Modélisation et évaluation de la sécurité des parcours d'authentification

Youssou Ndiaye

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Youssou NDIAYE

Modélisation et évaluation de la sécurité des parcours d'authentification

Thèse présentée et soutenue à Rennes, le 10.12.2019
Unité de recherche : IRISA – UMR6074
Thèse N° :

Rapporteurs avant soutenance :

Jean-Michel BRUEL	Professeur des universités Université de Toulouse
Anne ETIEN	Maître de conférences Université de Lille 1

Composition du Jury :

Rapporteurs

Jean-Michel BRUEL	Professeur des universités Université de Toulouse
Anne ETIEN	Maître de conférences Université de Lille 1

Examinateurs

Xavier BLANC	Professeur des universités Université de Bordeaux
Jacques KLEIN	Senior research scientist Université de Luxembourg

Directeur de thèse

Olivier BARAIS	Professeur des universités Université de Rennes

Encadrants

Nicolas AILLERY	Ingénieur Orange
Arnaud BLOUIN	Maître de conférences INSA Rennes
Ahmed BOUABDALLAH	Maître de conférences IMT Atlantique

Résumé

Les applications informatiques sont aujourd'hui largement déployées sur plusieurs plateformes (*e.g.*, smartphone, ordinateur, console de jeux). Une instance de la même application peut être accessible en simultané sur ces plateformes. Cette hétérogénéité augmente certes le confort d'utilisation, mais rend les applications plus complexes, notamment en terme de sécurité, particulièrement, lorsqu'il s'agit de vérifier l'*identité* de l'utilisateur final.

Cette vérification d'identité, bien qu'elle soit fondée sur des normes et des protocoles éprouvés, manque la considération des risques introduits par les éléments de contexte. Ces éléments de contexte sont principalement l'environnement d'exécution (*e.g.*, navigateur web, application mobile), les choix d'utilisabilité, le comportement humain et les erreurs de conception.

Cette thèse fournit des outils et méthodes qui permettent aux concepteurs d'améliorer la conception des parcours d'authentification en prenant en considération ces éléments de contexte. Cette démarche nécessite d'identifier les biens que l'on souhaite protéger ainsi que les risques et les menaces sur ces derniers. Pour une application Web/Mobile, le moyen d'authentification de ses utilisateurs est capital pour éviter tout accès non-autorisé. Cette authentification est souvent réalisée lors des phases de développement tout en étant dépourvue de spécification préalable. Cette absence de spécification empêche de déterminer les défauts qui en résultent, mais aussi ceux liés aux interfaces utilisateurs, aux choix d'usage et au comportement de l'utilisateur final.

Dans un premier temps, nous étudions l'impact de ces éléments de contexte sur le parcours d'authentification. Puis, nous en déduisons une *taxonomie* des défauts qualifiés de **défauts logiques**. Enfin, nous définissons un **ensemble d'exigences** à respecter lors des **phases de conception** afin de prévenir ces défauts de sécurité.

Dans un second temps, nous proposons de palier le manque de spécification du parcours d'authentification dans le processus de développement. Dans ce but, nous proposons un environnement de modélisation fondé sur un langage dédié (*i.e.*, Langage Spécifique au Domaine - *Domaine Specific Language* (DSL)) à la spécification des parcours d'authentification. Cet environnement, qui implémente un ensemble d'abstractions que nous définissons via un cadre d'évaluation du niveau de risques, offre

deux avantages importants. Premièrement, il permet d'étendre les moyens d'authentification existants en considérant les éléments de contexte (*i.e.*, plasticité de l'interface utilisateur, l'utilisabilité, etc.). Deuxièmement, pour une spécification de parcours donnée, il permet d'évaluer automatiquement le niveau de risque d'attaques logiques associées.

Abstract

Software Applications are being ubiquitous in our daily life. One application may run in different platforms (*e.g.*, Web, Mobile, Gaming Console) while being accessible simultaneously. This heterogeneity improves the usability from the user's perspective. However, it makes the applications more complex to maintain when they evolve, especially when it comes to verifying the user's identity and authenticity.

Approved standards and protocols provide a mean to ensure the user's identity. Nevertheless, they lack considering risks introduced by factors of this heterogeneous context. Some of the factors are : execution environment, usability choice, design errors and user's behaviour. This thesis provides tools and approaches that allow designers to improve the security design of their applications while considering elements from the real-life context. This approach implies to identify the main assets to protect, the risks and the threats. Our approach involves Web/Mobile applications and focuses on the authentication procedure of the end-user since this is vital to avoid unauthorized access to the legitimate user's resources. These authentications, while leveraging on approved authentication schemes, considerably lack a formal specification during the design phase.

First, we investigate the impact of the contextual elements on the authentication procedure. So, we identify the relevant flaws that we characterize and then define as logic flaws. To overcome their flaws, we provide a set of requirements that aim to tackle them during the design phase.

Second, to overcome the lack of formal specifications of the authentication procedure during the design phase, we provide a Domain-Specific Language (DSL). This dedicated language implements the abstractions of a risk assessment framework that we provide. The DSL extends existing authentication schemes while considering real-life contexts. Then, from a given specification, it provides the result of the risk assessment of the identified logic attacks.

Table des matières

1 Introduction **10**
1.1 Contexte . . . 10
1.1.1 Le parcours d'authentification . . . 11
1.1.2 Exemple illustratif . . . 13
1.2 Problématique de la thèse . . . 16
1.2.1 Vers les défauts logiques d'authentification . . . 17
1.2.2 De l'étude empirique à l'ontologie des défauts logiques d'authentification . . . 18
1.3 Contributions . . . 19
1.3.1 Contribution 1 : Une taxonomie des exploitations logiques et une démarche d'ingénierie des exigences . . . 19
1.3.2 Contribution 2 : Un cadre d'évaluation du niveau de risques . . . 19
1.3.3 Contribution 3 : Un environnement de modélisation du parcours d'authentification . . . 20
1.4 Organisation du document . . . 21

2 Contexte et état de l'art **22**
2.1 Éléments de contexte . . . 22
2.1.1 La sécurité des applications Web/Mobile . . . 22
2.1.2 Les Méthodes d'authentification . . . 23
2.2 État de l'art . . . 28
2.2.1 Taxonomie et ontologies des failles de sécurité . . . 30
2.2.2 Défauts logiques de conception . . . 34
2.2.3 Langage de modélisation . . . 36
2.2.4 Ingénierie des exigences . . . 38
2.2.5 Test et détection automatique de failles logiques . . . 41
2.2.6 Failles logiques dans la sécurité et la vie privée . . . 43
2.2.7 Évaluation du niveau de sécurité . . . 43
2.3 Conclusion de l'état de l'art . . . 49

3 Exigences de sécurité pour l'amélioration du parcours d'authentification 51
3.1 Étude empirique . . . 51
3.1.1 Les critères de choix des applications . . . 52
3.1.2 Démarche de l'étude empirique . . . 52
3.1.3 Résultats de l'étude empirique . . . 53
3.1.4 Conclusions de l'étude empirique . . . 58
3.1.5 Atteintes à la validité . . . 58
3.2 Caractérisation des exploitations logiques . . . 59
3.2.1 Défauts logiques liés à la phase d'enregistrement . . . 59
3.2.2 Défauts logiques liés au mécanisme d'authentification . . . 60
3.2.3 Vers une ontologie du parcours d'authentification . . . 63
3.3 Les exigences de sécurité . . . 64
3.3.1 Exigences sur la vérification des attributs d'identité . . . 64
3.3.2 Exigences sur les mécanismes d'authentification . . . 66
3.4 Conclusion du chapitre . . . 69

4 Cadre d'évaluation du niveau de risques des exploitations logiques relatives à un parcours d'authentification 71
4.1 Préambule . . . 71
4.1.1 Définition du niveau de risque dans ce cadre . . . 71
4.1.2 Critères d'évaluation du niveau de risque . . . 72
4.2 Évaluation du niveau de risque de souscription d'une entité frauduleuse . 74
4.2.1 Évaluation du niveau de risque associé aux attributs d'identité numérique . . . 74
4.2.2 Évaluation du niveau de risque associé aux attributs d'identité physique . . . 76
4.2.3 Récapitulatif de l'évaluation du niveau de risque de souscription d'une entité frauduleuse . . . 79
4.3 Évaluation du niveau de risque d'un accès non autorisé . . . 79
4.3.1 Authentification à un facteur . . . 80
4.3.2 Évaluation du niveau de risque associé à un mécanisme d'authentification à plusieurs facteurs . . . 83
4.3.3 Évaluation du niveau de risque associé à une fédération d'identité . 85
4.3.4 Niveau de risque d'un accès non autorisé : phase de login . . . 85
4.3.5 Niveau de risque d'un accès non autorisé : phase de récupération d'une preuve d'identité . . . 86
4.3.6 Processus d'évaluation globale du niveau de risque d'un accès non autorisé . . . 87
4.4 Usurpation d'identité : substitution de l'entité légitime . . . 88
4.5 Conclusion du chapitre . . . 89

5 Environnement de modélisation du parcours d'authentification 92
5.1 Concepts fondamentaux 92
5.2 Processus de description d'un parcours d'authentification 93
5.3 Concepts fondamentaux du DSL 95
5.3.1 Syntaxe abstraite et exigences 96
5.3.2 Syntaxe concrète 97
5.4 Processus d'évaluation du niveau de risque 101
5.4.1 Risque de souscription frauduleuse via la phase d'enregistrement . 102
5.4.2 Risque d'accès non autorisé via la phase de login et la phase de mise à jour de preuves de sécurité 104
5.5 Risque de substitution de l'entité légitime via la phase de mise à jour . . . 106
5.6 Conclusion du chapitre 108

6 Conclusion et Perspectives 110
6.1 Conclusion 110
6.2 Une ontologie pour améliorer l'écosystème du développement des parcours d'authentification 111
6.2.1 Cas d'usage 111
6.2.2 Implémentation et réponses aux questions 112
6.3 Vers la fin des cookies de session 113

Table des figures

1.1 Modélisation du parcours d'authentification par un système de transition . 12
1.2 Formulaire de login Amazon 14
1.3 Parcours de récupération d'un mot de passe perdu ou volé 15
1.4 Vue d'ensemble des contributions et leurs relations 20

2.1 Système d'authentification biométrique 25
2.2 Exemple illustratif des courbes FRR, FAR, ERR 26
2.3 Vue d'ensemble des différents domaines de l'état de l'art. 30
2.4 Exemple d'ontologie avec les classes, les relations et les instances de classes 32
2.5 Exemple illustratif du contournement de chemin de navigation 34

3.1 Illustration d'un déni de service d'accès à Ameli de la population des femmes du département 93 nées au mois de Mai 1970 56
3.2 Illustration d'une demande simultanée de validation d'une demande de connexion avec l'application *Mobile Connect et moi* ou *Paylib* 57
3.3 Illustration d'un changement de Pin faible avec l'application *Mobile Connect et moi* 58
3.4 Ontologie du parcours d'authentification avec les classes de défauts logiques et les exploitations correspondantes 65

4.1 Processus d'évaluation du niveau de risque 73
4.2 Les paramètres d'entrée de l'évaluation du niveau de risque de souscription d'une entité frauduleuse. 75
4.3 Paramètres d'entrée du processus d'évaluation du niveau de risque d'accès non autorisé 80
4.4 Arbre de décision de l'évaluation du niveau de risque d'un accès non autorisé associé à la phase de login 86
4.5 Arbre de décision de l'évaluation du niveau de risque d'un accès non autorisé via la phase de récupération de preuves d'identité 88
4.6 Arbre de décision de l'évaluation des niveaux de risque associé aux différents phase du parcours au vu des exploitations logiques. 91

5.1 Métamodèle simplifié du DSL 95
5.2 Environnement de modélisation du parcours d'authentification 96
5.3 Métamodèle complet du DSL 109

6.1 Ontologie du mécanisme d'authentification 112
6.2 illustration des requêtes DLQuery et leurs résultats 113
6.3 Illustration de l'usage du protocole FIDO U2F par l'alliance fido 114
6.4 Illustration de l'usage du protocole FIDO UAF par l'alliance fido 114

Liste des tableaux

2.1 Liste non exhaustive de menace sur les phases d'enregistrement et d'authentification ainsi que les contrôles (*i.e.,* #1 à #20) appropriés pour atteindre le niveau d'assurance requis. 45
2.2 Récapitulatif des différents niveaux xAL du Nist 46
2.3 Choix du *xAL* en fonction de l'évaluation du niveau des impacts 47

3.1 Récapitulatif des résultats de l'étude empirique 54
3.2 Taxonomie des exploitations logiques et les exigences correspondantes . . 70

4.1 Exemple d'évaluation du niveau de risque de compromission d'un site web 72
4.2 Récapitulatif de l'évaluation du niveau de risque d'une souscription d'une entité frauduleuse. 79
4.3 Résumé de l'évaluation du niveau de risque associé à une authentification à un facteur 82
4.4 Liste de téléphones ayant passé le test de déverrouillage avec une photo de l'utilisateur légitime. 83
4.5 Évaluation globale du niveau de risque d'un accès non autorisé (*i.e.,* Imitation de l'entité légitime). 89

5.1 Rapport d'évaluation du niveau de risque d'une souscription frauduleuse de l'application de partage voiture 102
5.2 Rapport de sensibilisation du niveau de risque d'une souscription frauduleuse de l'application de partage voiture 103
5.3 Rapport d'évaluation du niveau de risque d'un accès non autorisé via la phase de login et de récupération de preuve d'identité. 105
5.4 Rapport de sensibilisation d'un accès non autorisé de l'application de partage de voiture. 107
5.5 Rapport d'évaluation du niveau de risque de substitution de l'entité légitime. 108

Chapitre 1

Introduction

Afin de délimiter notre cadre de travail, nous présentons dans ce chapitre les différents points constituant les éléments de contexte et la problématique de cette thèse. Celle-ci s'inscrit dans le cadre de développement de méthodes appropriées pour la spécification et la vérification de parcours d'authentification dans le contexte des applications Web/Mobile. Ces méthodes doivent être outillées afin d'assister les concepteurs de parcours dans le processus de spécification, mais aussi doivent permettre d'évaluer le niveau de sécurité de ces parcours.
La section 1.1 présente le contexte général de cette thèse. La section 1.2 présente les problèmes que cette thèse résout en répondant aux différentes questions de recherche. La section 1.3 présente les différentes contributions de cette thèse. La section 1.4 donne l'organisation de ce document.

1.1 Contexte

Les processus de développement des applications Web/Mobile deviennent de plus en plus complexes du fait de plusieurs paramètres à prendre en considération. Parmi ces paramètres : la sécurité, particulièrement, l'authentification des utilisateurs de ces services digitalisés. L'authentification désigne le processus par lequel une *entité* prouve une identité qu'elle revendique auprès d'une autre *entité* en mesure de vérifier cette identité (*i.e.*, le *vérifieur*) [79, 95]. Ce concept est un point fondamental dans le domaine de la sécurité des systèmes et des services. D'un contexte à l'autre, d'un domaine applicatif à l'autre, les protocoles d'authentification utilisés diffèrent. On peut citer les applications client-serveur dont l'authentification entre l'application client et le serveur peut se faire via un système de certificats [46, 75] ou l'authentification des équipements réseaux avec le protocole *EAP* (*i.e.*, *Extensible Authentification Protocol*) [173].

Cette thèse se focalise sur l'authentification d'une entité par un *vérifieur* pour accéder à un service donné. Particulièrement, nous étudions les risques d'un défaut d'authentification dans le domaine des applications Web/Mobile en se focalisant sur les interactions entre l'utilisateur et le service.

Malheureusement, cette authentification des utilisateurs est souvent réduite au choix de protocoles et mécanismes d'authentification (*e.g.*, le couple identifiant/mot de passe, le reconnaissance de l'empreinte digitale). L'authentification, cependant, doit nécessiter davantage d'attention du fait des facteurs de risques généralement liés au cycle de vie de l'application dans la vie réelle.

Ces risques sont souvent occasionnés par des choix de conception (*e.g.*, utilisabilité, sécurité) ou par le comportement de l'utilisateur final (*e.g.*, choix d'usage, insouciance).

Pour illustrer comment ces facteurs peuvent conduire à des défauts de sécurité, nous proposons d'étudier le parcours d'authentification d'un site de commerce en ligne : Amazon France [1].

Avant de présenter cet exemple, il devient primordial d'introduire le concept de parcours d'authentification.

1.1.1 Le parcours d'authentification

Les recommandations du NIST [79, 26] et de l'ITU [95] ont défini des cadres de gestion des identités pour les services numériques. Ces cadres définissent le cycle de vie d'un attribut d'identité depuis sa création, son stockage, son usage pour l'authentification, son renouvellement et sa révocation.

Ces recommandations ont inspiré notre définition de parcours d'authentification. Souvent, le parcours d'authentification est réduit au processus par lequel, un utilisateur prouve la revendication de son identité devant une entité en mesure de vérifier cette identité. Il s'agit de la phase d'authentification (*login*) qui permet à un utilisateur de créer une session utilisateur en cas de succès (*i.e.*, accès au service demandé).

Cependant, d'autres composants de la gestion de l'identité, souvent ignorés ou négligés, peuvent permettre à un utilisateur d'accéder à un service sans pour autant passer par la phase classique de *login*. Par exemple, en cas d'oubli de son mot de passe, un utilisateur peut demander à le réinitialiser et créer un nouveau mot de passe pour accéder à son compte. Ce parcours alternatif aboutit donc à l'accès au service comme la phase login. Ainsi, focaliser toute l'attention (*i.e.*, en terme de sécurité) sur la phase de *login* tout en négligeant ou en ignorant d'autres phases peuvent potentiellement créer de nouveaux de risques.

Pour prendre en compte ces parcours alternatifs, nous considérons le parcours d'authentification comme étant structuré en quatre phases.

1. `https://www.amazon.fr`

— **La phase d'enregistrement ou l'enrôlement :** Cette phase correspond à l'enregistrement des attributs d'identité de l'utilisateur, mais aussi à la création des preuves de sécurité par lesquelles celui-ci prouve son identité lors des phases suivantes.
— **La phase d'authentification ou *login* :** Cette phase correspond au processus par lequel l'utilisateur déjà enregistré (*i.e.*, *entité souscrite*), revendique et prouve son identité auprès d'un vérifieur afin d'accéder au service.
— **La phase de récupération :** Cette phase correspond à la récupération (suivie d'un renouvellement) d'attributs d'identité ou de preuve d'identité d'un utilisateur préalablement enregistré.
— **La phase de renouvellement :** Cette dernière phase est similaire à la phase de récupération. Par ailleurs, celle-ci nécessite que l'utilisateur soit dans une session active (*i.e.*, un utilisateur déjà authentifié par le service).

Ces différentes phases du parcours d'authentification constitue une machine à états telle qu'illustrée par la Figure 1.1. Les phases du parcours représentent les transitions vers les différents états de l'utilisateur (*i.e.*, enrôlé, non connecté et connecté). La procédure d'enrôlement permet à une entité de souscrire à un service. Lorsqu'une entité est souscrite (*i.e.*, état *enrôlé*), celle-ci peut créer une session (*i.e.*, état *connecté*) via la procédure de *login* ou demander renouvellement de ces attributs d'identité via la procédure de *récupération*. Le renouvellement des attributs d'identité peut-être réalisé lors d'une session active via la procédure de *mise à jour*. Cette dernière procédure est différente de la procédure de *récupération* du fait que la première est accessible lorsque l'entité est à l'état *connecté* alors que la seconde n'est accessible que si l'utilisateur est à l'état *non connecté*.

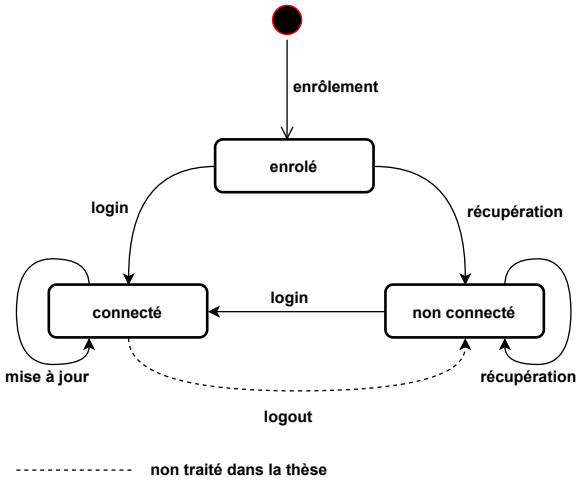

Figure 1.1 – Modélisation du parcours d'authentification par un système de transition

Cette définition du parcours d'authentification fixe le cadre d'étude des défauts de sécurité que cette thèse vise à éviter. La section suivante présente un cas d'usage qui illustre la problématique liée à ces défauts de sécurité.

1.1.2 Exemple illustratif

Cette section présente une étude qui permet d'illustrer comment les choix de conception et le comportement de l'utilisateur final peuvent conduire à des défauts de sécurité exploitables par une personne mal-intentionnée. Cette étude est réalisée avec le parcours d'authentification du site de commerce en ligne Amazon France.

Selon la définition du parcours d'authentification présenté dans la Section 1.1.1, le parcours d'authentification inclut les processus de *login*, de récupération et de mise à jour de preuves sécurité car la finalité de ceux-ci est la création d'une session utilisateur. Par ailleurs, il est important de préciser que le but de l'authentification est d'assurer que l'entité demandeuse d'accès est bien celle qu'elle prétend être [26]. Cela implique que tous les processus du parcours d'authentification devront disposer d'un moyen garantissant l'identité de l'entité.

Le formulaire de *login*

Sur le formulaire de *login* présenté sur la Figure 1.2, nous pouvons constater que celui-ci est pré-rempli par le gestionnaire de mot de passe installé sur le navigateur. Le bouton de sélection "*keep my session active*" permet de garder la session active pour une durée indéterminée. L'analyse montre également que le nombre de tentatives pour deviner le mot de passe est illimité.

Le processus de récupération de mot de passe perdu

Le parcours de récupération d'un mot de passe (oublié ou perdu) s'appuie sur un code à usage unique envoyé par SMS ou par mail. Ce processus est renforcé par la connaissance de la date d'expiration de la carte bancaire si celle-ci avait été ajoutée préalablement (*cf.* Figure 1.3).

Le processus de changement de mot de passe

Une entité préalablement authentifiée peut changer le mot de passe courant moyennant la connaissance de celui-ci. Cependant, le champ de remplissage de ce mot de passe courant est pré-rempli par le gestionnaire de mot de passe. De plus, le nombre de tentative pour deviner ce mot de passe est illimité.

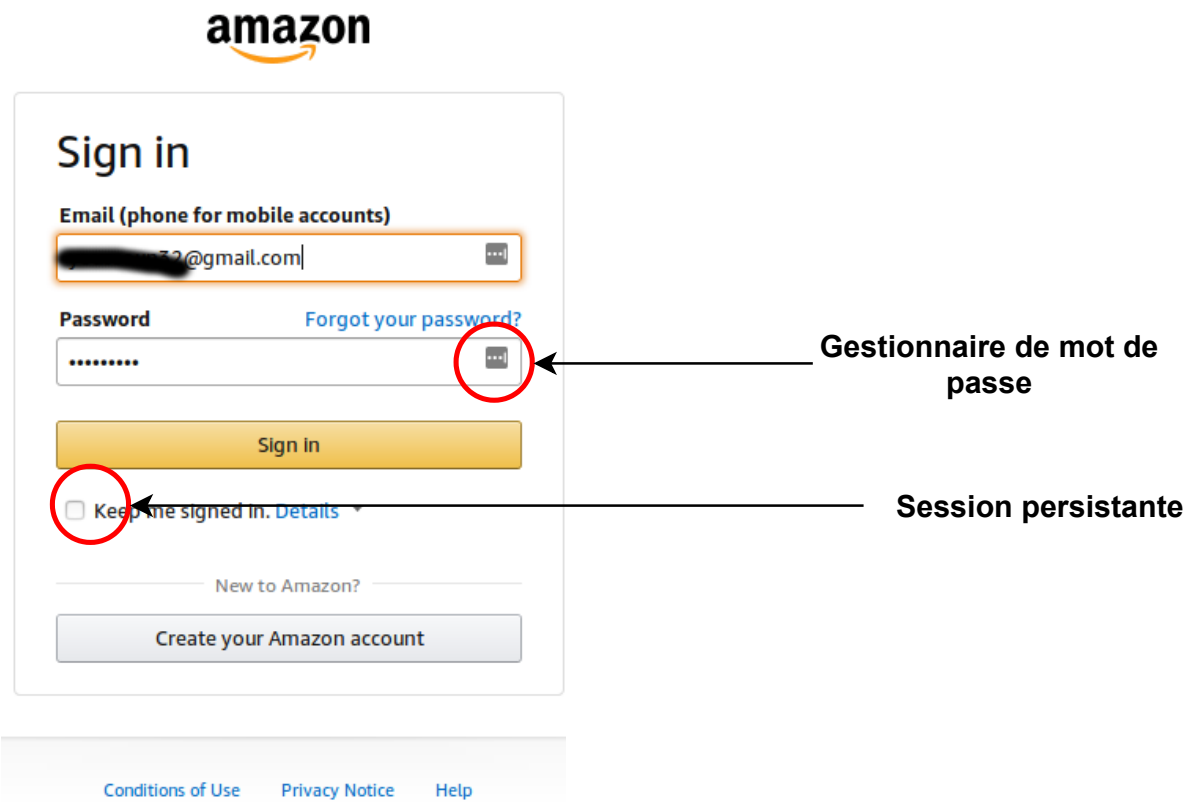

Figure 1.2 – Formulaire de login Amazon

Les exploitations possibles

Dans l'hypothèse où l'appareil est accessible par une entité autre que l'entité légitime, le cas d'un vol ou la perte de l'appareil [20, 177, 10], nous considérons que ce parcours comporte des défauts de sécurité. Ces défauts peuvent conduire à des exploitations telles que l'imitation de l'entité légitime ou sa substitution par une entité malicieuse [123].

Particulièrement, c'est l'authenticité de l'entité légitime qui est mise en cause pour les raisons suivantes.

— Le formulaire pré-rempli pose un problème fondamental d'authentification. En principe, l'authentification se fonde sur la connaissance d'un secret partagé entre un *vérifieur* et une *entité souscrite* (*i.e.*, le cas d'une authentification fondée sur la connaissance) [79]. Cette propriété n'est plus garantie si le formulaire est préalablement rempli du secret. Ainsi, le contrôle du dispositif de l'entité légitime garantit l'accès à l'ensemble des services sans connaître les secrets du mécanisme d'authentification.

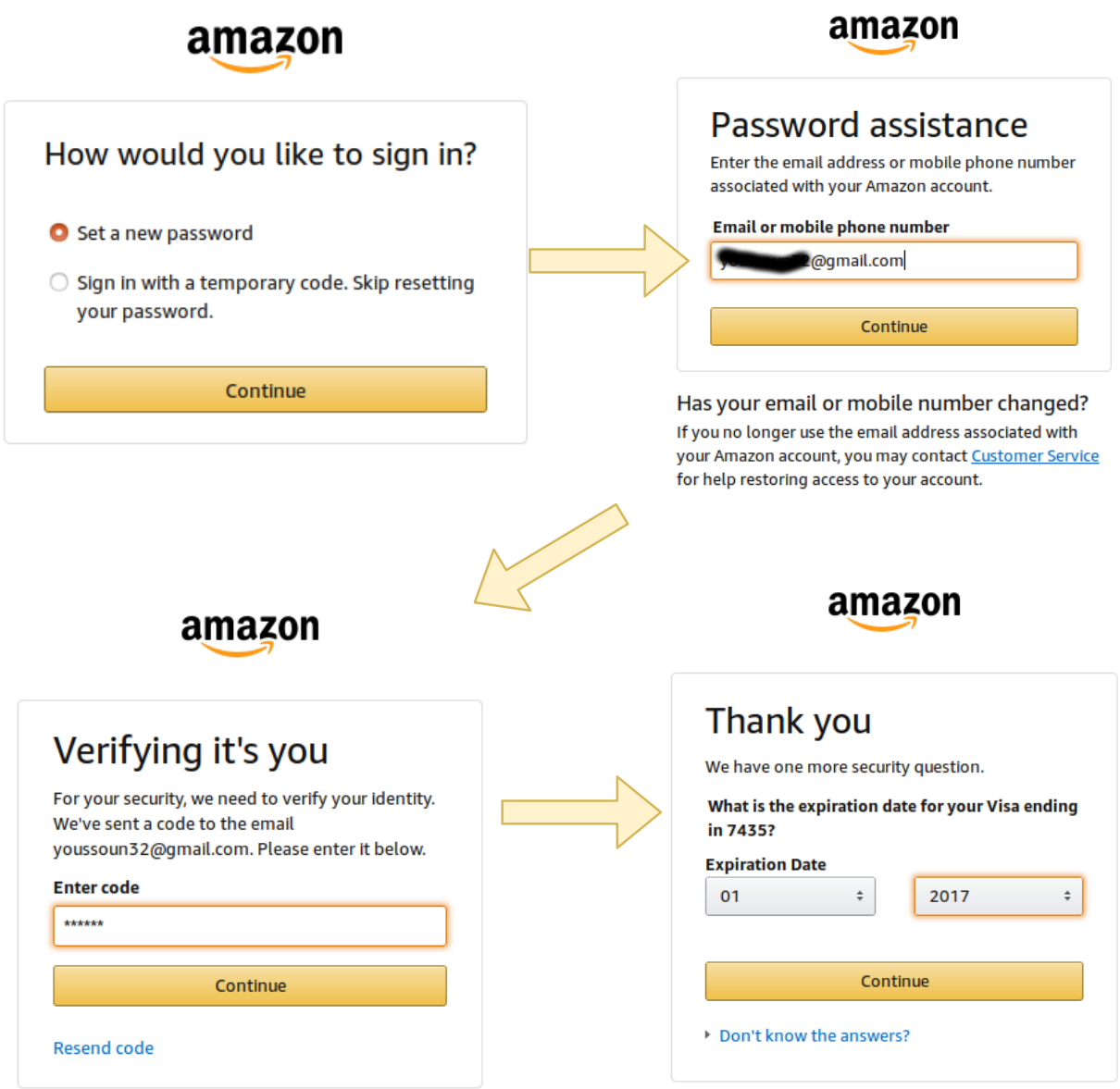

Figure 1.3 – Parcours de récupération d'un mot de passe perdu ou volé

— Concernant, la session persistante, c'est la durée de celle-ci qui fait défaut. Elle permet à une entité malicieuse d'accéder à un compte légitime sans passer par le formulaire d'authentification.
— Le parcours de récupération du mot de passe est faible pour deux raisons.

1. Le code à usage unique envoyé est susceptible d'être récupéré car les applications de messagerie (*i.e.*, mail ou SMS) sont souvent actives sur le même appareil à cause des sessions persistantes.

2. Le challenge de sécurité est faible car il nécessite un maximum de 60 essais pour être deviné. Ceci est dû au format des cartes bancaires qui ont en général une durée de validité 5 ans au plus ($12 \times 5 = 60$).

— Enfin, un intrus pourra se substituer à l'entité légitime en changeant ses attributs d'identité. En effet, la sécurité du changement de mot de passe est fondée sur la connaissance de l'ancien mot de passe. Par contre, cette sécurité est violée par le formulaire qui est préalablement rempli par des extensions du navigateur.

Conclusion de l'exemple illustratif

Les exploitations décrites ci-dessus permettent d'identifier de nouveaux risques tels que l'usurpation de l'utilisateur légitime en cas de perte ou de vol de son appareil personnel. L'étude montre que ce risque est dû, d'une part, aux choix d'usages l'utilisateur final (*i.e.*, usage de formulaires pré-remplis). D'autre part, ces risques sont dus à la spécification de l'application : l'usage de challenge de sécurité faible pour la récupération du mot de passe et maintient de session persistante.

Grâce à cet exemple illustratif, nous montrons que la sécurité du parcours d'authentification de l'application est fortement corrélée à la protection du dispositif de l'utilisateur légitime. Par exemple, si l'utilisateur établie une session persistante, alors, il suffirait à une personne malicieuse d'accéder au dispositif de l'utilisateur pour accéder au service en question sans avoir à s'authentifier. Ceci constitue un défaut de spécification qui semble supposer que l'appareil n'est accessible que par l'entité légitime. Cette "hypothèse" implicite de sécurité change fondamentalement la nature du parcours d'authentification et introduit de nouveaux défauts non pris en compte lors des phases de spécification.

Cette thèse s'intéresse à ces défauts de sécurité et prétend que la spécification des parcours d'authentification doit prendre en considération les facteurs à l'origine de ces défauts de sécurité dès la phase de conception.

1.2 Problématique de la thèse

L'exemple illustratif présenté dans la section précédente montre l'existence de défauts de sécurité sur les parcours d'authentification. Les causes de ces défauts de sécurité et leur impacts (négatifs) sur la sécurité des parcours d'authentification est le problème central que cette thèse propose de résoudre. La première question de recherche que nous adressons est la suivante :

> **RQ1 : Quels sont les risques relatifs aux parcours d'authentification, quelles sont les causes et origines de ces risques ?**

La réponse à cette première question de recherche nous amène à caractériser ces défauts en défauts logiques.

1.2.1 Vers les défauts logiques d'authentification

Pour identifier et caractériser les risques relatifs au parcours d'authentification, nous avons réalisé une enquête au sein des équipes (différentes entités au sein d'Orange) de développement et de conception des applications. Nous avons également analysé des défauts sur les applications existantes au sein d'Orange et d'ailleurs (*e.g.*, Skype, Yahoo).

Les résultats de cette étude révèlent entre autres que le processus de spécification des parcours d'authentification se limite à spécifier, uniquement, les mécanismes d'authentification à utiliser. De plus, grâce à la modélisation des menaces [153, 165], nous découvrons que les défauts identifiés ne sont pas dus à des défauts sur les mécanismes d'authentification mais aux détournements de l'usage légitime du système d'authentification d'où leur qualification en défauts logiques.

En effet, deux grands types de défauts de sécurité existent dans la sécurité des applications Web/Mobile [91, 171] : Les *bugs* (y compromis les vulnérabilités) et les défauts de conception (*i.e.*, failles) de sécurité. Les différences fondamentales sont dans leurs exploitations par un attaquant et leurs points d'entrées.

Un *bug* est un défaut localisé qui est dû à une erreur d'implémentation d'un service (*e.g.*, erreur de vérification des variables d'entrée, *buffer overflow*). Pour exploiter de tels défauts, l'attaquant tire parti de ces erreurs (*e.g.*, injection de code malvaillant [84, 149]). À l'opposé, une faille de sécurité (ou faute de conception) est due à une erreur dans la conception ou la définition des exigences [91]. Ainsi, indépendamment de l'implémentation du service, un défaut introduit durant ce stade de la conception risque de persister tout au long du cycle de vie de l'application concernée. Par exemple, une application de vente en ligne qui stocke les informations d'achats (*e.g.*, prix, nombre de produits) dans un cookie non sécurisé serait en défaut. En effet, une personne malicieuse peut modifier le prix de son achat sur le cookie malgré le fait que l'application soit bien implémentée.

Dans cette dernière catégorie de failles de sécurité, nous nous focalisons les failles logiques de sécurité définies par OWASP [72] comme étant le détournement de l'usage légitime d'un système à des fins malicieuses. L'adaptation de cette définition au parcours d'authentification donne la définition suivante :

> *Une faille ou défaut logique d'authentification est une conséquence de l'usage légitime du parcours d'authentification de manière à ce que cet usage produit un impact négatif pour le service ou l'application concernée.*

Du fait que le comportement logique d'une application varie en fonction du domaine applicatif et du contexte, il advient que les failles logiques aussi varient d'une

application à une autre. De plus, grâce à l'exemple illustratif, on constate également que ce détournement de l'usage légitime d'une application peut être dû aux choix d'usages de l'utilisateur ou aux choix de conception. Ainsi, intuitivement, nous cherchons à vérifier si ces failles logiques peuvent être généralisées sur d'autres parcours d'authentification. Cela introduit la question suivante :

> **Dans quelles conditions les failles logiques peuvent mettre en défaut les parcours d'authentification des applications Web/Mobile ? Comment peut-on les caractériser ?**

1.2.2 De l'étude empirique à l'ontologie des défauts logiques d'authentification

Pour répondre à la question posée dans la section précédente, nous conduisons une évaluation empirique du parcours d'authentification d'applications grands publiques. Cette étude empirique a permis, d'une part de généraliser l'existence de failles logiques sur ces applications. D'autre part, elle a permis de confirmer que ces failles de sécurité sont dues à des défauts de spécification qui ne prennent en considération ni le comportement de l'utilisateur final, ni les éléments de contexte qui peuvent modifier le schéma initial d'authentification et introduire de nouveaux risques. Ces découvertes introduisent la deuxième question de recherche à savoir :

> **RQ2 : Dans quelles mesures le processus de spécification pourrait-il prendre en considération les défauts de sécurité identifiés ?**

Pour répondre à cette deuxième question, nous avons caractérisé les défauts logiques à l'aide d'une ontologie du parcours d'authentification, puis en déduire une taxonomie qui va permettre de capturer un ensemble d'exigences de sécurité spécifiques à ce contexte de défauts logiques de sécurité.

Ces exigences permettent de prendre en considération ces défauts logiques dès la phase de conception. En effet, parce que ces défauts logiques dépendent du concept de l'application, ils sont difficilement détectables par les méthodes de test et de détection des vulnérabilités [91, 171]. De plus, notre enquête au sein des équipes de conception logiciel révèle que les parcours d'authentification ne sont pas formalisés durant la conception des applications et l'implémentation de ceux-ci est à la charge des développeurs. C'est tout naturellement que ces derniers ont recours à des *frameworks* ou des API d'authentification dont la conception initiale n'a pas prévu les failles logiques dont nous discutons dans cette thèse. Par conséquent, ces failles logiques ne sont jamais testées, et leur existence dans la vie réelle du parcours d'authentification est ignorée des concepteurs.

Par ailleurs, proposer des exigences sans un formalisme adéquat, peut entraîner des confusions quant à leur interprétation. De plus, cette thèse vise à réduire le gap

entres les concepteurs et les développeurs en proposant des outils pour faciliter leurs échanges et permettre l'application des exigences pertinentes.

C'est à la recherche d'une telle approche que nous explorons la question de recherche suivante :

> **RQ3 : Quelles approches permettent de spécifier les exigences de sécurité d'un parcours d'authentification ? Comment vérifier le respect ou non de ces exigences ?**

Cette question de recherche sous-entend une approche qui soit capable de prévenir les défauts logiques dès la phase de conception. Pour cela, celle-ci doit permettre la spécification des exigences de sécurité capturées, mais aussi la sensibilisation des concepteurs sur les risques liés à cette spécification.

C'est dans ce contexte que nous proposons un cadre d'évaluation du niveau de risques des défauts logiques relatifs à une spécification d'un parcours d'authentification. Afin de valider ce cadre d'évaluation, nous proposons un langage dédié qui, d'une part, permet de spécifier un parcours d'authentification avec ces exigences de sécurité, et d'autre part, se fonde sur le cadre d'évaluation afin de donner l'évaluation du niveau de risque des attaques logiques et en même temps sensibiliser les concepteurs sur ces risques.

Les démarches proposées pour répondre à ces questions de recherche sont organisées autour de trois contributions majeures.

1.3 Contributions

La Figure 1.4 présente une vue d'ensemble des différentes contributions détaillées ci-dessous.

1.3.1 Contribution 1 : Une taxonomie des exploitations logiques et une démarche d'ingénierie des exigences

Notre première contribution est une approche généraliste qui consiste à identifier les défauts logiques et à les caractériser en fonction des composants du parcours d'authentification que nous avons définis. Ensuite, nous proposons un ensemble d'exigences de sécurité dans le but de lutter contre ces défauts logiques dès la phase de conception du parcours d'authentification.

1.3.2 Contribution 2 : Un cadre d'évaluation du niveau de risques

La deuxième contribution s'inscrit dans le cadre de l'évaluation du niveau de sécurité ou inversement du niveau de risque de la spécification d'un parcours d'authen-

tification. En ce sens, elle propose un cadre d'évaluation du niveau de risque dont la sémantique se fonde, d'une part, sur la définition du parcours d'authentification et d'autre part, sur l'ensemble des exigences de sécurité que nous avons établi. Ce cadre permet ainsi d'évaluer la capacité de la spécification d'un parcours d'authentification à prévoir ou à éviter les classes de défauts logiques caractérisées dans la première contribution.

1.3.3 Contribution 3 : Un environnement de modélisation du parcours d'authentification

Enfin, la troisième et dernière contribution propose de formaliser la spécification d'un parcours d'authentification, via des abstractions adaptées, sous forme d'un environnement de modélisation fondé sur un langage dédié. Ces abstractions dont la sémantique est fondée sur le cadre d'évaluation, offrent deux avantages : d'une part, elles permettent de spécifier le parcours d'authentification en prenant en compte les éléments de contexte et améliorent également la communication entre les concepteurs et les experts sécurité, d'autre part, elles permettent de vérifier la satisfaction ou non des exigences de sécurité en fournissant un rapport d'évaluation des risques relatifs à cette spécification.

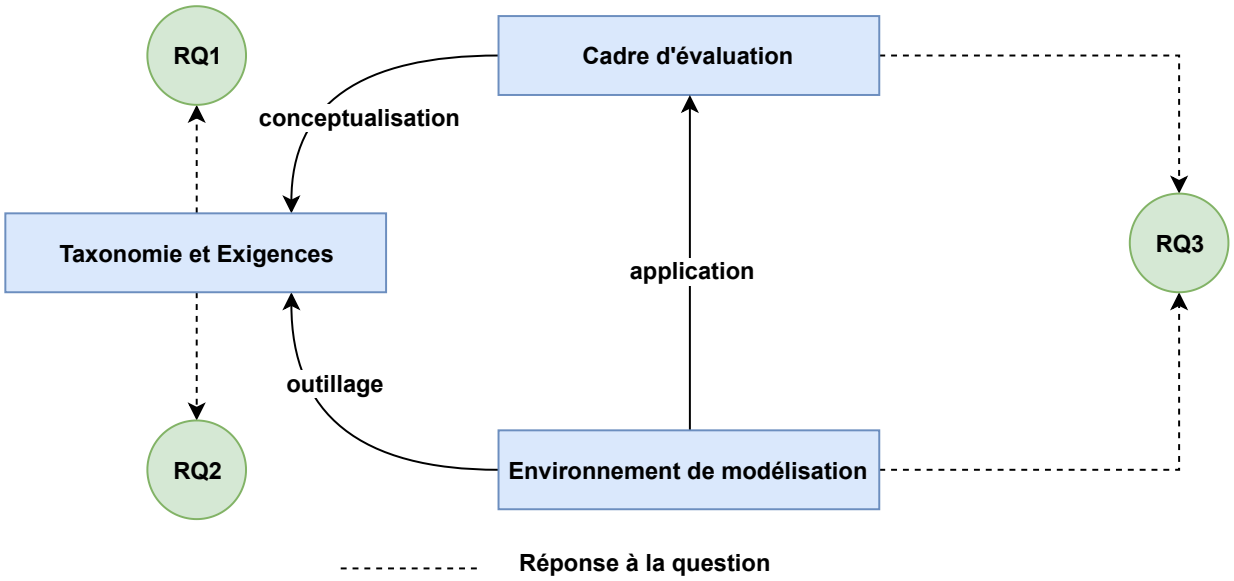

Figure 1.4 – Vue d'ensemble des contributions et leurs relations

1.4 Organisation du document

La suite de ce document est organisée en deux parties. La première partie est constituée du chapitre d'état de l'art (*i.e.*, Chapitre 2) où nous présentons les éléments de contexte général nécessaires à l'élaboration des concepts développés dans cette thèse. Également, nous y présentons l'état de l'art relatif aux différentes contributions de cette thèse.

La seconde partie du document introduit nos différentes contributions. Les trois premiers chapitres de cette partie présentent les trois contributions de cette thèse. Le Chapitre 3 étudie les défauts logiques en conduisant une étude empirique sur des applications grand public, puis, caractérise les défauts logiques et propose un ensemble d'exigences de sécurité pour se prémunir de ces défauts logiques.

Le Chapitre 4 propose un cadre d'évaluation du niveau de risque relatif à la spécification d'un parcours d'authentification. Ce cadre permet d'évaluer le risque en se basant sur les concepts introduits dans le Chapitre 3 (*i.e.*, classes de défauts logiques, exigences de sécurité).

Le Chapitre 5 propose un environnement dédié à la spécification d'un parcours d'authentification. Cet environnement se fonde sur un langage dédié qui permet de décrire un parcours d'authentification en prenant en compte les concepts du Chapitre 3. De plus, il permet une évaluation automatique du niveau de risque en s'appuyant sur la sémantique du cadre d'évaluation du Chapitre 4.

Enfin, le dernier chapitre (*i.e.*, Chapitre 6) de cette partie est consacré aux perspectives et à la conclusion de cette thèse.

Chapitre 2

Contexte et état de l'art

Ce chapitre d'état de l'art est constitué de deux parties. La première partie est consacrée aux éléments de contexte qui ont contribué à la construction de nos différentes contributions. La seconde partie est consacrée aux travaux connexes à cette thèse.

2.1 Éléments de contexte

2.1.1 La sécurité des applications Web/Mobile

La sécurité occupe une place importante dans le processus de développement des applications en général. Fondamentalement, le but de la sécurité est de s'assurer des propriétés ci-après pour un système donné.

— La confidentialité stipule que seules les entités autorisées peuvent accéder en lecture aux ressources qui leurs sont destinées. Cette propriété est souvent traitée à l'aide des méthodes de chiffrement, de gestion des droits d'accès et des permissions [150, 182] ou par l'usage de protocoles d'échanges via des canaux sécurisés [46, 75, 145].

— L'authentification garantit que tout utilisateur d'un système doit prouver l'identité qu'il revendique. Cette notion est différente de l'identification qui consiste à trouver l'identité d'une entité au sein d'un groupe [184].

— L'intégrité assure que les données échangées lors d'une communication sont telles qu'elles sont attendues. Autrement dit, que les données n'ont pas été altérées ni modifiées. La garantie de cette propriété est assurée à l'aide de fonctions de hachage ou de vérification de somme de contrôle (*checksum* en anglais).

— La disponibilité assure que l'accès aux ressources d'un système est garanti en permanence dans le temps imparti aux utilisateurs légitimes. Plusieurs aspects sont à prendre en considération pour garantir cette propriété ; les événements

les plus redoutés sont les attaques de déni de service (*i.e.*, *DoS - Denial of Service*) [127].

En plus de ces quatre propriétés fondamentales, d'autres propriétés peuvent être considérées notamment : la non-répudiation ou la traçabilité. Ceci étant dit, dans cette thèse nous nous focalisation sur la propriété d'authentification pour les raisons expliquées dans l'introduction. Les protocoles cryptographiques sous-jacents ne sont pas dans le cadre de cette thèse, nous présentons ci-après l'état de l'art des différents méthodes d'authentification existantes.

2.1.2 Les Méthodes d'authentification

Fondamentalement, il existe deux types d'authentification : l'authentification mutuelle et l'authentification simple. Dans l'authentification mutuelle, les entités concernées s'authentifient mutuellement l'une auprès de l'autre. Cet aspect de l'authentification n'est pas traité dans cette thèse. En effet, nous nous focalisons sur l'authentification d'une entité pour accéder à un service : processus par lequel une entité prouve son identité auprès d'un vérifieur.

Ce type d'authentification comprend deux déclinaisons principales : l'authentification à un seul facteur et l'authentification à plusieurs facteurs. Un facteur d'authentification étant :

— quelque chose que l'on connait : *e.g.*, un mot de passe ;
— quelque chose que l'on est : *e.g.*, un comportement unique (*e.g.*, la manière de marcher) ou un attribut biométrique (*i.e.*, empreinte digitale) ;
— quelque chose que l'on a : *e.g.*, il s'agit d'un appareil physique (*i.e.*, une clé physique) ou d'un ensemble d'information contenu dans un appareil physique protégé ou non (*i.e.*, carte à puce).

L'authentification fondée sur la connaissance d'un secret

L'authentification fondée sur la connaissance d'un secret est parmi les moyens d'authentification les plus utilisés aujourd'hui. Cette utilisation massive est favorisée par sa facilité de déploiement, de gestion et de stockage comparée aux autres facteurs d'authentification. Celle-ci peut offrir un niveau de sécurité satisfaisant quand les exigences sont bien définies (*e.g.*, qualité du secret, chiffrement). Cependant, elle peut aussi être source de faiblesse dans le système du fait de ces nombreuses limitations liées à la capacité de l'homme à retenir une succession de chiffres et de lettres [151, 96] ou à une mauvaise implémentation (*i.e.*, absence de chiffrement, faible entropie [121]). Ci-après quelques exemples de méthodes d'authentification fondées sur la connaissance.

Le mot de passe : Souvent associé à un identifiant, le mot de passe est le moyen d'authentification le plus utilisé pour garantir l'authenticité des entités utilisatrices d'une application web ou mobile. Malgré les nombreux avantages qu'il offre, le mot de passe est sujet à un risque majeur : le vol par une entité malicieuse (ou la découverte). Ce risque peut-être occasionné par l'utilisateur final ou par le fournisseur de service lui-même. Ci-après, la plupart des causes pouvant favoriser ce risque de vol.

- **La réutilisabilité** est l'usage d'un même mot de passe sur plusieurs services différents. Le problème majeur de cette pratique est l'effet domino que la compromission de celui-ci peut causer [96]. En effet, si le mot de passe est compromis via un site avec une protection faible des mots de passe ou par négligence de l'utilisateur [1, 151], alors l'ensemble des autres services seront également compromis.
- **L'absence de chiffrement ou protection anti-brute force** : Dans leur étude sur les mécanismes d'authentification fondés sur le mot de passe, Bonneau *et al.* [22] montrent que sur 150 sites web étudiés, 84% n'ont pas de mécanisme anti brute force et 57% n'utilisent pas de TLS [46] pour sécuriser la transmission de ce mot de passe. De plus l'étude montre que 29% des sites web étudiés envoient en clair le mot de passe à l'utilisateur ce qui permet de constater que les mots de passe n'étaient pas hachés avant d'être stockés en base de données. Cette dernière pratique constitue un problème de sécurité (*i.e.*, en cas de compromission de la base de données), mais aussi un problème d'atteinte à la vie privée (*i.e.*, le fournisseur de service a accès aux mots de passe de ces utilisateurs).
- **La qualité du mot de passe** constitue aussi un aspect très important dans la politique de sécurité d'une authentification. En effet, si le choix du mot de passe est très contraignant, cela peut causer des problèmes de sécurité tels que montré par Sasse *et al.* [1]. À l'inverse, ce mot de passe ne doit pas être faible sous peine de faciliter son exploitation par un utilisateur malicieux. Par conséquent, un compromis est nécessaire pour allier sécurité et utilisabilité [151, 48].

Le code pin : La particularité du code pin est qu'il est constitué uniquement de caractères numériques, très souvent entre 4 et 8 chiffres. Il est moins sécurisé que le mot de passe en terme d'entropie, cependant dans certains contextes d'usage (*e.g.*verrouillage d'écrans) il est considéré comme d'un niveau de sécurité acceptable [20]. En effet, le code pin est souvent utilisé pour protéger des appareils qui sont souvent sous le contrôle de l'utilisateur légitime (*i.e.*, téléphone portable). Par ailleurs, le code pin peut être facilement découvert par ingénierie sociale, surtout quand une personne malicieuse a accès aux données personnelles d'un utilisateur légitime (*e.g.*, date de naissance, code postale) [21, 152]. Notons aussi que le code pin est souvent

associé à d'autres facteurs (*e.g.*, code pin et carte à puce) afin de renforcer la sécurité du mécanisme d'authentification.

La question secrète L'authentification via une question secrète se fonde sur la capacité à répondre à une question connue de l'utilisateur légitime et du vérifieur. Souvent fondée sur les préférences (*e.g.*, date de naissance, ville préférée, nom de jeune fille de votre mère), elle est considérée comme très faible car facilement devinable [152, 98, 142]. Du fait qu'elle est plus facile à retenir que le mot de passe, la question secrète est beaucoup utilisée comme alternative pour récupérer un mot de passe ou un identifiant perdu ou révoqué [21]. Cependant, à cause de son niveau de sécurité faible, elle est exploitée pour frauduleusement accéder à une application [80].

L'authentification biométrique

L'authentification à facteur biométrique permet d'authentifier une personne en se fondant sur ses attributs physiologiques (*e.g.*, empreinte digitale, le visage) ou sur ses attributs comportementaux (*e.g.*, façon de parler, façon de marcher) [17].

Fondamentalement, un système d'authentification biométrique est constitué de deux phases : la phase d'enrôlement qui consiste à recueillir les points caractéristiques qui serviront de gabarit et la phase de comparaison où le gabarit préalablement stocké est comparé à la nouvelle acquisition [97] (Figure 2.1).

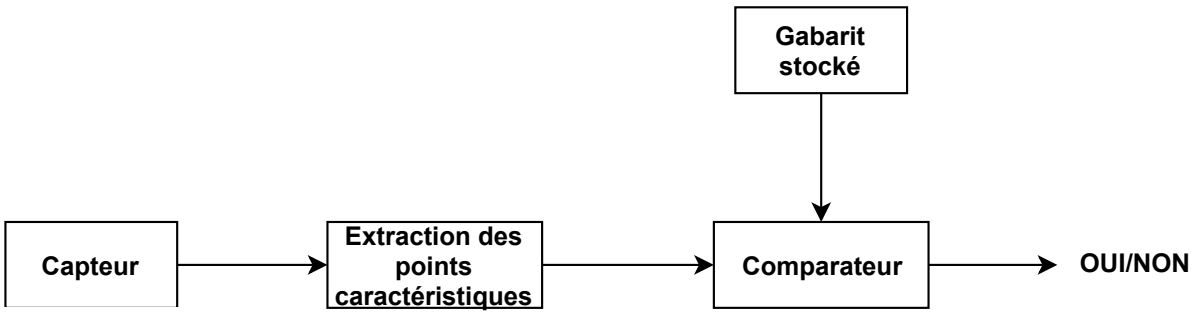

Figure 2.1 – Système d'authentification biométrique

Notons que la manière d'obtenir le gabarit de comparaison varie d'une modalité biométrique à l'autre (*e.g.*, extraction de minuties, Eigenface [16], DTW [129]), de même que la méthode de comparaison des gabarits (*e.g.*, distance euclidienne, similarité).

La performance d'un système biométrique peut être évaluée de différentes façons, cependant le Taux de Fausses Acceptations - *False Acceptance Rate* (FAR) et le Taux de Faux Rejets - *False Rejection Rate* (FRR) sont les caractéristiques les plus regardées. Le FAR désigne le nombre de personnes acceptées à tort alors que le FRR est le nombre de personnes rejetées à tort. Ainsi, plus le FAR est faible, plus le système est

réputé être fiable. De même, on cherche à avoir un FRR suffisamment faible, tout en gardant la latitude pour ne pas frustrer le vrai utilisateur qui se fera rejeter à tort. Ainsi, un compromis est nécessaire entre ces deux valeurs. Un troisième paramètre permet d'améliorer le système : le Taux d'Égal Erreur - Equal Erreur Rate (EER) qui permet de fixer le seuil (*i.e.*, *threshold* en anglais) optimal d'acceptation. La Figure 2.2 illustre la relation entre ces trois caractéristiques. En abscisse nous avons le score (*i.e.*, seuil d'acceptation) représenté en similarité ou en distance euclidienne et en ordonné nous avons les taux de rejet et d'acceptation.

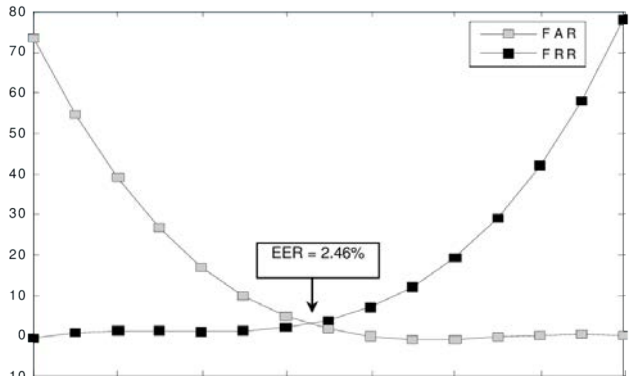

Figure 2.2 – Exemple illustratif des courbes FRR, FAR, ERR

L'une des faiblesses majeures d'un système d'authentification biométrique est l'irrévocabilité de la modalité biométrique. En effet, si une modalité biométrique est compromise, celle-ci le sera pour la vie, car il est impossible de changer ses caractéristiques biométriques. C'est pourquoi, la protection de cette empreinte biométrique est capitale pour un système d'authentification biométrique. Des techniques existent pour pallier à cette faiblesse, notamment le *bio-hashing* qui consiste à associer une empreinte biométrique et une matrice variable permettant la mise à jour du bio-code généré (*i.e.*, *bio-hash*) sans changer de modalité biométrique tout en garantissant l'authenticité de la personne légitime [99, 120].

En dehors du vol de la modalité biométrique (plus précisément des points caractéristiques d'une modalité biométrique), la parodie (*i.e.*, *spoofing* en anglais) est l'attaque la plus élaborée sur les systèmes biométriques. Fondamentalement, elle consiste à présenter une fausse empreinte biométrique de la personne légitime pour s'authentifier à sa place [179, 122]. Notons qu'il existe d'autres types d'attaque : l'attaque par interception (*i.e.*, *man in the middle*), l'attaque du capteur (*i.e.*, *template attack*) ou le rejeu (*i.e.*, *replay-attack*) [179].

L'authentification par la possession

L'authentification par la possession se fonde sur la capacité d'une entité à présenter un secret qu'elle est la seule à posséder. Ce secret est souvent un ensemble

d'informations stockées sur une carte physique telle qu'une carte à puce ou un *token* contenu sur une clé USB ou tout autre dispositif physique. Très exposé au vol ou à la copie, ce facteur d'authentification est souvent associé à un autre facteur afin d'augmenter le niveau de sécurité. On peut citer l'association d'une carte physique et d'un code PIN pour authentifier le détenteur d'une carte bancaire lors d'une transaction.

L'authentification multi-facteurs

L'authentification multi-facteurs est la combinaison de plusieurs facteurs d'authentification dans le but d'authentifier une entité. En combinant ainsi plusieurs facteurs, un système d'authentification multi-facteurs permet de combler les faiblesses de l'un des facteurs par les avantages de l'autre. Par exemple en combinant un facteur de possession et un facteur biométrique (*i.e.*, une carte à puce et empreinte digitale), le mécanisme d'authentification obtenu est moins vulnérable au vol de la carte à puce (*i.e.*, *Match-On-Card* [172]). En effet, même en cas de vol de celle-ci, l'absence de l'empreinte digitale de la personne légitime empêchera la validation de toutes les transactions. C'est cette capacité à renforcer la sécurité et parfois sans friction d'utilisabilité que l'authentification à deux facteurs est aujourd'hui très utilisée par les applications Web/Mobile. À cette effet, la spécification FIDO 2.0 est massivement implémentée par les acteurs de la sécurité [161]. Ce standard propose un protocole appelé U2F (*Universal Two Factor Authentification*) dont le but est de supprimer l'usage du mot de passe (*i.e.*, *Passwordless*) tout en garantissant une meilleure sécurité.

La fédération d'authentification

La fédération d'authentification pour un service consiste à confier la gestion de l'authentification de ses utilisateurs à une instance appelée Fournisseur d'Identités - *Identity Provider* (IdP) [134]. Ce processus d'authentification s'inscrit souvent dans un contexte de fédération d'identité où l'identité d'une entité est partagée entre plusieurs services sous le contrôle d'une autorité de confiance. Le mécanisme d'authentification le plus utilisé dans ce contexte est l'Authentification Unique - *Single Sign ON* (SSO). Le principe d'une SSO consiste à s'authentifier une seule fois pour accéder à l'ensemble des services fédérés par une même IdP via des assertions de sécurité [144, 118, 112]. Ce système de SSO permet ainsi d'améliorer l'expérience utilisateur en réduisant le nombre de demandes d'authentification. Par ailleurs, en plus d'être exposée aux mêmes risques que les autres méthodes d'authentification, l'authentification par SSO expose davantage l'utilisateur à l'effet domino [96]. En effet, si l'IdP est compromis, il en résulte la compromission de tous les services qu'il gère.

L'authentification continue et adaptative

L'authentification continue intervient toujours après la procédure classique d'authentification. Elle consiste à monitorer en temps réel l'activité de l'utilisateur d'un service dans le but d'évaluer l'authenticité de celui-ci en se basant sur ses habitudes d'utilisation et/ou sur des éléments de contexte (*e.g.*, la localisation, la durée d'utilisation, le dispositif habituel) [13, 74]. Ce principe d'authentification continue se fonde sur la dépréciation du niveau de confiance qu'un service peut avoir sur l'authenticité de l'utilisateur courant. Cette dépréciation peut-être fonction du temps ou de changement des éléments de contexte. C'est en se basant sur un seuil de confiance qui est fonction des éléments de contexte que le système d'authentification réévalue le niveau de confiance pour renouveler ou non la demande d'authentification. Ce système d'authentification est ainsi adaptatif lorsque le moyen d'authentification utilisé pour rétablir le niveau de confiance varie en fonction du résultat de l'évaluation des éléments de contexte [93].

2.2 État de l'art

Pour limiter, empêcher, prévenir la présence de défauts logiques dans la spécification des parcours d'authentification, cette thèse propose différentes contributions qui gravitent autour de trois axes : la modélisation dans le sens de description de parcours d'authentification, la définition d'exigences de sécurité et l'évaluation du niveau de sécurité ou inversement du niveau de risque d'un parcours d'authentification.

La Figure 2.3 présente les différents domaines de recherche connexes à ces contributions. Ci-après, nous détaillons en quoi ces domaines de recherche constituent l'état de l'art de cette thèse.

Taxonomies et Ontologies des failles de sécurité

Les failles logiques sont au cœur de la problématique de cette thèse. La Section 2.2.1 et 2.2.2 discute des taxonomies et ontologies des failles de sécurité en général, puis nous nous focalisons sur celles ayant considérées les failles logiques. En outre, nous nous sommes appuyé sur ces études pour construire une ontologie autour des failles logiques et le parcours d'authentification afin de capturer les exigences de sécurité.

Langage de spécification

Ce domaine de recherche discute des méthodes de modélisation fondées sur *UML* et des sémantiques pouvant spécifier les parcours d'authentification. Ces approches

sont à la fois des outils de modélisation et de spécification des exigences. C'est en ce sens qu'elles sont connexes à nos travaux.

Méthodes de test et détection automatique de failles logiques

Ce domaine de recherche discute des méthodes de test et de vérification qui se focalisent à la détection des failles logiques des applications Web. Les méthodes discutées dans cette section sont complémentaires à notre approche. Alors que, nous focalisons à limiter les failles logiques dès la phase de conception, ces dernières opèrent sur des applications en cours de déploiement ou déjà déployées.

Ingénierie des exigences

Cette thèse propose des exigences pour prendre en considération les défauts lors des phases de spécification des parcours d'authentification. Ainsi, nous nous appuyons sur l'ingénierie des exigences pour améliorer la sécurité des parcours d'authentification. C'est dans ce contexte que nous présentons, dans un premier temps, les concepts fondamentaux de l'ingénierie des exigences. Ensuite, nous discutons des études ayant proposé des exigences pour prévenir les défauts logiques et sensibiliser les concepteurs.

Impact des failles logiques dans la vie privée

Ce domaine de recherche discute des impacts négatifs de certaines pratiques dans la sécurité et la protection de la vie privée de l'utilisateur final. Particulièrement, nous discutons des pratiques qui sont de possibles sources de défauts logiques.

Méthode d'évaluation du niveau de sécurité des méthodes d'authentification

Ce domaine de recherche discute des travaux ayant présenté des cadres d'évaluation du niveau de sécurité de différents composants du parcours d'authentification. Elles sont connexes à nos travaux car nous proposons un cadre similaire. Cette section permet ainsi de discuter de ces approches.

Démarche de l'état de l'art

Le terme parcours d'authentification n'est pas un terme générique. Pour identifier les études se rapportant le plus à ce concept, nous avons utilisé des mots-clés du type *authentication mecanisms*, *authentication procedure* ou *authentication workflow*. Ensuite nous avons associé ces mots-clés avec le terme *logic flaws* ou *security flaws* pour designer les failles logiques. Cependant les résultats obtenus ne sont pas

satisfaisants. La plupart des résultats sont des études proposant des mécanismes et protocoles d'authentification ne considérant pas le concept de failles logiques.

Pour trouver des travaux visant à éviter les failles logiques, nous avons orienté nos recherches vers les thématiques cités plus haut en utilisant les mots clés adéquats (*e.g.*, *taxonomie of logic flaws*, *testing logic flaws*, *requirements*). Ensuite, nous regroupons ces papiers par thème en se référant sur les titres, les résumés et les autres sections telles que les conclusions et les travaux connexes. Puis, après une étude approfondie, nous sélectionnons les papiers complémentaires et/ou connexes à nos travaux.

Enfin, d'autres études dont les recommandations, les groupes d'études, les langages autour d'UML et les livres cités dans cette état de l'art ont été ciblées car indispensables à la compréhension des approches de cette thèse.

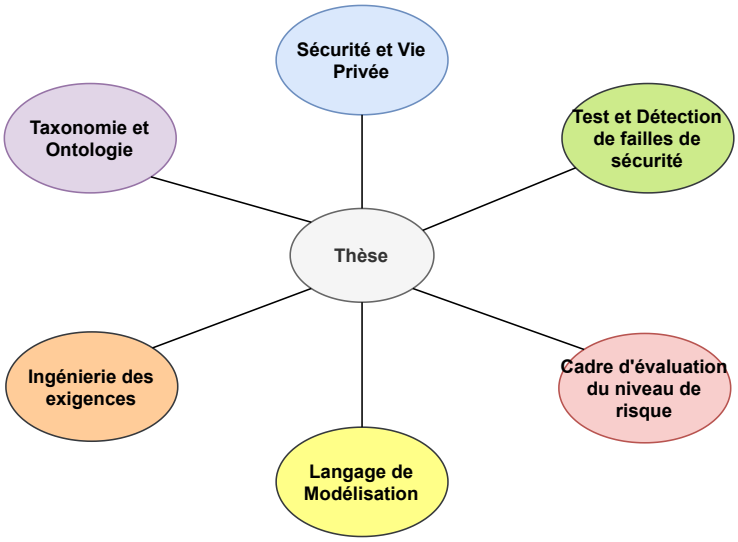

Figure 2.3 – Vue d'ensemble des différents domaines de l'état de l'art.

2.2.1 Taxonomie et ontologies des failles de sécurité

Fondamentalement, les défauts ou failles de sécurité sont classés en deux catégories : les défauts classiques de sécurité qui sont dus à des erreurs de spécification et/ou d'exigences de sécurité et les défauts d'implémentation ou *bug* ou *fault* en anglais qui sont dus à des erreurs sur l'implémentation d'une spécification ou des exigences [91, 171]. Le terme faille de sécurité (*flaws* en anglais) doit être aussi différencié de vulnérabilité ou de faiblesse (*i.e.,vulnerability* ou *weakness*). En effet, ces dernières sont des conséquences d'exploitation de défauts ou de manquements dans le programme qui peuvent être exploitées sous certaines conditions [156, 171]. Par exemple une attaque par injection SQL exploite une vulnérabilité communément appelée faille SQL [149]. Cette vulnérabilité peut être caractérisée de deux manières :

d'une part, elle est causée par un défaut de programmation dans l'assainissement des variables d'entrées (*i.e.*, *sanitization* en anglais), d'autre part, elle est causée par la spécification qui préconise l'usage d'une version obsolète du système de gestion de base de donnée (SGBD), même si, fondamentalement, la cause d'une faille SQL est toujours due à l'implémentation [86]. Ainsi, d'un point de vue à l'autre, la caractérisation de la faille SQL ou de l'attaque peut différer. C'est en ce sens que des études ont proposé des taxonomies dont le but est de caractériser les attaques et vulnérabilités en fonction de caractéristiques pertinentes pour guider l'analyse des risques [94].

Parmi ces taxonomies, on peut distinguer celles qui caractérisent les attaques dont la taxonomie de Mirkovic *et al.* [127] sur les attaques par déni de service (Dos), Sadehian *et al.* [149] sur les attaques par injection SQL et les travaux de Gupta *et al.* [84] sur les attaques XSS (injection de code malicieuse). D'autres études ont cherché à caractériser les failles et les vulnérabilités sur les systèmes informatiques. McPhee [125] est l'un des précurseurs à s'intéresser à ce sujet en 1974. Dans son papier, l'auteur propose des classes de défauts (*e.g.*, identifiant non unique, violation des systèmes de stockage) qui peuvent nuire à l'intégrité des systèmes d'exploitation. D'autres études ont suivi dont Landwehr *et al.* [109] qui propose de caractériser les failles selon trois dimensions : le comment (*i.e.*, *by genesis*), le quand (*i.e.*, *by time of introduction*) et le où (*i.e.*, *by location*). Selon cette étude, une faille de sécurité est au moins caractérisée par l'une de ces trois dimensions, sachant que "le comment" fait allusion à l'intentionnel ou l'inadvertance de la faille ; "le quand" se référant au moment d'introduction de la faille (*i.e.*, phase de développement, de maintenance ou d'exécution) ; enfin "le où" se reporte à sa localisation soit au niveau logiciel ou au niveau matériel. Chacune de ces trois dimensions comporte des sous-catégories afin d'affiner la caractérisation de la faille.

Cette taxonomie de Landwehr *et al.* [109] a influencé beaucoup d'autres travaux dans ce domaine notamment Jiwnani *et al.* [100] qui caractérisent les défauts de sécurité sur une matrice à deux dimensions : l'impact et la cause. D'autres études dans ce domaine peuvent être trouvées dont les travaux de Bishop [18] et Igure *et al.* [94] où un ensemble de taxonomies d'attaques et de vulnérabilités y sont décrites.

Il existe aussi des consortiums et groupes d'études spécialisés dans la classification et le suivi des défauts de sécurité (incluant failles, vulnérabilités, *bugs* et attaques). Parmi eux, on peut citer OWASP [73] qui, à notre connaissance, propose la plus large collection de défauts de sécurité relatifs aux domaines applicatifs. De plus, elle publie un *Top10* annuel des dix vulnérabilités les plus exploitées par domaine (*e.g.*, web, mobile) [69]. Pour chaque défaut de sécurité répertorié, OWASP propose des contremesures pour y remédier. Elle propose également des pages dédiées organisées en thème (*e.g.*, authentification, autorisation, déni de service) où des exigences y sont préconisées [66, 65].

Le Consortium de la Sécurité des Applications Web (*Web Application Security Consortium - WASC*) propose également une classification des menaces spécifiques aux applications web. Contrairement à OWASP dont il est difficile d'identifier une logique dans leur taxonomie, la *WASC* propose une taxonomie des menaces en fonction de leurs causes : la spécification, le développement ou le déploiement. De plus, ces catégories ne sont pas mutuellement exclusives car une menace peut appartenir à différentes catégories [175]. D'autres organisations similaires existent [36, 55]

Les ontologies sont des modèles de données qui permettent de représenter une base de connaissances sous forme de concepts dans un domaine et capturer les relations entre ces concepts [82]. Les concepts d'une ontologie peuvent être représentés en utilisant des standards de description de données relationnelles dont OWL [41] et RDF [111]. Ainsi, selon ces standards, une ontologie est représentée par des classes et des relations entre ces classes. La combinaison entre les classes via les relations est appelée un triplet, celui-ci est constitué d'un sujet, d'un prédicat et d'un objet. Ainsi, la définition des concepts grâce à ces triplets facilite la compréhension de l'ontologie et son exploration. Par exemple, sur la Figure 2.4, nous avons une ontologie avec deux classes (*e.g.*, Matière et Professeur) liées par la relation *dispense*. Avec les instances de ces classes (*i.e.*, Blouin pour Professeur et Java pour Matière), on peut construire le triplet (*i.e.*, *Blouin dispense Java*) qui permet de savoir que le professeur Blouin dispense la matière Java. On pourrait également rajouter d'autres classes et relations afin de mieux caractériser la matière ou le professeur.

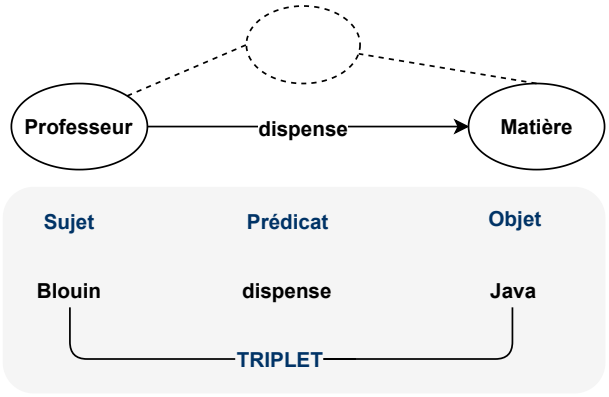

Figure 2.4 – Exemple d'ontologie avec les classes, les relations et les instances de classes

Fondamentalement, la procédure de développement d'une ontologie comprend les étapes de planification, de spécification (*e.g.*, le but et le contexte de l'ontologie), d'acquisition des connaissances (*i.e.*, expert, enquête, taxonomies, analyse de risque), de conceptualisation (*i.e.*, définition des concepts, des axiomes, des attributs et des relations), de formalisation (*e.g.*, OWL [124], RDF [111], DAML-S [8]) , d'intégration et

d'implémentation (*e.g.*, Protégé [92], OntoEdit [162]), d'évaluation (*e.g.*, vérification et validation), documentation et maintenance [60, 102].

Les ontologies sont utilisées dans différents domaines, cependant nous nous intéressons aux études des ontologies qui se focalisent sur la sécurité des applications, notamment sur le parcours d'authentification et sur les risques dont les risques logiques. Ainsi, dans un premier temps, nous nous appuyons sur les études de Blanco *et al.* [19] et Souag *et al.* [158] qui ont révisé et comparé les ontologies en relation avec la sécurité jusqu'en 2008 pour la première et jusqu'en 2012 pour la seconde. Ensuite, nous avons revu en détails les études en relations avec nos travaux en sélectionnant celles qui prennent en considérations entre autres, les concepts suivants : failles logiques et les attaques associées, les exigences de sécurité et les risques liés au parcours d'authentification.

Parmi ces études, on peut citer les travaux de Denker *et al.* [41, 42] qui dans leurs ontologies traitent les concepts de mécanismes d'authentification (*e.g.*, mot de passe, biométrie, certificat) et de mécanismes de sécurité (*e.g.*, protocole, chiffrement, signature) des services web. Ces deux sous-ontologies permettent aux agents de décrire les exigences de sécurité attendues. Ensuite un mécanisme de correspondance (*i.e.*, *matching algorithm*) permet de savoir quels services web satisfont un ensemble d'exigences requises ou une partie des exigences via les capacités préalablement spécifiées dans le *matchmaker*. L'ontologie proposée par Kim *et al.* [106] est composée de 7 ontologies qui prennent en considération différents aspects de la sécurité notamment les concepts généraux (*i.e.*, protocole, contrôle d'accès), les méthodes d'authentification, les algorithmes de chiffrement et les assurances de sécurité. Cette ontologie propose ainsi plus de concepts que Denker *et al.*, et de la même manière, elle propose un algorithme de correspondance qui permet à un demandeur de savoir quels services satisfont les exigences spécifiées à l'aide de ces 7 ontologies. Ces ontologies sont pertinentes sur les concepts de sécurité et des exigences, cependant elles manquent de prendre en considération les menaces (*e.g.*, risques et attaques) ainsi que les failles de sécurité.

Une ontologie des attaques (*e.g.*, attaque XML, attaque par de déni de service, attaque sur les API SOAP) sur les services web est proposée par Vorobiev [174]. Une autre ontologie des attaques par intrusion (*e.g.*, le déni de service, l'homme du milieu, *buffer overflow*) est proposée par Undercoffer *et al.* [167] ; Cette ontologie considère entre autres les failles logiques (*e.g.*, erreur de sérialisation des objets, erreur d'atomicité des variables, de gestion des exceptions). Tsoumas *et al.* [166, 104] proposent des ontologies pour faciliter le déploiement des contrôles de sécurité pour les experts ; leurs ontologies regroupent différents concepts dont les attaques, les menaces, les impacts et les risques sur les biens à protéger.

Après 2012, d'autres études ont proposé des ontologies pour la sécurité des applications et la définition des exigences [160, 143, 159], cependant ces dernières re-

prennent les mêmes concepts fondamentaux discutés plus hauts. Par ailleurs, à notre connaissance, aucun de ces travaux n'a considéré les éléments de contexte et l'utilisateur final comme éventuel concept pouvant conduire à des risques logiques.

Dans la suite de cette section, nous nous focalisons sur les défauts logiques de sécurité qui sont au coeur de notre problématique. Nous nous appuyons sur les taxonomies et les classes de défauts sus-mentionnées pour déterminer les classes de défauts dues à la spécification que nous caractérisons de défauts logiques de conception.

2.2.2 Défauts logiques de conception

Un défaut logique de sécurité (*i.e.*, *business logic flaws* ou *logic flaws* en anglais) est défini comme étant la conséquence du contournement de l'usage légitime d'un service pour un but malicieux [72]. Pour illustrer ce type d'erreur, considérons l'exemple d'un contournement de la navigation requise d'une application de vente en ligne : le but de l'attaquant étant d'effectuer une commande sans payer le montant dû. Sur la figure 2.5 on constate que le chemin de navigation prévisionnel pour effectuer une commande est la séquence en noir (1 à 5) ; un utilisateur malicieux qui connaît les URLs de chaque étape de cette séquence, pourrait passer de l'étape 3 à l'étape 5. Ainsi, si les contrôles nécessaires ne sont pas correctement réalisés, alors celui-ci pourrait effectuer un achat sans payer le montant dû. Cette faute est considérée comme logique du fait de son caractère à n'utiliser que le comportement légitime de l'application sans recourir à des techniques d'attaques exploitant les failles et vulnérabilités connues. Ceci fait la particularité des défauts logiques qui nécessitent d'abord de comprendre le comportement légitime d'une application afin de pouvoir l'exploiter [80]. C'est aussi pour cette raison que les défauts logiques sont difficiles à caractériser et donc à être détectés par les outils automatiques de test et de détection de vulnérabilités [171]. Ci-après quelques exemples de défauts logiques regroupés par catégorie.

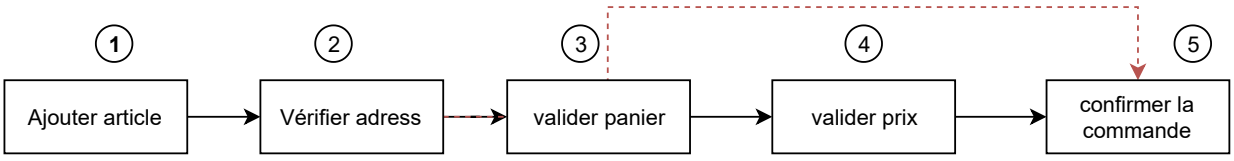

Figure 2.5 – Exemple illustratif du contournement de chemin de navigation

Contournement de chemin de navigation

Le contournement de chemin de navigation correspond à l'exemple illustré dans la Figure 2.5. Cette déviation du chemin de navigation initial peut viser tous les composants d'un système d'information, notamment le parcours d'authentification. Les causes d'une telle attaque sont diverses et variées. On peut citer une insuffisance de

vérification dans le système d'authentification comme un manque de renforcement de certains URLs dont les "URLs morts" (*i.e.*, *dead links*) qui ne sont pas répertoriés dans l'application mais accessibles directement via la barre d'URL [36, 91]. Certaines applications web autorisent des *cookies* de session de longue durée, cette pratique peut également conduire au contournement de chemin de navigation du point de vue d'un attaquant en cas de vol de ce cookie. En effet, si la vérification de la validité de ce cookie n'est pas réalisée correctement, un attaquant pourrait se connecter à la place de l'utilisateur légitime sans passer par le formulaire d'authentification (*i.e.*, fixation de session [71], vol de session [67]). Les URLs prévisibles peuvent aussi être source de contournement de chemin de navigation ; en effet, par brute force, un attaquant ayant connaissance du format de l'URL de changement de mot de passe, pourrait changer le mot de passe d'un utilisateur à son insu [80].

Usage de chemin faible

Un chemin faible existe lorsque l'accès à une ressource ou à un service est possible via plusieurs chemins de navigation dont les niveaux de sécurité sont différents. L'exemple le plus pertinent pour illustrer cette attaque est le cas d'un abus de changement de mot de passe [55]. Les applications actuelles ont tendance à imposer beaucoup de contraintes sur la création de mot de passe sécurisé ; par ailleurs, en cas d'oubli de celui-ci, le parcours de récupération est utilisé. Par ailleurs, ce parcours de récupération est souvent protégé par une question secrète qui est facilement connaissable [152, 142, 21]. Ce contournement du parcours d'authentification via le parcours de récupération de mot de passe est un bon exemple de combinaison de défauts logiques à des fins malicieuses. Cet exemple illustre parfaitement la difficulté des défauts logiques à être détectés ou caractérisés sans une connaissance parfaite de la logique décisionnelle de l'application ou du service en question [91].

Abus de fonctionnalité

L'abus de fonctionnalité est l'une des attaques logiques que l'on retrouve dans quasiment toutes les taxonomies de défauts logiques que nous avons étudiées [91, 175, 36, 73]. Cette attaque consiste à abuser d'une fonctionnalité à des fins malicieuses. Grossman *et al.* [80], dans leur papier sur les 7 défauts logiques à éviter, l'illustrent via différents exemples dont celui d'un site internet de ventes aux enchères. En effet l'abus consiste à bloquer le compte des autres enchérisseurs en testant plus de 5 fois leurs identifiants avec un mot de passe erroné. Ce qui a pour conséquence de bloquer le compte des victimes pendant 1h, les empêchant ainsi d'enchérir. Ainsi, le mécanisme utilisé pour lutter contre les attaques par force brute (*i.e.*, bloquer l'utilisateur après quelques tentatives) est détourné à des fins malicieuses (*i.e.*, bloquer délibérément un utilisateur en l'empechant d'enchérir). D'autres abus de fonctionnalité sont

décrits par Hope *et al.* [91] notamment la surcharge qui conduit aux attaques par déni de service ou la concurrence qui est une autre forme de déni de service empêchant certaines ressources d'être disponibles aux autres utilisateurs.

Divulgation d'informations

La divulgation d'informations ou fuite d'informations utilisateurs n'est pas en soi un défaut de sécurité, mais contribue à l'élaboration d'une attaque logique. Par exemple, dans l'illustration de la section précédente, l'attaque a été facilitée par le fait que les identifiants des enchérisseurs étaient visibles pour tous les utilisateurs du système, ce qui a permis à l'attaquant de connaître les identifiants de ces victimes. Une autre forme de fuite d'informations est la mauvaise gestion des messages d'erreurs, ceux-ci peuvent être des sources d'informations capitales pour un attaquant (*e.g.*, les messages d'erreurs sur les formulaires d'authentification). En effet, il est commun de voir les messages d'erreurs du type "identifiant non valide" ou "mot de passe non valide", ces types de messages permettent à un attaquant de se focaliser que sur ce qui n'est pas correcte et donc réduire sa surface d'attaque [80, 68]. Échanger des données sensibles sans chiffrement dans un tunnel non sécurisé est aussi considéré comme une fuite d'informations, car facilite l'attaque par l'homme du milieu (*i.e.*, *man-in-the-middle attack*) en transmettant les informations sensibles en clair [70, 29, 178].

Synthèse

Cette section discute des défauts logiques de sécurité. La liste présentée n'est pas exhaustive : d'autre défauts de sécurité peuvent aussi être considérés de défauts logiques, notamment les augmentations de privilèges et la prédiction de variable [91]. Parmi les exemples illustrés, on a pu constater que certains de ces défauts peuvent être causés par des erreurs d'implémentation comme la divulgation d'information via une mauvaise gestion des messages d'erreurs. Et d'autres sont dus à la spécification : un parcours de récupération de mot de passe faible. Nous nous sommes focalisés sur les défauts logiques de conception en les adaptant au parcours d'authentification.

2.2.3 Langage de modélisation

Le langage de modélisation UML (*i.e.*, *Unified Modeling Language [81]*) est très utilisé dans le domaine du développement logiciel. Il propose différents diagrammes pour couvrir un grand nombre d'aspects du processus de développement logiciel. Parmi eux, nous pouvons citer les diagrammes qui permettent de modéliser le comportement statique d'une application : les diagrammes de classes et d'objets, et ceux permettant de modéliser les interactions et le comportement dynamique : diagramme de séquences, diagramme état-transition et diagramme d'activité. En ce qui concerne le

parcours d'authentification, deux aspects d'UML nous intéressent particulièrement : sa capacité à décrire le parcours d'authentification et ses exigences de sécurité, mais aussi sa capacité à évaluer la sécurité de ce parcours.

Dans cette perspective, le diagramme état-transition permet de décrire le parcours d'authentification en terme de séquence de navigation et d'interaction. Cependant, celui-ci n'est pas assez explicite pour décrire toutes les exigences (*e.g.*, le moyen d'authentification, les différents composants du parcours). En utilisant le langage OCL [147], certaines exigences de sécurité peuvent être spécifiées sous formes de contraintes sur les objets : Par exemple la limite du nombre de tentatives pour ouvrir une session. Par contre, de telles contraintes sont statiques et doivent être testées durant la phase de développement. De plus nous n'avons pas de garantie du respect ou non de ces exigences lors des phases de spécification.

Pour prendre en considération l'aspect sécurité dans les spécifications UML, *UMLSec* et *SecureUML* ont été introduits, respectivement, par Jurjens [103] et Lodderstedt *et al.* [119] en 2002. Le premier se fonde sur RBAC [150] et permet d'intégrer les spécifications de contrôle d'accès dans la spécification UML. Tandis que, le second permet de spécifier davantage de propriétés de sécurité (*e.g.*, intégrité, confidentialité, disponibilité) en utilisant des stéréotypes, des *tags* et des contraintes. Bien que ces approches soient générales et riches, elles ne permettent pas de prendre en considération les défauts logiques de sécurité. De plus, elles manquent d'outillages pour permettre de vérifier la conformité des exigences de sécurité avec leur spécification.

D'autres profiles UML se sont focalisés sur l'aspect conceptuel et la structure de navigation des applications web. Parmi eux, on peut citer *IFML* (*Interaction Flow Modeling Language*) [25] qui est une extension de *WebML* [31] pour couvrir plus de domaines applicatifs. Il permet ainsi de décrire le parcours navigationnel, les interactions utilisateurs ainsi que les différentes interfaces graphiques de la même application en fonction du conteneur (*e.g.*, web, mobile). Tandis que *IFML* tend à être générique, *UWE* (*UML-Based Web Engineering*) [107] se focalise exclusivement sur la spécification des applications Web. Il permet ainsi de spécifier le contenu statique à l'aide de diagramme de classes, le parcours de navigation grâce aux classes de navigation entre les différents *hyperliens* puis le processus métier de l'application en question.

En plus de ces deux profiles UML, d'autres langages [14, 154, 90, 40, 163, 39, 49] ont proposé des approches similaires pour modéliser les structures de données, la structure de navigation et les aspects de l'interface utilisateur (*e.g.*, évènements, actions, interactions) des applications web. Des travaux ont proposé des sémantiques formelles [115] pour formaliser des spécifications fondés sur *UML-Statechart* sans pour autant fournir d'outils de vérification. Une approche est proposée pour la vérification de diagramme de classes *UML* fondée sur *OCL* [28]. Par contre, celle-ci se focalise sur la vérification de la cohérence du modèle de données. Bien que nous ayons

exploré le domaine de la vérification par modèle, celui-ci ne rentre pas dans le cadre de cette thèse. En effet, la vérification par modèle est pertinente pour les systèmes réactifs et pour la vérification de propriétés (*e.g.*, sûreté, vivacité, terminaison) qui ne sont pas abordées dans cette thèse [11]. Par ailleurs, d'autres langages de description d'interfaces utilisateurs et des tâches utilisateurs existent [101, 116, 130, 135]. Ces derniers peuvent permettre de décrire la structure de navigation d'un parcours d'authentification, mais ne permettent pas de spécifier les exigences liées aux défauts logiques de sécurité.

Tous ces travaux ont contribué à améliorer l'écosystème de développement d'applications Web fondées sur des modèles. Cependant, pour la plupart d'entre eux, la sécurité n'est pas mise en avant. Considérer les défauts logiques dans ce contexte est également difficile car elle nécessite une connaissance précise de la logique de l'application, mais aussi de l'environnement d'exécution de celle-ci. De plus, ces langages ne sont pas assez explicites pour spécifier de tels concepts. En effet, les défauts logiques sont intrinsèquement liés au monde réel, ils nécessitent d'être identifiés et analysés pour en déduire les exigences de sécurité correspondantes.

2.2.4 Ingénierie des exigences

L'ingénierie des exigences est définie par Zave [183] comme étant la branche qui s'intéresse aux objectifs, aux fonctions et aux contraintes du cycle de vie réel des systèmes informatiques. Elle s'intéresse également aux relations qui existent entre ces facteurs afin de préciser la spécification du comportement et de l'évolution du système. En ce sens, l'ingénierie des exigences est interdisciplinaire (*e.g.*, linguistique, sociologie), informelle (pas tout le temps) et idéalement doit faire intervenir toutes les parties prenantes (*e.g.*, concepteurs, développeurs, clients). Cette définition permet aussi de constater que les exigences doivent prendre en considération le monde réel afin de décrire, le plus précisément possible, ce qui est attendu du système [131]. Fondamentalement, l'ingénierie des exigences comprend : les exigences fonctionnelles qui définissent le comportement attendu d'un système et les exigences non-fonctionnelles qui s'intéresse à d'autres aspects du système (*e.g.*, disponibilité, sécurité, performance) [77, 85]. Dans cette thèse, nous nous focalisons sur les exigences non-fonctionnelles, et particulièrement, sur les exigences de sécurité.

Il existe plusieurs étapes dans l'ingénierie des exigences, cependant, celles-ci peuvent se réduire en un processus de trois étapes fondamentales [57] : élicitation (*i.e.*, capture), spécification et validation des exigences.

Méthodes d'élicitation des exigences

L'élicitation des exigences (*i.e.*, ou capture des exigences) consiste à recueillir l'ensemble des informations nécessaires pour définir les exigences pertinentes pour

l'accomplissement des objectifs attendus [57]. Plusieurs papiers sont proposés pour adopter la meilleure technique de capture des exigences [78, 85, 128, 157]. Ci-après quelques techniques d'élicitation d'exigences.

- Questionnaires et enquêtes : Ces techniques consistent à préparer la documentation avec l'ensemble des questions nécessaires pour capturer les exigences. Les questions doivent être concises avec un choix limité de réponses possibles. Les questions peuvent proposer des réponses à cocher.
- Cas d'usages et d'abus d'usages : Cette méthode est aussi utilisée dans la définition des exigences en utilisant les diagrammes de cas d'usage pour décrire la manière dont le système doit être utilisé [133]. En décrivant comment le système doit être utilisé, on peut en déduire les exigences à respecter par le système. De plus, on peut aussi décrire comment le système ne devrait pas être utilisé : abus d'usage (*i.e.*, *misuse case* pour capturer des exigences [136, 157].
- Représentations graphiques : Cette méthode est souvent utilisée par les équipes non techniques telles que les concepteurs des graphismes (*i.e.*, *designer*), les ergonomes et les équipes de *marketing*. Elle consiste à utiliser tout outil de représentation graphique (*e.g.*, dessin, graphisme, animation) permettant de décrire le comportement attendu du système et donc de capturer les exigences nécessaires [132].
- Les analyses d'applications similaires : La réutilisabilité d'exigences entre deux applications peut être source d'erreurs. Cependant, cette méthode n'est pas totalement à prohiber, en effet une bonne analyse d'applications similaires peut aider à capturer des exigences qui peuvent être réutilisées [63, 131].
- Les ontologies : Des méthodes d'élicitation des exigences fondées sur les ontologies ont été largement utilisées telles que discutées dans la Section 3.2.3.

La technique de capture des exigences que nous adoptons est une combinaison de différentes techniques. En effet, nos exigences sont dictées par nos objectifs, en l'occurrence la prise en charge des défauts logiques sur les parcours d'authentification dès la phase de conception. De ce fait, notre approche est avant tout orientée vers la découverte et la caractérisation des défauts logiques. Ainsi, c'est en combinant les méthodes d'analyse des risques d'applications similaires (*i.e.*, via une étude empirique), d'enquête avec les parties prenantes (*i.e.*, enquêtes aux niveaux des concepteurs et développeurs) pour identifier les failles existantes, et l'abus d'usage (*i.e.*, exploitation de défauts logiques sur des applications grand public) que nos exigences ont été capturées.

Méthodes de spécification des exigences.

Différentes méthodes de spécification des exigences existent [57, 165]. Parmi elles, les méthodes informelles fondées sur le langage naturel. Ces méthode consiste à spécifier les exigences en langage compréhensible par l'humain sans règle précise. Quand

celle-ci est structurée, on parle alors de méthodes fondées sur des *templates* [51, 63]. Cette dernière est moins ambiguë car fondée sur une terminologie connue des parties prenantes. Cependant, si elle est très contraignante, certaines exigences risquent ne pas pouvoir être spécifiées.

À coté de ces méthodes informelles, il y a celles fondées sur des sémantiques formelles [11, 33, 45, 105, 140, 164].

Ces techniques de spécification et de vérification fondé sur les modèles sont sans ambiguïtées et permettent une vérification automatique du respect ou non des exigences de sécurité. Malheureusement, ces avantages très convoités dans le domaine des systèmes critiques n'est pas très mis en avant dans le monde du développement logiciel. En effet, ces méthodes sont coûteuses en développement (*e.g.*, temps, compétence), moins *scalable* (*e.g.*, spécifiques et difficilement évolutives) et souffre du problème de l'explosion combinatoire du nombre d'états à explorer [34]. C'est entre autres ces raisons que ces méthodes sont peu utilisées dans les processus de développement logiciels qui favorisent des méthodes telles que l'agilité [155] et le DevOps [52]. Ces dernières favorisent la communication entre les parti-prenantes et des cycles de développement rapides principalement fondés sur les méthodes de test des fonctionnalités. Enfin, en plus des méthodes fondées sur *UML* discuté dans la Section 2.2.3, nous avons aussi celles fondées sur les "modèles de procédé d'affaire et notation" (*i.e.*, BPMN [132, 137, 148]) et celles fondées sur cas d'usages et d'abus [157, 63, 136]).

Méthodes de validation des exigences

La validation des exigences est l'étape où l'on vérifie et valide les exigences. La méthode utilisée peut dépendre de la méthode de spécification préalablement adoptée. La vérification formelle permet de valider des exigences en usant de techniques de vérification par modèle (*i.e.*, *model checking* [11, 33, 105, 110, 115, 126]). Les méthodes fondées sur UML peuvent être vérifiées et validées en utilisant *OCL* ou via des sémantiques formelles [28, 51, 133]. La matrice de traçabilité [51], qui consiste à comparer les objectifs de l'application par rapport aux exigences est aussi utilisée pour vérifier et valider des exigences.

La validation des exigences peut se faire à un stade de prototypage en développant des PoCs (*Proof of Concepts*) qui permettent de vérifier la satisfiabilité des exigences avant de se lancer dans des phases de développement effectif, l'audit, les *checklists* et l'analyse de risque sont aussi des méthodes utilisées dans la vérification et la validation des exigences [57].

2.2.5 Test et détection automatique de failles logiques

Différentes études ont contribué à lutter contre les défauts logiques de sécurité dans les applications web. Parce qu'elles sont spécifiques et difficilement caractérisables [171], la plupart des études se sont focalisées sur des méthodes de test et de détection automatique de ces défauts de sécurité. Parmi ces études, on peut en distinguer deux catégories : les méthodes boites blanches (*i.e.,white-box*) et boites noires (*i.e., black-box*). La différence entre les deux est que la première a accès au code source tandis que la seconde n'en a pas accès.

Dans la première catégorie des méthodes ayant accès au code source, Cova *et al.* [35] présente *Swaddler* un outil de détection de déviations du parcours initialement prévu d'une application web (*i.e., workflow violation attack*). Leur approche décrit l'état d'une application web en se fondant sur les variables de session. En effet, la valeur attendue de ces variables varie d'un contexte d'exécution de l'application à l'autre, cependant, pour le même contexte, la valeur de celle-ci est toujours la même malgré qu'elles puissent être transmises au serveur via différents moyens (*i.e.,*formulaire, cookies ou la barre d'URL) : ce sont des *likely invariant* identifiés grâce à l'outil Daikon [56]. Durant la phase de déploiement, les états légitimes de l'application sont construits à l'aide des invariants en instrumentant le code de l'application. Ainsi, en monitorant les états de l'application en exécution par comparaison des invariants attendus et détectés, *Swaddler* est capable de détecter les violations des états légitimes et donc les défauts logiques.

Mimosa [12] est une approche similaire à *Swaddler* qui permet de détecter des vulnérabilités exploitées via différents modules de la même application (*i.e., multi-module vulnerabilities*). Leur approche permet de détecter les défauts liés aux données d'entrées telles que les failles XSS et SQL, mais aussi les violations de logique de l'application. Pour détecter les failles logiques de contrôle, Mimosa se fonde sur un algorithme d'exploration de chemin similaire à la vérification de modèle. Après la création d'un graphe d'états de l'application, l'outil parcourt ce graphe et à chaque étape il teste s'il est possible d'atteindre un état qu'il ne devrait pas être possible d'atteindre. Une des limites de leur approche est qu'une deuxième occurrence d'erreur sur la même trace d'exécution ne pourra pas être détectée car l'algorithme s'arrête à la première détection d'erreur. Il faudra ainsi corriger la première erreur pour pouvoir détecter la seconde. De plus, comme la plupart des approches de vérification de modèle, leur approche souffre de la problématique de l'explosion combinatoire [11] du nombre d'états à explorer. Cependant, d'après leurs observations, la plupart des vulnérabilités sont identifiées en moins de 5 étapes.

Waler [59] est une extension de *Swaddler* et de *Mimosa* pour détecter plusieurs types de vulnérabilités, notamment les failles logiques. Tandis que les deux premières sont implémentées pour des applications développées en PHP, *Waler* se focalise sur

des applications développées en Java, particulièrement en *JSP* [62]. Une étude similaire à *Waler* permet de détecter les failles de sauts de privilège sur des applications web développées en Java [58] également.

À côté de ces méthodes statiques de détection de défauts logiques qui nécessite le code source de l'application, d'autres études ce sont focalisées sur la détection des défaut logiques de sécurité sans avoir accès au code source. Dans ce contexte, *Block* [113] est une approche boite noire similaire à *Swaddler*, mais se focalisant exclusivement sur toute erreur pouvant conduire à contourner le parcours d'authentification. Une approche fondée sur l'inférence de modèle est proposée par Pelegrino *et al.* [138]. Leur approche permet de générer les suites de test suivant 4 modèles d'attaques dont le saut d'opération ou le contournement de séquences de contrôle qui sont considérées comme des failles logiques. Leur approche est semi-automatique car nécessitant l'intervention de l'humain pour identifier les évènements à monitorer et spécifier les propriétés à vérifier en LTL [140] par l'*oracle*. Leur approche est validée sur différents sites de e-commerce et a permis d'identifier des failles logiques telles que la fixation de session ou le contournement de contrôle pour payer moins que ce que l'on devrait payer.

Di Lucca *et al.* [44] propose une méthode de test de détection des erreurs de navigation d'une application web en considérant les interactions avec les boutons de navigation du navigateur web. D'autres études ont proposé des approches similaires sans prendre en considération les interactions avec les boutons de navigation du navigateur [117, 146].

Dans cette section, nous avons restreint notre analyse aux méthodes de test et de détection automatique de failles logiques. Par ailleurs, d'autres études se sont focalisées plus généralement sur la détection de failles et de vulnérabilités sur les applications web [24, 30, 114]. Ces études ne sont pas détaillées car nous nous focalisons uniquement sur les approches en relation avec notre problématique, en l'occurrence la prise en charge des défauts logiques.

Ainsi, les études présentées dans cette section sont complémentaires à nos travaux car elles se focalisent sur les applications en phase de développement, alors que notre démarche est plutôt orientée vers la phase de conception. Notre approche vise à éviter les défauts logiques dès la phase de conception. De plus la plupart de ces études se focalisent sur des failles spécifiques, tandis que notre objectif est d'abord, de sensibiliser les concepteurs sur un large panel de défauts logiques susceptibles de mettre leurs applications en danger, puis de proposer des méthodes pour y remédier.

2.2.6 Failles logiques dans la sécurité et la vie privée

Les études présentées ci-après contribuent à renforcer un autre aspect de défauts logiques que nous considérons dans cette thèse, en l'occurrence les menaces d'attaques logiques issues d'éléments de contexte extérieurs à la spécification initiale. Dans cette thèse, nous prétendons que certaines situations doivent être prises en compte dès la phase de conception du fait de leurs impacts dans la sécurité et la vie privée de l'utilisateur final.

Dans ce contexte, plusieurs études peuvent être référencées. Par exemple, en considérant la faille de chemin de navigation faible décrite en Section 2.2.2, une étude de Florencio *et al.* [64] montre que 15% des utilisateurs de Yahoo effectuent une récupération de mot de passe par mois. Une autre étude [22] montre que sur 138 applications observées, 92% d'entre elles se fondent sur l'adresse mail [76] sachant que seule 12% de ces applications exigent de répondre à une question secrète avant de pouvoir récupérer son mot de passe. Partant de ce constat, il est évident que le parcours de récupération de mot de passe est moins sécurisé que le parcours de *login* car des études montrent que le mot de passe est plus sécurisé que la question secrète qui est facilement devinable [21, 142, 152]. Par ailleurs, ce parcours de récupération serait en effet sécurisé s'il est garanti que la boite mail est sécurisée et accessible que par l'utilisateur légitime. Nous prétendons que cette hypothèse de sécurité est faible du fait de la plasticité des applications (*i.e.*, applications qui peuvent s'exécuter en parallèle sur différents terminaux), des mécanismes de sessions persistantes qui font que les applications ne se déconnectent jamais. Une étude de Symantec [177] montre que sur 50 téléphones perdus intentionnellement, 96% des trouveurs essaient d'accéder aux données personnelles. Par conséquent assumer certaines hypothèses de sécurité peut faciliter ce type d'attaques logiques. De plus, certains utilisateurs ne se préoccupent pas de verrouiller leur téléphone [168] et qu'il est prouvé que les données personnelles sont accessibles via des applications sensibles telles la boîte mail [53].

Toutes les études présentées dans cette section sont très alarmantes et justifient la nécessité de prendre en considération ces situations dans la conception des parcours d'authentification. C'est pour aider à anticiper ces situations dès la phase de conception que nous avons étendu la notion de faille logique afin de proposer des exigences qui aident à les prendre en compte.

2.2.7 Évaluation du niveau de sécurité

Nos travaux sont aussi relatifs à un corpus d'études d'évaluation du niveau de sécurité des applications. Parmi ces études, les méthodes classiques de gestion et d'analyse de risque [7, 9, 47] qui, fondamentalement, proposent les étapes ci-après :

— identification et analyse des risques ;

— évaluation des coûts et de la vraisemblance des attaques résultantes de ces risques ;
— sélection des mesures appropriées ;
— évaluation des coûts des contre-mesures ;
— comparaison des coûts et la criticité des risques afin de prendre des décisions (*e.g.*, accepter le risque, gérer le risque, déléguer le risque).

Cette Section discute de l'évaluation des risques. Les autres points (*e.g.*, évaluations des coûts, criticité) ne sont pas traités dans le cadre de cette thèse car les mesures et les contre-mesures appropriées, ainsi que les coûts de prises en charge des risques sont sensibles au contexte de chaque application.

L'ITU [95] (*i.e.*, Union International des Télécommunications) propose un cadre d'évaluation du niveau de sécurité relatif à la gestion des identités numériques. Le guide propose 4 niveaux d'assurance (*i.e.*, *Level of Assurance*) pour la phase d'enregistrement (y compris, la vérification des attributs d'identité, le stockage et l'émission) et pour la phase d'authentification. En résumé :

— **LoA1** est requis pour un service exigeant peu de confiance sur l'identité déclarée par l'entité. L'exigence sur l'identité est que celle-ci doit être unique dans le contexte de l'application.
— **LoA2** est requis pour un service exigeant un niveau minimal de confiance sur l'identité déclarée par l'entité. En plus d'être unique, l'identité à laquelle celle-ci se rapporte doit objectivement exister. Autrement dit, le contrôle mis en place doit se fier à une identité fournie par une autorité de confiance.
— **LoA3** est requis pour un service nécessitant une grande confiance sur l'identité déclarée ou affirmée de l'entité. En plus des contrôles en *LoA1* et *LoA2*, les informations d'identités doivent être vérifiées localement ou à distance.
— **LoA4** requiert le même niveau de confiance que le niveau LoA3. Cependant, les contrôles d'identités doivent être réalisés physiquement afin de confirmer l'identité de l'entité.

Le cadre EAAF [95] (*i.e.*, *Entity Authentification Assurance Framework*) décrit ainsi un ensemble de contrôles à effectuer pour éviter les menaces sur chaque phase de la gestion des identités. La table 2.1 présente une liste non exhaustive des menaces auxquelles les contrôles adéquats doivent être appliqués pour atteindre le niveau d'assurance requis. Par exemple, pour éviter l'usurpation d'identité sur un service ne nécessitant pas de confiance sur l'identité de la personne, une vérification du respect de la politique suffit à lui garantir le niveau *LoA1*. Par ailleurs, si l'identité physique des entités est engagée, alors le contrôle d'identité devra être réalisé en physique pour atteindre le niveau *LoA4*.

L'évaluation est différente pour la partie authentification. En effet, quel que soit le niveau de *LoA* requis, l'ensemble des contrôles contre les menaces sur le mécanisme d'authentification sont exigés. On notera que, pour obtenir le niveau de *LoA3* ou *LoA4*,

l'authentification multi-facteurs est requise car celle-ci permet de lutter contre la plupart des menaces. La liste complète des menaces et contrôles est disponible dans la recommandation (*i.e.*, nous différencions les numérotations des contrôles avec le "a" devant les contrôles sur l'authentification ; les contrôles approrpiées sont disponibles *EAAF* [95]).

Menaces	Contrôles		Contrôles Requis			
	Enregistrement		LoA1	LoA2	LoA3	LoA4
Usurpation d'identité	Preuve d'identité : politique d'adhésion		#1	#1	#1	#1
	Preuve d'identité : Vérification en Personne					#2
Falsification d'identité	Protection des dispositifs physiques		#5	#5	#5	#5
Duplication d'identité	Sécurisation des lieux de stockages des attributs d'identité		#12	#13	#14	#15
Répudiation d'identité	Sauvegarde des historiques de d'enregistrement		#20	#20	#21	#21
	Authentification	LoA*	LoA1	LoA2	LoA3	LoA4
Menaces générales	Authentification multi-facteurs				#a1	#a1
deviner les preuves de sécurité	Mot de passe solide, mécanisme de blocage	#a2,#a3				
Re jeu d'authentification	Code à usage unique, horodatage	#a13,#a14				
Hameçonnage	Système anti-hameçonnage, authentification mutuelle	#a8,#a9,#10				
Attaque Homme du milieu	Authentification mutuel, utilisation de session chiffré	#a10,#a16				

Table 2.1 – Liste non exhaustive de menace sur les phases d'enregistrement et d'authentification ainsi que les contrôles (*i.e.*, #1 à #20) appropriés pour atteindre le niveau d'assurance requis.

Le Nist [26] propose un cadre de gestion d'identités similaire à la recommandation présentée ci-dessus, celui-ci propose également 4 niveaux d'assurance pour les procédures d'enrôlement et d'authentification. Cependant, ce cadre a évolué depuis Juin 2017 et propose désormais 3 niveaux de sécurité par processus [79] :

- **IAL** (*Identity Assurance Level*) : le niveau d'assurance du mécanisme de vérification des identités ;
- **AAL** (*Authenticator Assurance Level*) : le niveau d'assurance du mécanisme d'authentification ;
- **FAL** (*Federation Assurance Level*) : le niveau d'assurance de la fédération d'identité.

La table 2.2 présente le récapitulatif des différents niveaux d'assurance de cette recommandation.

Le choix du *xAL* est dicté par l'analyse des risques et les impacts que l'on souhaite éviter. La recommandation décrit 6 catégories de risques :

Enrôlement	Authentification	Fédération
IAL1 : Les attributs sont auto-assumées par l'utilisateur ou sont considérés comme tels	**AAL1** exige l'usage d'un moyen d'authentification fondé sur un seul facteur à travers un protocole sécurisé	**FAL1** exige que l'assertion d'authentification soit signée par l'IdP
IAL2 : Les attributs d'identité sont vérifiés en personne ou à distance avec le respect à minima de la procédure présentée dans la recommandation	**AAL2** Exige l'usage de mécanisme d'authentification fondé sur deux facteurs d'authentification différents. L'usage de protocole cryptographique éprouvé est aussi exigé	**FAL2** exige que le token d'assertion en plus d'être signé par l'IdP, celui doit être chiffré par la clé publique du SP
IAL3 : Les attributs sont vérifiés en personne par une autorité de confiance en présence de document physique d'identité	**AAL3** nécessite un niveau de confiance supérieur à AAL2, par conséquent exige l'usage d'un protocole cryptographique résistant à l'usurpation du vérifieur (*e.g.*, authentification mutuelle)	**FAL3**, en plus d'exiger FAL2, exige que la possession d'une clé cryptographique par l'utilisateur en relation avec l'assertion et ces artifacts (*e.g.*, un certificat)

Table 2.2 – Récapitulatif des différents niveaux xAL du Nist

1. gêne, détresse, atteinte à la réputation de la compagnie ;
2. perte financière ou engagement responsabilité de la compagnie ;
3. nuisance à la compagnie ou à ses intérêts publiques ;
4. divulgation d'informations sensibles ;
5. atteinte à sûreté des personnes ;
6. violation civile ou criminelle.

Le processus d'analyse de risques se fonde sur la recommandation FIPS [141] qui décrit trois niveaux d'impact : faible, modéré, élevé. Ainsi, chaque catégorie d'impact doit être évaluée à ces trois niveaux. Par exemple, une perte financière est évaluée à "faible" si on observe que ces pertes sont insignifiantes ; "modéré" si les conséquences

sont sérieuses pour la compagnie ou les utilisateurs ; "élevé" si les pertes sont catastrophiques pour l'une ou l'autre des parties prenantes.

La table 2.3 présente le niveau de *xAL* requis en fonction du résultat de l'analyse des risques. Cependant, la nature de ces impacts diffèrent d'un contexte à l'autre, il revient aux concepteurs du service de mener une analyse des risques relatifs à son activité afin de choisir le niveau *xAL* lui permettant de prévenir ces risques.

Catégories d'impact	xAL1	xAL2	xAL3
gêne, détresse, atteinte à la réputation de la compagnie	faible	modéré	élevé
perte financière ou engagement responsabilité de l'agence	faible	modéré	élevé
nuisance à la compagnie ou à ses intérêts publiques	faible/modéré		élevé
divulgation d'informations sensibles	faible/modéré		élevé
atteinte à la sûreté des personne	faible		modéré/élevé
violation civile ou criminelle	faible/modéré		élevé

Table 2.3 – Choix du *xAL* en fonction de l'évaluation du niveau des impacts

Les cadres de gestion d'identités présentés ci-dessus évaluent la sécurité du système en fonction des contrôles mis en place pour le premier et en fonction du niveau de risques des impacts pour le second. La recommandation du Nist considère la criticité dans l'évaluation des impacts (nous verrons plus tard que notre approche se focalise sur la probabilité d'occurrence calculée en fonction de la sémantique que nous proposons pour évaluer le niveau de risques). Ce choix de ne pas considérer la criticité dans notre approche s'explique du fait que la criticité d'un impact dépend du contexte de l'application. Par exemple, l'impact d'un accès non-autorisé dans un compte de magazine en ligne et l'accès non-autorisé dans une application de banque en ligne n'ont pas la même criticité en terme de pertes financières ou d'impact sur la vie privée de la victime. C'est la raison pour laquelle la criticité doit être définie par le concepteur du système. Des cadres similaires existent dont celui de l'Inde et de l'Australie sont décrits par Dubey *et al.* [50].

En 2010, Vapen *et al.* [170] propose d'étendre le cadre d'évaluation du niveau de sécurité proposé par le Nist [27]. Leur approche ne prend pas en compte la partie enregistrement et le vérification de l'identité. Les auteurs considèrent cette étape comme

inadéquate dans le contexte des applications web car au delà du niveau 1, les autres niveaux requièrent une preuve d'identité fournie par une autorité de confiance telle qu'un passeport ou un permis de conduire. Pour eux cette procédure est purement administrative tandis que leur approche se focalise sur la procédure d'authentification via le mobile. Ils proposent ainsi le niveau 2.5 et le 3.5.

— Le niveau 2.5 est similaire au niveau 3. Le niveau 3 exige une protection contre les attaques par l'homme du milieu et l'usurpation du vérifieur en utilisant une authentification multi-facteurs fondée sur le mot de passe ou la biométrie et l'usage d'un Code à Usage Nnique - *One Time Password* (OTP). Le niveau 2.5 exige uniquement la protection contre l'attaque du milieu ou l'usage du mécanisme de protection du smartphone comme un des facteurs d'authentification.
— le niveau 3.5 est similaire au niveau 4 de la recommandation du Nist. Cependant, ce niveau reprend les exigences du niveau 3 en plus de proposer des protections contre les détournements de session (*i.e.*, *session hijacking*) et l'usage d'un dispositif physique inviolable compatible FIPS-140-2 [15]. Au lieu d'un dispositif inviolable, le niveau 3.5 propose l'usage d'une carte SIM [1] ou USIM [2] qui ne sont pas certifiées FIPS-140-2.

En proposant ces deux niveaux supplémentaires, les auteurs sont ainsi capables d'évaluer plus finement les mécanismes d'authentification qui utilisent les caractéristiques du téléphone (*e.g.*, bluetooth, carte sim, nfc). De plus, leur méthode d'évaluation permet de prendre en considération d'autres facteurs tels que l'utilisabilité, la disponibilité et le coût économique en considérant les alternatifs qu'offrent les caractéristiques du téléphone portable.

En dehors des exigences proposées par ces recommandations, une étude de Bonneau *et al.* [20] présente une étude comparative sur plus de dix ans d'implémentations des mécanismes d'authentification dans le domaine du web (*e.g.*, gestionnaire de mot de passe, authentification biométrique, authentification multi-facteurs, proxy, fédération d'authentification). En plus d'évaluer les mécanismes d'authentification en fonction de leur résistance à des critères de sécurité (*e.g.*, usurpation, observation, vol, hameçonnage), l'étude prend aussi en considération des critères d'utilisabilité (*e.g.*, mémorabilité, scalabilité, facilité de récupération en cas de perte) et de deployabilité (*e.g.*, accessibilité, coût, compatibilité avec les navigateurs) de ces mécanismes d'authentification. Leur étude permet ainsi d'évaluer tous les mécanismes d'authentification avec les mêmes critères, ce qui permet à un concepteur de faire le choix du moyen d'authentification le plus pertinent pour son service. Certains de leurs critères dont l'observabilité et l'exposition au vol peuvent être à l'origine d'attaques logiques. Typiquement, le vol d'un carte d'accès sans code de sécurité peut permettre à une

1. `https://fr.wikipedia.org/wiki/Carte\_SIM`
2. `https://fr.wikipedia.org/wiki/Universal\_Subscriber\_Identity\_Module`

personne malicieuse d'accéder à un service ou à un bâtiment protégé par cette carte d'accès.

2.3 Conclusion de l'état de l'art

Dans cette section d'état de l'art, nous avons discuté d'études de domaines différents. Nous avons présenté les études de l'ingénierie des exigences sur lesquelles nous nous appuyons pour définir nos exigences. D'autres approches similaire à la notre ont aussi proposé des exigences pour prévenir et alerter les concepteurs des risques de failles logiques [43, 80, 151].

Dans notre démarche d'étendre les failles logiques dans le monde réel, les études sur l'impact de certaines erreurs humaines ou choix de design ont contribué à consolider notre problématique [1, 53, 64, 177, 168].

Bien que moins d'attentions soient portées aux failles logiques en comparaison des *bugs* de sécurité, certaines ontologies et taxonomies ont cherché à les prendre en considération [36, 41, 68, 73, 167]. D'autre part, des méthodes de détection automatique ont été proposées pour lutter contre ces défauts logiques [35, 44, 113]. Ces méthodes sont ainsi complémentaires à nos approches qui se focalisent sur la phase de conception.

Par ailleurs, des études ont proposé des cadres pour guider à la création d'authentification numérique. Ces cadres ont également proposé des niveaux de sécurité et mis en place des contrôles pour atteindre ces niveaux [26, 79, 95]. Ces études sont inspirantes, complémentaires et connexes à nos travaux. En effet, nous proposons une nouvelle définition du parcours d'authentification et proposons également un cadre d'évaluation du niveau de risques.

Enfin, nous n'avons pas trouvé de DSL qui permet de spécifier un parcours d'authentification et d'évaluer sa sécurité dans la littérature. Cependant, Des études de Ahmad *et al.* [4, 3, 5] proposent un DSL appelé *Relax* qui permet de spécifier et de vérifier des exigences non fonctionnelles dans le domaine des systèmes auto-adaptatifs (*i.e.*, *SAS : Self-Adaptive System*). Leur approche est outillée et permet la vérification formelle des exigences. Par ailleurs, un produit similaire à notre DSL est un outil commercialisé par ForgeRock appelé *Intelligent Authentication* qui permet de spécifier un parcours d'authentification via un éditeur graphique [3]. Celui-ci ne considère pas les failles logiques ni ne propose de méthode d'évaluation des risques.

En résumé, du fait de leurs spécificités, les failles logiques sont étudiées de façon marginale. Elles sont difficiles à être caractérisées, par conséquent sont moins pris en compte par les méthodes de test. Par ailleurs, nous n'avons pas trouvé d'études qui étendent les failles logiques dans le monde réel tel que nous le faisons. De plus

3. `https://www2.forgerock.com/platform/access-management/intelligent-authentication`

aucun des travaux présentés n'a proposé une approche visant à appréhender les failles logiques dès les phases de conception ni proposer un cadre d'évaluation et un outil pour décrire et évaluer la sécurité d'un parcours d'authentification.

Dans la suite de ce document, nous présentons comment nos travaux contribuent à l'amélioration de l'écosystème de la spécification des parcours d'authentification en proposant des moyens de prévenir les failles logiques.

Chapitre 3

Exigences de sécurité pour l'amélioration du parcours d'authentification

Ce chapitre présente la première contribution de cette thèse. Cette contribution apporte des solutions au manque de considération des défauts logiques lors de la spécification des parcours d'authentification.
C'est dans cette optique que nous proposons un ensemble d'exigences de sécurité à respecter par les concepteurs des parcours d'authentification dès la phase de conception.
La première section de ce chapitre présente une étude empirique qui généralise la présence des défauts logiques présentés dans l'exemple illustratif (*cf.* Section 1.1.2) sur des applications grand public.
Les Sections 3.2 et 3.3 présentent, respectivement, la caractérisation des défauts logiques et les exigences que nous préconisons pour y remédier.

3.1 Étude empirique

Cette section décrit une étude empirique sur neuf applications grand public. Le but de cette étude est de vérifier la généralisation des défauts logiques de sécurité discutés dans la Section 1.1.2 afin d'aider à les caractériser. Nous détaillons dans la section suivante les différents critères qui ont guidé le choix des applications étudiées.

3.1.1 Les critères de choix des applications

Plusieurs critères ont permis de choisir les applications de cette étude. Parmi ces critères, les plus importants sont : l'audience, le mécanisme d'authentification et les conséquences d'une exploitation sur l'utilisateur final.

L'audience

Malgré les recommandations de sécurité, les utilisateurs non formés à la sécurité privilégieront la facilité d'usage au détriment de la sécurité [1]. Ce qui fait des utilisateurs, le maillon faible de la sécurité des systèmes d'informations [151]. De ce fait, nous proposons d'étudier des applications dont l'audience, pour la majorité, ignorent les risques de sécurité qui peuvent être introduits par leurs propres choix.

Le mécanisme d'authentification

Le moyen d'authentification est au cœur de notre étude d'où l'importance de couvrir les moyens d'authentification représentatifs de la pratique actuelle. Le couple identifiant/mot de passe, est le moyen d'authentification le plus utilisé par les applications Web/Mobile pour protéger l'accès aux ressources de l'utilisateur. Ce mécanisme d'authentification comporte certaines limites tels que les problèmes de mémorisation et la ré-utilisabilité qui peuvent conduire à des exploitations [180, 96]. La fédération d'identités a été introduite afin de permettre à une application de déléguer son authentification à un fournisseur d'identité de confiance et de faciliter le parcours d'authentification via les mécanismes de SSO. En outre, l'authentification à deux ou plusieurs facteurs est utilisée pour garantir une meilleure sécurité. Nous cherchons ainsi à couvrir tous ces mécanismes d'authentification dans cette étude.

Les conséquences d'une exploitation sur l'utilisateur final

Les conséquences et l'impact d'un accès non-autorisé diffère d'une application à une autre. Celles-ci seraient dévastatrices pour une application qui traite des données personnelles. Toutes les applications de l'étude traitent des données sensibles de l'utilisateur (*i.e.*, carte bancaire, données personnelles).

3.1.2 Démarche de l'étude empirique

Pour étudier les parcours d'authentification des différentes applications, nous avons créé deux comptes temporaires sur Yahoo [1] et Facebook [2]. Le compte Yahoo sert de compte de mail qui est indispensable pour souscrire aux services Web actuels. Le

1. `https://www.yahoo.fr`
2. `https://www.facebook.fr`

compte Facebook est utilisé pour les mécanismes de fédération d'identités (*e.g.*, délégation d'authentification, SSO). Sur chaque application, nous analysons les différentes phases qui composent le parcours d'authentification de la manière suivante :

— nous utilisons chaque phase selon son usage normal en ajoutant ou modifiant les éléments de contexte ;
— nous relevons et classifions les exploitations logiques en fonction de leurs causes (*i.e.*, classes de défauts) et leur localisation (*i.e.*, quelle phase du parcours d'authentification) ;
— enfin, nous utilisons ces résultats pour construire une ontologie et en déduire un ensemble d'exigences permettant de les maîtriser.

3.1.3 Résultats de l'étude empirique

La table 3.1 récapitule les résultats de l'étude empirique. La description des exploitations sur chaque application est discutée ci-après.

Skype

Skype [3] est une application de messagerie maintenue par Microsoft. L'application est utilisée par des millions de personnes à travers le monde, notamment par les entreprises. En 2012, un détournement du comportement légitime de l'application permettait à une personne malicieuse de changer le mot de passe de n'importe quel autre utilisateur du service [108]. Pour se faire, celle-ci crée un nouveau compte sur Skype avec une adresse électronique existante et un nom d'usage unique (*i.e.*, le nom d'utilisateur). Puis, grâce au parcours de changement de mot passe, celle-ci choisit de changer le mot de passe associé au compte légitime. En effet, lors du changement de mot de passe, l'application propose le changement du mot de passe l'ensemble des comptes associés à l'adresse mail fournie. Ce qui permet de choisir le compte de l'entité légitime et de changer son mot de passe. Ci-dessous, les défauts logiques qui ont permis cette exploitation.

— L'application permet d'associer plusieurs comptes à une même identité sans vérifier l'appartenance de l'adresse mail au souscrivant.
— Le parcours de changement de mot de passe ne dispose pas de challenge de sécurité pour authentifier explicitement l'utilisateur courant.

Blablacar et Showroom Privé

Les deux applications proposent un formulaire de login qui utilise soit le couple identifiant/mot de passe ou la délégation d'identité avec Facebook comme fournisseur d'identités. Nous avons identifié qu'il était possible de créer deux comptes associés à

3. `https://www.skype.fr`

Application	Description de l'exploitation
Skype	Un utilisateur quelconque peut créer un compte frauduleux et l'associer à l'identité d'un utilisateur existant via la phase d'enrôlement. Ensuite, cet utilisateur peut changer le mot de passe de l'utilisateur légitime grâce à un manque de vérification lors du changement de mot de passe.
Blablacar	L'accès au dispositif d'une personne légitime permet d'accéder à son compte utilisateur grâce à la session persistante et aux formulaires automatiquement remplis.
Améli	La protection anti-brute force peut être exploitée par une personne malintentionnée pour empêcher des utilisateurs légitimes à accéder à leur espace personnel.
Showroom Privé	Idem que *Blablacar* ; de plus, un utilisateur non vérifié peut associer deux comptes à la même identité via la phase d'enrôlement et via la fédération d'identité. En fédération d'authentification, le parcours de changement de mot de passe est bloquant.
Leboncoin	L'application ne fixe pas de limite pour deviner le mot de passe lors de la phase de login. De plus elle autorise les sessions persistantes et les formulaires automatiquement remplis sur toutes les phases.
Amazon	voir exemple illustratif (*cf.*, Section 1.1.2)
Yahoo	l'application ne fixe pas de limite pour deviner le mot de passe, autorise les sessions persistantes et les formulaires automatiquement remplis. Par conséquent, il suffit d'accéder au dispositif de la personne légitime pour accéder au compte de l'utilisateur.
Paylib	La validation à distance ouvre les possibilités d'une attaque interposée et permet d'espérer que l'utilisateur final valide la mauvaise transaction.
Mobile Connect et moi	L'accès au dispositif de la personne légitime permet de changer le code PIN d'accès de celui-ci via la phase de récupération car cette phase ne requiert pas de challenge de sécurité supplémentaire. L'application souffre également de la validation à distance comme sur *Paylib*.

Table 3.1 – Récapitulatif des résultats de l'étude empirique

la même adresse mail. Pour ce faire, nous avons souscrit aux services en utilisant la souscription par adresse mail. Ensuite, nous utilisons cette même adresse mail pour créer un compte chez le fournisseur d'identité en l'occurrence Facebook. Ainsi, suite à une demande de connexion via *Facebook*, nous constatons que *Blablacar* propose d'associer les deux comptes en challengeant l'utilisateur avec son mot de passe initialement créé. Tandis que Showroom Privée associe automatiquement les deux comptes en se basant sur l'adresse mail et sans challenge de sécurité.

En outre, nous remarquons que la récupération d'un mot de passe perdu utilise le mécanisme Authentification et Identification fondée sur l'adresse e-mail - *Email Based Identification and Authentification* (EBIA) [76]). En principe, un lien de réinitialisation du mot de passe est envoyé par mail. Enfin, la mise à jour du mot de passe est bloquante pour les entités ayant recours à la fédération d'identités du fait que l'application demande un mot de passe qui n'est pas transmis lors de l'authentification ou l'enrôlement via la fédération d'identité. En effet, un des avantages de la fédération d'identités est de ne pas transmettre le mot de passe de l'utilisateur à l'application tierce, à la place, un jeton de confiance est transmis [144, 88].

Les défauts logiques identifiés sont les suivants.

— Du fait de la fédération d'identités, il est possible d'associer deux identités à un compte (*i.e.*, le cas de *Showroom Privée*). Ce manque de vérification de l'identité de la personne sur *Showroom Privée* permettrait à une personne qui aurait accès au compte Facebook d'un utilisateur légitime, d'accéder à son compte *Showroom Privée* par le simple fait que l'adresse mail de contact est le même.
— Malgré l'envoi d'un message de confirmation, le service reste accessible aux utilisateurs n'ayant pas confirmé leur adresse mail.
— De même que sur Skype, le parcours de récupération de mot de passe se base sur *EBIA* et ne requiert pas de challenge de sécurité supplémentaire. Ceci facilite l'usurpation d'identité de l'utilisateur légitime en cas d'accès au dispositif de celui-ci par une personne malicieuse.
— Pour une entité déjà authentifiée, le changement de mot de passe requiert la connaissance du mot de passe courant. Cependant, ce parcours est bloquant pour les raisons expliquées sur la fédération d'identités.
— Le formulaire de login ne fixe pas de limite pour deviner le mot de passe. Ceci facilite les attaques par force brute [127].

Yahoo

Nous n'avons pas remarqué de défauts logiques sur le parcours d'enrôlement de l'application. Néanmoins, celle-ci souffre des mêmes défauts que les applications *Blablacar* et *Showroom Privée* concernant le formulaire d'ouverture de session. Le changement de mot de passe ne requiert aucun challenge pour un utilisateur déjà authen-

tifié. La récupération d'un mot de passe perdu se base sur EBIA ou sur l'envoi d'un OTP par SMS.

Ameli

L'application Ameli[4] traite des données sensibles notamment des données médicales (*e.g.*, médecins consultés, remboursement de soin médicaux). Nous avons inclus cette application dans notre étude pour mettre en exergue un défaut logique résultant des mesures de protection contre les attaques par force brute.

En effet, après trois mauvais essais pour se connecter ; l'utilisateur est bloqué pendant 15 minutes. De plus, dans ce cas précis, la récupération de mot de passe ne pourra se faire que par voie postale (Il y a possibilité de récupération d'un code temporaire au moment où l'on rédige ce manuscrit, ce qui n'était pas le cas lors de l'étude empirique). Ceci peut s'avérer très gênant si l'erreur est d'origine légitime.

Par ailleurs, du fait de la nature discriminatoire de l'identifiant utilisé (*i.e.*,le numéro de sécurité sociale[5]). Une personne malicieuse pourrait détourner cette protection afin de mettre en déni de service une personne ou un groupe de personnes ciblées(*cf.*, Figure 3.1).

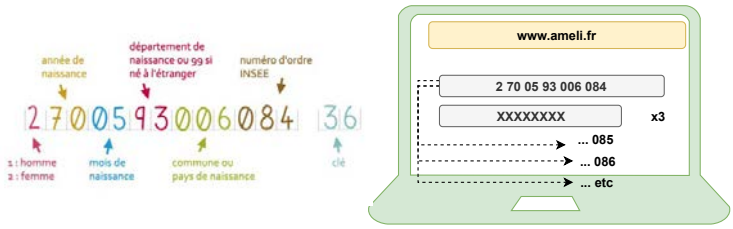

Figure 3.1 – Illustration d'un déni de service d'accès à Ameli de la population des femmes du département 93 nées au mois de Mai 1970

Paylib et Mobile Connect et Moi

Mobile Connect et moi est une implémentation, par Orange, du standard d'authentification *Mobile Connect* spécifié par la GSMA [83]. Alors que *Paylib* est une implémentation du protocole *3D-Secure*[6] par la Société Générale. Ces deux applications utilisent l'authentification à deux facteurs, d'où leur choix dans l'étude. L'authentification à deux facteurs permet d'authentifier une entité en se basant sur deux facteurs qui peuvent être : la connaissance (*e.g.*, un secret), la possession (*e.g.*, dispositif physique), le soi (*e.g.*, la biométrie) [79].

4. https ://www.ameli.fr
5. https ://www.service-public.fr/particuliers/vosdroits/F33078
6. https ://fr.wikipedia.org/wiki/3-D_Secure

Le principe de fonctionnement du parcours d'authentification est le même pour les deux applications. L'action de paiement ou de *login* est initiée sur l'application tierce. L'utilisateur renseigne son numéro de téléphone pour *Mobile Connect et moi* ou le numéro de sa carte bancaire pour effectuer un paiement dans le cas de Paylib. Une notification est envoyée sur l'application mobile de l'utilisateur. Puis, celle-ci valide la transaction avec le moyen d'authentification configuré (*i.e.*, code pin ou empreinte digital).

Ici, la validation à distance de la transaction constitue un défaut logique qui pourrait être exploité. Le cas le plus simple est une validation par inadvertance d'une transaction dont l'utilisateur légitime n'est pas à l'origine (*e.g.*, le cas d'un vol de carte bancaire sur internet). Le cas le plus complexe, est une attaque interposée qui consiste à faire une demande de validation parallèle à celle de l'utilisateur légitime et amener celui-ci à valider la demande frauduleuse (*cf.*, Figure 3.2).

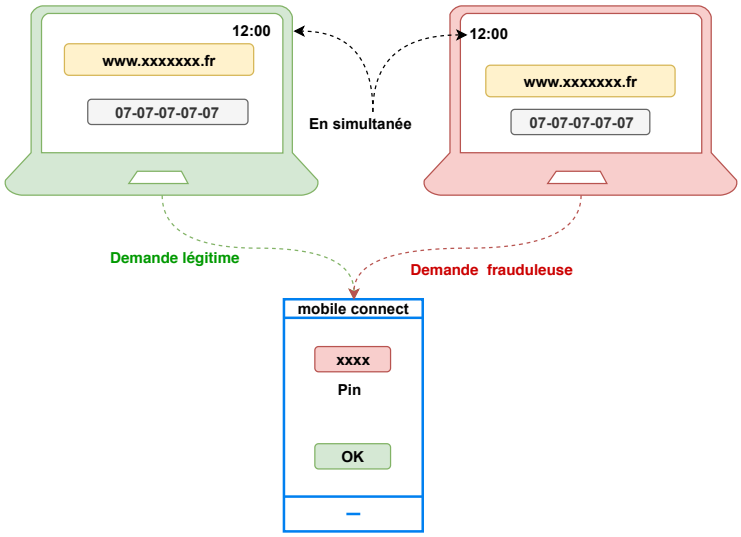

Figure 3.2 – Illustration d'une demande simultanée de validation d'une demande de connexion avec l'application *Mobile Connect et moi* ou *Paylib*

En outre, le parcours de récupération du code pin sur *Mobile Connect et moi* est faible. Il consiste à fournir son numéro de téléphone sur l'interface Web, et ainsi recevoir une notification de création d'un nouveau code Pin sans challenge de sécurité pour confirmer l'identité de la personne qui détient le téléphone. Ainsi, la compromission du téléphone permet d'accéder à tout service qui utilise *Mobile Connect et moi* comme moyen d'authentification via le parcours de récupération du code Pin (*cf.*, Figure 3.3).

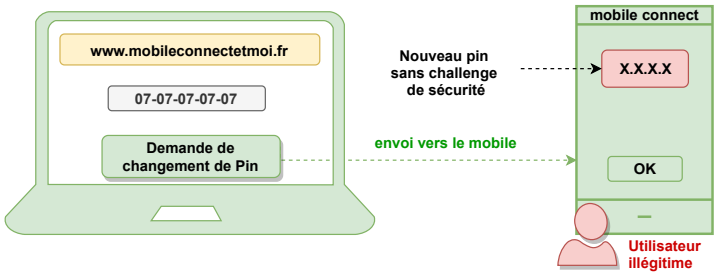

Figure 3.3 – Illustration d'un changement de Pin faible avec l'application *Mobile Connect et moi*

3.1.4 Conclusions de l'étude empirique

Nous avons montré que certains choix de design et le comportement humain constituent des défauts exploitables par une personne malicieuse. Les exemples d'exploitations décrites plus haut et résumés dans la Table 3.1 montrent que ces défauts sont sensibles aux contextes d'exécution des applications. Leur exploitation nécessite la connaissance de la logique décisionnelle de la dite application d'où leur qualification en défaut logique [72]. Leur détection par les méthodes de test et les outils d'analyse est difficile [171]. Ainsi, nous nous focalisons sur l'ingénierie des exigences afin de les prendre en considération dès la phase de conception du parcours d'authentification [57, 157, 165]. Afin de capturer les exigences pertinentes, nous proposons d'étudier la nature des relations qui existent entre les exploitations logiques, les défauts logiques et le parcours d'authentification.

3.1.5 Atteintes à la validité

Nous rappelons que le but de cette expérience est d'étudier la généralisation des défauts logiques de sécurité sur des applications grand public et de prouver que ces défauts peuvent nuire à l'utilisateur final.

En ce sens, le nombre limité d'applications étudiées pourrait constituer une atteinte à la généralisation des défauts logiques. Nous avons cherché à atténuer cette limitation dans nos critères de sélection en choisissant des applications de différents domaines d'activités et qui sont représentatives de l'implémentation actuelle des schémas d'authentification. Ainsi, intuitivement, on peut s'attendre aux mêmes résultats sur d'autres applications implémentant les mêmes schémas d'authentification que ceux étudiés.

Par ailleurs, cette étude est guidée par la définition du parcours d'authentification (*cf.*, Section 1.1.1). Ainsi, elle se focalise sur l'analyse des composants du parcours d'authentification. Cette démarche ne couvre malheureusement pas les défauts logiques liés à d'autres composants qui peuvent avoir des conséquences néfastes pour

l'utilisateur final. Dans ce même contexte, l'étude ne garantit pas l'exhaustivité des défauts logiques identifiés du fait que ces derniers évoluent en fonction du contexte de l'application concernée.

3.2 Caractérisation des exploitations logiques

L'étude empirique a montré que les exploitations des défauts logiques différaient d'un contexte à l'autre. Ce changement de contexte rend difficile le test et la détection automatique des défauts logiques. Cependant, à partir des résultats de l'étude empirique, nous disposons d'une base de connaissance nous permettant de construire une ontologie du parcours d'authentification. Cette ontologie définit les relations qui existent entre les composants du parcours d'authentification, les classes de défauts logiques et les exploitations logiques.

Il est important de rappeler que nous nous focalisons sur les défauts logiques dus aux choix de conception et non aux erreurs de codage. Un défaut logique est spécifique au développement quand celui-ci est causé par une erreur ou une insuffisance dans l'implémentation du programme de l'application. Ainsi, le défaut disparaît avec la correction du programme. C'est l'exemple des failles Injection de requêtes SQL - *SQL Injection* (SQLI) [149] ou *Cross-Site Scripting* (XSS) [84] qui exploitent une vulnérabilité dans le programme.

Par ailleurs, les défauts logiques considérés dans cette thèse sont intrinsèquement liés à la spécification. Ils persisteront malgré l'implémentation qui sera faite de l'application. Ainsi, seule leur prise en compte par la modification de la spécification permet de les prévenir. C'est la raison pour laquelle nous utilisons les termes défauts et insuffisances (ou manquement) plutôt que failles et vulnérabilités.

Il existe principalement deux fonctions qui sont partagées par les composants du parcours d'authentification : la fonction de vérification des attributs d'identité et la fonction d'authentification lors de l'utilisation du service. La vérification des attributs d'identité est nécessaire à la souscription d'une nouvelle entité. Tandis que le mécanisme d'authentification est utilisé tout au long du cycle de vie de l'application par les différents composants du parcours.

3.2.1 Défauts logiques liés à la phase d'enregistrement

Insuffisance dans la vérification des identités

Une insuffisance dans la vérification des attributs d'identité est constatée lorsque l'application échoue sur la vérification des propriétés d'unicité, d'appartenance et de validité des attributs d'identité fournies par l'entité. La validité assure que l'attribut est valide et existe. L'unicité assure que l'entité est unique dans le contexte de l'ap-

plication. Tandis que, l'appartenance prouve que les attributs fournis appartiennent à l'entité qui les a fourni. L'exploitation majeure d'une insuffisance dans la vérification d'identités est la souscription d'une entité frauduleuse qui pourrait utiliser le service à des fins malicieuses (*cf.* Section 3.1.1).

Insuffisance dans l'émission des preuves de sécurité

Les preuves de sécurité (*i.e.*, *credentials* en anglais) permettent à une entité d'accéder à un service via le formulaire de *login*. Une application peut nécessiter divers preuves de sécurité pour des actions liées à son fonctionnement : la réalisation d'actions sensibles (*e.g.*, validation d'un paiement en ligne, changement d'un mot de passe). Ainsi, une insuffisance dans l'émission de ces preuves de sécurité est constatée lorsque la réalisation d'une action est bloquée par l'absence de la preuve d'identité nécessaire pour l'accomplissement cette action.

Ce défaut impacte principalement l'utilisabilité d'un service en causant des états bloquants pour certains usages. (*i.e.*, voir l'application *Blablacar* et *Showroom Privée* en Section 3.1.1). Ce défaut est très courant chez les applications qui utilisent la délégation d'authentification et qui demandent le mot de passe d'origine pour la réalisation de certaines actions (*e.g.*, changement d'adresse, ajout de carte bancaire). Par ailleurs, il est important de préciser que l'acquisition d'une nouvelle preuve d'identité est possible ultérieurement dans le cycle de vie de l'application. Celle-ci est considérée comme une mise à jour des attributs d'identité, par conséquent, elle nécessite un mécanisme de sécurité afin de satisfaire les exigences requises.

3.2.2 Défauts logiques liés au mécanisme d'authentification

Les défauts classifiés dans cette catégorie concernent les mécanismes d'authentification utilisés lors des phases de *login*, les phases de récupération et de mise à jour d'une preuve d'identité. Particulièrement, c'est l'authenticité de l'entité légitime qui est mise en défaut. Le risque de sécurité majeur est l'accès non-autorisé dont les conséquences sont entre autres : l'imitation de l'entité légitime, la substitution de celle-ci et le vol d'identité. Ci-après les différentes classes de défauts logiques identifiées.

Défaut de limitation impropre du nombre de tentative d'authentification

La limitation du nombre d'essai pour trouver un secret est l'une des mesures de sécurité la plus utilisée pour éviter les attaques par force brute. Cette mesure est parfois accompagnée d'une Preuve d'interaction avec un Humain - *Human Interaction Proof* (HIP) tel que le *Captcha* afin de détecter les agissements robotisés [32].

Beaucoup de spécifications fixent un nombre d'essai relativement bas et choisissent de bloquer l'entité pendant une durée déterminée en cas d'échec. Cette mesure peut être détournée afin de bloquer l'accès à une entité cible (*i.e.*, voir l'application *Améli* en Section 3.1.1). Cependant, ne pas fixer un nombre limité de tentative expose l'application aux attaques par force brute. Un compromis est nécessaire pour protéger l'application contre les attaques par force brute d'une part. Et, d'autre part, éviter que la mesure ne soit détournée afin de mettre en déni de service les entités légitimes.

Défaut de validation d'une authentification multi-facteurs

La validation de certaines authentifications multi-facteurs pourrait être exploitée afin d'amener l'utilisateur légitime à valider une transaction dont il n'est pas à l'origine (voir l'application *Paylib* en Section 3.1.1). En effet, deux types de validation existent dans l'authentification multi-facteurs : la validation locale et la validation à distance. La validation est locale si la transaction est explicitement finalisée sur le canal de départ. Par exemple, le *3D secure* avec un OTP [87] à confirmer sur canal de départ (*i.e.*, interface d'une application Web/Mobile). Alors qu'une validation à distance est effectuée via un autre canal que celui d'origine. C'est l'exemple d'une validation via une application tierce telle que sur *Paylib* et *Mobile Connect et Moi*. Les méthodes d'ingénierie sociale qui permettent de récupérer l'OTP et de valider une transaction localement ne sont pas prises en compte dans cette thèse.

Par ailleurs, nous considérons la validation à distance comme étant un défaut logique car pouvant conduire l'utilisateur légitime à valider une transaction dont il n'est pas à l'origine. Deux types d'attaques sont possibles. Le premier consiste à initier la transaction à la place de l'entité légitime et espérer que celle-ci la valide par inadvertance. Le second, plus complexe, nécessite de réaliser une transaction frauduleuse en même temps qu'une transaction légitime et amener l'entité légitime à valider la transaction frauduleuse.

Défaut de vérification des attributs d'identité

Nous avons remarqué que certaines applications qui utilisent la fédération d'identités (*i.e.*, *Showroom Privée*) autorisent instantanément l'accès à une demande d'ouverture de session via la fédération d'identités s'il existe un compte associé à l'adresse e-mail de contact du fournisseur d'identités.

Pour illustrer l'attaque, nous allons souscrire une entité sur *Showroom Privée* avec l'adresse e-mail *entite@entite.fr*. Nous précisons ne pas utiliser la fédération d'identités pour la souscription de l'entité. Néanmoins, nous créons un compte *Facebook* avec la même adresse e-mail (*i.e.*, entite@entite.fr). Ainsi, en se connectant sur l'application *Facebook*, nous pouvons accéder à l'application *Showroom Privée* via le formulaire de

délégation d'authentification même si l'entité ne s'est pas inscrite avec la fédération d'identité. En effet, l'application se base sur l'adresse e-mail de contact afin de lier l'entité à un compte existant sur *Showroom Privée*. L'erreur fondamentale est le fait de considérer que l'entité connectée sur le fournisseur d'identité est la même que celle qui a souscrit au service.

Le mécanisme d'authentification, en l'occurrence *Facebook Login*[7], n'est pas mis en défaut dans cette exploitation. Cependant, la faute est due au mécanisme de vérification de l'application tierce (*i.e.*, *Showroom Privée*).

Défaut d'authentification explicite/hypothèse de sécurité

Le défaut d'authentification explicite caractérise toute conception ou choix de l'utilisateur pouvant, partiellement ou complètement, conduire à une absence d'authentification explicite de l'entité courante. Nous partons du postulat que le mécanisme d'authentification doit être supporté par le fournisseur de service ou à défaut, explicitement délégué à un tiers de confiance (*e.g.*, un fournisseur d'identités). Ainsi, est inclus dans cette catégorie, tout mécanisme d'authentification fondé sur une hypothèse de sécurité susceptible de ne pas être respectée. Par "hypothèse" de sécurité, on entend le fait de considérer que l'environnement d'exécution (*e.g.*, le téléphone, l'ordinateur portable) d'une application ou d'un service est sécurisé et n'est accessible que par l'entité légitime. Par conséquent, la sécurité de l'application est implicitement substituée par celle de son environnement d'exécution. De ce fait, la compromission de la sécurité de cet environnement d'exécution est propagée sur l'ensemble des services s'y exécutant. Par exemple, pour limiter le nombre de demandes d'authentification ; la pratique actuelle, entre autre, consiste à garder la session utilisateur actif via des *cookies* de session stockés dans le navigateur. De ce fait, l'accès à ce cooki de session [37] ou l'accès au navigateur [10, 177] garantit l'accès au service.

Nous précisons que la délégation d'authentification et l'authentification fondée sur la possession d'un secret (*e.g.*, carte à puce, téléphone mobile etc.) ne sont pas considérées comme des hypothèses de sécurité, mais plutôt, comme des mécanismes alternatifs.

Existence de chemins faibles

Plusieurs composants d'un parcours d'authentification peuvent permettre d'accéder à un service. Parmi eux, le formulaire de *login*, la récupération des preuves de sécurité ou encore la fédération d'identités. Chacun de ces composants utilisent un moyen d'authentification de niveau de sécurité différent. Des standards et recommandations existent pour définir ce niveau de sécurité [26, 79, 95]. Par contre, ces moyens ne prennent pas en compte les éléments de contexte qui influent sur ce parcours et sur

7. https ://developers.facebook.com/docs/facebook-login/

son niveau de sécurité. Nous considérons qu'il y a un *chemin faible* lorsqu'il existe un parcours dont le niveau de sécurité est plus faible que le niveau d'authentification requis. Cela implique au concepteur du parcours de définir un niveau de sécurité requis pour accéder au service.

3.2.3 Vers une ontologie du parcours d'authentification

L'étude empirique et la classification des défauts logiques constituent une base de connaissance pour la création d'une ontologie des risques relatifs au parcours d'authentification (*cf.* Figure 3.4). Cette ontologie est similaire à certaines ontologies discutées dans l'état de l'art [159, 104], cependant celle-ci est spécifique au contexte du parcours d'authentification dans le but de capturer les exigences pertinentes pour limiter les défauts logiques. Ainsi, cette ontologie définit les relations entre les composants du parcours d'authentification, les classes de défauts et leurs exploitations par une entité malicieuse. Les concepts de cette ontologie sont décrits ci-après.

- Le **parcours d'authentification** avec ces différentes composantes constitue l'abstraction de notre bien à protéger contres les risques. En réalité, le bien en question est le **compte utilisateur** d'une entité légitime qui doit être protégé contre les risques d'usurpation d'identité (*e.g.*, accès non autorisé, souscription frauduleuse) ou d'inaccessibilité.
- Les **risques** sont des évènements redoutés (*e.g.*, usurpation, inaccessibilité) occasionnés par les défauts logiques.
- Les **défauts logiques** constituent les failles de sécurité susceptibles d'exister sur le parcours d'authentification.
- Les facteurs sont des éléments de contexte ou de spécification qui peuvent conduire à une faille de sécurité soit de façon unique ou par combinaison de différents facteurs.
- Les **exploitations ou attaques logiques** sont des conséquences négatives sur le parcours d'authentification.
- Les **exigences** peuvent servir de mesures pour atténuer les risques.
- Les **mécanismes** permettent d'appliquer les exigences spécifiées.

Certains concepts présents dans les ontologies relatives à la sécurité tels que les menaces, les attaquants et les conséquences ou impacts ne sont pas traités dans cette ontologie. En effet, dans notre contexte, ces concepts ne nous apportent pas d'informations supplémentaires pour capturer nos exigences. Par exemple, le concept d'attaquant est ignoré car le seule attaquant considéré dans ce contexte de défaut logique est l'humain ; il en est de même pour les impacts des risques car celles-ci dépendent du contexte de l'application. Cependant, ces concepts pourraient être ajoutés afin d'étendre l'ontologie.

Les concepts de notre ontologie est les relations entre ces concepts sont présentées dans la Figure 3.4. Elle se lit comme suit : le parcours d'authentification (*e.g.*,

login, récupération) peut *contenir* un ou plusieurs défauts logiques (*e.g.*, erreur de vérification, validation) qui *sont causées* par des facteurs humains, choix de conception ou le dispositif de l'utilisateur final. Ces défauts logiques sont ainsi *exploités* par une personne malicieuse pour réaliser des exploitations logiques qui *conduisent* à des risques (*e.g.*, inaccessibilité, substitution de la personne légitime). Les exigences *appliquées* à travers les mécanismes de sécurité permettent *d'atténuer* ces risques.

Un exemple illustratif de l'ontologie est une attaque de déni de service qui conduit à l'inaccessibilité du compte de l'utilisateur légitime. Ainsi en utilisant l'ontologie on peut caractériser l'attaque comme exploitant un défaut logique causé par un choix de design du type *mauvaise limitation* du nombre de tentative d'authentification. Ainsi, en déduire le ou les exigences qui permettent d'atténuer ce risque. C'est dans cette optique d'aider à capturer les exigences pertinentes que nous avons construit cette ontologie. La section suivante présente ainsi la liste des exigences que nous préconisons.

3.3 Les exigences de sécurité

Cette section présente l'ensemble des exigences de sécurité que nous préconisons afin de limiter ou prévenir les exploitations logiques dès la phase de conception par la prise en compte des classes de défauts décrites en Section 3.2.

3.3.1 Exigences sur la vérification des attributs d'identité

Le premier groupe d'exigences que nous préconisons est relatif à la phase d'enrôlement d'une entité.

Exigence 1 (#Ex1) : Unicité de l'identité d'une entité

L'unicité de l'identité d'une entité dans un service est fortement recommandée afin d'éviter tout conflit dans la gestion des comptes utilisateurs. Le mécanisme de vérification des attributs d'identité doit s'assurer de cette unicité dans un contexte bien identifié. Particulièrement, les attributs de l'identité physique (*e.g.*, permis de conduire, passeport) ainsi que les identifiants (*e.g.*, numéro de téléphone, adresse mail, etc.) sont les plus concernés par cette exigence. En effet, autoriser plusieurs comptes pour la même identité, d'une part, complexifie la gestion des comptes utilisateurs, d'autre part, peut mener à des exploitations si le contrôle d'appartenance n'est pas correctement réalisé (*cf.* 3.1.1).

Figure 3.4 – Ontologie du parcours d'authentification avec les classes de défauts logiques et les exploitations correspondantes

.

Exigence 2 (#Ex2) : Validité et existence des attributs d'identité

Cette exigence permet de satisfaire les propriétés de validité et d'existence des attributs d'identité notamment les attributs essentiels à l'usage d'un service. La pertinence de ces propriétés diffèrent d'une application à l'autre. Elles peuvent se référer au format, à la non révocation et la validité d'un attribut d'identité. Par exemple, la validité d'une adresse mail assure que le format de celle-ci est correcte. Alors que

la propriété d'existence s'assure de l'existence du fournisseur de mail et de la non révocation de l'adresse mail.

Pour les attributs physiques (*e.g.*, état civil, passeport, etc.), la validité concernera la validité dans le sens commun du terme (valide au niveau de la loi) ; l'existence se référera à l'existence physique de l'entité.

Exigence 3 (#Ex3) : Appartenance des attributs d'identité à l'entité

Les propriétés de validité et d'existence ne garantissent pas l'appartenance d'attributs d'identité à une entité souscrivante. Ainsi, cette exigence stipule que le mécanisme de vérification doit s'assurer que les données fournies par l'entité appartiennent bien à celle-ci. Par exemple, un numéro de téléphone peut être valide mais appartenir à une autre entité que celle qui l'a fourni. Pour les mêmes raisons, les attributs physiques peuvent satisfaire les propriétés de validité et d'existence. Cependant, l'entité qui les fournit peut être frauduleuse.

Exigence 4 (#Ex4) : Disponibilité des preuves de sécurité

La phase d'enrôlement permet de créer les preuves de sécurité qui seront utilisées lors des phases nécessitant un mécanisme d'authentification telle que la phase d'ouverture de session. Cette exigence stipule que tout justificatif d'identité nécessaire à un futur usage dans l'application, devra être émis ou fourni durant la phase d'enrôlement. Il est possible que l'acquisition ou l'émission de nouvelles preuves de sécurité soit réalisée ultérieurement dans le cycle de vie de l'application. L'action est donc considérée comme sensible ; elle nécessitera un mécanisme d'authentification.

L'indisponibilité des preuves de sécurité peut entraîner des blocages sur l'usage de l'application et causer des gênes en terme d'utilisabilité pour l'utilisateur final(voir l'application Blablacar dans la Section 3.1.1).

3.3.2 Exigences sur les mécanismes d'authentification

Le principal but du mécanisme d'authentification est de garantir l'authenticité de l'identité de l'entité utilisatrice d'un service. Elle permet aussi la réalisation d'actions considérées importantes ou à fortes valeurs ajoutées pour un service donné. Le contrôle d'accès est un moyen éprouvé pour limiter et contrôler l'accès aux ressources d'un service en fonction des attributs de l'entité [181, 61, 38]. Cependant, cette thèse ne traite pas des problématiques liées au contrôle d'accès. Nous nous focalisons uniquement sur les actions liées aux composants du parcours d'authentification.

Exigence 5 (#Ex5) : L'hypothèse de sécurité et l'authentification explicite

L'étude empirique a permis de soulever deux défauts majeurs sur la pratique actuelle des mécanismes d'authentification : l'hypothèse de sécurité et le défaut d'authentification explicite. L'hypothèse de sécurité désigne le mécanisme par lequel la sécurité d'un service s'appuie sur un supposé sécurité dont le fournisseur de service ne maîtrise pas et qui est susceptible d'être violé (*e.g.*, l'usage de sessions persistante en supposant que le dispositif de l'utilisateur est sécurisé ou n'est accessible que par celui-ci).

L'hypothèse de sécurité est prohibée car la sécurité d'une application doit être assumée par celle-ci ou explicitement déléguée à un tiers de confiance. En effet, il n'est pas acceptable que la compromission d'un facteur externe à un service entraîne la compromission de celui-ci.

De même, l'authentification doit être explicite afin d'assurer un niveau de confiance élevé sur l'utilisateur d'une application. Nous proposons les exigences ci-après afin de permettre de limiter les exploitations en relation avec l'hypothèse de sécurité et l'authentification implicite (*i.e.*, manque d'authentification explicite)

Exigence 5' : Limitation de la durée des sessions utilisateur Nous savons que le niveau de confiance sur l'authenticité de l'identité d'un utilisateur d'un service déprécie dans le temps [89]. De même, une personne mal-intentionnée est susceptible d'avoir accès aux dispositifs d'exécution de ces applications par différents moyens [10, 23].

Pour ces deux raisons, cette exigence stipule que la durée des sessions utilisateur doit être limitée à une valeur minimale en fonction des besoins de l'application. Cela a pour effet, d'une part, de garantir le niveau de sécurité requis sans dépréciation temporelle. D'autre part, elle permet de limiter les exploitations qui peuvent conduire à l'imitation ou substitution de l'entité légitime en cas de compromission de l'environnement d'exécution.

Exigence 5" : Prohibition de formulaires auto-remplis. Le principe de base de l'authentification est la validation d'un facteur (*i.e.*, connaissance, être, possession) entre une entité et un vérifieur du dit facteur. Pour une authentification fondée sur la connaissance, le formulaire pré-rempli viole cette propriété intrinsèque car, l'entité courante peut ignorer le secret et accéder au service. Un autre problème posé par cette pratique est la divulgation des preuves d'identité. Il suffit de modifier le type de l'attribut de la balise HTML sur la console du navigateur pour visualiser le mot de passe. Ainsi, il y a risque de compromission des autres services du fait de l'effet domino causé par la réutilisabilité des mots de passe [96].

Exigence 5”’ : Authentification pour tous. Cette exigence est relative aux phases de récupération et de mise à jour des preuves d’identité. Une majorité des fonctions de récupération des preuves d’identité se fonde sur l’envoi d’un lien de réinitialisation par mail (*e.g.*, EBIA). Ce protocole est parfois renforcé d’un challenge de sécurité comme la réponse à une question sécrète.

La sécurité de cette fonction dépend ainsi de la capacité à recevoir le message d’initialisation. Ce qui est une forme d’authentification fondée sur la possession (*i.e.*, le fait de posséder le mail ou le SMS de réinitialisation). Dans le cas où l’application tierce et l’application du fournisseur de messagerie (ou SMS) s’exécute sur un même dispositif (*i.e.*, le cas d’une forte corrélation) ; avec le risque que la session de celle-ci soit persistante, alors cette phase de récupération est exploitable si l’appareil est accessible par une personne mal-intentionnée.

Par ailleurs, la compromission de la boite de messagerie d’une personne entraîne également l’effet domino sur l’ensemble des services n’usant pas de challenge de sécurité supplémentaire sur la phase de récupération du mot de passe.

Il en est de même pour la mise à jour d’un justificatif d’identité d’une personne préalablement authentifiée. Un accès frauduleux pourrait conduire à la substitution de l’entité légitime si l’action de mise à jour ne nécessite pas une authentification explicite instantanée.

Afin de limiter ces exploitations (*i.e.*, *imitation* et *substitution* de l’entité légitime), cette exigence rend obligatoire l’usage d’un challenge de sécurité sur toutes les actions sensibles, notamment sur la fonction de récupération et de modification des preuves d’identité.

Exigence 6 (#Ex6) : Limitation du nombre d’essai

Les deux exploitations de la *limitation impropre* sont : le déni de service intentionnel d’une entité légitime et la force brute pour deviner le secret (*cf.* 3.1.1). Les deux peuvent être gênants pour l’entité légitime. Cependant, nous préconisons de limiter le nombre d’essai sur les mécanismes d’authentification car nous privilégions la sécurité (*i.e.*, accès non autorisé) au détriment de l’utilisabilité (déni de service).

Par ailleurs, nous recommandons d’utiliser des moyens d’authentification alternatifs plutôt que le blocage temporaire du compte de l’entité légitime. Ces moyens ne devraient pas permettre des fautes du type *chemin faible*.

Exigence 7 (#Ex7) : Absence de chemin faible

Tout chemin permettant d’accéder au compte utilisateur d’une entité doit nécessiter une authentification préalable. C’est ce que préconise l’exigence *authentification pour tous*. En effet, il peut exister plusieurs niveaux de sécurité sur un parcours en

fonction des mécanismes implémentés. Ainsi, afin d'empêcher un abaissement du niveau de sécurité globale ; le niveau de sécurité de l'ensemble de ces chemins doit, au minimum, correspondre à un niveau de sécurité requis. L'évaluation du niveau de sécurité est discutée dans le chapitre suivant.

Exigence 8 (#Ex8) : Authenticité et appartenance en délégation d'authentification

L'application tierce doit s'assurer de l'authenticité et de l'appartenance des attributs d'identité avant tout accès via la délégation d'authentification. Nous recommandons de gérer un compte local avec le type d'inscription afin d'éviter tout abus de vérification utilisant un compte utilisateur compromis (*i.e.*, le compte chez le fournisseur d'identités).

Exigence 9 (#Ex9) : Validation locale d'authentification multi-facteur

Une authentification multi-facteur doit assurer l'authenticité de l'entité tout au long du processus. Nous recommandons que la validation soit faite en local afin de renforcer l'authenticité de l'entité initiatrice de la transaction.

3.4 Conclusion du chapitre

Ce chapitre présente un ensemble d'exigences relatives à la sécurité des parcours d'authentification. Nous avons montré via l'étude empirique présentée en Section 3.1 que les parcours d'authentification des applications Web/Mobile pourraient être détournés afin de réaliser des exploitations qui peuvent nuire à l'utilisateur final. Ces défauts sont dues, d'une part, à des choix de conception, d'autres parts, aux comportements humains et aux changements de contexte d'exécution de ces applications. Nous les avons caractérisées de défauts logiques et avons fourni un ensemble de classes abstraites permettant de les structurer. Cette démarche permet de répondre à la première question de recherche(*i.e.*, **RQ1**, Section 1.2).

Pour mieux caractériser les exploitations logiques, nous avons introduit une ontologie qui s'appuie sur la définition du parcours d'authentification. De cette ontologie, nous en déduisons une taxonomie (*cf.*, Table 3.2) qui permet de caractériser les exploitations logiques par leur(s) point(s) d'entrée (*i.e.*, composant du parcours d'authentification), leur(s) cause(s) (*i.e.*, classe de défauts logiques) et les conséquences sur l'utilisateur légitime (*i.e.*, inaccessibilité, imitation, substitution, enrôlement frauleux). Cette ontologie a aidé à la définition d'exigences qui répondent à la seconde question de recherche (*i.e.*, **RQ2**, Section 1.2) afin de limiter les exploitations des classes de défauts logiques dès la phase de conception d'un parcours d'authentification.

Exploitation	Point(s) d'entrée	Cause(s)	Conséquence	Exigences
Abus de vérification	Phase d'enrôlement	Insuffisance de vérification d'identité	Enrôlement frauduleux	#Ex1, #Ex2, #Ex3
État bloquant (indisponibilité)		Insuffisance d'émission de preuve d'identité	Inacessibilité	#Ex4
Déni de service (intentionnel)	*Login* et récupération	Limitation impropre du nombre d'essai	Inaccessibilité	#Ex6
Brute force		Imitation		
Abus de vérification		défaut vérification	Imitation	#Ex8
Abus de validation		défaut de validation	Imitation	#Ex9
Exploitation d'un chemin faible		Chemin faible	Imitation	#Ex7
Abus de vérification	Mise à jour	défaut d'authentification	Substitution	Ex5

Table 3.2 – Taxonomie des exploitations logiques et les exigences correspondantes

Chapitre 4

Cadre d'évaluation du niveau de risques des exploitations logiques relatives à un parcours d'authentification

Ce chapitre présente le cadre d'évaluation du niveau de risques liés à un parcours d'authentification. Le processus d'évaluation se focalise sur la capacité d'une spécification d'un parcours d'authentification à prévoir ou à éviter les défauts logiques pouvant conduire à des exploitations. En d'autres termes, nous étudions le niveau de risque induit d'une part, par les défauts de conception du parcours d'authentification et d'autre part, par l'entité légitime par ses choix d'usages. Ce cadre d'évaluation se base sur les exigences et les exploitations logiques présentées dans le Chapitre 3.

4.1 Préambule

4.1.1 Définition du niveau de risque dans ce cadre

Plusieurs cadres de gestion des risques existent, notamment, EBIOS [9], ISO27-000 [47] et OCTAVE [6, 7]. Une définition du risque est : *un évènement indésirable potentiel pouvant, s'il n'est pas anticipé ou maîtrisé, empêcher ou entraver de manière significative la marche d'un projet ou le déroulement d'une activité vers ses objectifs*. Ainsi, il convient d'identifier les évènements redoutés, leurs causes, sources et impacts. Ensuite, pour chaque évènement redouté, évaluer le niveau de risque associé afin d'en déduire les mesures adéquates (*e.g.*, gérer, transférer ou accepter le risque).

Une des façons de faire, est de définir des niveaux associés à la probabilité d'occurrence et la criticité d'un risque. Par exemple la Table 4.1 décrit une analyse de

risque avec deux niveaux de risque (*i.e.*, Acceptable, Intolérable). Le risque est "Acceptable" tant que celui-ci à un impact mineur sur le système indépendamment de sa probabilité. Alors qu'il est "Intolérable" quand l'impact est critique ou significatif avec une probabilité d'occurrence élevée. La mesure à prendre en fonction de l'évaluation (*i.e.*, Acceptable, Intolérable) dépend ainsi de la politique de sécurité de l'application concernée. De plus, la criticité est aussi sensible au contexte de l'application en question. Pour toutes ces raisons susmentionnées, ce cadre se focalise sur la probabilité d'occurrence qu'un risque soit avéré. Ainsi, pour la suite du document, le niveau de risque, se référera à la probabilité d'occurrence (*i.e.*, la vraisemblance) d'un risque.

Impact \ Probabilité	Peu Probable	Probable	Très probable
Mineure	Acceptable	Acceptable	Acceptable
Significative	Acceptable	Intolérable	Intolérable
Critique	Intolérable	Intolérable	Intolérable

Table 4.1 – Exemple d'évaluation du niveau de risque de compromission d'un site web

4.1.2 Critères d'évaluation du niveau de risque

Comme nous l'avons précisé plus haut (*cf.* Section 4.1.1), en gestion de risque, il convient d'identifier le ou les biens à protéger, les sources, l'origine des menaces et les impacts sur ces biens à protéger. Dans ce cadre, les évènements redoutés (*i.e.*, les risques) sont une conséquence directe des exploitations logiques décrites en Section 3.2. Particulièrement, nous nous focalisons sur la souscription d'une entité frauduleuse, l'accès non autorisé à l'espace utilisateur de l'entité légitime et la substitution de celle-ci. L'impact de ces trois risques est souvent néfaste pour l'utilisateur final d'une application ou d'un service(*e.g.*, vol d'identité, préjudice pénale, perte financière).

La Figure 4.1 décrit le processus d'évaluation du risque avec les différents facteurs d'entrées. Ces facteurs sont : la spécification du parcours d'authentification (*i.e.*, en vérifiant le respect ou non des exigences de sécurité définies dans le Chapitre 3), le comportement de l'utilisateur final (*e.g.*, non respect des recommandations de sécurité, choix de sécurité faible, etc.) et les éléments de contexte extérieurs à l'application (*e.g.*, dispositif non sécurisé, perte ou vol d'un dispositif, etc.). De ces trois facteurs, nous en déduisons les critères pertinents qui permettent d'évaluer le risque sur trois niveaux : les niveaux ÉLEVÉ, MODÉRÉ et FAIBLE.

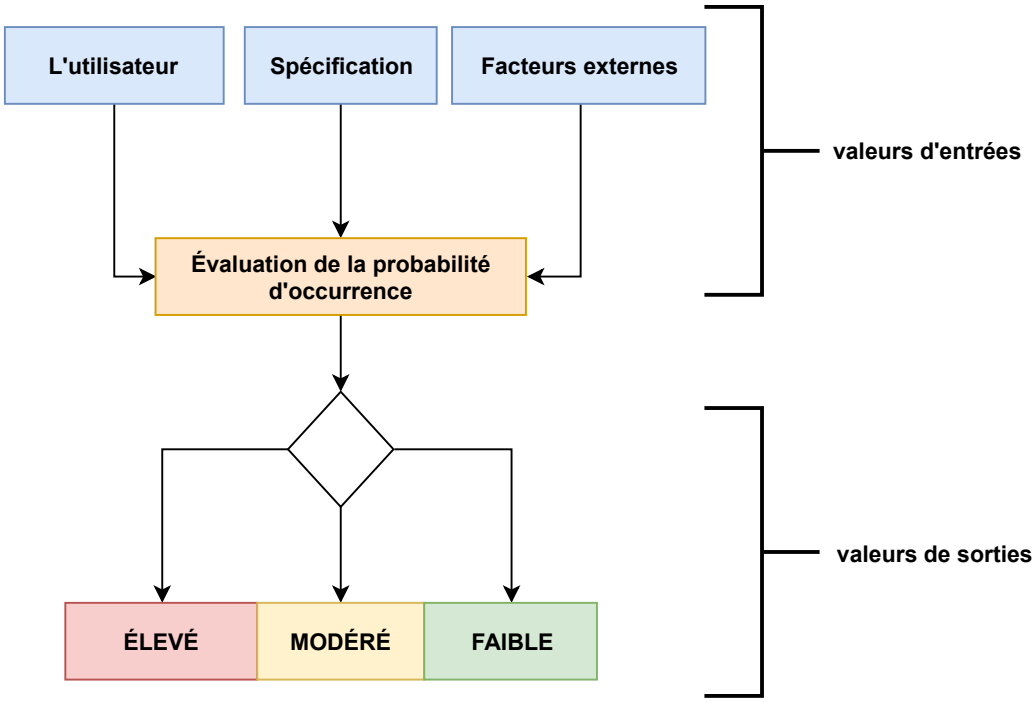

Figure 4.1 – Processus d'évaluation du niveau de risque

ÉLEVÉ

Le niveau de risque est ÉLEVÉ, s'il y a une évidence sur le non respect d'une ou de plusieurs exigences favorisant systématiquement l'exploitation relative à ce risque indépendamment des éléments extérieurs au parcours. La réussite de l'exploitation est donc "très probable".

MODÉRÉ

Le niveau de risque est MODÉRÉ, si le non respect d'une ou plusieurs exigences ou si une action (*e.g.*, changement, modification) des éléments de contexte est susceptible de conduire à l'exploitation correspondante. Le risque associé est donc "modérément probable" dans certaines conditions d'exécution.

FAIBLE

Le niveau de risque est FAIBLE, si tous les éléments connus (*e.g.*, exigences, environnement, mécanisme d'authentification, etc.) sont anticipés afin d'éviter le ou les exploitations identifiées. Nous gardons ce dernier niveau de risque du fait de la spécificité des défauts logiques à être imprévisibles et donc à favoriser de nouveaux risques durant le cycle de vie d'une application.

Organisation de la suite de ce chapitre Les sections suivantes détaillent respectivement la sémantique pour l'évaluation des risques d'une souscription d'une entité frauduleuse, d'un accès non autorisé à un service et la substitution de l'entité légitime. L'analyse et l'évaluation du niveau de risque ne pouvant pas être générique du fait de la particularité des évènements redoutés, les critères d'évaluation de chacun des risques sont présentés dans la section correspondante. De plus, nous proposons un exemple illustratif de la sémantique.

Exemple illustratif Nous allons considérer une plateforme de partage de voiture entre particuliers. Le principe consiste à mettre à disposition sa voiture à des particuliers moyennant une somme d'argent fixée en fonction de la durée de la location. La souscription à l'application nécessite un permis de conduire, un numéro de téléphone, la civilité de la personne (nom et prénom), une adresse postale, un mail et un numéro de téléphone. Afin de faciliter le suivi de la sémantique d'évaluation, nous faisons des hypothèses sur les moyens d'authentification et les facteurs utilisés dans les sections suivantes.

4.2 Évaluation du niveau de risque de souscription d'une entité frauduleuse

La figure 4.2 décrit les paramètres d'entrée pour l'évaluation du niveau de risque de souscription d'une entité frauduleuse. Principalement on considère le mécanisme de vérification des attributs d'identité lors de la phase d'enregistrement d'une nouvelle entité. Ce mécanisme permet de vérifier le respect ou non des exigences #Ex1, #Ex2, #Ex3 définies dans la Section 3.3. De plus on différencie le niveau de risque associé aux attributs relatifs à l'identité numérique et ceux se référant à l'identité physique.

4.2.1 Évaluation du niveau de risque associé aux attributs d'identité numérique

Il n'y a pas de consensus commun sur la définition d'une identité numérique dans la littérature. Ainsi, nous nous référons à la définition du Nist d'une identité numérique : la représentation par un ensemble d'attributs d'une identité physique sur un service digitalisé [79]. Ces attributs permettent d'identifier l'entité dans un groupe et ne permettent pas de prouver l'existence "physique" de cette entité. Ainsi, l'objectif des exigences de sécurité (*i.e.*, #Ex1, #Ex2, #Ex3 de la Section 3.3.1) est de garantir les propriétés d'existence ("objective"), d'unicité, de validité et d'appartenance des attributs d'identité numérique de l'entité déclarée. L'adresse e-mail demandée à la

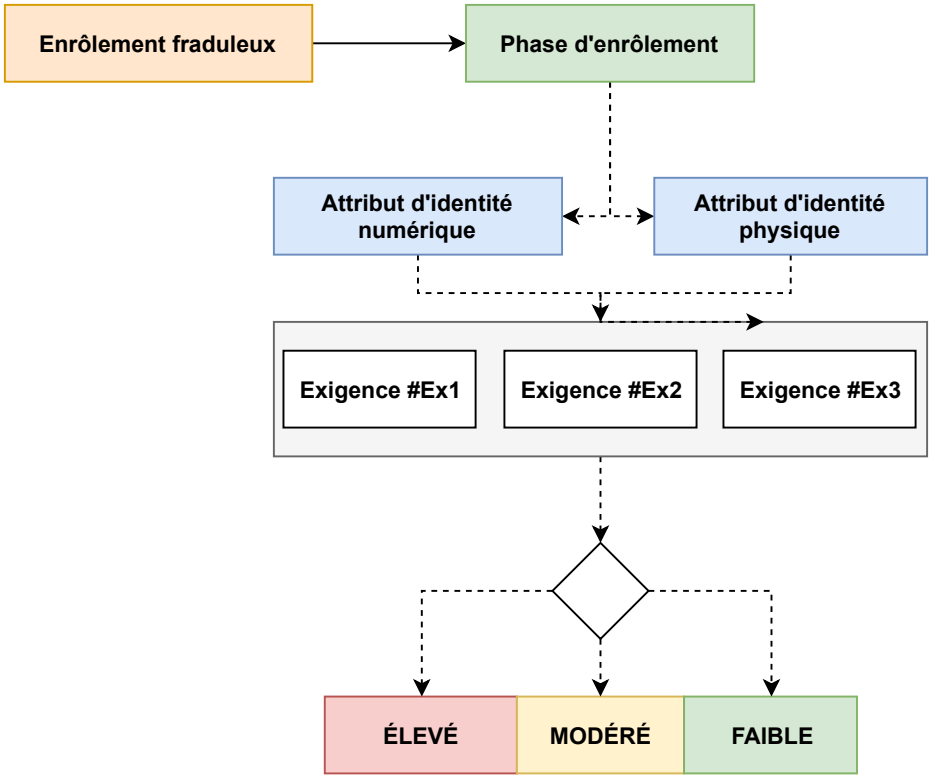

Figure 4.2 – Les paramètres d'entrée de l'évaluation du niveau de risque de souscription d'une entité frauduleuse.

souscription de l'exemple illustratif est un exemple d'attribut d'identité numérique (*cf.* Section 4.1.2).

Niveau de risque ÉLEVÉ

Le niveau de risque d'une souscription frauduleuse est *ÉLEVÉ* s'il n'existe pas de mécanisme de vérification d'identité ou si les mécanismes mis en œuvre sont insuffisants pour garantir à la fois les propriétés d'unicité, d'existence, de validité et d'appartenance.

> En considérant l'adresse mail de l'exemple illustratif, le niveau de risque est évalué à *ÉLEVÉ* si le service ne vérifie pas que l'e-mail est valide et existe (*e.g.*, vérification du format et du fournisseur), qu cette adresse est unique (*e.g.*, une entité n'a pas fourni cette adresse auparavant) et qu'elle appartient à l'entité l'ayant fournie (*e.g.*, envoi d'un message de confirmation).

Niveau de risque MODÉRÉ

Le niveau de risque est évalué à *MODÉRÉ* si au moins il y a une évidence sur le non respect d'une des exigences (*i.e.*, #Ex1 : unicité, #Ex2 : validité et existence, #Ex3 : appartenance).

> Toujours en considérant l'adresse mail, une évaluation à *MODÉRÉ* peut être causée soit par le fait que plusieurs utilisateurs peuvent souscrire avec la même adresse mail (*i.e.*, absence d'unicité), ou bien par une absence de la validité et existence (*i.e.*, un utilisateur peut souscrire avec un mail frauduleux ou inexistant), ou encore par une absence de vérification de l'appartenance (*i.e.*, une personne peut souscrire avec le mail d'une autre personne).

Niveau de risque FAIBLE

Le niveau de risque d'une souscription frauduleuse est évalué à *FAIBLE* si toutes les exigences sont satisfaites. Cependant un message d'alerte sera envoyé au concepteur dans le cas suivant :

Si, malgré la satisfaction des exigences, les mécanismes mis en place sont faibles ou influençables par les éléments de contexte ou le comportement de l'utilisateur final. Par exemple :

— s'il y a fédération d'identité : en effet, la compromission de l'autorité de confiance (*i.e.*, le IdP) est "susceptible" de conduire à une souscription d'une entité sans l'accord de l'entité légitime ;

— s'il y a une forte corrélation entre les facteurs intervenant dans la vérification. Par exemple, la vérification de l'appartenance d'un numéro de téléphone ou d'une adresse mail via l'envoi d'un OTP par mail ou SMS. Dans ce cas, l'accès au mail ou au SMS de l'entité peut conduire à une souscription frauduleuse.

> Ce dernier niveau peut résulter de la perte ou du vol du téléphone d'une personne. En effet, si l'application de partage autorise une souscription via une délégation d'identité (*i.e.*, avec Facebook par exemple) et que le compte Facebook de la victime est actif durant ce vol, alors une personne malicieuse pourrait souscrire au service au nom de la victime. On aura les mêmes conséquences avec la vérification de l'adresse mail si le compte mail de la victime est actif au moment de la perte du téléphone mobile [177].

4.2.2 Évaluation du niveau de risque associé aux attributs d'identité physique

Un attribut de l'identité physique est une information intrinsèquement liée à l'identité physique de la personne. Dans ce cadre, nous nous focalisons sur les attributs

d'identité qui participent à la logique décisionnelle de l'application (*i.e.*, le permis de conduire dans l'exemple illustratif). Les informations déclarées doivent ainsi être conformes à l'identité physique : par exemple, l'état civil (*e.g.*, nom, prénom), doit être conforme aux informations du permis de conduire.

Par ailleurs, nous précisons que les informations faisant foi sont celles consignées dans les documents officiels. En effet, les attributs d'identité sont déclaratifs sur la plupart des systèmes digitalisés et peuvent être modifiés à tout moment. Par conséquent, la vérification doit porter sur les documents officiels.

Niveau de risque ÉLEVÉ

Pour les mêmes raisons qu'une preuve d'identité numérique, le niveau de risque est *ÉLEVÉ* lorsque les exigences #Ex1, #Ex2 et #Ex3 ne sont pas satisfaites.

> Avec l'exemple illustratif, cela signifie que la plateforme ne vérifie pas que le permis de conduire est associé à une seule personne, qu'il est valide et existe (*e.g.*, date de validité, permis non falsifié ?) et qu'il appartient à la personne souscriptrice (*e.g.*, vérification physique avec la photo, demande d'informations supplémentaires).

Niveau de risque MODÉRÉ

Le niveau de risque d'une souscription frauduleuse est *MODÉRÉ* dans les cas de figure ci-après.

— Si l'une ou l'autre des propriétés d'unicité ou de validité n'est pas satisfaite. La propriété d'unicité est assurée par l'application tierce en fonction de son contexte. Par exemple vérifier qu'une adresse e-mail a déjà été fournie ou non par une autre entité. Tandis que la propriété de validité et d'existence s'appuie sur l'autorité émettrice de la preuve d'identité. Par exemple, la validité d'un permis de conduire peut se faire en vérifiant la date de validité. Alors que la véracité (est-ce que le document n'est pas falsifié) de ce permis de conduire ne peut être assurée que par l'autorité émettrice. Dans ce cadre, on considère que l'existence #Ex2 (*i.e.*, validité et existence) est satisfaite s'il existe un canal sécurisé garantissant l'intégrité et l'authenticité de la réponse de l'autorité de confiance émettrice (suite à une sollicitation de vérification).

> L'application de partage pourra vérifier la validité et l'existence du permis de conduire via une plate-forme sécurisé fournie par l'émetteur de ce permis de conduire ou par une autorité de confiance mandatée par l'émetteur.

— Si la propriété d'appartenance n'est pas satisfaite ou réalisée à distance. Prouver qu'une preuve d'identité se rapporte à l'entité qui la présente est difficile

dans le contexte des applications Web/Mobile du fait que ces services sont accessibles à distance. En effet, ce processus nécessite une identification formelle de l'individu via un attribut unique dont seul celui-ci est capable de fournir. Afin de simplifier le processus d'évaluation, on considérera cette exigence satisfaite pour les cas ci-après.

— Si la preuve d'identité dispose de l'image de l'individu. Alors, la vérification doit se faire en physique afin de confirmer l'identité de l'entité souscriptrice. Une communication via un canal WebRTC [1] (ou équivalent) sécurisé avec un flux vidéo suffisant pour identifier l'entité, la preuve d'identité et la photo de celle-ci est acceptée dans ce cadre.
— Si la preuve d'identité ne dispose pas de l'image de l'individu. Alors, l'individu devra confirmer son identité avec au moins deux preuves d'identité équivalentes (*i.e.*, les preuves à fournir sont définies par politique de sécurité de l'application). Ces preuves doivent provenir d'autorités de confiance différentes.

Pour la propriété d'appartenance, un appel vidéo ou un système similaire permettant d'identifier formellement la personne et la photo du document d'identité seraient nécessaires pour confirmer l'appartenance.

Niveau de risque FAIBLE

Nous considérons un niveau de risque *FAIBLE* pour les cas ci-après.

— Si toutes les exigences sont satisfaites, mais que la vérification est faite à "distances" pour une preuve d'identité vérifiable ou non localement.
— Si la vérification de la propriété d'appartenance est réalisée localement par un agent susceptible d'être compromis. En effet, un agent complice, une erreur d'appréciation ou d'identification peut toujours conduire à valider l'identité d'une personne non légitime.

Dans l'exemple illustratif, l'envoi d'une photo de la personne via un canal sécurisé pour confirmer son identité est évalué à ce niveau. Car même si la personne envoie une photo d'elle même, il y a toujours un risque qu'il ait eu accès aux photos de la victime. Un agent complice peut aussi aider à valider une identité frauduleuse.

1. https://fr.wikipedia.org/wiki/WebRTC

4.2.3 Récapitulatif de l'évaluation du niveau de risque de souscription d'une entité frauduleuse

Principalement, l'évaluation du niveau de risque s'appuie sur la fonction de vérification des attributs d'identité. La Table 4.2 récapitule le processus d'évaluation qui, indépendamment de la nature de l'attribut, se base sur le respect ou non des exigences de sécurité.

Nous précisons que chaque attribut est évalué de manière unitaire, ainsi, il incombe au concepteur d'en déduire les mesures à adopter en fonction de la politique de l'entreprise.

Niveau de risque	Critères d'évaluation	Description
ÉLEVÉ	$\neg\#Ex1 \wedge \neg\#Ex2 \wedge \neg\#Ex$	Si aucune exigence n'est satisfaite
MODÉRÉ	$\neg\#Ex1 \vee \neg\#Ex2 \vee \neg\#Ex3$	Si au moins une des exigences n'est pas satisfaite
FAIBLE	$\#Ex1 \wedge \#Ex2 \wedge \#Ex3$	Si toutes les exigences sont satisfaites

Table 4.2 – Récapitulatif de l'évaluation du niveau de risque d'une souscription d'une entité frauduleuse.

4.3 Évaluation du niveau de risque d'un accès non autorisé

Cette section évalue le risque qu'un défaut sur le parcours d'authentification puisse conduire à un accès non autorisé conduisant à une imitation de l'entité légitime ou à un vol d'identité. La Figure 4.3 décrit les différents facteurs sur lesquels nous nous focalisons pour évaluer ce risque. Particulièrement, nous évaluons les mécanismes d'authentification des phases de login et de récupération de preuves de sécurité. En effet la finalité des ces phases est d'accéder au service, ainsi elles sont directement visées par les exploitations logiques. En plus de ces facteurs, nous considérons les éléments extérieurs qui peuvent influencer ces mécanismes d'authentification.

Le mécanisme d'authentification étant l'élément central de cette évaluation, nous présentons d'abord l'évaluation unitaire des différents moyens d'authentification de ce cadre et l'impact des éléments de contexte sur ces moyens d'authentification. Ensuite, nous présentons l'évaluation du niveau de risques liés à chacune des deux phases considérées. L'exemple illustratif de la Section 4.1.2 permettra d'illustrer le processus d'évaluation.

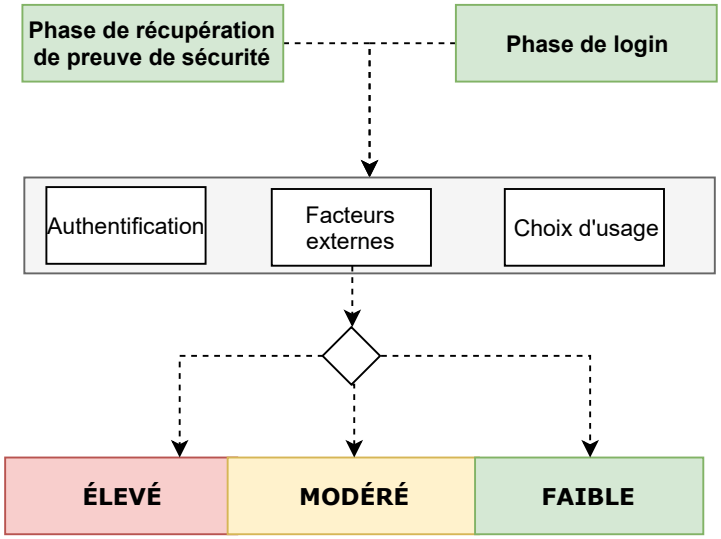

Figure 4.3 – Paramètres d'entrée du processus d'évaluation du niveau de risque d'accès non autorisé

4.3.1 Authentification à un facteur

Le mécanisme d'authentification est le moyen (*i.e.*, protocole, méthode, etc.) utilisé pour garantir l'authenticité d'une entité auprès d'un vérifieur [79]. Il se base sur des facteurs d'authentification qui sont souvent classés en trois catégories :

- facteur mémoriel (*i.e.*, quelque chose connue par l'entité) : mot de passe, code pin ;
- facteur de possession (*i.e.*, quelque chose que possède l'entité) : carte à puce ;
- facteur biométrique ou comportemental (*i.e.*, quelque chose que l'entité est ou sait faire) : empreinte digital, manière de marcher.

Une authentification à un facteur utilise un seul des facteurs précités pour authentifier une entité. Le niveau de risque d'un mécanisme d'authentification à un facteur est proportionnel à l'évaluation unitaire du facteur concerné et des éléments de contexte influant sur le mécanisme.

Facteur mémoriel

Les facteurs mémoriels tels que le code pin ou le mot de passe sont les plus utilisés dans les mécanismes d'authentification des applications Web/Mobile. Ils se déploient facilement et offrent un niveau de sécurité satisfaisant en fonction des besoins. Cependant, ils comportent certaines limites liées à la capacité des personnes à retenir des séquences de chiffres et de lettres [180, 22]. Ces limites et la facilité d'usage ont tendance à favoriser le choix de secrets facilement devinables par une entité malicieuse [1, 151]. Par ailleurs, des choix de conception tels que la limitation ou non du

nombre de tentative d'authentification (*cf.* Section 3.3.2) peuvent influencer le niveau de sécurité global.

Pour ces facteurs mémoriels, nos critères d'évaluation sont le niveau d'entropie ou la capacité du secret à être deviné [121] et la mise en place ou non d'un système anti brute-force [2].

Niveau de risque ÉLEVÉ Le niveau de risque est *ÉLEVÉ* si les deux affirmations ci-après sont vraies :

- si la qualité du secret est faible (*e.g.*, mot de passe faible, question d'ordre générale) ;
- et, s'il n'y a pas de système anti brute-force en ligne.

Niveau de risque MODÉRÉ Le niveau de risque est *MODÉRÉ* si l'un ou l'autre des cas ci-après est vrai :

- s'il y a absence de système anti brute-force indépendamment de la qualité du secret ;
- ou, si le secret est fondé sur des connaissances générales ou des préférences (*e.g.*, date de naissance, nom de jeune fille) facilement devinable par ingénierie sociale [20, 142].

Niveau de risque FAIBLE Le niveau de risque est *FAIBLE* si l'ensemble des mesures ci-après sont satisfaites par le mécanisme d'authentification.

- Le secret respecte un niveau d'entropie élevé (*i.e.*, mot de passe solide, pass-phrase, etc)
- Le mécanisme d'authentification implémente une mesure empêchant les attaques brute-force (*i.e.*, limitation du nombre de tentative, détection d'agissement robotisé).

Il est important de préciser que l'étude de la qualité d'un secret mémoriel (*e.g.*, mot de passe, code pin, pass-phrase) n'est pas dans le cadre de cette thèse [121, 20]. De même, beaucoup de contraintes sur la création d'un secret peuvent constituer un frein à l'utilisabilité d'un système et conduire l'utilisateur final à recourir à des méthodes qui affaiblissent la sécurité du système (*e.g.*, réutilisabilité du même mot de passe, mot de passe sur des post-it) [151, 180, 22, 96].

Facteur de possession

Un facteur de possession est un dispositif matériel (*e.g.*, badge d'identification) ou un ensemble d'informations stockées dans un dispositif matériel (*e.g.*, carte à puce stockant des preuves d'identité) servant à authentifier de manière unique une entité auprès d'un vérificateur (*i.e.*, un dispositif d'acquisition ou un individu). Une authentification fondée sur un facteur de possession se fonde sur la détention et le contrôle

Niveau de risque d'une authentification à facteur mémoriel	
ÉLEVÉ	Le risque est élevé si en plus d'un secret faible ou facilement devinable par ingénierie sociale, le mécanisme n'a pas de système anti force-brute.
MODÉRÉ	Le risque est modéré en cas d'absence de système anti force-brute indépendamment de la qualité du secret ou si le secret est faible ou facilement devinable malgré un système anti force-brute.
FAIBLE	Si le secret est difficile à deviner grâce à un la qualité ou le niveau d'entropie avec l'implémentation d'un mécanisme anti brute-force.

Table 4.3 – Résumé de l'évaluation du niveau de risque associé à une authentification à un facteur

de ce dispositif par une entité déclarante. Plusieurs risques sont liés à ce facteur d'authentification : la copie du dispositif (ou des preuves contenues dans le dispositif), la perte ou le vol de celui-ci. Le risque que le dispositif d'authentification soit perdu ou volé par une entité malicieuse est indépendant de la volonté du concepteur du système d'authentification. De même, calculer la probabilité d'occurrence de ces évènements est très difficile ; plusieurs facteurs non déterministes sont à considérer. Pour ces raisons, nous évaluons le risque associé à ce type d'authentification de MODÉRÉ.

Facteur biométrique ou comportemental

Un facteur biométrique ou comportemental permet d'authentifier une personne via un attribut biométrique ou via un comportement unique [176, 17]. L'authentification biométrique a longtemps été réservée aux systèmes critiques nécessitant un niveau de sécurité élevé (*i.e.*, la défense, la finance, la santé, etc). Par ailleurs, elle a aussi souffert de la réticence du grand public pour des raisons de préservation de leur vie privée [169]. Cependant, l'avènement de dispositifs d'acquisition plus compactes et acceptables, notamment les téléphones portables, a contribué à démocratiser ces mécanismes d'authentification pour le grand public.

Les critères pertinents pour l'évaluation de la performance d'un système biométrique sont le FRR et le FAR pour un seuil d'acceptation et un dispositif d'acquisition donné. Cependant, ces valeurs ne sont pas communiquées par les constructeurs, ce qui rend difficile l'évaluation du niveau de risque d'un mécanisme d'authentification fondé sur ces facteurs.

Dans le cadre de cette étude, nous nous focalisons que sur deux modalités qui sont les plus utilisées dans le contexte des applications Web/Mobile : la reconnaissance faciale sans distinction de l'iris ou de la forme du visage [169]) et la reconnaissance de l'empreinte digital.

De plus, nos critères d'évaluation inclus le dispositif (*e.g.*, le modèle du téléphone, modalité utilisé) car la fiabilité d'un système d'authentification biométrique est fortement corrélée au dispositif d'acquisition [54]. À titre d'exemple, la Table 4.4 récapitule une étude sur la reconnaissance faciale de 60 téléphones dont 26 se font compromettre uniquement par une photo de la personne légitime (La liste complète est disponible ici[2]. Ainsi, pour les modalités biométriques, il serait pertinent par exemple, pour l'empreinte digitale d'évaluer la capacité du dispositif à accepter une fausse empreinte (*i.e.*, FAR) ou l'empreinte digitale de la personne légitime capturée ou falsifiée [179]. Il en est de même, pour la reconnaissance faciale, on pourrait évaluer la capacité d'acceptation d'un faux visage ou d'une photo de la personne légitime. Ne disposant pas de ces informations, par défaut, on évalue une modalité biométrique de sûre tout en alertant le concepteur sur ces risques éventuels.

Téléphone déverrouillé avec une photo	Téléphone ayant passé le test
ASUS Zenfone 5	Iphone X/XS/Max
Samsung Galaxy A7 - A8 (2018)	Samsung Galaxy S8/S9/S9+
Sony Xperia XZ3	HTC U12
Xiaomi Mi A2	OPPO Find X
Nokia 7.1	Huawai Psmart/Mate20/20Pro
Honor 7A	OnePlus 5T/6

Table 4.4 – Liste de téléphones ayant passé le test de déverrouillage avec une photo de l'utilisateur légitime.

4.3.2 Évaluation du niveau de risque associé à un mécanisme d'authentification à plusieurs facteurs

L'évaluation d'un moyen d'authentification multi-facteurs est une combinaison de tous les facteurs pondérés de l'impact des éléments de contexte. Les éléments à prendre en considération sont : le canal utilisé pour transmettre les secrets (*i.e.*, est-ce que les dispositifs utilisé sont corrélés) et le type de validation (*i.e.*, validation locale ou à distance).

Considérons le mécanisme d'authentification multi-facteurs défini ci-après :

$$\zeta = f_1 + f_2 + ... + f_n.$$

2. https://www.consumentenbond.nl/veilig-internetten/gezichtsherkenning-te-hacken

Avec ζ le mécanisme d'authentification et f_i les facteurs d'authentification. Ci-après l'évaluation du niveau de risque de compromission d'un tel mécanisme.

— Si un des facteurs f_i est évalué avec un niveau de risque *FAIBLE,* alors ζ est évalué à un niveau de risque *FAIBLE* .

$$\forall i \in \{1..n\} \textit{ si } \exists E(f_i) = \textit{ FAIBLE } \textit{ alors } E(\zeta) = \text{ FAIBLE}$$

Avec $E(x)$ l'évaluation du niveau de risque du facteur x.

— Si toutes les $E(f_i)$ sont différentes de *FAIBLE* , alors s'il existe un f_i évalué à *MODÉRÉ* alors ζ est au pire évalué à *MODÉRÉ* et tendra vers un niveau de risque *FAIBLE* .

$$\forall i \in \{1..n\} E(f_i) \neq \textit{ FAIBLE } \text{ alors}$$
$$\textit{if } \exists E(f_i) = \textit{ MODÉRÉ } \textit{ alors MODÉRÉ} \leq E(\zeta) \leq \textit{ FAIBLE}$$

— Si toutes les évaluations des f_i sont différentes de *FAIBLE* et *MODÉRÉ* , alors s'il existe un f_i évalué à *ÉLEVÉ* alors ζ est au mieux évalué à *MODÉRÉ* .

$$\forall i \in \{1..n\} E(f_i) \neq \textit{ FAIBLE } \wedge E(f_i) \neq \textit{ MODÉRÉ } \textit{ alors}$$
$$\textit{ÉLEVÉ} \leq E(\zeta) \leq \textit{ MODÉRÉ}$$

D'autres critères sont à considérer dans l'évaluation du niveau de risque global d'un système d'authentification à plusieurs facteurs : le coefficient de corrélation entre les facteurs et le dispositif et le type de validation.

Le coefficient de corrélation Le coefficient de corrélation est le niveau de dépendance entre les éléments d'une authentification multi-facteurs (*i.e.,* les facteurs et le dispositif d'acquisition). Dans ce cadre, on considère qu'il y a une forte corrélation si le dispositif d'acquisition des secrets représente ou supporte le dispositif de vérification de ces mêmes secrets. Nous pouvons citer l'exemple d'une authentification fondée sur un mot de passe et un SMS OTP. Par ailleurs, la corrélation est faible si la compromission de l'un des acteurs n'entraîne pas la compromission des autres acteurs.

> Par exemple, si on considère une authentification à deux facteurs (*i.e.,* mot de passe et OTP) dans notre exemple illustratif. Il y a une forte corrélation si l'OTP est envoyé par SMS car il est probable que l'application SMS et l'application exigeant l'authentification soient accessibles dans le même téléphone. Alors que la corrélation est inexistante si l'OTP est généré via un dispositif à part.

Ci-après l'évaluation du niveau de risque avec le coefficient de corrélation.

$$\text{Soit : ÉLEVÉ} \leq E(\zeta) \leq \text{MODÉRÉ} = \begin{cases} E(\zeta) \rightarrow \text{MODÉRÉ}, & \textit{si } c = 0 \\ E(\zeta) \rightarrow \text{ÉLEVÉ}, & \textit{si } c \rightarrow 1 \end{cases}$$

Avec c le coefficient de corrélation et $E(\zeta)$ l'évaluation initiale du mécanisme d'authentification. Concrètement, s'il y a une corrélation alors le niveau de risque correspond au niveau du facteur le plus bas. Alors qu'en absence de corrélation, le niveau de risque tend vers le niveau de risque inférieur au niveau le plus bas. Par exemple, dans le cas du mot de passe et de l'OTP, si l'on évalue les deux facteurs à *MODÉRÉ*, alors le niveau de risque est évalué à *MODÉRÉ* s'il y a une corrélation, alors que le niveau tendra vers *FAIBLE* en l'absence de corrélation.

Le type de validation Le type de validation peut aussi contribuer à augmenter ou à réduire le niveau de risque. Deux types de validation sont à prendre en compte : la validation en locale et la validation à distance. La validation est locale, si le processus d'authentification multi-facteurs est validé sur le canal de départ. Autrement dit, si la personne valide explicitement l'authentification sur le canal de départ : confirmation de la réception du code à usage unique sur le canal de départ (*i.e.*, l'interface de l'application ou du dispositif d'acquisition). Par ailleurs, une validation à distance se fait via un canal différent : la validation d'un code à usage unique via une application ou un dispositif tierce.

> Dans l'exemple illustratif, si l'OTP doit être renseigné sur la plateforme avant de valider la transaction alors la validation est locale. Dans le cas contraire où celle-ci est validée via un autre canal (*e.g.*, via une autre application par exemple), alors on a une validation à distance.

Dans ce cadre, on considère la validation en locale plus sûre du fait de sa capacité à éviter les erreurs de validation par inadvertance et les attaques transposées (*cf.*, Section 3.1.1). Ci-après l'évaluation avec prise en compte du type de validation.

$$\text{Soit : } \textbf{ÉLEVÉ} \leq E(\zeta) \leq \textbf{MODÉRÉ} = \begin{cases} E(\zeta) \rightarrow \textbf{MODÉRÉ}, & \textit{si } v = locale \\ E(\zeta) \rightarrow \textbf{ÉLEVÉ}, & \textit{si } v = distant \end{cases}$$

Avec v le type de validation et $E(\zeta)$ l'évaluation initiale du mécanisme d'authentification.

4.3.3 Évaluation du niveau de risque associé à une fédération d'identité

La fédération d'identité permet à une application tierce de déléguer la gestion des preuves d'identité à une autorité de confiance : le fournisseur d'identité. Dans ce cadre, nous nous focalisons sur la délégation d'authentification qui consiste à déléguer le mécanisme d'authentification à un fournisseur d'identité. Nous ne présentons pas le processus d'évaluation d'un tel mécanisme d'authentification. En effet, cela revient à évaluer le parcours d'authentification du tiers de confiance avec la sémantique décrite jusqu'ici.

4.3.4 Niveau de risque d'un accès non autorisé : phase de login

Cette section présente l'évaluation du niveau de risque associé à la phase de login. Cette évaluation prend principalement trois facteurs en entrée : Le mécanisme d'authentification, le

type de session créée et la présence ou non de formulaires automatiquement remplis ou leurs équivalences pour les facteurs mémoriels.

L'évaluation des mécanismes d'authentification étant présentée dans les Sections 4.3.1 et 4.3.2, le reste de la section se focalise sur l'impact des autres facteurs. En effet, si l'application implémente une durée de session longue ou un formulaire de login préalablement remplissable, alors le niveau de risque sera évalué tel une authentification fondée sur un facteur de possession (*cf.* Section 4.3.1). Cela est dû au fait que l'accès au dispositif permet d'accéder au service sans connaissance préalable des preuves de sécurité. Dans l'étude, il faudrait prendre en compte la probabilité d'occurrence que le dispositif (*e.g.*, le téléphone) soit entre les mains d'une personne malicieuse. Cette probabilité étant difficile à évaluer, nous faisons le postulat que celui-ci est accessible par la personne malicieuse. Ainsi, si le dispositif est protégée, alors l'évaluation est donnée par le mécanisme de protection du dispositif. Dans le cas contraire, le niveau de risque est est évalué à *ÉLEVÉ* car accéder au dispositif garanti l'accès au compte de l'utilisateur (*cf.* Figure 4.4).

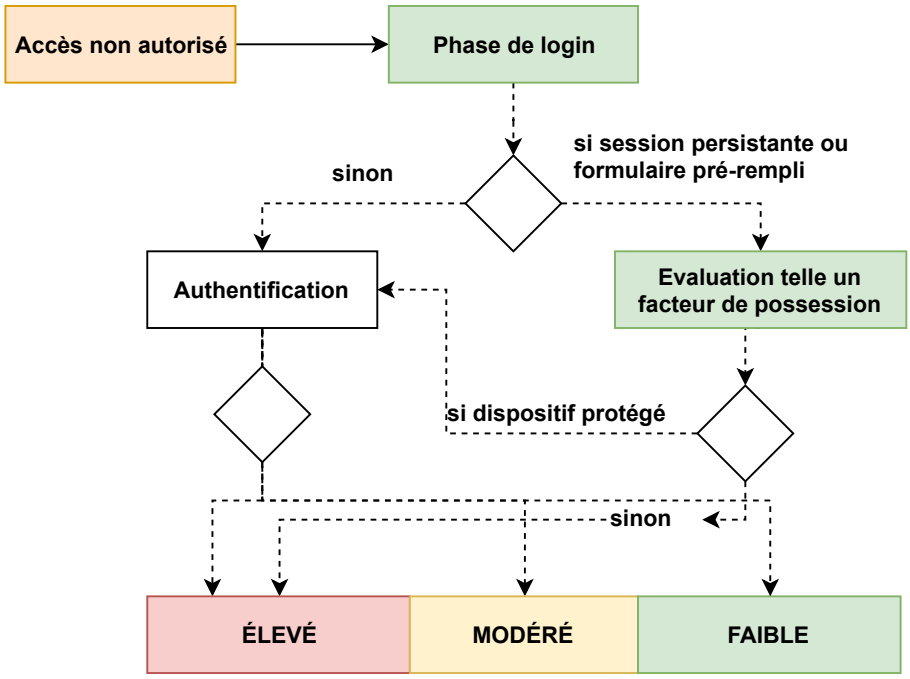

Figure 4.4 – Arbre de décision de l'évaluation du niveau de risque d'un accès non autorisé associé à la phase de login

4.3.5 Niveau de risque d'un accès non autorisé : phase de récupération d'une preuve d'identité

En ce qui concerne la phase de récupération de preuves de sécurité, nous nous focalisons sur les protocoles ci-après car ils sont représentatifs de la pratique actuelle dans le domaine des applications Web/Mobile.

— L'envoie de la preuve d'identité ou des instructions permettant sa récupération ou sa mise à jour par mail ou par SMS.
— L'envoie de la preuve d'identité ou des instructions permettant sa récupération ou sa mise à jour à une adresse postale.
— La récupération de la preuve ou de sa mise à jour directement sur l'interface du service concerné.

Ces trois protocoles peuvent nécessiter un challenge de sécurité (*i.e.*, la méthode d'authentification) avant d'autoriser la récupération ou la mise à jour des preuves de sécurité. On précise que la récupération d'une même preuve d'identité oubliée ou préalablement révoquée induit des défauts de sécurité et des problèmes de protection de la vie privée qui ne sont pas traités dans le cadre cette thèse (*e.g.*, accès des mot de passes des utilisateurs en clair par l'application tierce). En effet, nous nous focalisons sur les failles logiques liées au processus et non sur le stockage et la gestion des preuves de sécurité. Nous faisons le même postulat que le dispositif est accessible par une personne malicieuse. De ce fait, le processus d'évaluation se focalise sur le protocole utilisé et sur la présence ou non d'un mécanisme d'authentification.

Ainsi, on peut constater sur la Figure 4.5 que l'évaluation est considérée comme une évaluation à un facteur de possession si le dispositif est protégé en l'absence de challenge. Dans le cas contraire, on évalue le processus tel une authentification à deux facteurs constituée du challenge et du mécanisme de protection du dispositif. Enfin, si le protocole est local, alors l'évaluation est donnée par le challenge de sécurité, car le changement de preuves de sécurité peut se faire sur n'importe quel autre dispositif.

Il est important de préciser que dans ce cadre, nous faisons le postulat que les applications clientes pour le mail ou le SMS sont actives dans le dispositif de la personne légitime. De même, nous considérons que la boîte postale peut-être compromise facilement (*e.g.*, échange des étiquettes nominales pendant une durée limitée).

4.3.6 Processus d'évaluation globale du niveau de risque d'un accès non autorisé

Le niveau de risque d'un accès non autorisé dépend de l'évaluation de la phase de login (*cf.* Section 4.3.4) et de la phase de récupération des preuves de sécurité (*cf.* Section 4.3.5) car ces deux phases peuvent être exploitées pour le même but : accès non autorisé à un service.

Pour illustrer cette évaluation avec notre exemple de la plateforme de partage de voitures. Il y aurait un niveau de risque élevé d'un accès non autorisé si par exemple l'application autorise un changement de mot de passe localement sans demander un challenge de sécurité ou bien si une personne perd son téléphone sans protection avec l'application de partage active au moment de la perte. Le niveau de risque est modéré si le téléphone est protégé. En effet en fonction de la marque du téléphone et de la modalité utilisée, ce niveau de risque peut s'améliorer ou empirer (*cf.* Section 4.3.1). Enfin, le niveau de risque est faible voire inexistant, si le mot de passe est difficile à deviner et que l'application n'autorise pas des sessions persistantes.

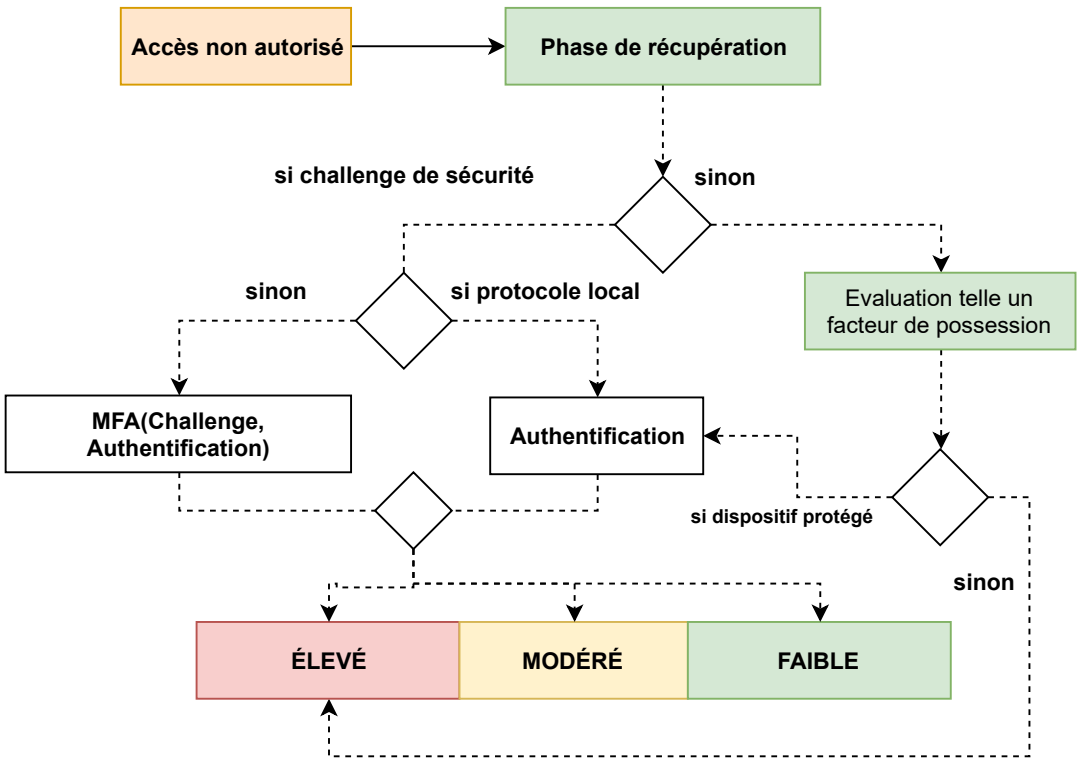

Figure 4.5 – Arbre de décision de l'évaluation du niveau de risque d'un accès non autorisé via la phase de récupération de preuves d'identité

La Table 4.5 récapitule l'évaluation globale. La valeur "X" indique que la case correspondante n'est pas concernée.

- $Auth(m)$: Méthode d'authentification de la phase correspondante
- $E(m)$: Évaluation du niveau de risque associé au mécanisme d'authentification utilisé (m)
- $E(l)$: Évaluation du niveau de risque associé au mécanisme de protection du dispositif (l) (*cf.* Section 4.3.1)
- $MFA(m, l)$: Évaluation multi-facteurs du niveau de risque associé à m et l(*cf.* Section 4.3.2)

4.4 Usurpation d'identité : substitution de l'entité légitime

La substitution est différente de l'imitation de l'entité légitime. En effet, le but de l'imitation est de se passer de l'utilisateur légitime en accédant à un service non autorisé, tandis que la substitution cherche à voler l'identité de celui-ci temporairement ou définitivement.

Facteurs	Dispositif non protégé		Dispositif protégé		Phase
	$Auth(m)$	$\neg Auth$	$Auth(m)$	$\neg Auth$	
Mise à jour via mail ou SMS	$E(m)$	*ÉLEVÉ*	$MFA(m,l)$	$E(l)$	Récupération
Mise à jour en locale	$E(m)$	*ÉLEVÉ*	X	X	
Session limitée et formulaire à remplir	$E(m)$	X	$MFA(m,l)$	X	*Login*
Session persistante ou formulaire auto-rempli	*ÉLEVÉ*	X	$E(l)$	X	

Table 4.5 – Évaluation globale du niveau de risque d'un accès non autorisé (*i.e.*, Imitation de l'entité légitime).

Dans l'exemple illustratif, il y a imitation si une personne malicieuse gagne un accès non autorisé à la plateforme. Cependant, la substitution serait qu'une personne légitime en plus de perdre son accès, n'arrive plus à révoquer ou à changer ses preuves de sécurité. Elle perd ainsi le contrôle de son identité au détriment de la personne malicieuse. Cela peut se faire en changeant le mot de passe de la personne légitime dans l'application, l'empêchant ainsi d'accéder au service.

Dans ce cadre, deux défauts conduisent à une potentielle substitution de l'entité légitime : la souscription d'une entité frauduleuse et le changement des preuves de sécurité. Les risques d'une souscription frauduleuse sont discutés dans la Section 4.2. Le changement des preuves de sécurité est relatif à la phase de changement ou de mise à jour des preuves de sécurité. Du fait que la phase de changement des preuves de sécurité est accessible que lors qu'une session est valide, nous faisons l'hypothèse que l'entité malicieuse a déjà réussi une imitation de l'entité légitime. Par conséquent, le mécanisme d'authentification mis en place lors de la mise à jour des preuves de sécurité permet d'évaluer le niveau de risque d'une substitution de l'entité légitime. Le niveau de risque est *ÉLEVÉ* s'il n'y a pas de mécanisme de sécurité ou si les formulaires sont auto-remplis. Sinon, le niveau de risque sera relatif aux mécanismes de sécurité mis en place.

4.5 Conclusion du chapitre

Ce cadre permet d'évaluer la capacité de la spécification d'un parcours d'authentification à prévoir les exploitations logiques pouvant conduire à la souscription d'une entité frauduleuse,

à l'imitation de l'entité légitime et à la substitution de celle-ci. Cette évaluation tire parti de la définition du parcours d'authentification et des concepts présentés dans le Chapitre 3 (*e.g.*, classes de défauts logiques, exigences de sécurité, etc.).

Nous avons ignoré les défauts logiques relatifs à l'utilisabilité (*i.e.*, blocage dans l'usage de l'application, déni de service intentionnel) dans l'évaluation des niveaux de risque car n'ayant pas d'impact sur la sécurité de l'utilisateur final. Cependant, leur prise en considération est fortement recommandée par les exigences du Chapitre 3.

La Figure 4.6 présente l'arbre de décision de l'évaluation globale des risques (*i.e.*, souscription frauduleuse, accès non autorisé et substitution de l'entité légitime) sur la spécification d'un parcours d'authentification. L'évaluation de certaines branches de l'arbre est influencée par d'autres facteurs (*i.e.*, éléments en ovale dans la Figure 4.6) qui nécessitent un postulat afin d'obtenir l'évaluation finale. La valeur de ces facteurs est discutée dans le chapitre suivant qui présente un langage dédié à la spécification d'un parcours d'authentification. En s'appuyant sur ce cadre, ce langage dédié permet d'évaluer automatiquement les niveaux de risques d'exploits logiques d'une spécification de parcours d'authentification donné. En ce sens, ce cadre apporte une partie des réponses de la question de recherche (*i.e.*, **RQ3**, Section 1.2) en définissant la sémantique d'évaluation du niveau de risques.

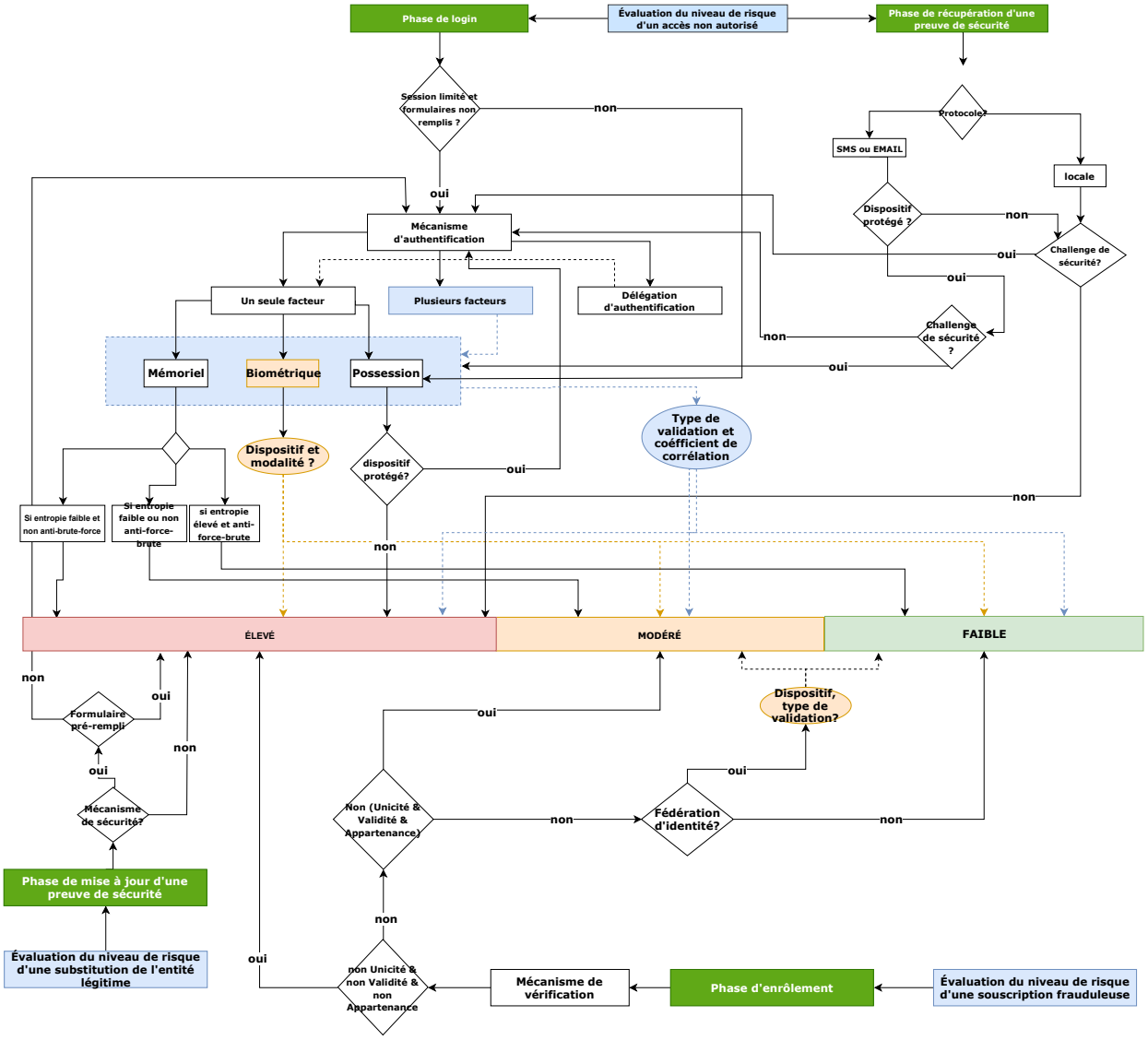

Figure 4.6 – Arbre de décision de l'évaluation des niveaux de risque associé aux différents phase du parcours au vu des exploitations logiques.

Chapitre 5

Environnement de modélisation du parcours d'authentification

Ce chapitre présente l'environnement de modélisation du parcours d'authentification. Cet environnement se fonde sur un langage dédié à la description de parcours d'authentification et permet d'évaluer de cette spécification. Le but de cet environnement est d'appliquer la sémantique d'évaluation du niveau de risque présentée dans le chapitre 4. Dans un premier temps, nous présentons les concepts fondamentaux qui ont permis la création du langage Ensuite nous présentons la syntaxe (abstraite et concrète) du langage. Enfin, nous détaillons le processus de description d'un parcours d'authentification, ainsi que l'évaluation des risques relatifs à cette spécification.

5.1 Concepts fondamentaux

Cette section présente les concepts fondamentaux sur lesquels reposent la sémantique du langage dédié. Dans les Chapitres 3 et 4, d'une part, nous avons démontré que les défauts logiques peuvent nuire à l'entité légitime et avons proposé des exigences pour prendre en compte ces défauts, d'autre part, nous avons présenté un cadre qui permet d'évaluer la capacité d'une spécification à éviter ou à prévenir les exploitations logiques.

L'environnement présenté dans ce chapitre contribue à ces travaux sur deux niveaux : 1), il offre un outil d'assistance qui permet aux concepteurs de spécifier les différents composants (*e.g.*, mécanismes d'authentification et d'identification, différentes phases, etc.) d'un parcours d'authentification. Ceci permet de profiter des mécanismes existants en les adaptant au contexte des applications Web/Mobile pour une meilleure prise en compte des défauts logiques liés aux facteurs externes et au comportement de l'utilisateur final. 2), il permet d'évaluer automatiquement les risques liés à cette spécification. Cette évaluation, en plus de donner le niveau de risque global d'un parcours d'authentification spécifié, permet aussi de sensibiliser le concepteur de ce parcours sur les moyens à adopter afin de prévenir ces risques.

La section suivante reprend l'exemple utilisé dans le chapitre précédent (*cf.*, Section 4.1.2) pour illustrer comment le DSL est utilisé pour spécifier un parcours d'authentification.

5.2 Processus de description d'un parcours d'authentification

Tout d'abord, il est important de préciser que le processus de description d'un parcours d'authentification doit respecter l'ordre ci-après :

1. décrire la phase d'enregistrement (*i.e.*, phase d'enrôlement) qui doit renseigner les attributs d'identités nécessaires pour le bon fonctionnement de l'application ;
2. décrire les facteurs d'authentification ;
3. décrire les moyens d'authentification qui doivent référencer les facteurs utilisés ;
4. enfin, décrire les phases du parcours sans ordre prédéfini.

L'ordre imposé permet de respecter les référencements croisés (*i.e.*, *cross-references*) entre les composants du parcours. Par exemple, un moyen d'authentification ne pourrait pas référencer un facteur d'authentification si celui-ci n'est pas déclaré préalablement.

Nous proposons d'utiliser l'application d'auto-partage utilisée dans le chapitre précédent (*cf.*, Section 4.1.2) afin d'illustrer la spécification d'un parcours d'authentification. Ci-après nous rappelons la description du parcours d'authentification de l'application.

— La souscription (*i.e.*, phase d'enrôlement) au service de partage de véhicule nécessite un permis de conduire, une adresse e-mail ou un numéro de téléphone. La civilité (*e.g.*, nom, prénom, etc) de la personne est déclarative, de ce fait les informations pertinentes, faisant fois en cas de préjudice, sont celles d'un document officiel (*i.e.*, le permis de conduire).

— Les moyens d'authentification possibles pour créer une session (*i.e.*, phase de *login*) sont : le couple identifiant/mot de passe (avec des contraintes sur le mot de passe) et la reconnaissance de l'empreinte digitale. Cette dernière étant réservée à la version mobile de l'application.

— Pour changer le mot de passe (*i.e.*, phase de mise de récupération), l'adresse mail est demandée en plus de la réponse à une question secrète.

— Enfin, un utilisateur déjà connecté peut changer à tout moment ses informations d'identité ainsi que ses informations de sécurité sans challenge de sécurité préalable (*i.e.*, phase de mise à jour).

Le Listing 5.1 présente la spécification correspondant à cette description.

```
1 Main : {
2     Registration :
3     {  // phase d'enrôlement
4         profile :
5         [
6             {
```

```
            "name" : email,  // L'adresse mail
            "povider" : SELF,
            "verification" : {"validity" : true, "unicity" : true, "binding" : true}
            },
             {
            "name" : phoneNumber,  // Le numéro de téléphone
            "povider" : SELF,
            "verification" : {"validity" : true, "unicity" : true, "binding" : true}
            },
            {
            "name" : driverLicense,  // Le permis de conduire
            "povider" : INSTITUTIONAL,
            "verification" : {"validity" : true, "unicity" : true, "binding" : true}
            }
        ]
    }
    Authenticators :
    [  // Liste des moyens d'authentification
        {
        "name" : password,   // Le mot de passe
        "value" : PASSWORD,
        "autifillable" : true,
        "limitedAttempts" : false
        },
        {
        "name" : secretQuestion,   // La question secrète
        "value" : PREFERENCES,
        "autofillable" : false,
        "limitedAttempts" : false
        },
        {
        "name" : fingerPrint,   // L'empreinte  digitale
        "value" : FingerPrint
        }
    ]

    Login :
    {   // Phase de Login
        "name" : mainLogin,
        "identifier" : email || phoneNumber,
        "authentication" : password || fingerPrint,
        "persistedSession" : true
    },

    Recovery :
    {  // Phase de récupération d'un mot de passe perdu ou oublié
        "name" : rec1
        "authenticator" : password ,
        "protocol" : Email-based || SMS-based ,
```

```
56          "authentication" : secretQuestion
57      }
58
59      Reset :
60      {  // Phase de mise à jour du mot de passe existant
61          "name" : r1,
62          "authenticator" : password
63      }
64  }
```

Listing 5.1 – Description de l'exemple illustratif

5.3 Concepts fondamentaux du DSL

Comme nous l'avons présenté dans la section précédente, ce DSL permet de décrire les facteurs d'authentification, les mécanismes d'authentification et les différentes phases d'un parcours d'authentification.

Il s'appuie sur les concepts de l'ontologie (*cf.*, Section 3.2.3) proposée dans le Chapitre 3. La figure 5.1 présente le métamodèle simplifié de celui-ci. Le métamodèle compl complet est présenté en fin de chapitre (*cf.*, Figure 5.3)

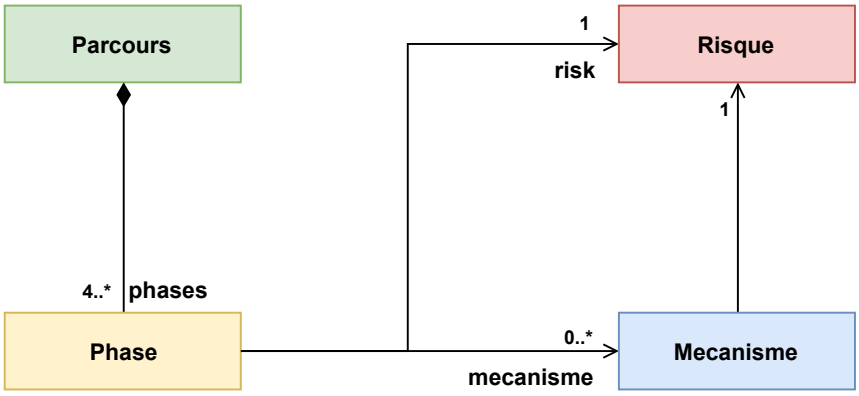

Figure 5.1 – Métamodèle simplifié du DSL

Fondamentalement, chaque composant (*i.e.*, le parcours, les phases, les mécanismes) comporte un niveau de risque évalué en se fondant sur le cadre d'évaluation du niveau de risque présenté dans le Chapitre 4. Les concepts d'attaques, d'exigences et de défauts logiques sont abstraits et ne sont pas représentés dans le métamodèle.

Pour évaluer le niveau d'un risque, le DSL se fonde sur la spécification des mécanismes (*i.e.*, vérification et authentification) étendus aux facteurs à l'origine des défauts logiques. Ainsi, en évaluant les différents facteurs d'un mécanisme, le DSL est capable de connaître les défauts logiques et d'en déduire les attaques possibles qui conduisent aux risques redoutés. Et comme les mécanismes appliquent les exigences, nous pouvons également en déduire les exigences correspondantes.

Nous avons utilisé EMF[1] (*Eclipse Modeling Framework*) pour implémenter le DSL dont l'environnement de modélisation est présenté dans Figure 5.2. Le fichier "main.light" représente le fichier de spécification et le fichier "eval.txt" le résultat de l'évaluation du risque de la spécification. La grammaire du langage est réalisée avec Xtext[2].

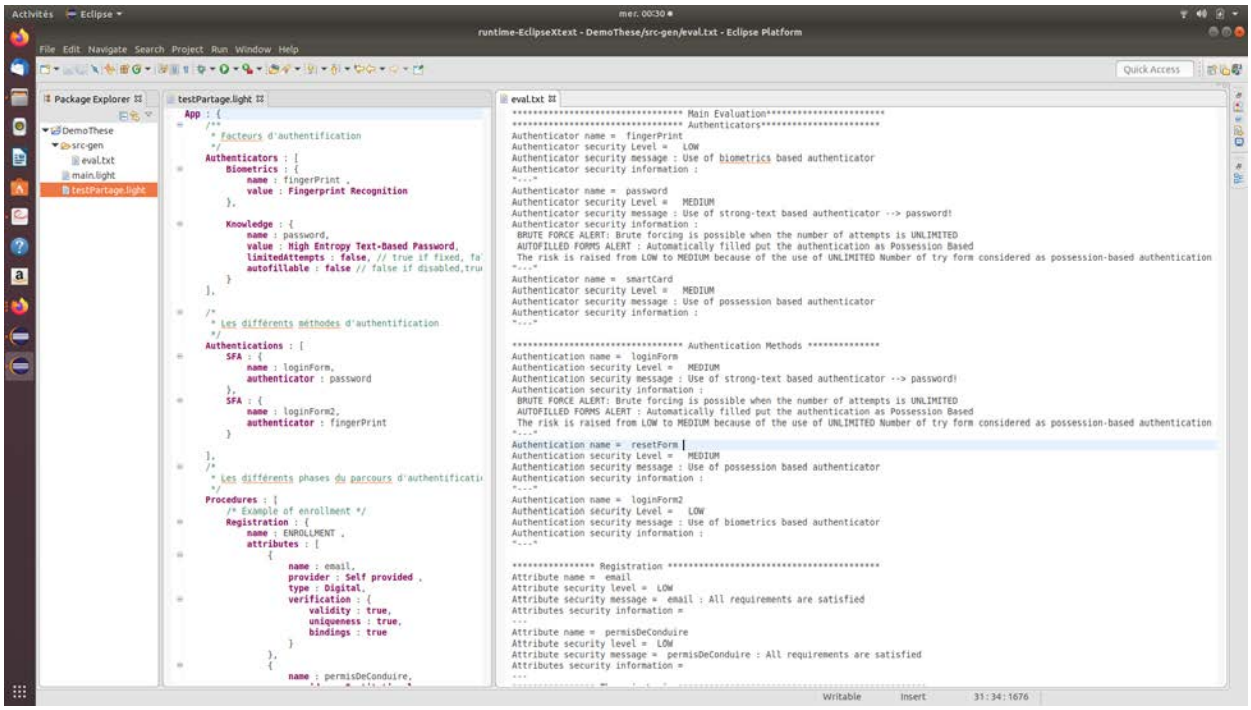

Figure 5.2 – Environnement de modélisation du parcours d'authentification

5.3.1 Syntaxe abstraite et exigences

Nous définissons quelques conventions afin de rendre le langage plus adapté à notre contexte.

Syntaxe abstraite

Ci-après les conventions qui permettent de comprendre la grammaire du langage, mais aussi la syntaxe utilisée pour décrire un parcours d'authentification.

— "?" : Les attributs définis avec ce symbole "?" sont optionnels, cela signifie qu'ils sont initialisés par défaut quand ils ne sont pas renseignés par le concepteur.

— "f1 & f2" : Cette convention permet de définir un "ET" logique des éléments f1 et f2. Par exemple, une authentification à deux facteurs est définie par cette convention.

1. https ://www.eclipse.org/modeling/emf/
2. https ://www.eclipse.org/Xtext/

— "[a1, a2, ...]" : Cette convention permet de définir une liste d'éléments tels que l'ensemble des attributs d'identité nécessaires à la phase d'enregistrement.
— "|" ou "||" : Le premier désigne une alternative sur la valeur retournée pour la grammaire. Alors que le second désigne un "OU" logique utilisé dans le langage.
— "*App* : {...}" désigne l'initialisation du parcours d'authentification.
— "*Object* : {...}" désigne l'initialisation d'un objet, avec *Object* égal à : *Login*, *Reset*, *Recovery*, ce qui correspond respectivement aux phases de *login*, de récupération et de mise à jour de preuves d'identité.

Les exigences fonctionnelles

Ci-après la liste des exigences fonctionnelles nécessaires pour la description d'un parcours. Certaines de ces exigences sont directement implémentées dans la sémantique, donc imposées aux concepteurs. Tandis que d'autres généreront des messages d'erreurs en cas de non respect (*i.e.*, le DSL ne compilera pas).

— Preuve d'identité et phase d'enregistrement (*Enrôlement)* : le parcours d'authentification doit comporter une seule phase d'enregistrement qui doit au moins déclarer un attribut d'identité.
— La phase de login (*Login*) : le parcours doit comporter au moins une phase de login qui peut utiliser différents moyens d'authentification.
— Phase de récupération et de mise à jour des preuves d'identité : le parcours peut comporter une ou plusieurs phases de récupération (*Recovery*) et de mise à jour des preuves(*Reset*) d'identité. En effet, ces phases peuvent différer de part l'attribut (*i.e.*, preuve d'identité ou attribut d'identité) à modifier ou à récupérer.
— Authentification à deux facteurs : l'authentification à plusieurs facteurs est limitée à deux facteurs afin de faciliter le processus d'évaluation du niveau de risque.
— La phase de mise à jour des preuves d'identité : par définition, la phase de mise à jour de preuves de sécurité est accessible uniquement en session ouverte. Le temps écoulé entre la dernière connexion (*i.e.*, *login*) et la demande de mise à jour n'est pas prise en compte.
— Le moyen d'authentification est obligatoire pour la phase de login, par contre, celui-ci est optionnel pour les phases de récupération et de mise à jour des preuves d'identité. Cette exigences permet de refléter la pratique actuelle des mécanismes de récupération et de mise à jour qui se fondent sur des protocoles tels que le protocole EBIA [76].

5.3.2 Syntaxe concrète

Cette section présente la syntaxe concrète des différents composants de la spécification d'un parcours d'authentification.

Les facteurs d'authentification

Le langage permet de définir les trois types de facteur d'authentification (*cf.*, Listing 5.2) : facteur de possession (*i.e.*, *Possession*), facteur de connaissance (*i.e.*, *Knowledge*) et biométrique ou comportemental (*i.e.*, *Biometrics*).

Ces facteurs sont principalement définis par un nom qui permet de les référencer et leur valeur concrète (*i.e.*, *value*).

Pour les facteurs de connaissance, il faudra spécifier s'ils peuvent être automatiquement remplis ou non(*i.e.*, *autofillable* : "true" si oui, sinon "false") par des extensions (*e.g.*, gestionnaire de mot de passe, navigateur). Il faut aussi définir le nombre de fois qu'il est nécessaire de tenter de deviner le secret (*i.e.*, *limitedAttempts* : "true" si le nombre est limité, "false" sinon).

```
1  Authenticator : // Authenticator description
2      "Knowledge" :
3          {
4              "name" : <string> ,
5              "value" : <KVALUE> ,
6              "autofillable": <boolean> ,
7              "limitedAttempts" : <boolean>
8          },
9
10     "Biometrics" :
11     {
12         "name" : <string> ,
13         "value" : <BVALUE> ,
14     }
15
16     "Possession" :
17     {
18         "name" : <string> ,
19         "value" : <PVALUE> ,
20     }
21     ;
22
23 // Literals
24 VALUE :
25     PVALUE | KVALUE | BVLAUE, // P : Possession, K : Knowledge, B : Biometrics.
26 ;
27 BVALUE :
28     FingerPrint | IRIS // Fingerprint or Iris recognition among others
29 ;
30 KVALUE :
31      PREFERENCES | PASSWORD | PIN // Preferences = by preference such as secret
           question among others
32 ;
33 PVALUE :
34     DEVICE | TOKEN // Device  : mobile ou laptop, token = ubikey for example.
```

```
;
```

Listing 5.2 – Grammaire de la description des facteurs d'authentification

Les méthodes d'authentification

Deux types de méthode d'authentification sont descriptibles dans ce langage : authentification à un facteur (*i.e.,* Authentification à un facteur - *Single Factor Authentication* (SFA)) et l'authentification à deux facteurs (*i.e.,* Authentification à deux facteurs - *2-Factor Authentication* (2FA)). Une SFA est définie par son nom (*name*) qui permet à la méthode d'être référencée par la phase qui l'utilise.

Par ailleurs, une authentification à deux facteurs est définie par les mêmes attributs en plus du facteur de corrélation (*i.e., correlation)* qui est un booléen pour spécifier si les facteurs sont corrélés ou non, et du type de validation (*i.e., validation*) qui est une énumération de deux valeurs : locale (*i.e., LOCALE*) pour une validation locale et distante (*i.e., REMOTE*) pour une validation à distance. Ci-après (*i.e.,* Listing 5.3) la grammaire pour décrire les facteurs d'authentification et les méthodes d'authentification.

```

Authentication : // Authentication methods
    2FA | SFA
    ;

    SFA :  //Single Factor Authentication
        {
            "name" : <string>,
            "authenticator" : <Authenticator>, // The authenticator
        }
        ;

    2FA : //Two Factors Authentification
        {
            "name" : <string> ,
            "authenticators" : <Authenticator> & <Authenticator> , // The two
                supported authenticators
            "correlation" : <boolean> , // Correlation between authenticator and
                devices
            "validation" : Local | Remote // Type of validation
        }
        ;
```

Listing 5.3 – Grammaire de la spécification des méthodes d'authentification

Phases du parcours

Comme nous l'avons décrit dans la Section 5.3.1, la spécification du parcours d'authentification doit comporter un minimum de quatre phases dont : une phase d'enregistrement, une phase de login, une phase de récupération et une phase de mise à jour de preuves d'identité.

Pour rappel, la phase d'enregistrement permet de recueillir les informations "pertinentes" pour l'application (*i.e.*, attributs d'identité et preuves d'identité). Nous précisons que dans le DSL, une *preuve d'identité* est uniquement constituée des facteurs d'authentification. Nous ignorons ainsi les identifiants pour des raisons de simplification.

La phase de *login*, en plus de référencer le ou les moyens d'authentification utilisés (*i.e.*, les moyens sont séparé par un "OU" logique), doit aussi spécifier la nature de la session créée en cas de succès (*i.e.*, $persistedSession$). L'attribut de la session est un booléen pour spécifier si la session est persistante ou non.

Le moyen d'authentification est optionnel pour les phases de récupération et de mise à jour de preuve d'identité. Cela pour permettre de spécifier les parcours autorisant le changement de preuve d'identité sans challenge de sécurité préalable. Néanmoins, elles devront référencer la preuve d'identité à mettre à jour ainsi que le protocole utilisé (uniquement pour la phase de récupération). Ci-après (*i.e.*, Listing 5.4) la grammaire de la description de ces différentes phases.

```

Phases :
    Registration | Login | Reset | Recovery
    ;

    Registration :
        profile : [<Attribute>, <Attribute>,...]
    ;

    Attribute :
        {
        "name" : <string> ,
        "provider" : SELF | IdP | INSTITUTIONAL ,
        "verification" :
            {
            "validity" : <boolean> ,
            "unicity" : <boolean> ,
            "bindings" : <boolean>
            }
        }
        ;
    IdP :
        Email | PhoneNumber // Only email and phnone number are supported
    ;
    INSTITUTIONAL :
        NationalID | DriverLicence | PassPort | ...

    Login :
```

```
    {
        "authentication" : <Authentication> (|| <Authentication> )*
        "persistedSession" : <boolean> // The type of session
    }
    ;

    Reset :
    {
        "authentication"? : <Authenticator> (|| <Authentication> )* // The security
            challenge
        "authenticator" : <Authenticator> // The factor to update
    }
    ;

    Recovery :
    {
        "authentication"? : <Authentication> (|| <Authenication>) * // the security
            challenge
        "authenticator" : <Authenticator> , // The factor to update
        "protocol" : Email-based | SMS-based | LOCAL
    }
;
```

Listing 5.4 – Grammaire d'un parcours global

5.4 Processus d'évaluation du niveau de risque

Le processus d'évaluation génère deux rapports : un rapport d'évaluation du niveau de risque et un rapport d'information sur ces risques. Le rapport d'évaluation donne le niveau de risque des évènements redoutés résultant de la description du parcours. Tandis que le rapport d'information sensibilise le concepteur sur les risques avérés, mais aussi sur les risques éventuels auxquels celui-ci pourrait ignorer.

Rappelons que les évènements redoutés sont : la souscription frauduleuse et l'accès non autorisé qui se décline en imitation ou en substitution de l'entité légitime. Trois niveaux sont à considérer : *ÉLEVÉ* si la probabilité d'occurrence du risque est très élevée ; *MODÉRÉ*, si l'attaque nécessite plus d'efforts à l'attaquant pour parvenir à ses fins ; *FAIBLE* si le risque est très minime voire inexistant en fonction des éléments connus.

L'évaluation du parcours commence par l'évaluation des facteurs d'authentification définis dans les Section 4.3.1 et 4.3.2. Pour rappel, les facteurs de possession sont évalués à un niveau de risque *MODÉRÉ* du fait qu'ils sont susceptibles d'être perdus ou volés. Les facteurs de connaissance et biométrique dépendent respectivement du secret et de la modalité biométrique et du dispositif d'acquisition. De plus, l'évaluation des facteurs de connaissance est influencée par le nombre d'essai autorisé pour trouver le secret et l'auto-remplissage des champs d'acquisition.

La seconde étape du processus consiste à évaluer les phases du parcours d'authentification en commençant par la phase d'enregistrement en vu d'une souscription frauduleuse.

5.4.1 Risque de souscription frauduleuse via la phase d'enregistrement

Rapport d'évaluation du niveau de risque

Le rapport d'évaluation du niveau de risques est obtenu via un programme dont le le pseudo-code est donné ci-après (Listing 5.5).

```
for attribute in profile :
    if(!attribute.validity AND !attribute.unicity AND !attribute.bindings) :
        attribute.LEVEL = HIGH
    else if (!attribute.validity OR !attribute.unicity OR !attribute.bindings) :
        attribute.LEVEL = MEDIUM
    else
        attribute.LEVEL = LOW
```

Listing 5.5 – Pseudo-code de l'évaluation du niveau de risques de souscription d'une entité frauduleuse.

Ce pseudo-code se fonde sur le processus d'évaluation décrit en Section 4.2.3. De plus, chaque attribut est évalué de façon unitaire en fonction du respect ou non des exigences de sécurité par la fonction de vérification.

Nous considérons la spécification de l'exemple illustratif (listing 5.1) pour illustrer le résultat du rapport d'évaluation relatif à une souscription frauduleuse. Le résultat est présenté dans la Table 5.1.

Attribut	Niveau de risque	Détails
e-mail	FAIBLE	Le niveau de risque global est évalué à FAIBLE car les trois attributs ont respecté les exigences recommandées.
phoneNumber	FAIBLE	
driverLicense	FAIBLE	

Table 5.1 – Rapport d'évaluation du niveau de risque d'une souscription frauduleuse de l'application de partage voiture

La fonction de vérification étant déclarative et peu explicite (*i.e.*, usage de booléen), nous générons un deuxième rapport de sensibilisation afin de renforcer la spécification du concepteur via des alertes et des recommandations.

Rapport d'information

En phase d'enregistrement, le rapport d'information permet de guider le concepteur sur les méthodes de vérification à mettre en place afin de satisfaire les exigences de sécurité. Ce rapport, qui se fonde sur les conditions de satisfiablité des exigences décrites en Section 4.2.1 et 4.2.2, est purement informationnel. Par conséquent, le concepteur est libre d'appliquer les méthodes de vérification qui lui semblent appropriées pour atteindre le niveau de sécurité escompté ou inversement, le niveau de risque à éviter.

La table 5.2 est un exemple de rapport d'information pour l'exemple illustratif. Un ensemble d'hypothèses sont émises sur chaque attribut en fonction de sa nature (*i.e.*, $provider$: *SELF* si c'est l'utilisateur qui le fourni, *IdP* si c'est un IdP et *INSTITUTIONAL* s'il s'agit d'un document d'identité physique) afin de guider le concepteur vers les méthodes de vérification pertinentes pour satisfaire les exigences.

Attributs	Fournisseur	Recommandations et Alertes
e-mail	Fournisseur de Service	— validité : vérification du format, envoi de mail de test. — unicité : vérification des doublons en base de données. — appartenance : protocole de *challenge-response* (*e.g.*, envoi d'un mail de confirmation).
phoneNumber	Fournisseur de service	— validité : vérification de format, envoi d'un SMS avec accusé de réception — unicité : vérification des doublons — appartenance : *challenge-response* ou appel direct.
driverLicense	INSTITUTIONNEL	— validité : validé par l'institution via une API ou une équivalence. — unicité : vérification des doublons, croisement des informations — appartenance : Si photo, alors vérification physique ou équivalence (via un canal *webRTC* par exemple). Sinon, demande d'un autre document institutionnel équivalent.

Table 5.2 – Rapport de sensibilisation du niveau de risque d'une souscription frauduleuse de l'application de partage voiture

5.4.2 Risque d'accès non autorisé via la phase de login et la phase de mise à jour de preuves de sécurité

Rapport d'évaluation

Le rapport d'évaluation d'un risque d'accès non autorisé via la phase de login se fonde, premièrement, sur l'évaluation des facteurs d'authentification, deuxièmement, sur la spécification de la phase de *login* selon la procédure décrite en Section 4.3.4. Pour rappel, si la session est persistante alors le risque est évalué tel une authentification fondée sur un facteur de possession. En l'absence de session persistante, l'évaluation est directement obtenue via l'évaluation du moyen d'authentification.

Le risque est évalué selon le parcours de récupération de preuves de sécurité en se basant sur la procédure décrite en Section 4.3.5. En principe, l'évaluation prend en compte le protocole utilisé en plus du moyen d'authentification quand ce dernier existe. Dans le cas contraire, seul le protocole utilisé est pris en compte. Notons que des annotations (*i.e.*, les alertes) sont ajoutées afin d'orienter le rapport d'information. Listing 5.6 présente l'algorithme d'évaluation du niveau de risque d'un accès non autorisé via le parcours de *login* et de récupération de preuves de sécurité.

```
'''fonction d'initialisation des facteurs d'authentification'''
def initAuthenticator(auth) :
    switch(auth.type) :
        case POSSESSION :
            auth.LEVEL = MEDIUM
        case BIOMETRICS :
            auth.LEVEL = LOW
        case KNOWLEDGE :
            if(auth.autifilled OR !auth.limitedAttempts) :
                auth.LEVEL = MEDIUM
            else :
                auth.LEVEL = auth.value.LEVEL
'''fonction d'evaluation de la phase de login.'''
def evaluateLogin (login) :
    if (login.persistedSesson)
        login.LEVEL = MEDIUM
    else :
        if(auth instanceof 2FA) :
            login.LEVEL = max (login.authentication[0].LEVEL,login.authentication[1].
                LEVEL )
        else
            login.LEVEL = login.authentication.auth.LEVEL

'''fonction d'evaluation de la phase de recuperation de preuves de securite.'''
def evaluateRecovery (recovery) :
    if (recovory.auth != null) :
        if(recovery.protocol == local) :
            recovery.LEVEL = recovery.auth.LEVEL
```

```
28          else :
29              recovery.LEVEL = min (recovery.auth.LEVEL, MEDIUM)
30      else
31           if(recovery.protocol == local) :
32              recovery.LEVEL = LOW
33          else
34              recovery.LEVEL = MEDIUM
```

Listing 5.6 – Pseudo-code de l'évaluation du niveau des risques d'un accès non autorisé via la phase de login et de récupération de preuves de sécurité.

Nous reprenons l'exemple illustratif spécifié dans le Listing 5.4 pour illustrer le rapport d'évaluation. Pour rappel, la plateforme de partage de véhicules propose deux moyens d'authentification pour ouvrir une session utilisateur : une authentification fondée sur la reconnaissance de l'empreinte digitale et une autre fondée sur la connaissance d'un mot de passe contraignant. Les deux ont un niveau de risque *FAIBLE*. Cependant, l'application autorise, en plus de la session persistante, le remplissage automatique du formulaire de mot de passe. Par conséquent, le niveau de risque est évalué à *MODÉRÉ* car considéré comme un facteur de possession.

Pour ce qui est de la phase de récupération, l'authentification est fondée sur la réponse à une question secrète qui est évalué à niveau de risque *ÉLEVÉ* car la question sécrète est considérée comme faible en terme de niveau de sécurité [22]. Cependant, le protocole utilisé est fondé sur l'adresse mail ou le SMS, de ce fait, le niveau de risque est évalué à *MODÉRÉ* car l'attaquant doit accéder au compte mail ou au SMS de la victime pour réussir l'exploitation.

La Table 5.3 présente le rapport d'évaluation du niveau de risque d'un accès non autorisé. Le niveau de risque global est *MODÉRÉ*, ce qui délègue la protection de l'application à la protection du dispositif (*e.g.*, téléphone portable, ordinateur) sur lequel celui-ci s'exécute.

Phase	Niveau de risque	Description
Phase de login	MODÉRÉ	Du fait de la session persistante ou des formulaires pré-remplis, la phase de login est évaluée comme une authentification fondée sur la possession.
Récupération de preuves de sécurité	MODÉRÉ	La question secrète est évaluée à *FAIBLE* en terme de niveau de sécurité. Par contre, le niveau de risque est réduit par le protocole qui nécessite d'accéder au compte mail ou au SMS de la victime.

Table 5.3 – Rapport d'évaluation du niveau de risque d'un accès non autorisé via la phase de login et de récupération de preuve d'identité.

Rapport d'information

Le rapport d'information permet d'alerter le concepteur sur les éventuels risques résiduels. Ce rapport est aidé par les annotations de l'évaluation du niveau de risque. Chaque annotation correspond à un message d'alerte qui permet d'orienter le rapport d'information vers les recommandations adéquates.

À titre d'exemple, La table 5.4 correspond au rapport de sensibilisation généré pour l'exemple illustratif.

5.5 Risque de substitution de l'entité légitime via la phase de mise à jour

La spécification de la phase de mise à jour des preuves d'identité permet d'évaluer le niveau de risque de substitution de l'entité légitime. Pour rappel, la substitution de l'entité légitime est possible en cas de session valide, elle consiste à réussir à changer les preuves d'identité de l'entité légitime par une entité malicieuse. Ainsi, celle-ci est dépossédée de son compte utilisateur.

Le rapport d'évaluation se focalise en priorité sur la spécification de la phase de mise à jour selon la procédure d'évaluation décrite en Section 4.4. L'évaluation prend aussi en compte le moyen d'authentification de la phase de login ainsi que le type de session créée en cas de succès. En effet, une session persistante ou son équivalent augmente le niveau de risque d'imitation de l'entité légitime, par conséquent, augmente aussi le risque de substitution de celle-ci. En l'absence de session persistante, l'attaquant devra ouvrir une session avant de pouvoir tenter une substitution, d'où le fait de considérer et la phase de login et la phase de récupération de preuves d'identité car les deux mènent à la création d'une nouvelle session utilisateur qui est indispensable pour une substitution de l'utilisateur légitime.

Le Listing 5.7 présente le pseudo-code correspondant à l'évaluation du niveau de risque. *App.login* correspond à l'évaluation du niveau de risque le plus élevé entre les phases de *login* et de récupération de preuves de sécurité s'il y en a eu plusieurs.

```
1  def evaluateReset (reset) :
2      if (App.login.persistedSession) :  '''persisted session in login phase'''
3          if(reset.auth) :
4              reset.LEVEL = min (reset.auth.LEVEL, MEDIUM) ''' Take the minimum risk
                   between Medium and the authenticatio risk LEVEL.'''
5          else :
6              reset.LEVEL = MEDIUM ''' considered as possession-based'''
7      else :
8          if (reset.auth) :
9              reset.LEVEL = min (reset.LEVEL, App.login.LEVEL) ''' minimum between login
                   level and reset level'''.
10         else :
```

Phase	Alerte	Recommandation
Login	Session persistante ou formulaire pré-rempli	L'usage de session persistante ou de formulaire pré-rempli est interdit par les exigences #Ex5' et #Ex5" (*cf.*, Section 3.3.2). En effet, dans un environnement ubiquitaire où le dispositif de l'entité légitime n'est pas correctement protégé, l'application de partage de voiture sera exposée à un niveau de risque élevé d'accès non autorisé.
Login	Usage de biométrie	Si l'acquisition de l'empreinte digitale est faite par le téléphone de l'entité légitime, il est fortement recommandé de vérifier le modèle de celui-ci avant d'autoriser l'authentification. En effet, la fiabilité du système biométrique de certains téléphones n'est pas prouvée ce qui peut augmenter le niveau de risque d'un accès non autorisé(*cf.* Section 4.4).
Récupération	Récupération par SMS ou mail	L'exigence #Ex7 (*cf.*, Section 3.3.2) recommande de limiter les chemins alternatifs qui peuvent affaiblir le niveau de sécurité. Le challenge de sécurité fondé sur la question secrète est assez faible, par conséquent, l'application est exposée à un niveau de risque élevé en cas de protection faible du dispositif de l'entité légitime.

Table 5.4 – Rapport de sensibilisation d'un accès non autorisé de l'application de partage de voiture.

```
11              reset.LEVEL = min (MEDIUM, App.login.LEVEL)
```

Listing 5.7 – Évaluation du risque de la phase de mise à jour de preuves de sécurité.

Le rapport d'évaluation de l'exemple illustratif est présenté dans la Table 5.5. Pour rappel, la mise à jour du mot de passe, nécessite de connaître l'ancien mot de passe. Mais, le formulaire est automatiquement rempli.

Phases	Niveau de risque	Description
Mise à jour	MODÉRÉ	Le niveau de risque est le même que la phase de login présentée dans la Table 5.3. En effet, malgré la nécessité d'un mot de passe, le formulaire de saisi de celui-ci est pré-rempli, par conséquent, il suffit d'accéder à l'application pour pouvoir changer le mot de passe de l'entité légitime.

Table 5.5 – Rapport d'évaluation du niveau de risque de substitution de l'entité légitime.

Enfin, le rapport d'information met l'accent sur le recours aux formulaires pré-remplis qui sont prohibés pour tout moyen d'authentification fondé sur la connaissance.

5.6 Conclusion du chapitre

Ce chapitre présente un environnement dédié à la spécification du parcours d'authentification et l'évaluation du niveau de risque associé à cette spécification. Cet environnement apporte deux avantages aux concepteurs des parcours d'authentification. D'une part, celui-ci permet de spécifier les mécanismes d'authentification qui permettent d'appliquer les exigences préconisées dans le Chapitre 3 pour atténuer les risques d'attaques logiques. D'autres part, il permet une évaluation automatique du niveau de risque d'exploitations logiques relatifs à une spécification en se fondant sur le cadre d'évaluation présenté dans le Chapitre 4. Ce environnement vient ainsi outiller les approches proposées pour répondre à la dernière question de recherche (*i.e.*, **RQ3**, Section 1.2).

Par ailleurs, il manque à cet environnement d'une validation empirique au niveau des concepteurs de parcours d'authentification. Cette validation pourrait permettre d'évaluer son expressivité et la traçabilité des exigences qu'il permet de spécifier sur un plus large panel d'applications que l'exemple illustratif discuté tout au long de ce chapitre. Elle permettrait également d'évaluer le gain qu'apporte une telle approche dans la découverte et la prise en compte des défauts logiques.

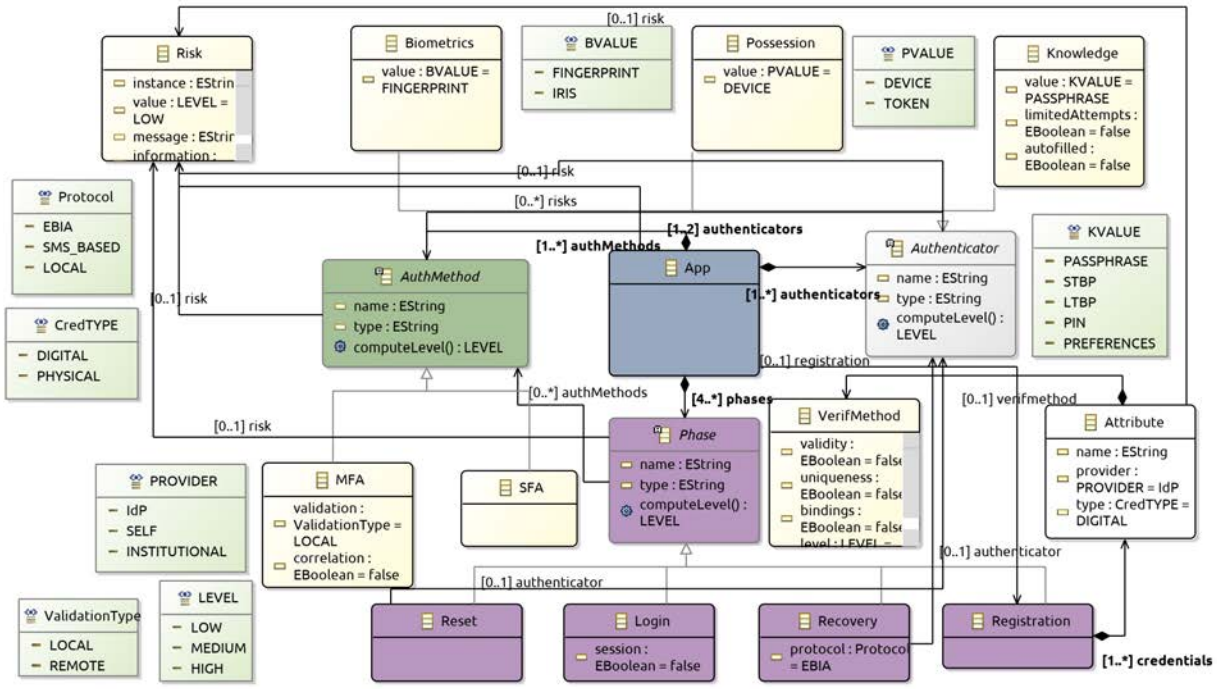

Figure 5.3 – Métamodèle complet du DSL

Chapitre 6

Conclusion et Perspectives

Cette section conclut les travaux de cette thèse. La première présente la conclusion globale de la thèse.
La deuxième section introduit une perspective d'étude en rapport avec l'ontologie présentée dans la Section 3.2.3. En effet, dans le cadre d'évaluation du niveau de risque (*cf.*, Chapitre 4) nous avons étudié les mécanismes d'authentification et d'identification nécessaires pour satisfaire les exigences. Le but de cette perspective, est d'étendre et d'implémenter cette ontologie en collaboration avec les experts de sécurité, les concepteurs et développeurs dans le but d'acquérir davantage de connaissance pour permettre de répondre à un ensemble de questions relatives aux risques et failles de sécurité sur parcours d'authentification.
La dernière Section présente une seconde perspective à l'attention de l'utilisateur final ou des concepteurs. Le but étant de montrer que la sécurité peut être améliorée en préconisant ou prohibant certaines tendances. À titre d'exemple nous proposons un mécanisme qui permet de mettre fin aux cookies de session pour augmenter la sécurité des parcours d'authentification *password-less*.

6.1 Conclusion

Dans cette thèse, nous avons exploré la problématique de la présence des défauts logiques sur les parcours d'authentification. Nous avons étendu la notion de défaut logique en prenant en compte des éléments extérieurs à la spécification d'un parcours d'authentification. Nous avons par la suite montré à l'aide d'une étude empirique que ces défauts logiques peuvent nuire aux parcours d'authentification en terme de sécurité et d'utilisabilité.

Par la suite, nous avons proposé une ontologie qui décrit les relations entre les facteurs à l'origine des défauts logiques, le parcours d'authentification et les éventuelles attaques qui exploitent ces défauts. De cette ontologie, nous en avons déduit un ensemble d'exigences que nous préconisons afin d'améliorer la sécurité des parcours d'authentification

Pour aller au delà des exigences dont l'objectif n'est pas de statuer sur les mécanismes à mettre en place pour les satisfaire [85], nous avons introduit un cadre d'évaluation des risques relatifs à la spécification d'un parcours d'authentification. Ce cadre permet d'évaluer les mé-

canismes de vérification et d'authentification mis en place pour appliquer les exigences. Ce cadre étant théorique, nous l'avons conceptualisé à l'aide d'un langage dédié. Ce langage qui reprend les abstractions des concepts du parcours d'authentification a pour objectif de réduire le fossé entre les exigences issus des experts et concepteurs, et les mécanismes pour appliquer ces exigences implémentées par les développeurs. Ainsi, les concepteurs et les experts, peuvent, sur une même plateforme spécifier le comportement attendu d'un parcours d'authentification avec les mécanismes à mettre en place et en considérant les risques issues des autres facteurs de contexte que les développeurs doivent éviter.

Enfin, nous sommes conscients du fait que les failles logiques sont difficiles à appréhender dans un contexte où les applications évoluent très rapidement. Cependant, malgré ces contraintes, cette thèse prétend que les concepteurs de ces services doivent davantage analyser un maximum de facteurs (*e.g.*, utilisabilité, dispositifs, comportement utilisateur, tendances) susceptibles d'introduire de nouveaux risques. Dans cette optique, en plus des travaux de cette thèse, nous proposons deux perspectives d'études pour continuer à améliorer l'écosystème des parcours d'authentification.

6.2 Une ontologie pour améliorer l'écosystème du développement des parcours d'authentification

Après avoir défini le contexte et le domaine d'une ontologie, l'un des points les plus importants est de déterminer l'objectif de cette ontologie, autrement dit, les questions auxquelles cette ontologie permet de répondre [19]. Le domaine et le contexte de notre ontologie sont discutés dans la Section 3.2.3, ainsi, dans cette section nous présentons comment l'ontologie pourrait permettre de répondre entre autres, aux questions suivantes :

- Quelles sont les attaques logiques relatives à un parcours d'authentification ?
- Quels composants du parcours sont affectés par une attaque donnée ?
- Quelle attaque conduit à un risque donné, quelle est la faille exploitée par cette attaque ?
- Quelles exigences permettent d'atténuer un risque donnée, quels mécanismes appliques ces exigences ?

Pour illustrer les réponses attendues à ces questions, nous proposons d'implémenter l'ontologie en utilisant Protégé [92]. Protégé supporte la plupart des formats pour développer des ontologies, notamment OWL [124] et RDF [111]. De plus, l'outil permet d'interroger l'ontologie via des langages de requête structurée dont DLQuery [1] et SPARQL [139]. Nous allons illustrer cette perspective d'étude avec un cas d'usage.

6.2.1 Cas d'usage

Pour les besoins de l'exemple, considérons un risque d'imitation de l'utilisateur légitime qui exploite la phase de *login* via une attaque par force brute. Une implémentation de l'exi-

1. https://protegewiki.stanford.edu/wiki/DLQueryTab

gence #Ex6 (*cf.*, Section 3.3.2) permet d'atténuer ce risque. Cette exigence peut être appliquée par différents mécanismes. Pour l'exemple nous proposons un mécanisme d'authentification qui utilise un mot de passe solide et qui limite le nombre d'essai à trois tentatives. En cas d'échec, une authentification alternative de même niveau de sécurité est proposée. Ci-après les questions qu'un concepteur pourrait poser à l'ontologie à titre d'exemple.

Q1 Quelles attaques logiques conduisent à un risque d'imitation de l'utilisateur légitime via la phase de de *login* ?

Q2 Quelles exigences permettent d'atténuer le risque d'imitation d'un utilisateur via la phase de login ?

Q3 Quels mécanismes permettent d'appliquer ces exigences ?

La figure 6.1 présente l'extension de l'ontologie avec le mécanisme d'authentification. Un mécanisme d'authentification est ainsi constitué d'attributs (*i.e.*, authentificateur et autre facteur). L'attribut *authentificateur* représente le facteur d'authentification à utiliser (*i.e.*, nous somme dans le cas d'une authentification à un seule facteur) et les autres facteurs représentent des éléments de contexte pertinent à prendre en considération par le mécanisme (*e.g.*, type de session à créer en cas de succès, nombre de tentatives autorisé).

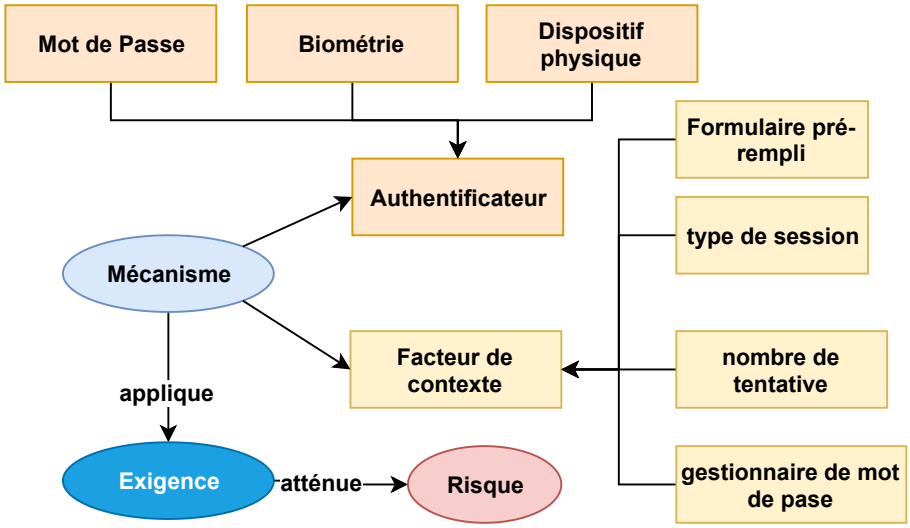

Figure 6.1 – Ontologie du mécanisme d'authentification

6.2.2 Implémentation et réponses aux questions

Nous utilisons Protégé pour implémenter l'ontologie. Horridge *et al.* [92] présente un guide complet pour créer une ontologie avec Protégé et OWL. En résumé, nous avons crée des *individuals* qui représentent les instances des classes de notre ontologie, puis nous avons construit les données d'attributs et les relations entre les concepts. Ensuite nous utilisons une extension de DLQuery pour interroger notre ontologie. La Figure 6.2 présente les requêtes DLQuery correspondantes aux trois questions et les résultats retournés par l'ontologie.

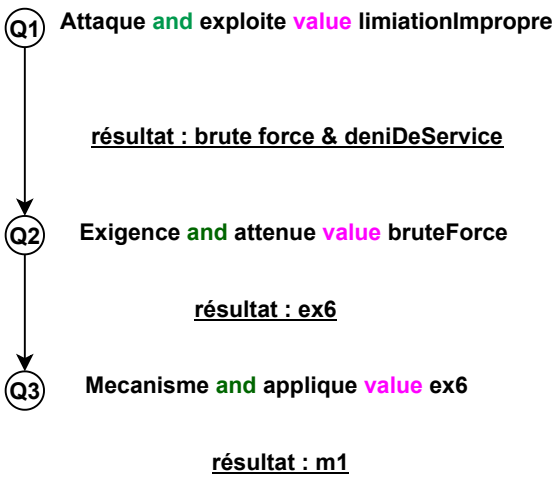

Figure 6.2 – illustration des requêtes DLQuery et leurs résultats

En résumé, sachant que l'attaque est due à une *limitation impropre* du nombre de tentative, l'exigence (*i.e.*, ex6) exige que tout mécanisme d'authentification doit "proprement" limiter le nombre de tentative. Ainsi, le mécanisme (*i.e.*, m1) applique cette exigence en fixant un nombre limité de tentative pour deviner un mot de passe solide. Le code source de l'ontologie et les résultats des requêtes sont disponibles sur GitHub [2].

Ce cas d'usage simple d'une attaque par force brute permet d'illustrer l'utilité d'une ontologie pour des concepteurs susceptibles d'ignorer les risques de sécurité relatifs aux spécifications qu'ils produisent. Nous sommes convaincus que disposer d'un tel outil peut aider à améliorer les parcours d'authentification. D'autant plus qu'une ontologie est facilement extensible ; d'autres problématiques de sécurité peuvent ainsi être intégrées.

6.3 Vers la fin des cookies de session

L'authentification à deux facteurs est en train de devenir la norme dans le domaine de l'authentification Web et Mobile avec l'avènement du protocole *U2F* (*i.e.*, authentification universelle à deux facteurs) de l'alliance FIDO soutenue par des acteurs connus du numérique et des compagnies internationales (*e.g.*, Amazon, Google, Thales, VISA). Le protocole *U2F* (*cf.*, Figure 6.3) permet aux services numériques d'étendre leurs mécanismes d'authentification fondés sur le mot de passe avec un second facteur physique d'authentification [161]. En principe, ce second facteur contient les clés privées de l'utilisateur pour les services auxquels celui-ci s'est préalablement enrôlé. Ce protocole est qualifié de *password-less* (sans mot de passe) car sur un dispositif de confiance déjà enregistré, l'utilisateur a juste à présenter une signature numérique contenu dans une clé usb protégé (*i.e.*, un dispositif iinviolable) à chaque

2. `https://github.com/youssou/ontologie`

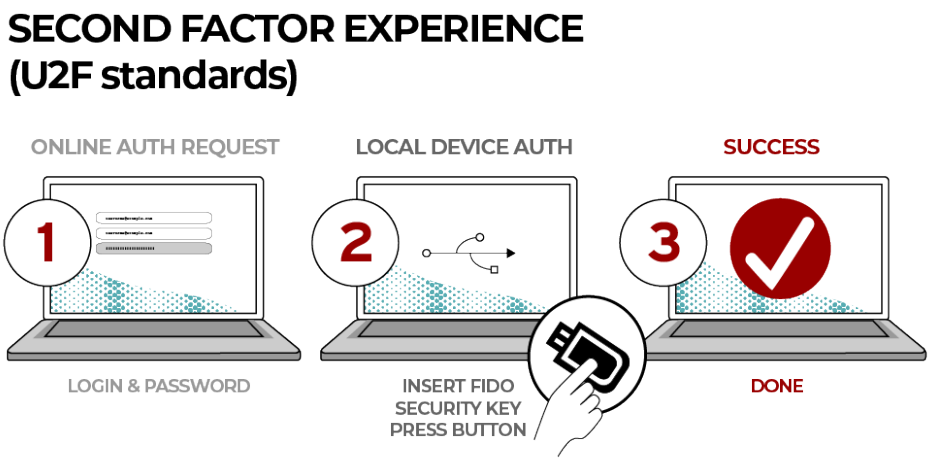

Figure 6.3 – Illustration de l'usage du protocole FIDO U2F par l'alliance fido

demande d'authentification. Notons que le dispositif physique peut être renforcé avec un code PIN ou un lecteur d'empreinte digital grâce au protocole FIDO *UAF* (*i.e.*, cadre universel d'authentification) tel qu'utilisé par Google dans ses smartphones Android (*cf.*, Figure 6.4).

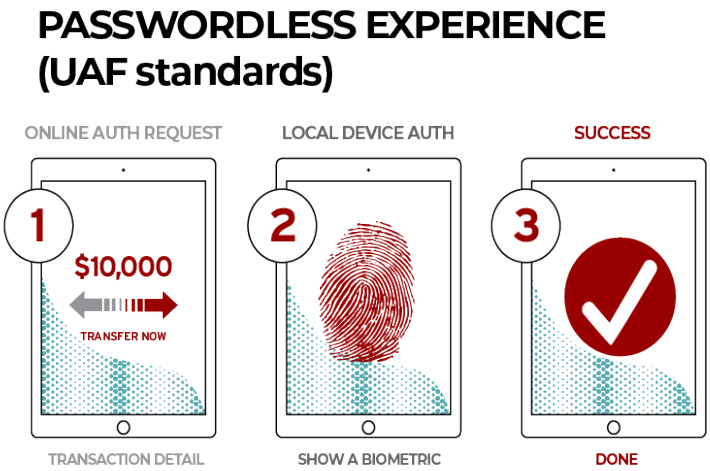

Figure 6.4 – Illustration de l'usage du protocole FIDO UAF par l'alliance fido

Ce protocole d'authentification à deux facteurs, sans corrélation entre les facteurs et le dispositif d'acquisition permet ainsi de contrer des attaques éprouvées telles que l'attaque par homme du milieu ou l'hameçonnage comme nous l'avions discuté dans la Section 2.2.7.

Cependant, après une authentification réussie, le service tiers est libre de gérer un cookie de session à sa guise. Ainsi, en cas de perte d'un dispositif avec des sessions utilisateur actives, si ce dispositif n'est pas protégé ou qu'il dispose d'une sécurité faible, alors l'ensemble des services actifs sont susceptibles d'être compromis. Ainsi, pour augmenter la sécurité en ce sens, une perspective d'étude serait de mettre fin aux cookies de session persistants de deux manières différentes.

1. La première solution est à l'attention de l'utilisateur final. Celle-ci consiste à proposer une extension pour les navigateurs qui supprime l'ensemble des cookies de session d'un utilisateur à la fin de son navigation. La navigation privée, disponible dans la plupart des navigateurs, permet de supprimer l'historique de navigation, y compris les cookies et les mot de passes. Cependant, cette fonctionnalité ne permettrait pas de faire du *password-less* car à chaque nouvelle navigation le mot de passe est redemandé. Alors qu'en utilisant une extension dédiée, seule les cookies de session sont supprimés à la fin de la navigation.
2. La seconde solution est à l'attention des concepteurs des services. Celle-ci consiste tout simplement à ne pas garder de cookies de session persistants. Cette pratique est très courante chez les services bancaires qui ne pouvant tolérer un accès non-autorisé.

Mettre fin aux cookies de session permet également de réduire les risques d'attaques de vol et de fixation de session. Par ailleurs, même si le mécanisme *password-less* est pratique et facile à utiliser, il serait pertinent d'étudier l'impact de la suppression des cookies de session sur l'utilisabilité des mécanismes d'authentification pour l'utilisateur final.

Abbréviations

2FA Authentification à deux facteurs - *2-Factor Authentication* 99

DSL Langage Spécifique au Domaine - *Domaine Specific Language* 1

EBIA Authentification et Identification fondée sur l'adresse e-mail - *Email Based Identification and Authentification* 55

EER Taux d'Égal Erreur - Equal Erreur Rate 26

FAR Taux de Fausses Acceptations - *False Acceptance Rate* 25

FRR Taux de Faux Rejets - *False Rejection Rate* 25

HIP Preuve d'interaction avec un Humain - *Human Interaction Proof* 60

IdP Fournisseur d'Identités - *Identity Provider* 27

OTP Code à Usage Nnique - *One Time Password* 48

SFA Authentification à un facteur - *Single Factor Authentication* 99

SP Fournisseur de Service - *Service Provider*

SSO Authentification Unique - *Single Sign ON* 27

SQLI Injection de requêtes SQL - *SQL Injection* 59

XSS *Cross-Site Scripting* 59

Glossaire

Application tierce Une application tierce (ou application partie prenante) fait confiance à une assertion provenant d'un fournisseur d'identité auquel celle-ci fait confiance. L'intégrité et la confidentialité doivent être assurées par le fournisseur d'identité, sachant que leur vérification incombe à l'application tierce.

Authentification L'authentification est le processus permettant de déterminer la validité d'un ou plusieurs facteurs d'authentification utilisés pour revendiquer une identité digitale ou physique. Elle permet de s'assurer qu'une entité souscrite demandant un accès à un service dispose de moyens permettant de l'authentifier.

Entité Une entité est une chose (*i.e.*,objet, humain) ayant une existence séparée et distincte, qui peut être identifiée dans un contexte. C'est aussi une chose qui déclare une identité physique ou numérique.

Entité souscriptrice Une entité souscriptrice correspond à une entité qui demande à s'enrôler sur un service. Lorsque l'enrôlement est réussi, on parlera d'entité souscrite.

Entité souscrite Voir entité souscriptrice.

Facteur d'authentification ou token une facteur d'authentification ou token est :

- quelque chose que l'on est : une donnée biométrique ;
- quelque chose que l'on connaît : un mot de passe ;
- quelque chose que l'on a : une carte à puce.

Le facteur d'authentification contient toujours un secret qui peut-être fondée sur un protocole cryptographique asymétrique ou symétrique (Le mot de passe étant considéré comme une clé cryptographique symétrique ou secret partagé).

Fournisseur de preuve d'identité Acteur de confiance délivrant ou gérant des preuves d'identité

Identité Ensemble d'attributs se rapportant à une entité. Une identité peut avoir un ou plusieurs identificateurs pour permettre à une entité d'être reconnue de façon unique dans un contexte.

Identificateur ou identifiant Un ou plusieurs attributs identifiant d'une manière unique une entité dans un contexte donné.

Identité numérique Une représentation unique d'une entité engagée dans une transaction en ligne. Cette unicité est relative à un contexte dans un service digital.

Imitation L'imitation désigne l'action d'imiter, via un accès frauduleux, un utilisateur légitime par une autre personne mal-intentionnée sur un service ou application donnée

Preuve d'identité Une preuve d'identité (*i.e., Credential*) est une pièce d'information qui permet de lier un ou plusieurs facteurs d'authentification à une entité souscrite via un identifiant.

Substitution La substitution d'un utilisateur désigne l'action d'usurper l'identité d'un utilisateur légitime tout en modifiant ses attributs et ces preuves d'identité. En cas de substitution, l'utilisateur légitime peut perdre définitivement le contrôle de son identité sur un service donné.

Protocole d'authentification Séquence définie de messages entre une entité et un contrôleur, qui permet à celui-ci de vérifier l'identité de l'entité.

Vérifieur Entité disposant d'assez d'aptitudes pour vérifier l'identité revendiquée par une autre entité.

Publications

SAC'19 Youssou Ndiaye, Olivier Barais, Arnaud Blouin, Ahmed Bouabdallah, Nicolas Aillery. **Requirements for preventing logic flaws in the authentication procedure of web applications**. SAC 2019- 34th ACM/SIGAPP Symposium On Applied Computing, Apr 2019, Limassol, Cyprus. pp.1-9,10.1145/3297280.3297438. hal-02087663

CIEL'17 Youssou Ndiaye, Nicolas Aillery, Olivier Barais, Arnaud Blouin, Ahmed Bouabdallah. **Modélisation et Évaluation de la Sécurité des IHM**. CIEL 2017 : 6ème Conférence en Ingénierie du Logiciel, Jun2017, Montpellier, France. hal-01611324

Bibliographie

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12) :40–46, 1999.

[2] Carlisle Adams, Guy-Vincent Jourdan, Jean-Pierre Levac, and François Prevost. Lightweight protection against brute force login attacks on web applications. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 181–188. IEEE, 2010.

[3] Manzoor Ahmad. First step towards a domain specific language for self-adaptive systems. In *2010 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE)*, pages 285–290. IEEE, 2010.

[4] Manzoor Ahmad, Nicolas Belloir, and Jean-Michel Bruel. Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems. *Journal of Systems and Software*, 107 :50–70, 2015.

[5] Manzoor Ahmad, Jean-Michel Bruel, Régine Laleau, and Christophe Gnaho. Using relax, sysml and kaos for ambient systems requirements modeling. *Procedia Computer Science*, 10 :474–481, 2012.

[6] Christopher Alberts, Audrey Dorofee, James Stevens, and Carol Woody. Introduction to the octave approach. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2003.

[7] Christopher J Alberts and Audrey Dorofee. *Managing information security risks : the OCTAVE approach*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[8] Anupriya Ankolekar, Mark Burstein, Jerry R Hobbs, Ora Lassila, David Martin, Drew McDermott, Sheila A McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, et al. Daml-s : Web service description for the semantic web. In *International Semantic Web Conference*, pages 348–363. Springer, 2002.

[9] ANSSI. Ebios — expression des besoins et identification des objectifs de sécurité.

[10] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. *Woot*, 10 :1–7, 2010.

[11] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[12] Davide Balzarotti, Marco Cova, Viktoria V Felmetsger, and Giovanni Vigna. Multi-module vulnerability analysis of web-based applications. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 25–35. ACM, 2007.

[13] Brandon Barbello. Continuous user authentication on mobile devices : Recent progress and remaining challenges. 2016.

[14] Luciano Baresi, Sebastiano Colazzo, Luca Mainetti, and Sandro Morasca. W2000 : A modelling notation for complex web applications. In *Web Engineering*, pages 335–364. Springer, 2006.

[15] Elaine B Barker, William C Barker, and Annabelle Lee. Sp 800-21 second edition. guideline for implementing cryptography in the federal government. 2005.

[16] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces : Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7) :711–720, 1997.

[17] Debnath Bhattacharyya, Rahul Ranjan, Farkhod Alisherov, Minkyu Choi, et al. Biometric authentication : A review. *International Journal of u-and e-Service, Science and Technology*, 2(3) :13–28, 2009.

[18] Matt Bishop. Vulnerabilities analysis. In *Proceedings of the Recent Advances in intrusion Detection*, pages 125–136, 1999.

[19] Carlos Blanco, Joaquin Lasheras, Rafael Valencia-García, Eduardo Fernández-Medina, Ambrosio Toval, and Mario Piattini. A systematic review and comparison of security ontologies. In *2008 Third International Conference on Availability, Reliability and Security*, pages 813–820. Ieee, 2008.

[20] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords : A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[21] Joseph Bonneau, Mike Just, and Greg Matthews. What's in a name ? In *International Conference on Financial Cryptography and Data Security*, pages 98–113. Springer, 2010.

[22] Joseph Bonneau and Sören Preibusch. The password thicket : Technical and market failures in human authentication on the web. In *WEIS*, 2010.

[23] Joseph Bonneau, Sören Preibusch, and Ross Anderson. A birthday present every eleven wallets ? the security of customer-chosen banking pins. In *International Conference on Financial Cryptography and Data Security*, pages 25–40. Springer, 2012.

[24] Mustafa Bozkurt, Mark Harman, Youssef Hassoun, et al. Testing web services : A survey. *Department of Computer Science, King's College London, Tech. Rep. TR-10-01*, 2010.

[25] Marco Brambilla and Piero Fraternali. *Interaction flow modeling language : Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann, 2014.

[26] WE Burr, Donna F Dodson, EM Newton, RA Perlner, WT Polk, S Gupta, and EA Nabbus. Nist special publication 800-63-2 : Electronic authentication guideline. *National Institute of Standards and Technology, Tech. Rep*, 2013.

[27] William E Burr, Donna F Dodson, William T Polk, et al. *Electronic authentication guideline*. Citeseer, 2004.

[28] Jordi Cabot, Robert Claris, Daniel Riera, et al. Verification of uml/ocl class diagrams using constraint programming. In *2008 IEEE International Conference on Software Testing Verification and Validation Workshop*, pages 73–80. IEEE, 2008.

[29] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security & Privacy*, 7(1) :78–81, 2009.

[30] Gerardo Canfora and Massimiliano Di Penta. Service-oriented architectures testing : A survey. In *Software Engineering*, pages 78–105. Springer, 2007.

[31] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (webml) : a modeling language for designing web sites. *Computer Networks*, 33(1-6) :137–157, 2000.

[32] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720. ACM, 2005.

[33] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2 : An open-source tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pages 359–364. Springer, 2002.

[34] Edmund M Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. Model checking and the state explosion problem. In *LASER Summer School on Software Engineering*, pages 1–30. Springer, 2011.

[35] Marco Cova, Davide Balzarotti, Viktoria Felmetsger, and Giovanni Vigna. Swaddler : An approach for the anomaly-based detection of state violations in web applications. In *International Workshop on Recent Advances in Intrusion Detection*, pages 63–86. Springer, 2007.

[36] Common Weakness Enumeration (CWE). A community-developed list of software weakness types.

[37] Italo Dacosta, Saurabh Chakradeo, Mustaque Ahamad, and Patrick Traynor. One-time cookies : Preventing session hijacking attacks with stateless authentication tokens. *ACM Transactions on Internet Technology (TOIT)*, 12(1) :1, 2012.

[38] Ni Dan, Shi Hua-Ji, Chen Yuan, and Guo Jia-Hu. Attribute based access control (abac)-based cross-domain access control in service-oriented architecture (soa). In *2012 International Conference on Computer Science and Service System*, pages 1405–1408. IEEE, 2012.

[39] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2) :3–50, 1993.

[40] OMF De Troyer and Corneli Jan Leune. Wsdm : a user centered design method for web sites. *Computer Networks and ISDN systems*, 30(1-7) :85–94, 1998.

[41] Grit Denker, Lalana Kagal, and Tim Finin. Security in the semantic web using owl. *Information Security Technical Report*, 10(1) :51–58, 2005.

[42] Grit Denker, Lalana Kagal, Tim Finin, Massimo Paolucci, and Katia Sycara. Security for daml web services : Annotation and matchmaking. In *International Semantic Web Conference*, pages 335–350. Springer, 2003.

[43] Rachna Dhamija and Lisa Dusseault. The seven flaws of identity management : Usability and security challenges. *IEEE Security & Privacy*, 6(2), 2008.

[44] Giuseppe A Di Lucca and Massimiliano Di Penta. Considering browser interaction in web application testing. In *Web Site Evolution, 2003. Theme : Architecture. Proceedings. Fifth IEEE International Workshop on*, pages 74–81. IEEE, 2003.

[45] Gregorio Diaz, Juan-José Pardo, María-Emilia Cambronero, Valentin Valero, and Fernando Cuartero. Verification of web services with timed automata. *Electronic Notes in Theoretical Computer Science*, 157(2) :19–34, 2006.

[46] Tim Dierks and Christopher Allen. The tls protocol version 1.0. 1999.

[47] Georg Disterer. Iso/iec 27000, 27001 and 27002 for information security management. 2013.

[48] P.S. Dowland D.Katsabas, S.M. Furnell. Hci principles to promote usable security.

[49] Michel dos Santos Soares and Jos LM Vrancken. Model-driven user requirements specification using sysml. *JSW*, 3(6) :57–68, 2008.

[50] Ankita Dubey, Zia Saquib, and Surabhi Dwivedi. Electronic authentication for e-government services-a survey. 2015.

[51] Amador Durán Toro, Beatriz Bernárdez Jiménez, Antonio Ruiz Cortés, and Miguel Toro Bonilla. A requirements elicitation approach based in templates and patterns. 1999.

[52] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3) :94–100, 2016.

[53] Serge Egelman, Sakshi Jain, Rebecca S Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. Are you ready to lock ? In *Proc. of CCS'14*, pages 750–761. ACM, 2014.

[54] Mohamad El-Abed, Baptiste Hemery, Christophe Charrier, and Christophe Rosenberger. Evaluation de la qualité de données biométriques. 2011.

[55] Common Weakness Enumeration. Weak password recovery mechanism for forgotten password.

[56] Michael D Ernst, Jeff H Perkins, Philip J Guo, Stephen McCamant, Carlos Pacheco, Matthew S Tschantz, and Chen Xiao. The daikon system for dynamic detection of likely invariants. *Science of computer programming*, 69(1-3) :35–45, 2007.

[57] M. Jose Escalona and Nora Koch. Requirements engineering for web applications-a comparative study. *J. Web Eng.*, 2(3) :193–212, 2004.

[58] Zhejun Fang, Qixu Liu, Yuqing Zhang, Kai Wang, Zhiqiang Wang, and Qianru Wu. A static technique for detecting input validation vulnerabilities in android apps. *Science China Information Sciences*, 60(5) :052111, Sep 2016.

[59] Viktoria Felmetsger, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna. Toward automated detection of logic vulnerabilities in web applications. In *USENIX Security Symposium*, volume 58, 2010.

[60] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. Methontology : from ontological art towards ontological engineering. 1997.

[61] David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (rbac) : Features and motivations. In *Proceedings of 11th annual computer security application conference*, pages 241–48, 1995.

[62] Duane K Fields, Mark A Kolb, and Shawn Bayern. *Web development with Java server pages*. Manning Publications Co., 2001.

[63] Donald Firesmith. Specifying reusable security requirements. *Journal of Object Technology*, 3(1) :61–75, 2004.

[64] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[65] OWASP Foundation. Access control cheat sheet.

[66] OWASP Foundation. Authentication cheat sheet.

[67] OWASP Foundation. Business logic security cheat sheet.

[68] OWASP Foundation. Information leakage and improper error handling.

[69] OWASP Foundation. Owasp top ten project.

[70] OWASP Foundation. Sensitive data exposure.

[71] OWASP Foundation. Session fixation attack.

[72] OWASP Foundation. Session hijacking attack.

[73] OWASP Foundation. Vulnerability.

[74] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics : On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security*, 8(1) :136–148, 2012.

[75] Alan Freier, Philip Karlton, and Paul Kocher. The secure sockets layer (ssl) protocol version 3.0. 2011.

[76] Simson L Garfinkel. Email-based identification and authentication : An alternative to pki ? *IEEE security & privacy*, 99(6) :20–26, 2003.

[77] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26. IEEE, 2007.

[78] Joseph A Goguen and Charlotte Linde. Techniques for requirements elicitation. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 152–164. IEEE, 1993.

[79] PA Grassi, ME Garcia, and JL Fenton. Nist special publication 800–63-3 : Digital identity guidelines. In *NIST*, 2017.

[80] Jeremiah Grossman. Seven business logic flaws that put your website at risk. *WhiteHat Security, October*, 2007.

[81] Object Management Group. Unified modeling language (uml v 2.5.1), 2017.

[82] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing ? *International journal of human-computer studies*, 43(5-6) :907–928, 1995.

[83] GSMA. Introducing mobile connect - the new standard in digital authentication, 2012-11-14.

[84] Shashank Gupta and Brij Bhooshan Gupta. Cross-site scripting (xss) attacks and defense mechanisms : classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8(1) :512–530, 2017.

[85] MA Hadavi, VS Hamishagi, and HM Sangchi. Security requirements engineering ; state of the art and research challenges. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 19–21, 2008.

[86] William G Halfond, Jeremy Viegas, Alessandro Orso, et al. A classification of sql-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, volume 1, pages 13–15. IEEE, 2006.

[87] Neil Haller. The s/key one-time password system. 1995.

[88] Dick Hardt. The oauth 2.0 authorization framework. Technical report, 2012.

[89] Julien Hatin, Estelle Cherrier, Jean-Jacques Schwartzmann, Vincent Frey, and Christophe Rosenberger. A continuous loa compliant trust evaluation method. In *International Conference on Information Systems Security and Privacy (ICISSP)*, 2016.

[90] Rolf Hennicker and Nora Koch. Modeling the user interface of web applications with uml. *pUML*, 7 :158–172, 2001.

[91] Paco Hope and Ben Walther. *Web security testing cookbook : Systematic techniques to find problems fast*. " O'Reilly Media, Inc.", 2008.

[92] Matthew Horridge, Simon Jupp, Georgina Moulton, Alan Rector, Robert Stevens, and Chris Wroe. A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 2. *The university of Manchester*, 107, 2009.

[93] RJ Hulsebosch, Mortaza S Bargh, Gabriele Lenzini, PWG Ebben, and Sorin M Iacob. Context sensitive adaptive authentication. In *European Conference on Smart Sensing and Context*, pages 93–109. Springer, 2007.

[94] Vinay M Igure and Ronald D Williams. Taxonomies of attacks and vulnerabilities in computer systems. *IEEE Communications Surveys & Tutorials*, 10(1), 2008.

[95] International Communication Union (ITU). Entity authentication assurance framework, 2012-09-07.

[96] Blake Ives, Kenneth R Walsh, and Helmut Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4) :75–78, 2004.

[97] Anil K Jain, Arun Ross, and Sharath Pankanti. Biometrics : a tool for information security. *IEEE transactions on information forensics and security*, 1(2) :125–143, 2006.

[98] Markus Jakobsson, Liu Yang, and Susanne Wetzel. Quantifying the security of preference-based authentication. In *Proceedings of the 4th ACM workshop on Digital identity management*, pages 61–70. ACM, 2008.

[99] Andrew Teoh Beng Jin, David Ngo Chek Ling, and Alwyn Goh. Biohashing : two factor authentication featuring fingerprint data and tokenised random number. *Pattern recognition*, 37(11) :2245–2255, 2004.

[100] Kanta Jiwnani and Marvin Zelkowitz. Susceptibility matrix : A new aid to software auditing. *IEEE security & privacy*, 2(2) :16–21, 2004.

[101] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques : Comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4) :320–351, December 1996.

[102] Dean Jones, Trevor Bench-Capon, and Pepijn Visser. Methodologies for ontology development. 1998.

[103] Jan Jürjens. Umlsec : Extending uml for secure systems development. In *International Conference on The Unified Modeling Language*, pages 412–425. Springer, 2002.

[104] Maria Karyda, Theodoros Balopoulos, S Dritsas, L Gymnopoulos, S Kokolakis, C Lambrinoudakis, and S Gritzalis. An ontology for secure e-government applications. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 5–pp. IEEE, 2006.

[105] Raman Kazhamiakin, Marco Pistore, and Marco Roveri. Formal verification of requirements using spin : A case study on web services. In *Proceedings of the Second International Conference on Software Engineering and Formal Methods, 2004. SEFM 2004.*, pages 406–415. IEEE, 2004.

[106] Anya Kim, Jim Luo, and Myong Kang. Security ontology for annotating resources. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1483–1499. Springer, 2005.

[107] Nora Koch, Alexander Knapp, Gefei Zhang, and Hubert Baumeister. Uml-based web engineering. In *Web Engineering : Modelling and Implementing Web Applications*, pages 157–191. Springer, 2008.

[108] Ivan Koldaev. Hack any skype account in 6 easy steps, 2012-11-14.

[109] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys (CSUR)*, 26(3) :211–254, 1994.

[110] Kim G Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1) :134–152, 1997.

[111] Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification. 1998.

[112] Kelly D Lewis. Web single sign-on authentication using saml. *arXiv preprint arXiv :0909.2368*, 2009.

[113] Xiaowei Li and Yuan Xue. Block : a black-box approach for detection of state violation attacks towards web applications. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 247–256. ACM, 2011.

[114] Yuan-Fang Li, Paramjit K Das, and David L Dowe. Two decades of web application testing—a survey of recent advances. *Information Systems*, 43 :20–54, 2014.

[115] Johan Lilius and Iván Porres Paltor. Formalising uml state machines for model checking. In *International Conference on the Unified Modeling Language*, pages 430–444. Springer, 1999.

[116] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Murielle Florins. Usixml : A user interface description language supporting multiple levels of independence. In *ICWE Workshops*, pages 325–338, 2004.

[117] Chien-Hung Liu, David Chenho Kung, Pei Hsia, and Chih-Tung Hsu. Structural testing of web applications. In *Proceedings 11th International Symposium on Software Reliability Engineering. ISSRE 2000*, pages 84–96. IEEE, 2000.

[118] Hal Lockhart and B Campbell. Security assertion markup language (saml) v2. 0 technical overview. *OASIS Committee Draft*, 2 :94–106, 2008.

[119] Torsten Lodderstedt, David Basin, and Jürgen Doser. Secureuml : A uml-based modeling language for model-driven security. In *International Conference on the Unified Modeling Language*, pages 426–441. Springer, 2002.

[120] Alessandra Lumini and Loris Nanni. An improved biohashing for human authentication. *Pattern recognition*, 40(3) :1057–1065, 2007.

[121] Wanli Ma, John Campbell, Dat Tran, and Dale Kleeman. Password entropy and password quality. In *2010 Fourth International Conference on Network and System Security*, pages 583–587. IEEE, 2010.

[122] Emanuela Marasco and Arun Ross. A survey on antispoofing schemes for fingerprint recognition systems. *ACM Computing Surveys (CSUR)*, 47(2) :28, 2015.

[123] Oleksiy Mazhelis and Seppo Puuronen. A framework for behavior-based detection of user substitution in a mobile context. *computers & security*, 26(2) :154–176, 2007.

[124] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10) :2004, 2004.

[125] William S. McPhee. Operating system integrity in os/vs2. *IBM Systems Journal*, 13(3) :230–252, 1974.

[126] Huaikou Miao and Hongwei Zeng. Model checking-based verification of web application. In *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, pages 47–55. IEEE, 2007.

[127] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2) :39–53, 2004.

[128] Deepti Mishra, Alok Mishra, and Ali Yazici. Successful requirement elicitation by combining requirement engineering techniques. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 258–263. IEEE, 2008.

[129] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv :1003.4083*, 2010.

[130] David Navarre, Philippe Palanque, Jean-Francois Ladry, and Eric Barboni. Icos : A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4) :18, 2009.

[131] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering : a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000.

[132] Business Process Model OMG. Notation (bpmn). *FTF Beta*, 1, 2011.

[133] Gunnar Övergaard and Karin Palmkvist. A formal approach to use cases and their relationships. In *International Conference on the Unified Modeling Language*, pages 406–418. Springer, 1998.

[134] Andreas Pashalidis and Chris J Mitchell. A taxonomy of single sign-on systems. In *Proc. of ACISP'03*, pages 249–264. Springer, 2003.

[135] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi. Concurtasktrees : A diagrammatic notation for specifying task models. In *Human-computer interaction INTERACT'97*, pages 362–369. Springer, 1997.

[136] Joshua J Pauli and Dianxiang Xu. Misuse case-based design and analysis of secure software architecture. In *International Conference on Information Technology : Coding and Computing (ITCC'05)-Volume II*, volume 2, pages 398–403. IEEE, 2005.

[137] Christopher J. Pavlovski and Joe Zou. Non-functional requirements in business process modeling. In *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling - Volume 79*, APCCM '08, pages 103–112, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[138] Giancarlo Pellegrino and Davide Balzarotti. Toward black-box detection of logic flaws in web applications. In *NDSS*, 2014.

[139] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3) :16, 2009.

[140] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977.

[141] FIPS Pub. Standards for security categorization of federal information and information systems. *NIST FIPS–199*, 2004.

[142] Ariel Rabkin. Personal knowledge questions for fallback authentication : Security questions in the era of facebook. In *Proceedings of the 4th symposium on Usable privacy and security*, pages 13–23. ACM, 2008.

[143] Abdul Razzaq, Zahid Anwar, H Farooq Ahmad, Khalid Latif, and Faisal Munir. Ontology for attack detection : An intelligent approach to web application security. *computers & security*, 45 :124–146, 2014.

[144] David Recordon and Drummond Reed. Openid 2.0 : a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16. ACM, 2006.

[145] Eric Rescorla. *SSL and TLS : designing and building secure systems*, volume 1. Addison-Wesley Reading, 2001.

[146] Filippo Ricca and Paolo Tonella. Analysis and testing of web applications. In *Proceedings of the 23rd international conference on Software engineering*, pages 25–34. IEEE Computer Society, 2001.

[147] Mark Richters and Martin Gogolla. Ocl : Syntax, semantics, and tools. In *Object Modeling with the OCL*, pages 42–68. Springer, 2002.

[148] Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE transactions on information and systems*, 90(4) :745–752, 2007.

[149] Amirmohammad Sadeghian, Mazdak Zamani, and Azizah Abd Manaf. A taxonomy of SQL injection detection and prevention techniques. In *Proc. of ICICM'13*, pages 53–56. IEEE, 2013.

[150] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2) :38–47, 1996.

[151] Martina Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security. *BT technology journal*, 19(3) :122–131, 2001.

[152] Stuart Schechter, AJ Bernheim Brush, and Serge Egelman. It's no secret. measuring the security and reliability of authentication via "secret" questions. In *Proc. of SnP'09*, pages 375–390. IEEE, 2009.

[153] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12) :21–29, 1999.

[154] Daniel Schwabe and Gustavo Rossi. The object-oriented hypermedia design model (oohdm). *Communications of the ACM*, 35(8), 1995.

[155] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[156] Robert C Seacord and Allen D Householder. A structured approach to classifying security vulnerabilities. Technical report, DTIC Document, 2005.

[157] Guttorm Sindre and Andreas L Opdahl. Eliciting security requirements with misuse cases. *Requirements engineering*, 10(1) :34–44, 2005.

[158] Amina Souag, Camille Salinesi, and Isabelle Comyn-Wattiau. Ontologies for security requirements : A literature survey and classification. In *International Conference on Advanced Information Systems Engineering*, pages 61–69. Springer, 2012.

[159] Amina Souag, Camille Salinesi, Raúl Mazo, and Isabelle Comyn-Wattiau. A security ontology for security requirements elicitation. In *International symposium on engineering secure software and systems*, pages 157–177. Springer, 2015.

[160] Amina Souag, Camille Salinesi, Isabelle Wattiau, and Haris Mouratidis. Using security and domain ontologies for security requirements analysis. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, pages 101–107. IEEE, 2013.

[161] Sampath Srinivas, Dirk Balfanz, Eric Tiffany, FIDO Alliance, and Alexei Czeskis. Universal 2nd factor (u2f) overview. *FIDO Alliance Proposed Standard*, pages 1–5, 2015.

[162] York Sure, Michael Erdmann, Jürgen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. Ontoedit : Collaborative ontology development for the semantic web. In *International Semantic Web Conference*, pages 221–235. Springer, 2002.

[163] Chamseddine Talhi, Djedjiga Mouheb, Vitor Lima, Mourad Debbabi, Lingyu Wang, and Makan Pourzandi. Usability of security specification approaches for uml design : A survey. *Journal of Object Technology*, 8(6) :102–122, 2009.

[164] Li Tan, Oleg Sokolsky, and Insup Lee. Specification-based testing with linear temporal logic. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, 2004. IRI 2004.*, pages 493–498. IEEE, 2004.

[165] Inger Anne Tondel, Martin Gilje Jaatun, and Per Hakon Meland. Security requirements for the rest of us : A survey. *IEEE software*, 25(1), 2008.

[166] Bill Tsoumas and Dimitris Gritzalis. Towards an ontology-based security management. In *20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06)*, volume 1, pages 985–992. IEEE, 2006.

[167] Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. Modeling computer attacks : An ontology for intrusion detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 113–135. Springer, 2003.

[168] Dirk Van Bruggen, Shu Liu, Mitch Kajzer, Aaron Striegel, Charles R Crowell, and John D'Arcy. Modifying smartphone user locking behavior. In *Proc. of SOUPS'13*, page 10. ACM, 2013.

[169] Irma Van der Ploeg. Biometrics and privacy a note on the politics of theorizing technology. *Information, Communication & Society*, 6(1) :85–104, 2003.

[170] Anna Vapen and Nahid Shahmehri. Security levels for web authentication using mobile phones. In *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life*, pages 130–143. Springer, 2010.

[171] Denis Verdon and Gary McGraw. Risk analysis in software design. *IEEE Security & Privacy*, 2(4) :79–84, 2004.

[172] Benoit Vibert, Christophe Rosenberger, and Alexandre Ninassi. Security and performance evaluation platform of biometric match on card. In *2013 World Congress on Computer and Information Technology (WCCIT)*, pages 1–6. IEEE, 2013.

[173] John R Vollbrecht, Bernard Aboba, Larry J Blunk, Henrik Levkowetz, and James Carlson. Extensible authentication protocol (eap). 2004.

[174] Artem Vorobiev and Jun Han Jun Han. Security attack ontology for web services. In *2006 Semantics, Knowledge and Grid, Second International Conference on*, pages 42–42. IEEE, 2006.

[175] Web Application Security Consortium (WASC). Threat classification.

[176] Alfred C Weaver. Biometric authentication. *Computer*, 39(2) :96–97, 2006.

[177] Scott Wright. The symantec smartphone honey stick project. *Symantec Corporation, Mar*, 2012.

[178] Haidong Xia and José Carlos Brustoloni. Hardening web browsers against man-in-the-middle and eavesdropping attacks. In *Proceedings of the 14th international conference on World Wide Web*, pages 489–498. ACM, 2005.

[179] Qinghan Xiao. Security issues in biometric authentication. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 8–13. IEEE, 2005.

[180] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security : Empirical results. *IEEE Security & privacy*, 2(5) :25–31, 2004.

[181] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*, page 569, July 2005.

[182] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.

[183] Pamela Zave. Classification of research efforts in requirements engineering. In *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, pages 214–216. IEEE, 1995.

[184] Moshe Zviran and Zippy Erlich. Identification and authentication : technology and implementation issues. *Communications of the Association for Information Systems*, 17(1) :4, 2006.