : text/plain, text/html, text/*\r\n");
11: outw.print("Host : " + FQDN + "\r\n");
12: outw.print("\r\n");
```

Choix du scénario : De ces deux scénarios, la solution retenue est donc celle présentant le scénario le plus favorable, à savoir : l'écriture de socket.

Notre implémentation, dont l'ordonnancement est très similaire à celle de la première proposition (cf. section 5.3.3), est illustrée par les algorithmes 8 et 9.

A noter que dans cette nouvelle implémentation, nous avons également supprimé la notion d'"heure" dans les noms de fichiers contenant PageDef et PageRef. Ceci afin d'éviter tout problème de traitement ultérieur des pages (cf. section 5.3.3.2), traitement devenu indispensable pour tester les nouvelles techniques de comparaisons de pages étudiées en section 6.1.5.1.

Identification de l'adresse IPdef et choix de l'adresse IPref : Un autre avantage de la fonction *socket* est de pouvoir identifier avec certitude l'adresse IP utilisée pour la récupération de la page par défaut. En effet, reprenons l'exemple utilisé précédemment, à savoir la récupération de PageDef associée à l'URL `https://login.yahoo.com/config/mail?&.src=ym&.intl=fr`. Alors que notre serveur DNSdef a retourné les adresses IP 217.12.8.76 et 217.146.187.123 pour le FQDN `login.yahoo.com`, l'affichage de la socket utilisée pour récupérer PageDef retourne les éléments suivants : `Socket[addr=login.yahoo.com/217.12.8.76,port=443,localport=55271]`. On y retrouve aisément les informations de : FQDN, adresse IP associée, port destination et port source. L'adresse IPdef est alors extraite grâce à une analyse du champ *addr* de la socket affichée.

Algorithme 8 Seconde approche avec scénario *socket* : DNS par défaut

Entrées: le fichier *.txt* contenant la liste des n URLs de login.

Sorties: 1 fichier contenant les n réponses du serveur par défaut, 1 fichier contenant les scores (approche par caractères) des n pages par défaut, 1 fichier contenant les n adresses IPdef utilisées pour récupérer les n pages par défaut, et n fichiers contenant les codes sources des pages webs par défaut.

1: **pour** $i = 0$ **à** n **faire**
2: Requête DNS auprès du serveur par défaut pour le FQDN(i).
3: Sauvegarde des adresses IP (*IPdef*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
4: Récupération et sauvegarde du code source de la page web par défaut `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html` en créant une socket HTTPS basique (1 fichier par URL, nommé *date_def_FQDN*(i)*.txt*).
5: Extraction[1] et sauvegarde de l'adresse IP (*IPdef*) utilisée pour récupérer Pagedef.
6: Calcul du score (approche par caractères) de la page(i) récupérée, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs).
7: **fin pour**

[1] cf. explications sur la méthode d'extraction dans le paragraphe "Identification de l'adresse IPdef" de la section en cours.

Algorithme 9 Seconde approche avec scénario *socket* : DNS de référence

Entrées: le fichier *.txt* contenant la liste des n URLs de login et le fichier contenant les n adresses IPdef utilisées pour récupérer les n pages par défaut.

Sorties: 1 fichier contenant les n réponses du serveur de référence, 1 fichier contenant les scores (approche par caractères) des n pages de référence, 1 fichier contenant les scores (approche par mots) des n pages défaut vs. les n pages de référence, et n fichiers contenant les codes sources des pages webs de référence.

1: **pour** $i = 0$ **à** n **faire**
2: Requête DNS auprès du serveur de référence pour le FQDN(i).
3: Sauvegarde des adresses IP (*IPref*) retournées pour le FQDN(i) (un même fichier pour les n URLs).
4: Lecture de l'adresse IPdef utilisée pour récupérer la page par défaut du FQDN(i) et comparaison avec les résultats retournés par DNSdef :
5: **si** DNSref a retourné une seule adresse IP **alors**
6: l'adresse IPref utilisée pour récupérer PageRef est l'adresse IP retournée par DNSref.
7: **sinon si** DNSref a retourné plusieurs adresses IP **alors**
8: l'adresse IPref utilisée pour récupérer PageRef est la première adresse IP trouvée différente de IPdef[1].
9: **finsi**
10: Récupération et sauvegarde du code source de la page web de référence `https://FQDN(i)/arborescence1/.../arborescenceX/fichier.html` en créant une socket HTTPS avancée, destinée à l'IPref sélectionnée précédemment (1 fichier par URL, nommé *date_nomDNSref_FQDN*(i)*.txt*).
11: Calcul du score (approche par caractères) de la page(i) récupérée, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs)
12: Calcul du taux de similitude des PageDef et PageRef de la page(i) avec l'approche par mots, puis sauvegarde du résultat dans un fichier *.txt* (1 fichier pour les n URLs)
13: **fin pour**

[1] cf. explications détaillées et exemple dans le paragraphe "Identification de l'adresse IPdef" de la section en cours.

A noter que pour améliorer notre seconde proposition (vs. la première proposition), nous avons choisi d'utiliser une adresse IPref obligatoirement différente de l'adresse IPdef dès lors que cela s'avérait possible. En effet, dans le cas où DNSref retourne plusieurs adresses IP, l'adresse IPref utilisée est désormais la première adresse IP trouvée différente de IPdef.

Par exemple, supposons que l'adresse IPdef utilisée pour récupérer la page web par défaut est

Tableau 6.3 – Comparaison des réponses du DNS par défaut vs. 3 serveurs DNS de référence, sur 11 localisations géographiques

	Taux de convergence avec les adresses IP retournées par le serveur DNS par défaut (min ≤ moyenne ≤ max)	Écart-Type [1]	Intervalle de confiance à 95% [1]
OpenDNS	82.62% ≤ **86.29%** ≤ 92.66%	2.44%	[84.85% ; 87.73%]
GoogleDNS	85.80% ≤ **87.72%** ≤ 93.60%	2.51%	[86.24% ; 89.20%]
DNSAdvantage	84.36% ≤ **86.51%** ≤ 87.50%	1.05%	[85.89% ; 87.13%]

[1] entre localisations

199.59.148.83. DNSref retourne les adresses IP : 199.59.148.11, 199.59.148.10 et 199.59.148.83. L'adresse IPref sélectionnée est alors 199.59.148.11.

6.1.4.2 Résultats

Cette section s'inscrit dans les analyses pour la définition des techniques de comparaison les plus pertinentes. Elle se décompose en deux temps : une étude multi-localisations et une étude temporelle depuis une même localisation (pour plus de détails sur les échantillons et zones de tests associées, cf. section 6.1.2).

6.1.4.2.1 Variabilité des adresses IP utilisées par les pages de login :

Etude multi-localisations : Le tableau 6.3 indique les taux de convergence des adresses IP (c.-à-d. nous retrouvons au minimum 1 adresse *IPdef* dans les adresses *IPref*) obtenus auprès des différents serveurs DNS de référence. Ceux-ci confortent et améliorent les conclusions de la première approche (cf. section 5.3.4.2). En effet, les résultats obtenus ici nous indiquent que les adresses IP utilisées par les sites de login sont fortement convergentes : les moyennes des taux de convergence, entre serveur DNS par défaut et serveurs de référence, varient de 86 à 87% (contre 81 à 82% dans la première proposition). De plus, l'écart-type (entre localisations) s'est considérablement réduit : 1.05 à 2.51%, contre 3.62 à 5.00% précédemment. Enfin, l'intervalle de confiance à 95% (c.-à-d. l'incertitude d'estimation, entre localisations) varie de 84 à 89%. Nous constatons que les 3 DNS de référence donnent toujours le même type de résultats, avec un léger avantage de convergence pour DNSAdvantage (vs. OpenDNS dans la première proposition).

Cette analyse qui inclut les 104 URLs à FQDN unique de la première proposition, précédemment testées 3 à 5 mois plus tôt sur 11 localisations, tend à indiquer une certaine stabilité temporelle des résultats.

De par les améliorations apportées dans notre seconde proposition, nous sommes désormais en mesure d'analyser les adresses IP utilisées pour récupérer PageDef et PageRef. Une deuxième analyse menée sur les adresses IP (cf. tableau 6.4) nous indique alors que 77 à 79% des FQDNs interrogés ont retourné une seule et même adresse IP (c.-à-d. lorsque la réponse du DNS défaut est comparée à la réponse du DNS de référence), avec des écart-types oscillant entre 0.97 et 2.55%. Par conséquent, ceci nous indique que dans 77 à 79% des cas, les deux pages webs (PageDef et PageRef) ont été récupérées auprès du même serveur web.

Etude multi-temporelle depuis une même localisation : Depuis une même localisation, les résultats obtenus sur une période de 3 mois (cf. tableau 6.5) nous indiquent que le taux de convergence des adresses IP associées aux pages de login sont très stables. En effet, les écart-types sont inférieurs à 1%, pour des taux de convergence moyens situés entre 86 et 87%. A noter que lors de l'analyse multi-localisations, le site géographique testé ici avait indiqué un taux de convergence moyen oscillant entre 85.80 et 86.75%, selon le serveur DNS de référence utilisé.

Les analyses sur le taux d'unicité des adresses IPdef et IPref utilisées pour récupérer les pages webs associées (cf. tableau 6.6), nous indiquent que 77 à 78% des FQDNs interrogés (avec un écart-type maximum de 1.02%) ont retourné une seule et même adresse IP. A noter que lors de l'analyse

Tableau 6.4 – Comparaison des adresses IPdef et IPref utilisées pour récupérer les 328 pages webs légitimes, sur 11 localisations géographiques

	Taux d'unicité des adresses IPdef et IPref (min ≤ moyenne ≤ max)	Écart-Type [1]	Intervalle de confiance à 95% [1]
OpenDNS	75.61% ≤ **78.08%** ≤ 84.40%	2.34%	[76.70% ; 79.46%]
GoogleDNS	76.52% ≤ **79.14%** ≤ 84.76%	2.55%	[77.63% ; 80.65%]
DNSAdvantage	75.84% ≤ **77.75%** ≤ 79.01%	0.97%	[77.18% ; 78.32%]

[1] entre localisations

Tableau 6.5 – Comparaison des réponses du DNS par défaut vs. 3 serveurs DNS de référence, pour une même localisation géographique, sur une période de 3 mois

	Taux de convergence avec les adresses IP retournées par le serveur DNS par défaut (min ≤ moyenne ≤ max)	Écart-Type [1]	Intervalle de confiance à 95% [1]
OpenDNS	85.08% ≤ **86.33%** ≤ 88.09%	0.93%	[85.75% ; 86.91%]
GoogleDNS	85.40% ≤ **86.69%** ≤ 87.77%	0.78%	[86.20% ; 87.18%]
DNSAdvantage	85.71% ≤ **86.97%** ≤ 88.79%	0.88%	[86.39% ; 87.52%]

[1] entre les différentes dates de tests

multi-localisations, cette localisation avait retourné un taux d'unicité moyen variant de 76.52 à 78.35%, selon le serveur DNS de référence utilisé.

6.1.4.2.2 Taux d'échec des requêtes DNS : Précisons tout d'abord que les requêtes DNS effectuées ne comportaient qu'un seul essai de résolution par FQDN.

Pour l'étude multi-localisations, nous obtenons des taux d'échec moyens des requêtes DNS très faibles, variant de 0.08% à 0.44% (cf. tableau 6.7). En effet, nous obtenons généralement entre 0 et 2 erreurs par site géographique (sur les 327 URLs testées). Deux exceptions sont à noter toutefois : 1/ L'une des localisations située en Île de France nous retourne jusqu'à 11 erreurs auprès du serveur DNS par défaut, mais aucune erreur auprès des serveurs de référence. 2/ La localisation située en Chine nous retourne jusqu'à 7 erreurs auprès de GoogleDNS.

Pour l'étude multi-temporelle, nous obtenons des taux d'échec moyens des requêtes DNS nuls auprès des serveurs DNS de référence, contre un taux moyen d'échec de 3.57% auprès du serveur DNS par défaut (cf. tableau 6.8). Ce taux d'échec notablement plus élevé auprès du serveur DNS par défaut (vs. les résultats de l'étude multi-localisations), s'explique par le fait que l'étude multi-temporelle se concentre sur le site géographique observé comme le plus défavorable lors de l'étude multi-localisations. En effet, sur cette localisation nous notons entre 4 et 15 erreurs auprès du serveur DNS par défaut, selon la date des tests. Nous constatons par ailleurs que ces erreurs se produisent sur des URLs différentes selon les dates à laquelle les tests sont effectués.

6.1.4.3 Synthèse IP

Au travers des résultats obtenus sur les 327 URLs testées dans cette seconde proposition, nous confirmons la staticité des adresses IP associées aux pages de login étudiées et ce, tant en terme de répartition géographique que temporelle.

Nous confirmons également que les serveurs DNS de référence interrogés donnent le même niveau de résultats en terme de convergence des adresses IPdef vs. IPref (86 à 87% en moyenne).

Parce que nous sommes désormais en mesure d'analyser l'adresse IP utilisée pour récupérer PageDef et PageRef, nous constatons également que dans une large majorité des cas (77 à 79% en moyenne), cette adresse IP est unique. Ceci semble donc confirmer que la vérification de l'adresse IP associée à une page de login est un critère pertinent pour la détection du pharming. D'autant plus que les taux d'échec

Tableau 6.6 – Comparaison des adresses IPdef et IPref utilisées pour récupérer les 328 pages webs légitimes, pour une même localisation géographique, sur une période de 3 mois

	Taux d'unicité des adresses IPdef et IPref (min ≤ moyenne ≤ max)	Écart-Type [1]	Intervalle de confiance à 95% [1]
OpenDNS	75.91% ≤ **77.29%** ≤ 78.05%	0.71%	[76.85% ; 77.73%]
GoogleDNS	76.22% ≤ **77.68%** ≤ 78.96%	1.02%	[77.05% ; 78.31%]
DNSAdvantage	76.52% ≤ **78.08%** ≤ 78.08%	1.00%	[75.90% ; 78.70%]

[1] entre les différentes dates de tests

Tableau 6.7 – Taux d'échec des requêtes DNS auprès des 4 serveurs DNS, sur 11 localisations géographiques

	Taux d'échec des requêtes DNS (min ≤ moyenne ≤ max)	Écart-Type [1]
Défaut	0% ≤ **0.44%** ≤ 3.35%	0.99%
OpenDNS	0% ≤ **0.11%** ≤ 0.61%	0.21%
GoogleDNS	0% ≤ **0.36%** ≤ 2.13%	0.67%
DNSAdvantage	0% ≤ **0.08%** ≤ 0.30%	0.14%

[1] entre localisations

des requêtes DNS observés auprès des serveurs DNS de référence sont quasi-nuls (entre 0 et 0.36%). Néanmoins, ceci peut être tempéré par des taux d'échec parfois plus élevé auprès des serveurs DNS par défaut (jusqu'à 3.57% en moyenne dans nos tests). Il reste donc indispensable d'utiliser une autre technique d'analyse, en combinaison avec la vérification de l'adresse IP, pour déterminer la légitimité d'une page web.

Enfin, il est à noter que la difficulté rencontrée dans la première approche, à savoir la réinitialisation du cache DNS Java, demeure un problème. A ce stade, nous n'avons trouvé aucune solution Java satisfaisante. Il pourra donc s'avérer nécessaire d'évaluer la portabilité de ce problème dans d'autres langages, en cas d'implémentation réelle de notre solution.

6.1.5 Analyse et comparaison du code source des pages webs

Les résultats obtenus lors de la première approche - bien qu'encourageants - nous ont amenés à conclure qu'aucune des deux méthodes évaluées (approche par caractères et par mots, appliquées au code complet) n'était suffisamment aboutie pour différencier de manière fiable (c.-à-d. sans faux-positifs et/ou faux-négatifs) les sites légitimes des sites contrefaits. En effet, la fenêtre de recouvrement entre le score maximum obtenu lors des comparaisons des sites légitimes-contrefaits, et le score minimum obtenu lors des comparaisons de sites légitimes, est de 11%.

Ces mêmes résultats laissent également entrevoir de meilleures performances avec l'approche par mots, c.-à-d. davantage d'écart entre les scores obtenus pour les comparaisons des pages légitimes-contrefaits vs. les scores obtenus pour les comparaisons de sites légitimes.

De plus, nous avons constaté que dans le cadre des comparaisons de sites légitimes, nos résultats étaient parfois impactés par des pages d'erreur récupérées en lieu et place de la page légitime réellement attendue.

Dans cette seconde proposition, nous concentrons donc nos efforts sur l'élaboration d'une méthode de détection plus aboutie et la définition d'un seuil de décision. De plus, nous cherchons à éliminer toute page d'erreur récupérée en lieu et place d'une page légitime. Enfin, nous évaluons l'efficacité de notre nouvelle méthode de récupération de PageRef, afin de voir si les taux d'échec sont diminués.

Tableau 6.8 – Taux d'échec des requêtes DNS auprès des 4 serveurs DNS, sur une même localisation géographique, durant 3 mois

	Taux d'échec des requêtes DNS (min ≤ moyenne ≤ max)	Écart-Type [1]
Défaut	0% ≤ **3.57%** ≤ 4.57%	1.19%
OpenDNS	**0%**	-
GoogleDNS	**0%**	-
DNSAdvantage	**0%**	-

[1] entre les différentes dates de tests

Les tests effectués dans cette section portent sur l'intégralité des 328 URLs sélectionnées (cf. section 6.1.2).

6.1.5.1 Méthodes étudiées : introduction de nouvelles techniques d'analyse du code source des pages webs

En complément des méthodes utilisées lors de la première approche, tant pour améliorer nos performances de détection que pour focaliser nos analyses aux zones de codes les plus pertinentes, nous nous sommes intéressés à deux nouvelles familles de pistes pour l'analyse du contenu des pages webs : 1/ application des méthodes de calcul à des sous-parties du code source complet, et 2/ analyse des balises contenues dans le code source.

En effet, à partir de la structure type d'un code source HTML rappelée en figure 6.3, nous avons scindé le code HTML des pages webs récupérées en plusieurs sous-parties. Ce découpage s'est effectué selon deux critères/hypothèses :

- La recherche d'une exhaustivité des zones analysées, par rapport au contenu global du code source récupéré. Ceci afin d'analyser la pertinence des différents éléments du code dans la différenciation d'un site légitime d'un site contrefait.
- La mise en exergue des éléments qui nous apparaissent comme les plus sensibles : les liens et les balises présents dans le code source. En effet, dans le cas d'une corruption a minima d'une page web, les premiers éléments modifiés seront les liens. Sinon, dans le cas d'une modification plus conséquente, nous supposons que l'attention portée à l'ordonnancement et/ou la présence des balises pourrait s'avérer révélateur d'une contrefaçon. En effet, toute modification de code (ajout, suppression, modification - hors substitution de taille identique) modifiera automatiquement le nombre et/ou l'emplacement des balises. A vérifier toutefois que les pages légitimes récupérées depuis plusieurs sources (c.-à-d. grâce aux informations fournies par plusieurs serveurs DNS) ne soient pas également impactées par des problèmes d'ordonnancement de balises liés à la localisation. En effet, si le serveur DNS de référence interrogé nous renvoie vers une page web hébergée dans une zone géographique très éloignée de notre point de téléchargement, nous serions peut-être confrontés à des modifications de la structure HTML, telles qu'évoquées en section 5.2.2.

Au final, nous avons donc porté notre analyse sur 7 zones de code source explicitées ci-après.

6.1.5.1.1 Application de la méthode de calcul au code source complet : Tel que vu dans la première proposition, nous avons appliqué nos méthodes de calcul par caractères et par mots à l'intégralité du *Code complet*. A noter que pour les pages légitimes, il s'agit en fait du *Code complet sans en-tête HTTP*. Comme son nom l'indique, cette sous-version *Code complet sans en-tête HTTP* contient le code source intégral, auquel a été retiré l'en-tête HTTP présent lors de la récupération de la page web effectuée par notre programme. Cette sous-version n'a donc d'intérêt que pour une application aux pages légitimes récupérées depuis les informations fournies par différents serveurs DNS. En effet, les pages contrefaites et légitimes associées ayant quant à elles été récupérées depuis un navigateur web, elles sont dépourvues de cet en-tête HTTP. Notons que cette sous-version a vu son apparition suite à la volonté d'élimination des pages d'erreur récupérées en lieu et place des pages légitimes attendues

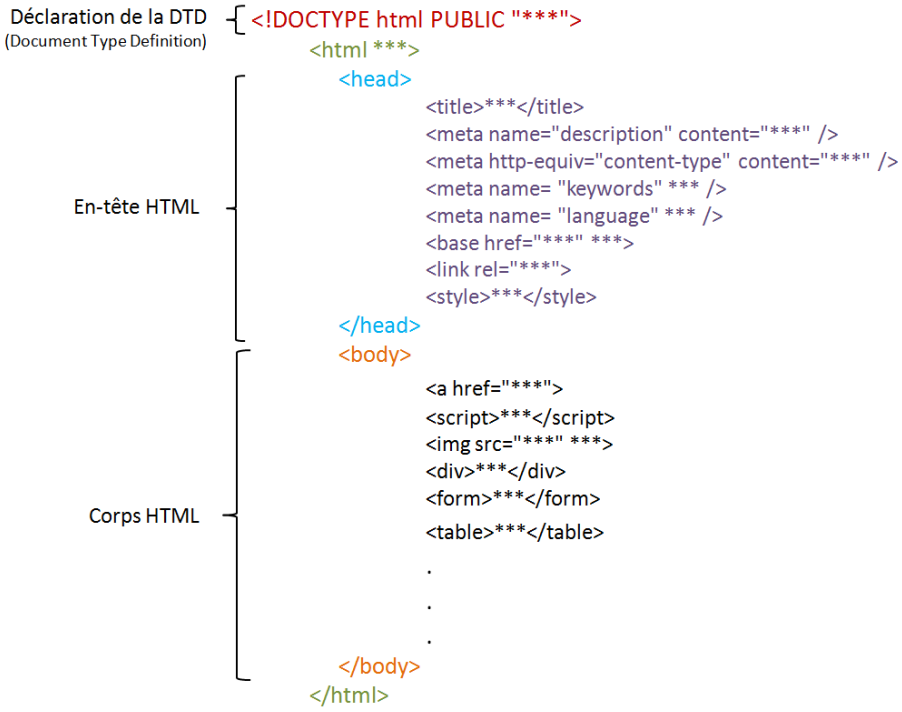


FIGURE 6.3 – Structure type du code source d'une page HTML

(pour plus de détails, cf. section 6.1.5.2).

6.1.5.1.2 Application de la méthode de calcul sur des sous-parties du code source complet : Le découpage du code source des pages webs en sous-parties s'est quant à lui intéressé à 4 zones : *Body*, *Head complet*, *Head contenu* et *Liens*.

Les sous-parties *Head complet* et *Body* correspondent respectivement à l'en-tête HTML incluant la déclaration de la DTD (Document Type Definition), et au corps de la page web. L'utilisation de la DTD, en amont du code source de la page web, n'est pas obligatoire mais elle est présente dans la quasi-totalité des pages légitimes récupérées. Située en préambule du document HTML, elle est interprétée et utilisée par le navigateur web du client, afin d'obtenir un meilleur rendu d'affichage de la page web. De par sa présence quasi-permanente et notre volonté d'exhaustivité dans les analyses menées, nous avons jugé utile de l'intégrer à nos comparaisons.

La sous-partie *Head contenu* correspond au contenu de l'en-tête HTML (incluant la DTD), hors balises. En effet, nous nous sommes aperçus que dans bon nombre de pages webs, le contenu de l'en-tête HTML contenait peu de "contenu réel" en dehors des balises et de leurs attributs.

Enfin, la sous-partie *Liens* contient les URLs relatives à un lien ou une image affichés dans la page web.

Sur chacune de ces sous-parties de code source, nous avons appliqué les deux techniques de comparaison : approches par caractères et par mots.

6.1.5.1.3 Analyse des balises contenues dans le code source complet : Cette analyse s'est portée sur l'extraction de 13 balises - considérées comme les plus pertinentes - contenues dans le code source complet.

En effet, le nombre de balises potentiellement utilisables en HTML étant tellement important, nous avons choisi de concentrer nos efforts sur celles que nous avons considéré comme les plus pertinentes, parmi une liste de plus de 90 balises. Ainsi nous avons sélectionné :

– 5 balises structurelles représentatives des paragraphes, sections, sauts de ligne, tableaux et fenêtres incorporées à la page.

- 2 balises relatives aux formulaires, typiquement utilisés pour des zones de saisie de login et mot de passe.
- 2 balises descriptives de la page qui contiennent le titre ou un descriptif de la page.
- 2 balises relatives aux scripts exécutés sur le poste client.
- 2 balises relatives aux liens et images qui ont un grand rôle dans l'aspect de la page, et/ou son appartenance à un domaine/arborescence de site.

A ces balises, nous avons appliqué deux types de calculs de scores qui réutilisent un peu le principe de l'approche par caractères : l'un de ces scores est dit par *occurrence*, tandis que l'autre est dit par *localisation*. Le principe de l'analyse par *occurrence* repose sur l'hypothèse suivante : si une partie de code est ajoutée ou supprimée, le nombre de balises est forcément impacté (à moins d'effectuer une substitution de balises). Le principe de l'analyse par *localisation* repose sur une deuxième hypothèse : si une modification est apportée à l'intérieur d'une balise (en rajoutant ou supprimant du contenu), ou de manière plus globale au code, la prise en compte du numéro de ligne où se trouve chaque balise sera forcément impacté.

Ainsi, basés sur ces hypothèses, le calcul de score par *occurrence* retourne, pour chaque balise, un score représentatif du nombre de fois où cette dernière apparaît dans le code complet :

$$\text{Score occurence balise(x)} = \sum_{i=1}^{n} présence(balise(x), i)$$

où x représente l'une des 13 balises étudiée

et n correspond au nombre de lignes du fichier étudié[1]

Sur le même principe, le calcul de score par *localisation* retourne, pour chaque balise, un score représentatif du nombre de fois où une balise apparaît, ainsi que la ligne où elle se situe dans le code complet :

$$\text{Score localisation balise(x)} = \sum_{i=1}^{n} présence(balise(x), i) \cdot i$$

où x est l'une des 13 balises étudiées,

n correspond au nombre de lignes du fichier étudié[1]

et i est le numéro de ligne où la balise a été trouvée dans le code source complet

.

6.1.5.2 Implémentation : points spécifiques

L'implémentation du GET HTTP permettant de récupérer PageRef - envoyé à destination d'une adresse IP pré-définie -, ainsi que la sélection de l'adresse IPref utilisée pour récupérer PageRef sont détaillées dans la section 6.1.4.1.

A noter que, dans le cas des pages légitimes récupérées depuis différentes localisations, les calculs de score utilisant les approches par caractères ou par mots, appliquées au code complet, ont été réalisés au téléchargement des pages webs. Tous les tests effectués sur le code modifié ou des sous-parties de celui-ci ont été réalisés après tests, selon les techniques décrites ci-après.

Les traitements réalisés sur couples de pages légitimes et couples de pages légitimes-contrefaites sont identiques.

6.1.5.2.1 Elimination des pages d'erreur et application de la méthode de calcul au code source complet : Une analyse plus détaillée des scores obtenus lors des comparaisons de pages légitimes effectuées dans la première approche, nous indiquent que bon nombre de pages délivrent un taux de similitude de l'ordre de 99.95 à 99.99%. Après investigations sur le contenu de ces pages, nous nous sommes aperçus que l'une de leurs différences majeures se résume à un horodatage. Celui-ci est présent dans l'en-tête HTTP incorporé dans le contenu de la page (cf. exemple en figure 6.4) dès lors que cette dernière est récupérée par un GET HTTP (tel qu'effectué par notre programme Java).

De plus, nous sommes confrontés à une difficulté conséquente, à savoir que certaines pages légitimes sont indisponibles pour diverses raisons (site en maintenance, page déplacée, etc.). Par conséquent, il nous arrive de récupérer des pages d'erreur en lieu et place des pages légitimes attendues. La comparaison effectuée en est alors faussée, ce qui impacte nos résultats.

1. Un exemple de fichier étudié est disponible en figure 6.7

```
HTTP/1.1 200 OK
Date: Sun, 03 Apr 2011 23:37:09 GMT
Server: Apache
Pragma: no-cache
Expires: 0
P3P: CP="NON CUR ADM DEV PSA OUR LEG STA"
Set-Cookie: sid25050000=test; path=/cgi; domain=banking.nordlb.de; secure
Cache-control: private, no-cache, no-store
Content-Length: 5038
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

FIGURE 6.4 – Exemple d'en-tête HTTP trouvé en préambule du code source d'une page web légitime

TABLEAU 6.9 – Exemples de codes d'en-tête HTTP, en préambule d'une page web légitime

Code HTTP	Catégorie	Signification	Quantité d'URLs associées sur le site de Bruxelles	
200	Succès	Page récupérée avec succès	282	85.98%
301	Redirection	Page déplacée de façon permanente	1	12.20%
302		Page déplacée de façon temporaire	37	
303		La réponse à cette requête est ailleurs	2	
400	Erreur du client	La syntaxe de la requête est erronée	1	0.91%
403		Authentification refusée	2	
500	Erreur du serveur	Erreur interne du serveur	3	0.91%

Théoriquement, toute indisponibilité de page web se doit d'être proprement spécifiée dans l'en-tête HTTP via un code de connexion HTTP approprié. Des exemples de codes HTTP courants que nous avons rencontrés sont spécifiés dans le tableau 6.9. A titre d'exemple, ce tableau mentionne également la proportion d'URLs associées à ces codes, sur une des localisations testées : Bruxelles.

Pour résoudre ces problèmes, nous avons donc créé une sous-version du code complet, intitulée *Code complet sans en-tête HTTP* qui contient le code source de la page web, hors en-tête HTTP.

Pour la suite des analyses, toutes nos comparaisons de pages légitimes effectuées sur le code complet se basent désormais sur ce nouveau fichier *Code complet sans en-tête HTTP* qui exclut l'en-tête HTTP.

6.1.5.2.2 Application de la méthode de calcul sur des sous-parties du code source complet : Le découpage du code source des pages webs en 4 sous-parties s'effectue, à raison d'un fichier créé par sous-partie, grâce aux techniques suivantes :

- le *Body* est récupéré par extraction de l'intégralité du contenu placé entre les balises <BODY> et </BODY>
- Le *Head complet* est obtenu par extraction de l'intégralité de la DTD indiquée dans l'élément <!DOCTYPE>, ainsi que du contenu de la page placé entre les balises <HEAD> et </HEAD>.
- *Head contenu* est récupéré par extraction du contenu de la DTD indiqué dans l'élément <!DOCTYPE>, ainsi que du contenu (hors balises) placé entre les balises <HEAD> et </HEAD> (par ordre de balises de même type). La figure 6.5 montre un exemple de fichier *Head contenu* créé.

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr" dir="ltr"
class="html-ltr firefox">
<head>
<meta name="description" content="Le site de MaBanque">
<script>Monscript</script>
<meta name="keywords" content="MaBanque, argent, prêt, livret A">
<title>Page de MaBanque</title>
</head>
...
```

(a) Code source

```
html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
xmlns="http://www.w3.org/1999/xhtml" lang="fr" dir="ltr" class="html-ltr
firefox"
Page de MaBanque
description Le site de MaBanque
keywords MaBanque, argent, prêt, livret A
Monscript
```

(b) Fichier de HEAD CONTENU créé

FIGURE 6.5 – Exemple de fichier créé pour la sous-partie HEAD CONTENU, à partir du code source complet d'une page web

– *Liens* est obtenu par extraction du contenu des balises <A ...> et <IMG>, tel qu'illustré en figure 6.6.

```
<html>
<head>
<title>Bienvenue sur MaBanque</title>
</head>
<body>
<img src="/images/Logo_MaBanque.jpg">
<br>Vous êtes ici sur la <b>page d'accueil</b>.
<br><a href="login.php">Se connecter</a>.
</body>
</html>
```

(a) Code source

```
IMG : images/Logo_MaBanque.jpg
URL: login.php
```

(b) Fichier de LIENS créé

FIGURE 6.6 – Exemple de fichier créé pour la sous-partie LIENS, à partir du code source complet d'une page web

6.1.5.2.3 Analyse des balises contenues dans le code source complet : Dans un premier temps, nous créons un fichier qui extrait l'ensemble des balises ouvrantes contenues dans le code complet, en association avec le numéro de ligne où elles apparaissent. La figure 6.7 illustre un exemple de fichier créé.

```
<html>
<head>
<title>Bienvenue sur MaBanque</title>
</head>
<body>
<img src="/images/Logo_MaBanque.jpg">
<br>Vous êtes ici sur la <b>page d'accueil</b>.
<br><a href="login.php">Se connecter</a>.
</body>
</html>
```

(a) Code source

```
1 html
2 head
3 title
5 body
6 img
7 br
7 b
8 br
8 a
```

(b) Fichier de balises créé

FIGURE 6.7 – Exemple de fichier créé pour l'analyse des balises, à partir du code source complet d'une page web

A partir de ce fichier, nous générons ensuite deux fichiers de scores (un par occurrence et un par localisation) : le premier ne tient compte que des noms de balises, tandis que le second tient également compte du numéro de ligne.

Les 13 balises précédemment sélectionnées sont identifiées par la recherche des champs suivants :

– les 5 balises structurelles : <p> pour les paragraphes, pour les sections,
 pour les sauts de ligne, <table> pour les tableaux et <iframe> pour les fenêtres incorporées à la page.
– les 2 balises relatives aux formulaires : <form> et <input>.
– les 2 balises descriptives de la page : <title> et <description>.
– les 2 balises relatives aux scripts exécutés sur le poste client : <script> et <noscript>.
– les 2 balises relatives aux liens et images : <a ...> (incluant <a href>, <a target>, etc.) et <img>.

Nous avons également envisagé d'extraire la balise <base href>, qui permet d'indiquer un chemin absolu commun aux liens de la page, mais nos analyses manuelles sur quelques dizaines de page ont montré une faible utilisation de celle-ci.

A noter que certaines balises peuvent contenir des attributs (p.ex. des indications de couleur, d'alignement ou de taille), placés entre guillemets, à l'intérieur du champ de balise. Dans ce cas, nous extrayons la balise sans ses attributs.

6.1.5.3 Résultats des analyses pour la définition des techniques de comparaisons les plus pertinentes

Cette section s'inscrit dans les analyses pour la définition des techniques de comparaisons les plus pertinentes. Elle porte sur l'application des approches par caractères et par mots aux code complet et

Tableau 6.10 – Taux de similitude des 328 pages légitimes avec l'approche par caractères appliquée au code complet, sur 11 localisations géographiques

	APPROCHE PAR CARACTÈRES sur code complet		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	99.67% ≤ **99.83%** ≤ 99.91%	0.07%	[99.79% ; 99.87%]
GoogleDNS	99.70% ≤ **99.84%** ≤ 99.89%	0.05%	[99.81% ; 99.87%]
DNSAdvantage	99.73% ≤ **99.84%** ≤ 99.90%	0.06%	[99.81% ; 99.87%]

[1] entre localisations

Tableau 6.11 – Taux de similitude des 328 pages légitimes avec l'approche par mots appliquée au code complet, sur 11 localisations géographiques

	APPROCHE PAR MOTS sur code complet		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	98.58% ≤ **98.86%** ≤ 99.07%	0.14%	[98.72% ; 99.00%]
GoogleDNS	98.34% ≤ **98.84%** ≤ 99.04%	0.19%	[98.65% ; 99.03%]
DNSAdvantage	98.55% ≤ **98.90%** ≤ 99.02%	0.14%	[98.76% ; 99.04%]

[1] entre localisations

sous-parties du code complet, ainsi qu'aux calculs de scores par occurrence et localisation appliqués aux balises.

6.1.5.3.1 Analyses sur le code complet : Les analyses sur le code complet portent sur les pages légitimes issues des 11 localisations, ainsi que sur celles issues d'une même localisation sur une période de 3 mois. En comparaison, elles s'intéressent également aux résultats obtenus sur les 75 couples de pages légitimes-contrefaites. Pour plus d'informations sur les échantillons et zones de tests, cf. section 6.1.2.

Taux de similitude des pages légitimes - étude multi-localisations : Les tableaux 6.10 et 6.11 indiquent les taux de similitude obtenus entre les pages PageDef et PageRef, en utilisant les deux approches par caractères et par mots. On constate que ces résultats confirment et améliorent les conclusions tirées lors de la première approche, à savoir que : le degré de convergence des pages est extrêmement élevé (entre 98 et 99% en moyenne) avec des écart-types très faibles (inférieurs à 0.20%). De plus, on peut noter que quel que soit le serveur DNS de référence interrogé pour récupérer IPref, les résultats des comparaisons sont très similaires, c.-à-d. les 3 serveurs DNS de référence sont comparables.

Ces excellents résultats peuvent notamment s'expliquer par la forte proportion d'URLs (77 à 78%, cf. section 6.1.4.2) qui ne retournent qu'une seule et même adresse IP pour le FQDN interrogé, et ce quel que soit le serveur DNS interrogé.

De plus, on peut également observer que la suppression de l'en-tête HTTP du code source complet a permis d'améliorer les résultats obtenus. Ceci est d'autant plus significatif avec l'approche par mots, dont les résultats sont désormais de niveau quasi-identique à ceux obtenus avec l'approche par caractères.

Taux de similitude des pages légitimes - étude multi-temporelle depuis une même localisation : Depuis une même localisation, les tableaux 6.12 et 6.13 nous amènent aux mêmes conclusions que celles observées lors de l'étude multi-localisations, à savoir : des degrés de convergences des PageDef et PageRef extrêmement élevés (97 à 99% en moyenne), des écart-types très faibles (inférieurs à 0.30%), et des résultats quasi-identiques quel que soit le DNS de référence utilisé. Ceci semble donc être vecteur d'une bonne stabilité des résultats dans le temps.

TABLEAU 6.12 – Taux de similitude des 328 pages légitimes avec l'approche par caractères appliquée au code complet, pour une même localisation sur une période de 3 mois

	APPROCHE PAR CARACTÈRES sur code complet		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	99.64% ≤ **99.81%** ≤ 99.89%	0.09%	[99.75% ; 99.87%]
GoogleDNS	99.62% ≤ **99.84%** ≤ 99.90%	0.08%	[99.79% ; 99.89%]
DNSAdvantage	99.65% ≤ **99.85%** ≤ 99.91%	0.08%	[99.80% ; 99.90%]

[1] entre localisations

TABLEAU 6.13 – Taux de similitude des 328 pages légitimes avec l'approche par mots appliquée au code complet, pour une même localisation sur une période de 3 mois

	APPROCHE PAR MOTS sur code complet		
	Taux de similitude avec la page issue de IPdef (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]
OpenDNS	96.89% ≤ **97.36%** ≤ 97.60%	0.25%	[97.20% ; 97.52%]
GoogleDNS	96.88% ≤ **97.44%** ≤ 97.65%	0.29%	[97.26% ; 97.62%]
DNSAdvantage	96.88% ≤ **97.49%** ≤ 97.66%	0.24%	[97.34% ; 97.64%]

[1] entre localisations

Taux de similitude des pages légitimes-contrefaites : Concernant les résultats obtenus sur les pages légitimes-contrefaites (cf. tableau 6.14), on constate que - tel qu'aperçu lors de la première approche - la méthode par mots se démarque nettement. En effet, le taux de similitude moyen observé est plus faible avec l'approche par mots (43%, contre 76% avec l'approche par caractères), même si les valeurs minimales (0 à 10%) et maximales (96 à 99%) sont de niveau comparable pour les deux approches.

De plus, bien que les intervalles de confiance à 95% des deux approches nous retournent des valeurs maximales (50 ou 83% selon l'approche) bien en deçà des valeurs minimales obtenues sur les comparaisons de pages légitimes (96% au minimum), on peut constater que le taux de faux-négatifs reste bien trop élevé avec l'approche par caractères. En effet, un éventuel seuil de décision fixé à 90 ou 96% laisse apparaître un facteur 10 entre les taux de faux-négatifs résiduels obtenus avec les deux approches (57 vs. 5% avec un seuil décisionnel à 90%, ou 17 vs. 1% avec un seuil décisionnel à 96%).

6.1.5.3.2 Analyses sur les sous-parties du code complet : Les analyses sur les sous-parties du code complet portent exclusivement sur les 75 couples de pages légitimes-contrefaites. En effet, de par les résultats de comparaisons extrêmement élevés obtenus sur l'analyse du code complet des pages légitimes, nous avons jugé inutile de détailler plus avant les résultats de même calibre obtenus sur les sous-parties de code complet associées.

A contrario, il apparaît nettement plus intéressant de focaliser ces analyses sur les pages légitimes-contrefaites, afin d'essayer d'identifier des zones révélatrices des contrefaçons perpétrées.

Avec l'approche par caractères (cf. tableau 6.15), on peut constater que les sous-parties *Liens* et *Body* semblent être, en moyenne, les plus affectées par des changements. Néanmoins, aucune des 4 sous-parties étudiées ne se distingue vraiment. Toutes conduisent à des taux de faux-négatifs trop élevés, et ce quel que soit le seuil décisionnel envisagé (de 21 à 40% pour un seuil à 90%, et de 13 à 22% avec un seul de décision à 96%).

En étudiant les résultats obtenus avec l'approche par mots (cf. tableau 6.16), on constate également que les sous-parties *Liens* et *Body* sont, en moyenne, les plus affectées par les changements. La partie *Liens* est d'ailleurs celle qui obtient le score moyen le plus bas. Ceci peut notamment s'expliquer par

Tableau 6.14 – Taux de similitude du code complet de 75 couples de pages légitimes-contrefaites

	Taux de similitude entre pages légitimes et contrefaites (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]	Taux de faux-négatifs si le seuil de décision est 90%	96%	Approche déterminante pour % couples de sites
CARACTÈRES	10.83% ≤ **76.91%** ≤ 99.93%	27.12%	[70.78% ; 83.05%]	57.33%	17.33%	0%
MOTS	0% ≤ **43.45%** ≤ 96%	33.11%	[35.96% ; 50.95%]	5.33%	1.33%	100%

[1] entre couples de pages

Tableau 6.15 – Taux de similitude des sous-parties du code complet de 75 couples de pages légitimes-contrefaites, en utilisant l'approche par caractères

	APPROCHE PAR CARACTÈRES **sur sous-parties du code complet**					
	Taux de similitude entre pages légitimes et contrefaites (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]	Taux de faux-négatifs si le seuil de décision est 90%	96%	Approche déterminante pour % couples de sites
BODY	12.10% ≤ **68.68%** ≤ 100%	28.20%	[62.26% ; 75.11%]	26.67%	16.00%	6.67%
HEAD COMPLET	11.05% ≤ **76.57%** ≤ 100%	23.32%	[71.29% ; 81.85%]	40.00%	22.67%	16.00%
HEAD CONTENU	12.41% ≤ **72.49%** ≤ 100%	26.23%	[66.55% ; 78.43%]	34.67%	20.00%	28.00%
LIENS	0.55% ≤ **58.96%** ≤ 99.87%	33.37%	[51.36% ; 66.56%]	21.33%	13.33%	20.00%
CODE COMPLET[2]	10.83% ≤ **76.91%** ≤ 99.93%	27.12%	[70.78% ; 83.05%]	57.33%	17.33%	0%

[1] entre couples de pages
[2] ces résultats issus du tableau 6.14 sont reportés ici pour une meilleure lisibilité et comparaison des résultats

les contrefaçons de pages webs effectuées à minima - afin de rendre la détection de contrefaçon plus difficile - qui ne change que quelques liens (typiquement des zones de login) de la page originale.

De plus, on constate que la sous-partie *Liens* est celle qui conduit au taux de faux-négatifs résiduel le moins élevé : si le seuil décisionnel est établi à 90%, le taux de FNR s'avère identique à celui observé sur code complet (c.-à-d. 5.33%). Il devient alors nul si le seuil est fixé à 96%.

6.1.5.3.3 Analyses sur les balises : Les analyses sur les balises portent sur les 75 couples de pages légitimes-contrefaites, ainsi que sur les pages légitimes issues de 6 localisations : Bruxelles, Mexico, Montpellier, Samoreau, Dakar et Shenzhen.

On peut constater, sans surprise, que l'analyse des balises par occurrence et localisation tend à indiquer très peu de changements sur le nombre et l'emplacement des balises contenues dans les pages webs légitimes. En effet, les taux de similitude moyens observés sur 6 localisations varient de 98 à 100% pour des écart-types de 0 à 12% (cf. tableau 6.17 et 6.18).

Concernant l'analyse des pages légitimes et contrefaites, on constate cette fois que les résultats obtenus sont nettement plus bas (vs. ceux obtenus avec les pages légitimes) tant en terme d'occurrence que de localisation : les taux de similitude sont respectivement de 31 à 75% (hors balise <title>) et 16 à 59% (cf. tableaux 6.19 et 6.20). Néanmoins, quelle que soit la balise observée, les écart-types sont très importants : 24 à 45% (hors balise <title>) pour l'occurrence, et 38 à 43% pour la localisation. Ceci rend difficilement exploitables les résultats des balises comme seuls critères de décision.

A noter que les balises descriptives (<title> et <description>) sont non pertinentes pour l'analyse par occurrence et/ou localisation, car toujours présentes/identiques ou absentes.

6.1.5.3.4 Synthèse des méthodes les plus pertinentes : Au vu des résultats d'analyses sur l'étude des méthodes les plus pertinentes, on peut retenir que :

- L'approche par caractères, qu'elle soit appliquée au code complet ou sous-parties du code complet, ne permet pas d'aboutir à des résultats concluants. En effet, le taux de faux-négatifs résiduel potentiel demeure trop important. Cette méthode n'est donc pas retenue pour l'élaboration de notre technique de comparaison finale.
- L'approche par mots appliquée au code complet donne des résultats très intéressants mais, le taux

Tableau 6.16 – Taux de similitude des sous-parties du code complet de 75 couples de pages légitimes-contrefaites, en utilisant l'approche par mots

	APPROCHE PAR MOTS sur sous-parties du code complet					
	Taux de similitude entre pages légitimes et contrefaites (min ≤ moyenne ≤ max)	Écart-Type[1]	Intervalle de confiance à 95%[1]	Taux de faux-négatifs si le seuil de décision est 90%	96%	Approche déterminante pour % couples de sites
BODY	0% ≤ **44.35%** ≤ 97%	33.31%	[36.76% ; 51.94%]	13.33%	1.33%	93.33%
HEAD COMPLET	0% ≤ **53.33%** ≤ 100%	35.58%	[45.28% ; 61.39%]	24.00%	6.67%	84.00%
HEAD CONTENU	0% ≤ **54.40%** ≤ 100%	36.31%	[46.18% ; 62.62%]	24.00%	8.00%	72.00%
LIENS	0% ≤ **40.86%** ≤ 96.00%	31.20%	[33.76% ; 47.97%]	5.33%	-	80.00%
CODE COMPLET	0% ≤ **43.45%** ≤ 96%	33.11%	[35.96% ; 50.95%]	5.33%	1.33%	100%

[1] entre couples de pages
[2] ces résultats issus du tableau 6.14 sont reportés ici pour une meilleure lisibilité et comparaison des résultats

Tableau 6.17 – Taux de similitude par occurrence des balises de 328 pages légitimes de 6 localisations

	Balises												
	title	description	a	img	script	noscript	p	br	table	div	iframe	form	input
Moyenne[1]	99%	-	99%	99%	99%	98%	99%	99%	99%	99%	100%	99%	99%
Écart-type[1]	5%	-	6%	5%	6%	10%	7%	6%	7%	5%	0%	7%	7%

[1] entre couples de pages

de faux-négatifs résiduel - bien que faible - n'est pas nul. Cette technique ne peut donc se suffire à elle-même pour déterminer la légitimité d'une page web.
- L'analyse des sous-parties du code complet, en utilisant l'approche par mots, laisse apparaître que la sous-partie *Liens* est la plus pertinente. Elle permet en effet d'obtenir le score moyen le plus bas et le taux de FNR résiduel le plus intéressant. Une attention particulière est donc à accorder à cette sous-partie *Liens* dans l'élaboration de notre méthode de comparaison finale.
- L'analyse des balises - tant en terme d'occurrence que de localisation - nous démontre une nette différence entre les taux de similitude obtenus sur pages légitimes vs. pages légitimes-contrefaites. Néanmoins, ces résultats sont tempérés par des écart-types très élevés obtenus sur pages légitimes-contrefaites. L'intégration des scores des balises dans la méthode de comparaison finale peut donc s'avérer intéressante, mais non suffisante.
- Enfin, les taux de similitude minimum obtenus sur pages légitimes laissent entrevoir un seuil de décision maximum de 96% (cf. tableaux 6.10 à 6.13). En effet, les résultats de nos études multi-localisations et multi-temporelles sur code complet sont toutes supérieures à cette valeur. De plus, les intervalles de confiance à 95% associés sont supérieurs à 97%.

6.1.5.4 Résultats d'analyses des taux d'échec de récupération des pages Défaut et Référence :

En comparaison des résultats obtenus lors de la première approche, nous constatons que notre nouvelle technique de récupération de PageRef (c.-à-d. basée sur une URL non modifiée, envoyée à destination d'une adresse IP définie) a permis de diminuer considérablement le taux d'échec de récupération de page associée. En effet, nous observons désormais que le taux d'échec constaté lors de l'étude multi-localisation est de l'ordre de 4% en moyenne (cf. tableau 6.21), contre 22% en moyenne lors de la première approche, rejoignant ainsi les taux d'échec rencontrés pour PageDef.

A noter que pour l'étude multi-localisations, un site géographique nous retourne des taux d'échec plus élevés que la moyenne : 15% constatés au Vénézuéla. Ces échecs s'expliquent notamment par les pertes de connectivité rencontrées sur ce site.

Deux autres sites géographiques (Samoreau et Shenzhen) nous retournent des taux d'échec d'environ 6 et 7%, taux constatés tant pour PageDef que PageRef. Une étude plus poussée sur les raisons de ces échecs nous amène à deux constats : 1/ sur une même localisation, les URLs concernées sont les mêmes,

Tableau 6.18 – Taux de similitude par localisation des balises de 328 pages légitimes de 6 localisations

	Balises												
	title	description	a	img	script	noscript	p	br	table	div	iframe	form	input
Moyenne[1]	99%	-	99%	99%	99%	98%	99%	99%	99%	99%	100%	99%	99%
Écart-type[1]	7%	-	9%	8%	8%	12%	9%	8%	8%	8%	1%	9%	9%

[1] entre couples de pages

Tableau 6.19 – Taux de similitude par occurrence des balises de 75 couples de pages légitimes-contrefaites

	Balises												
	title	description	a	img	script	noscript	p	br	table	div	iframe	form	input
Moyenne[1]	100%	-	75%	74%	72%	57%	68%	61%	62%	77%	31%	85%	71%
Écart-type[1]	0%	-	29%	26%	34%	42%	36%	38%	45%	27%	46%	24%	29%

[1] entre couples de pages

que la page web soit récupérée auprès de IPdef ou de l'une des trois adresses IPref, et 2/ ces erreurs sont dues à des échecs d'échanges SSL réalisés en préambule de la récupération de la page web.

Enfin, les 8 sites géographiques restants retournent des taux d'échec inférieurs à 4%.

L'étude multi-temporelle porte sur le site géographique de Samoreau(France-IdF) qui nous a retourné des taux d'échec d'environ 6% lors de l'étude multi-localisations. On constate ici que les taux d'échec demeurent relativement stables dans le temps, autour de 6% en moyenne (cf. tableau 6.22). A nouveau, une investigation plus poussée sur ces taux d'échec indique : 1/ une constance des URLs concernées sur une même date (c.-à-d. les pages web récupérées depuis IPdef ou l'une des 3 IPref ont toutes échoué), 2/ les URLs concernées sont assez variables d'une date de test à une autre, et 3/ ces erreurs sont dues à des échecs d'échanges SSL réalisés en préambule de la récupération de la page web. A ce stade, nous n'avons pas eu le temps de revenir à une éventuelle correction de notre implémentation pour supprimer ces erreurs. De plus, le caractère assez changeant des URLs concernées d'une date à l'autre laisse perplexe.

Les résultats obtenus ici confirment donc nos hypothèses sur les causes supposées des taux d'erreur de récupération de PageRef, tels qu'observés dans la première approche. La seconde approche, telle qu'elle a été conçue, permet donc de revenir à des taux d'échec raisonnables. Toutefois, on constate que des erreurs résiduelles demeurent sur certaines localisations, erreurs engendrées par des problèmes d'établissement de la connexion SSL.

6.1.5.5 Résultats des Analyses pour l'élaboration de la méthode de comparaison finale

Au travers des conclusions tirées à l'issue de la section 6.1.5.3, nous avons donc choisi d'élaborer notre méthode de comparaison finale en tenant compte des éléments suivants :

- un seuil de décision qui ne peut excéder 96%,
- l'application de la méthode par mots au code complet,
- l'application de la méthode par mots au sous-fichier *Liens*,
- et l'utilisation des scores par occurrence et localisation associés aux balises.

A noter que nous nous sommes concentrés ici sur 5 des 13 balises étudiées précédemment, balises que nous considérons comme les plus importantes, à savoir : les 2 balises relatives aux liens et images (<a> et <img>), la balise de <script> et les 2 balises de formulaires (<form> et <input>). L'ensemble des 13 balises nous a en effet donné le même type de résultats tant sur pages légitimes-contrefaites que sur pages légitimes. Par manque de temps, nous avons concentré nos efforts sur les 5 balises les plus assujetties aux attaques.

Notons également qu'il n'y a aucun recoupement entre les pages utilisées pour l'étalonnage et la vérification des résultats obtenus ici.

Tableau 6.20 – Taux de similitude par localisation des balises de 75 couples de pages légitimes-contrefaites

	Balises												
	title	description	a	img	script	noscript	p	br	table	div	iframe	form	input
Moyenne[1]	59%	-	49%	44%	43%	31%	43%	44%	46%	51%	16%	47%	46%
Écart-type[1]	39%	-	39%	38%	37%	38%	42%	39%	43%	41%	32%	40%	38%

[1] entre couples de pages

Tableau 6.21 – Taux d'échec de récupération des pages webs légitimes en utilisant IPdef ou IPref, sur 11 localisations géographiques

URL utilisant l'adresse IP fournie par le DNS	**Taux d'échec** de récupération des pages webs (min ≤ moyenne ≤ max)	Écart-Type[1]
Défaut	1.52% ≤ **4.52%** ≤ 15.55%	4.13%
OpenDNS	1.52% ≤ **4.43%** ≤ 15.85%	4.15%
GoogleDNS	1.52% ≤ **4.68%** ≤ 15.85%	4.13%
DNSAdvantage	1.83% ≤ **4.63%** ≤ 15.55%	2.35%

[1] entre localisations

6.1.5.5.1 Étalonnage : L'étalonnage de notre méthode de comparaison finale s'est donc effectué sur 55 couples de pages légitimes-contrefaites et les pages légitimes issues de Bruxelles(Belgique) / couples de pages DNSdef et GoogleDNS (pour plus de détails, cf. section 6.1.2).

Après analyse des scores intermédiaires (sur balises par occurrence, balises par localisation, code complet et *Liens*) obtenus tant sur pages légitimes que sur pages légitimes-contrefaites, nous nous sommes aperçus que l'application de la méthode par mots sur la sous-partie *Liens* pouvait s'avérer fortement intéressante sur pages légitimes-contrefaites, tandis qu'elle pouvait s'avérait parfois pénalisante sur pages légitimes. En effet, en dehors du problème d'horodatage des pages éliminé précédemment, l'essentiel des divergences entre pages légitimes résident dans des modifications de liens, utilisées pour l'affichage de contenus dynamiques tels que des images.

De ce constat, nous avons donc introduit 3 méthodes de calcul :

1. le score final est obtenu en accordant un poids identique aux : balises par occurrence, balises par localisation et code complet. Cette méthode sera nommée **(B) : Balises + Code complet**.
2. le score final est calculé en accordant un poids identique aux balises vs. le code complet. Ainsi le score est déterminé à partir de : balises par occurrence, balises par localisation et (2 × code complet). Cette méthode sera nommée **(C) : Balises + Code complet × 2**.
3. le score final est obtenu en accordant un poids identique aux : balises par occurrence, balises par localisation, *Liens* et code complet. Cette méthode sera nommée **(D) : Balises + Liens + Code complet**.

Précisons qu'à des fins de comparaisons, les 3 méthodes sont systématiquement comparées entre elles, ainsi qu'à la méthode par mots appliquée au code complet seul. Cette dernière est nommée **(A) : Code complet**.

A noter que dans les tableaux qui suivent, les meilleurs résultats (c.-à-d. les plus intéressants pour une décision de légitimité des pages) apparaissent en gras afin d'améliorer leur visibilité.

Pages légitimes - Bruxelles : Le tableau 6.23 sur pages légitimes indique clairement que la méthode **(B) : Balises + Code complet** est la plus intéressante sur pages légitimes. En effet, c'est la méthode qui délivre le taux de similitude moyen le plus élevé (99.61%), l'écart-type le plus faible (0.92%) et le taux de similitude minimum le plus élevé (90.96%).

Pages légitimes-contrefaites - 55 couples de pages : Le tableau 6.24 sur pages légitimes-contrefaites délivre un message plus complexe. En effet, le taux de similitude moyen le plus faible (44.67%) est

Tableau 6.22 – Taux d'échec de récupération des pages webs légitimes en utilisant IPdef ou IPref, pour une même localisation sur une période de 3 mois

URL utilisant l'adresse IP fournie par le DNS	**Taux d'échec** de récupération des pages webs (min ≤ moyenne ≤ max)	Écart-Type[1]
Défaut	5.49% ≤ **6.52%** ≤ 7.32%	0.60%
OpenDNS	5.49% ≤ **6.28%** ≤ 7.01%	0.56%
GoogleDNS	5.79% ≤ **6.55%** ≤ 7.93%	0.66%
DNSAdvantage	5.49% ≤ **6.28%** ≤ 7.01%	0.46%

[1] entre localisations

Tableau 6.23 – Résultats d'étalonnage des méthodes de comparaisons finales sur 328 pages légitimes récupérées à Bruxelles

	Taux de similitude Moyen	Écart-Type[1]	Taux de similitude Minimum	Quantité de Faux-positif[2]
(A) : Code complet	99.02%	2.08%	74.00%	2
(B) : Balises + Code complet	**99.61%**	**0.92%**	**90.96%**	**0**
(C) : Balises + Code complet × 2	99.46%	1.17%	86.72%	1
(D) : Balises + Liens + Code complet	99.12%	2.27%	82.72%	4

[1] entre couples de pages
[2] pour un seuil de décision à 90%

obtenu avec l'approche **(A) : Code complet**, tandis que l'écart-type le plus faible (20.83%) et le taux de similitude maximum le plus faible (87.24%) sont obtenus avec l'approche **(D) : Balises + Liens + Code complet**. Par ailleurs, en comparaison avec l'étalonnage réalisé sur pages légitimes, la seule méthode qui ne présente aucun espace de recouvrement est la méthode **(B) : Balises + Code complet** : le taux maximum obtenu sur pages légitimes-contrefaites est de 89.37%, tandis que le taux minimum sur pages légitimes est de 90.96%. Il apparaît donc qu'un seuil de décision fixé à 90% semble le meilleur choix. En effet, il permettrait dans les deux cas d'avoir des taux de faux-négatifs (c.-à-d. pages contrefaites déclarées légitimes à tort) et de faux-positifs (c.-à-d. pages légitimes déclarées contrefaites à tort) nuls.

On peut toutefois remarquer que l'approche **(C) : Balises + Code complet × 2** délivre des résultats relativement proches de la méthode choisie.

Il est également notable que les 3 approches proposées délivrent de bien meilleurs résultats que la seule approche par mots appliquée au code complet.

Au final, suite à cet étalonnage, nous retenons la méthode **(B) : Balises + Code complet**, associée à un seuil de décision de **90%** :

$$TS_{final} = \text{Moyenne}\ (TS_{Balises\_occurence},\ TS_{Balises\_localisation},\ TS_{Code\_complet})$$

où TS est le taux de similitude

6.1.5.5.2 Test de l'efficacité de la méthode de comparaison retenue : Les tests sur l'efficacité de notre méthode de comparaison finale se sont effectués sur 58 couples de pages légitimes-contrefaites et les pages légitimes issues de 5 localisations : Mexico, Montpellier, Samoreau, Dakar et Shenzhen, récupérées à partir des adresses retournées par différents serveurs DNSdef et DNSref (pour plus de détails, cf. section 6.1.2).

Tableau 6.24 – Résultats d'étalonnage des méthodes de comparaisons finales sur 55 couples de pages légitimes-contrefaites

	Taux de similitude Moyen	Écart-Type[1]	Taux de similitude Maximum	Quantité de Faux-négatif[2]	Espace de recouvrement[3]
(A) : Code complet	**44.67%**	32.49%	96%	2	22%
(B) : Balises + Code complet	59.00%	21.64%	89.37%	**0**	-
(C) : Balises + Code complet × 2	55.42%	23.46%	87.53%	**0**	1%
(D) : Balises + Liens + Code complet	54.57%	**20.83%**	**87.24%**	4	5%

[1] entre couples de pages
[2] pour un seuil de décision à 90%
[3] avec les résultats sur pages légitimes, cf tableau 6.23

Pages légitimes - étude multi-localisations : Le tableau 6.25 sur pages légitimes issues de 5 localisations indique très clairement que l'approche **(B) : Balises + Code complet** délivre les meilleurs résultats tant en terme de taux de similitude moyen, que de faux-positifs, et ce quelle que soit la localisation. Concernant les écart-types et le taux de similitude minimum, cette même approche délivre les meilleurs résultats sur, respectivement, 4 et 3 localisations.

Nous nous sommes alors intéressés aux faux-positifs afin de comprendre les raisons de ces mauvaises détections. Nous identifions alors 2 pages qui posent problème :

– la page du site `twitter.com` qui contient une grande part de contenu dynamique, telles que des images publicitaires. On peut d'ailleurs voir un aperçu de cette page web où apparaît un bandeau d'images dynamiques sur la figure 6.8. Ainsi, bien que les balises soient inchangées, beaucoup de liens sont partiellement modifiés ce qui impacte à la baisse le score du code complet (et donc le score global de la méthode). Précisons toutefois que la méthode de calcul finale retenue délivre le score le plus élevé (88.62%) pour cette page, par rapport aux 3 autres méthodes. Cette page correspond à 1 faux-positif observé sur 3 localisations : Montpellier, Dakar et Mexico.

Figure 6.8 – Page de login `twitter.com` incluant un bandeau d'images dynamiques

– la page du site `www.myspace.com` pose également problème sur la localisation de Mexico. Nous constatons en effet que la PageDef récupérée est écrite en espagnol, tandis que la PageRef récupérée est écrite en anglais. Ainsi, bien que les occurrences de balises soient inchangées, le texte contenu est très différent. Ceci impacte à la baisse à la fois le score obtenu pour le code complet et la localisation des balises. Précisons toutefois que la méthode de calcul finale retenue délivre le score le plus élevé (71.39%) pour cette page, par rapport aux 3 autres méthodes.

Pages légitimes-contrefaites - 58 couples de pages : Le tableau 6.26 sur pages légitimes-contrefaites délivre le même type de message que lors de l'étalonnage. A savoir que les meilleurs résultats

Tableau 6.25 – Résultats des méthodes de comparaisons finales sur 328 pages légitimes de 5 localisations

	Localisation	Taux de similitude Moyen	Écart-Type[1]	Taux de similitude Minimum	Quantité de Faux-positif[2]
A	Mexico	98.66%	4.85%	33.00%	4
	Montpellier	98.80%	3.84%	43.00%	3
	Samoreau	98.89%	2.14%	74.00%	2
	Dakar	98.98%	2.28%	71.00%	2
	Shenzhen	98.94%	2.21%	73.00%	2
				Total	*13*
B	Mexico	**99.47%**	2.23%	66.33%	2
	Montpellier	**99.54%**	**1.36%**	82.51%	1
	Samoreau	**99.57%**	**1.01%**	**90.90%**	0
	Dakar	**99.59%**	**0.99%**	**89.96%**	1
	Shenzhen	**99.58%**	**0.89%**	**91.00%**	0
				Total	***4***
C	Mexico	99.36%	**1.91%**	**74.50%**	2
	Montpellier	99.41%	1.41%	**85.38%**	1
	Samoreau	99.43%	1.25%	86.67%	1
	Dakar	99.44%	1.28%	85.22%	1
	Shenzhen	99.42%	1.19%	86.50%	1
				Total	*6*
D	Mexico	98.89%	3.26%	66.33%	6
	Montpellier	99.04%	2.39%	85.25%	5
	Samoreau	99.03%	2.52%	83.67%	5
	Dakar	99.06%	2.51%	81.47%	5
	Shenzhen	99.07%	2.32%	82.25%	5
				Total	*26*

[1] entre couples de pages
[2] pour un seuil de décision à 90%

seraient obtenus avec l'approche **(D) : Balises + Liens + Code complet**. Toutefois, tel que vu précédemment, cette méthode délivre des résultats catastrophiques sur les pages légitimes en produisant 26 faux-positifs. Il apparaît alors ici que la méthode choisie **(B) : Balises + Code complet** ne délivre que 3 sites faux-négatifs.

Nous nous sommes intéressés à ces faux-négatifs afin de comprendre les raisons des mauvaises détections. Nous identifions alors 3 pages qui posent problème :

- les pages des sites `westernunion` et `scotiabank` présentent le même cas de figure. En effet, le chemin de base des URLs est modifié en début de page web puis, toutes les URLs suivantes sont modifiées en chemins relatifs (alors que les pages légitimes utilisent des chemins absolus). Ainsi, de par notre élimination des *Liens* dans la méthode de comparaison finale, ces changements sont moins visibles. Toutefois, ils apparaissent très nettement dans le score obtenu sur le code complet.
- la page du site `rbc` présente un autre cas de figure, totalement invisible pour l'utilisateur si utilisé dans une attaque de pharming (cf. figure 6.9). On constate en effet que la (quasi-)seule modification de cette page - par rapport à son pendant légitime -, est l'inclusion d'un script (20 lignes ajoutées par rapport à une page de 502 lignes, cf. figure 6.10) utilisé dans le cadre d'une redirection des zones de saisie des champs login et mot de passe. Ainsi, les modifications apportées sont uniquement visibles dans le score des balises par localisation, ce qui affecte insuffisamment le score final avec la méthode retenue.

Pour compléter cette analyse des faux-négatifs, la section 6.1.5.7 s'intéresse à étudier la quantité de modifications qu'il faut apporter à une page web, afin que celles-ci soient détectées par notre méthode de calcul finale.

6.1.5.6 Problèmes rencontrés

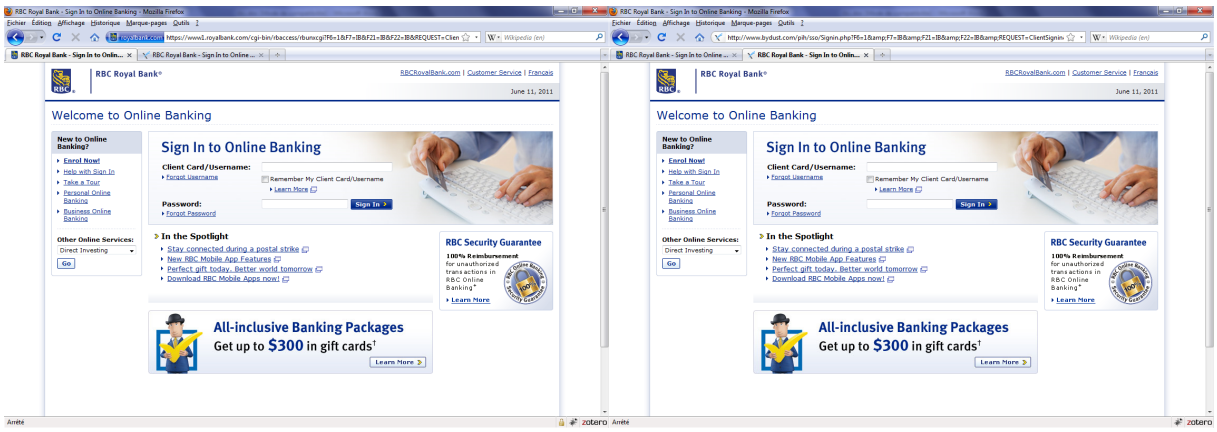
6.1.5.6.1 Elimination des pages d'erreur : La technique d'élimination des pages d'erreur, telle qu'expliquée en section 6.1.5.2 n'est pas infaillible. En effet, il nous est arrivé de rencontrer des pages d'erreur improprement codées dans l'en-tête HTTP. Par exemple, alors que l'en-tête HTTP indiquait un code "200

Tableau 6.26 – Résultats des méthodes de comparaisons finales sur 58 couples de pages légitimes-contrefaites

	Taux de similitude Moyen	Écart-Type[1]	Taux de similitude Maximum	Quantité de Faux-négatif[2]
(A) : Code complet	**47.21%**	33.10%	96.00%	4
(B) : Balises + Code complet	52.47%	27.13%	94.03%	**3**
(C) : Balises + Code complet × 2	51.15%	27.77%	93.01%	**3**
(D) : Balises + Liens + Code complet	49.58%	**25.87%**	**92.51%**	1

[1] entre couples de pages
[2] pour un seuil de décision à 90%

OK", le contenu de la page faisait mention d'une indisponibilité de site. Par conséquent, dès lors que nous obtenions des résultats de comparaisons de pages légitimes surprenants, nous nous sommes assurés à chaque fois manuellement qu'aucune des deux pages récupérées ne correspondait pas à ce cas de figure. Si tel était le cas, le couple de page était éliminé des résultats d'analyses.



(a) Site légitime (b) Site contrefait

Figure 6.9 – Captures d'écran du site légitime RBC (`https://https://www1.royalbank.com/cgi-bin/rbaccess/...`) et d'une contrefaçon (`http://www.bydust.com/pih/sso/Signin.php?...`), récupérées le 11 Juin 2011

```
<script language="JavaScript" type="text/javascript">
<!--
  function checkFields()
            {
            if (document.rbunxcgi.K1.value == '')
            {
                alert ( 'Please Enter Your Client Card/Username' );
    document.rbunxcgi.K1.focus();
    return false;
            }

            if (document.rbunxcgi.Q1.value == '')
            {
              alert ( 'Please Enter Your Password' );
    document.rbunxcgi.Q1.focus();
              return false;
            }

            }
          //-->
</script>
```

Figure 6.10 – Script utilisé dans le cadre d'une redirection des champs login et mot de passe, ajouté à la page légitime RBC

6.1.5.6.2 Création des sous-parties du code complet et analyse des balises : Une difficulté associée à notre technique d'extraction des balises est la présence d'erreurs syntaxiques au sein du code source (p.ex. utilisation de caractères spéciaux inappropriés, balises mal fermées, etc.). Ces types d'erreurs ont été rencontrés aussi bien dans les couples de pages légitimes que dans les couples de pages légitimes-contrefaites. Dans ce dernier cas, les couples de pages posant problèmes ont été éliminés et remplacés par d'autres (car ils nécessitaient trop de corrections manuelles). Dans le cas des couples de pages légitimes, nous avons été amenés à corriger manuellement 3 couples de pages qui comportaient des erreurs (p.ex. des apostrophes remplacées par des guillemets, des "http" manquants en amont d'une URL, des balises mal orthographiées, etc.). A noter que les corrections manuelles apportées ne faussent aucunement les résultats de comparaison obtenus, puisque nous avons rencontré (et corrigé) exactement les mêmes erreurs dans les deux pages : PageDef et PageRef.

Précisons également que ces erreurs ne posent aucun problème aux navigateurs webs actuels qui savent s'adapter et délivrer un affichage correct de la page. A contrario, notre programme développé pour l'analyse des balises n'est actuellement pas capable de faire cette correction automatique. Par conséquent, à chaque erreur de syntaxe dans le nom d'une balise ou à chaque oubli de balise de fermeture (ces erreurs sont d'autant plus fréquentes dans les pages contrefaites), le programme de création des sous-fichiers échoue et nous ne pouvons effectuer le calcul de score global. Notre implémentation se devra donc d'être perfectionnée afin de ne pas être mise en échec par un attaquant qui trufferait volontairement sa page d'erreurs syntaxiques, en vue de mettre en défaut notre programme d'analyse.

6.1.5.6.3 Phase d'établissement de la connexion sécurisée : A ce stade, nous conservons quelques cas d'échec de récupération des pages webs PageDef et/ou PageRef associés à des échecs d'établissement de la connexion sécurisée SSL/TLS (cf. section 6.1.5.4). Nous n'avons toutefois pas eu le temps de revenir à notre implémentation pour corriger ces erreurs.

6.1.5.7 Synthèse HTML

6.1.5.7.1 Résultats obtenus : Les résultats sur l'analyse et la comparaison de pages webs obtenus dans cette seconde proposition confirment et améliorent les résultats de la proposition précédente.

En outre, ils attestent que l'approche par mots est plus appropriée que l'approche par caractères pour l'analyse des codes sources des pages webs.

Les nouvelles techniques de comparaisons développées ici, ainsi que les améliorations apportées par l'élimination de l'en-tête HTTP (qui introduisait une part variable sur les pages légitimes), permettent d'améliorer les résultats de comparaisons. Elles permettent également d'aboutir à une méthode de comparaison basée sur les analyses de balises par occurrence et par localisation, associées à l'analyse du code complet. Elles révèlent également un intérêt pour une analyse spécifique des *Liens* inclus dans les pages webs, mais celle-ci s'avère difficilement exploitable sur pages légitimes. Enfin, elles permettent d'aboutir à un seuil de décision de 90%.

Il est important de souligner que l'ensemble de ces résultats doivent être tempérés par la quantité de pages testées ainsi que les faux-positifs et/ou faux-négatifs résiduels. En effet, bien que les résultats obtenus lors des deux approches soient convergents, ils nécessitent d'être confirmés sur davantage d'URLs. De plus, la méthode de calcul finale retenue délivre quelques résultats erronés. Elle nécessite donc des analyses plus poussées afin d'être améliorée.

Un des verrous majeurs à l'issue de la première proposition concernait un taux d'échec trop élevé pour la récupération de PageRef. Ce problème a été résolu dans la seconde proposition, ce qui confirme les causes probables d'échec supposées précédemment. Notons toutefois que quelques erreurs subsistent concernant la phase d'établissement de la connexion SSL/TLS.

Il est également important de souligner que la nouvelle technique de récupération des pages webs développée ici, permet désormais d'identifier et de distinguer clairement les adresses IPdef et IPref utilisées.

6.1.5.7.2 Seuil de détection des modifications : De par les résultats obtenus précédemment ainsi que les analyses portées sur les cas de faux-positifs et faux-négatifs résiduels, nous nous sommes intéressés à déterminer la part minimale de page web devant être modifiée pour aboutir à une détection

Tableau 6.27 – Estimation des seuils de détection des modifications apportées au code source, d'après l'exemple de la page web Paypal

	Seuil de détection[1] des modifications, exprimé[2] en MOTS	en CARACTÈRES	Score méthode finale	Scores intermédiaires Balises occurrence	Balises localisation	Code complet	Equivalence script RBC[3]
Ajout - début code	+2.95 %	+1.52%	89.41%	99.09%	72.15%	97%	0.75
Ajout - fin code	+17,49%	+9.61%	89.99%	96.15%	91.81%	82%	4
Modification	29.90%	1.28%	89.67%	100%	100%	69%	N/A
Suppression - début code	−4.78%	−2.36%	89.70%	100%	74.09%	95%	0.5
Suppression - fin code	−47.70%	−11.86%	89.37%	92.22%	87.88%	88%	2.5

[1] pour obtenir un taux de similitude final < 90% avec la méthode retenue
[2] exprimé en % par rapport à la taille du code légitime
[3] exprimé en quantité de scripts de 20 lignes, cf. figure 6.10

par la méthode de comparaison finale retenue (c.-à-d. le score obtenu doit être inférieur au seuil de décision de 90%).

Ainsi, nous avons pris en exemple le code source de la page légitime Paypal `https://www.paypal.com` constitué de[1] : 22485 caractères, 109 lignes de codes et 983 mots. A savoir que la notion de "mots" indiquée ici correspond au découpage effectué par notre approche par mots. A ne pas confondre avec la notion de mots telle qu'interprétée par l'esprit humain (cf. section 5.2.2.4.2).

A partir de la page source légitime récupérée, nous avons créé 5 pages contrefaites, incluant l'un des 5 changements suivants : *modifications* en cours de code, *ajout* de code (en *début* ou *fin de page*), *suppression* de code (en *début* ou *fin de page*).

Concernant les *ajouts* et les *suppressions* de code, nous nous sommes placés dans 2 cas : le plus favorable (changements en début de page) et le plus défavorable (changements en fin de page). En effet, des ajouts et/ou suppressions réalisés en début de code induiront automatiquement des modifications majeures sur les localisations des balises. A contrario, des ajouts et/ou suppressions effectués en fin de code auront un impact moindre sur les balises (impact variable selon le type de modifications apportées), laissant ainsi tout le poids de la détection à l'analyse du code complet.

L'*ajout* a consisté à incorporer un script utilisé à des fins malveillantes, autant de fois que nécessaire, jusqu'à aboutir à une détection. Pour ce faire, nous avons utilisé un script à notre disposition : celui rencontré dans la page contrefaite RBC (cf. figure 6.10).

Les *modifications* ont été réalisées grâce au rajout d'un caractère par mot.

Les changements apportés en *début de page* ont été effectués dès les premières lignes de code, juste après la déclaration de la DTD.

Enfin, les changements apportés en *fin de page* ont été réalisés en partance des dernières lignes de code.

En conséquence, le tableau 6.27 délivre les seuils de détection résultant de ces 5 comparaisons de pages légitimes-contrefaites. Pour l'*ajout*, nous aboutissons à une détection dès 1.52 à 9.61% de caractères (ou 2.95 à 17.49% de mots) insérés, fonction de la zone où sont réalisés les changements. Pour les *modifications*, la détection se fait dès 1.28% de caractères (ou 29.90% de mots) modifiés. Enfin, la *suppression* est détectée dès 2.36 à 11.86% de caractères (ou 4.78 à 47.70% de mots) ôtés, selon la zone impactée.

A noter que les estimations indiquées ici sont toutes relatives. En effet, dans le code ajouté/modifié/supprimé, tout est fonction de la proportion de balises vs. contenu. Plus la part de balises sera importante, plus le seuil de détection sera abaissé, et réciproquement. Enfin, de véritables pages contrefaites utilisent rarement un seul type d'altération de contenu. Elles mixent généralement ajouts et/ou modification et/ou suppression. Les chiffres indiqués ici sont donc à interpréter comme des valeurs extrêmes de détection.

1. en date du 28 Juillet 2011.

Tableau 6.28 – Temps de traitement moyens relevés sur 11 localisations, portant sur 328 URLs de login légitimes

	Temps de traitement moyen par URL (min ≤ moyenne ≤ max)	Écart-type[1]
Défaut	2 sec. ≤ **4 sec.** ≤ 7 sec.	2 sec.
OpenDNS	1 sec. ≤ **3 sec.** ≤ 6 sec.	2 sec.
GoogleDNS	2 sec. ≤ **3 sec.** ≤ 5 sec.	1 sec.
DNSAdvantage	1 sec. ≤ **5 sec.** ≤ 12 sec.	4 sec.
Moyenne globale	***3.5 sec.***	

[1] entre localisations

6.1.6 Temps de traitement

La dernière partie de notre analyse a consisté à estimer le temps de traitement associé à notre seconde approche, c.-à-d. le temps nécessaire pour la récupération des pages et les calculs de scores (utilisant les approches par caractères et par mots). Le tableau 6.28 donne une estimation moyenne des temps de traitement relevés sur 11 localisations, pour la récupération et les calculs de score des pages légitimes.

On peut constater que le temps de traitement moyen relevé par page est de 3.5 sec. en moyenne, toutes localisations confondues. Ceci semble présager d'un temps de traitement "raisonnable" en environnement utilisateur.

Néanmoins des études plus poussées devront être menées sur le temps de traitement associé à la méthode finale retenue. A noter toutefois que les zones de traitement les plus consommatrices de temps sont celles qui concernent la récupération des pages.

6.2 Limitations

Dans cette section, nous nous intéressons aux limitations et verrous techniques demeurant à l'issue de notre étude. En effet, bien que les résultats obtenus dans les sections 5.3 et 6.1 soient globalement favorables, de nombreuses questions et/ou problématiques demeurent en suspens.

6.2.1 Vérification de l'adresse IP du domaine visité

Filtrage des requêtes DNS : Lors des tests réalisés sur pages légitimes, nous avons rencontré des problèmes liés à la vérification de l'adresse IP sur 3 localisations (c.-à-d. Japon et France(Bretagne) dans la première approche, et Australie dans la seconde approche). Les données résultantes, inexploitables, n'ont donc pas été incluses dans les résultats de tests énoncés précédemment. Sur ces 3 localisations, nous avons été confrontés à un filtrage des requêtes DNS. En effet, seules étaient autorisées les requêtes DNS à destination du serveur DNS par défaut (tel que préconisé dans certaines études [SRM07]). A noter que 2 des 3 localisations concernées ont effectué les tests depuis un réseau d'entreprise, tandis que la troisième a effectué les tests depuis un réseau public (c.-à-d. un hôtel). La mise en œuvre de filtrage des requêtes DNS peut donc s'avérer bloquante pour l'utilisation de notre proposition. Néanmoins, à ce jour nous n'avons pas été confrontés à ce problème dans des tests effectués depuis des réseaux personnels.

Établissement de la connexion sécurisée : Tel qu'énoncé dans nos résultats de tests, nous avons observé des erreurs résiduelles lors des échanges SSL, échanges indispensables à la récupération des pages de login. Bien que nous ayons implémenté une solution visant à accepter tous les certificats proposés par les serveurs webs interrogés, quelques erreurs SSL demeurent. Néanmoins, parce que l'implémentation est fortement dépendante du langage utilisé et que les erreurs constatées sont parfois changeantes, nous n'avons pas encore exploré plus avant cette problématique. Elle devra cependant être étudiée avec attention en cas d'implémentation réelle.

6.2.2 Analyse et comparaison du code source des pages webs

Identification des pages de login : Un des postulats de notre proposition est de s'adresser aux pages de login exclusivement. En effet, les résultats obtenus sur pages webs au sens large sont si peu satisfaisants que nous avons choisi de nous concentrer sur les pages de login, principales cibles des attaques de phishing/pharming. Ce postulat induit automatiquement une nouvelle problématique, à savoir : lors de la navigation web de l'utilisateur, comment savoir que l'analyse doit être effectuée. Autrement dit, comment savoir que l'utilisateur consulte une page de login. Une réponse simple semblerait être : dès lors que la connexion s'effectue en mode sécurisé (c.-à-d. en HTTPS), l'analyse doit être effectuée. Néanmoins, certaines pages de login s'affichent en HTTP dans le navigateur client (p.ex. Facebook), même s'il y a bien utilisation d'une connexion sécurisée (de manière totalement transparente pour l'utilisateur) lors de l'envoi des login et mot de passe. Ces pages seraient donc exclues, à tort, de nos analyses. Une autre solution pourrait consister à faire rechercher les zones de login (p.ex. via la recherche de balises de type <form>, <input>) à notre moteur de détection avant son exécution. Néanmoins, là aussi certains sites seront exclus de la comparaison. En effet, si la zone de login apparaît dans une nouvelle fenêtre ouverte spécifiquement à cet effet, le code source de la page web ne contiendra qu'un lien hypertexte servant à la redirection.

Redirection de contenu ou de site web : Le point précédent introduit une des limitations de notre approche. En effet, notre comparaison de pages webs s'intéresse exclusivement au code source des pages principales récupérées. Autrement dit, si la page principale est décomposée en sous-fenêtres, nous ne comparons que les liens hypertexte conduisant à ces sous-fenêtres, sans analyser leurs contenus réels. Ceci peut être considéré comme une vulnérabilité de notre approche.

Lors d'une éventuelle implémentation de notre proposition, une précaution particulière devra être prise concernant le choix du moment où s'exécute notre moteur de détection. En effet, tel qu'évoqué dans le Chapitre 3, certaines URLs de login procèdent à une redirection automatique. Par conséquent, si notre moteur de détection est lancé trop tôt, la comparaison de pages ne s'exécutera pas sur la page réellement visitée par l'utilisateur.

Authentification du site web visité : Il est important de souligner que notre approche ne vise en aucun cas à assurer l'authentification du site web visité. Elle ne se substitue donc aucunement à l'utilisation des techniques d'authentification existantes, ni à la nécessité de vigilance de l'utilisateur sur la validité du certificat utilisé par le site de login visité.

Il est donc tout à fait possible que 2 pages visitées amènent à des scores identiques ou très similaires - particulièrement avec l'analyse des balises ou l'approche par caractères -, alors que celles-ci sont totalement décorrélées. Ce point pourrait donc conduire à une nouvelle vulnérabilité de notre proposition, à savoir : le taux de similitude des pages comparées se révèle élevé, alors que les pages comparées n'ont aucun rapport. Néanmoins ceci peut être tempéré par le fait que le but premier des attaquants est de leurrer un maximum d'utilisateurs, avec des imitations visuelles (quasi-)parfaites des pages légitimes usurpées. On peut donc estimer que l'utilisateur sera à même d'identifier un affichage de page qui ne correspondrait pas à ses attentes.

Pages d'erreur : Tel que nous l'avons détaillé en section 6.1.5.5, les pages d'erreur improprement codées au niveau de l'en-tête HTTP conduiront automatiquement à des taux de similitude des pages comparées très faibles. Si cela concerne PageDef, il ne peut s'agir d'un réel problème puisque l'utilisateur sera informé de l'indisponibilité du site via la page affichée dans son navigateur. Toutefois, si cela concerne PageRef, notre moteur de détection délivrera une décision erronée.

Pages de login avec contenu dynamique : Une page de login présentant une forte proportion de contenu dynamique pose problème. En effet, des liens hypertexte renouvelés trop fréquemment - ou à différents moments selon la localisation -, peuvent conduire à des faux-positifs. La page du site `twitter.com` en est un exemple flagrant (pour plus de détails, cf. section 6.1.5.5). Par conséquent, plus les pages de login incluront du contenu dynamique, moins notre moteur de détection sera efficace. A ce jour, on peut cependant remarquer que seul le cas de l'URL `twitter.com` a réellement présenté ce problème parmi 328 URLs testées. A contrario, on peut noter que la page `login.yahoo.com` qui contient

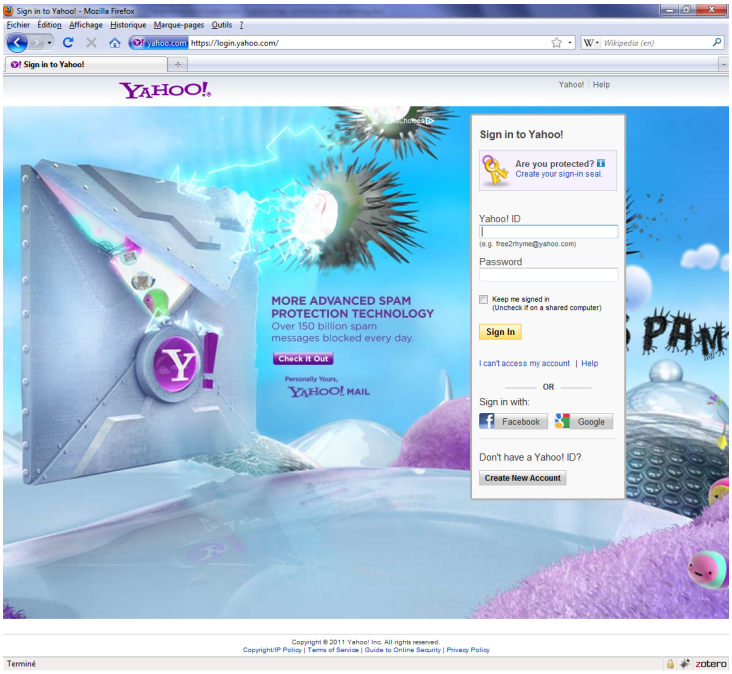


FIGURE 6.11 – Page `login.yahoo.com` incluant du contenu dynamique

elle-aussi du contenu dynamique (via une image de fond, cf. figure 6.11) retourne des taux de similitude très bons, de l'ordre de 99%.

Localisation du DNSref vs. l'utilisateur : Enfin, une autre limitation associée au calcul de score des pages webs peut résider dans les divergences de localisations qui hébergent PageDef et PageRef. De par l'interrogation de 2 serveurs DNS différents (DNSdef et DNSref), nous ne pouvons garantir d'aboutir à des serveurs webs géographiquement proches pour la récupération des 2 pages webs.

Parce que le langage de la page affichée dans le navigateur est automatiquement adapté selon les préférences de l'utilisateur, il est nécessaire que la page de référence le soit également. Nous avons vu que dans nos tests un site géographique a posé problème (cf. section 6.1.5.5, cas de l'URL `www.myspace.com` sur la localisation de Mexico). Toutefois, dès lors que notre approche sera intégrée dans le navigateur client, les 2 pages demandées (PageDef et PageRef) devraient toutes deux bénéficier des mêmes indications de préférence.

6.2.3 Intégration de l'approche dans le navigateur client

L'intégration de notre proposition dans le navigateur client de l'Internaute n'a pas encore été réalisée à ce jour. Pour rappel, notre solution vise à s'intégrer sous la forme :

- d'un indicateur visuel très simple et binaire indiquant le niveau de confiance envers la page web visitée,
- et d'une notification active effectuée via l'affichage d'un message d'alerte de type pop-up, dès lors que le site visité semble suspicieux.

Le choix d'un indicateur visuel très simple et d'un message de notification actif sont basés sur les conclusions d'études précédentes, qui précisent les types d'alertes les plus efficaces auprès des utilisateurs (cf. explications détaillées dans le Chapitre 2).

Un aperçu schématique de l'intégration visuelle envisagée est visible en figure 6.12. En complément, un menu de configuration du serveur DNS de référence devra être ajouté, afin de proposer à l'Internaute les choix de configurations DNSref détaillés en section 6.1.1.

Cette intégration visuelle dans le navigateur client amène à plusieurs nouvelles vulnérabilités et/ou limitations, à savoir :

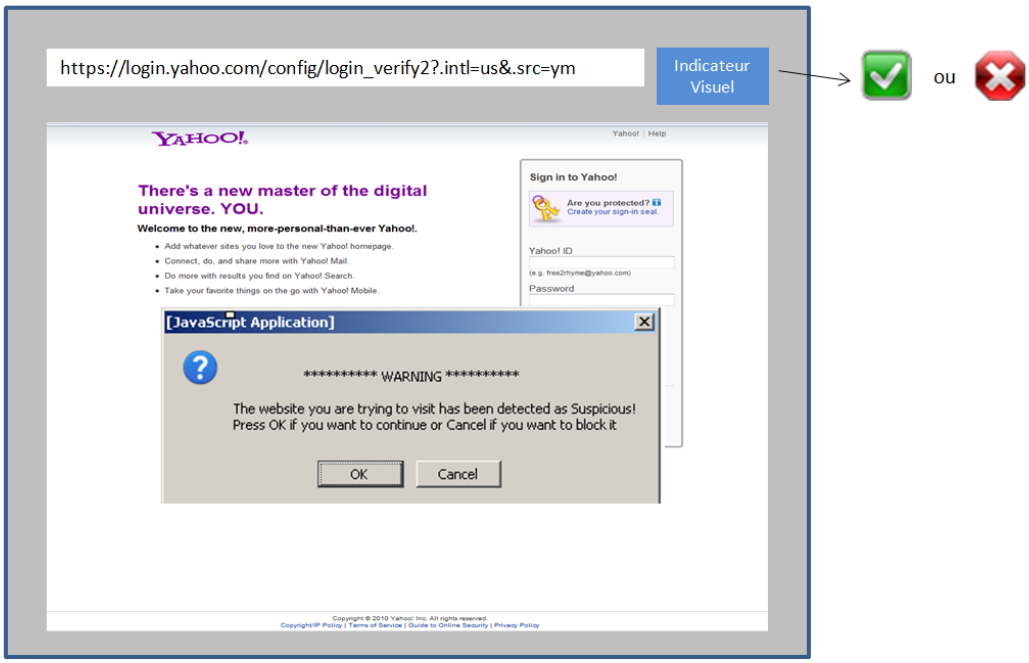


FIGURE 6.12 – Schéma d'intégration visuelle envisagée pour notre proposition de détection du pharming

- notre proposition se voit exposée aux vulnérabilités intrinsèques du navigateur et, plus globalement, à celles de la machine de l'Internaute,
- la corruption éventuelle de l'indicateur visuel,
- et la corruption éventuelle de la configuration du serveur DNS de référence.

Nous devrons donc, à minima, sécuriser le stockage de l'information de configuration du serveur DNS de référence sur la machine client. Une piste éventuelle pourrait être de sauvegarder cette information de manière chiffrée et signée par l'utilisateur par exemple. Ce point devra faire l'objet d'une analyse plus poussée.

A noter que nous devrons également apporter une attention toute particulière à d'éventuels problèmes de réinitialisation de cache DNS au sein du navigateur client et/ou du système d'exploitation. Ceci afin de s'assurer que nos requêtes sont belles et bien générées.

Enfin, nous n'avons pas pu explorer plus avant l'intégration de nos développements menés en Java (langage nécessaire pour la génération des requêtes DNS) au sein d'une barre d'outils en Javascript (langage recommandé pour un meilleur rendu visuel, mais ne pouvant satisfaire aux besoins exigés par les requêtes DNS). Néanmoins, d'après les problèmes rencontrés à ce propos dans le Chapitre 3, il est fort probable que la tâche s'annonce difficile, voire qu'elle nous conduise à des changements de langage.

6.3 Synthèse du chapitre

Ce chapitre ainsi que le précédent ont présenté deux solutions visant à proposer une méthode de détection des attaques de pharming côté client. Ces solutions reposent toutes deux sur une combinaison de la vérification de l'adresse IP du domaine visité, associée à une analyse de contenu des pages webs de login.

Les résultats obtenus dans la seconde proposition exposée dans ce chapitre ont confirmé la staticité des adresses IP associées aux pages de login entrevue dans la première approche (cf. Chapitre 5). Ils ont également indiqué une convergence des résultats en provenance de 3 serveurs DNS de référence différents, permettant ainsi d'aboutir à une meilleure robustesse de la solution. Concernant l'analyse du

code source des pages webs, l'approche par mots s'est révélée être la plus intéressante. Combinée à des techniques d'analyses des balises, elle a permis d'aboutir à la définition d'un seuil de décision. Enfin, la nouvelle technique de récupération de la page de référence a permis de revenir à des taux d'erreurs raisonnables.

A l'issue des résultats convergents et prometteurs obtenus dans nos deux propositions - certes sur un nombre restreint d'URLs -, une implémentation et intégration dans le navigateur client peut être envisagée (moyennant une étude préalable sur le(s) langage(s) les plus approprié(s) à utiliser). Celle-ci peut se présenter, par exemple, sous la forme d'une intégration aux techniques déjà utilisées pour la détection du phishing, à savoir : les barres d'outils intégrées dans le navigateur du client.

Offrir une telle fonctionnalité de détection des attaques de pharming perpétrées sur le réseau client peut s'avérer souhaitable pour améliorer la sécurité globale de la navigation web de l'Internaute, en complément des techniques d'authentification et de protection déjà disponibles et/ou en cours de déploiement côté réseau Internet (p.ex. HTTPS, DNSSEC).

Néanmoins, de nombreuses questions et limitations associées à nos deux propositions restent à résoudre. De plus, la technique d'analyse de comparaison des pages webs se doit d'être perfectionnée. Enfin, des tests sur davantage d'URLs devront être menés.

7 Conclusion et perspectives de recherche

De toutes les vulnérabilités auxquelles s'expose l'Internaute qui souhaite réaliser des transactions à distance, il y en a certainement une qui retient son attention plus que les autres : celle qui s'en prend directement à son portefeuille. Plus connue sous le terme de phishing, cette attaque - majoritairement véhiculée par des campagnes de spam [Gas09] - vise à voler des informations confidentielles (p.ex. identifiant, mot de passe, numéro de carte bancaire, etc.) aux utilisateurs en usurpant l'identité des sites marchands et/ou bancaires. Face à la prolifération de ces attaques, les acteurs d'Internet (p.ex. FAI, banques) se mobilisent et diffusent très régulièrement des campagnes d'informations à destination des utilisateurs. Un des moyens de protection les plus courants et les plus faciles d'accès proposés aux Internautes sont les barres d'outils anti-phishing. Qu'elles soient ou non natives dans les navigateurs webs, ces barres reposent depuis toujours sur l'une et ou l'autre des méthodes de détection suivantes : listes noires et tests heuristiques. De nombreux articles existent à ce propos et bien que les avis divergent, il apparaît difficile de trancher définitivement en faveur de l'une ou l'autre de ces méthodes. Les listes noires semblent en effet plus exactes dans leur détection, mais non moins incomplètes, insuffisamment réactives ou encore, trop assujetties à d'éventuelles compromissions. A contrario, les tests heuristiques semblent plus autonomes car ils ne nécessitent pas de mises à jour aussi fréquentes. Ils s'en retrouvent donc moins exposés aux attaques mais ils n'en sont pas pour autant mieux adaptés. En effet, leur décision peut s'avérer moins précise (c.-à-d. trop de faux-positifs et/ou de faux-négatifs) et leur pérennité est fragile (c.-à-d. les attaquants peuvent réagir et adapter leur contrefaçons pour contrer les tests effectués). En Première Partie de ce mémoire, notre contribution [GGL11] s'est donc attachée à identifier la pertinence et la pérennité de 20 tests heuristiques utilisés à la détection des sites de phishing. Nous en avons conclu qu'analyser le code source HTML et vérifier que la connexion s'effectue de manière sécurisée sont des caractéristiques hautement révélatrices d'un site web contrefait (pour 96.52% des URLs testées concernant le code source HTML, et 78.26% des URLs testées concernant la connexion sécurisée). Ces deux caractéristiques sont ensuite secondées par la recherche de mots clés au sein de l'URL visitée (à hauteur de 71.30% des URLs testées). A contrario, la simplicité des URLs légitimes (à hauteur de 96.65% des URLs testées) et l'étude du code source HTML de la page (à hauteur de 63.04% ou 66.09% des URLs testées) sont les critères d'analyse les plus pertinents pour identifier un site web original. En complément, cette étude fait apparaître que les critères les plus pérennes sont ceux qui concernent la connexion sécurisée et l'étude du code source HTML. Ils sont en effet plus difficiles à usurper.

Suite à cette étude, dans la Seconde Partie de ce mémoire, nous avons voulu étudier la portabilité de l'étude du code source HTML aux attaques de pharming, plus difficiles à détecter côté client mais pourtant non moins redoutables. Ces attaques s'appuient en effet sur la mise en ligne d'un site web contrefait, associée à une corruption des informations DNS effectuée, en amont, dans la chaîne de liaison client - domaine visité (c.-à-d. réseau où le site web visité est hébergé).

Même si la sécurisation du protocole DNS a longtemps souffert d'un défaut d'attention de la part des acteurs de la communauté réseau (registres, FAI, entreprises, etc.), l'ampleur/la portée des faiblesses du protocole a été exposée à tous durant l'été 2008 (cf. section 4.1.3.1). Depuis, les acteurs d'Internet ont donc vraiment entamé/accéléré le déploiement des extensions de sécurité DNSSEC. On peut donc imaginer que dans les années à venir, le protocole DNS va s'en retrouver de plus en plus renforcé. Néanmoins, au vu de l'impressionnante quantité de serveurs DNS existants, ce renforcement risque de prendre encore bon nombre d'années. En attendant, le réseau client demeure insuffisamment protégé contre d'éventuelles corruptions DNS, et les zones de vulnérabilités associées auxquelles il est exposé resteront de toute façon encore trop nombreuses (cf. section 4.1.3.2).

Offrir une fonctionnalité de détection des attaques de pharming perpétrées sur le réseau client peut donc s'avérer souhaitable pour améliorer la sécurité globale de la navigation web de l'Internaute, en complément des techniques d'authentification et de protection déjà disponibles et/ou en cours de déploiement côté réseau Internet (p.ex. HTTPS, DNSSEC).

Nous avons donc essayé de répondre à ce problème en développant deux approches visant à détecter les attaques de pharming réalisées côté client. Le cœur de notre proposition repose sur l'étude du code source HTML du site web visité, combinée à des requêtes DNS.

La première approche proposée [GGL11b] décide de la légitimité de la page de login visitée basée sur une vérification de l'adresse IP du domaine, combinée à une analyse du contenu de la page web. Celles-ci s'effectuent à partir d'éléments de référence (c.-à-d. une adresse IP et une page web) fournis par un serveur DNS différent de celui auquel le client est connecté. En particulier, la page web de référence est récupérée en remplaçant le FQDN de l'URL visitée par l'adresse IP de référence. L'implémentation réalisée nous a permis de vérifier la validité de cette approche, mais elle nous a également démontré ses faiblesses : un taux d'erreur de récupération des pages de référence trop important (de l'ordre de 22%), et une méthode de comparaison des pages webs insuffisamment précise (avec une fenêtre de recouvrement de l'ordre de 11%).

Au travers d'une seconde approche [GL11], nous avons donc amélioré la solution proposée, en modifiant le processus de récupération de la page web de référence. Celle-ci est désormais obtenue en effectuant une redirection de la requête HTTP vers une adresse IP déterminée (c.-à-d. l'adresse IP de référence). Ceci nous permet d'aboutir à des taux d'erreur de récupération des pages nettement plus raisonnables (de l'ordre de 6%). En complément, nous avons également amélioré la méthode de comparaison des pages webs afin d'aboutir à un seuil de décision établi à 90%, et minimiser les faux-positifs ($< 0.61\%$) et faux-négatifs (5.17%) obtenus sur les échantillons testés.

Perspectives et axes de recherche futurs

Côté phishing

Pour réaliser notre étude des tests heuristiques, il s'est avéré difficile d'appréhender avec précision les tests heuristiques couramment utilisés par les barres d'outils anti-phishing, et encore plus d'en connaître les éventuels seuils de décision. Ceci peut aisément s'expliquer par la recherche d'une pérennité/efficacité de ces solutions. Nous nous sommes donc basés sur notre analyse des sites légitimes/contrefaits ainsi que sur différents travaux existants pour définir nos propres tests heuristiques. A cette occasion, nous avons au départ volontairement restreint la quantité d'heuristiques étudiés et leur portée.

Néanmoins, au vu des résultats obtenus dans la Première Partie de ce mémoire sur l'efficacité et la prépondérance des heuristiques, nous envisageons plusieurs pistes d'amélioration. Il est en effet évident que - comme pour tout mécanisme de détection - il est indispensable de faire évoluer en permanence les tests heuristiques utilisés :

- Nous pensons notamment à un renouvellement régulier des triplets/mot-clés dits de phishing, une augmentation du nombre de mots-clés (p.ex. grâce à des classements des sites les plus contrefaits), un rafraîchissement régulier des TLDs réputés pour l'hébergement des sites de phishing, etc.

- Pour améliorer les heuristiques qui recherchent un numéro de port dans l'URL ou la présence des caractères (//), il serait intéressant de les compléter avec la recherche des caractères d'encodage associés (p.ex. en ASCII, `:8080` peut être encodé avec `%3A%38%30%38%30`, ou `://` qui suit généralement un `http` peut être encodé avec `%3A%2F%2F`).

- Il faudrait également réétudier le critère associé à l'examen de la cohérence entre TLD et pays hébergeur de site. En effet, aujourd'hui, tous les sites qui utilisent des TLDs génériques échappent à toute forme de pénalisation, ce qui est source de contournement du test. Pour ces TLDs génériques, il faudrait peut-être alors simplement se baser sur l'indication de pays retournée par *World IP* pour pénaliser ou non le site visité.

- Concernant l'étude des balises, il faudrait commencer par enrichir la base de caractères de substi-

tution utilisée. Cette base, qui permet de remplacer un caractère accentué par son pendant sans accent, est fondamentale pour vérifier la cohérence des balises contenant du texte vs. le FQDN visité. Ceci permettrait notamment de limiter les faux-positifs.

- Pour les balises également, il faut impérativement étendre l'analyse à l'ensemble des balises de liens contenues dans la page (et non uniquement se focaliser sur la première balise, tel qu'effectué aujourd'hui). Sinon, la technique de contournement est triviale : il suffit de porter mention du FQDN visité dans toutes les premières balises de chaque type.

- Pour améliorer la recherche de zones de login, il faudrait également être capable d'exécuter l'analyse sur les fenêtres additionnelles qui s'ouvrent parfois pour la saisie des données d'authentification.

- Pour augmenter le nombre de critères aidant à la décision de légitimité, certaines approches visent à créer des bases de signatures générées à partir de l'aspect visuel d'une page [MKK08]. En effet, le principe est de comparer la signature de la page visitée à une base de signatures légitimes des sites les plus contrefaits. Nous pourrions alors imaginer appliquer systématiquement ce traitement aux pages affichées via le protocole `http://`. Néanmoins, de par l'augmentation de contenus dynamiques au sein des pages webs, il faut s'assurer que ce type d'approche reste valable.

- Dans le même esprit d'augmenter le nombre de critères de décision, nous pensons qu'il serait intéressant d'intégrer un heuristique qui évalue l'âge du site visité, à l'image de ce qui est proposé par d'autres barres d'outils. En effet, de par la très courte durée de vie des sites de phishing (cf. section 2.3), ceci pourrait s'avérer fort utile pour distinguer les sites légitimes des sites contrefaits.

- En complément, nous pensons qu'il serait intéressant d'appliquer notre phase d'identification des heuristiques déterminants aux heuristiques eux-mêmes (et non seulement aux catégories auxquelles ils sont rattachés). Ceci afin d'affiner la précision des résultats obtenus.

Pour terminer, notre discussion sur la pérennité éventuelle des heuristiques nous amène également à plusieurs recommandations et/ou sujets de réflexion :

- Le développement d'une barre d'outils anti-phishing se doit impérativement d'être couplé avec l'intégration d'un moyen de sécurisation du moteur de détection et/ou des blacklist/whitelist utilisées par ce dernier. En effet, par exemple il ne peut être question d'envoyer l'URL à contrôler à un serveur tiers, sans aucune sécurisation de cet envoi, tel qu'effectué par *Netcraft* (cf. section 3.4). Ce point devra donc faire l'objet d'une étude plus spécifique.

- Nous pensons également que d'autres phases d'étalonnage des heuristiques pourraient être menées (particulièrement à partir des résultats obtenus lors de l'identification des heuristiques déterminants) afin d'améliorer la robustesse/longévité de la solution. Nous pourrions alors tenter de définir, par exemple : 1/ Différentes échelles de seuils de décision. Ainsi, nous pourrions introduire plusieurs schémas de décision qui seraient aléatoirement utilisés par le moteur de détection à chaque vérification d'une URL, ou 2/ Une pondération aléatoire applicable au calcul de score global. Les 20 heuristiques ne seraient alors plus considérés avec le même poids, poids qui deviendrait variable d'un instant à un autre pour chacun d'entre eux.

- Enfin, puisqu'il apparaît que l'analyse du code source HTML est primordiale, aussi bien pour identifier un site légitime qu'un site contrefait, nous pensons qu'il serait fortement souhaitable que les développeurs de sites légitimes portent une plus grande attention au remplissage des balises, et plus particulièrement à celles qui concernent les liens, afin d'inclure une mention au domaine/FQDN. En effet, ceci permettrait de limiter les cas de faux-positifs. De plus, cela compliquerait la tâche des attaquants qui basent aujourd'hui essentiellement la conception de leurs sites contrefaits, sur la réutilisation d'une très large majorité de liens légitimes, afin de rendre leurs attaques plus transparentes.

Côté pharming

Lors de notre étude sur le pharming, nous avons au départ volontairement restreint notre champ d'analyse aux méthodes de comparaison de texte. Néanmoins, les améliorations apportées dans la seconde approche proposée tendent à indiquer qu'une analyse axée autour de la structure du document HTML est également à considérer. Comme expliqué dans ce document, les méthodes de comparaison de structure existantes sont difficilement applicables au HTML de par son implémentation trop permissive. Par ailleurs, il convient de constater que bon nombre de codes sources de pages webs visitées sont encore en HTML (et non XHTML). Nous pensons qu'il pourrait être alors intéressant d'adapter notre méthode de comparaison selon la nature du document récupéré, en nous basant sur les indications mentionnées dans la DTD (cf. section 2.2.2).

A partir des limitations de notre proposition détaillées dans les chapitres précédents, nous pensons que plusieurs pistes d'amélioration peuvent être étudiées :

- Au travers des réponses DNS reçues, nous constatons qu'il reste quelques échecs (c.-à-d. des requêtes DNS infructueuses). Aujourd'hui l'implémentation proposée n'effectue qu'une seule tentative de requête auprès du serveur DNS de référence. Celle-ci pourrait être éventuellement complétée par une deuxième tentative complémentaire.

- L'intégration dans le navigateur web, typiquement au sein d'une barre d'outils anti-phishing, doit être réalisée. Les développements jusqu'ici menés en Java montrent leurs limitations, ne serait-ce qu'en terme de réinitialisation du cache DNS. D'autres alternatives devront donc être étudiées. En complément, une étude supplémentaire doit être menée sur la sécurisation du moteur de détection, afin de limiter les éventuelles compromissions.

- Aujourd'hui, l'ensemble des opérations (ajout/modication/suppression de code) effectuées entre deux pages sont considérées avec la même importance. Il en va de même pour les changements de balises. Nous pensons qu'il serait intéressant que les ajouts et modifications aient un poids nettement plus important que les suppressions, à l'image des dangers induits. De même en ce qui concerne les balises, nous pourrions imaginer qu'une augmentation de balises de type <script> ait un poids important (vs. les autres balises), et que la distance entre des balises d'ouverture et de fermeture de script soit un critère important (c.-à-d. pour détecter d'éventuelles modifications de scripts malgré un nombre de balises inchangés).

- Une autre piste d'amélioration serait d'envisager une analyse de type Damerau-Levenshtein, où la transposition de mots serait également considérée, en complément des opérations d'ajout, de suppressions et de modifications. Celle-ci pourrait éventuellement aider à limiter les cas de faux-positifs pour les pages légitimes issues de différentes localisations (c.-à-d. récupérées depuis différents serveurs DNS qui eux-mêmes redirigent vers des serveurs webs géographiquement très éloignés) pour lesquelles nous avons souvent noté des changement d'ordonnancement de code.

- De par l'importante quantité de balises existantes en HTML, nous avons volontairement restreint notre champ d'analyse à une liste réduite. Nous pensons qu'il serait judicieux d'étoffer le nombre de balises analysées afin d'être plus exhaustif/représentatif de l'ensemble des modifications apportées.

- Le renouvellement trop fréquent des contenus dynamiques au sein des pages webs pose problème pour la comparaison de leurs contenus. Afin de détecter des corruptions de pages webs - sous réserve que PageDef et PageRef soient récupérées à partir de serveurs web différents -, nous pourrions envisager une amélioration de la méthode de comparaison via l'ajout d'une analyse des *Liens* (différente de celle envisagée dans les Chapitres précédents), en étroite relation avec le score délivré pour le code complet. Le(la) poids/pondération accordé(e) pourrait être proportionnel(le) au type de changement effectué. Basés sur une analyse préalable de la page légitime, nous pourrions par exemple envisager de déduire un profil type des liens contenus (p.ex. nombre de liens de type "moteur de recherche", balises où ils apparaissent, nombre de niveaux d'arborescence dans les liens, etc.), et l'utiliser en comparaison avec un profil calculé pour la page visitée. Via un

calcul de distance, nous pourrions alors ajouter un critère de décision supplémentaire qui viserait à impacter le score global, à la hausse (pour éventuellement compenser le mauvais score obtenu sur code complet) ou à la baisse.

– Nous pensons qu'il pourrait être également judicieux d'étendre notre sélection de serveurs DNS de référence, sous réserve de tests préalables qui garantissent une réactivité et un taux de réponse satisfaisants. Une étude menée par Ager et al. [AMSU10] met en garde contre le risque d'un temps de latence plus élevé pour les réponses DNS issues d'un serveur alternatif vs. celui proposé par le FAI.

– Une des pistes complémentaires les plus intéressantes serait probablement de rechercher l'ensemble des fenêtres qui sont appelées depuis le code source principal d'une page, afin d'en analyser également leurs contenus. Ceci permettrait de faire une analyse exhaustive du contenu des codes sources de la page. Néanmoins, au-delà de l'éventuelle difficulté de récupération de ces "sous-pages", l'impact sur le temps de traitement risque d'être considérable.

– Face au problème d'identification des pages de login - c.-à-d. il est impératif que le moteur de détection sache distinguer les pages de login pour lancer sa vérification -, le moyen le plus rapide et/ou logique de lui donner connaissance de cette information est de regarder si la page visitée utilise une connexion sécurisée HTTPS. Néanmoins, cette solution a ses limites. Nous avons en effet constaté que certaines pages de login s'affichent en HTTP (p.ex. Facebook), bien que l'envoi des identifiants de l'utilisateur soit fait en HTTPS. Notre vérification ne serait alors pas exécutée, à tort. Pour pallier ce défaut, en complément du point précédent, nous pensons qu'il faudrait alors au minimum inspecter la page web ainsi que toutes ses sous-fenêtres, à la recherche de zones de saisie de login (p.ex. via la recherche de balises de type <form>, <input>). Toutefois, là encore, l'impact sur le temps de traitement doit être étudié avec attention.

Par ailleurs, d'autres points plus insolubles sont à considérer : un filtrage des requêtes DNS qui ne seraient autorisées qu'à destination du serveur DNS par défaut est un problème. Nous sommes également dans l'incapacité de détecter des pages webs d'erreur improprement codées au niveau de l'en-tête HTTP, ce qui peut conduire à quelques cas de faux-positifs lors des indisponibilités temporaires de sites (p.ex lors d'une opération de maintenance). Une étude sur un plus grand nombre d'URLs devra donc être menée afin d'appréhender la portée de ces problèmes, et conforter nos résultats.

Il est également indispensable que lors d'une intégration dans le navigateur web, une attention toute particulière soit accordée au temps de traitement nécessaire à la prise de décision. En effet, il faut que celui-ci soit au plus bas, afin d'éviter une saisie des informations confidentielles de l'Internaute. Ce dernier point risque d'avoir un impact significatif sur les types/quantités de calculs appliqués.

Enfin, au delà de la détection des attaques de pharming proposée dans cette étude, nous aimerions explorer plus avant d'éventuels moyens de protection visant à empêcher les corruptions DNS effectuées sur le poste client, en amont.

Table des figures

Liste des tableaux

Table des algorithmes

Glossaire des Acronymes

ADN : Acide DésoxyriboNucléique
API : Application Programming Interface
APWG : Anti-Phishing Working Group
ASCII : American Standard Code for Information Interchange
BLOSUM : BLOcks SUbstitution Matrix
CAPTCHA : Completely Automated Public Turing test to tell Computers and Humans Apart
cc-TLD : country-code Top-Level Domain
CN : Common Name
CSS : Cascading Style Sheet
DHCP : Dynamic Host Configuration Protocol
DNS : Domain Name System
DNSKEY : Domain Name System KEY
DNSSEC : Domain Name System SECurity
DOM : Document Object Model
DS : Delegation Signer
DSA : Digital Signature Algorithm
DTD : Document Type Definition
FAI : Fournisseur d'Accès Internet
FNR : False Negative Rate
FPR : False Positive Rate
FQDN : Fully Qualified Domain Name
FTP : File Transfer Protocol
g-TLD : generic Top-Level Domain
GSLB : Global Server Load Balancing
HTML : HyperText Markup Language
HTTP : HyperText Transfer Protocol
HTTPS : HyperText Transfer Protocol Secure
ICANN : Internet Corporation for Assigned Names and Numbers
IP : Internet Protocol
ISBN : International Standard Book Number
KSK : Key Signing Key
LCS : Longest Common Subsequence
MAAWG : Message Anti-Abuse Working Group
MD5 : Message-Digest 5 algorithm
MMS : Multimedia Messaging Service

MX : Mail eXchange
NS : Name Server
NSEC : Next-SECure record
OSI : Open Systems Interconnection
OTP : One-Time Password
PAM : Point Accepted Mutation
PC : Personal Computer
pcap : packet capture
RDF : Resource Description Framework
RFC : Request For Comments
RR : Resource Record
RRset : Resource Record set
RRSIG : Resource Record SIGnature
RSA : Rivest Shamir Adleman
RSS : Rich Site Summary, *ou* Rdf Site Summary, *ou* Really Simple Syndication
S/MIME : Secure/Multipurpose Internet Mail Extensions
SHA : Secure Hash Algorithm
SMS : Short Message Service
SMTP : Simple Mail Transfer Protocol
SOP : Same Origin Policy
SQL : Structured Query Language
SSL : Secure Socket Layer
TCP : Transmission Control Protocol
TLD : Top-Level Domain
TLS : Transport Layer Security
TTL : Time To Live
UDP : User Datagram Protocol
URI : Uniform Resource Identifier
URL : Uniform Resource Locator
URN : Uniform Resource Name
VoIP : Voice over IP
W3C : World Wide Web Consortium
WOT : Web Of Trust
XML : eXtensible Markup Language
XSS : cross-Site Scripting
XUL : Xml-based User interface Language
ZSK : Zone Signing Key

Bibliographie

[AC75] Aho, Alfred V. et Margaret J. Corasik: *Efficient string matching : an aid to bibliographic search*. Communications of the ACM, 18(6) :333–340, juin 1975, ISSN 0001-0782.

[ACC+09] Agrawal, Sanjay, Kaushik Chakrabarti, Surajit Chaudhuri, Venkatesh Ganti, Arnd Christian Konig et Dong Xin: *Exploiting web search engines to search structured databases*. Dans *Proceedings of the 18th International Conference on World Wide Web (WWW)*, page 501–510, Madrid, Spain, 2009. ACM, ISBN 978-1-60558-487-4.

[AFNa] AFNIC: *Comprendre et gérer le système de nommage sur Internet*. `http://www.afnic.fr/ext/dns/index.html`.

[AFNb] AFNIC: *DNS Sécurisé*. `http://www.afnic.fr/noncvs/formations/dnssec_court/dnssec-afnic-v1.7.2.2.pdf`.

[AFN09] AFNIC: *DNS : types d'attaques et techniques de sécurisation*, juin 2009. `http://www.afnic.fr/data/divers/public/afnic-dossier-dns-attaques-securite-2009-06.pdf`.

[AFN10] AFNIC: *DNSSEC - les extensions de sécurité du DNS*, septembre 2010. `http://www.afnic.fr/data/divers/public/afnic-dossier-dnssec-2010-09.pdf`.

[AHDT10] Aburrous, Maher, M. A. Hossain, Keshav Dahal et Fadi Thabtah: *Predicting Phishing Websites Using Classification Mining Techniques with Experimental Case Studies*. Dans *Proccedings of the 7th International Conference on Information Technology : New Generations (ITNG)*, pages 176–181, Las Vegas (Nevada), USA, avril 2010. IEEE, ISBN 978-1-4244-6270-4.

[Al] Alexa Internet, Inc.: *Alexa Top 500 Global Sites*. http ://www.alexa.com/topsites. `http://www.alexa.com/topsites`.

[AMSU10] Ager, Bernhard, Wolfgang Mühlbauer, Georgios Smaragdakis et Steve Uhlig: *Comparing DNS resolvers in the wild*. Dans *Proceedings of the 10th annual Conference on Internet Measurement (IMC)*, Melbourne, Australia, novembre 2010. ACM, ISBN 978-1-4503-0483-2. `http://www.net.t-labs.tu-berlin.de/papers/AMSU-CDRW-10.pdf`.

[AN08] Abu-Nimeh, Saeed et Suku Nair: *Bypassing Security Toolbars and Phishing Filters via DNS Poisoning*. Dans *Proceedings of the 2008 Global Telecommunications Conference (GLOBECOM)*, pages 1–6, New Orleans (Louisiana), USA, décembre 2008. IEEE, ISBN 978-1-4244-2324-8.

[apw] *APWG - Anti-Phishing Working Group*. `http://www.apwg.org/`.

[APW10] APWG: *Phishing Activity Trends Report - 2nd Half 2010*. rapport technique, décembre 2010. `http://www.antiphishing.org/reports/apwg_report_h2_2010.pdf`.

[AS] Artiles, Javier et Satoshi Sekine: *Tagged and Cleaned Wikipedia (TC Wikipedia) and its Ngram*. `http://nlp.cs.nyu.edu/wikipedia-data/`.

[atUoCfI10] ICANN, NORC at the University of Chicago for: *Draft Report for the Study of the Accuracy of WHOIS Registrant Contact Information*. Rapport technique 6558, 6636, janvier 2010. `http://www.icann.org/en/compliance/reports/whois-accuracy-study-17jan10-en.pdf`.

[BFSB10] Barth, Adam, Adrienne Porter Felt, Prateek Saxena et Aaron Boodman: *Protecting Browsers from Extension Vulnerabilities*. Dans *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego (California), USA, mars 2010. ISOC.

[BHR00] Bergroth, L., H. Hakonen et T. Raita: *A Survey of Longest Common Subsequence Algorithms*. Dans *Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE)*, Coruna, Spain, 2000. IEEE Computer Society, ISBN 0-7695-0746-8.

[Bil05] Bille, Philip: *A Survey on Tree Edit Distance and Related Problems*. Theorical Computer Science, pages 217–239, juin 2005, ISSN 0304-3975.

[BKKB11] Bilge, Leyla, Engin Kirda, Christopher Kruegel et Marco Balduzzi: *EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis*. Dans *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, San Diego (California), USA, février 2011. ISOC.

[BM77] Boyer, Robert S. et J. Strother Moore: *A fast string searching algorithm*. Communications of the ACM, 20(10) :762–772, octobre 1977, ISSN 0001-0782.

[BM10] Bau, Jason et John C. Mitchell: *A Security Evaluation of DNSSEC with NSEC3*. Dans *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego (California), USA, mars 2010. ISOC.

[Bou] Boulle, Marc: *Software for Data Mining : France Telecom Scoring Tool*. `http://www.khiops.com/`.

[Bou08] Boullé, Marc: *Khiops : outil de préparation et modélisation des données pour la fouille des grandes bases de données*. Dans *8ème journées francophones Extraction et Gestion des Connaissances (EGC)*, page 229–230, Sophia Antipolis, France, 2008.

[BQX10] Bin, Sun, Wen Qiaoyan et Liang Xiaoying: *A DNS Based Anti-phishing Approach*. Dans *Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, tome 02, pages 262–265, Wuhan, China, avril 2010. IEEE Computer Society, ISBN 978-0-7695-4011-5.

[cap] *CAPTCHA*. `http://www.captcha.net/`.

[CDM10] Chen, Teh-Chung, Scott Dick et James Miller: *Detecting visually similar Web pages : Application to phishing detection*. ACM Transactions on Internet Technology (TOIT), 10(2) :5 :1–5 :38, mai 2010, ISSN 1533-5399.

[CG06] Chen, Juan et Chuanxiong Guo: *Online Detection and Prevention of Phishing Attacks*. Dans *Proceedings of the Conference on Communications and Networking in China (CHINACOM)*, Beijing, China, octobre 2006. ISBN 1-4244-0463-0.

[CHL08] Cao, Ye, Weili Han et Yueran Le: *Anti-phishing based on automated individual white-list*. Dans *Proceedings of the 4th ACM workshop on Digital Identity Management (DIM)*, pages 51–60, Alexandria (Virginia), USA, octobre 2008. ACM. `homepage.fudan.edu.cn/~wlhan/paper/dim08-han.pdf`.

[CL97] Charras, Christian et Thierry Lecroq: *ESMAJ - Exact String Matching Algorithms*, 1997. `http://www-igm.univ-mlv.fr/~lecroq/string/index.html`.

[CLT+04] Chou, Neil, Robert Ledesma, Yuka Teraguchi, Dan Boneh et John C. Mitchell: *Client-side defense against web-based identity theft*. Dans *Proceedings of 11th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego (California), USA, février 2004. ISOC, ISBN 1-891562-18-5, 1-891562-17-7. `http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Chou.pdf`.

[CT94] Cavnar, William B. et John M. Trenkle: *N-Gram-Based Text Categorization*. Dans *Proceedings of SDAIR*, 1994.

[Dam64] Damerau, Fred J.: *A technique for computer detection and correction of spelling errors*. Communications of the ACM, 7(3) :171–176, mars 1964, ISSN 0001-0782.

[DG09] Dhawan, Mohan et Vinod Ganapathy: *Analyzing Information Flow in JavaScript-Based Browser Extensions*. Dans *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC)*, pages 382–391, Honolulu (Hawaii), USA, décembre 2009. IEEE Computer Society, ISBN 978-0-7695-3919-5.

[DGE07] Dredze, Mark, Reuven Gevaryahu et Ari Elias-Bachrach: *Learning Fast Classifiers for Image Spam*. Dans *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, Moutain View (California), USA, août 2007.

[dns] *DNS Advantage*. http://www.dnsadvantage.com/.

[DPLL08] Dagon, David, Niels Provos, Christopher P. Lee et Wenke Lee: *Corrupted DNS Resolution Paths : The Rise of a Malicious Resolution Authority*. Dans *Proceedings of the 16th Annual Network & Distributed System Security Symposium (NDSS)*, tome Session 1 :Large Scale Systems, San Diego (California), USA, février 2008. The Internet Society (ISOC). http://www.citi.umich.edu/u/provos/papers/ndss08_dns.pdf.

[DT03] Dulucq, Serge et Hélène Touzet: *Analysis of tree edit distance algorithms*. Dans *Proceedings of the 14th annual symposium of combinatorial pattern matching (CPM)*, pages 83–95, Morelia, Mexico, juin 2003. Springer-Verlag.

[DT05] Dhamija, Rachna et J.D. Tygar: *The Battle Against Phishing : Dynamic Security Skins*. Dans *Proceedings of the 2005 Symposium On Usable Privacy and Security (SOUPS)*, tome 93, pages 77–88, Pittsburg (Pennsylvania), USA, juillet 2005. ACM, ISBN 1-59593-178-3.

[ECH08] Egelman, Serge, Lorrie Cranor et Jason Hong: *You've been warned : an empirical study of the effectiveness of web browser phishing warnings*. Dans *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1065–1074, Florence, Italy, avril 2008. ACM, ISBN 978-1-60558-011-1.

[Emi05] Emigh, Aaron: *Online Identity Theft : Phishing Technology, Chokepoints and Countermeasures*, octobre 2005. http://www.antiphishing.org/Phishing-dhs-report.pdf.

[Fac05] Factor, Hacker: *Anti-Phishing : Page Encoding - Whitepaper*, avril 2005. http://www.hackerfactor.com/papers/ap-page_encoding.pdf.

[FC10] Fung, Adonis P. H. et K. W. Cheung: *HTTPSLock : Enforcing HTTPS in Unmodified Browsers with Cached Javascript*. Dans *Proceedings of the 4th International Conference on Network and System Security (NSS)*, pages 269 – 274, Melbourne, Australia, septembre 2010. IEEE Computer Society, ISBN 978-1-4244-8484-3.

[FDWL06] Fu, Anthony Y, Xiaotie Deng, Liu Wenyin et Greg Little: *The methodology and an application to fight against Unicode attacks*. Dans *Proceedings of the 2nd Symposium On Usable Privacy and Security (SOUPS)*, page 91–101, Pittsburgh (Pennsylvania), USA, 2006. ACM, ISBN 1-59593-448-0.

[FHH+02] Friedman, Batya, David Hurley, Daniel C. Howe, Edward Felten et Helen Nissenbaum: *Users' conceptions of web security : a comparative study*. Dans *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 746–747, Minneapolis (Minnesota), USA, avril 2002. ACM, ISBN 1-58113-454-1.

[Fou] Foundation, Free Software: *GNU Wget*. http://www.gnu.org/software/wget/.

[Gat] Gathman, Stuart D.: *GNU Diff for Java*. http://www.bmsi.com/java/#diff.

[GJ09] Gupta, Pooja et Kalpana Johari: *Implementation of Web crawler*. Dans *Proceedings of the 2nd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, pages 838–843, Nagpur, India, décembre 2009. ISBN 978-1-4244-5250-7.

[GK05] Garofalakis, Minos et Amit Kumar: *XML stream processing using tree-edit distance embeddings*. ACM Transactions on Database Systems (TODS), 30(1) :279–332, mars 2005, ISSN 0362-5915.

[gooa] *Google Public DNS*. http://code.google.com/intl/fr/speed/public-dns/index.html.

[Goob] Google: *Google Whitelist*. http://sb.google.com/safebrowsing/update?version=goog-white-domain:1:1.

[Gooc] Google, Inc.: *Top 1000 sites - DoubleClick Ad Planner*. http://www.google.com/adplanner/static/top1000/.

[GPCR07] Garera, Sujata, Niels Provos, Monica Chew et Aviel D. Rubin: *A Framework for Detection and Measurement of Phishing Attacks*. pages 1–8, Alexandria (Virginia), USA, 2007. ACM, ISBN 978-1-59593-886-2.

[Gra03] Graham, Paul: *Better Bayesian Filtering*. Dans *Proceedings of 2003 MIT Spam Conference*, Cambridge (Massachusetts), Etats-Unis, janvier 2003. http://spamconference.org/proceedings2003.html.

[Gue06] Guette, Gilles: *Extensions de sécurité DNS (DNSSEC)*. Editions Techniques de l'Ingénieur, TE7553 :1–14, novembre 2006.

[Ham50] Hamming, Richard W.: *Error Detecting and Error Correcting Codes*. The Bell System Technical Journal, pages 147–161, avril 1950.

[Hir75] Hirschberg, D. S.: *A linear space algorithm for computing maximal common subsequences*. Communications of the ACM, 18(6) :341–343, juin 1975, ISSN 0001-0782.

[HYM09] Hara, Masanori, Akira Yamada et Yutaka Miyake: *Visual similarity-based phishing detection without victim site information*. pages 30–36, Nashville (Tennessee), USA, avril 2009. IEEE, ISBN 978-1-4244-2769-7.

[IET94] IETF: *RFC 1738 - Uniform Resource Locators (URL)*, décembre 1994. `http://www.ietf.org/rfc/rfc1738.txt`.

[Inf] InformAction: *NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience !* `http://noscript.net/`.

[IV10] ICANN et Verisign: *Root DNSSEC*, 2010. `http://www.root-dnssec.org/2010/07/16/status-update-2010-07-16/`.

[Jam06] James, Lance: *Phishing Exposed*. Syngress, 1re édition, janvier 2006, ISBN 978-1597490306.

[Jas] Jaspard, Emmanuel: *Les matrices de substitution*. `http://ead.univ-angers.fr/~jaspard/Page2/BIOINFORMATIQUE/7ModuleBioInfoJMGE/99Matrice/1Matrice.htm`.

[JBB+09] Jackson, Collin, Adam Barth, Andrew Botz, Weidong Shao et Dan Boneh: *Protecting browsers from DNS rebinding attacks*. ACM, 3, Issue 1(2), janvier 2009, ISSN 1559-1131.

[JWZ94] Jiang, Tao, Lusheng Wang et Kaizhong Zhang: *Alignment of trees — An alternative to tree edit*. Dans *Combinatorial Pattern Matching*, tome 807 de *Lecture Notes in Computer Science*, pages 75–86. Springer-Verlag Berlin, Heidelberg, crochemore, maxime and gusfield, dan édition, 1994.

[KCA+09] Kumaraguru, Ponnurangam, Justin Cranshaw, Alessandro Acquisti, Lorrie Faith Cranor et Jason Hong: *School of Phish : A Real-Word Evaluation of Anti- Phishing Training*. Technical Report CMU-CyLab-09-002, Carnegie Mellon University, mars 2009. `http://repository.cmu.edu/cgi/viewcontent.cgi?article=1011&context=cylab&sei-redir=1#search=%22real-word%20evaluation%20anti-phishing%20training%22`.

[KK05] Kirda, Engin et Christopher Kruegel: *Protecting Users against Phishing Attacks*. Tome 1, Edinburgh, Scotland, juillet 2005. IEEE Computer Society Press, ISBN 0-7695-2413-3.

[Kle] Kleiweg, Peter: *Levenshtein Demo*. `http://www.let.rug.nl/~kleiweg/lev/`.

[Kle98] Klein, Philip N.: *Computing the Edit-Distance between Unrooted Ordered Trees*. Dans *Proceedings of the 6th Annual European Symposium on Algorithms (ESA)*, pages 91–102, Venice, Italy, août 1998. Springer-Verlag.

[KMP77] Knuth, Donald, James H. Morris et Vaughan Pratt: *Fast pattern matching in strings*. SIAM Journal on Computing, pages 323–350, 1977. `http://en.wikipedia.org/wiki/Knuth-Morris-Pratt_algorithm`.

[KPKC07] Kim, Yeonjung, Jeahyun Park, Taehwan Kim et Joongmin Choi: *Web Information Extraction by HTML Tree Edit Distance Matching*. Dans *Proceedings of the 10th International Conference on Convergence Information Technology (ICCIT)*, page 2455–2460, Dhaka, Bangladesh, décembre 2007. IEEE Computer Society, ISBN 0-7695-3038-9.

[KR87] Karp, Richard M. et Michael O. Rabin: *Efficient randomized pattern-matching algorithms*. IBM Journal of Research and Development, pages 249–260, mars 1987, ISSN 0018-8646.

[KSTW07] Karlof, Chris, Umesh Shankar, J.D. Tygar et David Wagner: *Dynamic pharming attacks and locked same-origin policies for web browsers*. Dans *Proceedings of the 14th ACM conference on Computer and communications security*, pages 58–71, Alexandria (Viriginia), USA, 2007. ACM, ISBN 978-1-59593-703-2.

[LCJ+10] Lin, Dekang, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Larthbury, Vikram Rao, Kapil Dalwani et Sushant Narsale: *New Tools for Web-Scale N-grams*. Dans *Proceedings of the 7th International*

Conference on Language Resources and Evaluation (LREC), Malta, mai 2010. http://www.lrec-conf.org/proceedings/lrec2010/pdf/233_Paper.pdf.

[lev] *Levoyageur - Banks in the world*. http://www.levoyageur.net/banks.php.

[Lev66] Levenshtein, Vladimir I.: *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady, pages 707–710, février 1966.

[lib] *Librairie Java DNSJava*. http://www.xbill.org/dnsjava/.

[Lik05] Likic, Vladimir: *The Needleman-Wunsch algorithm for sequence alignment*, juillet 2005. http://www.ludwig.edu.au/course/lectures2005/Likic.pdf.

[LLL+07] Li, Jianxin, Jixue Liu, Chengfei Liu, Guoren Wang, Jeffrey Xu Yu et Chi Yangt: *Computing structural similarity of source XML schemas against domain XML schema*. Dans *Proceedings of the 19th Conference on Australasian Database (ADC)*, tome 75, page 155–164, Australia, jan 2007. Australian Computer Society, Inc., ISBN 978-1-920682-56-9.

[LMKK07] Ludl, Christian, Sean McAllister, Engin Kirda et Christopher Kruegel: *On the Effectiveness of Techniques to Detect Phishing Sites*. Dans *Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, tome Lecture Notes In Computer Science ; Vol. 4579, pages 20–39, Lucerne, Switzerland, 2007. Springer-Verlag Berlin, Heidelberg, ISBN 978-3-540-73613-4.

[MAA11] MAAWG: *Email Metrics Program : The Network Operators' Perspective - Report #14 - Third and Fourth Quarter 2010*, mars 2011. http://www.maawg.org/sites/maawg/files/news/MAAWG_2010_Q3Q4_Metrics_Report_14.pdf.

[Mai03] Maigron, Patrick: *Sécurité des e-mails : PGP et S/MIME*, octobre 2003.

[MG08] McGrath, D. Kevin et Minaxi Gupta: *Behind Phishing : An Examination of Phisher Modi Operandi*. Dans *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, page Article No. 4, San Francisco (California), USA, 2008. USENIX Association Berkely, CA, USA.

[Mic] Microsoft: *Phishing Filter and Resulting Internet Communication in Windows Vista*. http://technet.microsoft.com/en-us/library/cc721947(WS.10).aspx.

[Mic08] Microsoft: *Program :Win32/WareOut - Microsoft Malware Protection Center*, 2008. http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Program%3aWin32%2fWareOut.

[Mic09] Microsoft: *Microsoft Security Bulletin MS08-037 – Important : Vulnerabilities in DNS Could Allow Spoofing (953230)*, 2009. http://www.microsoft.com/technet/security/Bulletin/MS08-037.mspx.

[Mic10] Microsoft: *DNSSEC Deployment Guide for Windows Server 2008 R2 and Windows 7*, mars 2010. http://www.microsoft.com/download/en/details.aspx?DisplayLang=en&id=15204.

[MKK08] Medvet, Eric, Engin Kirda et Christopher Kruegel: *Visual-Similarity-Based Phishing Detection*. Dans *Proceedings of the 4th international conference on Security and privacy in communication networks*, page Article No. 22, Istanbul, Turkey, septembre 2008. ACM, ISBN 978-1-60558-241-2. http://iseclab.org/papers/visual-phishing-technical.pdf.

[Moz] Mozilla: *Phishing Protection : Design Documentation - MozillaWiki*. https://wiki.mozilla.org/Phishing_Protection:_Design_Documentation.

[MS05] Mikhaiel, Rimon et Eleni Stroulia: *Accurate and Efficient HTML Differencing*. Dans *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP)*, pages 163–172. IEEE Computer Society, 2005, ISBN 0-7695-2639-X.

[MSSV09] Ma, Justin, Lawrence K. Saul, Stefan Savage et Geoffrey M. Voelker: *Beyond Blacklists : Learning to Detect Malicious Web Sites from Suspicious URLs*. Dans *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254, Paris, France, juillet 2009. ACM, ISBN 978-1-60558-495-9.

[Mye86] Myers, Eugene W.: *An O(ND) Difference Algorithm and Its Variations*. Algorithmica, pages 251–266, 1986.

[Net] Netcraft: *Top 100 Most Visited Web Sites*. http://toolbar.netcraft.com/stats/topsites.

[Net09] Networks, F5: *The End of DNS As We Know It*, août 2009. http://devcentral.f5.com/weblogs/macvittie/archive/2009/08/28/the-end-of-dns-as-we-know-it.aspx.

[NW70] Needleman, Saul B. et Christian Wunsch: *A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins*. Molecular Biology Journal, pages 443–553, 1970.

[oK] Kentucky, University of: *BLOSUM62 Matrix*. http://www.uky.edu/Classes/BIO/520/BIO520WWW/blosum62.htm.

[Oll04] Ollmann, Gunter: *The Phishing Guide - Understanding and Preventing Phishing Attacks*, septembre 2004. www.ngssoftware.com/papers/nisr-wp-phishing.pdf.

[Oll05] Ollmann, Gunter: *The Pharming Guide*, juillet 2005. http://www.ngssoftware.com/papers/ThePharmingGuide.pdf.

[Olz06] Olzak, Tom: *DNS Cache Poisoning : Definition and Prevention*, mars 2006. http://adventuresinsecurity.com/Papers/DNS_Cache_Poisoning.pdf.

[ope] *OpenDNS*. http://www.opendns.com/.

[Ora05] Oracle: *Bug ID : 6247501 java.net.InetAddress cache is not disabled, even if networkaddress.cache.ttl=0*, mars 2005. http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6247501.

[PD06] Pan, Ying et Xuhua Ding: *Anomaly Based Web Phishing Page Detection*. Dans *Proceedings of the 22nd Annual Computer Security Applications Conference*, pages 381–392. IEEE Computer Society Press, décembre 2006, ISBN 0-7695-2716-7.

[Phe10] Phenoelit, Darklab.org affiliate: *Default Password List*. http ://www.phenoelit-us.org/dpl/dpl.html, octobre 2010. http://www.phenoelit-us.org/dpl/dpl.html.

[phi] *PhishTank*. http://www.phishtank.com/.

[PKKG10] Prakash, Pawan, Manish Kumar, Ramana Rao Kompella et Minaxi Gupta: *PhishNet : Predictive Blacklisting to Detect Phishing Attacks*. Dans *IEEE INFOCOM Proceedings*, pages 1–5, San Diego (California), USA, mars 2010. IEEE, ISBN 978-1-4244-5836-3.

[Reu11] Reuters: *Le navigateur Firefox détrône Internet Explorer en Europe*, janvier 2011. http://fr.reuters.com/article/idFRPAE7030FN20110104.

[RJM+05] Ross, Blake, Collin Jackson, Nick Miyake, Dan Boneh et John C Mitchell: *Stronger password authentication using browser extensions*. Dans *Proceedings of the 14th Conference on USENIX Security Symposium*, tome 14, Baltimore (Maryland), USA, juillet 2005. USENIX Association.

[RKKF07] Rosiello, Angelo P. E., Engin Kirda, Christopher Kruegel et Fabrizio Ferrandi: *A layout-similarity-based approach for detecting phishing pages*. pages 454–463, Nice, France, septembre 2007. IEEE, ISBN 978-1-4244-0974-7.

[RRJ11] Reddy, Venkata Prasad, V. Radha et Manik Jindal: *Client Side protection from Phishing attack*. International Journal of Advanced Engineering Sciences and Technologies (IJAEST), pages 39–45, 2011.

[RSA08] RSA: *One Sinowal Trojan + One Gang = Hundreds of Thousands of Compromised Accounts*, octobre 2008. http://blogs.rsa.com/rsafarl/one-sinowal-trojan-one-gang-hundreds-of-thousands-of-compromised-accounts/.

[Sch08] Schneier, Bruce: *Lesson From the DNS Bug : Patching Isn't Enough*, juillet 2008. http://www.schneier.com/essay-230.html.

[Ser06] Services, WOT: *WOT (Web of Trust) now with PhishTank*, novembre 2006. http://www.mywot.com/en/blog/now-with-phishtank.

[She] Shed, Developer: *Domain Typo Generator - SEO Tools - Search Engine Optimization, Google Optimization*. http://www.seochat.com/seo-tools/domain-typo-generator/.

[SL05] Simon, Kai et Georg Lausen: *ViPER : augmenting automatic information extraction with visual perceptions*. Dans *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, page 381–388, Bremen, Germany, octobre 2005. ACM, ISBN 1-59593-140-6.

[Sou] Sourceforge.net: *B.A.B.A - Basic Algorithms of Bioinformatics Applet*. `http://baba.sourceforge.net/`.

[spe05] spectrum, IEEE: *Panix Attack*, février 2005. `http://spectrum.ieee.org/telecom/security/panix-attack`.

[Squ] Squirrel, Blue: *WebWhacker 5.0*. `http://www.bluesquirrel.com/products/webwhacker/`.

[SRM07] Stamm, Sid, Zulfikar Ramzan et Jakobsson Markus: *Drive-By Pharming*. Dans *Proceedings of the 9th international conference on Information and communications security*, tome Network Security, pages 495–506, Zhengzhou, China, 2007. ACM, ISBN 978-3-540-77047-3.

[SW81] Smith, Temple F. et Michael S. Waterman: *Identification of Common Molecular Subsequences*. Molecular Biology Journal, pages 195–197, 1981.

[SWW+09] Sheng, Steve, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong et Zhang Chengsan: *An Empirical Analysis of Phishing Blacklists*. Moutain View (California), USA, juillet 2009.

[Sym08] Symantec: *Trojan.Flush.M Technical Details*, décembre 2008. `http://www.symantec.com/security_response/writeup.jsp?docid=2008-120318-5914-99&tabid=2`.

[Sym10] Symonds, Peter: *Coding Theory - Part2 : Hamming Distance*, 2010. `http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf`.

[Tec] Technologies, Sly: *Librairie Java jNetPcap*. `http://jnetpcap.com/`.

[tin] *TinyURL*. `http://tinyurl.com/`.

[Tou05] Touzet, Hélène: *A linear-time algorithm for comparing similar ordered trees*, juillet 2005. `http://www.cs.ucr.edu/~stelo/cpm/cpm05/cpm05_8_4_Touzet.pdf`.

[Tur50] Turing, Alan Mathison: *Computer Machinery and Intelligence*. Mind, 1950.

[UoC] California, Irvine University of: *Librairie Jpcap*. `http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/`.

[W3Ca] W3C: *Document Object Model*. `http://www.w3.org/DOM/`.

[W3Cb] W3C: *The global structure of an HTML document*. `http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html`.

[W3Cc] W3C: *Le W3C confirme qu'HTML5 sera disponible en mai 2011 pour appel à commentaires*. `http://www.w3.org/2011/02/htmlwg-pr.html.fr`.

[W3C01] W3C: *URIs, URLs, and URNs : Clarifications and Recommendations 1.0*, septembre 2001. `http://www.w3.org/TR/uri-clarification/`.

[WAP08] Wendlandt, Dan, David G. Andersen et Adrian Perrig: *Perspectives : Improving SSH-style Host Authentication with Multi-Path Probing*. Dans *Proceedings of the 2008 USENIX Annual Conference*, Boston (Massachussets), USA, juillet 2008.

[web] *Web Browser Market Share*. `http://statowl.com/web_browser_market_share_trend.php?1=1&timeframe=last_6&interval=month&chart_id=4&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&chart_id=`.

[WF74] Wagner, Robert A. et Michael J. Fischer: *The String-to-String Correction Problem*. Journal of the ACM, pages 168–173, janvier 1974, ISSN 0004-5411.

[WHX+05] Wenyin, Liu, Guanglin Huang, Liu Xiaoyue, Zhang Min et Xiaotie Deng: *Detection of phishing webpages based on visual similarity*. Dans *Special interest tracks and posters of the 14th international conference on World Wide Web (WWW)*, page 1060–1061, Chiba, Japan, mai 2005. ACM, ISBN 1-59593-051-5.

[Win06] Winkler, William E.: *Overview of Record Linkage and Current Research Directions*. Rapport technique 2006-2, Statistical Research Division, U.S. Census Bureau, Washington D.C., USA, février 2006. `http://www.census.gov/srd/papers/pdf/rrs2006-02.pdf`.

[WMG06] Wu, Min, Robert C. Miller et Simon L. Garfinkel: *Do security toolbars actually prevent phishing attacks ?* Dans *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, Montreal (Quebec), Canada, 2006. ACM, ISBN 1-59593-372-7.

[WML06] Wu, Min, Robert C. Miller et Greg Little: *Web Wallet : Preventing Phishing Attacks by Revealing User Intentions*. Dans *Proceedings of the 2nd Symposium On Usable Privacy and Security (SOUPS)*, pages 102–113, Pittsburg (Pennsylvania), USA, juillet 2006. ACM, ISBN 1-59593-448-0. `groups.csail.mit.edu/uid/.../phishing/soups-webwallet.pdf`.

[WMMM90] Wu, Sun, Udi Manber, G. Myers et Web Miller: *An O(NP) sequence comparison algorithm*. Information Processing Letters, pages 317–323, septembre 1990, ISSN 0020-0190.

[wor] *WorldIP free geolocation database, service and tools*. http ://www.wipmania.com/en/. `http://www.wipmania.com/en/`.

[WTV+10] Wang, Kuansan, Christopher Thrasher, Evelyne Viegas, Xiaolong Li et Bo june (Paul) Hsu: *An overview of Microsoft web N-gram corpus and applications*. Dans *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, Demonstration Session, page 45–48, Los Angeles (California), USA, juillet 2010. Association for Computational Linguistics.

[YS06] Yee, Ka-Ping et Kragen Sitaker: *Passpet : convenient password management and phishing protection*. Dans *Proceedings of the 2nd Symposium On Usable Privacy and Security (SOUPS)*, page 32–43, Pittsburg (Pennsylvania), USA, juillet 2006. ACM, ISBN 1-59593-448-0. `http://passpet.org/`.

[YW08] Yue, Chuan et Haining Wang: *Anti-Phishing in Offense and Defense*. Dans *Proceedings of the 2008 Annual Computer Security Applications Conference (ACSAC)*, page 345–354, Anaheim (California), USA, décembre 2008. IEEE Computer Society, ISBN 978-0-7695-3447-3.

[ZECH07] Zhang, Yue, Serge Egelman, Lorrie Cranor et Jason Hong: *Phinding Phish : Evaluating Anti-Phishing Tools*. Dans *Proceedings of the 14th Annual Network & Distributed System Security Symposium*, San Diego (California), USA, mars 2007.

[ZHC07] Zhang, Yue, Jason Hong et Lorrie Cranor: *CANTINA : A Content-Based Approach to Detecting Phishing Web Sites*. Dans *Proceedings of the 16th international conference on World Wide Web*, pages 639–648, Banff (Alberta), Canada, mai 2007. ACM, ISBN 978-1-59593-654-7.

[ZS89] Zhang, K. et D. Shasha: *Simple fast algorithms for the editing distance between trees and related problems*. SIAM Journal on Computing, pages 1245–1262, décembre 1989, ISSN 0097-5397.

Liste des publications

Conférences

[GGL11] Gastellier-Prevost, Sophie, Gustavo Gonzalez Granadillo et Maryline Laurent: *Decisive heuristics to differentiate legitimate from phishing sites.* Dans *Proceedings of the 6th Conference on Network and Information Systems Security (SAR-SSI)*, pages 1–9, La Rochelle, France, mai 2011. ISBN 978-1-4577-0735-3.

[GGL11b] Gastellier-Prevost, Sophie, Gustavo Gonzalez Granadillo et Maryline Laurent: *A dual approach to detect pharming sites at the client-side.* Dans *Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, Paris, France, février 2011. ISBN 978-1-4244-8705-9.

[GL11] Gastellier-Prevost, Sophie et Maryline Laurent: *Defeating pharming attacks at the client-side.* Dans *Proceedings of the 5th International Conference on Network and System Security (NSS)*, pages 33–40, Milan, Italy, septembre 2011. ISBN 978-1-4577-0458-1.

Revue/Collection

[Gas09] Gastellier-Prevost, Sophie: *Le spam.* Editions Techniques de l'Ingénieur, H5450 :1–22, avril 2009.