Audit et monitorage de la sécurité

Radu State

▶ **To cite this version:**

Radu State. Audit et monitorage de la sécurité. Réseaux et télécommunications [cs.NI]. Université Henri Poincaré - Nancy I, 2009. tel-00442530

HAL Id: tel-00442530
https://theses.hal.science/tel-00442530

Submitted on 21 Dec 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Advanced Security Monitoring and Assessment: Audit et monitorage de la sécurité

MÉMOIRE

présenté et soutenu publiquement le 07/12/2009

pour l'obtention de l'

Habilitation de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Radu State

Composition du jury

Président :	Claude Godart, Professeur, ESSTIN, Nancy
Rapporteurs:	Ludovic Mé, Professeur, Supélec, Rennes Aiko Pras, Associate Professor, University of Twente Stéphane Ubeda, Professeur, National Institute of Applied Sciences of Lyon
Examinateurs:	Pascal Bouvry, Professeur, Université du Luxembourg Olivier Festor, Directeur de Recherche à l'INRIA Nancy, Grand Est Eric Filiol, Directeur de la Recherche ESIEA Jean-Yves Marion, Professeur, Ecole des Mines, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

1 Problématique et activités de recherche

Ce manuscrit synthétise les activités de recherche que nous avons menées, au cours des ces dernières années dans le domaine de la gestion de réseaux, et contient une présentation des enjeux de la discipline pour la décénie à venir. Il contient également un projet de recherche ambitieux visant à répondre aux défis de la sécurité de l'Internet du futur. Nous avons élaboré ce manuscrit en français en complémentarité au manuscrit associé rédigé en anglais. Le français n'étant pas ma langue maternelle, j'ai préféré par un souci de pédagogie d'utiliser la langue anglaise afin de mieux présenter nos travaux de recherche. Le document en français est une synthèse du document associé en anglais. Dans une première partie, nous abordons le monitorage des réseaux ad-hoc et le sécurisation du plan de gestion. Ensuite, nous présentons nos travaux récents sur l'audit de sécurité. Nous abordons dans cette deuxième partie le problème du "fingerprinting", c'est à dire la détection par prise d'empreintes d'une couche protocolaire et/ou d'un équipement. La prise d'empreintes d'un système est une action essentielle dans l'audit de sécurité d'un réseau. L'objectif de ce processus consiste en la détection d'une version spécifique d'un service/équipement par l'analyse du trafic véhiculé sur le réseau. Ceci peut se faire par l'injection de trafic vers la cible (approche connue aussi comme méthode active) ou de manière passive, par observation de trafic uniquement. Dans le dernier cas, le trafic capturé est analysé sans aucune interférence avec la cible. Nous avons développé dans nos travaux cette approche passive en raison de son importance pour la détection d'intrusions et la surveillance de sécurité. Une approche de surveillance est de sa nature passive et la prise d'empreintes des systèmes se montre particulièrement adaptée dans ce contexte. Par exemple, la découverte sur un réseau d'un nouveau périphérique, ayant une empreinte différente peut cacher une intrusion. La prise d'empreintes des systèmes occupe une place importante dans le contenu de ce manuscrit. Nous présentons en détail nos travaux sur ce sujet. Nous avons élaboré deux approches de "fingerprinting". La première s'appuie sur l'analyse des arbres de l'analyse syntactique pour les messages d'un protocole. Nous montrons que les différences issues des arbres de parsing peuvent facilement identifier un système. Nous abordons ensuite la problématique du "fingerprinting" pour le cas des protocoles pour lesquels nous ne disposons pas de spécifications. L'approche que nous présentons est capable d'inférer les divers types de messages et de construire une (ou plusieurs) machine d'états. Ces machines d'états sont les fondements pour définir le comportement. Nous proposons à la suite une approche de fingerprinting qui permet la prise en compte du comportement d'une souche protocolaire.

Nous développons ensuite la problématique liée à la surveillance de sécurité d'un réseau. Celle-ci comprend deux parties principales : le monitoring pour la détection d'intrusions et un pot de miel pour la voix sur IP (VoIP). Nous présentons nos travaux portant sur la sécurité VoIP comportant la conception d'un pot de miel SIP et d'une approche de monitoring pour le trafic de signalisation SIP. L'approche de monitoring s'appuie sur l'analyse des propriétés des flux SIP afin de détecter les attaques spécifiques à la VoIP. Nous nous sommes intéressés aussi à la détection pro-active des failles de sécurité dans les implémentations SIP. La dernière contribution est une approche pro-active élevée pour la détection de failles de sécurité par un processus de type "fuzzing". Notre contribution est une approche de fuzzing permettant la détection de failles dans un protocole complexe de signalisation SIP. La spécificité de notre méthode consiste dans la génération efficace des

données d'entrée ainsi que dans la gestion d'une machine d'état d'une complexité élevée.

2 Supervision de réseaux ad-hoc

Les réseaux mobiles sans fil ont connu un très fort développement ces dernières années pour répondre à la hausse constante des besoins en mobilité. Les exemples les plus significatifs sont probablement les réseaux de téléphonie cellulaire, mais aussi le large déploiement des réseaux locaux sans fil avec la multiplication des points d'accès dans les lieux publics et chez les particuliers. Cette croissance est maintenue par l'augmentation des débits offerts aux utilisateurs avec le développement de la téléphonie de troisième génération [52] et des premières offres commerciales de réseaux sans fil WiMax. L'arrivée de nouvelles normes de transmission de quatrième génération [39] permettra la réelle convergence des différentes technologies mobiles réunies dans un cœur de réseau entièrement sous protocole IP. Ces normes constitueront le socle d'environnements pervasifs dans lesquels les utilisateurs pourront accéder à des services, communiquer, travailler avec les autres usagers en tout lieu, à tout instant et depuis n'importe quel équipement mobile ou non.

Les réseaux ad-hoc représentent une composante de cette évolution et leurs fondements seront inévitablement intégrés aux générations futures de réseaux sans-fil. Ces réseaux auto-organisés [50] sont formés spontanément à partir d'un ensemble d'entités mobiles communicantes, sans nécessiter d'infrastructure fixe préexistante telle qu'une station de base ou un point d'accès par exemple. Les entités mobiles constituent en elles-mêmes le réseau. Elles peuvent être de formes variées : ordinateurs portables, téléphones mobiles, assistants électroniques, capteurs et présentent par conséquent des capacités non homogènes en termes de communication, de puissance de calcul et de stockage. Elles sont libres de se déplacer de manière aléatoire et de s'organiser arbitrairement, si bien que la topologie du réseau est fortement dynamique dans le temps et dans l'espace. Les entités mobiles peuvent intervenir en tant que routeurs pour assurer l'acheminement des paquets entre elles par sauts successifs. Elles communiquent donc soit directement lorsqu'elles se trouvent dans le même voisinage direct, soit par communication multi-sauts le cas échéant en faisant appel à des nœuds intermédiaires. Grâce aux réseaux ad-hoc, l'utilisateur peut déployer son propre réseau très facilement et sans coût supplémentaire. Mais l'apparente simplicité du concept cache de nombreux défis scientifiques et techniques.

La supervision de ces réseaux regroupe un ensemble d'activités qui permet de surveiller et contrôler les réseaux et leurs services. Elle est aujourd'hui confrontée à des environnements de plus en plus dynamiques dont les réseaux ad-hoc en sont un des exemples les plus caractéristiques. Les architectures de gestion traditionnelles, initialement conçues pour les infrastructures fixes, sont inadaptées à la nature dynamique et aux contraintes fortes des réseaux ad-hoc. Ainsi, elles prennent difficilement en charge les changements fréquents de topologie du réseau et sont souvent trop consommatrices en ressources dans un contexte ou lq bande passante et l'énergie sont fortement limitées. De nouveaux verrous scientifiques et techniques doivent donc être levés pour intégrer les réseaux ad-hoc dans une démarche de gestion.

La problématique de mes travaux de recherche a porté sur une nouvelle approche de gestion pour les réseaux et services ad-hoc capable de prendre en compte leur nature dynamique et leurs ressources limitées. Cette approche doit être facilement intégrable aux infrastructures de gestion actuelles, suffisamment flexible pour s'adapter dynamiquement aux changements au sein du réseau ad-hoc, et suffisamment économe pour limiter la charge

induite par l'activité de gestion sur le fonctionnement même du réseau.

Ce travail de recherche s'est organisé autour de trois axes majeurs qui correspondent respectivement à (1) la construction d'un modèle d'information générique pour les réseaux ad-hoc, (2) une réorganisation plus souple du plan de gestion à partir d'une méthode probabiliste et enfin (3) l'adaptation des opérations de gestion, dans le contexte de la gestion de performances en utilisant des techniques de filtrage, et dans le contexte de la gestion de fautes en s'appuyant sur la théorie de l'information.

Nos travaux de recherche ont porté sur une nouvelle approche intégrée, flexible et économe pour la gestion des réseaux et services ad-hoc. Celle-ci se traduit par la construction d'un modèle d'information, la réorganisation du plan de supervision et l'adaptation des opérations de gestion. Cette partie inclut de nombreux résultats expérimentaux obtenus par la simulation. Ceux-ci permettent l'évaluation de nos travaux et sont complétés par des résultats analytiques.

Modélisation étendue de l'information de gestion Nous avons abordé la modélisation de l'information de gestion à travers l'identification des éléments caractéristiques d'un réseau ad-hoc et leurs spécifications à l'aide d'un formalisme commun. Les approches de gestion dédiées aux réseaux ad-hoc négligent le modèle d'information en omettant de le définir ou en le définissant de manière très partielle [14]. Ce modèle est pourtant essentiel car il fournit un cadre formel pour la description des ressources gérées et la structuration de l'information de gestion. Nous définissons notre modèle de manière générique sous la forme d'une extension du modèle commun de l'information (CIM) [4]. Il prend notamment en considération l'organisation du réseau ad-hoc, les échanges en son sein à différentes échelles et la participation des nœuds à son bon fonctionnement. Nous introduisons également un sous-modèle permettant la prise en charge du protocole de routage ad-hoc OLSR [25]. Ce modèle permet l'intégration d'un réseaux ad-hc dans un plan de gestion.

Organisation probabiliste du plan de gestion Nous avons travaillé sur une nouvelle organisation du plan de gestion à partir d'une méthode probabiliste. Une démarche de gestion au sens pur du terme, où l'ensemble des nœuds serait géré à tout moment, est trop stricte pour les réseaux ad-hoc. Au lieu de considérer la gestion du réseau dans son intégralité, nous relachons les contraintes sur le plan de gestion en ne considérant que certains nœuds, ceux qui disposent à la fois d'une forte présence dans le réseau et d'une forte connectivité avec leur voisinage. En partant de cette approche sélective, nous obtenons des garanties sur le pourcentage de nœuds qui seront actifs dans le plan de gestion. Nous avons détaillé [16,18] la méthode algorithmique considérée en définissant une mesure de connectivité spatio-temporelle, l'extraction de composantes spatio-temporelles et le déploiement de mécanismes électifs utilisant la centralité de degré et la centralité par vecteur propre. Nous montrons comment cette méthode peut être intégrée au sein de l'architecture de gestion ANMP [24].

Gestion de performances par filtrage et analyse de graphes Nous avons considéré la gestion de performances dans les réseaux ad-hoc. Nous avons définis de nouvelles méthodes d'analyse permettant de construire une vue fonctionnelle synthétique du réseau

et de déterminer l'impact des nœuds sur son fonctionnement. Une technique de filtrage compare l'état des nœuds dans un voisinage local tandis que l'analyse de graphes met en évidence les dépendances entre ceux-ci. La stratégie permet de faire apparaître les chemins qui sont les plus utilisés lors de communications multi-sauts et qui représentent en fait les *backbones* du réseau ad-hoc. Elle permet aussi de quantifier l'impact des nœuds sur le fonctionnement global du réseau en mettant en évidence les disparités entre nœuds. Cet impact peut être positif (par exemple, les nœuds qui ont une activité de routage importante et qui font partie d'un *backbone*) ou négatif (par exemple, les nœuds qui refusent d'intervenir comme routeurs ou qui consomment abusivement les ressources). Les données issues de cette observation peuvent être exploitées à des fins de reconfiguration, de positionnement de sondes et de provisionnement de nœuds.

Gestion de fautes par inférence Dans une démarche similaire, nous avons abordé la gestion de fautes. Si la gestion de fautes est un problème connu dans les réseaux fixes classiques, la nature fortement dynamique des réseaux ad-hoc amène à repenser cette activité. Nous avons élaboré une méthode de gestion qui consiste à analyser l'intermittence des nœuds ad-hoc et à détecter des fautes/pannes par infèrence. L'intermittence d'un nœud peut être provoquée par des causes bénignes telles que la mobilité ou la dégradation temporaire de la connéctivité. Elle devient cependant pathologique lorsqu'elle est causée par des erreurs de configuration, des pannes de routage, des problèmes de batterie. Un problème majeur consiste à différencier une intermittence pathologique d'une intermittence régulière pour pouvoir ainsi détecter les nœuds non opérationnels. Nous avons proposé pour ce faire une mesure fondée sur la théorie de l'information permettant de caractériser l'intermittence d'un nœud et avons introduit différentes méthodes collaboratives de détection, incluant un mécanisme d'auto-configuration, pour identifier les nœuds pathologiques de manière distribuée. Les résultats obtenus, nous permettent de détecter les nœuds qui ne participent pas au routage ad-hoc ainsi que les nœuds qui essaient de perturber le processus de supervision.

Nous avons été parmi les premiers à aborder la supervision des réseaux ad-hoc. Nos travaux ont aborder la modélisation de l'information de gestion, la spécification d'un plan de gestion, ainsi qu'un plan de gestion distribué capable de s'adapter aux besoins spécifiques des réseaux ad-hoc. Les publications issues de ce travaux sont : [13–15, 17–20].

3 Sécurisation du plan de gestion

L'élargissement des acteurs du monde des réseaux IP (constructeurs et éditeurs de logiciels) a généré une grande hétérogénéité des équipements interconnectés. Ce phénomène, accentué par les intiatives indépendantes de sécurisation des protocoles de gestion de réseaux, rend leur administration d'autant plus difficile. Le défi majeur dans ce thème est aujourd'hui de parvenir à définir une plateforme de sécurité commune pour d'une part réduire les coûts de configuration des politiques de sécurité, et d'autre part avoir une cohérence globale de sécurité sur l'ensemble des équipements.
Pour atteindre cet objectif d'unification des politiques de sécurité, nous avons proposé un premier pas une solution. Dans un contexte où la sécurité devient un enjeu crucial, de nombreuses propositions et implémentations, comme par exemple SNMPv3 [40], ont été intégrées à ces solutions de gestion. On voit alors coexister, au sein d'un même réseau, des agents SNMP (v1, v2, v3), des agents de gestion utilisant XML [22] (Junoscript) [45]), ou encore des interfaces propriétaires (CLI). La figure 1 illustre un tel réseau.

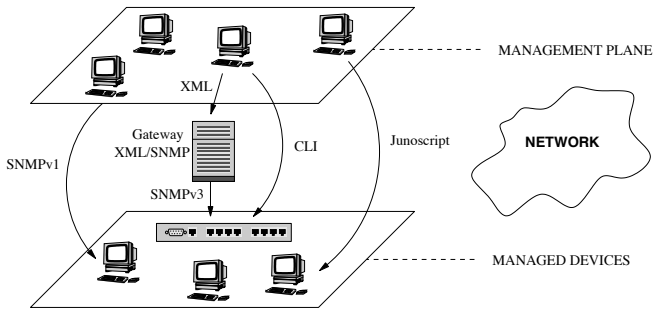


Fig. 1 – Example de réseau avec de multiples interfaces de gestion

Les utilisateurs autorisés à modifier le comportement du réseau sont multiples et doivent être identifiés pour chaque environnement d'administration, et souvent plusieurs fois pour un même environnement si l'on utilise par exemple SNMPv3 car chaque agent dispose d'une politique de sécurité locale. De plus, un même équipement peut avoir plusieurs interfaces de gestion. La difficulté consiste à garder une cohérence dans les différentes politiques de sécurité déployées. Les droits d'accès à une ressource doivent être les mêmes quelle que soit l'interface de gestion utilisée. Si un utilisateur n'est pas autorisé à modifier une table de routage via SNMP, il ne doit pas pouvoir le faire via une interface de commande en ligne par exemple. De la même façon, si la disponibilité d'une donnée est conditionnée par l'utilisation de l'authentification et de la confidentialité, les mêmes contraintes doivent s'appliquer via d'autres interfaces.

Tous protocole de gestion s'appuie sur le modèle agent-gestionnaire. D'un côté, un ensemble d'agents s'éxécutent sur les équipements et sont capables de répondre à des requêtes ou d'émettre spontanément des messages. De l'autre, les managers émettent des requêtes vers les agents. Il existe trois grands types de requêtes :

- récupération des données de l'agent (par exemple des statistiques sur les interfaces réseau),
- positionnement des valeurs de l'agent,
- d'avertissement des gestionnaires en cas d'événement.

Les protocoles de gestion définissent aussi un modèle de données et un schéma d'adressage. Ainsi une donnée est désignée par une suite d'identifiants. Par exemple, en SNMP, 1.3.6.2.2.1 désigne la MIB-2 qui modélise l'état d'une pile IP dans un terminal.

La nature des données manipulées dans le cadre de la gestion de réseau, et donc par SNMP, impose de définir une politique de sécurité. Pour ne pas permettre à un tiers de pouvoir réaliser des opérations de gestion sur les équipements d'un domaine, il est nécessaire de définir clairement quels utilisateurs sont autorisés à interagir avec les agents et quelles sont les permissions de chacun d'entre eux. De plus, il n'est pas souhaitable que quelqu'un puisse accéder aux données manipulées par SNMP en transit sur le réseau. Savoir qui est à l'origine d'un message correspond au mécanisme d'authentification. La confidentialité consiste à ne rendre lisible les données que par les personnes dûment authentifiées et autorisées, c'est-à-dire disposant des informations de sécurité nécessaires au chiffrement et déchiffrement des messages. L'autorisation ou contrôle d'accès, consiste à limiter le champ des opérations possibles à un utilisateur préalablement authentifié.

Maintenir la cohérence dans un tel système est une tâche difficile. Nous avons couvert cette problématique dans le domaine de la gestion de réseaux et nous avons proposé plusieurs approches à ces fins. Nous avons abordé la sécurité des passerelles de gestion en proposant un algorithme de translation de politiques de sécurité. Nous avons considéré les passerelles de gestion multi-protocoles XML/SNMP/CLI pour lesquelles nous avons également conçu un algorithme de détection des incohérences.

Nous avons également conçu une solution d'intégration des mécanismes de sécurité dans un protocole fondé sur XML : Netconf ([37]). Netconf est un protocole de gestion de réseau orienté configuration qui est fondé sur le langage XML. Il est actuellement en cours de standardisation à l'IETF. Netconf spécifie que la couche sur laquelle il repose doit fournir les mécanismes de sécurité voulus. Nous avons proposé un modèle de sécurité et nous avons également évalué les performances obtenues.

Les publications issues de ce travaux sont : [27–32].

4 L'identification et la prise d'empreintes des systèmes

L'appellation « fingerprinting » regroupe en informatique communicante, l'ensemble des techniques qui permettent d'identifier une entité distante (un composant physique, un pilote de périphérique, un système d'exploitation, un service ou une application) par l'empreinte que celle-ci génère en échangeant des messages sur un réseau informatique. Comer et Lin [35] ont, dans le milieu des années 1990 contribué à populariser cette technique qui s'est fortement développée depuis et qui trouve de nombreux domaines d'applications en gestion de réseaux (gestion de l'inventaire, suivi des changements) et en sécurité.

Il existe aujourd'hui deux grandes familles d'approches pour le fingerprinting : les approches actives et les approches passives. Le fingerprinting actif utilise l'injection de messages pour déclencher des réactions spécifiques chez sa cible. Ces réactions (en général des réponses à l'injection), sont utilisées pour identifier la source. Cette identification est réalisée par comparaison de signatures de couples injection/réponse à des couples similaires dans une base de connaissance comportant des triplets (injection/réponse/identification de cible). La difficulté essentielle de cette approche est la construction de la base de connaissances. Ces approches sont en général extrêmement précises et donnent d'excellents taux de réussite. Elles ont cependant le désavantage d'être invasives en générant un trafic supplémentaire perturbateur sur le réseau.

L'approche passive se limite à observer du trafic standard sur un réseau et ne s'appuie que sur ces données pour établir une identification. Naturellement non intrusive, elle donne en général de moins bon résultats que sa contrepartie active.

Un composant essentiel dans tous système d'audit de sécurité ou de surveillance de sécurité est le service de fingerprinting des différents systèmes présents dans une infrastructure de communication. Nous avons proposé deux approches pour le fingerprinting. La première s'appuie sur les différences structurelles identifiées dans les arbres de « parsing » de messages provenant de différents systèmes VoIP. Cette approche considère que la syntaxe utilisée pour les messages protocolaires, est connue d'avance. La deuxième approche se base sur l'apprentissage comportemental et permet un apprentissage automatique des types de messages ainsi que l'usage d'informations issues de l'enchaînement temporel des messages reçus. Ces systèmes sont présentés ci-après.

4.1 Contribution 1 : Fingeprinting structurel

Nous avons proposé une nouvelle méthode de fingerprinting passif basée sur l'exploitation de la structure des messages plutôt que de son contenu, contenu trop facilement modifiable pour leurrer les algorithmes de fingerprinting existants. La motivation de notre recherche consiste dans le fait suivant : si plusieurs développeurs doivent implémenter une couche protocolaire, des choix spécifiques sont faits sur la manière dont le "parsing" du protocole sera fait. Ces choix peuvent servir à identifier un système ayant généré un message.

L'ensemble des éléments qui interviennent dans notre approche de fingerprinting ainsi que leur enchaînement sont illustrés dans la figure 2.

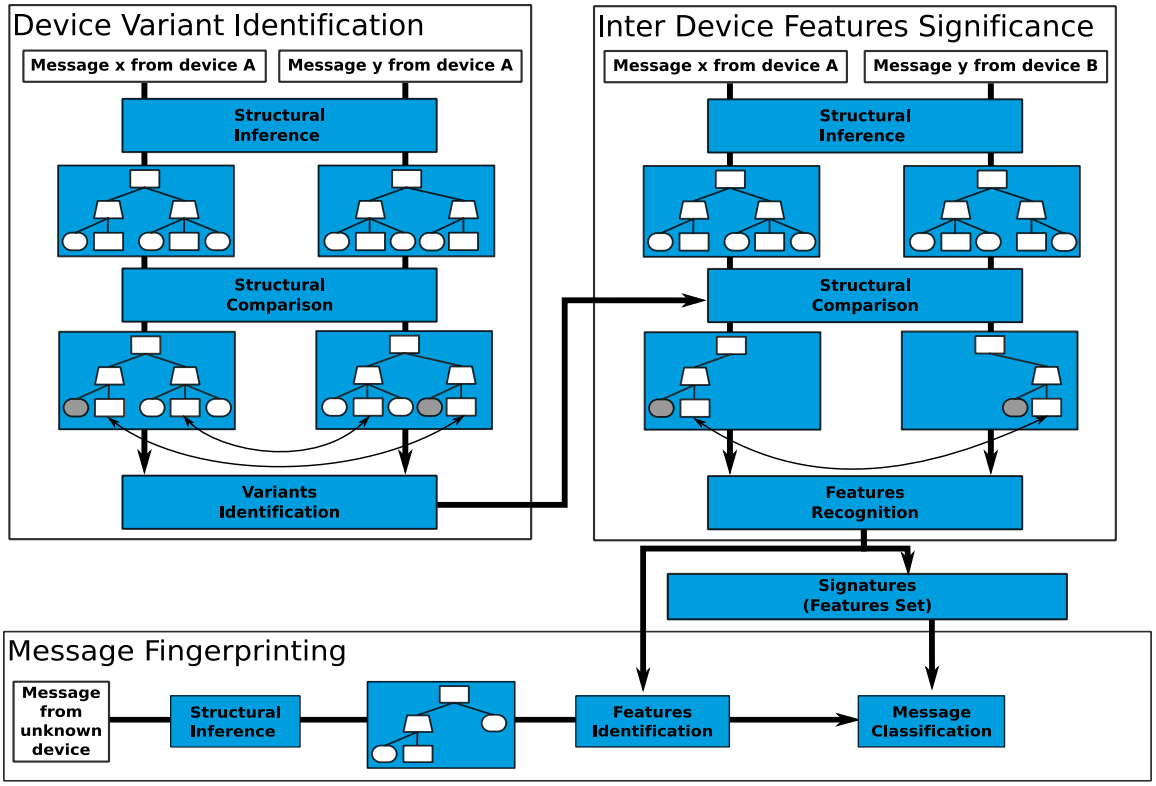


Fig. 2 – Blocs fonctionnels de l'architecture de fingerprinting

Notre approche nécessite la connaissance préalable de la grammaire du protocole (de ses messages). Celle-ci nous sert de surface d'observation. Sur la base de la grammaire, nous effectuons une inférence structurelle d'un ensemble de messages collectés. Une base de signatures est ainsi construite. Une signature se compose d'un ensemble de "shingles" qui peuvent différencier deux arbres. Le choix des nœuds à comparer se base sur un algorithme de calcul des ressemblances pour des structures arborescentes. Partant d'un ensemble de traces des équipements appris, nous recherchons au sein des signatures de chaque équipement celles qui sont uniques pour un équipement donné et qui vont permettre de le distinguer de manière unique. Une fois cette base établie, il suffit lors de l'observation d'un message sur le réseau de rechercher quels sont ses invariants et quels équipements il caractérisent.

Le principe est celui-décrit ci-dessus. L'implémentation réalisée opère de façon complémentaire. La première phase calcule tous les variants d'un équipement. La seconde opère non pas sur les invariants des équipements mais sur les variants et compare les messages de traces de tous les équipements pour en extraire les invariants intra-équipement et surtout ceux d'entre eux qui vont être des variants inter-équipements et former les caractéristiques.

Nous avons implementé le modèle de fingerprinting et l'avons testé sur des traces réseau réelles (21981 messages issus de 26 équipements différents). L'apprentissage a été réalisé sur 15 % des traces. Les résultats de classification de l'ensemble des 21981 messages sont donnés dans la table 1.

Classification	**Vrai Positif** 21422	**Faux Positif** 32
	Faux Négatif 490	**Vrai Négatif** N.A.
Justesse 0.998	**Sensibilité** 0.976	**Spécificité** 0.999

TAB. 1 – Résultats obtenus avec notre méthode de fingerprinting

Ces résultats démontrent la qualité de l'algorithme. L'analyse des faux négatifs montre que ceux-ci proviennent essentiellement de deux primitives de services qui véhiculent trop peu de données pour être classifiées correctement. Ce travail a été effectué dans le cadre de la thèse de Humberto Abdelnur - les publications relevantes sont : [6, 7, 10, 38]. Les travaux ont également fait l'objet d'un dépôt de brevet international en 2007 [8].

4.2 Contribution 2 : Fingeprinting des protocoles sans connaissance préalable de la grammaire sous-jacente

Nous avons également abordé la problématique liée à la prise d'empreintes des systèmes dont nous ne connaissons pas d'avance le protocole utilisé. Nous avons proposé une méthode qui ne requiert aucune connaissance préalable de la grammaire utilisée pour la génération des messages protocolaires. Notre approche est passive et s'appuie sur la classification automatique des messages protocolaires. La contribution essentielle consiste en deux méthodes. La première méthode peut identifier les différents types de messages utilisés par une couche protocolaire. Cette méthode s'appuie sur des fonctions de distance qui mesurent les différences entre deux messages protocolaires. Nous avons proposé quatre fonctions possible et nous avons également évalué leur performance. La deuxième méthode classifie les messages dans des clusters. Chaque cluster est considéré comme un type de message.

Dans la suite, nous passons en revue les quatre fonctions de distance.

Distribution de caractères Cette distance d s'appuie sur les distributions stochastiques des caractères d'un message. Si deux messages m_a et m_b sont du même type, ceci devrait se retrouver dans la similitude de leurs distributions. La première distance dérive de l'entropie relative Kullback-Leibler :

$$char\_dist(m_a, m_b) = \sum_i d(m_a)_i log(\frac{d(m_a)_i}{d(m_b)_i}) + \sum_i d(m_b)_i log(\frac{d(m_b)_i}{d(m_a)_i}) \quad (1)$$

i représente tous les caractères possibles.

Une deuxième distance utilise la distribution relative des caractères ASCII dans un message ([43]) m : rel_char_dist.

$$rel\_char\_dist(m \oplus k) = rel\_char\_dist(m) \forall\ key\ k \quad (2)$$

Une troisième métrique calcule la position des caractères et permet de déterminer les champs ayant des contenus similaires.

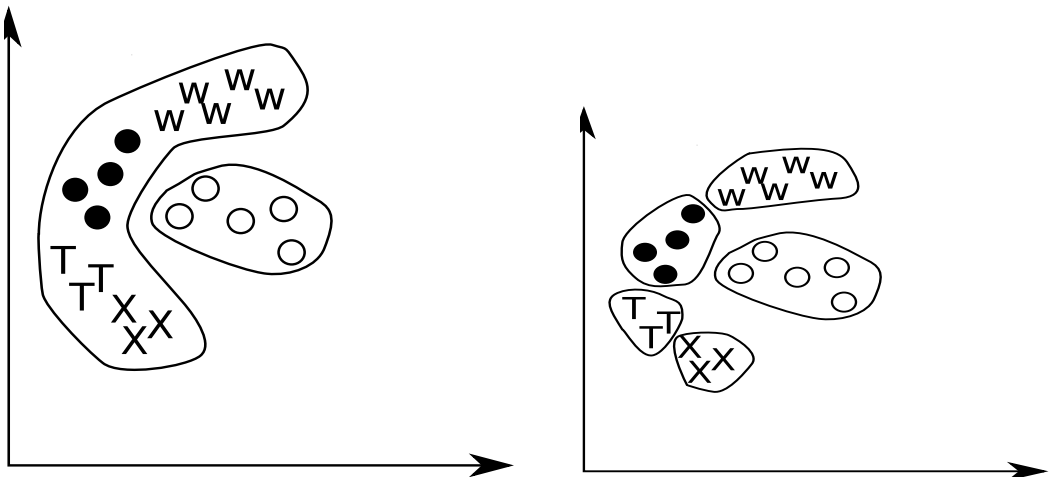
$$char\_pos(m)(c) = \frac{\sum_{i=1}^{i=k} pos(a_i)}{k} \quad (3)$$

i représente la position du caractère c ayant k occurrences dans le message. La fonction $pos()$ retourne l' index d'un caractère.

Finalement, la quatrième métrique prend en compte à la fois les positions des caractères mais également leurs distances par rapport au début du message. En effet les caractères se retrouvant au début du message pèsent plus dans la décision de classification.

$$\forall\ caractere\ c\ présent\ k\ fois,\ p_2(m)(c) = \frac{\sum_{i=1}^{i=k} pos(a_i)^{-1}}{k} \quad (4)$$

Nous avons proposé une méthode combinée (en deux phases) pour analyser les données - pour détecter le bon nombre de clusters - i.e. de classes des messages. La première phase s'appuie sur la méthode de support vector clustering [21, 26], afin de construire



(a) Construction de clusters par SVC

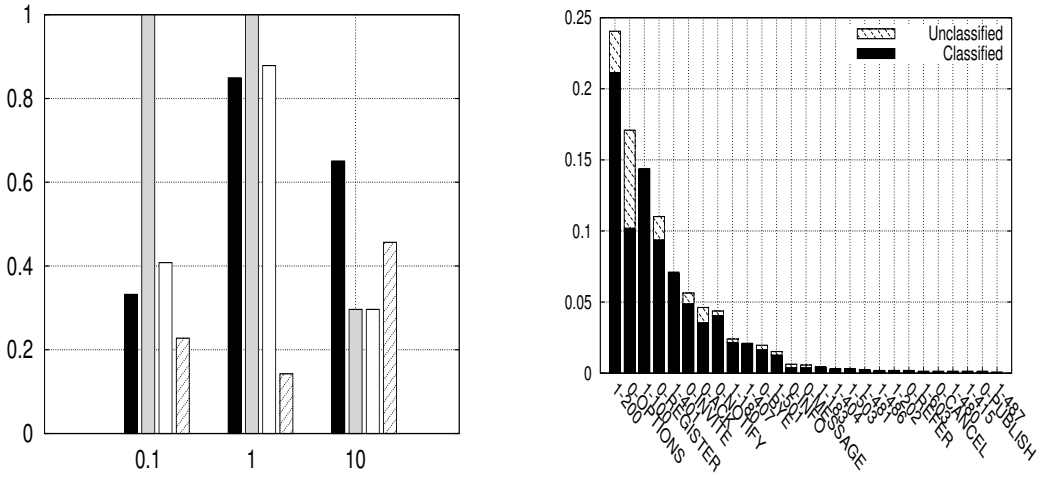
(b) Raffinage des clusters par application de l'algorithme des plus proches voisins

FIG. 3 – Méthode globale

automatiquement les clusters. La deuxième phase utilise une approche de "agglomerative nearest neighbor method" [34].

Nous avons validé la performance de notre approche sur un jeu de tests composé de 1580 messages SIP. Une synthèse des résultats est illustrée dans la figure 4.

Nous avons également analysé deux autres protocoles : SMTP [42] (150 paquets et 10 types) ainsi que IMAP [33] (289 paquets et 24 types de message). Nous avons constaté que la méthode SVC combinée améliore de manière significative la performance de la classification. Ce travail a été effectué dans le cadre de la thèse de Mr. Jerome François - les publications issues de ces travaux sont : [10, 38]



(a) Précision de notre méthode

(b) Détails des clusters

FIG. 4 – Méthode globale

4.3 Contribution 3 : Fingeprinting comportemental

Nous avons proposé une méthodologie de fingerprinting qui s'appuie sur la notion de comportement. Le comportement est modélisé par un ensemble de « Random Tree Parametrized Extended Finite State Machines - TR-FSM. Chaque machine de ce type représente une extension stochastique et temporelle du modèle introduit par D. Lee dans [51].

Le modèle TR-FSM est un tuple $M =< S, s_{init}, I, O, \vec{X}, T, \vec{Y} >$: avec

- S un ensemble fini d'états $|S| = n$
- s_{init} l'état initial
- $I = \{i_0(\vec{v_0}), i_1(\vec{v_1}), \ldots, i_{p-1}(\vec{i_{p-1}})\}$ est l'alphabète de données d'entrée. Le cardinal de l'alphabet est p. Chaque symbole est paramétré par un vecteur.
- $O = \{o_0(\vec{w_0}), o_1(\vec{w_1}), \ldots, o_{q-1}(\vec{w_{q-1}})\}$ est l'alphabet de sortie. Le cardinal de l'alphabet est q.
- $\vec{X}$ est un vecteur de variables de configuration.
- T est un ensemble finis de transitions, tel que pour chaque t dans T, $t =< s_1, s_2, i(\vec{v}), o(\vec{w}), P(\vec{X}, i(\vec{v})), Q(\vec{X}, i(\vec{v}), o(\vec{w})) >$.
  s_1 et s_2 sont les états initiaux respectivement finaux. i est l'entrée qui a généré la transition. Le prédicat $P(\vec{X}, i(\vec{v}))$ représente la condition requise pour réaliser la transition. $Q(\vec{X}, i(\vec{v}), o(\vec{w}))$ dénote l'action issue par l'exécution de la transition.
- $\vec{Y}$ est un vecteur aléatoire de dimension $n - 1$. Ce vecteur modélise le temps recquis pour effectuer les transitions. Le temps nécessaire pour chaque transition est modélisé comme une variable aléatoire.

Nous définissons une condition liée à la topologie des états - les transitions sous-adjacentes doivent former un arbre. La prise d'empreintes se fait de la manière suivante : une analyse passive du trafic génère des structures TR-FSM. Une structure arborescente est construite en analysant plusieurs sessions. Chaque nœud dans une structure TR-FSM est le le type du

message. Deux nœuds (A et B) sont liés dans l'arbre si le message de type B a directement suivi le message de type A dans la capture. Dans le cas du protocole SIP, nous illustrons un exemple dans les figures 5(a) et 5(b). Les types de messages dans ce cas sont les types de requêtes SIP et des codes de réponse. Les préfixes (? ou !) indiquent le sens du message (reçu ou émis).

La figure 5(b) montre une structure TR-FSM pour un serveur Asterisk SIP. Une transition est indiquée par une flèche entre deux états. Le vecteur $\vec{Y}$ correspond à la moyenne des temps de transitions.

La construction d'une structure Tree-FSM s'appuie sur l'identification du préfixe commun maximal afin d'établir le chemin commun dans l'arbre résultat.

Les schémas 6(a) and 6(b) illustrent l'idée de base derrière le fingerprinting comportemental. Dans la figure 6(a) nous retrouvons une empreinte pour un téléphone SIP logiciel (softphone). Dans la figure 6(b) nous montrons le même scénario pour un téléphone physique (hardphone). Même si les deux téléphones partagent une structure commune, (la sous-structure grise) l'un de deux réalise une transition dix fois plus vite que l'autre. Le problème de "fingerprinting" se pose de la manière suivante :

- étant donné un ensemble d'implémentations $C = \{M_1, M_2, \ldots, M_k\}$ ainsi que l'ensemble d'empreintes comportementales $\{N_{j1}, N_{j2}, \ldots, N_{jp}\}$ nous cherchons un mécanisme de classification qui peut correctement mettre en correspondance chaque implémentation M_j, avec sa classe correspondante.
- étant donné un ensemble d'empreintes comportementales $\{N_1, N_2, \ldots, N_p\}$, l'objectif consiste à déterminer un ensemble d'implémentations $M = \{M_1, M_2, \ldots, M_k\}$ tel que pour chaque N_j,$1 \leq j \leq p$, il existe i, $1 \leq i \leq k$ ou N_j est une empreinte comportementale de M_i.

Nous supposons que deux fonctions de distance existent. Celles-ci évaluent deux structures du type TR-FSM M_1 and M_2 :

- $\Delta_1(M_1, M_2)$ est une distance prenant en compte les différences entre les états et les transitions,
- $\Delta_2(M_1, M_2)$ est une distance prenant en compte les différences entre les états, les transitions ainsi que les entrées du vecteur $\vec{Y}$.

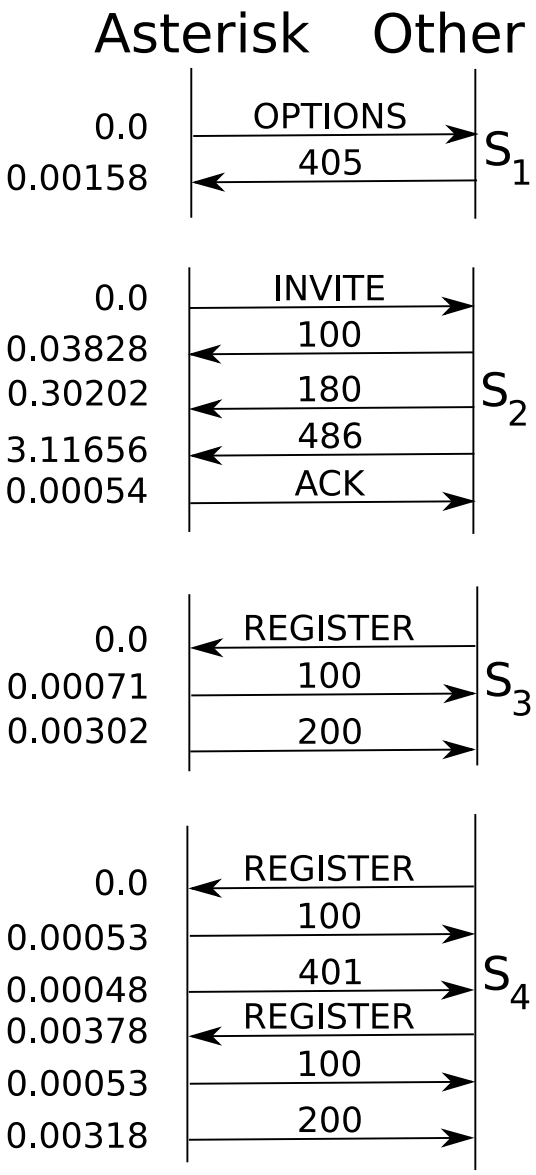
Nous utilisons une méthode basée sur les machines à vecteur support qui peut recevoir comme données directement les structures TR-FSM. Pour ceci, nous avons étendu la méthode décrite dans [23]. L'objectif est de pouvoir comparer deux structures de type TR-FSM et retourner une valeur qui mesure leur similitude. En principe, si les sous-structures (sous-arbres) sont similaires, les structures composées devraient l'être aussi. Afin de comparer les sous-structures, nous procédons à une extraction de tous les chemins (paths) entre la racine et tous les nœuds de type feuille de l'arbre $paths^i$. Cet ensemble est composé de m chemins : $path_1^i, \ldots path_m^i$. Un chemin individuel est dénoté par $path_j^i$.

Les nœuds situés sur un chemin $path_j^i$ sont dénotés par $nodes(path_j^i)$. Pour un ensemble de chemins, la fonction $nodes(paths^i)$ retourne tous les chemins de cet ensemble.

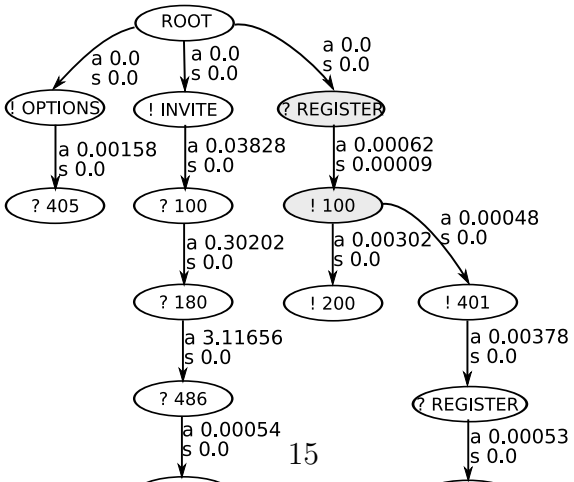
L'intersection de deux arbres t_i and t_j est définie par :

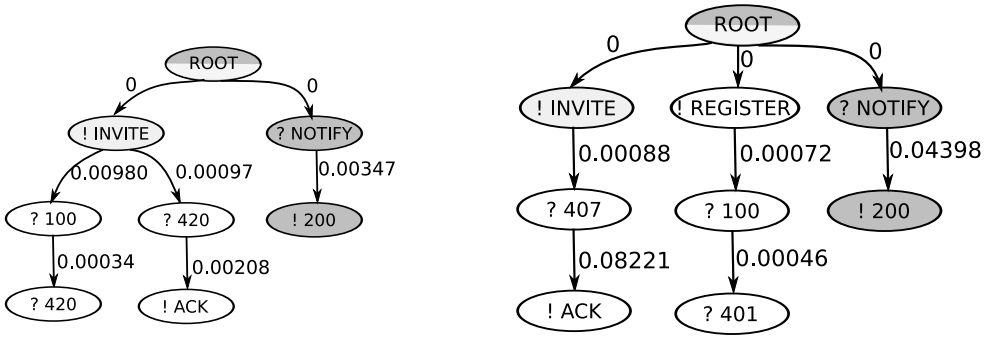
$$I_{ij} = nodes(paths^i) \cap nodes(paths^j) \tag{5}$$

Pour tous les chemins (dans cette intersection), on peut définir une mesure de similitude comme suit :



(a) Sessions avec les temps de transition





(a) Twinkle 1.10 (Softphone) (b) Cisco 7940 firmware 8.93 (hardphone)

FIG. 6 – Exemples de sessions pour de téléphones VoIP

$$inter\_sim = \sum_{\substack{p \in I_{ij} \\ nodes(path_k^i)=p \\ nodes(path_l^j)=p}} weight(paths_k^i, paths_l^j) \tag{6}$$

La fonction $weight(paths_k^i, paths_l^j)$ prend en compte tous les temps de transitions des arrêts qui forment l'ensemble I_{ij} .

$$weight(p_1, p_2) = \sum_{n_p \in p_1} e^{-\alpha |f_{delay}(n,p_1) - f_{delay}(n,p_2)|} \tag{7}$$

ou $f_{delay}(n, p)$ retourne le temps moyen de la transition ayant comme destination (dans le chemin p) le nœud n.

Finalement, la fonction kernel (noyau)utilisée est :

$$K(t_i, t_j) = K_1(t_i, t_j) K_2(t_i, t_j) \tag{8}$$

ou :

$$\begin{aligned} K_1(t_i, t_j) &= \sum_{\substack{p \in I_{ij} \\ nodes(path_k^i)=p \\ nodes(path_l^j)=p}} \sum_{n_p \in p} e^{-\alpha |a_{delay}(n,p_1) - a_{delay}(n,p_2)|} \\ K_2(t_i, t_j) &= \sum_{\substack{p \in I_{ij} \\ nodes(path_k^i)=p \\ nodes(path_l^j)=p}} \sum_{n_p \in p} e^{-\alpha |s_{delay}(n,p_1) - s_{delay}(n,p_2)|} \end{aligned} \tag{9}$$

Les résultats du tableau 2 donnent une vue d'ensemble sur les performances obtenues. Les nuances de gris du tableau indiquent la précision de notre approche. Notre solution ne permet pas l'identification des systèmes ayant une seule session (dans la base de signatures). La colonne la plus ambrée correspond aux sessions d'un minimum de 5 sessions de

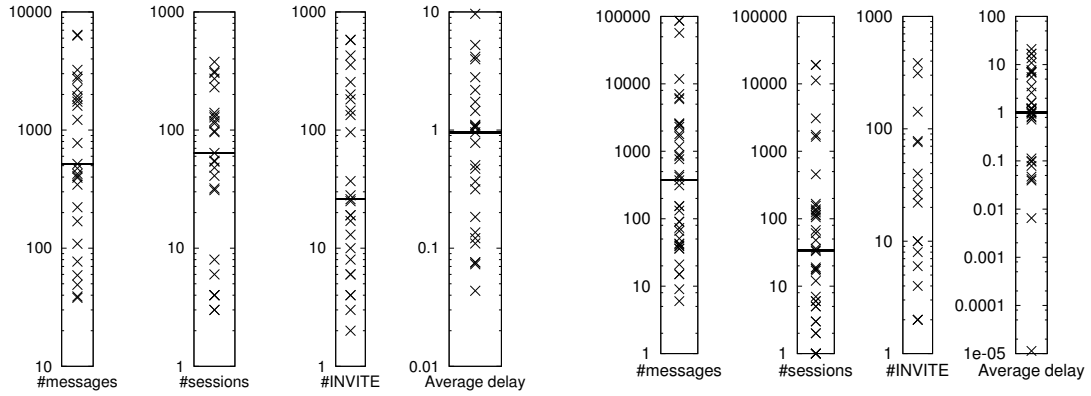
Nombre de sessions de calibrage	Nombre de sessions de test				
	1	5	10	20	40
1	0.682 (0.009)	0.819 (0.013)	0.830 (0.013)	0.805 (0.031)	0.745 (0.034)
5	0.469 (0.028)	0.858 (0.013)	0.905 (0.011)	0.883 (0.025)	0.800 (0.035)
10	0.376 (0.044)	0.809 (0.011)	0.894 (0.013)	0.873 (0.021)	0.819 (0.035)
20	0.272 (0.028)	0.656 (0.028)	0.821 (0.015)	0.864 (0.015)	0.837 (0.012)
40	0.221 (0.027)	0.469 (0.026)	0.627 (0.030)	0.764 (0.037)	0.762 (0.038)

$< 50\%$	50-70%	70-80%	80-85%	85-90%	$\geq 90\%$

TAB. 2 – `testbed dataset` : Performance du système - précision moyenne

calibration. Si nous utilisons comme paramètres des valeurs entre 5 (pour l'entraînement) et 10 (pour l'évaluation), nous obtenons une performance maximale de ($\sim 90\%$).

En regardant les performances liées à l'identification par équipement (voir le tableau 3), la meilleure configuration reste la même. La performance obtenue est de 65%. Ces résultats sont dus à la faible présence de quelques équipements (voir le schéma 7(a)).



(a) Testbed local

(b) Opérateur télécom

FIG. 7 – Testbed : statistiques

Ce travail a été effectué dans le cadre de la thèse de Jérome François - la publication relevante issue est [38].

Train session	Test session size				
	1	5	10	20	40
size 1	0.504 (0.011)	0.542 (0.034)	0.553 (0.032)	0.535 (0.044)	0.529 (0.043)
5	0.294 (0.026)	0.605 (0.035)	0.647 (0.035)	0.648 (0.047)	0.580 (0.045)
10	0.224 (0.028)	0.550 (0.017)	0.625 (0.023)	0.636 (0.024)	0.599 (0.047)
20	0.145 (0.021)	0.452 (0.050)	0.572 (0.030)	0.615 (0.045)	0.622 (0.027)
40	0.109 (0.028)	0.316 (0.030)	0.399 (0.032)	0.505 (0.050)	0.522 (0.038)

$< 30\%$	30-40%	40-50%	50-55%	55-60%	$\geq 60\%$

TAB. 3 – `testbed dataset` : Précision du système (par type d'équipement)

4.4 Synthèse

Etre capable d'identifier de façon explicite l'ensemble des entités qui opèrent dans une infrastructure est essentiel à toute opération d'audit de sécurité. Nous avons pour cela conçu une nouvelle méthode passive exploitant les propriétés structurelles des messages échangés sur un réseau. Notre approche est automatique et son évaluation a démontré son efficacité dans une infrastructure Voix sur IP intégrant un nombre conséquent de terminaux hétérogènes.

Nous avons conçu une nouvelle méthode pour générer des systèmes de fingerprinting basés sur l'analyse structurelle des messages d'un protocole. Notre solution permet d'automatiser ce processus lourd, manuel dans la grande majorité des approches concurrentes. Le service de fingerprinting implantant notre méthode a été réalisé en Python et testé sur plusieurs instances de réseaux VoIP. Les évaluations montrent d'excellents résultats.

5 Surveillance des systèmes VoIP

Ce chapitre présente nos contributions portant sur les approches conceptuelles et pratiques de surveillance relevantes dans un contexte VoIP. Nous avons abordé cette problématique en proposant deux contributions majeures. La première consiste en la définition d'un pot de miel (honeypot) spécifique aux services de type VoIP. La deuxième contribution définit une méthode basée sur les machines à support vectoriel pour surveiller la signalisation VoIP. Parmi les menaces spécifiques à la VoIP, nous mentionnons :

1. Le déni de service (DoS) spécifique au domaine/plan VoIP. Il peut avoir des conséquences catastrophiques : chute totale de l'infrastructure, paralysie de services de secours, etc.
2. Les virus, vers et portes dérobées peuvent permettre la prise de contrôle à distance des téléphones IP et des serveurs mandataires à de fins malicieuses. La reconstruction d'une conversation VoIP est facilement réalisable en capturant et en décodant le trafic VoIP. Les dénis de service peuvent se faire à la fois par une inondation massive du plan de signalisation, ou par l'injection de paquets SIP malformés, capable de mettre hors jeux un équipement SIP.
3. L'usurpation d'identité (par changement de l'identificateur de l'appelant, ou du destinataire) peut facilement servir à des attaques contre la vie privée de personnes. Une attaque de type Vishing (Voice over IP Phishing) pourra aider a attaquant à se faire passer pour une autre entité (banque, institution/service public) er pourra avoir des conséquences majeures sur les usagers VoIP que sur ceux de la téléphonie classique.
4. Finalement, le SPIT (Spam over Internet telephony) représente la transposition du SPAM dans le monde VoIP. Son effet par contre est beaucoup plus gênant. Les appels téléphoniques en quantité massive et à des heures de repos peuvent facilement déranger ses destinataires (qui peuvent utiliser un service VoIP ou avoir un service de téléphonie classique).

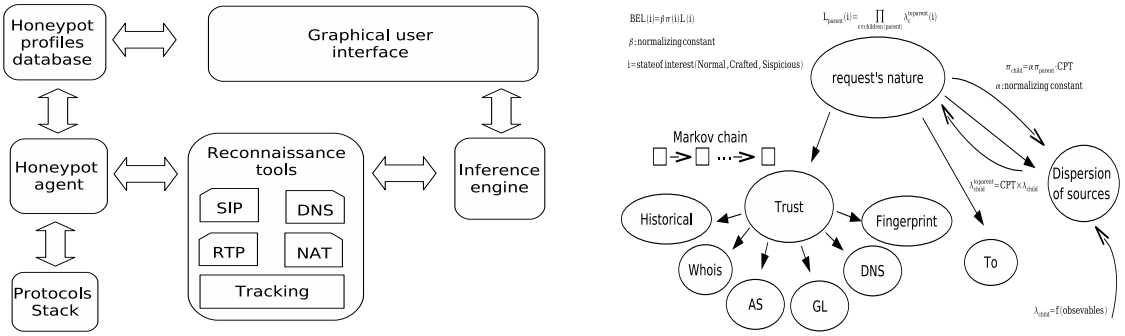
5.1 Architecture d'un téléphone de miel (pot de miel)

Par définition, un pot de miel consiste en un environnement ou des vulnérabilités sont introduites délibérément dans le but d'y observer des attaques et des intrusions. Les pots de miel sont déployés sur des systèmes n'ayant pas de rôle opérationnel – c'est à dire que tout trafic à destination d'un pot de miel est fortement suspect. Un pot de miel réseau est un outil capable de servir différents buts : 1) il peut détecter les attaques ciblées sur le réseau, 2) il peut également révéler des machines du même réseau (administratif) ayant un comportement malicieux (infectées, ou sous le contrôle d'un attaquant) et 3) il peut servir à mieux connaître les attaquants (en récupérant les traces de leurs actions, leurs outils ainsi que par la capture de leur échanges/communications).

Nous avons conçu un pot de miel dédié au service VoIP. L'idée consiste en l'émulation de faux usagers VoIP qui peuvent être appelés via VoIP. Pendant ces appels, notre système récupère le plus d'informations possible sur la source de l'attaque. Techniquement, cela se traduit par la conception d'un téléphone SIP (user agent) qui s'enregistre avec son adresse

IP et un nombre d'URIs dans un serveur de d'enregistrement SIP. Les URIs de "honeyphone" ne sont pas déclarées au monde extérieur. Les usagers légitimes ne devraient pas contacter ces faux clients. Cependant, en cas d'attaque/action malicieuse (énumération de domaine, SPIT, DoS sur le domaine), celle si sera détectée et le téléphone SIP va émuler un usager. L'architecture fonctionnelle de "honeyphone" proposée est donnée dans la figure 8(a). Elle se compose de 5 composantes :

1. L'agent : est le coeur de l'architecture de pot de miel et la partie intelligente de cette application. Il est responsable de l'acceptation les appels arrivants et a en charge d'enquêter sur les attaques. Il réagit suivant un profil prédéterminé.
2. La pile des protocoles (SIP, RTP, SDP) : est construite en respectant les normes standards de ces protocoles. Elle est responsable de la construction et de l'analyse syntaxique et grammaticale des messages, ainsi que de la transmission et de la réception des paquets à travers la couche transport.
3. La base de données des profils : celle-ci contient plusieurs profils de configuration qui permettent de configurer les réactions de l'agent. Il sert à bien configurer l'agent par rapport à son environnement. Cette configuration comprend à la fois les paramètres associés au service, ainsi que le comportement spécifique au honeypot. Les réactions du honeypot sont réalisées par deux composants essentiels :
4. Les outils de reconnaissance : ces outils permettent la prise d'empreintes IP et TCP (par l'intégration des approches actives ou passives - voir nmap/p0f) ainsi que la prise d'empreinte niveau service (SIP).
5. Le moteur d'inférence interprète automatiquement les résultats obtenus par les outils de reconnaissance et s'appuie sur des modèles d'inférence de type bayesien afin de fournir une vision globale.



(a) Honeyphone architecture

(b) Analyse d'un message SIP

FIG. 8 – Pot de miel VoIP

Les messages SIP reçus par un honeyphone sont analysés afin d'évaluer leur pertinence. L'outil de reconnaissance réalise les étapes suivantes :

On vérifie pour chaque entité (adresse IP, nom de serveur) du message INVITE les informations suivantes, et pour chaque test, une valeur entre 0 et 1 indique le degré de confiance dans l'information.

– `WHOIS` : Des informations sur les enregistrements connus pour cette adresse. Une valeur de confiance est issue de ce processus.
– informations DNS : nous utilisons de requêtes DNS, Reverse DNS, DNS SRV [2] ou ENUM [3] afin d'estimer si le message SIP est authentique
– `AS` : les informations sur le domaine autonome.
– `GL` : la location géographique correspondante à l'adresse de la hôte IP.
– `Has Port` : on vérifie si le port (SIP ou RTP) est réellement ouvert
– `Fingerprint` : l'empreinte de la source du message afin de détecter les discordances entre le champ `User-Agent` et celle-ci.
– `Historical cache` : On analyse l'historique de l'adresse IP afin de détecter des anomalies potentielles.

Ces informations sont à la suite utilisées par un réseau bayesien (voir la figure 8(b)) pour classifier l'état général du système. Les nœuds évidence du réseau bayesien correspondent aux valeurs de confiance établies par le processus précédent. Les hypothèses (des variables causales dans le nœud racine du réseau) correspondent aux objectifs du message analysé. Nous considérons trois cas possibles :

– `Crafted` : le message comporte des anomalies qui démontrent que son origine est suspecte - c'est à dire les informations contenues dans les champs SIP sont très probablement générées par un outil d'attaque.
– `Suspicious` : le message semble légitime -cependant les serveurs intermédiaires ont une mauvaise réputation
– `Normal` : message SIP normal et légitime.

5.2 L'analyse du trafic SIP

La deuxième contribution, que nous avons apportée, porte sur la surveillance et la détection automatique des attaques dans la VoIP. En général, une anomalie dans la VoIP peut intervenir à tous les niveaux – dans la couche IP, ainsi que dans le plan de signalisation (SIP) et des données de type voix (RTP). Nous avons abordé en priorité la détection d'anomalies spécifiques au niveau VoIP. Notre contribution porte sur une approche basée sur les machines à vecteur support –(support vector machines - SVM) pour la surveillance du trafic SIP. Une deuxième extension annexe propose une méthode bayesienne, afin de corréler les résultats issus du processus de classification SVM. Nous avons identifié 37 caractéristiques qui peuvent servir pour l'analyse du plan de signalisation SIP. Ces caractéristiques servent comme données d'entrée à un processus d'apprentissage (voir les figures 5.2 et 5.2) basé sur les SVMs. Le fonctionnement du processus de classification peut être assuré en mode « online » ou « offline ». Afin d'évaluer la performance du système, nous avons implémenté la version « offline ». Bien qu'une liste exhaustive de toutes les caractéristiques ne sera pas détaillée dans cette synthèse, les attributs/mesures sont répartis en cinq groupes : statistiques générales, statistiques basées sur l'identifiant Call-Id, statistiques prenant en compte la distribution des états finaux des sessions SIP (dialogues/Call-ID), la distribution des requêtes, ainsi que la distribution des réponses.

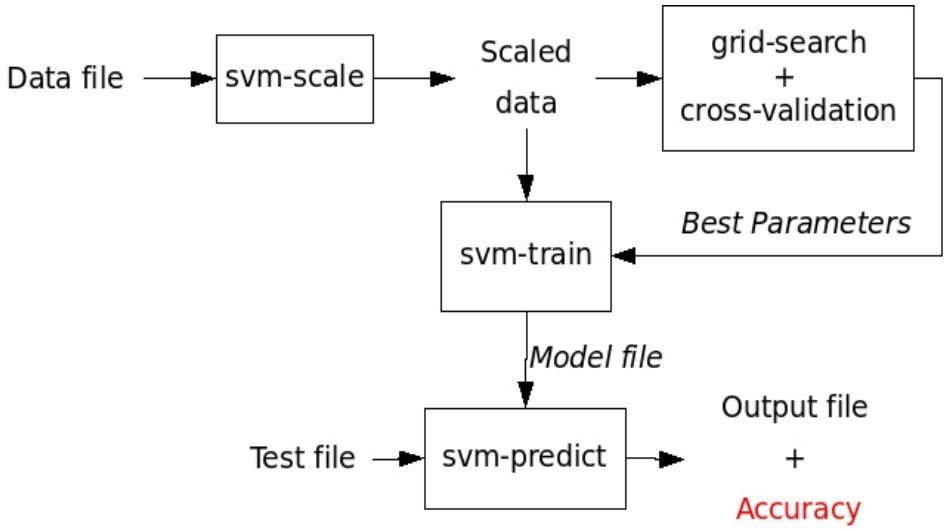


Fig. 9 – SVM Flow Chart

Nous considérons que le temps de surveillance est divisé en périodes (époques) et qu'il calcule ces caractéristiques pour chaque période (voir figure 11). Un classificateur est responsable de détecter si une période représente une attaque. Nous avons considéré dans notre approche un apprentissage de type supervisé : en mode entraînement, des vecteurs labelisés sont fournis pour en construire un modèle de classification de trafic (en principe "normal" et "attaque"). En mode test, des vecteurs non labelisés sont fournis afin d'être prédits comme "normal" ou "attaque".

Les résultats individuels (étape par étape) peuvent être corrélés afin de fournir une vision synthétique. Nous avons également contribué à cette problématique en proposant une méthode bayesienne. Les observations spécifiques à la couche SIP forment les nœuds

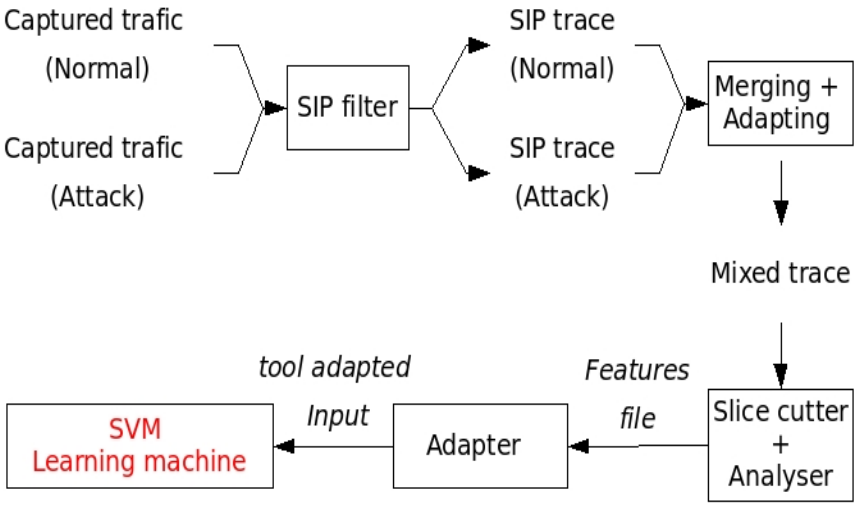
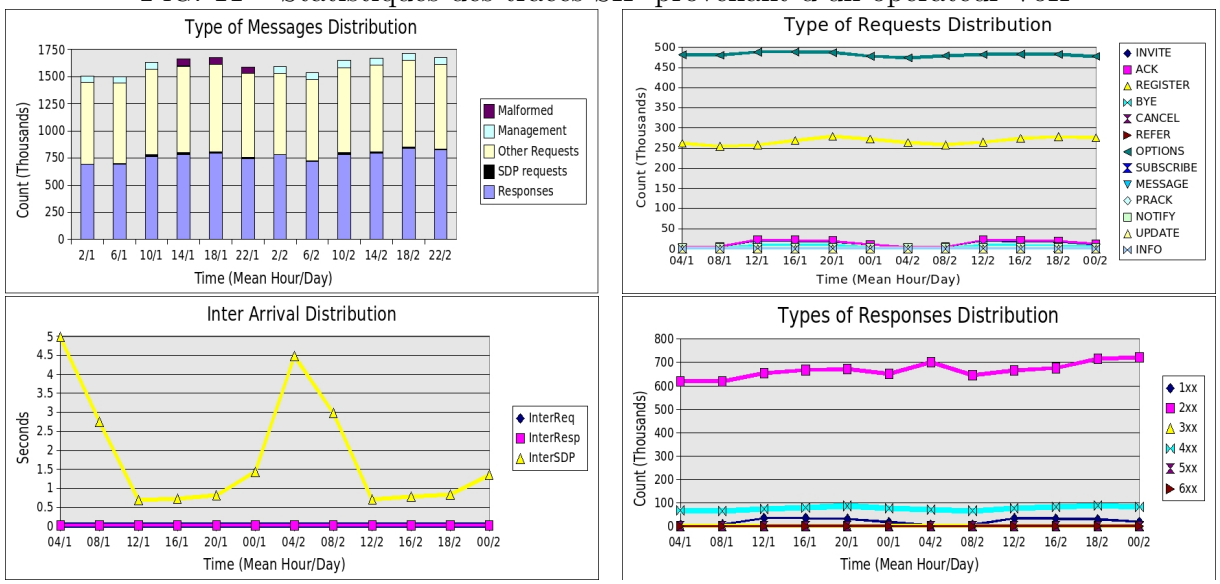


FIG. 10 – Analysis Flow Chart

FIG. 11 – Statistiques des traces SIP provenant d'un opérateur VoIP



de type "évidence". Les causes sont les différentes causes d'attaque (SPIT, DOS, Vishing). Une cause spéciale est celle qui correspond à l'état de normalité. Ce modèle probabiliste est utilisé afin de pouvoir estimer à tout moment l'état du service.

5.3 Evaluation de performances

Nous avons utilisé plusieurs traces VoIP afin de valider notre approche. Nous avons utilisé des traces provenant d'un fournisseur de services VoIP réel. Nous avons aussi produit des traces d'attaque dans un réseau local en utilisant 100 utilisateurs VoIP émulés. Les utilisateurs émulés sont des robots VoIP preconfigurés selon des profils de comportement déterministes ou aléatoires. Par exemple, le robot va choisir aléatoirement pour combien de temps il va "sonner", s'il va répondre ou non à l'appel, pour combien de temps il va "écouter" après avoir "décroché", s'il va diriger l'appel vers une autre destination (ex. boîte vocale) ou non, etc. Les deux types de traces peuvent être mixes puisque les deux réseaux adjacents ont quasiment la même architecture et puisque les traces sont collectées au même point de l'architecture dans les deux cas (un proxy OpenSER à l'entrée/sortie d'un domaine VoIP). Nous avons également injecté des attaques dans les traces collectées. Nous avons dû suivre à cette démarche en supposant que tout le trafic VoIP collecté chez l'opérateur est normal. Ce pas nous a permis de labélliser les donnés d'entrée, car nous connaissons dans ce cas la bonne étiquette pour chaque ensemble de données. Les injections effectuées concernent :

1. Des inondations générées à l'aide de l'outil Inviteflood[1].
2. Des attaques de type SPIT partiel (voir la figure 4) : l'attaquant ne connaissant pas les extensions du domaine cible, il sera obligé d'envoyer massivement des requêtes INVITE vers des extensions aléatoires. Cette attaque est similaire à un balayage (scanning) utilisant des messages INVITE. En principe, il devrait générer beaucoup de réponses de type "User Not Found".
3. Dans une situation de SPIT **total** (voir la figure 5), l'attaquant connaît auparavant les extensions valides dans le domaine cible et il y envoie ses requêtes. Les deux formes de SPIT (partiel et total) sont générées à l'aide de l'outil Spitter/Asterisk [36]. Dans nos expériences, nous avons configuré le taux de réussite du SPIT partiel à 10% (celui de SPIT total étant de 100 %).

5.4 Synthèse et conclusions

Avec la croissance continue des attaques contre la VoIP, des mécanismes de monitorage et de gestion de sécurité deviennent indispensables. Dans le cadre de cette problématique, nous avons indentifié les pièces manquantes dans les plans de défense actuels. Tout en accentuant la spécificité de l'application visée d'une part, et sa sensibilité financière et confidentielle d'autre part, ce travail est tout à fait complémentaire aux efforts poursuivis à la fois dans la communauté recherche et dans l'industrie. Nous avons proposé des solutions intégrées qui comblent le manque en matière de monitorage de trafic, de défense préventive et de corrélation des événements. En effet :

[1] `http://www.hackingvoip.com/sec_tools.html`

Tab. 4 – Détection du SPIT partiel

# Appels Concurrents	Vrais Positifs (%)	Vrais Négatifs (%)
RBF ; C= 1 ; $\gamma = 1/38$; Précision/apprentissage = 99.0249		
1 10 50 100	0 (0/3697) 1.30 (10/766) 10.01 (62/619) 18.31 (102/557)	100
Linear ; C=1 ; Précision = 99.0197		
1 10 50 100	0 (0/3697) 2.09 (16/766) 10.66 (66/619) 19.39 (108/557)	100

Tab. 5 – Détection du Full SPIT

# Appels Concurrents	1	10	50	100
RBF ; C= 1 ; $\gamma = 1/8$; Précision/apprentissage = 98.9057				
Vrais Positifs	0.03 2/7015	3.05 15/492	12.18 85/698	23.41 184/786
Vrais Négatifs	100			

– nous avons proposé un système de monitorage de trafic de signalisation (SIP) se basant sur la théorie de l'apprentissage. L'idée de base est de découper le trafic en de petits morceaux ou créneaux et d'en extraire des valeurs pour des variables statistiques pré-définies ou attributs. Pour chaque créneau, les valeurs de ces attributs sont regroupées dans un vecteur qui sera donné à la machine d'apprentissage pour le classifier comme normal ou suspects. Des règles de corrélation sont ainsi appliquées pour inférer une conclusion sur le trafic en cours. Nous avons utilisé deux approches différentes : les réseaux Bayesiens et les machines à vecteurs support et nous avons comparé leurs performances.
– Nous avons présenté une approche innovante pour désigner un pot de miel spécifique à la VoIP. Le pot de miel est muni d'une interface applicative qui lui permet de gérer un grand nombre des outils réseau. En plus de collecter et enregistrer les attaques, le pot de miel est capable de mener une enquête en temps réel sur les messages reçus. Un composant important dans son architecture est un moteur d'inférence qui peut juger si un message reçu est une faute de routage ou énumération, s'il s'agit d'un message lancé pour exploiter une certaine vulnérabilité, ou s'il s'agit d'un appel qui semble être correct mais qui n'est pas digne de confiance. Des évaluations des exemples simulés montrent l'efficacité de notre procédure d'enquÍte et l'exactitude des décisions prises par le moteur d'inférence.
– Nous avons présenté une approche distribuée à multi-niveaux pour la corrélation des événements de sécurité dans un domaine VoIP. Notre approche est basée sur

des systèmes de détection d'intrusion ainsi que sur les autres composants de défense (pot de miel, moniteur de trafic).

Nous avons expérimenté notre approche de monitorage à l'aide des traces réseau d'un fournisseur de service VoIP et des traces d'attaques générées dans notre banc d'essai et insérées dans les traces "normales". Nous avons comparé les performances de deux techniques utilisées (réseaux Bayesiens et SVM) pour détecter l'ensemble des attaques à court-terme et à long-terme. Les résultats ont montré une grande capacité pour un déploiement temps-réel et une grande précision de détection des attaques DoS d'inondation et du SPIT. Les réseaux Bayesiens se montrent plus performants en détectant des variantes d'une même attaque et dans la détection d'anomalies (lorsque seules des données de trafic normal sont disponibles pour apprentissage) alors que les SVM sont plus précises quand des données labélisées comme normal ou attaque sont disponibles pour apprentissage. Plusieurs directions sont intéressantes et importantes pour le futur :

- Dans le système de monitorage : d'autres filtres et algorithmes de sélection peuvent être considérés pour redéfinir et ordonner notre ensemble des attributs. Les régles de corrélation et de filtrage des événements doivent être étudiées profondément dans le but de détecter les attaques et à un niveau plus haut révéler la stratégie des attaquants et d'améliorer la réponse de prévention.
- Les techniques d'apprentissage non supervisé sont attirantes parce qu'elles ne demandent pas une connaissance à priori du trafic et qu'elles peuvent détecter des attaques nouvelles et inconnues auparavant. Ces techniques devront être étudiées dans le contexte du monitorage VoIP.
- Un pot de miel seul a un champ de vision limité et ne peut détecter qu'un ensemble limité d'attaques. Un réseau de pots de miel (honeynet) muni d'une grande interactivité peut attirer beaucoup plus d'attaques dans un environnement sécurisé et supervisé en offrant un ensemble riche de services. Un honeynet VoIP devra être conçu et implanté pour émuler tout un réseau VoIP (utilisateurs, infrastructure et services).
- Finalement, notre IDS peut être amélioré en utilisant les mêmes types de règles mais avec une structure événementielle améliorée. Un langage standard pour la description des signatures des attaques va faciliter l'incorporation de ces signatures dans les pare-feu et IDS dédiés-VoIP.

Ces travaux ont fait l'objet de plusieurs publications [46–49] et ont été effectués dans le cadre de la thèse de Mohamed Nassar.

6 Audit de sécurité pour les systèmes VoIP

Nous avons abordé l'audit de sécurité d'un système VoIP en proposant un cadre d'audit de la sécurité VoIP ainsi qu'une méthode pour détecter les vulnérabilités logicielles dans les implémentations SIP. L'intérêt pour ces travaux est venu d'un besoin réel et actuel. Comment peut-on évaluer le niveau de sécurité pour le cas d'un réseau VoIP ? Nous avons proposé une architecture qui permet 1) de modéliser les services et ressources VoIP dans un seul modèle de gestion, 2) d'instancier ce modèle de manière autonome et 3) de représenter de façon unifiée les tests d'audit ainsi que les objets/ressources/services y prennant part. Sur un plan complémentaire, nous avons abordé la sécurité du plan de signalisation SIP en proposant des techniques de détection de failles dans la couche logicielle SIP. Ces travaux sont relevants pour la sécurité VoIP car tout équipement VoIP (physique ou logique) intègre une partie logicielle spécifique au traitement du protocole de signalisation SIP. Cette partie logicielle est responsable du traitement de la signalisation SIP par l'implémentation de la couche protocolaire SIP et d'une machine d'états SIP associée. Nous avons abordé la détection automatique des vulnérabilités en proposant une approche de type boîte noire. Même si en français on ne retrouve pas encore l'équivalent du mot anglais fuzzing, il s'agit de tests d'injection de données erronées capables d'identifier des failles de sécurité. Nous avons considéré le cas du type boîte noire pour deux raisons différentes. Dans la pratique, pour la plupart de terminaux VoIP, l'accès au code source est rarement disponible et en plus l'instrumentation des équipements embarqués relève des défis considérables liés au outils de développement qui sont propriétaires et non interopérables. Le cas concret de SIP est particulièrement intéressant pour ceci. Le protocole (SIP) se compose de séries de messages, ayant des structures riches et demandant une gestion d'une machine d'états protocolaires assez complexe. De plus, du point de vue complexité, l'élaboration de tests en absence d'une information additionnelle (obtenue par exemple via l'accès au code source où l'instrumentation du système) est beaucoup plus difficile à mettre en œuvre.

Le test de vulnérabilités appelé « Fuzzing » est un élément important dans le processus d'évaluation de la sécurité d'un système. Le principe est très simple ; il consiste dans l'injection de données aléatoires dans toutes les interfaces (surface d'attaque) du système testé. La difficulté réside (1) dans la génération de données qui vont permettre de révéler une vulnérabilité dans un équipement cible et (2) dans la construction d'architectures qui vont pouvoir démontrer de telles fonctions. Le fuzzing vient se placer en complément d'autres approches d'évaluation de la sécurité des logiciels tels que l'audit de code, le reverse engineering, le test actif, la mesure de risques par modélisation d'attaques (graphes, arbres, réseaux d'attaques) et le déploiement de mécanismes de contrôle d'accès et d'inspection de trafic. Nous avons proposé une nouvelle approche de fuzzing capable, contrairement aux approches existantes sur le marché, d'identifier des failles dans les états avancés d'une interaction protocolaire. Pour cela nous avons développé de nouveaux algorithmes de génération et de suivi de fuzzing couplant apprentissage de protocoles et mutation de données. Nous avons implanté ces algorithmes dans une architecture de fuzzing modulaire et l'avons mise en œuvre sur de multiples équipements. L'approche, l'architecture et les résultats de sa mise en oeuvre sont décrits dans cette section.

6.1 Architecture de fuzzing

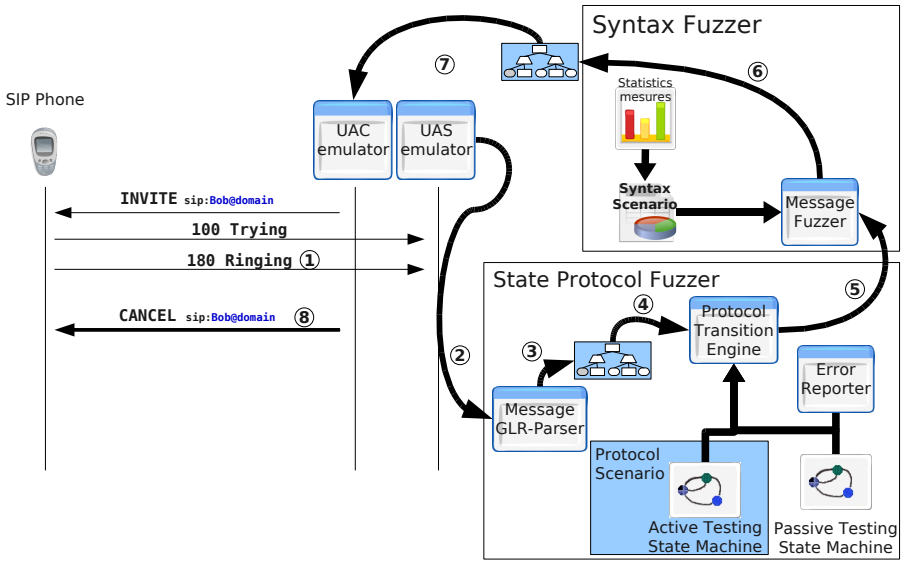


FIG. 12 – KiF framework

L'architecture de fuzzing que nous avons conçue comporte trois grands blocs fonctionnels. Le premier assure l'échange des messages avec la cible en servant d'interface au point d'accès de service. Le second bloc a en charge les états protocolaires (fuzzer protocolaire) et le troisième gère les données des messages (fuzzer syntaxique). Leurs interactions sont illustrées dans la figure 12.

Le fuzzer syntaxique a pour objectif unique de générer des messages individuels d'attaque. Il s'appuie pour cela sur la grammaire de ces messages exprimés à l'aide de la méta-syntaxe ABNF (Augmented Backus Naur Form) ainsi que sur un scénario de fuzzing. Ce scénario pilote la génération des règles de production dans la grammaire de la syntaxe. Il peut également dépendre du fuzzer protocolaire afin de générer le message final qui sera envoyé à l'entité cible.

Le fuzzer protocolaire effectue du test passif et actif. Pour cela, deux automates sont recquis. Le premier spécifie la machine à états SIP ; l'autre spécifie la machine à états de l'activité de test. La première machine est utilisée dans le test passif. Elle contrôle l'occurrence d'un comportement anormal issu de la cible durant la phase de test. Cet automate peut être inféré d'un ensemble de traces SIP relatives à la cible collectées durant des phases opérationnelles normales. Le second automate est utilisé pour du test actif ; il pilote le profil du test de sécurité. Cet automate est défini par l'utilisateur et peut évoluer dans le temps.

Notre algorithme de fuzzing syntaxique a pour objectif de générer un message fuzzé à destination d'une cible. Pour cela, il prend deux paramètres : la grammaire du protocole et un scénario de fuzzing de syntaxe. Ce dernier comprend les règles à appliquer (nous l'appelons l'évaluateur de fuzzing dans la thèse). Ces concepts sont formalisés au sein d'une grammaire d'expression de fuzzer.

La fonction de fuzzing protocolaire est spécifique à un protocole donné car elle requiert

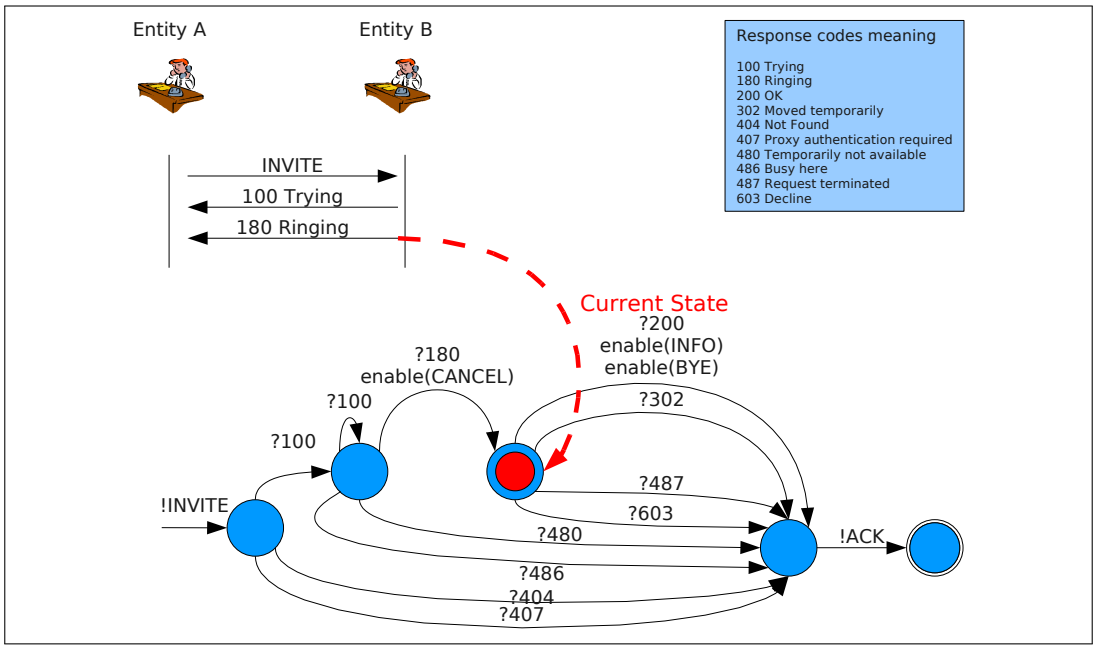


FIG. 13 – Automate de fonctionnement appris à partir d'un message INVITE
Les transitions marquées (?) indiquent que les messages correspondent à l'entité qui répond. Le marquage (!) indique que les messages proviennent de l'entité cliente.

une connaissance du comportement normal d'un protocole afin de pouvoir déterminer des états non-consistants d'une implémentation de celui-ci. Dans ce but, notre approche s'appuie à la fois sur du test actif et passif. Le test passif consiste à surveiller l'intégralité du trafic entre l'attaquant (le fuzzer) et la cible et de le comparer au comportement normal de l'automate d'interaction du protocole. Dans notre approche, cet automate est construit automatiquement par notre approche à partir de l'observation de traces de la cible en conditions d'utilisation normales (sans attaques). Ces observations nous permettent de construire les automates de base que constituent les transactions dans SIP comme illustré dans la figure 13.

Les transitions marquées (?) indiquent que les messages correspondent à l'entité qui répond. Le marquage (!) indique que les messages proviennent de l'entité cliente.

Une alternative à ce mécanisme aurait été de construire l'automate à partir des spécifications du protocole dans les documents normatifs. Ceci n'aurait cependant pas permis de construire les automates adaptés aux équipements cibles car d'une part la norme est très (trop) permissive et d'autre part, nombre d'équipements ne la respectent pas.

Par opposition au test passif qui vise à identifier les états de failles, le test actif consiste à guider l'attaquant ou le fuzzer dans ses échanges avec la cible. Concrètement, le test actif se traduit par des scénarios qui peuvent être soit implémentés à la main par un testeur, soit générés automatiquement. Dans l'environnement actuel, nous utilisons le modèle d'automates finis étendus pilotés par des événements (EEFSM) défini par Lee et al. [44]. Un test actif est centré sur un dialogue et peut intégrer plusieurs transactions. Tout nœud de l'automate comprend les éléments suivants :

1. la nature, le type et la direction du message attendu,
2. la pré-condition à satisfaire dans l'environnement de l'automate,
3. la fonction à appliquer sur la transition (scénario de traitement de la syntaxe),
4. des contraintes temporelles,
5. un poids permettant la sélection de transitions concurrentes.

6.1.1 Tests et évaluations

L'application de notre approche à un grand nombre d'équipements voix sur IP a permis de découvrir de multiples vulnérabilités. A ce jour, aucun équipement testé ne s'est montré infaillible. Dans cette section, nous synthétisons les grandes familles de vulnérabilités identifiées.

Faiblesses dans la validation des entrées La vulnérabilité que nous avons rencontrée le plus fréquemment est liée à un filtrage extrêmement faible (voir inexistant) des données fournies en entrée d'entités voix sur IP via le canal SIP. Ce filtrage, lorsqu'il existe, ne traite pas proprement les méta caractères, les caractères spéciaux, les données de grande longueur ou les caractères spécifiques de formatage. Les failles qui en résultent sont dues à des débordements de tampon/tas ou des vulnérabilités de type "format string". La cause la plus probable à cela est que les développeurs de ces systèmes sont partis d'un modèle de menaces dans lequel la signalisation SIP est supposée générée par des piles protocolaires saines et éprouvées. Une raison plus simple encore peut être l'absence dans les processus de conception de certains de ces équipements de toute ou partie de la dimension sécurité. Le véritable danger de cette vulnérabilité provient du fait que dans la grande majorité des cas, un très faible nombre de paquets peut paralyser un réseau VoIP complet. Ceci est d'autant plus dangereux que dans le cas présent, les messages SIP sont transportés sur UDP, ouvrant la porte à des attaques efficaces effectuées de façon furtive par des techniques simples de spoofing IP. Nous mettons en exergue deux cas extrêmes de vulnérabilités découvertes par notre approche : la première vulnérabilité (publiée dans CVE-2007-4753) révèle que dans le cas étudié, même le test le plus simple de vérification de l'existence de données en entrée n'est pas effectué. Cette absence de vérification permet des attaques extrêmement simples et efficaces telles que l'envoi d'un paquet vide. Le second cas (CVE-2007-1561) est situé à l'extrême du premier sur l'échelle de la complexité. Ici un serveur VoIP est vulnérable à une attaque dont la structure de données d'entrée est relativement complexe. Le danger repose dans ce cas sur le fait qu'un unique paquet va détruire le serveur voix sur IP de coeur et ainsi rendre indisponible l'ensemble du service VoIP associé. Se prémunir de telles attaques à un niveau de défense réseau est possible via des techniques d'inspection profonde de paquets couplées à des équipements de filtrage de paquets spécifiques au domaine.

Vulnérabilités de suivi protocolaire Les vulnérabilités de suivi protocolaire vont au delà du simple filtrage d'un unique message SIP. Dans ce type de vulnérabilités, plusieurs messages vont amener un équipement cible dans un état inconsistant ; tout message utilisé dans cette chaîne d'attaque considéré en isolation ne violera pas la spécification normative du protocole SIP. Ces vulnérabilités proviennent en grande majorité d'une faiblesse dans

l'implémentation des automates du protocole. Elles peuvent être exploitées de trois façons différentes :

1. L'équipement peut recevoir des entrées qui ne sont pas attendues dans l'état courant du protocole : par exemple en envoyant au système un BYE alors qu'il s'attend à recevoir un INVITE,
2. L'entrée peut prendre la forme de messages simultanés dirigés vers plusieurs états du protocole,
3. De faibles variations dans les champs de suivi de dialogues et/ou transaction SIP peuvent amener un équipement vers un état inconsistant.

La découverte de telles vulnérabilités est un problème difficile. Le processus de fuzzing doit ici être capable d'identifier ou et à quel moment un équipement cible ne suit pas rigoureusement le protocole et quels champs des messages peuvent être »fuzzés » pour révéler la vulnérabilité. L'espace de recherche est dans ce cas gigantesque, couvrant de multiples messages et champs de données ; l'utilisation de techniques de fuzzing avancées pilotées par des méthodes d'apprentissage est ici indispensable. Comme pour le cas précédent (vulnérabilités liées au filtrage des données), les vulnérabilités révélées par l'application de notre méthode sont de complexité variable.

Un cas simple est celui de la (CVE-2007-6371). Ici, l'envoi prématuré d'un message CANCEL peut amener l'équipement dans un état inconsistant qui aboutit à un déni de service. Le danger majeur de ce type d'attaques est qu'à ce jour, aucun pare-feu applicatif ne peut suivre et inspecter un si grand nombre de flux et que même dans le cas ou les signatures sont connues, des versions polymorphiques d'attaques efficaces peuvent aisément être obtenues et ainsi passer entre les mailles des systèmes de protection. A ce jour malheureusement, aucune solution efficace pour la prévention de ce type d'attaque n'existe.

Nous avons également identifié un nombre conséquent de vulnérabilités liées à des faiblesses dans les implémentations. Ces faiblesses concernent des implémentations cryptographiques faibles (CVE-2007-5468 et CVE-2007-5469), des supports d'injection SQL et/ou Javascript permettant de la fraude à la facturation et la capacité de certains téléphones à permettre des écoutes distantes sans aucune action du destinataire (CVE-2007-4498).

Vulnérabilités dans la spécification du protocole Nous avons consacré une part importante de notre activité à la recherche de vulnérabilités sur des implémentations spécifiques du protocole SIP sans initialement considérer la sécurité du protocole en soi. C'est lors de l'exécution d'un scénario de fuzzing complexe qui nous avons relevé la même anomalie (et vulnérabilité apparente) sur tous les équipements sous test. Ceci nous a naturellement conduit a lancer une analyse sur la spécification du protocole SIP, notamment en utilisant des techniques formelles et outils supports tels AVISPA [1]. Cette analyse nous a permis d'identifier la vulnérabilité dans la conception même du protocole, vulnérabilité qui rend toute attaque d'escroquerie à la facturation possible sur tout réseau voix sur IP.

Le problème vient du fait qu'une attaque classique de type relais est possible en forçant une entité appelée à émettre un message de type RE-INVITE. Cette attaque étant nouvelle, générique et sévère, elle est naturellement dangereuse. Voici comment

elle se matérialise : un attaquant établit un appel avec sa victime. Sa victime répond (décroche) et est amenée à mettre l'appelant en attente (il existe plusieurs méthodes pour la conduire à entreprendre cette action, la plus simple étant qu'un complice appelle la victime alors que celle-ci est en communication avec l'attaquant). Lorsque l'attaquant reçoit le message SIP re-invite qui spécifie la mise en attente, celui-ci peut demander à la victime de s'authentifier. Cette dernière authentification peut être utilisée par l'attaquant pour se substituer à la victime sur son propre proxy et passer ainsi des appels frauduleux à l'insu de la victime.

Ce travail a été effectué dans le cadre de la thèse de Mr. Humberto Abdelnur - les publications relevantes issues sont [7, 9, 11, 12].

6.2 Synthèse

Il existe aujourd'hui un grand nombre d'outils de fuzzing sur le marché. Tous génèrent de façon plus ou moins intelligente les données à injecter dans la cible afin de la perturber. Notre contribution sur ce domaine est un nouveau modèle de fuzzing qui peut aller tester une cible dans des états protocolaires avancés, ce qu'aucun autre fuzzer ne fait à ce jour.

Nous avons implanté notre méthode de fuzzing et l'avons instanciée sur SIP. Les résultats sont très encourageants avec un ensemble important de vulnérabilités identifiées sur tous les équipements testés. L'outil KiF qui réalise l'architecture de fuzzing est distribué sous License Open Source.

7 Conclusions et perspectives

Nous avons résumé dans ce document nos travaux de recherche portant sur l'audit de sécurité, le fingerprinting et le fuzzing. Nous avons abordé un composant essentiel dans toute approche d'audit -le fingerprinting - la prise d'empreintes des systèmes par l'analyse du trafic généré. Nous avons proposé trois méthodes différentes pour ceci. La première méthode se base sur l'extraction des caractéristiques complexes des sous-arbres de parsing. Les résultats obtenus sont encourageants tant du point de vue des performance e et de son applicabilité. Nous avons également abordé le cas d'un protocole dont nous ignorons la spécification formelle. Pour ce cas, nous sommes capable d'identifier les catégories de messages et construire une machine à état générique. Notre approche utilise des métriques définies sur les messages qui peuvent grouper des messages ayant des structures similaires. Les machines à états génériques s'avèrent de bons candidats pour identifier le systèmes correspondant. Nous avons proposé une méthode de fingerprinting qui exploite ceci. Elle s'appuie sur des empreintes comportementales. Un système est associé avec plusieurs machines à états. Chaque machine à états est induite à partir de traces réseaux. Une telle machine modélise un sous-comportment observé. Un comportement est identifié par une séquence d'états ainsi que par des informations relatives au temps d'exécution. Nous avons montré que ces informations sont suffisantes pour révéler le système les ayant générées. Notre approche s'appuie sur des machines à support vectoriel ou les fonctions de noyau (kernel) sont définies directement sur ces machines d'états. Nous avons à la suite présenté nos travaux de recherche portant sur la détection d'intrusions dans la VoIP. Nos contributions portent sur deux points. Le premier concerne la

définition architecturale et fonctionnelle d'un pot de miel VoIP. Le deuxième comprend la surveillance de flux de signalisation VoIP. Nous proposons une méthode de surveillance qui extrait un ensemble de caractéristiques (SIP) afin d'identifier si une attaque a eu lieu. Finalement, la dernière partie du manuscrit est dédiée au fuzzing. Nous avons traité la découverte de failles de sécurité en élaborant un fuzzer pour le protocole SIP et ayant participé activement à un processus de "disclosure" étique.

7.1 Vers une théorie du fuzzing

Mes objectifs futurs sont de proposer un cadre formel et des paradigmes auto-adaptatifs pour le fuzzing protocolaire. Développer le cadre formel est une activité de longue durée qui vise à introduire des concepts comme l'assurance, l'efficacité, et de la complexité dans le fuzzing. L'idée fondamentale est qu'étant donné une spécification d'un protocole, le cadre devrait pouvoir donner une complexité théorique algorithmique des limites fuzzing aussi bien que probabilistes sur la qualité des résultats. Dans ce travail, je considérerai également un cadre de modélisation basé sur le traitement des signaux de théorie des jeux de jeux. Deuxièmement, je désire travailler vers des paradigmes de fuzzing auto-adaptatifs capables traiter des protocoles (stateful) complexes et accorder le processus fuzzing par rapport au comportement observé. Le fuzzer KIF développé au cours de dernières années est un résultat très prometteur, prouvant qu'un processus autoprogrammé dynamique - où des actions fuzzing de base se composent sur une feedbak dans les essais fuzzing - est particulièrement intéressant. Les techniques d'apprentissage étant à ce jour très simplistes, plus de travail de recherche est nécessaire afin de développer des algorithmes d'apprentissage. Mon travail principal consistera dans la juxtapose d' algorithmes d'apprentissage en ligne avec les processus de décision optimale pour le fuzzing. Le problème fondamental dans le fuzzing est de trouver une donnée d'entrée dans un espace de dimension énorme. Même si à ce jour, nous ne disposons pas des résultats théoriques sur la complexité du fuzzing, nous estimons que la recherche des entrées qui peuvent identifier des failles de sécurité, reste très dure. L'espace de recherche dans lequel nous nous ne trouvons est de grande dimension et en conséquence, je vais m'investir dans la conception des fondements du fuzzing basés sur la programmation dynamique approximative. L'intérêt de cette approche consiste dans le fait que l'énumération totale de l'espace de recherche n'est plus requise. La connaissance d'un sous-ensemble de l'espace peut être couplée avec un modèle d'interpolation afin de pouvoir trouver la meilleure stratégie de fuzzing.

Si nous considérons une direction secondaire des travaux futurs qui doivent être adressés, le sujet le plus important pour ma recherche est lié à la détection et prévention automatisés en infrastructures de VoIP. A ce jour, les solutions existantes ne sont pas appropriées au trafic de VoIP. Les méthodes de dépistage basées par sur les signatures fonctionnent bien pour la surveillance de niveau de réseau, où des attaques sont effectuées dans la plupart des cas dans une seule interaction. Cependant, dans VoIP, la plupart des vulnérabilités sont liées au mauvais cheminement et suivi du protocole/état. Dans ces scénarios, une attaque est répartie sur une grande quantité de messages, tels que tous doivent être traités et dépistés Les défis dans cette activité sont liés à la conception du langage de signature permettant d"éxprimer des vulnérabilités/signatures réparties de messages multiples aussi bien que la mise en place automatisée de firewalls/IDS pour ceci. La dernière activité abordera les aspects algorithmiques liés à la conception du trafic très efficace tel que la

surveillance à grande vitesse intégrée peut être faite.

7.2 Rétro-ingénierie du code malveillant

Mes travaux futurs porteront également la problématique liée au code malveillant - connu sous le nom de "malware". Ce concepte groupe un ensemble de logiciels concernant les outils de spam, phishing, rootkits, botnets ou vers et virus informatique. Nous constatons aussi que depuis peu les cycles de création et déploiement de malwares deviennent de plus en plus efficaces sans que la communauté scientifique puisse trouver une réponse adaptée. Afin de parer à ces attaques, nous sommes obligés de mieux connaître les moyens et méthodes de travail de leurs créateurs. Cette connaissance porte à la fois sur les technologies, outils et failles exploitées, ainsi que sur l'identité et la localisation de l'attaquant. Plusieurs enjeux majeurs y existent. L'identification et la localisation de l'entité de commande d'un grand réseau de type botnet est très difficile. Un réseau de bots est constitué par une armée de machines compromises par un attaquant. Celui ci peut les contrôler à sa volonté afin de commettre toutes sortes d'activitésillégales. Le plan de contrôle est dans la plupart des cas réalisé par une architecture de communication au sein du botnet. L'usage des architectures de type pair à pair, pour la communication dans des réseaux de bots, pose ici de multiples problèmes - la localisation de la source des commandes deviens particulièrement difficile en confrontation avec les réseaux de type "mix-networks" s'appuyant sur le chiffrement de données. La seule solution reste l'analyse dynamique et statique de code malveillant. Dans ce but, je compte aborder la rétro-ingénierie du code malveillant et plus particulièrement celle du code blindé/protégé contre l'analyse dynamique. Je compte développer une méthode globale d'analyse de code malveillant fondée sur trois composants principaux. Le premier composant est l'apprentissage semi-structuré (semi-structured). Cette technique d'apprentissage considère que sur un grand ensemble de données, seulement une quantité infime est annotée. Le reste le sera d'une manière automatique suite à un processus itératif. Pour le code malveillant, cette-ci permettra de classifier les sessions/applications par rapport à leur distance à des sessions/applications déjà annotées. Le deuxième composant est la conception d'un modèle d'information permettant de représenter une application. Cette représentation devra servir de support au processus d'annotation et permettra de capturer les éléments essentiels qui caractérisent une application. Il s'agit de capturer les informations sur les appels système, sur l'accès aux ressources physiques et sur l'utilisation du système des fichiers. Finalement, le troisième composant consiste dans la définition des métriques pour comparer deux applications. Ces métriques s'appuient sur le modèle d'information afin d'identifier les applications similaires en terme du comportement. J'envisage la conception d'une nouvelle famille de fonctions de type "kernel" afin de pouvoir bénéficier du cadre offert par les machines à support à vecteurs.

7.3 Adaptabilité et sécurité

Une troisième direction de recherche que j'envisage porte sur les approches adaptatives de sécurité. L'adaptabilité devient incontournable pour construire de nouvelles approches permettant de mieux émuler de multiples comportements défensifs afin de parer à des outils d'attaques de plus en plus performants. Les grandes questions de recherche se

posent sur la modélisation et l'analyse de ces approches adaptatives. Si nous prenons par exemple le cas des pots de miel, nous constatons qu'à ce jour deux grandes familles y existent. Un pot de miel peut avoir une interaction réduite avec l'attaquant (il s'agit de low interaction honeypots) et révéler que les attaquants peu compétents ou les outils d'attaque automatisés. Ce type de pot de miel émule un service vulnérable sans que l'attaquant aie ma maîtrise totale du système compromis. Il y a également des pots de miel à haute interaction qui donnent une liberté totale aux attaquants. Ce genre de pot de miel pose des enjeux majeurs de sécurité et de légalité. Nous sommes convaincus qu'une troisième voie permettant de réagir avec une grande variété de stratégies devrait exister. Dans notre vision, un pot de miel devrait utiliser de multiples comportements afin d'obtenir le plus d'informations sur la source d'une attaque. Par exemple, un tel pot de miel pourrait accepter la commande du téléchargement issue par un attaquant. Par contre, cette commande pourrait être volontairement finie avec un message d'erreur - et dans ce cas mener l'attaquant à essayer une autre location. Le résultat pour le pot de miel sera dans ce cas la connaissance d'au moins deux locations utilisées par l'attaquant. L'injection de fautes n'est pas limitée au simple téléchargement de code. On pourrait évaluer le niveau technique d'un attaquant par l'injection de fautes dans sa session. Comment définir les stratégies et comment évaluer leur impact est un des problèmes que j'aborderai. Je suis intéressé dans une approche fondée sur la théorie des jeux pour concevoir un cadre d'analyse. Dans ce contexte, je compte définir des types de jeux reflètent la réalité observée sur le terrain. Dans le cadre de ces travaux, je vais développer les modèles de stratégies défensives et offensives. Je vais considérer les jeux répétitifs ou plusieurs interactions successives prennent en compte l'apprentissage qui a lieu autant de coté de l'attaquant ainsi que de celui du mécanisme de défense. Les résultats attendus sont multiples. Je suis convaincu que la théorie des jeux pourra donner une vue complémentaire sur les interactions entre les attaquants et les mécanismes de défense. Les points d'équilibre de Nash pourront identifier les meilleures stratégies pour ces jeux et servir ainsi comme support à la configuration du mécanisme de sécurité. Nous avons déjà proposé [53] et déployé un pot de miel qui gère son comportement par ces concepts. Le pot de miel exécute partiellement les commandes d'un intrus. Nous avons démontré que de cette façon, l'information obtenue sur l'attaquant/intrus est plus significativement plus riche en termes de collecte d'outils et de localisations de l'intrus. L'équilibre de Nash est un profil de stratégie qui spécifie des choix de stratégie optimales pour tous les joueurs, en considérant qu'aucun des joueurs n'a une motivation pour diverger de l'équilibre de Nash, comme un joueur ne peut pas gagner de plus grands profits en choisissant une autre stratégie quand tous les autres joueurs choisissent les stratégies données par le profil. Pour calculer l'équilibre de Nash nous avons besoin N comme série de joueurs, A_i une série de stratégies, et R_i comme fonction de gain.

- N : série de n joueurs
- A_i : série de stratégie ($a_i \epsilon A_i$)
- R_i : fonction de gain $A \rightarrow \mathbb{R}$, où $A = A_1 \times ... \times A_n$

L'équilibre de Nash peut avoir une stratégie pure, qui donne une définition complète de la façon dont un joueur va jouer le jeu, ou une stratégie mixte, qui est une sélection sur un ensemble de stratégies pures. Une stratégie mixte prévue pour le joueur i est l'ensemble des distributions de probabilités sur l'action A_i décrite par l'opérateur simplex $\triangle$.

$$\triangle(\mathrm{A}_i) = \{\ \mathrm{q}_i : \mathrm{A}_i \rightarrow [0,1] \mid \sum_{i=0} q_i(a_i) = 1\}$$

pour des stratégies mixtes :

$$\mathrm{Q} = \prod_i \triangle(A_i) \qquad et \qquad q = (q_i, q_{-i}) \epsilon Q$$

Le gain espéré pour joueur i avec le profil de stratégie q et des stratégies mixtes est :

$$\mathbb{E}_{a \sim q}[R_i(a)] = \sum_{a \epsilon A} q(a) R_i(a)$$
$$\text{où} \quad q(a) = \prod_{j=1}^{N} q_j(a_j)$$

La signification essentielle d'une stratégie mixte est que les actions sélectionées peuvent réaliser un meilleur profit moyen, et l'équilibre dans des stratégies mixtes est associé à la distribution de probabilité sur l'ensemble d'actions. Dans une stratégie mixte, des actions sont effectuées aléatoirement selon la fonction de répartition de probabilité.

Dans notre contexte, un jeu peut être formulé en termes de niveau de sécurité comme bénéfice. Nous supposons que l'attaquant et le défenseur sont rationnels, afin de maximiser leurs gains. L'attaquant et le défenseur ont des objectifs opposés : le défenseur s'efforce de maintenir son infrastructure, tandis que l'attaquant est motivé pour créer des dommages sur le réseau. L'interaction entre un attaquant et un défenseur peut être modélisée comme un jeu stratégique.

Dans cette optique, je souhaite formuler le problème de la configuration d'un mécanisme de sécurité en termes de jeux et d'équilibre Nash. Une configuration peut être assimilée à une stratégie dans la théorie des jeux. Cette configuration peut être vue à plusieurs résolutions - elle couvre à la fois l'ensemble des variables de configuration mais également l'ensemble des politiques de gestion. Une politique de gestion est typiquement retrouvée dans le cadre d'architectures de type "policy based management" et permet la spécification déclarative du comportement de gestion. Les domaines d'applications sont : 1) la sécurité des réseaux VoIP, 2) l'audit de sécurité et 3) les pots de miel avancés. Dans le domaine de la sécurité VoIP, je compte travailler sur la communication P2P. Dans le domaine de l'audit de la sécurité, je compte aborder la conception d'un plan d'audit basé sur la théorie des jeux. Celle-ci servira à évaluer le niveau de sécurité ainsi que les mesures de défense déployées. Finalement, je vais concevoir des nouvelles architectures de pots de miel qui s'appuient sur cette théorie afin de guider leur fonctionnement.

.

Références

[1] Avispa project. `http://www.avispa-project.org`.

[2] A dns rr for specifying the location of services (dns srv). `http://www.ietf.org/rfc/rfc2782.txt`.

[3] The e.164 to uniform resource identifiers (uri) dynamic delegation discovery system (ddds) application (enum). `http://www.ietf.org/rfc/rfc3761.txt`.

[4] Common Information Model (CIM). Distributed Management Task Force, Inc., DSP 0004 (Standard), 1999.

[5] Integrated Network Management, IM 2005. 9th IFIP/IEEE International Symposium on Integrated Network Management, 15-19 May 2005, Nice, France. IEEE, 2005.

[6] H. J. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP Authentication. In IAS '08 : Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security, pages 237–242, Washington, USA, 2008. IEEE Computer Society.

[7] H. J. Abdelnur, R. State, I. Chrisment, and C. Popi. Assessing the security of VoIP Services. In The 10th IFIP/IEEE Symposium on Integrated Management (IM 2007), Munich, Germany, May 2007.

[8] H. J. Abdelnur, R. State, and O. Festor. a method for fingerprinting : cas 193. 2007.

[9] H. J. Abdelnur, R. State, and O. Festor. KiF : a stateful SIP fuzzer. In IPTComm '07 : Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications, pages 47–56, New York, USA, 2007. ACM.

[10] H. J. Abdelnur, R. State, and O. Festor. Advanced Network Fingerprinting. In RAID '08 : Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 372–389, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] H. J. Abdelnur, R. State, and O. Festor. Failles et VoIP. In MISC Magazine - Edition française : Multi-System & Internet Security Cookbook. Misc #39, Septembre/Octobre 2008.

[12] H. J. Abdelnur, R. State, and O. Festor. Fuzzing for vulnerabilities in the VoIP space. In EICAR '08 : 17th Annual Conference of the European Institute for Computer Anti-Virus Research, Laval France, 2008.

[13] R. Badonnel, R. State, and O. Festor. Management of mobile ad-hoc networks : evaluating the network behavior. In Integrated Network Management [5], pages 17–30.

[14] R. Badonnel, R. State, and O. Festor. Management of mobile ad hoc networks : information model and probe-based architecture. Int. Journal of Network Management, 15(5) :335–347, 2005.

[15] R. Badonnel, R. State, and O. Festor. Fault monitoring in ad-hoc networks based on information theory. In F. Boavida, T. Plagemann, B. Stiller, C. Westphal, and E. Monteiro, editors, Networking, volume 3976 of Lecture Notes in Computer Science, pages 427–438. Springer, 2006.

[16] R. Badonnel, R. State, and O. Festor. Probabilistic management of ad-hoc networks. In 10th IEEE/IFIP Network Operations and Management Symposium - NOMS 2006. IEEE Computer Society, 2006.

[17] R. Badonnel, R. State, and O. Festor. Probabilistic management of ad-hoc networks. In Hellerstein and Stiller [41], pages 339–350.

[18] R. Badonnel, R. State, and O. Festor. A probabilistic approach for managing mobile ad-hoc networks. IEEE Transactions on Network and Service Management, 4(7), 2007.

[19] R. Badonnel, R. State, and O. Festor. Self-configurable fault monitoring in ad-hoc networks. Ad Hoc Networks, 6(3) :458–473, 2008.

[20] R. Badonnel, R. State, O. Festor, and A. Schaff. A framework for optimizing end-to-end connectivity degree in mobile ad-hoc networks. J. Network Syst. Manage., 13(4) :479–497, 2005.

[21] A. Ben-hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. Journal of Machine Learning Research, 2 :125–137, 2001.

[22] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. cis Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation, February 2004.

[23] D. Buttler. A Short Survey of Document Structure Similarity Algorithms. In The 5th International Conference on Internet Computing, june 2005.

[24] W. Chen, N. Jain, and S. Singh. ANMP : Ad-Hoc Network Management Protocol. IEEE Journal on Selected Areas in Communications (JSAC), 17(8) :1506–1531, Aug. 1999.

[25] T. Clausen and P. Jacquet. Optimized Link State Routing (OLSR) Protocol. http ://www.ietf.org/rfc/rfc3626.txt, Oct. 2003. IETF Request for Comments 3626.

[26] C. Cortes and V. Vapnik. Support-vector networks. Machine Learning, 20(3) :273–297, 1995.

[27] V. Cridlig, H. J. Abdelnur, R. State, and O. Festor. A voip security management architecture. In Hellerstein and Stiller [41].

[28] V. Cridlig, H. J. Abdelnur, R. State, and O. Festor. Xbgp-man : an xml management architecture for bgp. Int. Journal of Network Management, 16(4) :295–309, 2006.

[29] V. Cridlig, O. Festor, and R. State. Role-based access control for xml enabled management gateways. In A. Sahai and F. Wu, editors, DSOM, volume 3278 of Lecture Notes in Computer Science, pages 183–195. Springer, 2004.

[30] V. Cridlig, R. State, and O. Festor. An integrated security framework for xml based management. In Integrated Network Management [5], pages 587–600.

[31] V. Cridlig, R. State, and O. Festor. Role-based access control for xml enabled multi-protocol management gateways. IEEE Transactions on Network and Service Management, 2(1), 2006.

[32] V. Cridlig, R. State, and O. Festor. A model for checking consistency in access control policies for network management. In Integrated Network Management, pages 11–19. IEEE, 2007.

[33] M. Crispin. Internet Message Access Protocol - Version 4rev1. RFC 3501 (Proposed Standard), Mar. 2003. Updated by RFCs 4466, 4469, 4551, 5032, 5182.

[34] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification, 1(1) :7–24, December 1984.

[35] Douglas Comer and John C. Lin. Probing TCP implementations. In USENIX Summer, pages 245–255, 1994.

[36] D. Endler and M. Collier. Hacking Exposed VoIP : Voice Over IP Security Secrets and Solutions. McGraw-Hill Professional Publishing, 2007.

[37] R. Enns. NETCONF Configuration Protocol. Internet Draft, June 2004.

[38] J. Francois, H. J. Abdelnur, R. State, and O. Festor. Behavioral Fingerprinting . In RAID '09 : Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2009.

[39] S. G. Glisic. Advanced Wireless Networks : 4G Technologies. John Wiley and Sons, Hoboken, NJ, USA, 2006.

[40] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management FrameWorks. STD 62, http ://www.ietf.org/rfc/rfc3411.txt, December 2002.

[41] J. L. Hellerstein and B. Stiller, editors. Management of Integrated End-to-End Communications and Services, 10th IEEE/IFIP Network Operations and Management Symposium, NOMS 2006, Vancouver, Canada, April 3-7, 2006. Proceedings. IEEE, 2006.

[42] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), Apr. 2001. Obsoleted by RFC 5321, updated by RFC 5336.

[43] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In SAC '02 : Proceedings of the 2002 ACM symposium on Applied computing, pages 201–208, New York, NY, USA, 2002. ACM.

[44] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, and X. Yin. A Formal Approach for Passive Testing of Protocol Data Portions. In ICNP '02 : Proceedings of the 10th IEEE International Conference on Network Protocols, pages 122–131, Paris, France, 2002. IEEE Computer Society.

[45] T. Mauro. JUNOScript API Guide for JUNOS Release 6.1. Juniper Networks, 1194 North Mathilda Avenue. Sunnyvale, CA 94089, USA, sonia saruba edition, September 2003.

[46] M. Nassar, S. Niccolini, R. State, and T. Ewald. Holistic voip intrusion detection and prevention system. In Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07), pages 1–9, New York, NY, USA, 2007. ACM.

[47] M. Nassar, R. State, and O. Festor. Intrusion detections mechanisms for VoIP applications. In Third annual security workshop (VSW'06). ACM Press, Jun 2006.

[48] M. Nassar, R. State, and O. Festor. VoIP honeypot architecture. In Proc. of 10 th. IEEE/IFIP Symposium on Integrated Management, Jun 2007.

[49] M. Nassar, R. State, and O. Festor. Monitoring SIP traffic using support vector machines. In Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID '08), pages 311–330, London, UK, 2008. Springer-Verlag.

[50] C. Prehofer and C. Bettstetter. Self-organization in Communication Networks : Principles and Design Paradigms. IEEE Communications Magazine, 43 :78–85, July 2005.

[51] G. Shu and D. Lee. Network Protocol System Fingerprinting - A Formal Approach. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pages 1–12, Apr. 2006.

[52] C. Smith. 3G Wireless Networks. Mc Graw Hill Professional, Columbus, OH, USA, 2002.

[53] G. Wagener, R. State, A. Dulaunoy, and T. Engel. Self adaptive high interaction honeypots driven by game theory. In Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009), 2009.

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Advanced Security Monitoring and Assessment: Audit et monitorage de la sécurité

MÉMOIRE

présenté et soutenu publiquement le 07/12/2009

pour l'obtention de l'

Habilitation de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Radu State

Composition du jury

Président :	Claude Godart, Professeur, ESSTIN, Nancy
Rapporteurs:	Ludovic Mé, Professeur, Supélec, Rennes Aiko Pras, Associate Professor, University of Twente Stéphane Ubeda, Professeur, National Institute of Applied Sciences of Lyon
Examinateurs:	Pascal Bouvry, Professeur, Université du Luxembourg Olivier Festor, Directeur de Recherche à l'INRIA Nancy, Grand Est Eric Filiol, Directeur de la Recherche ESIEA Jean-Yves Marion, Professeur, Ecole des Mines, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Acknowledgments

I would like to thank my friend Olivier for all the human and professional support during my years at INRIA, where I enjoyed all the scientific freedom that I wanted, but could also lay back on your advise when needed. Thank you Olivier. I have been privileged to work with brilliant and dedicated Ph.D students: Humberto, Mohamed, Gérard , Jérôme, Rémi and Vincent. It was a pleasure for me to have you around and work with you. We made a rewarding scientific journey together and I do thank you all. I have been hosted over the past two years by two institutions and research groups -Madynes at INRIA-LORIA, France and the SECAN-LABS at the University of Luxembourg in Luxembourg - in both places I found remarkable people and friends to which I remain indebted.

I would like to express my thanks to the distinguished members of the jury. I appreciate very much and feel honored that you accepted to take part in my journey.

Radu

Contents

List of Figures

General Introduction and Research Context

This Habilitation Degree manuscript provides an overview of some of our major research activities performed over the past few years. I have started to perform research in the early 2000 s working on network management and security. At that time moment, I have addressed two topics: Integrated security models for the network management plane and security monitoring/fault detection in ad-hoc networks. Network and service management is addressing five functional domains (FCAPS) which stands for *Fault Management*, *Configuration Management*, *Accounting Management*, *Performance Management*, *Security Management*. My contributions cover three of these functional areas, namely fault, configuration and security.

Figure 1 gives a high level view on my research activities and context. I have considered *Fault Management* for the specific case of ad-hoc networks. This work was done in the context of Remi Badonel's Ph.D research, which I co-advised with Andre Schaff. Ad-hoc and spontaneous networks raise several major challenges for their management. The definition of a management domain is fuzzy: networks can join and leave a domain at their will. Some nodes might cooperate, others might allow only partial management and a third category can be even malicious and provide rogue management data and a non cooperative behavior. The dynamic and intermittent nature of such environments does also lead to difficulties for detecting faults. A node might be just out of reach, but operate properly, albeit the missing reachability could interpreted as a fault. The limited resources (the most notable being battery life) might also lead to the shutdown of the manager. Reliable fail-over and new management mechanisms are thus needed.

I have investigated two main research directions: the first one addressed the dynamic on the fly composition of management domains, where temporal and spatial well connected nodes build the backbone of the management plane. We have also proposed a cooperative distributed monitoring architecture aiming at avoiding malicious nodes and/or faulty ones. The main idea is that everybody is monitoring everybody and periodically a collective data fusion process is responsible to filter out erroneous data. We have also addressed the fault management in ad-hoc networks by differentiating between normal out of reach conditions and real faults. The main contribution consists in a hidden markov chain fault model that characterizes the topological neighborhood relation as well as fault occurrences. We have assessed information theoretical measures for their potential to differentiate between mobility related out-of reach conditions and real faults. We have proposed a piggybacking extension to OLSR (one of the dominant routing algorithms in ad-hoc networks) in order to efficiently implement our approach. The most relevant publications are [28, 30, 31, 33, 34, 29, 32].

I have addressed the *Configuration Management* and *Security Management* from a twofold perspective. In the framework of Vincent Cridlig's Ph.D research (co-advised with Olivier Festor) we addressed the security of the management plane. We have started this research theme because of the current lack of an integrated security model in the management plane. Existing management frameworks come with dedicated security models and each device can support

several frameworks. For instance, SNMP (the traditional network management protocol) has a security architecture which was designed to support access control and data confidentiality. For historical reasons, these mechanisms did not rely on an external well known method, but the creators of SNMP defined their own. Similarly, the command line interface used on most routers had its own security architecture. The problem with different frameworks comes when inconsistencies arise: for instance one person can change the configuration of a device using one protocol, albeit she is not allowed to perform such an operation using other management frameworks. This inconsistency has two main sources: firstly, it is difficult to manually configure two different security frameworks to be coherent, when no common data model exists. Secondly, the different management frameworks might be incompatible. For instance, one such popular framework allows only a strict, and limited (15) security levels, while SNMP for instance has no such limitations. We have proposed a unified architecture, where access control policies can be expressed in a generic language and we did also propose a translation mechanism towards the different management frameworks. We have built a model that can be used to prove the matching of the two translated versions. As a secondary contribution, we have also proposed an algorithm to detect inconsistencies between access control policies expressed in two different management frameworks. We have also addressed the security mechanisms of an emerging new network management protocol (NetConf). We have defined a RBAC access control mechanism for NetConf and assessed its performance. The most relevant publications describing our work in this topic are [59, 60, 55, 56, 58, 57].

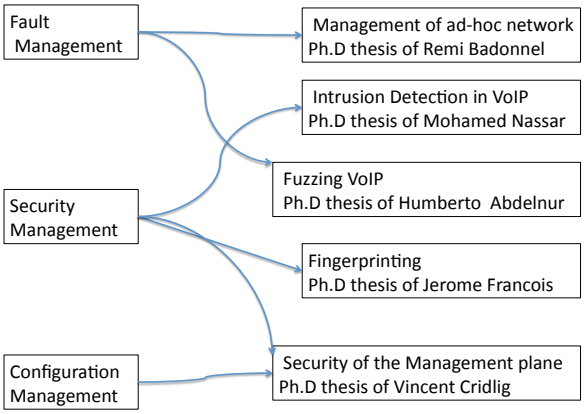


Figure 1: The Overall Picture

In the remaining of this document, I will not cover the previous two research themes, but focus on the more recent activities.

Starting with the year 2005, I have gradually shifted my research interests towards the area of *Fault, Security and Performance Management*.

1 Fault Management

I have co-advised with Olivier Festor, the research work of Humberto Abdenur, where we addressed the automated security testing of VoIP (Voice over IP communications) implementations, concept also known as fuzzing. Software fuzzing emerged as a key approach for discovering vulnerabilities in software implementations. The conceptual idea behind fuzzing is simple: generate

random and malicious input data and inject it in an application. This approach is different from the well established discipline of software testing where functional verification is checked. In fuzzing, this functional testing is marginal; much more relevant is the goal to rapidly find vulnerabilities. Protocol fuzzing is important for two main reasons. Firstly, having an automated approach eases the overall analysis process. Such a process is usually tedious and time consuming, requiring advanced knowledge in software debugging and reverse engineering. Second, there are many cases where no access to the source code/binaries is possible, and where a "black box" type of testing is the only viable solution. Protocol fuzzing can be applied to a broad scope of applications, ranging from device level implementations [45] and up to application layer [123]. We have designed a SIP specific fuzzer, which leverages advanced state tracking mechanisms with efficient input data generation. We did experiment on real VoIP equipments and have followed an ethical disclosure policy.

The VoIP software stacks range from embedded and closed source implementations and up to widely deployed open source software. The challenge on which I have worked addresses the algorithms and approaches for making security testing efficient, such that implementation and/or security design errors can be comprehensively detected in reasonable time frames. The practical outcome of this work consists in identifying vulnerabilities in VoIP equipments (VoIP phone, proxies and VoIP services), notifying vendors and releasing security advisories. This activity is beneficial to the community, since our responsible disclosure policy allows the vendor to fix highly sensible equipment, before malicious users are able to find and exploit such types of vulnerabilities.

The relevant publications are [14, 18, 17, 23, 22, 24, 19].

2 Performance Monitoring

I have co-advised with Olivier Festor, Mohamed Nassar's Ph.D research work on intrusion detection in VoIP.
This work is a logical continuation of some older research activities that I have undertaken in the area of fault and network monitoring. We have looked at the security monitoring of VoIP networks by designing and evaluating specific VoIP intrusion detection and prevention mechanisms. In the current state of deployment, the voice over IP world is facing a large set of threats. SPAM on email systems takes a new and very annoying form on IP telephony advertising. This threat is known as SPIT (Spam over Internet Telephony). Leveraging IP to support voice communications exposes this service (voice) to the known denial of service attacks that can be easily implemented by service or network request flooding on the Internet. Resource exhaustion thus automatically finds its place against SIP proxies and back-to-back user agents, which are essential to support this critical infrastructure. The list of potential threats is huge and ranges from VoIP bots (that can spread by malware and perform distributed attacks, perform SPIT or toll fraud), to eavesdropping and Vishing (an attack similar to Phishing using VoIP as the transport vehicle). Our work did address these threats by proposing a VoIP specific honeypot as well as a SIP-based intrusion detection mechanism. However, the previously mentioned threats are not complete. The major research challenge in the VoIP security monitoring is to provide adequate solutions that can be used to protect a VoIP infrastructure. Within this research activity, we have considered the use of statistical and machine learning techniques for VoIP specific intrusion and attack mitigation approaches. We were the first to propose a VoIP specific honeypot and show its design and usage. In terms of practical outcomes, I expect to see such solutions being offered in existing/future network defense frameworks.

Our relevant publications on this topic are [106, **?**, 107, 105, 104].

3 Security Management

The security assessment of network and service infrastructures depends on the remote identification of a device or service. I have co-advised the research of Humberto Abdelnur and Jérôme François research activities in this domain. We have proposed a network assessment architecture and advanced fingerprinting approaches for this purpose.

Our contributions are threefold. We have developed an approach that extracts syntactical information from protocol elements in order to fingerprint a specific device/stack. We considered next the case of an unknown protocol and have addressed the automated analysis of protocols with respect to the type of exchanged messages. Thus, our approach can be used as an essential preprocessing phase in the automated learning of any protocol related state machine. This is a building block for learning the device/stack specific behavior resulted from reconstructing/reverse engineering a state machine for a device under test. We have integrated time based fingerprinting by considering how elapsed time measured among one or several protocol interactions can disclose the identity of a protocol stack.

The relevant publications in this topic are [21, 76, 17, 14].

3.1 Manuscript organization

The first chapter presents a syntax driven fingerprinting scheme, where parse trees of captured messages were used to learn distinctive features capable to perform fingerprinting. Our assumptions are that specifications are available and that individual messages can be used to infer vendor/stack specific implementation characteristics. This is different from the current approach where no a-priori knowledge of the syntax is assumed.

We have addressed in the same chapter the automated fingerprinting of unknown protocols. Our approach is based on the unsupervised learning of the types of messages that are used by actual implementations of that protocol. The unsupervised learning method relies on support vector clustering - SVC. Our technique is using a new metric - the weighted character position metric. This metric is immune to simple XOR encryptions and does not suppose any knowledge about the protocols: header fields specification, number of messages. One main advantage of our technique is its improvement of the accuracy of the classification for large datasets.

We have also proposed a semi automated method that allows to choose the best parameters. The observed message types can be used to induce a tree-like representation of the underlying state machines. The nodes in this tree represent the different types of observed messages and the edges do indicate an invocation relationship between the nodes.

The second chapter considers the behavioral fingerprinting, based on the analysis of temporal and state machine induced features. We introduce the TR-FSM model, a tree structured parametrized finite state machine having time annotated edges. A TR-FSM represents a fingerprint for a device/stack. Several such fingerprints are associated with a device. We propose a supervised learning method, where support vector machines do use kernel functions defined over the space of TR-FSMs. We validated our approach using SIP as a target protocol.

In the third chapter we present new monitoring approaches for VoIP specific environments. We have developed a monitoring scheme based on Support Vector Machines for efficient flow classification. We continuously monitor a set of 38 features in signaling time slices and use these features as the raw input to the classification engine. A threshold based alarm generator is placed on top of the classification engine. We show that the system is both efficient and accurate and study the impact of the various features on the efficiency.

The fourth chapter of this thesis reflects the practical outcomes of our fuzzing approach. We summarize the fuzzing architecture and give an overview on some of the most surprising

vulnerabilities that have been found with our approach. We present a short positioning of our work with respect to relevant ongoing international activities in the fifth chapter. The final chapter of this manuscript concludes and points out the future research activities to be undertaken.

Chapter 1

Syntactic Fingerprinting of Network Protocols

We present a new approach of grammar-based protocol fingerprinting. The soundness of our approach relies on the obsrvation that the use of syntactic parse trees does reveal enough information for fingerprinting a particular implementation stack. The rationale behind our work consists in the fact that software developers, implementing a protocol stack, will make choices in the implementation of a parsing engine. This work was done in the context of Jerome Francois and Humberto Abdelnur's Ph.D thesis. The relevant publications are [14, 18, 21, 76]

Our contribution automates the fingerprinting process by detecting important and relevant complex tree structures in the parse trees of a given protocol. Features that can serve as fingerprints are identified by paths as well as their associated values in the parse tree. Most known network and application level protocols use a syntax specification based on formal grammars. The essential issue is that each individual message can be represented by a tree like structure. We have observed that stack implementers can be tracked by some specific subtrees and/or collection of subtrees appearing in the parse trees. The key idea is that structural differences between two devices can be detected by comparing the underlying parse trees generated for several messages. A structural signature is given by features that are extracted from these tree structures. Such distinctive features are called fingerprints. We will address in the following the automated identification of them.

If we focus on individual productions (in a grammar rule), the types of signatures might be given by:

- Different **content** for one field. This is in fact a sequence of characters which can determine a signature. (e.g. a prompt or an initialization message).
- Different **lengths** for one field. The grammar allows the production of a repetition of items (e.g. quantity of spaces after a symbol, capabilities supported). In this case, the length of the field is a good signature candidate.
- Different **orders** in one field. This is possible, when no explicit order is specified in a set of items. A typical case is how capabilities are advertised in current protocols.

We derived a learning method to automatically identify the structural signatures which analyzes and compares captured message traces. The overview of the learning and classification process is illustrated in Fig.1.1.

The upper boxes in Fig.1.1 constitute the training period of the system. The output is a set of signatures for each device presented in the training set. The lowest box represents the fingerprinting process. The training is divided in two phases:

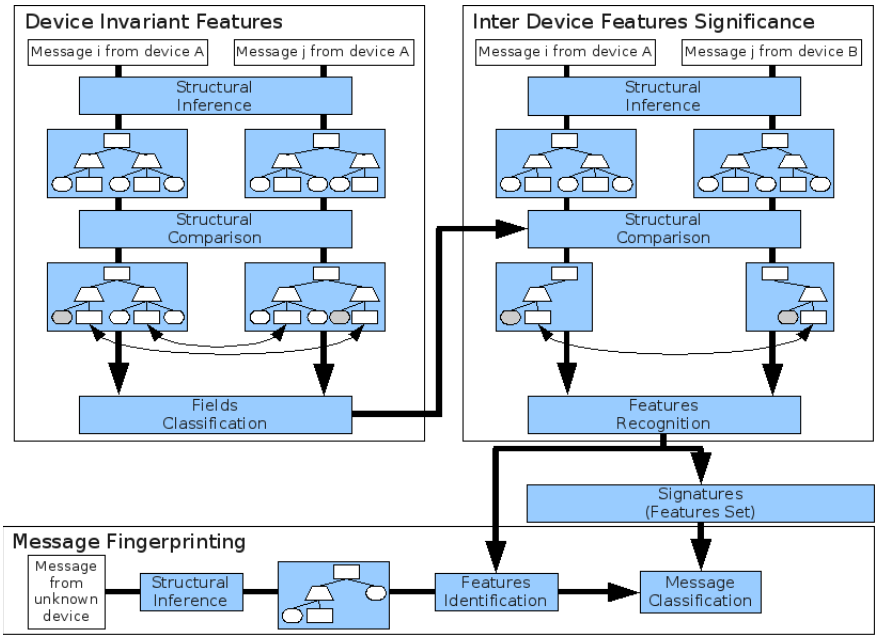


Figure 1.1: Fingerprinting training and classification

Phase 1 (*Device Invariant Features*). In this phase, the system automatically classifies each field in the grammar. This classification is needed to identify which fields may change between messages coming from the same device.

Phase 2 (*Inter Device Features Significance*) identifies among the Invariant fields of each implementation, those having different values for at least two group of devices. These fields will constitute parts of the signatures set.

When one message has to be classified, the values of each invariant field are extracted and compared to the signature values learned in the training phase.

1.1 Structural inference

1.1.1 Formal grammars and protocol fingerprinting

We assume that an Augmented Backus-Naur Form (ABNF) grammar [63] specification is a priori known for a given protocol. Such a specification is made of some simple rules as shown in Fig.1.2.

- A **Terminal** can represent a fixed string or a character to be chosen from a range of legitimate characters.
- A **Non-Terminal** is reduced using some rules to either a Terminal or a Non-Terminal.
- A **Choice** defines an arbitrary selection among several items.
- A **Sequence** involves a fixed number of items, where the order is specified.
- A **Repetition** involves a sequence of one item/group of items, where some additional constraints might be specified.

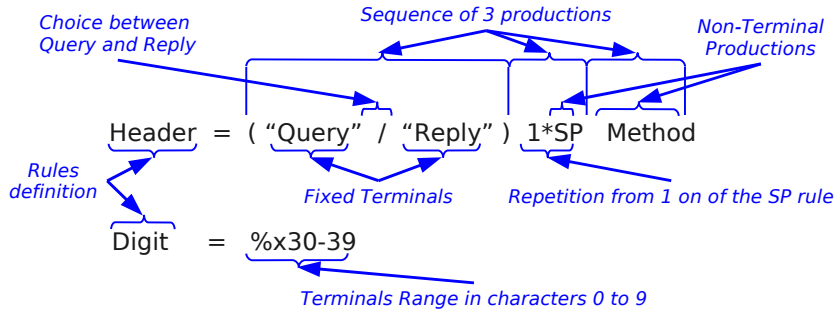


Figure 1.2: Basic elements of a grammar

A given message is parsed according to the fields defined in the grammar. Each element of the grammar is placed in an n-ary tree which obeys the following rules:

- A **Terminal** becomes a leaf node with a name associated (i.e. the terminal that it represents) which is associated to the encountered value in the message.
- A **Non-Terminal** is an internal node associated to a name (i.e. the non-terminal rule) and it has a unique child which can be any of the types defined here (e.g. Terminal, non-Terminal, Sequence or Repetition).
- A **Sequence** is an internal node that has a fixed number of children. This number is in-line with the rules of the syntax specification.
- A **Repetition** is also an internal node, but having a number of children that may vary based on the number of items which appear in the message.
- A **Choice** does not create any node in the tree. However, it just marks the node that has been elected from a choice item.

It is important to note that even if sequences and repetitions do not have a defined name in the grammar rules, an implicit name is assigned to them that uniquely distinguishes each instance of these items at the current rule.

Figure 1.3 shows a Toy ABNF grammar defined in (a), messages from different implementation compliant with the grammar in (b/c) and (d) the inferred structure representing one of the messages in (d).

With respect to the usage, fields can be classified in three categories:

- **Cosmetics Fields**: these fields are mandatory and do not really provide a value added interest for fingerprinting purposes. The associated values do not change in different implementations.
- **Static Fields**: are the fields which values never change in a same implementation. These values do however change between different implementations. Obviously, these are the type of fields which may represent a signature for one implementation.
- **Dynamic Fields**: these fields are the opposite of static fields and do change their values in relation to semantic aspects of the message even in a single implementation.

An additional sub-classification can be defined for dynamic and static fields:

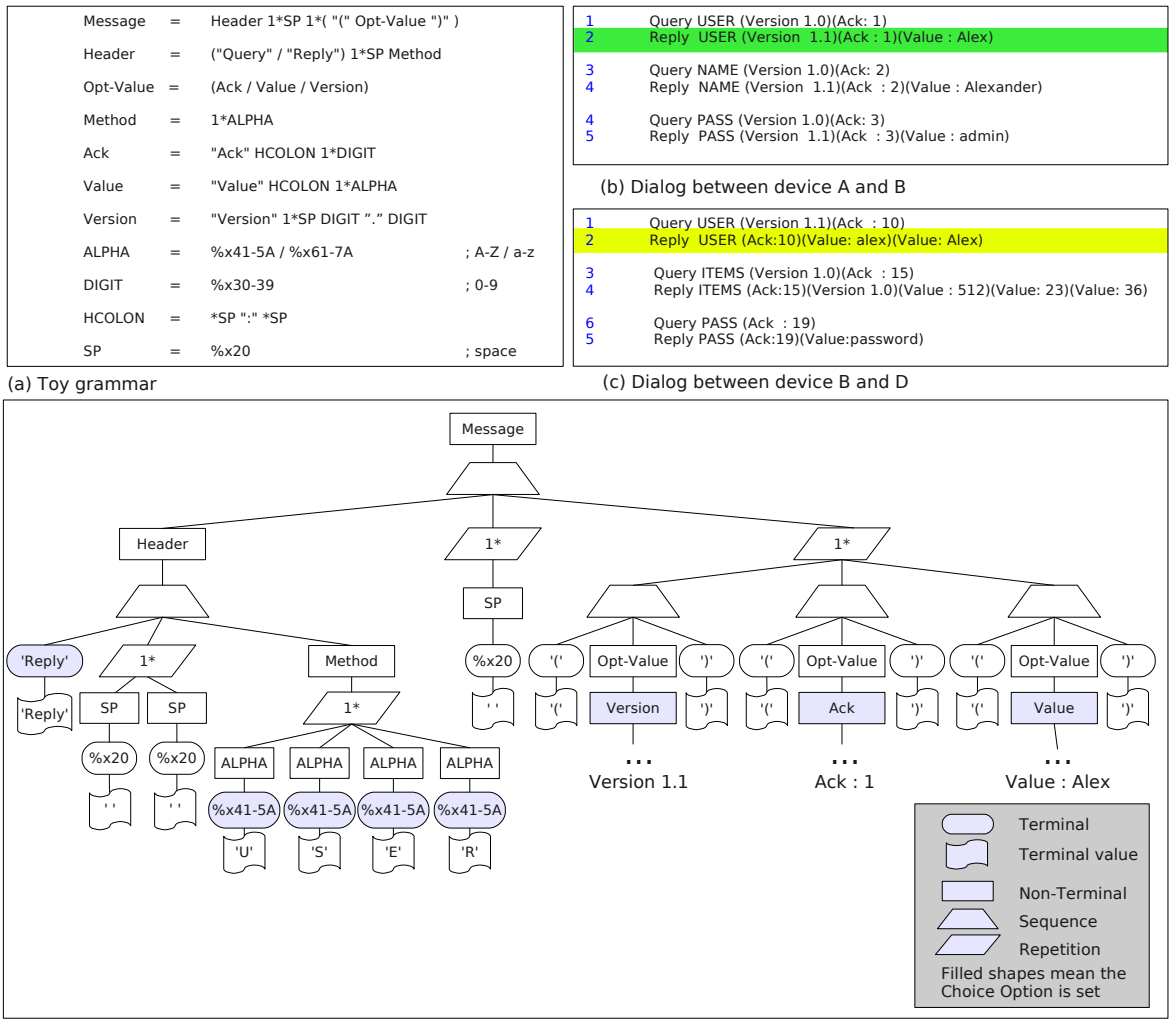


Figure 1.3: Parsed Structure Grammar

- **Value Type** relates to the String reduction of the node (i.e. the text information of that node),
- **Choice Type** relates to the selected choice from the grammar,
- **Length Type** corresponds to the number of items in a Repetition reduction,
- **Order Type** corresponds to the order in which items of a Repetition reduction appear.

Even if one implementation may generate different kind of values for the same field, such values could be related by a function and then serve as a feature. Therefore, a **Function Type** can be also defined to be used to compute the value from a node of the tree and return an output that is useful for the fingerprinting process. Essentially, this type is used for manually tuning the training process.

1.1.2 Node signatures and distance functions

From the parse trees that have been obtained, we can extract subtrees that might identify a specific implementation (or families of related protocol stacks). We will call such subtrees signatures. In principle, good signatures should reflect the following properties:

- As more items are shared between trees, the more similar their signatures must be.
- Nodes that have different tags or ancestors must be considered different.
- In cases where the parent node is a Sequence, the location order in the Sequence should be part of the tree signature.
- If the parent node is a repetition, the location order should not be part of the tree signature, order will be captured later on in the fingerprinting features.

There are several ways, that signatures can be compared. The common approach is that a dissimilarity metric (or distance) should capture the previously mentioned characteristics. Among the most appropriate metrics, the resemblance method [43] uses the elements of the set as tokens. This resemblance is based on shingles, where a shingle is a contiguous sequence of tokens from the document. Between documents D_i and D_j the resemblance is defined as:

$$r(D_i, D_j) = \frac{|S(D_i, w) \bigcap S(D_j, w)|}{|S(D_i, w) \bigcup S(D_j, w)|} \tag{1.1}$$

where $S(D_i, w)$ creates the shingles of length w for the document D_i.

Definition 1. *The **Node Signature** function is defined to be a Multi-Set of all partial paths belonging to the sub-branch of the node.*

The *partial paths* start from a non-root current node rather than from the root of the tree. The structure used is a Multi-Set rather than a Set in order to store the quantity of occurrences for specific nodes in the sub-branch. For instance, the number of spaces after a specific field can determinate a signature in an implementation.

Another dissimilarity measure can leverage tree kernels introduced in [53, 130, 100]. These kernels are based on the tree substructures of the original tree. We consider two kernel types introduced in [99], [101] and [100] : the subtree (ST) kernel and the subset tree kernel (SST). Simply stated a subtree (ST) of a node is just the complete subtree rooted in that node. A subset tree corresponds to a cut in the tree - a subtree rooted at that node that does not include

the original leaves of the tree. Parse trees of a device can be mapped to a a set of ST and SST features by extracting all underlying SSTs and STs. Two parse trees generated by two different devices can now be compared by decomposing each tree in its SSTs and STs followed by a pair-wise comparison of the resulted SSTs and STs. This can be done using tree kernels.

1.1.3 Structural difference identification

Algorithm 1 is used to identify differences between two nodes. These nodes are located in different trees and share the same ancestor path.

Algorithm 1 Node differences Location

```
procedure NODEDIFF(node_a, node_b)
    if Tag(node_a) = Tag(node_b) then
      if Type(node_a) = TERMINAL then
        if Value(node_a) != Value(node_b) then
          Report_Difference('Value', node_a, node_b)
        end if
      else if Type(node_a) = NON-TERMINAL then
        NODEDIFF(node_a.child_0, node_b.child_0))      ▷Non_Terminals have
                                                       ▷an unique child
      else if Type(node_a) = SEQUENCE then
        for i = 1..#node_a do                          ▷In a Sequence
          NODEDIFF(node_a.child_i, node_b.child_i)     ▷#node_a = #node_b
        end for
      else if Type(node_a) = REPETITION then
        if not (#node_a = #node_b) then
          Report_Difference('Length', node_a, node_b)
        end if
        matches := Identify_Children_Matches(node_a, node_b)
        if ∃ (i, j) ∈ matches : i != j then
          Report_Difference('Order', node_a, node_b)
        end if
        forall (i, j) ∈ matches do
          NODEDIFF(node_a.child_i, node_b.child_j)
        end for
      end if
    else
      Report_Difference('Choice', node_a, node_b)
    end if
end procedure
```

where the *Tag*, *Value*, and *Type* functions return the name, value and respectively the type of the current node. Note that $Tag(node_a) = Tag(node_b) \Rightarrow Type(node_a) = Type(node_b)$.

The **Report_Difference** function takes the type of difference to report and the corresponding two nodes. Each time the function is called, it creates one structure that stores the type of difference, the partial path from the root of the tree to the current nodes (which is the same for both nodes) and a corresponding value. For differences of type 'Value' it will store the two terminal values, for 'Choice' the two different Tags names for 'Length' the two lengths and for 'Order' the matches.

The **Identify_Children_Matches** function identifies a match between children of different repetition nodes. The similitude between each child from $node_a$ and $node_b$ (with n and m children respectively) is represented as a matrix, M, of size n x m where:

$$M_{i,j} = dissimilarity(node_a.child_i, node_b.child_j)$$

Assuming a training set Msg_set, of messages compliant with the grammar as

$$Msg = \bigcup_{i=0}^{n} msg\_set_i$$

where n is the quantity of devices and msg_set_i is the set of messages generated by device i, the total number of comparisons computed in this process is

$$cmps_1 = \sum_{i=0}^{n} \frac{|msg\_set_i| * (|msg\_set_i| - 1)}{2} \tag{1.2}$$

Some features are essential for an inter-device classification. Our system recognizes these features as follows:

Algorithm 2 Features Recognition Algorithm

procedure featuresRecognition($fieldClassifications,\ DevID_{a,b},\ differences_{a,b}$)
  forall $diff \in differences_{a,b}$ **do**
    if not $(diff.type, diff.path) \in fieldClassifications$ **then**
      if $diff.type == 'Value'$ **then**
        $addFeature('Value',\ diff.path,\ DevID_{a,b},\ diff.value_{a,b})$
      else if $diff.type == 'Choice'$ **then**
        $addFeature('Choice',\ diff.path,\ DevID_{a,b},\ diff.name_{a,b})$
      else if $diff.type == 'Length'$ **then**
        $addFeature('Length',\ diff.path,\ DevID_{a,b},\ diff.length_{a,b})$
      else if $diff.type == 'Order'$ **then**
        if $(\exists\ (x, z) \in diff.matches : x \neq z)$ **then**
          $addFeature('Order',\ diff.path,\ DevID_{a,b},$
            $diff.match,\ diff.children\_nodes_{a,b})$
        end if
      end if
    end if
  end for
end procedure

The **add_Feature** function stores in a global variable, **recognizedFeatures**, the partial path of the node associated with the type of difference (i.e. Value, Name, Order or Length) and a list of devices with their encountered value. However, the 'Order' feature presents a more complex approach, requiring minor improvements.

Assuming the earlier Msg_set set, this process will do the following number of comparisons:

$$cmps_2 = \sum_{i=0}^{n} |msg\_set_i| * \sum_{j=i+1}^{n} |msg\_set_j| \tag{1.3}$$

From the *recognizedFeatures* only the *Static* fields are used. The recognized features define a sequence of items, where each one represents the field location path in the tree representation and a list of Device ID with their associated value.

The recognized features can be classified in:

- Features that were found with each device and at least two distinct values are observed for a pair of devices,

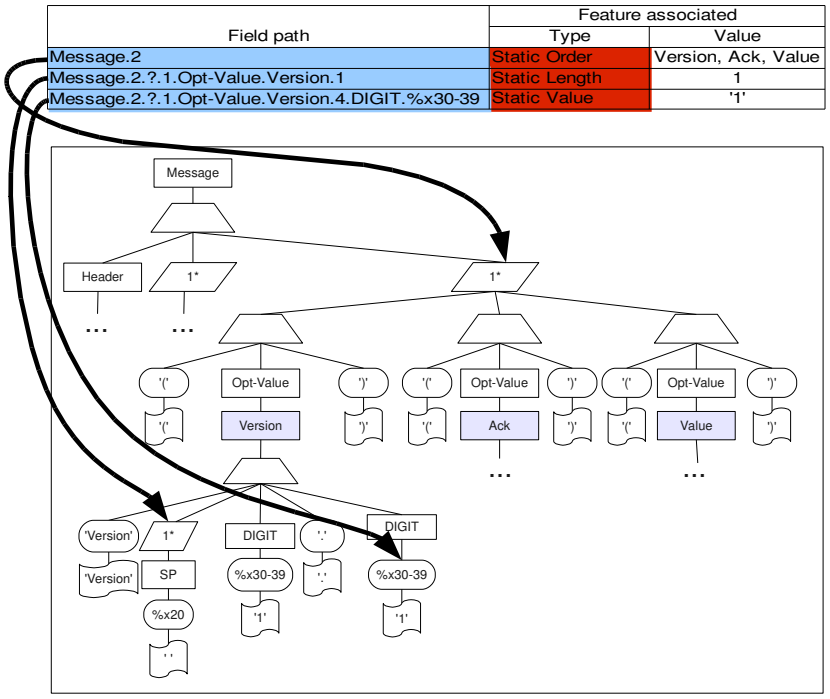


Figure 1.4: Features Identification

- Features that were found in some of the devices for which such a location path does not exists in messages from other implementations.

Figure 1.4 illustrates some identified features for an incoming message.

1.2 Experimental Results

We have implemented the fingerprinting framework approach in Python. A scannerless Generalized Left-to-right Rightmost (GLR) derivation parser has been used (Dparser [3]) in order to solve ambiguities in the definition of the grammar. The training function could easily be parallelized. We have instantiated the fingerprinting approach on the SIP protocol. The SIP messages are sent in clear text (ASCII) and their structure is inspired from HTTP. Several primitives - REGISTER, INVITE, CANCEL, BYE and OPTIONS - allow session management for a voice/multimedia call. Some additional extensions do also exist -INFO, NOTIFY, REFER, PRACK- which allow the support of presence management, customization, vendor extensions etc. We have captured 21827 SIP messages from a real network, summarized in Table 4.1.

The system was trained with only 12% of the 21827 messages. These messages were randomly sampled. However, a proportion between the number of collected messages and the number used for the training was kept; they ranged from 50 to 350 messages per device. We identified 172 features among all the different types of messages. These features represent items order, different lengths and values of fields where non protocol knowledge except its syntax grammar had been used. Between two different devices the distance of different features ranges between 26 to 95 features, where most of the lower values correspond to different versions of the same device. Usually, up to 46 features are identified in one message.

Table 1.2 summarizes the sensitivity, specificity and accuracy. The results were obtained using the test data set.

Device	Software/Firmware version
Asterisk	v1.4.4
Cisco CallManager	v5.1.1
Cisco 7940/7960	vP0S3-08-7-00
	vP0S3-08-8-00
Grandstream Budge Tone-200	v1.1.1.14
Linksys SPA941	v5.1.5
Thomson ST2030	v1.52.1
Thomson ST2020	v2.0.4.22
SJPhone	v1.60.289
	v1.60.320
	v1.65
Twinkle	v0.8.1
	v0.9
Snom	v5.3
Kapanga	v0.98
X-Lite	v3.0
Kphone	v4.2
3CX	v1.0
Express Talk	v2.02
Linphone	v1.5.0
Ekiga	v2.0.3

Table 1.1: Tested equipment

Classification	**True Positive** 18881	**False Positive** 20	**Positive Predictive Value** 0.998
	False Negative 2909	**True Negative** N.A.	**Negative Predictive Value** 0.993
	Sensitivity 0.866	**Specificity** 0.999	**Accuracy** 0.993

Table 1.2: Accuracy results obtained with the system

Type of Message	False Negatives	Message quantity	Miss percentage
200, 100, ACK	1613 (710, 561, 347)	9358 (4663, 1802, 2893)	17% (15%, 31%, 11%)
501, 180, 101 BYE, 486	824 $\begin{pmatrix} 257, & 215, & 148 \\ & 104, & 100 \end{pmatrix}$	3414 $\begin{pmatrix} 385, & 1841, & 148 \\ & 892, & 176 \end{pmatrix}$	24% $\begin{pmatrix} 65\%, & 11\%, & 100\% \\ & 11\%, & 67\% \end{pmatrix}$
489, 487, 603 202, 480, 481 380, 415, 400	213 $\begin{pmatrix} 84, & 57, & 28 \\ 21, & 13, & 6 \\ 2, & 1, & 1 \end{pmatrix}$	636 $\begin{pmatrix} 84, & 230, & 118 \\ 52, & 42, & 18 \\ 2, & 38, & 51 \end{pmatrix}$	33% $\begin{pmatrix} 100\%, & 24\%, & 23\% \\ 40\%, & 30\%, & 33\% \\ 100\%, & 2\%, & 2\% \end{pmatrix}$
INVITE, OPTIONS REGISTER, CANCEL SUBSCRIBE	117 $\begin{pmatrix} 38, & 34 \\ 25, & 19 \\ & 1 \end{pmatrix}$	5694 $\begin{pmatrix} 3037, & 628 \\ 1323, & 297 \\ & 409 \end{pmatrix}$	2% $\begin{pmatrix} 1\%, & 5\% \\ 1\%, & 6\% \\ & .00\% \end{pmatrix}$
INFO, REFER PRACK, NOTIFY PUBLISH	0	2223 $\begin{pmatrix} 1830, 163 \\ 117, 77 \\ 36 \end{pmatrix}$	0%
11 other Response Codes	0	492	0%

Table 1.3: False Negative classification details

We can conclude that the results are very good due to the high specificity and accuracy. Some of the false negatives (about 2/5) belong to only one implementation (percentage that represents 50% of its messages), 2/5 belongs to three more device classes (representing 18% of their messages), the final 1/5 belongs to 8 classes (representing 10% of their messages) and the 7 classes left do not have false negatives. This issue can be a consequence of the irregularity in the quantity from the set of messages in each device, but also to the fact that many SIP messages do not contain valuable pieces of information (e.g. intermediary messages). Table 1.3 shows all the 38 types of messages collected in our test with information concerning their miss-classification (i.e. False Negatives).

1.3 Adressing unknown protocols

The natural evolution of our work addressed the case of devices that speak a common, yet unknown to the fingerprinting engine, protocol. Over the past few years, there has been an increased effort in the research community towards the automated analysis and reverse engineering of network protocols [96, 42, 77]. The driving forces are multiple and range from practical needs to analyze network traffic generated by malware where the most notorious case is the Storm bot up to the development of open source implementation for poorly documented protocols, as it was the case of the SMB protocol [127] for example. A related problem is the automated and passive fingerprinting of devices using an unknown protocol. While some research efforts in this direction have been recently undertaken in [92] in order to learn the syntax and grammar that generated the protocol messages, to our knowledge, none until now has addressed the automated learning of the specific behavior of a protocol in order to fingerprint a device or a protocol stack. [64] and [91] are close and complementary works as they aim to learn an unknown protocol to automatically respond to requests.

We consider an automated approach for differentiating protocol message types using an unsupervised support vector clustering algorithm. Our solution is passive and does not assume active and stimulus triggered behavior templates. We instantiate our solution to the particular case

of a VoIP specific protocol (SIP) and validate it using extensive data sets collected on a large size VoIP testbed. We assume in this case zero knowledge about the syntax and state machine underlying the protocol.

The research challenges that we have faced are related to learning the relevant protocol operations/primitives and modeling the protocol message sequences such that automated learning is possible. If packet captures from an unknown protocol are given, we aim first to automatically discover the unknown types of messages. We assume furthermore that no learning set with labeled protocol messages exists, that no encryption is used and that no reverse engineering of the application using such a protocol is possible. We also assume that the number of different message types is a-priori unknown.

We will start first with presenting a metric that can be use to identify and cluster similar protocol messages.

Weighted character position Most protocol messages are formed by a header containing the type of the message followed by options, arguments and an additional payload. This comes from good and established protocol design patterns. The $weighted\_char\_pos(m)$ metric, is couting for frequent patterns occuring among the first characters:

$$\forall \; character \; c \; occurring \; k \; times, \; p(m)(c) = \frac{\sum_{i=1}^{i=k} pos(a_i)^{-1}}{k} \tag{1.4}$$

The key assumption is that messages of the same types should start with similar headers even if the message contents are totally different. Each message can be thus represented in a 256 dimensional vector space. For each ASCII character, its corresponding coordinate is $p(m)(c)$.

We have proposed two different approaches that leverage the previous metric in order to perform a clustering on captured messages The first technique is relying on the unsupervised support vector clustering [40]. The second method is based on the well known agglomerative nearest neighbor method [66].

1.3.1 Support vector clustering

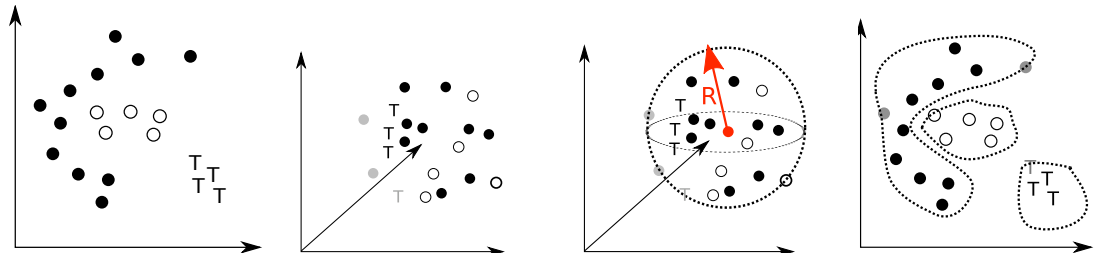
The support vector clustering (SVC) technique has been introduced in [40] and leverages machine learning paradigms based on support vector machines techniques and motivation [54]. Such techniques show good accuracy with a limited overhead in different domain [133]. The initial data points (see figure 1.5) are mapped from the input space to a high dimensional space using a non linear transformation. The goal is to find the smallest hyper-sphere which contains all the points in the high dimensional space. This sphere is mapped back to the original input space and forms a set of enclosings which are considered as the clusters boundaries. The final step determines the cluster of each point by checking which boundaries it is contained in .

Consider Φ, a nonlinear transformation and $\{x_i\}$ the set of N points in the original d-dimensional input space. The first phase consists of finding the smallest hyper-sphere containing all the transformed points $\{\Phi(x_i)\}$ which is characterized by its radius R and its center a. Therefore we have:

$$\|\Phi(x_i) - a\|^2 \leq R^2 \; \forall i \tag{1.5}$$

The original problem is casted into the Lagrangian form by introducing the lagrangian multipliers (β_i and μ_i) and the penalty term ($C \sum_i \xi_i$):

$$L = R^2 - \sum_i (R^2 + \xi_i - \|\Phi(x_i) - a\|^2)\beta_i - \sum_i \xi_i \mu_i + C \sum_i \xi_i \tag{1.6}$$



(a) Initial data points (b) Mapping to high dimensional space (c) Looking for the hyper-sphere (d) Hyper-sphere projection in initial dimensions

Figure 1.5: SVC example

Then, the problem is turned into its Wolfe dual form and the variables a and R are eliminated due to Lagrangian constraints.:

$$W = \sum_i \Phi(x_i)^2 \beta_i - \sum_{i,j} \beta_i \beta_j K(x_i, x_j) \tag{1.7}$$

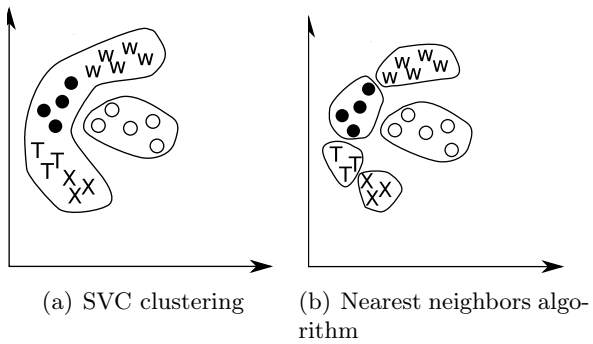
where $K(x_i, x_j)$ is typically defined by a Gaussian Kernel:

$$K(x_i, x_j) = e^{-q\|x_i - x_j\|^2} \tag{1.8}$$

where q is another parameter named Gaussian width.

Next, a labeling step has to determine the points that belong to the same clusters by a geometric approach. In fact, two points are considered being in the same clusters if all the points on the segment between them in the original space are in the hypersphere in the high dimensional feature space. The connected components in the graph represent the different clusters.

1.3.2 Global method



(a) SVC clustering (b) Nearest neighbors algorithm

Figure 1.6: Global Method

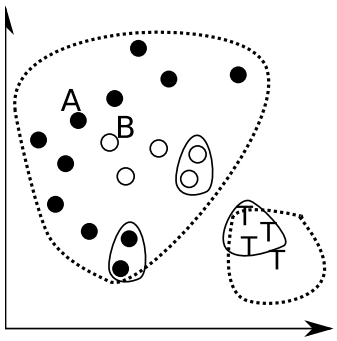


Figure 1.7: Limitation of nearest neighbors clustering

Even if SVC enables the discovery of intertwined clusters, the accuracy can be limited when a single shape comprises different clusters. The figure 1.6 shows such a case, where in figure 6(a), the SVC method is able to isolate two large clusters but none of the single ones which composes the largest one. These constructed clusters can be furthermore split by an additional nearest neighbors technique applied to each of them. Hence, this second step is necessary.

Obviously, it depends also on the data to classify and our experiments in the next sections show the benefits of the combination of these two methods. Furthermore, several multi-pass clustering methods exist and are introduced in [41]. Complex clusters boundaries are discovered by the SVC technique. By applying the nearest neighbors technique, the result shown in figure 6(b) can be obtained. However, applying solely the nearest neighbors technique will entail a bad classification as illustrated in figure 1.7. Therefore, we propose a global method which consists in two steps:

- an initial clustering using SVC,
- a second cluster splitting pass using nearest neighbors technique.

Evaluation metrics

We consider several metrics in order to assess the quality of the clustering method. Consider n messages to be classified, $m_1 \dots m_n$, divided into r types and k clusters found: $c_1 \dots c_k$ with $k \leq n$. At the end of the classification, a label is assigned to each cluster which is the predominant type of the messages within. However, only one cluster per type, the largest one, is allowed. If $c(m_i)$ represents the cluster containing m_i then $t(m_i)$ is the real type of the message m_i and $t(c_i)$ is the type assigned to the cluster (c_i).

The first metric is the classification rate cr and represents the ratio of messages which are classified in the right clusters:

$$cr = \frac{\sum_{i|t(m_i)=t(c(m_i))} 1}{n} \tag{1.9}$$

The second metric is the proportion of different message types which were discovered:

$$cf = \frac{r}{k} \tag{1.10}$$

$$cr_{type}(y) = \frac{\sum_{i|t(m_i)=y} x_i}{\sum_{i|t(m_i)=y} 1} \; where \; x_i = 1 \; if \; t(m_i) = t(c(m_i)) \; else \; 0 \tag{1.11}$$

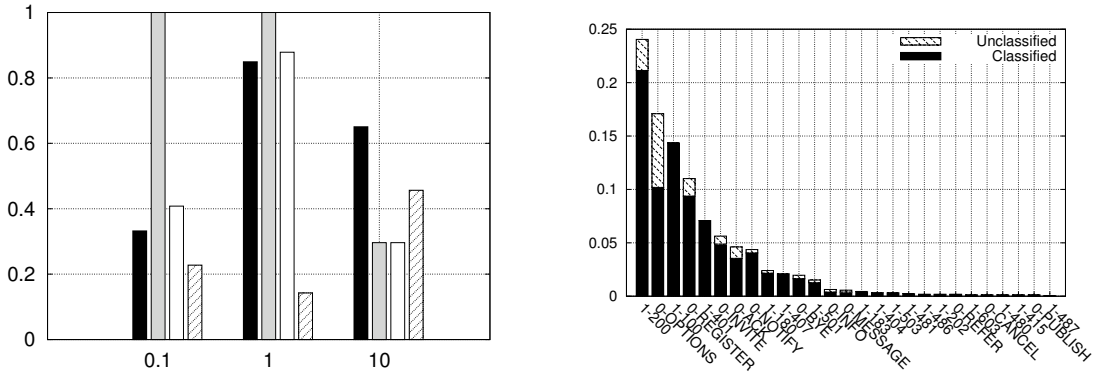
1.4 Experimental analysis

We considered in a first instance the case of SIP. SIP messages are divided into two categories: the requests and the responses. Each request begins with one of the following keywords: `REGISTER, OPTIONS, INVITE,UPDATE, CANCEL, ACK, BYE, SUBSCRIBE, NOTIFY, PRACK, PUBLISH, INFO, REFER, MESSAGE`. The SIP responses begin with a numerical code of 3 digits divided into 6 classes identified by the first digit.

We have built a dataset of 1580 SIP messages, generated using several phones coming from different manufacturers. In our traces we minded 27 different kinds of messages which are the most important. We have selected these types based on our experience with VoIP.

Nearest neighbors technique results

The results from the figure 8(a) are very good and show that it is possible to find all the different kinds of message with a global and per type classification rate close to 85%.



(a) Accuracy

(b) Clusters details with $t = 1$

Figure 1.8: Weighted characters position results

SVC technique results

We applied the SVC method with different values for the Gaussian width q and the penalty factor C. The best possible accuracy is 0.73 for the classified messages with all types of messages found. This result is good but slightly lower than the nearest neighbors technique on figure 1.8 (85% of good classification). This is mainly due to a poor discovery of the smallest clusters because the standard deviation of the specific classification rate per type is higher. When the Gaussian width q increases between 0.1 and 1, the difference between the packets is emphasized in the high dimensional feature space. Hence, the message clusters are more split and the accuracy is improved. However, when q is too high, the number of clusters continues to increase with redundant types. The number of found clusters is then still good but due to many redundant cluster types, the classification rate drops.

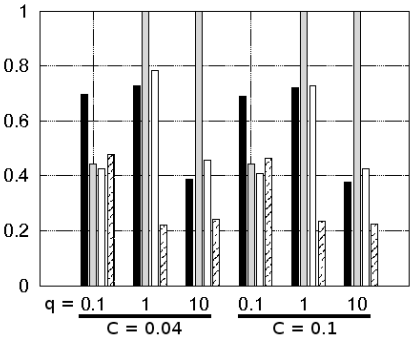


Figure 1.9: Accuracy results

By looking for the best settings, we found $t = 1.1$ which allows to classify 91% of the messages and to discover 96% of the types of the message.

We have applied our method also to other protocols: SMTP [85] (150 packets and 10 types) and IMAP [61] (289 packets and 24 types). To ease the comparison of results, a classical standardization of the data is done. When the nearest neighbors technique is applied, the classification rates are similar for these protocols with less than 50% of the messages are well identified.

The SVC method instantiated with the IMAP does not improve the clustering accuracy since in the best case, only 36% of the messages are well classified. Hence, doing the second step with

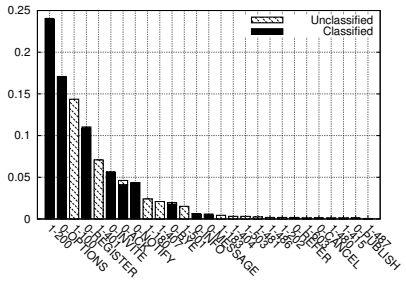


Figure 1.10: Clusters details with $q = 0.1$

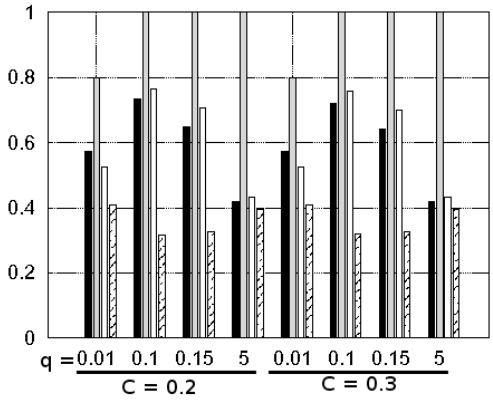


Figure 1.11: SVC - SMTP dataset

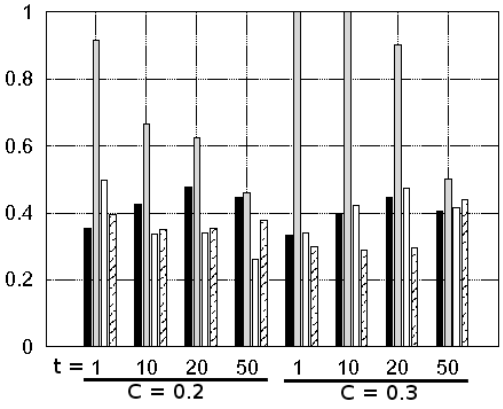


Figure 1.12: Nearest neighbors clustering

nearest neighbors technique is necessary and allows to obtain 49% of good classification. This is slightly better than the nearest neighbors technique which was 47% on the figure 1.12. Obviously, this difference is very low but the number of different types found increases from 62% to 96% with SVC. Therefore, even if the combined method doesn't improve the classification rate, it is able to keep the classification rate stable and at the same time discovering more message types. The figure 1.11 shows the accuracy of SVC for the SMTP traffic. Since the nearest neighbors technique was able to find most of the types on the figure 1.12, SVC can also find them. Moreover, the classification rate is greatly improved: 72% of messages are correctly identified and 80% of kinds are found. By applying the nearest neighbors technique on the obtained clusters, the results are very close because only one additional type is identified with one packet. Hence, the number of kinds is 90% with. The standard deviation of the classification rate per type is quite high (0.39) principally due to one type totally ignored in both cases.

1.5 Conclusion

We have described a novel approach for generating fingerprinting systems based on the structural analysis of protocol messages. The key idea is to use a structural approach, where formal grammars and collected network traffic are used. Features are identified by paths and their associated values in the parse tree. The obtained results of our approach are very good. This is due to the fact that a structural message analysis is performed. Most existing fingerprinting systems

are built manually and require a long lasting development process. Our solution automates the generation by using both formal grammars and collected traffic traces. It detects important and relevant complex tree like structures and leverages them for building fingerprints. The applicability of our solution lies in the field of intrusion detection and security assessment, where precise device/service/stack identification are essential. We have implemented a SIP specific fingerprinting system and evaluated its performance. The obtained results are very encouraging. Future work will consist in improving the method and applying it to other protocols and services. Our work is relevant to the tasks of identifying the precise vendor/device that has generated a captured trace. We do not address the reverse engineering of unknown protocols, but consider that we know the underlying protocol. The current approach does not cope with cryptographically protected traffic. A straightforward extension for this purpose is to assume that access to the original traffic is possible. Our main contribution consists in a novel solution to automatically discover the significant differences in the structure of protocol compliant messages. We plan to extend our work towards the natural evolution, where the underlying grammar is unknown.

We have also partially addressed the automated inference of unknown protocols. Our approach is based on the unsupervised learning of the types of messages that are used by actual implementations of that protocol. The unsupervised learning method relies on support vector clustering - SVC. Our technique is using a new metric - the weighted character position metric. This metric is immune to simple XOR encryptions and does not suppose any knowledge about the protocols: header fields specification, number of messages. One main advantage of the SVC technique is its improvement of the accuracy of the classification for large datasets. We have also proposed a semi automated method allowing to choose the best parameters. The observed message types can be used to induce a tree-like representation of the underlying state machines. The nodes in this tree represent the different types of observed messages and the edges do indicate an invocation relationship between the nodes. This first phase is completed by a second state, where the behavioral differences are extracted and mined. This second phase is using tree kernel support vector machines to model the finite state machines induced from the first phase. The main novelty of this approach lies in the direct usage and mining of the induced state machines. We did test our approach on extensive datasets for several well known protocols: SIP, SMTP and IMAP. The observed empirical accuracy is very good and promising.

Chapter 2

Behavorial and Temporal Fingerprinting

This chapter addresses the fingerprinting of communication protocols based on temporal and behavioral information. Our key contribution is a fingerprinting scheme, where individual fingerprints are represented by tree based temporal finite state machines. We have developed a fingerprinting scheme that leverages supervised learning approaches based on support vector machines for this purpose. We have validated the proposed approach on the Session Initiation Protocol and concluded that very good classification performance is achieved.

The main contribution is a new fingerprinting scheme that is accurate even in the case of fully identical protocol stacks operated however over hardware that has different capabilities (CPU power, memory resources, etc). We look at the fingerprinting problem from another perspective and under more restrictive constraints. We propose a fingerprinting scheme that can learn distinctive patterns in the state machine of a particular implementation. Such a pattern is in our vision a restricted tree finite state machine that provides additional time related information about the performed transition. We define a similarity metric between patterns that is highly accurate for the classification of a given network capture.

2.1 Formal Model

We model a behavioral fingerprint using a Temporal Random Parameterized Tree Extended Finite State Machine (TR-FSM). The TR-FSM is an extension of the parameterized extended finite state machine introduced in [122]. The name of this construct is relative long, we will justify this in the following: the "temporal" aspect concerns the fact that we take into account the temporal properties of the transitions from a finite state machine. The "random" aspect comes from the fact, that we use a probabilistic model for the temporal modeling, where a random variable models the time required for each transition. Our extension concerns the introduction of temporal information and one additional constraint on the transitions in the state machine.

A TR-FSM is formally defined by a tuple $M =< S, s_{init}, I, O, \vec{X}, T, \vec{Y} >$
where:

- S is a finite set of states with $|S| = n$;
- s_{init} is the initial state;
- $I = \{i_0(\vec{v_0}), i_1(\vec{v_1}), \ldots, i_{p-1}(\vec{v_{p-1}})\}$ is the input alphabet set of size p. Each symbol is associated with a vector of parameters;

- $O = \{o_0(\vec{w_0}), o_1(\vec{w_1}), \ldots, o_{q-1}(\vec{w_{q-1}})\}$ is the output alphabet set of size q. Each symbol is associated with a vector of parameters;
- $\vec{X}$ is a vector of variables;
- T is a finite set of transitions and each $t \in T$ is defined as $t =< s_1, s_2, i(\vec{v}), o(\vec{w}), P(\vec{X}, i(\vec{v})), Q(\vec{X}, i(\vec{v}), o(\vec{w})) >$ s_1 and s_2 are the start and end state, i is the input symbol triggering the transition and o is the triggered output symbol. $P(\vec{X}, i(\vec{v}))$ represents the condition to achieve the transition and $Q(\vec{X}, i(\vec{v}), o(\vec{w}))$ is the action triggered by the transition, based on an operation on the different parameters;
- $\vec{Y}$ is a $n-1$ dimensional random vector described later.

Additionally, the transitions of state machine are restricted to form a tree:

$$\forall s \in S \mid s \neq s_{init}, \exists\ !\ r \ states\ s_{i1}, s_{i2}, \ldots, s_{ir}$$

such that:

$$s_{i1} = s_{init} \wedge s_{ir} = s$$

where the notation ij represents a single index. The structure is a tree if there is only one possible sequence of transitions from the initial state to the destination state. Thus, we denote the corresponding transitions:

$$\forall j, 1 \leq j < r, t_{ij} \in T$$

$$t_{ij} =< s_{ij}, s_{i(j+1)}, i_{ij}(\vec{v_{ij}}), o(\vec{w}), P_{ij}(\vec{X}, i(\vec{v}))$$

$$Q_{ij}(\vec{X}, i_{ij}(\vec{v}), o_{ij}(\vec{w})) >$$

Hence, the cardinality of T is defined by $|T| = n - 1$ and $T = \{t_1, \ldots, t_{n-1}\}$.

Finally $\vec{Y}$ is a $n-1$ dimensional random vector with Y_{t_j} representing the (measured) average time to perform the transition t_j.

In the reminder of this chapter, states and transitions are synonyms for nodes and edges because the TR-FSMs are also trees and state machines. Thus, a TR-FSM can be characterized by its **height** and its **cardinality** corresponding to $|S|$.

One important note should be made. The location at which the measure of time is taken is important, especially when done from a remote site and over a network. The inherent additional noise due to the round-trip time can be filtered out. This is done by taking the network round-trip time into account. Alternatively, if the fingerprinting is integrated within an intrusion detection system, the measurements can be used directly without any other additional filtering, because in this case the system is learning local and deployment-specific parameterized device signatures.

The problem of fingerprinting can be now stated as follows. Given a candidate group of implementations $C = \{M_1, M_2, \ldots, M_k\}$ and a set of behavioral fingerprints $\{T_{j1}, T_{j2}, \ldots, T_{jp}\}$ for each implementation M_j, the goal is to find a classifier that correctly maps behavioral fingerprints to the corresponding classes.

We assume a similarity measure $\Delta(T_1, T_2)$, which is a distance based on the the tree structure and the vector $\vec{Y}$, between two TR-FSMs T_1 and T_2.

2.2 Fingerprinting framework

SIP messages are divided into two categories: requests and responses. Each request begins with a specific keyword like `REGISTER, INVITE, OPTIONS, UPDATE, NOTIFY`... The SIP responses

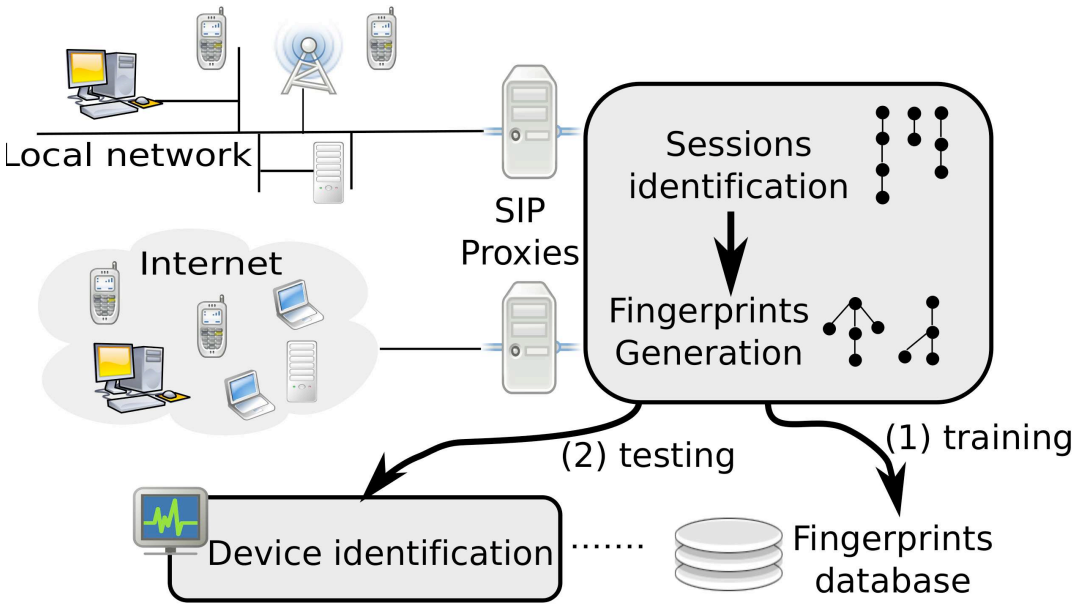


Figure 2.1: Fingerprinting architecture

begin with a three-digit numerical code divided into six classes identified by the first digit. Figure 2(a) gives some examples of SIP sessions.

A **session** is composed of a sequence of messages and its delimitation depends on the protocol. Considering SIP protocol, a session is identified by a specific identifier (SIP call ID). Because an identifier can be reused several times, a session is considered finished after an inactivity period, or after reception of specific messages.

2.2.1 Architecture

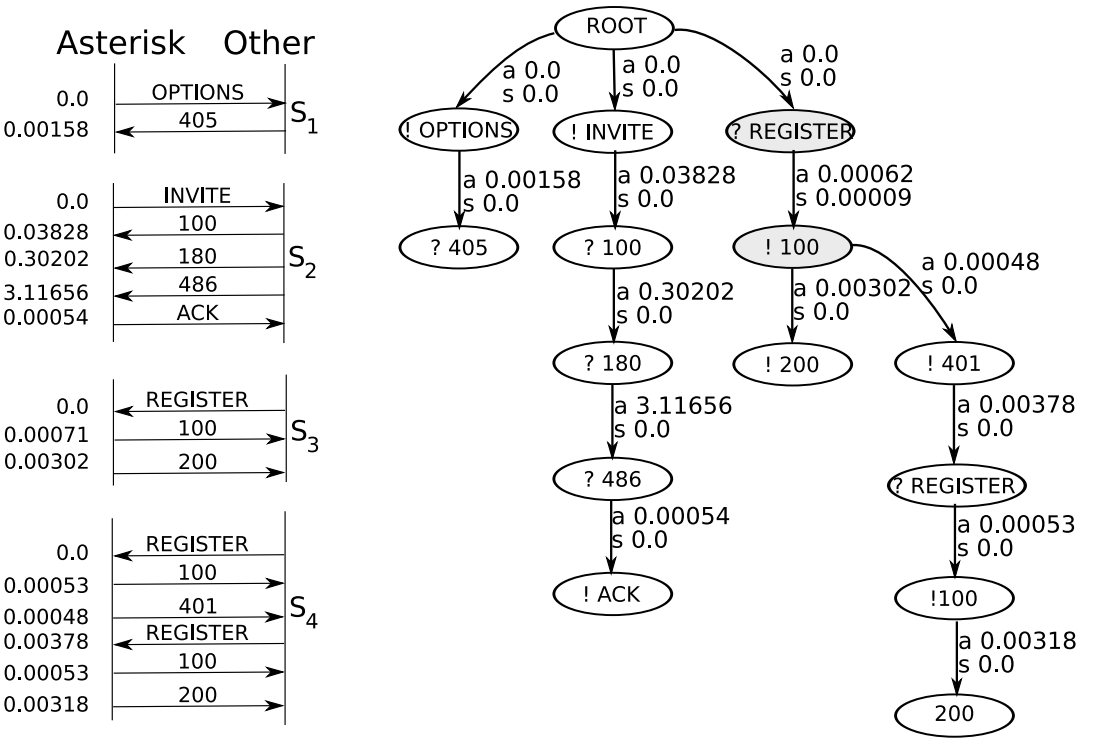
Figure 2.1 depicts our fingerprinting architecture. First, SIP traces are collected from the local network or Internet through a proxy where the clients are connected. Consequently, the clients are not connected via a dedicated network, entailing much noise on the traffic. The first step aims to identify the different sessions and to create the corresponding fingerprints as TR-FSMs (the next section details this step). The next stage is divided into two parts:

- during the learning phase (1), the fingerprint database is generated by identifying the devices using some knowledge of their characteristics. For example, the SIP user agent field (device identifier) can be used if the collected traces are assumed to be free of malformed messages.

- during the testing phase (2), the device identification module tests new fingerprints against the database in order to detect device changes or to check newly connected devices.

2.2.2 Fingerprint generation

The fingerprint is a tree with a generic ROOT node. The fingerprint represents a specific device and is generated from a subset of sessions in which this device participates. Each state of the TR-FSM is represented by a type composed of SIP request type or the SIP response code



(a) Sessions (left value = time) (b) A signature for an Asterisk server generated from four sessions

Figure 2.2: Example of the fingerprint generation

prefixed by **!** (outgoing message at the device fingerprinted) or **?** (ongoing message at the device fingerprinted). Figure 2(b) illustrates a TR-FSM corresponding to an Asterisk server. Therefore, nodes prefixed by **!** are messages sent by Asterisk, whereas those prefixed by **?** are emitted by any second party. The value corresponding to **a** (see figure 2(b)) represents the average time for the transition, while the value corresponding to **s** represents the standard deviation. This tree represents a signature for the Asterisk SIP proxy. A transition is indicated by an arrow between two states. In addition, the vector $\vec{Y}$ corresponds to the average delays put on the edges like in figure 2(b).

The signature in figure 2(b) is generated from the sessions shown in figure 2(a). In fact, each session is represented by a sequence of states and the shared prefixes are merged. For instance, sessions S_3 and S_4 of the figure 2(a) have two first messages in common and so they share the first two nodes which are gray colored in figure 2(b).

2.3 Automated fingerprinting

2.3.1 Terminology

A dataset is composed of TR-FSMs. For a given dataset, the size N is the number of TR-FSMs $t_1, t_2, \ldots t_N$ that it contains. We follow the standard methodology in supervised learning. Each dataset is divided into a learning set used to train the system and a testing set. The testing set is used to evaluate the performance of the system when generalizing on new data. Each sub-dataset also has an associated size: N_train and N_test with $N = N\_train + N\_test$.

The number of sessions extracted for building each tree is named **session size**: **training session size** for the training set and **test session-size** for testing set. These are important parameters for our method.

There are $N\_devices$ distinct devices:
$D = d_1, d_2, \ldots d_{N\_devices}$.

Two functions can be applied to each tree t_i:

- $real(t_i)$ returns the real identifier (device or implementation stack) for a TR-FSM t_i
- $assigned(t_i)$ returns the class name (device or implementation stack) for a TR-FSM t_i that is assigned by the fingerprinting scheme.

2.3.2 Supervised learning for fingerprinting

We briefly review the basics of support vector machines (SVM) in this section in order to make the chapter self-contained. Additional reference material can be found in [133]. We rely on the multi-class classification [67] and adapted it to our fingerprinting task. The chosen approach is known as the one-to-one technique due to its good trade-off between classification accuracy and computational time [81].

Assuming the terminology of the previous section, the SVM classes correspond to the $N\_devices$ devices, and the input space data points are the N_train trees from the training set. Firstly, each point t_i of the training set is mapped to a high-dimensional feature space thanks a non-linear map function $\phi(t_i)$. Then, for each class pairwise $< c_l, c_k >$, an hyperplane with the maximum separation from both classes is found. First, we define the points involved for these classes:

$$\begin{aligned} T_l &= \{t_i | real(t_i) = c_l\} \\ T_k &= \{t_i | real(t_i) = c_k\} \end{aligned} \tag{2.1}$$

Then, the hyperplane is defined by a vector w^{lk}, a scalar b^{lk} and is constrainted by:

$$\begin{aligned} &\forall t_i \in \{T_l \cup T_k\} \\ \langle \phi(t_i) \cdot w^{lk} \rangle + b^{lk} &\geq 1 - \xi^{lk}_{t_i},\ if\ real(t_i) = c_l \\ \langle \phi(t_i) \cdot w^{lk} \rangle + b^{lk} &\geq -1 + \xi^{lk}_{t_i},\ if\ real(t_i) = c_k \end{aligned} \tag{2.2}$$

where the ξ terms are slack variables allowing some classification errors, some points not on the correct side of the hyperplane because this is necessary when data points are not totally separable. The corresponding optimization problem can be converted to its dual form using the Lagrangian. Assuming that $\rho^{lk}_{t_i}$ is equal to 1 when $t_i \in T_L$ and -1 when $t_u \in T_K$, the problem is:

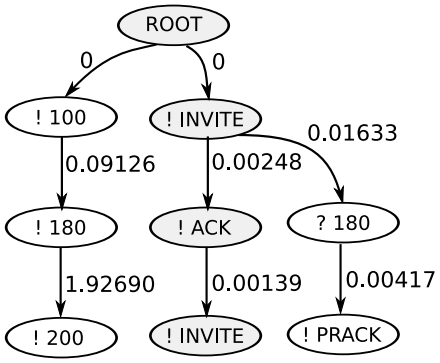
$$max \sum_{t_i \in \{T_l \cup T_k\}} \alpha^{lk}_{t_i} - \frac{1}{2} \sum_{\substack{t_i \in \{T_l \cup T_k\} \\ t_j \in \{T_l \cup T_k\}}} \alpha^{lk}_{t_i} \alpha^{lk}_{t_j} \rho^{lk}_{t_i} \rho^{lk}_{t_j} K(t_i, t_j) \tag{2.3}$$

subject to:

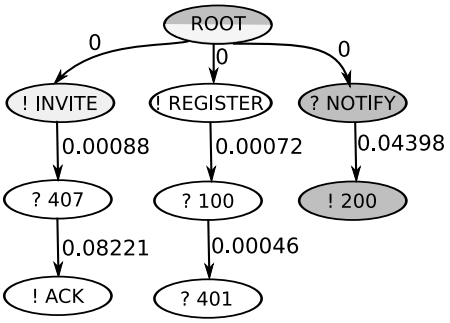
$$\begin{aligned} \sum_{t_i \in \{T_l \cup T_k\}} \alpha^{lk}_{t_i} \rho^{lk}_{t_i} &= 0 \\ 0 \leq \alpha^{lk}_{t_i} \leq C,\ t_i &\in \{T_l \cup T_k\} \end{aligned} \tag{2.4}$$

where K is a kernel function such as the following dot product holds:

$$K(t_i, t_j) = \langle\ \phi(x_i).\phi_(x_j)\ \rangle \tag{2.5}$$



(a) Twinkle 1.10 (softphone)



(b) Cisco 7940 firmware 8.93 (hardphone)

Figure 2.3: Sessions tree examples of one hardphone and one softphone. The attribute on a directed edge is the average delay of the transition. Shared paths are grey colored.

This kernel trick allows the problem to be solved without computing or knowing the ϕ function. The only requirement is a kernel function which has to be applied to each pair of data points. It is basically a similarity function constrained by Mercer's theorem [62]. Finally, a decision function, applied to each t_x of the testing set, is defined as:

$$f_{lk}(t_x) = \sum_{t_i \in \{T_l \cup T_k\}} \alpha_{t_i}^{lk} \rho_{t_i} K(t_i, t_x) + b^{lk} \tag{2.6}$$

In fact, the support vectors are the trees t_i with non-zero $\alpha_{t_i}^{lk}$ and form the set SV^{lk} from which b^{lk} is obtained:

$$b^{lk} = \frac{1}{|SV^{lk}|} \sum_{t_i \in SV^{lk}} (\rho_{t_i}^{lk} - \sum_{t_j \in \{T_l \cup T_k\}} \alpha_{t_i}^{lk} \rho_{t_j}^{lk} K(t_j, t_i)) \tag{2.7}$$

During the testing stage, each decision function f_{lk} is applied to t_i, where t_i is a TR-FSM to classify. Depending on the return value, t_i is assigned to the class c_l or c_k. Using a voting scheme, the class chosen most often is considered to be correct.

Figure 3(b) shows a behavioral fingerprint for a SIP hardphone, while figure 3(a) presents a fingerprint for a softphone. However, the softphone makes one transition almost ten times faster then the hardphone. Therefore, if properly captured and used, time-related information can be be very useful when the same application is executed on different hardware, it will reflect differences in the architectural and computational features. For instance, the same SIP stack running on a CPU-limited capabilities hardphone will show higher transition times than the same stack on a high-performance workstation (softphone). The figures 3(b) and 3(a) illustrate this hypothesis.

2.3.3 Kernel function

The kernel function is one important parameter in SVM applications. The Gaussian kernel is a well-known possible function for simple data points given by a tuple of values. However, the current problem data points are trees with labelled edges. Therefore, we proposed an extension of our previous method [21], based on the tree comparison method proposed in [46]. The goal is to obtain a similarity equal to 1 for exactly the same trees and 0 for totally different ones. Firstly, the set of paths from the root to each node of the tree t_i is designated by $paths^i$ and composed

of m paths: $path_1^i, \dots path_m^i$ where $path_j^i$ represents a single path. The function $nodes(path_j^i)$ returns only the nodes and transitions without delay properties. The function $nodes(paths^i)$ returns the set of the different paths $paths^i$ of the tree t_i without delays the tree structure.

The intersection of the trees t_i and t_j is defined as:

$$I_{ij} = nodes(paths^i) \cap nodes(paths^j) \tag{2.8}$$

In figure 2.3, the two fingerprint intersections are shaded in gray.

For all shared paths, weight are derived from the delay differences and summed to obtain the similarity measure:

$$inter\_sim = \sum_{\substack{p \in I_{ij} \\ nodes(path_k^i)=p \\ nodes(path_l^j)=p}} weight(paths_k^i, paths_l^j) \tag{2.9}$$

Without considering the delays, $path_l^j$ and $path_k^i$ are exactly the same for a given p. A comparison function is then calculated for each node $n_p \in p$ based on the Laplace kernel. Consequently, the new similarity measure is:

$$weight(p_1, p_2) = \sum_{n_p \in p_1} e^{-\alpha |f_{delay}(n,p_1) - f_{delay}(n,p_2)|} \tag{2.10}$$

where $f_{delay}(n, p)$ is a time-based function which returns the average delay for the ongoing edge from node n in the path p. Because a fingerprint concerns one device only, the delay due to other equipment has to be discarded, and so $f_{delay}(n, p) = 0$ for n a message received by the device (node name prefixed by ?).

Theorem . *The following function is a valid kernel which satisfies Mercer's theorem (Chapter 3 of [62]):*

$$K(t_i, t_j) = \sum_{\substack{p \in I_{ij} \\ nodes(path_k^i)=p \\ nodes(path_l^j)=p}} \sum_{n_p \in p} e^{-\alpha |f_{delay}(n,p_1) - f_{delay}(n,p_2)|} \tag{2.11}$$

Proof. Eq. (2.10), which forms the inner sum, is a valid kernel known as Laplace Kernel K_l. The function $f_{delay}(n, p)$ can be expressed as a real-valued function $f(t_i)$ because n and p are subparts of t_i as well as t_j. Hence, the terms in the sum of K are expressed as $K_l(f(t_i), f(t_j))$, which is also a kernel due to kernel construction properties. Finally, a sum of kernels is also a kernel and so K is a kernel. Readers interested in kernel construction and related proofs are referred the section 3.3 in [62]. □

2.4 Performance evaluation

2.4.1 Metrics

Standard metrics for multi-class classification are defined in [36]. Obviously, the following functions are applied to testing trees only. The number of trees corresponding to a particular device d is denominated as x_d. The number of trees classified as device d is y_d. The number of trees classified as device d_1 and which correspond in reality to the device d_2 is $z_{d_2 d_1}$

The sensitivity of a device type d represents the percentage of the corresponding trees which are correctly identified:

$$sens(d) = z_{dd}/x_d \tag{2.12}$$

The specificity of a device d represents the percentage of trees which are labelled as d and which are really of this type.

$$spec(d) = z_{dd}/y_d \tag{2.13}$$

The overall metric, designated fingerprinting accuracy in this paper, corresponds to the percentage of trees correctly identified. The corresponding formula is:

$$acc = \sum_{d \in D} z_{dd}/N\_test \tag{2.14}$$

The mutual information coefficient (IC) is a combination of entropies using the following distribution: $\mathbf{X} = x_i/N\_test$, $\mathbf{Y} = y_i/N\_test$, $\mathbf{Z} = z_{ij}/N\_test$. It is defined as:

$$IC = \frac{H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{Z})}{H(\mathbf{X})} \tag{2.15}$$

where H is the entropy function. This IC is a ratio between 0 and 1 and is maximal for a perfect classification. It is very useful to compare classifications with the same overall accuracy. In this case, the ratio can be degraded when each class is not well represented. For example, it is easy to obtain an accuracy of 80% if 80% of data points are of the same type by assigning all of them to a single class. However, in this case the information coefficient will be 0.

2.5 Experimental datasets

We made extensive use of network traces from which we could extract the SIP user agent (device type) in order to perform both the training and the testing our system. We assumed that our traces did not contain malicious messages, where for instance an attacker spoofed the user agent field. Our implementation is based on the LIBSVM library [49].

We used two kinds of datasets. The first was generated from our testbed composed of various end-user equipment including softphones like `Twinkle` or `Ekiga` and hardphones from the following brands: Cisco, Linksys, Snom or Thomson. The testbed also used servers such as `Asterisk`, OpenSer and Cisco Call Manager. This dataset will be described as `testbed dataset` in the remainder of the chapter. The other datasets designated `operator dataset` (T1 to T4) were provided by four real VoIP operators (about 45MB of traces were extracted). Most equipment is hardphones or SIP servers. The main difference between the two kinds of dataset is the network environment. The first characterizes a local network, while the `operator dataset` capture traffic from devices that connect from the Internet. This implies greater noise and longer delays, as shown in the table 2.1. We used these different target environments intentionally in order to validate the robustness of our approach in noisy conditions. Obviously, the time delays are relevant when comparing different datasets, but within one dataset, the fingerprinting process should be able to properly identify each device. Table 2.1 shows main characteristics of the datasets: the number of different devices, the number of messages, as well as the number of INVITE messages, which indicates the number of VoIP calls made through the network. Although the `operator dataset` are more complete in terms of messages and devices, the number of INVITEs is quite low, indicating that most of the SIP sessions are not phone calls, but registration requests. This reflects realistic SIP traffic, as all SIP user agents have to periodically send out a registration request in order to maintain the binding between a SIP

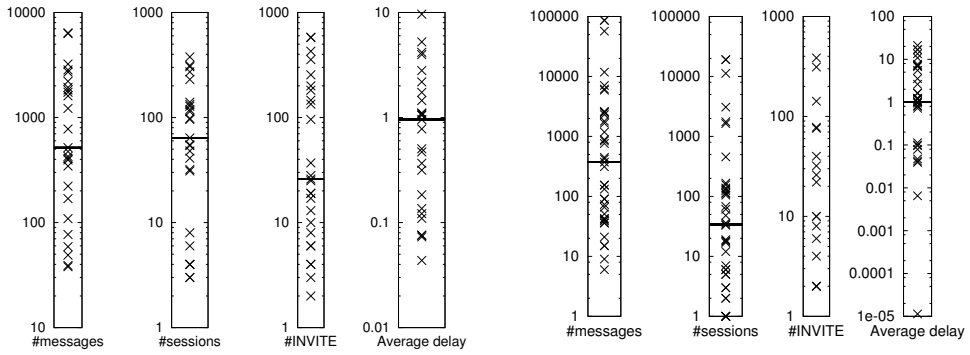
	Testbed	T1	T2	T3	T4
#devices	26	40	42	40	40
#messages	18066	96033	95908	96073	96031
#INVITE	3183	1861	1666	1464	1528
#sessions	2686	30006	29775	30328	30063
Avg #msgs/session	6.73	3.20	3.22	3.16	3.20
Avg delay (sec)	1.53	7.32	6.76	6.11	8.52

Table 2.1: Experimental datasets statistics

AOR (the generic and global identifier for a user) and the current IP address. Being able to fingerprint devices just by looking at the registration messages is also important for device level authentication.

Table 2.4 highlights some of the differences between the devices for the `testbed dataset` and the first operator T1. Each point in the figure represents one device. We considered only messages emitted by the corresponding device and we used a logarithmic scale. For the two datasets, the distribution of messages per device is obviously not uniform, reflecting reality because some devices are used more than others. This implies that the differences between devices for the number of sessions and INVITE messages are similar. Additionally, the distribution ranges of the number of messages and the number of sessions is greater for the operator T1 (figure 4(b)). Hence, the differences between devices are highlighted. For instance, one device has generated only one SIP session while another has generated more than 10,000 as shown on the second graph of figure 4(b).

Due to the difference in the average time delays, it seems possible to fingerprint devices based on such pieces of information. However, when these differences are however insignificant, additional information is needed. Our approach combines the temporal aspect with the behavioral aspect. For example, in figure 4(b), four or five groups of devices can be easily identified just by comparing the average delays. Considering the dataset T1, the transition delays are generally higher than for `testbed dataset` and the median value is doubled.



(a) Testbed

(b) Operator T1

Figure 2.4: Experimental dataset statistics by device (Logarithmic scale; horizontal black bar is the median value; each point represents a device)

2.6 `testbed dataset` results

The characteristics of the `testbed dataset` are depicted in figure 4(a) and in table 2.1. We used it to assess the accuracy of the behavioral and temporal fingerprinting. One objective was to determine the impact of the different parameters on these performance metrics and tune them. These tuned parameters would then be used on the larger `operator dataset`.

We randomly selected 40% of the sessions of each device to form the training set. The remainder (60%) represents the testing set. Each experiments was run ten times, shuffling the sessions before selection in order to improve the validity of the experiments. Then, the average values over the different instances of the classification metric are considered. Furthermore, we use quartiles to gain an idea of the distribution of the results. Figure 2.5 represents quartiles, where the extrema are the minimal and maximal observed values. The lower limit of the box indicates that 25% of the observations are below this value. The upper limit of the box is interpreted in the same way with a percentage of 75%. Finally the horizontal line inside the box is the median value. Therefore, the box contains the 50% of the observations closest to the median.

With the exception of Section 2.6.3, α_1 and α_2 are set to 1000.

2.6.1 Session-size tree

We first investigate the optimal session sizes for training. The test session-size is more important because it shows how reactive the system is. In the best case, a session size of one implies the recognition of one device with only one session. Secondly, we look at the relationship between testing session size and training session size.

Table 2.2 provides a short summary of this data. The shading key simply highlights the main observations concerning fingerprinting accuracy. Our technique cannot be applied to detect a device with only one session (first column is very pale). The darkest row corresponds to a train session-size of five. Using a training session size of five and a testing session size of ten, the maximal accuracy ($\sim 90\%$) is obtained. Subsequent experiments assume this optimal configuration. It can be seen that, even if our technique is not designed for single session device identification, its results are very good. Using only ten sessions or even five sessions, the corresponding accuracy is about 86%.

Finally, the low standard deviation shown in brackets indicates that the accuracy is still about the same during the different experiments especially in the best configrations (dark gray).

Regarding the average sensitivity appearing in table 2.3, the optimal configuration is still the same and the corresponding accuracy is 65%. This relatively low result is due mainly to some incorrectly fingerprinted devices. In fact, some devices are poorly represented in the dataset as shown in figure 4(a). For instance, a training session size of five and a training set of 40% of sessions results in a minimal number of $\lceil 5/0.4 \rceil = 13$, sessions which is not the case for six devices (figure 4(a). Furthermore, this minimal value implies only one training tree and all learning clustering techniques need more training data for efficiency. The impact of training set size is studied in the next subsection.

Although comparing identically-sized trees seems more logical and probably more efficient, this experiment shows the reverse due primary to our comparison function, which considers the various paths in the trees separately (see equations (2.8)-(2.11)).

2.6.2 Training set size

As it was previously mentioned, the fingerprinting accuracy per device is highly affected by underrepresented devices. We assess the minimal training trees per device capable of achieving good results. This number varies from 1 to 20 in figure 2.5. Firstly, if there are at least two trees

Training session size	Testing session size				
	1	5	10	20	40
1	0.682 (0.009)	0.819 (0.013)	0.830 (0.013)	0.805 (0.031)	0.745 (0.034)
5	0.469 (0.028)	0.858 (0.013)	0.905 (0.011)	0.883 (0.025)	0.800 (0.035)
10	0.376 (0.044)	0.809 (0.011)	0.894 (0.013)	0.873 (0.021)	0.819 (0.035)
20	0.272 (0.028)	0.656 (0.028)	0.821 (0.015)	0.864 (0.015)	0.837 (0.012)
40	0.221 (0.027)	0.469 (0.026)	0.627 (0.030)	0.764 (0.037)	0.762 (0.038)

$< 50\%$	50-70%	70-80%	80-85%	85-90%	$\geq 90\%$

Table 2.2: `testbed dataset`: Average fingerprinting accuracy (standard deviation is put in brackets)

Training session size	Testing session size				
	1	5	10	20	40
1	0.504 (0.011)	0.542 (0.034)	0.553 (0.032)	0.535 (0.044)	0.529 (0.043)
5	0.294 (0.026)	0.605 (0.035)	0.647 (0.035)	0.648 (0.047)	0.580 (0.045)
10	0.224 (0.028)	0.550 (0.017)	0.625 (0.023)	0.636 (0.024)	0.599 (0.047)
20	0.145 (0.021)	0.452 (0.050)	0.572 (0.030)	0.615 (0.045)	0.622 (0.027)
40	0.109 (0.028)	0.316 (0.030)	0.399 (0.032)	0.505 (0.050)	0.522 (0.038)

$< 30\%$	30-40%	40-50%	50-55%	55-60%	$\geq 60\%$

Table 2.3: `testbed dataset`: Average sensitivity (standard deviation is put in brackets)

for each device, the accuracy is more than 80% in most cases. Thus, a training session size of 5 implies at least $5 \times 2 = 10$ sessions for the training process, which is reasonable. Going further, the accuracy is close to 90% for a minimal training set size equals eight.

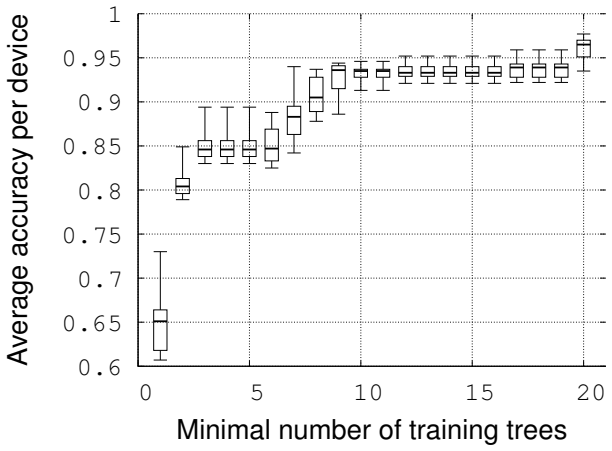


Figure 2.5: `testbed dataset`: Learning trees minimal number impact (test session-size = 10, training session size = 5, $\alpha = 1000$)

2.6.3 Effect of the α parameter

The parameter α is introduced in formula (2.11), and has a potential impact on fingerprinting accuracy, since it impacts the average delay weight. The higher α is, the more important are small delay differences. Figure 2.6 highlights the impact of α on average accuracy by showing the quartiles. Its shape is a parabola with smallest values at the extremities. Broadly, when considering a reference time, a difference between 1 second and 4 second has to be interpreted differently from the difference between 56 and 59 seconds. This can be achieved by increasing α. However, when α is too high, the difference between 0.1 second and 0.2 second could be too discriminatory. This means that the correct trade-off is the maximal value on figure 2.6. In fact, the values 100, 1000 or 10000 are possible choices because accuracy is similar. However, we prefer $\alpha = 1000$ as the median value is the best, and above all the results are closely concentrated around the median. We expect a similar accuracy independent of the sessions selected for use in the training stage.

2.7 Global results

We consider a training session-size of five and a test-session of ten because this configuration previously gave the best results. Table 2.2 gives all statistics and results. The initial rows are related to dataset statistics. Considering the `testbed dataset`, even when more sessions are selected for the testing process, the number of testing trees is lower due to a higher test session-size. Each experiment is performed three times for the `operator dataset` and ten times for the `testbed dataset`. Except for the number of trees, which is fixed for all experiments, the average values are given, with the standard deviation in brackets. For the `operator dataset`, only 10% of sessions are used for the training stage. It is important to note that the standard deviation of maximal and average heights and cardinality is high. This shows that our experiments cover

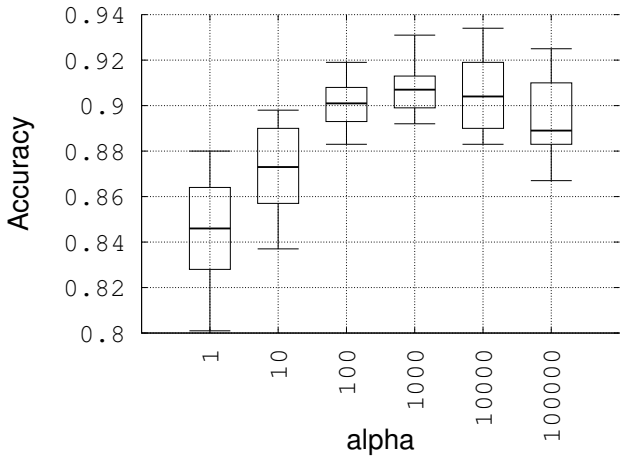


Figure 2.6: `testbed dataset`: α parameter impact (testing session size = 10, training session size = 5)

many configurations. At the same time, the classification results in the lower part of the table are stable, as highlighted by a low standard deviation, demonstrating that our fingerprinting approach is suited to many distinct configurations. Obviously, the TR-FSMs of the `operator dataset` are higher and bigger because the datasets are more complete.

Considering the operators, the overall accuracy reaches about 86%, which is lower than the `testbed dataset` (91%), due principally to additional noise on Internet. Moreover, the mutual information coefficient (IC) for the `testbed dataset` is very high, indicating that the high accuracy is not due to an over-represented device. However, this coefficient is lower for the `operator dataset` because some devices are clearly presenting higher numbers than others, as highlighted in 4(b). Once again, for several devices, the number of sessions is too low to have complete training sets and so the average sensitivity is concentrated between 45% and 58%. However, the specificity is always high, meaning that the misclassified trees are well-scattered among the different devices.

2.8 Conclusion

In this chapter, we have addressed the problem of fingerprinting devices and/or implementation stacks. Our approach is based on the analysis of temporal and state-machine-induced features. We introduced the TR-FSM, a tree-structured parameterized finite state machine having time-annotated edges. A TR-FSM represents a fingerprint for device/stack. Several such fingerprints are associated with a device. We propose a supervised learning method, where support vector machines use kernel functions defined over the space of TR-FSMs. We validated our approach using SIP as a target protocol. We will continue this work in two main directions. Firstly, we will look at other protocols — for instance file transfer protocols — and assess the operational applicability in this scenario. This would for instance allow the identification of rogue access points within a large wireless access infrastructure. A second research direction consists of defining other kernel functions specific to the TR-FSMs that allow the modeling of the probability distribution of transition times at each edge. This will leverage not only the average transition

Metric	Testbed	T1	T2	T3	T4
#Training trees	440	1223	1217	1237	1224
#Testing trees	332	5409	5367	5471	5423
Max height	71.95 (32.03)	464.67 (41.35)	476.33 (38.58)	420.33 (30.56)	431.33 (0.94)
Min height	1.9 (0.30)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Avg height	9.53 (2.13)	8.80 (1.53)	8.85 (1.89)	8.70 (1.73)	9.05 (1.38)
Max card	89.00 (35.72)	492.67 (44.68)	491.17 (47.65)	540.84 (157.00)	464.84 (21.52)
Min card	3.95 (1.56)	2.67 (0.47)	2.00 (0.00)	2.00 (0.00)	3.00 (0.00)
Avg card	18.97 (4.69)	12.93 (2.68)	12.94 (3.09)	12.85 (2.98)	13.23 (2.56)
Accuracy	0.91 (0.011)	0.81 (0.004)	0.86 (0.001)	0.85 (0.002)	0.83 (0.004)
Sensitivity	0.64 (0.030)	0.53 (0.019)	0.58 (0.026)	0.54 (0.012)	0.43 (0.015)
Specificity	0.91 (0.035)	0.79 (0.001)	0.81 (0.025)	0.77 (0.028)	0.77 (0.028)
IC	0.87 (0.012)	0.64 (0.001)	0.65 (0.001)	0.65 (0.003)	0.63 (0.004)

Table 2.4: Experimental datasets results ($\alpha = 1000$, test session-size = 10, train session-size = 5). Average values given and standard deviations in brackets

time for one edge, but also the underlying probability distribution.

Chapter 3

Security Monitoring in VoIP

3.1 Introduction

This chapter addresses our contributions in developing efficient monitoring solutions for VoIP. We have done this work in the framework of Mohamed Nassar's Ph.D research thesis, which I co-advised with Olivier Festor. The most relevant publications made on these topics are [106, 107, 105, 104]. We have made two major contributions : the first one proposes a monitoring framework, where SIP level information is extracted and a supervised learning method is used to detect VoIP specific attack. The second contribution is considering the design of a VoIP specific honeypot.

3.2 Monitoring SIP Traffic

We proposed a monitoring scheme that is illustrated in Fig. 3.1. We track slices of SIP messages. Each slice is mapped to a set of features that is used to classify this slice. Our approach is based on a supervised learning phase in which couples (vector, class Id) have been used to learn the differences between normal slices and attack related slices. From a high level view, the features are grouped in four categories:

- **General statistics :** these statistics represent the general shape of the traffic and indicate the degree of congestion. The fraction of requests carrying SDP bodies (normally INVITE, ACK or UPDATE) is a good indicator because it will not exceed a certain threshold. An excessive use of re-INVITE or UPDATE for media negotiation or maybe QoS theft increases the number of SDP bodies exchanged and decrements the average inter-arrival of them. Flooding attacks are associated with peaks of all these statistics.
- **Call-Id based Statistics:** similar to the Erlang model used in the telecommunication networks, where the arrival rate of calls and the average duration of a call characterize the underling traffic, the arrival rate of Call-Ids (can be starting a call or any kind of SIP dialog) and the interval time of messages having the same Call-Ids, can be used to characterize the overlay SIP traffic. Nevertheless, we notice that non-INVITE dialogs have shorter durations and fewer number of messages than INVITE dialogs. Thus their Call-Id statistics can be taken as different features.
- **Distribution of final state of dialogs/Call-Ids:** the following states are defined: NOTACALL: for all non-INVITE dialogs, CALLSET: for all calls/INVITE dialogs that do not complete the initiation, CANCELED: when the call is cancelled before it is established, REJECTED: for all redirected or erroneous sessions, INCALL: when the call is established

but not realized yet, COMPLETED: for a successful and ended call and RESIDUE: when the dialog does not start with a request. This latter is a residual of messages in a previous slice. In a normal situation where the size of the unit is large enough, NOTACALL, COMPLETED and REJECTED (in busy or not found situations) dominate this distribution. Major deviations may indicate an erroneous situation.

- **Distribution of SIP requests and SIP responses:** for instance, the number of REGISTER sent by a user within a time interval is indirectly proportional to the period of registration (`expires` parameter or `Expires` header). Violations of this rule might indicate anomalies. Similarly, an unexpected high rate of error responses is a good indication for error situations.

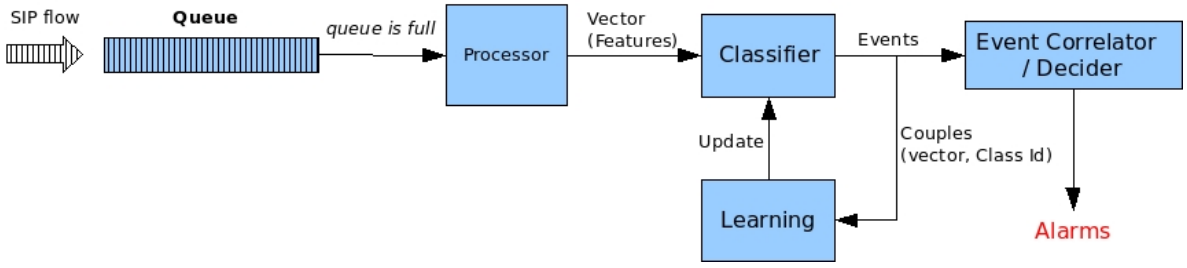


Figure 3.1: Real-time Online SIP Traffic Monitoring

We consider a support vector machine as underlying anomaly detection engine [50]. Fig. 3.2 illustrates the overall processing of our system.

Figure 3.2: SVM Flow Chart

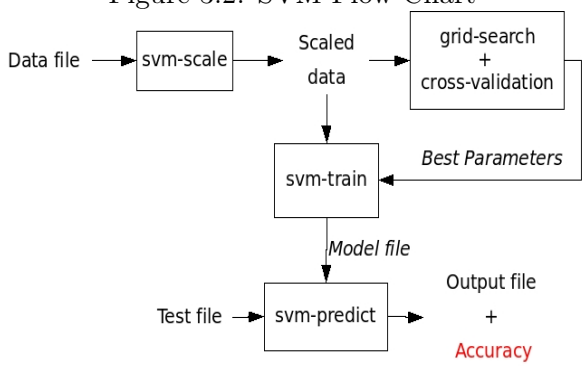
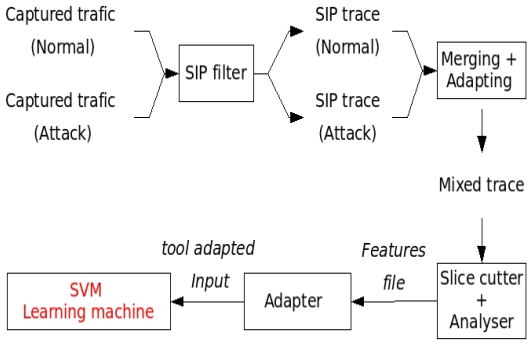


Figure 3.3: Analysis Flow Chart



For each slice, the analyzer evaluates a set of predefined features (38 variables are defined in our study) and builds a vector for the SVM. All vectors are assembled in one file and annotated as either attack vector or normal vector. In Fig. 3.3, this process is shown for a mixed trace.

We have used the Inviteflood tool [72] to launch SIP flooding attacks. We have used INVITE flooding with an invalid domain name (which is the most impacting on any SIP server). We have generated five attacks at five different rates, where each attack lasts for one minute. After adaptation (we assume that one machine of the real world platform is performing the attack), each one minute attack period is injected into a normal trace of two hours duration. The time of the attack is fixed to five minutes after the start of the two hours period. Each mixed trace is then analyzed and labeled properly (positively along the period of attack and negatively in all the remaining time).

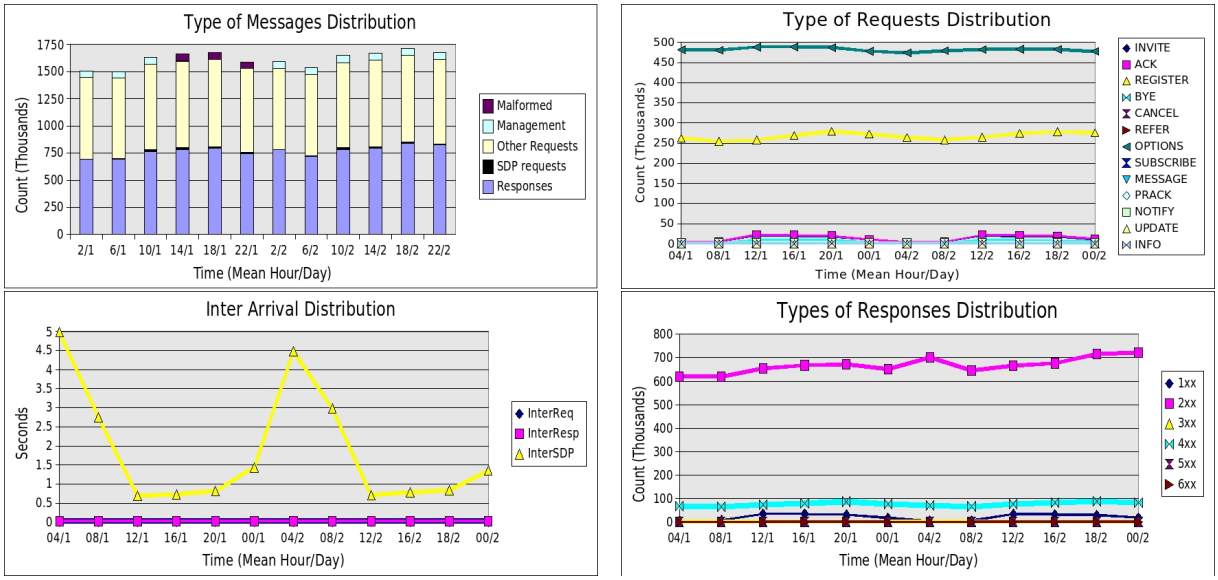


Figure 3.4: Long Term Statistics over Real World Traces

We have trained the system with the mixed trace (flooding at 100 INVITE/s - normal trace) in the learning stage. This means that 100 INVITE messages are taken as a critical rate (the rate we consider as threshold to launch an alarm).

As shown in Fig. 3.5 (for simplification and clarity sake a slice is sized to only three packets), we take the period of attack and we calculate the corresponding SVM estimation. The estimated probability is the average of the estimated probabilities for the elementary slices composing the attack traffic. This granular probability is given by the LibSVM tool and is useful for both the probability estimate option in both learning and testing stages. We define the detection accuracy as the percentage of vectors correctly classified as attack over all vectors of the attack period.

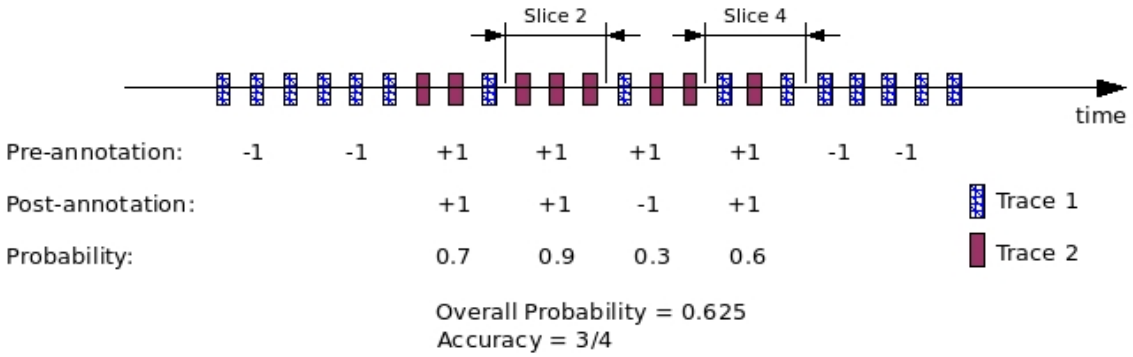


Figure 3.5: Attack Detection in a Mixed Trace

Even though stealthy attacks cannot be detected, the results show a promising opportunity to fine-tune the defensive solution. The threshold rate can be learnt by a dual trace : the ongoing normal/daily traffic and a stress condition where the server was troubleshooted or was noticed to be under-operating. In this way, SVM is promising for an adaptive online monitoring solution against flooding attacks.

SPIT mitigation is one of the open issues in VoIP security today. Detection of SPIT alone is not sufficient if it is not accompanied by a prevention system. In-depth search in the suspicious

# of Concurrent Calls	True Positives (%)	True Negatives (%)
RBF; C= 1; $\gamma = 1/38$; Training accuracy = 99.0249		
1	0 (0/3697)	100
10	1.30 (10/766)	
50	10.01 (62/619)	
100	18.31 (102/557)	
Linear ; C=1 ; Training accuracy = 99.0197		
1	0 (0/3697)	100
10	2.09 (16/766)	
50	10.66 (66/619)	
100	19.39 (108/557)	

Table 3.1: Detection of Partial SPIT in Four Mixed Traces With Different Intensities

traffic is needed to build a prevention system to block the attack in the future. Elements like IP source and the destination in the SIP headers can be automatically extracted.

We have performed two experiments with two different hit rates. The former is a partial SPIT: Spitter targets the proxy with hundred destinations and among these only ten are actually registered users. In this case the hit rate is just 10%. This emulates the real world scenario where attackers are blindly trying a list of extensions. The latter is a total SPIT: we assume that attackers knew already the good extensions so the hit rate is 100%. This emulates the real world scenario where attackers knew already the good extensions either by a previous enumerating attack or from a web crawler.

In the partial SPIT experiment (SPIT not covering all the domain extensions, hit rate < 100 %), we send four successive campaigns with respectively one, ten, fifty and hundred concurrent calls. In the first campaign, Spitter does not start a dialog before the previous dialog is finished. In the second campaign, ten dialogs go on at the same time and only when a dialog is finished, a new dialog is started.

The four resulting traces (duration about two minutes each) are injected - after adaptation (we assume that one agent of the real trace is performing the attack against the hundred other agents) - in four different normal traces (duration of two hours each). The traces are then cut into slices of thirty messages and analyzed. These are annotated positively for the period of attack and negatively in all the remaining duration. The mixed trace with fifty concurrent calls SPIT is used in the training stage. The SVM prediction results are shown in Table 3.1. True positives are the percentage of vectors correctly classified as attack over all the vectors of the attack period. True negatives are the percentage of vectors correctly classified as normal over all the vectors of the normal period. These results should be considered under the larger umbrella of event correlation. For instance, the example with ten concurrent calls:

- Most of the two hours traffic is normal and is correctly detected (47436 slices).
- 16 out of the 766 slices that compose the attack traffic are detected. This means that we have ten correct events in a period of two minutes, because the detection of one slice is highly relevant to all ongoing traffic around this slice.

In addition, the attacks are partial since they target a small fraction of the users of the VoIP server (more than 3000 users are identified in the two hours period). We agree that a stealthy SPIT of the magnitude of one concurrent call is never detected, but in the case of hundred

# of Concurrent calls	1	10	50	100
RBF; C= 1; $\gamma = 1/8$; Training accuracy = 98.9057				
True Positives	0.03 2/7015	3.05 15/492	12.18 85/698	23.41 184/786
True Negatives	100			

Table 3.2: Detection of Full SPIT in Four Mixed Traces With Different Intensities

concurrent calls, one over five positives is successfully detected when training was done using half of this intensity attack.

With the help of a set of deterministic event correlation rules, our online monitoring system is able to detect the attacks efficiently:

Predicate	**SPIT intensity**
10 distributed positives in a 2 minutes period	Low
Multiple Series of 5 Successive Positives	Medium
Multiple Series of 10 Successive Positives	High

In the full SPIT experiment, we request hundred VoIP bots to register with the proxy. Results are slightly better than in the partial SPIT experiment (Table 3.2). Partial SPIT generates an abnormal traffic at the same level as full SPIT does.

3.3 VoIP honeypots

Our second contribution has addressed the design and specification of a VoIP specific honeynet, which we call honeyphone. The honeyphone registers a number of SIP URIs at the SIP that has to be defended. The SIP URIs of the honeypot may be declared to the outside world as users of the domain, but because they do not represent real users they should theoretically never be called. When this SIP proxy receives an INVITE to one of these registered numbers, it will forward the request to the honeyphone agent. The overall architecture is presented in figure 3.6 and described below:

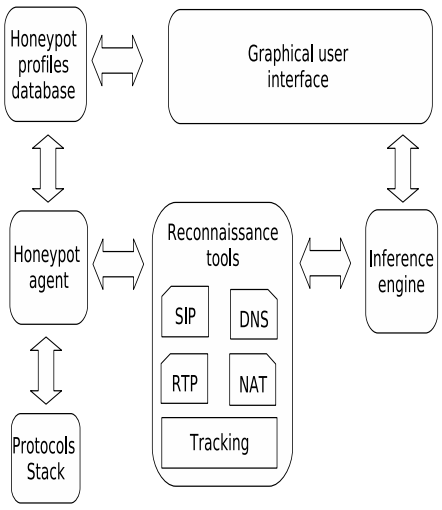


Figure 3.6: Honeyphone architecture

- The honey-pot agent: is the core of the honeypot architecture and the intelligent part of the application. It is responsible for accepting incoming calls and investigating possible attacks.

- The protocol stacks (SIP, SDP, RTP): are built based on the protocol standards, and are responsible for building and parsing the messages, as well as the transmission and reception of the packets over the transport layer.

- The honeypot profiles database: contains several configuration files and thus allows the administrator to choose one profile which is suitable to its needs rather than build it by itself. The benefit of this component is twofold, first, to set up the honeypot in its environment, and second, to control its behavior. We strive for application scalability and dynamism in the way we build up our profiles. Indeed, profiles in the database could be far from the honeypot area. They can be bots assessing the performance or the security of the domain.

- Reconnaissance tools: are used in the investigation procedure to check the received messages (e.g. service scanning: nmap [1], sip messages crafting: SIPSAK [2], passive OS finger printing tool: pOf [3]).

- The inference engine: is able to interpret automatically the results of investigation by means of special metrics and a Bayes model. we have chosen the Bayes model due to its adaptive capability, scalability and real time performance. The belief output is on behalf of the domain administrator.

- The graphical user interface: allows the administrator to choose and setup a honey-pot profile, as well as visualize traces, alerts and statistics.

Once a suspicious INVITE message is received, the inference engine will assess as much as possible from the underlying pieces of information, like for instance the IP addresses and protocol ports. The concerned pieces of the INVITE message are the domain part in the `From` header, `Contact` and `Via` headers, `Record-Route` and `Route` if present, and from the SDP body `Connection(c), Owner(o)` and `Media(m)` parameters. For each IP address or host name in the above cited list, the inquiry procedure aims to evaluate the following fields:

- `WHOIS` : ownership and real world information by means of Internet registries. In the honeyphone database, a WHOIS trust table assigns for each entry a trust coefficient between 0 and 1.

- Valid DNS : checking of DNS information by means of DNS, Reverse DNS, DNS SRV [2] or ENUM [4] requests according to the case;

- `AS` : the autonomous number where the IP or host resides. In the honeyphone database, an AS trust table assigns for each entry a trust coefficient between 0 and 1.

- `GL` : the geographic location of the IP or host. In the honeyphone database, a location trust table assigns for each city a trust coefficient between 0 and 1.

- `Has Port` : if a port (SIP or RTP) proclaimed to be open is really open;

[1] Nmap:Network Mapper, `http://www.insecure.org/nmap/`
[2] Sipsak: SIP Swiss Army Knife, `http://sipsak.org/`
[3] pOf: passive OS fingerprinting tool, `http://lcamtuf.coredump.cx/p0f.shtml`

- `Fingerprint` : normally found in the `User-Agent` header, however we can actively fingerprint the caller device by sequence of OPTIONS requests as proposed in [138]. A fingerprint trust table assigns for each device a trust coefficient between 0 and 1. The fingerprinting can be enhanced with techniques that have resulted from our own work and described in the previous chapters.

- `Historical cache`: check in if and how frequently the IP or host is seen by the honeyphone. The maximum frequency is 1 and is reached if the IP or host was seen 10 times or more. An aging threat decreases the frequency and can delete the entry if it was not seen for a long time.

In addition, the inquiry procedure makes IP trace routes starting from the honeypot and arriving to the different hosts and proxies mentionned in the INVITE. The outputs of the inquiry procedure are brought towards the inference engine.

3.3.1 Conclusions

This chapter has presented an online monitoring methodology based on support vector machines. The ongoing signalling (SIP) traffic is sliced into small slices. From each slice, we extract a vector of defined features characterizing each slice. Vectors are then pushed into a SVM for classification based on a learning model. A deterministic event correlator is used to raise an alarm when suspicious and abnormal situations occur.

We have validated our approach by offline tests over a set of real world traces and attacks which are generated in our customized testbed and inserted in the normal traffic traces. Results showed a real time performance and a high accuracy of detecting flooding and SPIT attacks especially when coupled with efficient event correlation rules. Detection of other types of attacks are future work.

Unsupervised learning techniques are appealing because they don't need a priori knowledge of the traffic and can detect new and previously unknown attacks. We consider currently to redefine and reorder our set of features based on different features selection algorithms. We will extend the current event correlation and filtering algorithm in order to reveal attack strategies and improve intrusion prevention/detection accuracy.

We have also addressed an innovative approach to design a VoIP specific honeypot. The honeyphone is supplied by an application program interface which controls a rich set of network tools. This honeyphone is able to gather information in real time about the received messages. We will have to assess our honeyphone in an operational environment in order to measure its performance.

Chapter 4

Fuzzing for vulnerabilities

4.1 Introduction

This chapter shares some essential practical experience gathered over a two years period in searching for vulnerabilities in the VoIP space. We have identified several attacks that completely change the VoIP threat model defined in [73]. All of the described vulnerabilities have been disclosed responsibly by our group and were discovered using our in-house developed fuzzing software KIF

This work was done in the context of Humberto Abdelnur's Ph.D research, which I did co-supervise wth Olivier Festor. The relevant publications issued from our work are : [13, 17, 23, 22, 24, 19, 14].

We have described the stateful fuzzing approach in [23, 22]. This chapter will overview the fuzzing architecture from a high level perspective and focus more on the applied results that were obtained. We believe that the current VoIP security threat model should be revised taking into account these new attacks.

We performed fuzzing of different devices and SIP stacks in order to validate our research on automated and smart fuzzing. All of the described tests, were performed with our developed tool, described in [20]. Our fuzzing approach is based on stateful protocol fuzzing for complex protocols (like for instance SIP). To the best of our knowledge, this is the first SIP fuzzer capable to go beyond the simple generation of random input data. Our method is based on a learning algorithm where real network traces are used to calibrate an attack automata. This automata is evolving during the fuzzing process. Our work in this area was motivated by two major factors: firstly we practically validated the formal research contributions in the area of fuzzing. Secondly, we discovered vulnerabilities and followed a responsible disclosure policy by helping vendors to fix them and notifying the affected parties via large distribution mailing lists, web sites and podcast.

4.2 Fuzzing Voice over IP devices

All VoIP devices do embed SIP stacks (which are in charge of processing SIP messages) and have to implement a rather complex state machine. In most cases, access to the source code of the SIP stacks is impossible and for most VoIP hardphones, running in dedicated equipment, no debugging possibility exists for an independent security researcher. The only resort to perform security assessment is in this case to perform black-box security testing. We have performed our security and fuzzing experiments over a broad scoped and heterogeneous testbed which is summarized in table 4.1.

Device	Firmware
Asterisk	v1.2.16, v1.4.1
	asterisk-addons-v1.2.8 asterisk-addons-v1.4.4
Cisco 7940/7960	vP0S3-07-4-00
	vP0S3-08-6-00
	vP0S3-08-7-00
Cisco CallManager	v5.1.1
FreePBX	v2.3.00
Grandstream Budge Tone-200	v1.1.1.14
Grandstream GXV-3000	v1.0.1.7
Linksys SPA941	v5.1.5
	v5.1.8
Nokia N95	v12.0.013
OpenSer	v1.2.2
Thomson ST2030	v1.52.1
Trixbox	v2.3.1

Table 4.1: Tested equipment

All the experiments were performed with our tool, KiF [20]. KiF consists in two autonomous components, the Syntax Fuzzer and the State Protocol Fuzzer. The tests may be similar to the normal behavior or can flood the device with malicious input data. Such malicious data can be syntactically non compliant (with respect to the protocol data units), or contain semantic and content wide attack payload (buffer overflows, integer overflows, formatted strings, or heap overflows).

The Syntax Fuzzer takes a fuzzer scenario and the provided ABNF [63] syntax grammar to generate new and crafted messages. The fuzzer scenario drives the generation of the rules in the syntax grammar and may also depend on the State Protocol Fuzzer in order to generate the final message (appropriated or not) to be sent to the target entity.

The State Protocol Fuzzer does passive and active testing. Therefore, two state machines are required: 1) one specifying the SIP state machine and 2) one specifying the testing state machine. The first state machine is used for the passive testing and controls if there is any abnormal behavior coming from the target entity during the execution of the tests. This state machine may be infered from the SIP traces of the target entity. The second state machine is used for the active testing and is driving the profile of the security test. This state machine is defined by the user and can evolve over time. Figure 4.1 shows the overall framework of KiF.

4.3 Weak Input Validation

The most frequent vulnerability that we encountered is related to weak filtering of input data. This filtering does not properly deal with metacharacters, special characters, over lengthy input data and special formatting characters. Most of these vulnerabilities are due to buffer/heap overflows, or format string vulnerabilities. The most probable cause is that developers assumed a threat model in which VoIP signaling data would be generated only by legitimate SIP stacks. The real danger of this vulnerability comes from the fact that in most cases, one or very few packets can completely take down a VoIP network. This is even more dangerous when realizing that in these cases the SIP traffic is carried over UDP, such that highly effective attacks can be performed stealthy via simple IP spoofing techniques. Table 4.2 shows some of our published vulnerabilities, where we have decided to highlight two extreme cases: The first vulnerability (disclosed in *CVE-2007-4753*) reveals that even the simplest check for the existence of the input is not performed and that even simple attacks can be very effective. The second case, (*CVE-2007-1561*) is situated at the opposite site, since a VoIP server is concerned by an attack with a

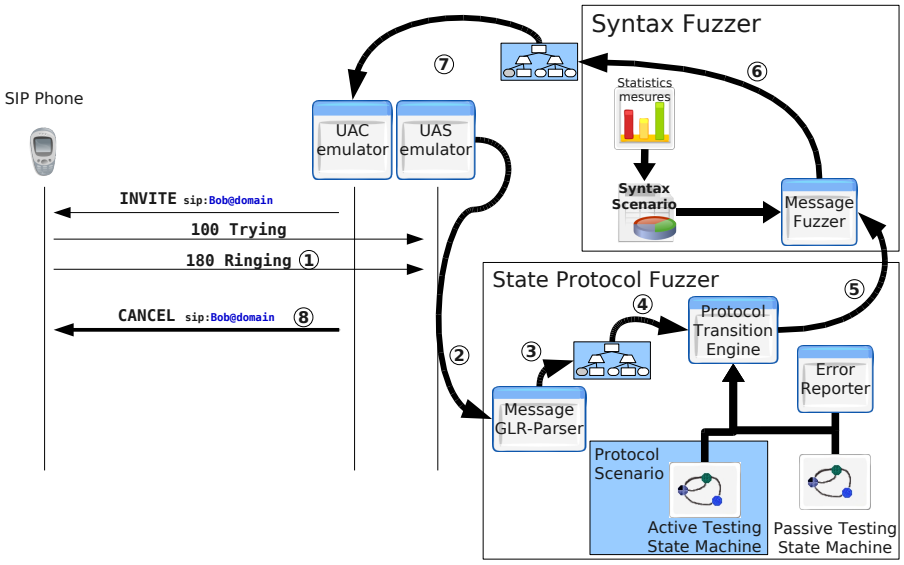


Figure 4.1: KiF framework

Device	Synopsis	CVE-Identifier	Impact
Asterisk v1.4.1	Invalid IP address in the SDP body	CVE-2007-1561	DoS
Cisco 7940/7960 vP0S3-07-4-00	Invalid Remote-Party-ID header	CVE-2007-1542	DoS
Grandstream Budge Tone-200 v1.1.1.14	Invalid WWW-Authenticate header	CVE-2007-1590	DoS
Linksys SPA941 v5.1.5	Invalid handling of the \377 character	CVE-2007-2270	DoS/String overflow
Thomson ST2030 v1.52.1	Invalid SIP version in the Via header	CVE-2007-4553	DoS
	Invalid URI in the To header	CVE-2007-4753	
	Empty packet		
Linksys SPA-941 v5.1.8	Unescaped user info	CVE-2007-5411	XSS attacks
Asterisk v1.4.3	Unescaped URI in the To header	CVE-2007-54881	SQL injection and Toll-fraud
FreePBX v2.3.00 Trixbox v2.3.1	Unescaped URI in the To header	[16]	XSS attacks

Table 4.2: Input Validation Vulnerabilities - Examples

rather complex input structure. The danger in this case is that one single packet will crash the core VoIP server and thus lead to a complete take down of the whole VoIP network.

Preventing these types of attacks at a network defense level is possible with deep packet inspection techniques and proper domain and application specific packet filtering devices [89].

4.3.1 Attacks against the internal network

Most VoIP devices have embedded web servers that are typically used to configure them, or to allow the user to see the missed calls, and all the call log history. The important issue is that the user will check the missed calls and other device related information from her machine, which is usually on the internal network. If the information presented is not properly filtered, this same user will expose her machine (located on the internal network) to malicious and highly effective malware. We will illustrate the following example discovered during a fuzzing process (see *CVE-2007-5411*). The Linksys SPA-941 (Version 5.1.8) VoIP phone has an integrated web server that allows for configuration and call history checking. A Cross Site Scripting vulnerability (XSS) [75] allows a malicious entity to perform XSS injection because the "FROM" field coming from the SIP message is not properly filtered. By sending a crafted SIP packet with the FROM field set

to :

```
"<script x='" <sip:'src='beef.js'>@domain>;tag=1"
```

The browser is redirected to include a javascript file (y.js) from an external machine (baloo) as shown in Figure 4.2. This external machine is under the control of an attacker and the injected javascript [75] allows a remote attacker to use the victim's machine in order to scan the internal network, perform XSRF (Cross Site Request Forgery) attacks, as well as obtain highly sensitive information (call record history, configuration of the internal network), deactivate firewalls or even redirect the browser towards malware infested web pages (like for instance MPACK [6]) to compromise the victim's machine. The major and structural vulnerability comes in this case, from the venture of two technologies (SIP and WEB) without addressing the security of the cross-technological information flow.

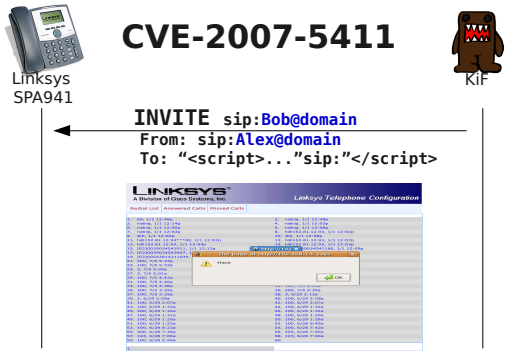


Figure 4.2: Linksys SPA-941 XSS attack

The impact of this vulnerability is very high : most firewalls/IPS will not protect the internal network against XSS attacks delivered over SIP. Additionally, users will connect to these devices directly from the internal network and therefore the internal network can be compromised. Jeremiah Grossmann [75] showed how firewalls can be deactivated with XSS attacks. Unfortunately, most VoIP devices have weak embedded WEB applications, such that other vulnerable systems exist and are probably exploited in the wild.

4.4 Protocol Tracking Vulnerabilities

Protocol tracking vulnerabilities go beyond simple input filtering of single messages. In this type of vulnerability, several messages will lead a targeted device in an inconsistent state, albeit each message on its own does not violate the SIP RFC [116]. These vulnerabilities are caused by weak implementations of protocol state engines. Exploiting these vulnerabilities can be done in two main ways:

1. the device might receive inputs that are not expected in its current protocol state: for instance, when waiting for a BYE method, an INVITE is received,

2. slight variations in SIP dialog/transaction tracking fields leading a device towards an inconsistent state.

The discovery of such vulnerabilities is difficult. The fuzzing process should be able to identify where a targeted device is not properly tracking the signaling messages and which fields can be fuzzed in order to detect it. The search space in this case is huge being spread over many

Device	Synopsis	CVE-Identifier	Impact
Cisco 7940/7960 vP0S3-08-6-00	Does not handle unexpected messages (e.g. OPTIONS) inside an existing INVITE transaction	CVE-2007-4459	DoS
Grandstream GXV-3000 v1.0.1.7	Unexpected message inside an INVITE transaction allows to remotely accept the call	CVE-2007-4498	Remote Eavesdropping
CallManager v5.1.1 OpenSer v1.2.2	Authentication uses not one-time nonces	CVE-2007-5468 CVE-2007-5469	Replay Attacks
SIP Protocol Relay Attack	Attacker can trigger the target entity to authenticate to him	[15]	Toll-Fraud
Cisco 7940/7960 vP0S3-08-7-00	Does not handle six INVITE transaction destinated to any user	CVE-2007-5583	DoS
Nokia N95 v12.0.013	Does not handle a CANCEL at an unexpected timing in an INVITE transaction	CVE-2007-6371	DoS

Table 4.3: Stateful Vulnerabilities

messages and numerous protocol fields, requiring thus advanced and machine learning driven fuzzing approaches. Table 4.3 shows such disclosed vulnerabilities having different complexity grades.

A simple case (*CVE-2007-6371*), where a CANCEL message arrives earlier than expected, can turn the device into an inconsistent state which will end up in a Denial of Service state, as shown in Figure 4.3.

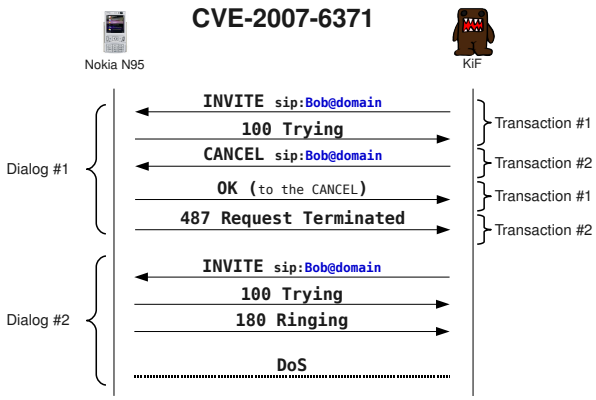


Figure 4.3: Nokia N95 DoS attack

The major danger with this type of attacks is that no application level firewall can completely track so many flows in real time and that even in the case of known signatures, polymorphic versions of known attacks can be easily obtained and these will remain under the security radar. As of today, unfortunately no effective solution to prevent this type of attacks exists.

4.4.1 Toll Fraud vulnerabilities

Toll frauds occur when the true source of a call is not charged. This can happen by the usage of a compromised VoIP infrastructure or by manipulating the signaling traffic. It is rather amazing to see that although technology evolved, the basic conceptual trick of the 70's, where phreakers reproduced the 2600 Hz signal used by the carriers is still working. Thirty years after, the signaling plane can still be tampered with and manipulated by a malicious user. What did change however, is the needed technology. Nowadays, we can inject SQL commands [93] in the signaling plane, and the toll fraud is possible. In the following, we will describe in detail one vulnerability found during a fuzzing process [16]. Some SIP proxies store information gathered

from SIP headers into databases. This is necessary for billing and accounting purposes. If this information is not properly filtered, once it will be displayed to the administrator it can perform a second order SQL injection, that is during the display, the data is interpreted as SQL code by the application. In this case, two consequences can result: First, the database can be changed -for instance the call length can be changed to a small value - and thus the caller can do toll fraud. If we consider Asterisk [9], the popular and largely deployed open source VoIP PBX, Call Detail Records (CDR) are stored in a MySQL database.

FreePBX [5] and Trixbox [11] use the information stored in such database in order to manage, compute and generate billing reports or display the load of the PBX.

Some functions do not properly escape all the input characters from fields in the signaling packets.

A first flavor of this specific attack can be performed by a subscribed user of the domain and the attack consists of injecting negative numbers in the CDR table in order to change the recorded length/other parameters of a given call. The direct consequence is that no accurate accounting is performed and the charging process is completely controlled by an attacker.

A second and more serious consequence is that this attack can be escalated by injecting JavaScript [75] tags to be executed by the administrator PC when she will perform simple management operations. In this case, a Cross-Site Scripting Attack (XSS) [75] can be performed, because malicious JavaScript can be stored into the database by the SQL injection. This malware gets executed on the browser when the administrator will check it - this is a similar process to the log injection attacks known by the Web application security community. Similarly to the previous case, tools like Beef and XSS proxy can scan the internal network, deactivate firewalls and realize all the CSRF/XSRF specific attacks.

The main issue is that most current applications that deal with CDR data are not considering this type of threat. If the target system is not well secured, SQL injection can lead to system compromise because most database servers allow some interaction with the target OS [93].

This type of vulnerability is rather dangerous because few applications (none of which we have tested) implement filtering on SIP headers. All applications do consider SIP related information to be sourced from a trusted origin and no security screening is performed. The mitigation should be proper input and output filtering whenever data is stored/read from another software component.

4.4.2 Remote Eavesdropping Vulnerabilities

We discovered a rather unexpected vulnerability in *(CVE-2007-4498)*. Several SIP messages sent to the affected device put the phone off-hook without ringing or making any other visual notification. The attacker is thus capable to remotely eavesdrop all the conversations performed at the remote location. Figure 4.4 shows the messages exchanged by the attack. The impact if this vulnerability goes beyond the simple eavesdropping of VoIP calls, because an entire room/location can be remotely monitored. This risk is major and should be considered when deploying any VoIP equipment. Although in the presented case, a software error was probably the cause, such backdoors left by a malicious entity/device manufacturer represent very serious and dangerous threats.

4.4.3 Weak Cryptographic Implementations

The authentication mechanism in SIP is a standard shared secret and challenge-response based one [82]. Nonces are generated by the server and submitted to an authenticating entity. The latter must use its shared key to compute a hash which is afterwards sent to the authenticator. This hash is computed on several values: SIP headers, nonces and random values. A received

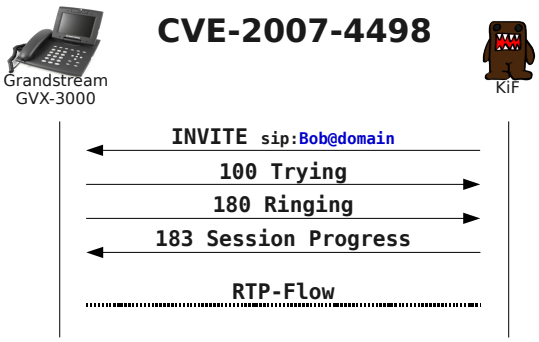


Figure 4.4: Grandstream GXV-3000 remote eavesdrop

hash is validated by the server and checked to authenticate a client. For efficiency reasons, very few server implementations track the life-cycle of a valid token. We have found at least two vulnerabilities *CVE-2007-5468 and CVE-2007-5469*, where intercepted tokens could be replayed. These vulnerabilities are not simple man in the middle attacks, since intercepted tokens were reusable for long periods of time and they could be used for the authentication of any other call. Figure 4.5 shows the flow of messages for such attack. The impact of such a vulnerability is very high. Toll frauds and spoofing call identifiers are the straightforward consequences. The mitigation consists in trading off performance versus security and implementing efficient and secure cryptographic token management procedures.

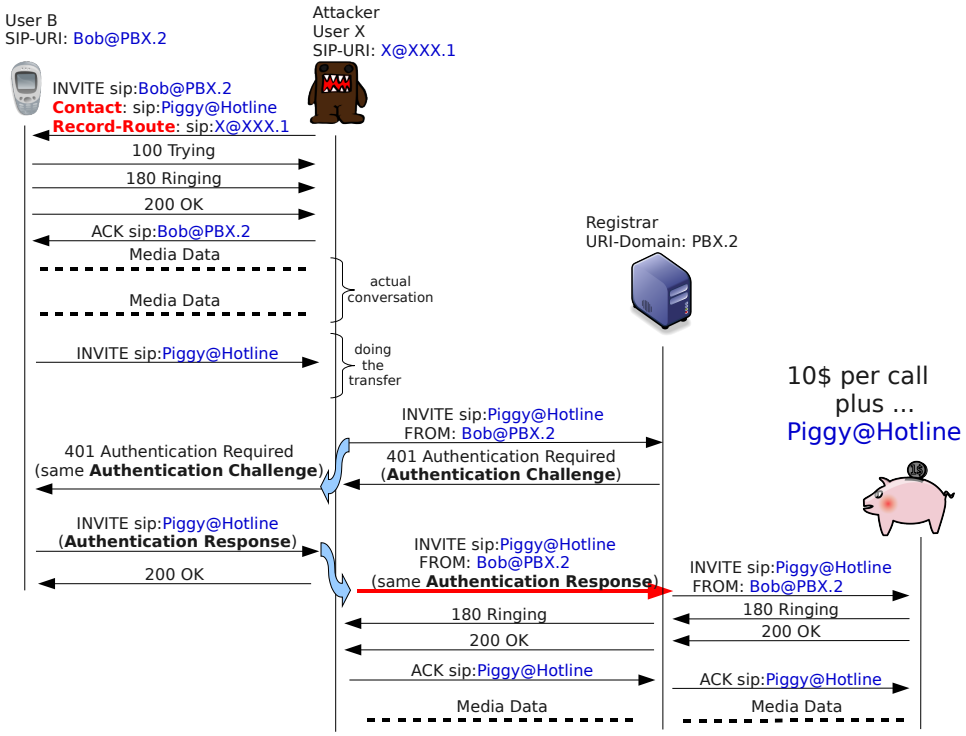


Figure 4.5: Replay Attack

4.5 Specification Level Vulnerabilities

We discovered during a complex fuzzing scenario the same anomaly (and apparent vulnerability) shared by all devices under test (table 4.1). We realized that in fact the SIP protocol itself has a major design vulnerability that makes toll fraud possible on any VoIP network [15]. The major issue is that a classical relay attack is possible by forcing a called party to issue a Re-Invite operation. Due to the novelty and severity of it, we will detail the attack in the following.

An attacker issues a call to the victim, the victim answers it and later on puts the attacker on hold. To address this put on hold, an accomplice of the attacker may initiate another call. Once the attacker receives the re-INVITE specifying the "On hold", he/she will request the victim to authenticate. This last authentication may be use by the attacker to impersonate the victim at its own proxy.

Notations:

- P is the proxy located at URL: proxy.org
- X is the attacker located at URL: attacker.lan.org
- V is the victim located at URL: victim.lan.org
- V is also registered with P under the username victim at proxy.org
- Y is the accomplice of X (it can be in fact X), but we use another notation for clarity sake

The described attack will show how X calls a toll fraud number 1-900-XXXX impersonating V.

1. X calls' directly V.
   "The route set MUST be set to the list of URIs in the Record-Route header field from the request...The remote target MUST be set to the URI from the Contact header field of the request." RFC 3261 [116] Section 12.1.1 UAS calls:

   ```
   X ------ INVITE victim.lan.org ------> V
       From : attacker at attacker.lan.org
       To: victim at victim.lan.org
       Contact: 1900-XXXX at proxy.org
       Record-Route: attacker.lan.org
   ```

2. The normal SIP processing:

   ```
   X <----------- 180 Ringing ----------- V
   X <------------- 200 OK -------------- V
   X <------------ Media Data ----------> V
   ```

3. The accomplice Y steps in and invites victim V, and then the victim decides to put X on hold.

4. The victim, V, sends a re-INVITE to X (to put it on hold)
   "The UAC uses the remote target and route set to build the Request-URI and Route header field of the request." RFC 3261 [116] 12.2.1.1 Generating the Request (Requests within a Dialog)

```
X <--- INVITE 190XXXX at proxy.org --- V
    From: victim at victim.lan.org
    To : attacker at attacker.lan.org
```

5. X calls 1900-XXXX using the proxy P and the proxies asks X to authenticate using a Digest Access Authentication with nonce="Proxy-Nonce-T1" and realm ="proxy.org"

6. X request the victim to authenticate the re-INVITE from step 4 using the same Digest Access Authentication received in step 5

```
X ------- 401/407 Authenticate ------> V
   Digest: realm ="proxy.org",
       nonce="Proxy-Nonce-T1"
```

7. In this step the victim will do the work for X (Relay Attack)

```
X <--- INVITE 190XXXX at proxy.org --- V
   Digest: realm ="proxy.org",
       nonce="Proxy-Nonce-T1"
       username= "victim",
       uri="1900XXXX at proxy.org",
       response="the victim response"
```

8. X may reply now to the Proxy with the valid Digest Access Authentication computed by the victim. Note that the Digest itself it is a perfectly valid one.

4.5.1 Formal Model

To examine the SIP protocol security we have used the AVISPA tool. This work was done in collaboration with the INRIA project Cassis and published in [13, 19, 18].

AVISPA is a push-button tool for the Automated Validation of Internet Security Protocols [1] and provides a modular and expressive formal language, HLPSL (The High-Level Protocol Specification Language) for specifying protocols and their security properties. HLPSL allows the specification of control-flow patterns, data-structures, alternative intruder models, complex security properties, as well as different cryptographic primitives and their algebraic properties. We have confirmed the protocol specification vulnerability using AVISPA and have shown that extending SIP with one more operation can mitigate this attack. The solution has been automatically validated by the tool.

4.6 Conclusions

The quantitative conclusions after a long term work on searching vulnerabilities in the VoIP space are rather pessimistic. Feedback and support when contacting vendors remains highly unpredictable and poor. All tested devices have been found vulnerable. The scope of the detected vulnerabilities is very large. Trivial input validation vulnerabilities affecting highly sensitive communication materials are rather usual. More complex and protocol tracking related ones do also exist, though their discovery and exploitation is rather complex. The cause of these vulnerabilities is the weak software security life-cycle of their vendors. The integration of Web

and VoIP technology is a Pandora's box comprising even more powerful and hidden dangers. Web specific attacks can be carried out over the SIP plane leading to potential devastating effects, like for instance the complete compromise of an internal network. This is possible since no application specific firewall today can easily interwork with several technologies and no proper guidelines for the secure interworking of Web and VoIP exist. The more structural cause is a missing VoIP specific threat model. The VOIPSA did develop a threat model [10] which however does not reflect the current state of knowledge on VoIP vulnerabilities. Highly efficient Denial of Service attacks can be done with single-shot packets, remote eavesdropping goes beyond the simple call interception and the VoIP plane itself can be a major security threat to the overall IT infrastructure. Much remains to be done in the future, among which "Security Build in VoIP devices" remains the major among them. Changes in the software development cycles must be followed by a comprehensive security assessment and testing. Protocol fuzzing is one essential building block in this landscape, since no other additional approach can be used by independent security research. We have described in this chapter our practical and hand-on experience in testing embedded SIP stack implementation. These tests were performed in order to validate our research on advanced security fuzzing techniques and the discovered vulnerabilities were properly and responsibly disclosed. Our future work will extend it by addressing additional protocols, case studies, implementations and formal approaches.

Chapter 5

Relation with current and prior research work

5.1 Protocol Fingerprinting

Network and service fingerprinting is a common task that is often used by attackers to design efficient attacks. However, it is also a useful and legitimate tool for security assessment, penetration testing and monitoring the diversity of hardware and software on the network. The key assumption is that subtle differences due to development choices and/or incomplete specification can be traced back to the specific device/protocol stack [69]. Several approaches for the remote identification of a device have already been proposed. In most cases, these approaches were based on identifying vendor/device specific deviations in the implementation of a given protocol. Such deviations are possible because of lacks in the specifications/standards - many current specifications either do not completely cover all the exception cases or do lack the necessary precision and thus leave too many degrees of freedom to the software implementers. The existing fingerprinting approaches do exploit this in order to fingerprint a device. For instance, individual fields in the TCP header can lead to the passive fingerprinting as it is implemented in [8].

There are two main classes of fingerprinting scheme: active and passive. Passive fingerprinting monitors network traffic without any interaction. The most efficient tool for this purpose is p0f [8], which uses a set of signatures to identify the operating system that generated a TCP packet. Each signature is based on the specific values in particular TCP/IP header fields. In contrast, active fingerprinting generates specific requests directed to a device and monitors the responses. For instance, [7] implements this scheme in order to detect the operating system and service versioning of a remote device. A related work described in [47], uses active probing and proposes a mechanism to automatically explore and select the right requests to make. These requests can themselves be considered as fingerprints.

The accuracy of these techniques relies on the good definition of messages to send, which is basically done manually. Therefore, [47] describes a mechanism to learn them. Fingerprinting is not limited to OS identification and several works target to correctly distinguish the different kinds of network traffic with different granularity level. The authors of [95] and [79] try to automatically identify a specific protocol. Our work is different and complementary since its goal is to determine precisely which implementation of a protocol is used. Determining the version of a SIP equipment could be based on the bad randomness value of the Call-id field [118], but one major problem with such an approach is that these fields can be easily changed by an attacker. We have addressed this attack vector and proposed a scheme that doesn't relay on the specific value or on the flat structure of the message, but all the underlying hierarchical syntactic structure. This structure is related to the protocol grammar and contains valuable information

that can hardly be spoofed.

SIP fingerprinting is also addressed in [139] where an active probing technique is proposed. This is complementary to our approach, which is entirely passive and does not require any interactions with the target device. Our work relies only on multiple sessions of messages without syntactic knowledge and is well designed for protocols with partial or without specification and grammar. We also introduced the use of syntactic information in [21] to create one generic global tree per device type. We have addressed a related topic and we looked at the identification of the different message types used by an unknown protocol and were able to build up the tracking state machines from network traces. That approach can serve to build TR-FSMs for an unknown protocol without any domain-specific knowledge, especially of the grammar of the protocol.

Application layer fingerprinting techniques, specifically for SIP, were first described in [118]. These approaches proposed active as well as passive fingerprinting techniques. Their common baseline is the lack of an automated approach for building the fingerprints and constructing the classification process. Historically, the first fingerprinting technique in place was to analyze by hand a dump file which was captured by a packet sniffing software like tcpdump [124]. Obviously, this technique is limited as it is both error prone and time consuming. Therefore, new methods appeared. The Protocol Informatics project [39] proposes a solution which uses a well known bioinformatics algorithm and a technique based on sequence alignment. Giving a set of messages from a protocol, the program tries to determine both the unchanging and the variables fields. Recently, the work by J. Caballero et al. [47] described a novel approach for the automation of Active Fingerprint generation which resulted in a vast set of possible signatures. It is one of the few automatic approaches found in the literature and it is based on finding a set of queries (automatically generated) that identify different responses in different implementations. While our work addresses specifically the automation for passive fingerprinting, we can easily imagine these two complementary approaches working together

Several approaches consider the context semantics - the destination computer behavior itself: [92] looks for extracted bytes on the message to rebuild the different fields, [65] is based on the execution trace of the system calls, [48] proposes a dynamic binary analysis to identify separators and keywords. [78] introduces a semi supervised learning method to determine the message fields. Closer to our goal, the authors of [134] cluster the messages captured on the network by types before trying to infer their format. To achieve it, they propose to tokenize each message and to find the different fields by considering that each sequence of binary bytes is a binary token and that each text sequence between two binary bytes is a field. The technique proposed in [136] is also identifying the different fields thanks to a delimiter. This is done by instrumenting the protocol application by studying how the program parses the messages. Application dialog replay is a related domain as the goal is to construct a valid replay dialog by identifying the contents which need to be modifed thanks to sequence alignment techniques [64] or by building a model from application inputs and outputs [108].

.

No previous work did consider both behavioral and temporal aspects of the fingerprinting task at the same time. We have addressed this work and we have leveraged differences in induced state machines in order to perform fingerprinting. Support vector machines have been already proposed in the context of network security monitoring and intrusion detection. For instance, [102] addresses the issue of SVMs in intrusion detection approaches. None of the previous related work addressed the construction of time based behavioral fingerprints. The induction of a protocol state machine from a set of examples has been studied in the past. Although known to be NP complete (see [114], [27] and [117] for good overviews on this topic), the existing heuristics for this task it are based on building tree representations for the underlying finite state machine. In our approach we do not prune the tree and, although the final tree

representation is dependent on the order in which we constructed the tree, we argue that the resulting substrees have good discriminative features.

5.2 Fuzzing

The current generation of fuzzers is the first one, where most of the existing approaches rely on randomly injecting input data without taking into account the syntax, semantics and state of the targeted applications. Such approaches are very useful when testing local applications within a confined environment (debugging session), but are hardly applicable when testing protocol/application stacks, like for instance a SIP stack. Although some results can be obtained in many cases, these are limited to the processing of one single message (INVITE) lacking the capability to track the state of the remote application and to use behavioral level information in the fuzzing process. Our work is motivated by these limits and addressing the building of a second generation of fuzzing concepts. Our first main achievement (described to some extent in our publications) is a stateful protocol fuzzer approach for complex protocols (like for instance SIP). The main characteristics of our approach are flexibility, adaptive fuzzing capability and protocol tracking of the targeted application and device. One of the components of our work is quite generic and reusable for any protocol for which an underlying grammar is known. The second one is dependent on the domain specifics (SIP). To the best of our knowledge, this is the first SIP fuzzer capable to go beyond the simple generation of random input data. Our method is based on a learning algorithm where real network traces are used to learn and train an attack automata. This automata is evolving during the fuzzing process. At an international level, the main academic actors are the University of Oulu (Finnland) and the team at Georgia Tech in the USA (lead by Kevin Almeroth). Based on published results, the work done by the Georgia Tech team is conceptually less developed that ours. Our model allows to evolve the fuzzing tool based on reinforcement messages, while the SNOOZE work (Georgia Tech) is more or less oriented towards a pure random fuzzing. As far as we know, the Oulu team developed a fuzzing concept for SIP which however does not keep the full state of the protocol. Most SIP stack developers use fuzzing techniques to check their code, but none published insofar is even marginally close in terms of efficiency and complexity to our approach. Finally, the well known team at Columbia University, lead by Prof. Henning Schulzrinne, has been performing research on VoIP vulnerabilities, but in their work they defined some static test cases that were used with an open source fuzzer tool (Protos). Therefore, the process was manually driven and not automatic as in our work. Among the pioneering work in the field, the Mini-Simulation Toolkit by Kaksonen in [83], proposed an excellent framework for automatically generating crafted messages with a certain knowledge of states. This work assumes a certain knowledge of the protocol to be tested, and an additional grammar which merges the syntax message and the transition behavior. The same work defines some semantic rules which allow to produce calculated fields like for instance the checksum. However, the syntax grammar and the state protocol are mixed in a same definition, which in fact provides a reduced set of scenarios. Another issue is that the rules to create exceptional messages are limited in functionality. Another successful approach is the one followed by D. Aitel in SPIKE [25, 26], where the concept of block-based fuzzing is introduced. This is based on the fact that protocols are always composed of the same primitives: invariants, blocks and variants. In this case the invariants are kept intact and the block are filled with fuzzed data. However, SPIKE is too low level, so it becomes highly effort-consuming when applied to complex protocols. An academic research by G. Banks et al. called SNOOZE [37] claims to be a stateful protocol fuzzer. It is based in user defined scenarios and a protocol specification. However, the protocol specification as well as the use cases are highly complex to describe while the operations to fuzz the data are limiting. Meanwhile, the stateful concept of the approach is

not clear. By contrast, the approach presented by S. Emblenton in Sidewinder [71], describes a potential approach for the new generation of fuzzers, where a grammar specifying the syntax is provided and rules to evolve it according to the obtained results are presented. Meanwhile, a list of most popular fuzzers can be found on the ThreatMind[4] website.

Very few fuzzer approaches consider the evaluation of the effect of the crafted message. For instance, the work led by the Mini-Simulation Toolkit as the Protos SIP test-suite [128] stands out for its deployment and large test cases. Protos uses a set of test cases for which the analysis of success was done using in/out of band monitoring instrumentation. The research by P.M. Maurer at [98] also proposed several ideas of different approaches to evaluate the impact of the crafted messages. The first consists in generating the crafted message and anticipating what the regular reply should look like. This approach requires knowledge of the protocol and the effect that the crafted data may have. The second approach proposed is to test more than one entity at the same time, where all of them should receive the same message and each one will reply according to its stack. If the replies are different, there is a chance that one of the entities is not respecting the protocol definition or abusing the freedom that it leaves, and therefore, vulnerabilities may be found. David Lee reveals in [90] a technique to test the data portions of a protocol. That work uses a state machine describing the state in which the protocol is at the current moment. Based on the types and properties of the incoming/outgoing messages, transitions of states are done. Each transition that is not compliant with the ones described by the state machine are considered as a protocol error in the implementation. This can be extended for a fuzzing process by generating input data that violates the properties of both the state machine as well as the message types.

5.3 Security Monitoring of VoIP networks

VoIP security is a recent research domain that emerged over the last few years with the increasing use of this technology by enterprises and individuals. Many approaches to protect the emergent VoIP applications have been proposed in the research community. VoIP background, threats explanation and practical recommendations to configure properly, validate and monitor the security infrastructure are subject of recently published books [111]. Dynamic firewall design for IP telephony environments is evaluated and improved in [115, 89]. Suitable intrusion detection and prevention architectures are proposed with prototype implementations as in [137].The authors of [119] present a threat model of the integrated signaling between PSTN and IP networks and propose a global solution to secure the gateways.

Our current work on VoIP security monitoring is based on multilayer intrusion detection and prevention system architecture for VoIP infrastructures, coupled with statistical/machine learning driven intelligence. The key components of the approach are based on a VoIP-specific honeypot and on an application layer event correlation engine based on supporting rule based systems and bayesian networks/support vector machines. We brought the honeypot concept into the VoIP world and showed the interest and potential of such an approach for VoIP. Quittek et al. [112] apply hidden Turing tests and compare the resulting patterns with typical human communication patterns. Passing these tests causes significant resource consumption on the SPIT generation side. The authors of [35] propose a call rank mechanism based on call duration, social networks and global reputation to filter SPIT calls. Other ideas include a progressive and multi (short term -long term) grey level algorithm [121] and incorporating active stack fingerprinting [138]. Their work on SPIT is based on an inline monitoring approach, without taking into account statistical observed behavior. Our work is using Bayesian techniques, where

[4] http://www.scadasec.net/secwiki/FuzzingTools

a dynamical reinforcement/calibration is possible.

There have been relevant results published on detecting malformed SIP packets. Among these works, I would highlight the one done at the University of Columbia (in the team lead by Prof Henning Schulzrinne) [135] aiming at determining malformed SIP packets and mapping SIP vulnerabilities in a ontology. The NEC research lab (Heidelberg, Germany) is working on detecting malicious malformed packets [109, 112] aimed at provoking Denial of service attacks and they also address the prevention against SPIT attacks. The Fokus team (Berlin, Germany) is working on statistical CUSUM based detection of denial of service attacks [140, 74]. We are also addressing denial of service attack detection and the conceptual solution that we consider is based on Support Vector Machines and more generally on machine learning techniques. These solutions are complementary and future joint work with Fokus and NEC Research labs will be initiated. In the USA, the team lead by Pr. Ram Dantu (University of North Texas) [86, 119] is considering greylist/blacklist type of solutions for SPIT detection, which is orthogonal to our solution.

The authors of [113] design application and transport sensors to protect enterprise networks from VoIP DoS attacks based on previous works on TCP DoS protection and study different recovery algorithms. The authors of [51] modify the original state machine of SIP transactions to detect transaction anomalies and apply different thresholds to detect flooding attacks. More adaptive to such attacks is the work of Sengar et al. [120] where the Hellinger distance between learning and testing periods is used to detect TCP SYN, SIP INVITE and RTP floods. Their approach shows good performances. There have been many papers on generic intrusion detection methods [129,68,88] - none of which addressed the specifics of SIP sessions. Over the past, many security related applications have leveraged machine learning techniques and the reader is referred to [110] and [97] for an overview. The closest work to ours is the study of [84] where the authors have presented a traffic behavior profiling methodology and demonstrated its applications in problem diagnosis and anomaly detection. Our work is more oriented towards attack detection and classification rather than proposing a global and multi level profiling methodology.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

We have addressed in this manuscript some of our contributions in the area of security assessment and monitoring. We have started with one essential component of any assessment architecture - a fingerprinting component. We have described a novel approach for generating fingerprinting systems based on the structural analysis of protocol messages. Our solution automates the generation by using both formal grammars and collected traffic traces. It detects important and relevant complex tree like structures and leverages them for building fingerprints. The applicability of our solution lies in the field of intrusion detection and security assessment, where precise device/service/stack identifications are essential. We have implemented a SIP specific fingerprinting system and evaluated its performance. We have extended our work towards the natural evolution, where the underlying grammar is unknown. A second contribution considers the automated fingerprinting of unknown protocols. Our approach is based on the unsupervised learning of the types of messages that are used by actual implementations of that protocol. The unsupervised learning method relies on support vector clustering - SVC. Our technique is using a new metric - the weighted character position metric. This metric is immune to simple XOR encryptions and does not suppose any knowledge about the protocols: header fields specification, number of messages. One main advantage of the SVC technique is its improvement of the accuracy of the classification for large datasets. The observed message types can be used to induce a tree-like representation of the underlying state machines. The nodes in this tree represent the different types of observed messages and the edges do indicate an invocation relationship between the nodes. The first phase is completed by a second state, where the behavioral differences are extracted and mined. The second phase is using tree kernel support vector machines to model the finite state machines induced from the first phase. The main novelty of this approach lies in the direct usage and mining of the induced state machines. We tested our approach on extensive datasets for several well known protocols: SIP, SMTP and IMAP. Another contribution is considering the analysis of temporal and state machine induced features. We introduced the TR-FSM, a tree structured parametrized finite state machine having time annotated edges. A TR-FSM represents a fingerprint for devices/stacks. Several such fingerprints are associated with a device. We propose a supervised learning method, where support vector machines do use kernel functions defined over the space of TR-FSMs. We have also addressed new monitoring approaches for VoIP specific environment as well as VoIP specific honeypots. Our monitoring scheme is based on Support Vector Machines for efficient classification. We have shown that the system is both efficient and accurate and have studied the impact of the various features on the efficiency. We have proposed a novel online monitoring approach to distinguish between attacks and normal activity in SIP-based Voice over IP environments. We demonstrated the efficiency of

the approaches even when only limited data sets are used in learning phase. The solution builds on the monitoring of a set of 38 features in VoIP flows and uses Support Vector Machines for classification. We validated our proposal through large offline experiments performed over a mix of real world traces from a large VoIP provider and attacks locally generated on our own testbed. Results show high accuracy of detecting SPIT and flooding attacks and promising performance for an online deployment are measured. We have validated our approach by offline tests over a set of real world traces and attacks which are generated in our customized testbed and inserted in the normal traffic traces. Results showed a real time performance and a high accuracy of detecting flooding and SPIT attacks especially when coupled with efficient event correlation rules. We have also presented a VoIP specific honeypot that can detect VoIP specific attacks. Finally, we have presented a fuzzing framework for VoIP, which combines stateful fuzzing with efficient and smart data generation techniques. We did test our fuzzing framework on real equipment and did pursued an ethical disclosure policy. We focussed in this manuscript more on the practical outcomes of this research direction because of its high impact on large user basis but also to emphasize the real world applicability of our work. One of our main achievement (described to some extent in our publications) is a stateful protocol fuzzer approach for complex protocols (like for instance SIP). The main elements of our approach are flexibility, adaptive fuzzing capability and protocol tracking of the targeted application and device. One of the components of our work is quite generic and reusable for any protocol for which an underlying grammar is known. The second one is dependent on the domain specifics (SIP). To the best of our knowledge, this is the first SIP fuzzer capable to go beyond the simple generation of random input data. Our method is based on a learning algorithm where real network traces are used to learn and train an attack automata. This automata is evolving during the fuzzing process.

6.2 Research Program

6.2.1 Research in adaptive Security mechanisms

Current Malware becomes in the evolving Internet, the de-facto major threat that comprises large and distributed denial of service attacks, phishing and botnet enabler infrastructure. As of today, most information known about this social and technological threat, comes from the capture of sample code and behavioural analysis of infested systems. These studies are typically performed with a varied ranged of specialized systems that attract malware while still preserving the necessary integrity with respect to the current legislation and scientific methodology. Most existing solutions are statically: they are deployed over a fixed amount of time, using a fixed allocated address range. Unfortunately, we have witnessed over the recent past, a trend in malware to defy such static capturing infrastructures by detecting or avoiding a priori-known honeypots and/or honeyclients. These anti-honeypots enabled malware drive the defensive activities towards new frontiers, by requiring new conceptual paradigms, where allocated IP addresses can vary at rapid pace over a broad range of network infrastructures and geographically spread destinations, coupled with reactive and anti-anti-honeypots measures. My main research interest consists in adaptive security solutions for these new threats.

My objective is to address the conceptual and experimental work that lay out the bases of a new generation of dynamic and adaptive security mechanisms. Taking into account the requirements of dynamically deploying flexible capturing infrastructures over heterogeneous networks (wireless/wired) and services (web server, P2P services, database servers) using either large contiguous address spaces or sparsely spread capturing devices, the addressed work will consists in driving the scientific and technological enablers that make such a vision possible. Among the major research questions that will be considered are:

- how can such a process be deployed?
- what are the underlying analysis and event correlation techniques that can address such a process ?

I will consider a game theoretical framework for this purpose. I am actively pursuing this direction and as of today I have considered the case of reliable realtime P2P communications and showed how the configuration of security mechanisms can be configured using game theoretical concepts, in which the defendant is played by the management plane having to face adversaries which play the attacker role. We have worked on a novel approach for the configuration of reliable P2P realtime communication services [38]. We assumed a threat model in which an attacker can control a subset of the infrastructure and can perform basic and combined attacks based on flooding, incorrect routing and silent dropping, while from a defensive site configuration settings are the only counter-methods. We have casted the configuration problem in the context of game theory and derived optimal strategies for both attacker and defendant. These strategies form the Nash equilibrium(s) of the game.

Our main research direction is the development of a risk assessment framework based on the reliability and performance of the communication plane. Essential measures in this framework could provide the payoffs functions used to compute the Nash equilibrium. We plan to extend this work towards more configuration actions and a broader threat model. As far as we know, this is the first approach on combining network management and game theory.

The Nash equilibrium, named after the economist John Nash who received the Nobel prize for his contribution to his discipline, is a solution concept of a game involving two or more players. According to [103], a Nash equilibrium is a strategy profile specifying optimal strategic choices for all players by reason that none of the players has any motivation to diverge from the Nash equilibrium because one player cannot gain greater payoffs by choosing another strategy when all the other players choose the strategies given by the profile. To calculate the Nash equilibrium we need N as a set of players, A_i as a finite strategy set, and R_i as a payoff function.

- N $\rightarrow$ set of n players
- $A_i \rightarrow$ finite strategy set $(a_i \epsilon A_i)$
- $R_i : A \rightarrow \mathbb{R}$ is a payoff function, where $A = A_1 x ... x A_n$

A Nash equilibrium can either have a pure strategy, which provides a complete definition of how a player will play a game, or a mixed strategy, that is a randomization over a set of pure strategies. A mixed strategy set for player i is the set of probability distributions over the action set A_i described by the simplex operator $\triangle$.

$$\triangle(A_i) = \{ q_i : A_i \rightarrow [0,1] \mid \sum_{i=0} q_i(a_i) = 1 \} \equiv Q_i$$

for mixed strategies:

$$Q = \prod_i Q_i \qquad and \qquad q = (q_i, q_{-i}) \epsilon Q$$

Expected payoffs to player i from strategy profile q in mixed strategies are:

$$\mathbb{E}_{a \sim q}[R_i(a)] = \sum_{a \epsilon A} q(a) R_i(a)$$
$$\text{where} \quad q(a) = \prod_{j=1}^{N} q_j(a_j)$$

The essential meaning of a mixed strategy is that randomized actions can achieve a better average payoff, and the equilibrium in mixed strategies is associated with the probability distribution over the set of actions, where this equilibrium can be achieved. In a mixed strategy, actions are performed randomly according to the probability distribution function.

We do believe that such a framework is a solid building block for building better security mechanisms. For instance, honeypots and security monitoring solutions could be the first target frameworks. Intuitively, the fact that attackers connecting to the can be seen as game between the honeypot and the attackers. We assume that attackers try to achieve their goal as fast as possible. Hence they try to minimize the number of interactions with the honeypot and the honeypot aims either to maximize the number of interactions, to learn as much as possible from attackers or to distract them as long as possible from critical resources. Several strategies can be defined for both the honeypot and the attacker. Attackers normally follow a fixed goal while penetrating an honeypot. Some attackers want to hide their connections to bot nets via compromised hosts, other attackers want to scan other hosts or they want to share illegal content via the honeypot. In the most cases attackers find out that the honeypotnot immediately ready for their malicious activities. Thus, they acquire their tools, install them and execute them. On the one hand, they download tar balls containing a pre-compiled version of their program and on the other hand they download the source code of their program that they compile on the honeypot. In both cases attackers often configure their programs on the honeypot. All these actions results in interactions with the honeypot and can be thus integrated in the definition of strategies. The honeypot aims at revealing as much as possible about the attacker. This information might include the location, tools, repositories and psychological profile of an attacker. There are different games that will be researched on: starting with simple zero sum payoff games, and ending up with more advanced games, where irrational behavior and bluffing tactics are deployed by the honeypot. We have already performed a preliminary work, which is described in [132]. In that paper, we have proposed and validated a honeypot that is configured using game theoretical concepts. The honeypot will block the execution of a system call, while the attacker can try to continue, quit or choose another target goal. Our first experiments are done on a real operational environment and the the results are promising.

A second challenge that I will address is the development of a malware analysis lab. Most malware is not analyzed now by academic actors and unfortunately many academic and research sourced approaches are more than needed. The recent rise in the rapid creation and deployment cycles for Malware, required adapted response from the security community. Malware, ranging from worms, viruses, bots, trojans and rootkits form one of the major threats in the current security and internet landscape. Criminal groups develop highly efficient malware which is used in spam, end user identity stealing, phishing, cybercrime and high staking industrial and political espionage. From a research point of view, some major challenges must be addressed in order to fight these threats. One of the challenges is related to the physical localization of the command and control structure. In a traditional malware scenario, a compromised machine is becoming remotely manageable by an attacker. Many (up to ten of thousands) of compromised machines can be commanded and controlled over a communication middleware. The well known cases, use the traditional IRC (Internet Relay Chat) based logical multicast server architecture. The recent evolution however shows that a shift towards P2P type communication middleware, where encryption and onion-routing schemes allow an attacker not to be tracked and identified by network level monitoring. Tracking the command and control channel in such an environment is on my research agenda. I intent to use timing related information from multiple honeyclients (limited bots, capable to join a botnet) in order to locate the source of the commands. The main idea is to use IP level triangulation techniques, where timing information is however measured on

an overlay network. Within this research theme, very few works exist yet and no relevant state of the art can be mentioned. Most literature addressed the issue of either tracking IRC based botnets (via high interaction honeypots) or by deploying low interaction virtualized honeypots, where only limited information is gathered. A second issue that makes this tracking difficult is the fact that commands are protected by strong cryptography. Although direct cryptographic attacks are not an option, it is viable to construct replay and fingerprinting software that can identify and emulate a malware communication channel. The most similar work in this are is the roleplayer approach [64]. The key idea is to use algorithms (multiple sequence alignment, hidden markov based models) that have proved their efficiency in the analysis of large gene structures in order to learn and simulate unknown traffic. For instance, this would make it possible to deploy on the fly a smart honeypot client, capable to join an unknown botnet and lure the attacker into revealing its location. Current malware poses important challenges for the analyst. Heavily armed by anti-reverse and anti-debugging mechanisms, using sophisticated obfuscation techniques, the malware of today comes is many variants and flavors. It is known that malware authors create new and extended malware based on pieces of existing malware (copy/cut process) and by adding more and extended functionalities. With respect to the state of the current state of the art, we will not rely on undoing obfuscation (done for instance by Mihai Christodorescu [52], (2007 the university of Wisconsin), USA) or on graph based polymorphism detection in malware [44] . Similar to our work is the approach followed by [94] and [125], where state machines are used to model behavior. In our approach, behavior is not modeled by an automata, but rather a statistical profile of the system calls. This is somehow similar to work done by the work of Lee [126] and C. Kruegel [87] for spyware analysis. The differences are given by the granularity level of the code analysis as well as the underlying conceptual approach. In our vision, the virtualized and safe execution of a malware is required to determine the profile and many fundamental approaches for building the notion of a profile are yet to be developed. I have done prior work (joint work with Gerard Wagener and Alexandre Dulaunoy) [131] on automated classification and analysis of malware. Our work is based on extracting a profile of the underlying behavior and classifying an unknown malware based on this profile. The extracted profile is obtained by recovering all system calls, and information theoretic measures are used for the classification. Our work addressed also the case of armored malware, where strong protection and anti-reverse engineering is used. We proposed an analysis method, where a virtualized and customized environment allows to learn the internal behavior of a piece of malware. This work needs to be extended towards more innovative analysis algorithms (for instance coding theory can error correction codes might be appropriate) and automated on demand virtual environments, that can be spawned on demand in order to analyze a piece of malware. The latter area of work will allow to leverage virtualized environments for capturing, analyzing and classifying unknown malware. The interest of such a work is to detect new offsprings (of an existing malware) or new and previously unknown pieces of malware. For this work, I look forward to cooperate with local actors in the larger region.

6.2.2 Research in VoIP Intrusion Detection

If we consider the future work that has to be addressed, the most important topic for my research is related to the automated detection and prevention in VoIP infrastructures. Right now, existing IDS type of solutions are not suited for VoIP traffic. Signature based detection methods work well for network level monitoring, where attacks are spread in most cases in one interaction. However, in VoIP, most of the vulnerabilities are related to poor protocol/state tracking. In these scenarios, an attack is spread over a large quantity of messages, such that all of them must be processed and tracked. The challenges in this activity are related to the design of signature

language allowing to express vulnerabilities/signatures spread over multiple messages as well as the automated implementation of firewalls/IDS for this purpose. The latter activity will address the algorithmic aspects related to the design of highly efficient traffic tracking software such that inline high speed monitoring can be done. On a longer time run, I intent to work on the development of automatic application level protection/fingerprinting systems, that can use a protocol specification (state machine + message format) in order to generate dedicated application level monitoring infrastructures. Such infrastructures can be firewalls, intrusion detection systems or passive fingerprinting devices.

We will continue the research related to the automated detection and prevention in VoIP infrastructures. Right now, existing IDS type of solutions are not suited for VoIP traffic. Signature based detection methods work well for network level monitoring, where attacks are spread in most cases in one interaction. However, in VoIP, most of the vulnerabilities are related to poor protocol/state tracking. In these scenarios, an attack is spread over a large quantity of messages, such that all of them must be processed and tracked. The challenges in this activity are related to the design of signature language allowing to express vulnerabilities/signatures spread over multiple messages as well as the automated implementation of firewalls/IDS for this purpose. The latter activity will address the algorithmic aspects related to the design of highly efficient traffic tracking software such that inline high speed monitoring can be done. On a longer time run, I intent to work on the development of automatic application level protection/fingerprinting systems, that can use a protocol specification (state machine + message format) in order to generate dedicated application level monitoring infrastructures. Such infrastructures can be firewalls, intrusion detection systems or passive fingerprinting devices.

6.2.3 Research in protocol fuzzing

My main objectives are to propose a formal framework and self-adapting paradigms for protocol fuzzing. Developing the formal framework is a long run activity which aims at introducing concepts like coverage, efficiency and computational complexity to fuzzing. The underlying idea would be that given a protocol specification, the framework should be able to give an algorithmic theoretical complexity of fuzzing as well as probabilistic bounds on the quality of the results. The six main research directions that we plan to address are the following:

1. Given a fuzzer, how can we measure the impact and the coverage of its actions ?
2. When should a fuzzing process stop ?
3. Given several fuzzers, how can we assess the performance of each of them ?
4. How can be leverage several fuzzers in order to build a comprehensive set of test cases that is optimal with respect to the impact ?
5. How can be fine tune the fuzzing in order to detect at a higher granularity the input data that can identify a vulnerability ?
6. Can we develop fuzzing approaches that are able to learn from experience ?

The notion of coverage is relatively well understood in the software testing community, has though a complex counterpart in the world of fuzzing. There are several possible ways to to define and measure coverage. In a blackbox approach, the coverage could take into account the possible ranges of individual input fields. In a greybox approach, where system level tracing is possible, the coverage could be defined in terms of control flow execution paths and memory

access. In order to instrument a greybox fuzzing approach, some system specific tracing is required. This tracing should provide at least the set of memory related operations and control flow execution paths. We intend to couple a tracing based approach with a tainted data injection method in order to link specific fields of the input data to associated memory locations and code execution addresses. The rationale is to know for each part of the input data where it gets treated in the control flow graph and what memory locations will hold content that is derived from it. These pieces of information will enable to define the impact of a fuzzing process. One direct consequence of such a coverage is to define stopping rules for a fuzzing process. A rule of thumb would be to stop whenever no new code execution paths are traversed. There is significant research work that must be performed in order to cast the stopping process in terms of a stochastic process/martingale for which optimal stopping rules can be proven. A somehow related question concerns the evaluation of several fuzzers. This question is important in a context where several fuzzing framework coexist and no qualitative assessment for them has been done yet. From a practical perspective, users are not able to identify the best and most accurate fuzzer among them. We will address this issue and propose both metrics as well as an assessment framework for comparing fuzzers. We expect that fuzzers will show very different behaviors. Some will probably cover a large part of the underlying application related control flow graph. Such a behavior could be using only few different values to test a logical condition and branch code. Other fuzzing frameworks might reveal a more localized behavior, a subset of the control flow graph will be heavily tested with a large quantity of different data values. Therefore, we will consider a multi-fuzzer architecture, where a battery of tests is generated using multiple fuzzers. The key challenge is to combine the fuzzers such that an optimal coverage of the control flow graph is done. This would mean that localized fuzzing will be mixed with more depth driven fuzzing. The expected result is to have a homogenous coverage of the control flow graph. This process is somehow similar to the use of multiple classifiers in machine learning. We expect that "cooperative fuzzing" to be a fundamental building block for providing high quality test cases. One important issue that needs to be addressed is related to the tuning of a fuzzing environment. Mutation based fuzzers do already rely on prior input data that will be mutated and used for fuzzing. It is crucial to improve this simple scheme with a tuning phase that will mutate only a subpart of the input. This subpart should be considered with respect to resulted impact (measured in terms of coverage). We will address this research topic by identifying the relevant subpart responsible for higher coverage. We will refine the mutation process by leveraging the identified subparts. One final topic of interest consists in developing a framework for smarter fuzzing. The concept of smart fuzzing is strongly dependent on having intrinsic machine learning capabilities within the fuzzing framework. For this last question, I will also consider a modeling framework based on the game theory and machine learning. Game theory is a modeling enabler since the interactions between an fuzzer and a device under test can be seen like a game. Defining the type of the game and the associated reward functions is a challenging issue. One idea that we will investigate consists on evolutionary game models, where different fuzzing strategies will drive a population of fuzzers. This framework has the advantage of assuring that discounted payoffs can be taken into account and that learning via repetitive interactions is possible. The hard question that will need to be solved consists in solving such games, where the tuple action/payoff value is huge. One promising idea is to rely on approximate dynamic programming, where explicit state enumeration is not required. Secondly, I intend to working towards self-adapting fuzzing paradigms able to deal with complex stateful protocols and tune the fuzzing process with respect to the observed behavior. The KIF fuzzer developed over the past year by us is a very promising result, showing that a dynamical self-learning process - where basic fuzzing actions are composed into complex fuzzing tests in a semi-random and closed loop guided learning automation - is the key ingredient for a robust and highly efficient solution.

The underlying learning techniques are still basic and more research work is needed in order to develop learning algorithms for such purposes. My main work will consist in combining online learning algorithms and optimal decision processes. With respect to this research agenda, we will consider reinforcement learning and evolutionary computation.

6.2.4 Research in Protocol fingerprinting

We will continue to do research on service and network fingerprinting by integrating semi-supervised learning approaches. This learning paradigm focuses on data sets with only a small amount of labeled data and a lot of unlabeled data samples. Methods like *k-means*, a traditional clustering method assume that adjacent data samples have tendency to similar labels, with result to propagate their labels to unlabeled data samples.

Semi-supervised clustering was introduced in [141] in order to leverage the knowledge about a small subset from the data. This knowledge is given by the mapping between one data item and its corresponding cluster identifier. The objective of a semi-supervised clustering algorithm is to use this information and an additional distance function in order to classify the remaining data items. This approach lies in-between the supervised clustering technique (where learning algorithms are applied in a supervised learning framework) and the unsupervised approach, where only distances between individual data items are driving the clustering process. The main idea in semi-structured clustering is to construct a fully connected graph, where some nodes are labeled. This label represents the class name of the node. We assume to have C different classes. Each class is initially represented by a flow that was tapped at the tagging server. The edges between two nodes have weights associated. A weight is dependent on the distance between the two nodes. Iteratively, all the labels are propagated to unlabeled regions in the fully connected graph. The nodes in this graph are either labeled or unlabeled data samples. Nodes that are not labeled, will iteratively estimate the probabilities f belonging to each class. At the end of the interactions, an unlabeled node will be allocated to the most probable class . The semi-supervised clustering algorithm is also known as the *label propagation* algorithm and operates as follows:

We have a set composed of labeled data (x_1, y_1),...,(x_l, y_l) and unlabeled data (x_{l+1}, y_{l+1}),...,(x_{l+u}, y_{l+u}) with l$\ll$u, where $Y_L = \{y_1, ..., y_l\}$ are the class labels of the labeled data and $Y_U = \{y_l+1, ...y_l+u\}$ unobserved yet. It is assumed that the number of classes C is known and that all classes are in the labeled data samples [141]. Let $X=\{x_1, ..., x_{l+u}\}$ be the different data items x_i. We want to estimate the class labels of the unlabeled samples Y_U from the data items X and their class labels Y_L. This is done using a distance function. If two cluster tuples (x_i, y_i) and (x_j, y_j) have to be compared (y_i respectively y_j represent the class labels), the distance functions $d(i, j)$ measures the difference between the two data items x_i and x_j . The performance of the semi-clustering algorithm is dependent on this distance function. A good distance function should capture the essential differences in the structures of the data items. We will consider in this work, data items that are represented in structured input spaces like labeled trees and finite state machines and where good distance functions are essential.

The labeled and unlabeled data samples are represented in a fully connected graph, where the edge between node i, j is weighted. The edge weight w_{ij} is given by the following expression:

$$w_{ij} = exp(-\frac{d_{ij}^2}{2}) \tag{6.1}$$

Node labels are propagated through the edges to all other nodes. As in [141] we define a $(l + u)\times$ $(l + u)$ transition matrix T, where T_{ij} gives the probability to jump from node i to j.

$$T_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{jk}} \tag{6.2}$$

We define a $(l+u) \times C$ label matrix Y, where a row reflects the label probability distribution of a node. The element Y_{ic} is the probability that element Y_i belongs to the class c. Initially this probabilities are initialized with $1/C$ for the unlabeled items.

The label propagation algorithm by [141] has three different steps.

1. Propagate Y $\leftarrow$ TY
   For one step, all nodes propagate their labels.

2. Row normalization of Y
   This maintains a probability distribution.

3. Clamping of labeled data
   We clamp the label distribution of labeled data to Y_{ic}=1 if item Y_i had an initial label of c. This step assures that initial labels are maintained.

The previous steps are repeated from *step*1 to Y, until Y converges. It has been shown in [141] that the previous algorithms does always converge. I will research the potential of semi-supervised clustering for both syntactic fingerprinting as well as temporal and behavioral fingerprinting. The work will address the specification of proper and suitable similarity functions as well as optimal settings of the clustering algorithm. Somehow related to this work, I will also evaluate the performance of alternative clustering algorithms, like for instance ROCK [70]. This new kind of unsupervised approach is dedicated to categorical and leverages a graph representation where two nodes are linked if they share at least one common neighbor. Two points are neighbors if their inter-distance is less than a threshold τ. The overall process is in fact an agglomerative clustering technique, where each unique point is a cluster at the beginning. Clusters are grouped together based on a score measure which measures the linkage degree (the number of shared neighbors) comparing with the estimation of the maximal number of possible shared neighbors. The main advantage is the discovery of irregular shapes of cluster in the data.

Résumé

Ce manuscrit synthétise les activités de recherche que nous avons menées, au cours des ces dernières années dans le domaine de la gestion de réseaux, et contient une présentation des enjeux pour la décénie venir. Il contient également un projet de recherche ambitieux visant á répondre aux défis de la sécurité de l'Internet du futur. Nous présentons nos travaux sur l'audit de sécurité en abordant le problème du "fingerprinting", c'est à dire la détection par prise d'empreintes d'une couche protocolaire et/ou d'un équipement. La prise d'empreintes d'un système est une action essentielle dans l'audit de sécurité d'un réseau. L'objectif de ce processus consiste dans la détection d'une version spécifique d'un service/équipement par l'analyse du trafic véhiculé sur le réseau. Nous avons elaboré deux approches de "fingerprinting". La première s'appuie sur l'analyse des arbres d'analyse syntactique pour les messages d'un protocole. Nous abordons ensuite la problématique du "fingerprinting" pour le cas des protocoles dont nous ne disposons pas de spécifications. L'approche que nous présentons est capable d'inférer les divers types de messages et de construire une (ou plusieurs) machine d'états. Ces machines d'états sont les fondements pour définir le comportement. Nous proposons à la suite une approche de fingerprinting qui permet la prise en compte du comportement d'une souche protocolaire. Nous développons ensuite la problématique liée à la surveillance de sécurité d'un réseau. Celle-ci comprend deux parties principales: le monitoring pour la détection d'intrusions et un pot de miel pour la VoIP. La dernière contribution est une approche pro-active élevée pour la détection de failles de sécurité par un processus de type "fuzzing".

Mots-clés: audit de sécurité, VoIP, machines à support de vecteurs

Abstract

This document describes my main research activities, performed over the pas few years. It starts with a summarized historical overview of my activities and continues with detailed content on a selection of topics. The document starts with syntax driven approaches for service and network fingerprinting scheme, showing how parse trees of captured messages are serving to learn distinctive features capable to perform fingerprinting. We consider next the case of unknown protocols and propose an unsupervised learning method based on support vector clustering - SCV. The follow-up part of the document considers the behavioral fingerprinting, based on the analysis of temporal and state machine induced feature. We introduce the TR-FSM, a tree structured parametrized finite state machine having time annotated edges. A TR-FSM represents a fingerprint for device/stack. Several such fingerprints are associated with a device. We propose a supervised learning method, where support vector machines do use kernel functions defined over the space of TR-FSMs. We validated our approach using SIP as a target protocol. We address also the security monitoring of VoIP and present new monitoring approaches for VoIP specific environments. We address next the the practical outcomes of our fuzzing approach. We summarize the fuzzing architecture and give an overview on some of the most surprising vulnerabilities that we have found. We present a short positioning of our work with respect to relevant ongoing international activities in the sixth chapter. The final chapter of this manuscript concludes and points out the future activities to be undertaken.

Keywords: fingerprinting, security assessment, VoIP, support vector machines

Bibliography

[1] Avispa project. http://www.avispa-project.org.

[2] A dns rr for specifying the location of services (dns srv). http://www.ietf.org/rfc/rfc2782.txt.

[3] DParser. http://dparser.sourceforge.net/.

[4] The e.164 to uniform resource identifiers (uri) dynamic delegation discovery system (ddds) application (enum). http://www.ietf.org/rfc/rfc3761.txt.

[5] FreePBX: full-featured PBX web application. http://freepbx.org.

[6] MPack: Insight into MPACK Hacker kit . http://www.malwarehelp.org/news/article-6268.html/.

[7] Nmap. http://www.insecure.org/nmap/.

[8] P0f. http://lcamtuf.coredump.cx/p0f.shtml.

[9] The Asterisk PBX. http://www.asterisk.org/.

[10] The Voice over IP Security Alliance (VOIPSA). http://www.voipsa.org/Activities/taxonomy.php.

[11] trixbox: Asterisk-based IP-PBX . http://www.trixbox.com/.

[12] Integrated Network Management, IM 2005. 9th IFIP/IEEE International Symposium on Integrated Network Management, 15-19 May 2005, Nice, France. IEEE, 2005.

[13] H. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP Authentication. In Information Assurance and Security (ISIAS) Information Assurance and Security, 2008. ISIAS '08., pages 237–242, Naples Italy, 2008. IEEE.

[14] H. Abdelnur, R. State, I. Chrisment, and C. Popi. Assessing the security of voip services. In Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, pages 373–382, 21 2007-Yearly 25 2007.

[15] H. Abdelnur, R. State, and O. Festor. Security Advisory: SIP Digest Access Authentication RELAY-ATTACK for Toll-Fraud. http://voipsa.org/pipermail/voipsec_voipsa.org/2007-November/002475.html.

[16] H. Abdelnur, R. State, and O. Festor. Security Advisory: SQL injection in asterisk-addons and XSS injection in WWW application in Areski, FreePBX and Trixbox. http://voipsa.org/pipermail/voipsec_voipsa.org/2007-October/002466.html.

[17] H. Abdelnur, R. State, and O. Festor. Fuzzing for vulnerabilities in the VoIP space. In 17th EICAR Annual Conference 17th EICAR Annual Conference, Laval France, 2008. European Institute for Computer Antivirus Research.

[18] H. J. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP Authentication. In IAS '08: Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security, pages 237–242, Washington, USA, 2008. IEEE Computer Society.

[19] H. J. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP Authentication. Journal of Information Assurance and Security (JIAS), 2009 to appear.

[20] H. J. Abdelnur, R. State, and O. Festor. KiF: a stateful SIP fuzzer. In IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications, pages 47–56, New York, USA, 2007. ACM.

[21] H. J. Abdelnur, R. State, and O. Festor. Advanced Network Fingerprinting. In RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 372–389, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] H. J. Abdelnur, R. State, and O. Festor. Failles et VoIP. In MISC Magazine - Edition française: Multi-System & Internet Security Cookbook. Misc #39, Septembre/Octobre 2008.

[23] H. J. Abdelnur, R. State, and O. Festor. Fuzzing for vulnerabilities in the VoIP space. In EICAR '08: 17th Annual Conference of the European Institute for Computer Anti-Virus Research, Laval France, 2008.

[24] H. J. Abdelnur, R. State, and O. Festor. Fuzzing for vulnerabilities in the VoIP space. ACM International Journal of Computer Virology, 2009 to appear.

[25] D. Aitel. The Advantages of Block-Based Protocol Analysis for Security Testing. Immunity Inc, `http://www.immunitysec.com/resources-papers.shtml`, Feb. 2002.

[26] D. Aitel. MSRPC Fuzzing with SPIKE 2006. Immunity Inc, Aug. 2006.

[27] D. Angluin. Learning regular sets from queries and counterexamples. Inf. Comput., 75(2):87–106, 1987.

[28] R. Badonnel, R. State, and O. Festor. Management of mobile ad-hoc networks: evaluating the network behavior. In Integrated Network Management [12], pages 17–30.

[29] R. Badonnel, R. State, and O. Festor. Management of mobile ad hoc networks: information model and probe-based architecture. Int. Journal of Network Management, 15(5):335–347, 2005.

[30] R. Badonnel, R. State, and O. Festor. Fault monitoring in ad-hoc networks based on information theory. In F. Boavida, T. Plagemann, B. Stiller, C. Westphal, and E. Monteiro, editors, Networking, volume 3976 of Lecture Notes in Computer Science, pages 427–438. Springer, 2006.

[31] R. Badonnel, R. State, and O. Festor. Probabilistic management of ad-hoc networks. In Hellerstein and Stiller [80], pages 339–350.

[32] R. Badonnel, R. State, and O. Festor. A probabilistic approach for managing mobile ad-hoc networks. IEEE Transactions on Network and Service Management, 4(7), 2007.

[33] R. Badonnel, R. State, and O. Festor. Self-configurable fault monitoring in ad-hoc networks. Ad Hoc Networks, 6(3):458–473, 2008.

[34] R. Badonnel, R. State, O. Festor, and A. Schaff. A framework for optimizing end-to-end connectivity degree in mobile ad-hoc networks. J. Network Syst. Manage., 13(4):479–497, 2005.

[35] V. A. Balasubramaniyan, M. Ahamad, and H. Park. CallRank: Combating SPIT using call duration, social networks and global reputation. In Fourth Conference on Email and Anti-Spam (CEAS2007), Mountain View, California USA, 2007.

[36] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics, 16(5):412–24, 2000.

[37] G. Banks, M. Cova, V. Felmetsger, K. C. Almeroth, R. A. Kemmerer, and G. Vigna. SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZEr. In of Lecture Notes in Computer Science, pages 343–358. Springer, 2006.

[38] S. Becker, R. State, and T. Engel. Using game theory to configure p2p sip. In Proceedings of the 3rd international conference on Principles, systems and applications of IP telecommunications (IPTComm '09), New York, NY, USA, 2009. ACM.

[39] M. Beddoe. Protocol informatics, http://www.4tphi.net (accessed on 02/05/09).

[40] A. Ben-hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. Journal of Machine Learning Research, 2:125–137, 2001.

[41] P. Berkhin. A survey of clustering data mining techniques. In Grouping Multidimensional Data, pages 25–71. Springer, 2006.

[42] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: when randomness plays with you. SIGCOMM Comput. Commun. Rev., 37(4):37–48, 2007.

[43] A. Broder. On the Resemblance and Containment of Documents. In SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997, page 21, Washington, USA, 1997. IEEE Computer Society.

[44] D. Bruschi, L. Martignoni, and M. Monga. Recognizing self-mutating malware by code normalization and control-flow graph analysis. IEEE Security & Privacy, 2007. in press.

[45] L. Butti and J. Tinnes. Discovering and exploiting 802.11 wireless vulnerabilities. Journal in Computer Virology, 4(1):25–37, February 2008.

[46] D. Buttler. A Short Survey of Document Structure Similarity Algorithms. In The 5th International Conference on Internet Computing, june 2005.

[47] J. Caballero, S. Venkataraman, P. Poosankam, M. G. Kang, D. Song, and A. Blum. FiG: Automatic Fingerprint Generation. In The 14th Annual Network & Distributed System Security Conference (NDSS 2007), Feb. 2007.

[48] J. Caballero, H. Yin, Z. Liang, and D. Song. Polyglot: automatic extraction of protocol message format using dynamic binary analysis. In CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pages 317–329, New York, NY, USA, 2007. ACM.

[49] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[50] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[51] E. Chen. Detecting DoS attacks on SIP systems. In Proceedings of 1st IEEE Workshop on VoIP Management and Security, pages 53–58, San Diego, CA, USA, apr 2006.

[52] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pages 5–14, New York, NY, USA, 2007. ACM Press.

[53] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In ACL02, 2002.

[54] C. Cortes and V. Vapnik. Support-vector networks. Machine Learning, 20(3):273–297, 1995.

[55] V. Cridlig, H. J. Abdelnur, R. State, and O. Festor. A voip security management architecture. In Hellerstein and Stiller [80].

[56] V. Cridlig, H. J. Abdelnur, R. State, and O. Festor. Xbgp-man: an xml management architecture for bgp. Int. Journal of Network Management, 16(4):295–309, 2006.

[57] V. Cridlig, O. Festor, and R. State. Role-based access control for xml enabled management gateways. In A. Sahai and F. Wu, editors, DSOM, volume 3278 of Lecture Notes in Computer Science, pages 183–195. Springer, 2004.

[58] V. Cridlig, R. State, and O. Festor. An integrated security framework for xml based management. In Integrated Network Management [12], pages 587–600.

[59] V. Cridlig, R. State, and O. Festor. Role-based access control for xml enabled multi-protocol management gateways. IEEE Transactions on Network and Service Management, 2(1), 2006.

[60] V. Cridlig, R. State, and O. Festor. A model for checking consistency in access control policies for network management. In Integrated Network Management, pages 11–19. IEEE, 2007.

[61] M. Crispin. Internet Message Access Protocol - Version 4rev1. RFC 3501 (Proposed Standard), Mar. 2003. Updated by RFCs 4466, 4469, 4551, 5032, 5182.

[62] N. Cristianini and J. Shawe-Taylor. An introduction to support Vector Machines: and other kernel-based learning methods. Cambridge University Press, New York, USA, 2000.

[63] D. H. Crocker and P. Overell. Augmented BNF for Syntax Specifications: ABNF, 1997.

[64] W. Cui, V. Paxson, N. Weaver, and R. H. Katz. Protocol-Independent Adaptive Replay of Application Dialog. In NDSS. The Internet Society, 2006.

[65] W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irun-Briz. Tupni: automatic reverse engineering of input formats. In CCS '08: Proceedings of the 15th ACM conference on Computer and communications security, pages 391–402, New York, NY, USA, 2008. ACM.

[66] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification, 1(1):7–24, December 1984.

[67] R. Debnath, N. Takahide, and H. Takahashi. A decision based one-against-one method for multi-class support vector machine. Pattern Anal. Appl., 7(2):164–175, 2004.

[68] D. E. Denning. An intrusion-detection model. In IEEE Symposium on Security and Privacy, pages 118–133. IEEE Computer Society Press, Apr 1986.

[69] Douglas Comer and John C. Lin. Probing TCP implementations. In USENIX Summer, pages 245–255, 1994.

[70] M. Dutta, A. Mahanta, and A. Pujari. Qrock: A quick version of the rock algorithm for clustering of categorical data. 26(15):2364–2373, November 2005.

[71] S. Embleton, S. Sparks, and R. Cunningham. Sidewinder: An Evolutionary Guidance System for Malicious Input Crafting. In Black Hat 2007, Las Vegas, USA, Aug. 2006.

[72] D. Endler and M. Collier. Hacking Exposed VoIP: Voice Over IP Security Secrets and Solutions. McGraw-Hill Professional Publishing, 2007.

[73] D. Endler, D. Ghosal, R. Jafari, A. K. M. Kolenko, N. Nguyen, W. Walkoe, and J. Zar. VoIP Security and Privacy Threat Taxonomy. VOIPSA. `http://voipsa.org/Activities/taxonomy.php`, Oct. 2005.

[74] J. Fiedler, t. Kupka, S. Ehlert, T. Magedanz, and D. Sisalem. VoIP defender: Highly scalable SIP-based security architecture. In Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07), New York, NY, USA, 2007. ACM.

[75] S. Fogie, J. Grossman, R. Hansen, A. Rager, and P. D. Petkov. XSS Exploits: Cross Site Scripting Attacks and Defense. Syngress, 2007.

[76] J. Francois, H. J. Abdelnur, R. State, and O. Festor. Behavioral Fingerprinting . In RAID '09: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2009.

[77] K. Gopalratnam, S. Basu, J. Dunagan, and H. J. Wang. Automatically Extracting Fields from Unknown Network Protocols. In Systems and Machine Learning Workshop 2006, Saint-Malo, France, June 2006.

[78] K. Gopalratnam, S. Basu, J. Dunagan, and H. J. Wang. Automatically extracting fields from unknown network protocols, June 2006.

[79] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In S. Sen, C. Ji, D. Saha, and J. McCloskey, editors, MineNet, pages 197–202. ACM, 2005.

[80] J. L. Hellerstein and B. Stiller, editors. Management of Integrated End-to-End Communications and Services, 10th IEEE/IFIP Network Operations and Management Symposium, NOMS 2006, Vancouver, Canada, April 3-7, 2006. Proceedings. IEEE, 2006.

[81] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on, 13(2):415–425, Mar 2002.

[82] A. B. Johnston and D. M. Piscitello. Understanding Voice over Ip Security. Artech, 2006.

[83] R. Kaksonen. "A Functional Method for Assessing Protocol Implementation Security", Licentiate Thesis. VTT Publications 447. ISBN 951-38-5873-1, 2001.

[84] H. J. Kang, Z.-L. Zhang, S. Ranjan, and A. Nucci. Sip-based voip traffic behavior profiling and its applications. In MineNet '07: Proceedings of the 3rd annual ACM workshop on Mining network data, pages 39–44, New York, NY, USA, 2007. ACM.

[85] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), Apr. 2001. Obsoleted by RFC 5321, updated by RFC 5336.

[86] P. Kolan and R. Dantu. Socio-technical defense against voice spamming. ACM Trans. Auton. Adapt. Syst., 2(1):Article 2, 2007.

[87] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Automating mimicry attacks using static binary analysis. In SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium, pages 11–11, Berkeley, CA, USA, 2005. USENIX Association.

[88] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In SAC '02: Proceedings of the 2002 ACM symposium on Applied computing, pages 201–208, New York, NY, USA, 2002. ACM Press.

[89] A. Lahmadi and O. Festor. SecSip: A Stateful Firewall for SIP-based Networks. In 11th IFIP/IEEE International Symposium on Integrated Network Management - IM 2009, Long Island États-Unis d'Amérique, 2009. D.: Software, K.: Computing Milieux, K.: Computing Milieux/K.6: MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS/K.6.5: Security and Protection.

[90] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, and X. Yin. A Formal Approach for Passive Testing of Protocol Data Portions. In ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols, pages 122–131, Paris, France, 2002. IEEE Computer Society.

[91] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. Computer Security Applications Conference, Annual, 0:203–214, 2005.

[92] Z. Lin, X. Jiang, D. Xu, and X. Zhang. Automatic protocol format reverse engineering through conectect-aware monitored execution. In 15th Symposium on Network and Distributed System Security (NDSS), 2008.

[93] D. Litchfield, C. Anley, J. Heasman, and B. Grindlay. The Database Hacker's Handbook: Defending Database Servers. John Wiley & Sons, 2005.

[94] A. M. M. Swimmer, B. Le Charlier. Dynamic detection and classification of computer viruses using general behavior patterns. In Proceedings of the 5th International Virus Bulletin Conference, pages 75–88, 1995.

[95] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In J. M. Almeida, V. A. F. Almeida, and P. Barford, editors, Internet Measurement Conference, pages 313–326. ACM, 2006.

[96] G. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and application protocol scrubbing. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages 1381–1390 vol.3, Mar 2000.

[97] M. Maloof. Machine Learning and Data Mining for Computer Security: Methods and Applications. Springer, 2005.

[98] P. M. Maurer. "Generating Test Data with Enhanced Context-Free Grammars". IEEE Softw., 7(4):50–55, 1990.

[99] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In ECML: Proceedings of the 17th European Conference on Machine Learning, 2006.

[100] A. Moschitti. Making tree kernels practical for natural language learning. In Proceedings of the Eleventh International Conference on European Association for Computational Linguistics, 2006.

[101] A. Moschitti, D. Pighin, , and R. Basili. Tree kernel engineering for proposition re-ranking. In Proceedings of Mining and Learning with Graphs (MLG 2006), 2006.

[102] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection: Support vector machines and neural networks. The IEEE Computer Society Student Magazine, 10(2), 2002.

[103] J. F. Nash. Non-Cooperative Games. The Annals of Mathematics, 54(2):286–295, 1951.

[104] M. Nassar, S. Niccolini, R. State, and T. Ewald. Holistic voip intrusion detection and prevention system. In Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07), pages 1–9, New York, NY, USA, 2007. ACM.

[105] M. Nassar, R. State, and O. Festor. Intrusion detections mechanisms for VoIP applications. In Third annual security workshop (VSW'06). ACM Press, Jun 2006.

[106] M. Nassar, R. State, and O. Festor. Voip honeypot architecture. In Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, pages 109–118, 21 2007-Yearly 25 2007.

[107] M. Nassar, R. State, and O. Festor. Monitoring SIP traffic using support vector machines. In Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID '08), pages 311–330, London, UK, 2008. Springer-Verlag.

[108] J. Newsome, D. Brumley, J. Franklin, and D. Song. Replayer: automatic protocol replay by binary analysis. In CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, pages 311–321, New York, NY, USA, 2006. ACM.

[109] S. Niccolini, R. Garroppo, S. Giordano, G. Risi, and S. Ventura. SIP intrusion detection and prevention: recommendations and prototype implementation. In VoIP Management and Security, 2006. 1st IEEE Workshop on, pages 47–52, April 2006.

[110] P. Ning and S. Jajodia. Intrusion Detection in Distributed Systems: An Abstraction-Based Approach. Springer, 2003.

[111] T. Porter. Practical VoIP Security. Syngress Publishing, 800 Hingham Street, Rockland, MA 02370, March 2006.

[112] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald. Detecting SPIT calls by checking communication patterns. In IEEE International Conference on Communications (ICC 2007), Jun 2007.

[113] B. Reynolds and D. Ghosal. Secure IP Telephony using Multi-layered Protection. In Proceedings of The 10th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, feb 2003.

[114] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing, pages 411–420, New York, NY, USA, 1989. ACM.

[115] U. Roedig, R. Ackermann, and R. Steinmetz. Evaluating and improving firewalls for ip-telephony environments. In 1st IP telephony workshop, Berlin, Germany, April 2000.

[116] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002.

[117] R. E. Schapire. Diversity-based inference of finite automata. Technical report, Cambridge, MA, USA, 1988.

[118] H. Scholz. SIP Stack Fingerprinting and Stack Difference Attacks. Black Hat Briefings, 2006.

[119] H. Sengar, R. Dantu, and D. Wijesekera. Securing VoIP and PSTN from integrated signaling network vulnerabilities. In 1st IEEE workshop on VoIP Management and Security (VoIP MaSe), Vancouver, Canada, April 2006.

[120] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia. Detecting VoIP Floods using the Hellinger Distance. Transactions on Parallel and Distributed Systems : Accepted for future publication, sep 2007.

[121] D. Shin and C. Shim. Progressive multi gray-leveling: A voice Spam protection algorithm. IEEE Network, 20(5):18–24, Sep/Oct 2006.

[122] G. Shu and D. Lee. Network Protocol System Fingerprinting - A Formal Approach. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pages 1–12, Apr. 2006.

[123] M. Sutton, A. Greene, and P. Amini. Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley Professional, 2007.

[124] tcpdump. http://www.tcpdump.org/ (accessed on 02/05/09).

[125] G. M. V. D. C. J. G. M. G. Thomason. Probabilistic trees and automata for application behavior modeling. In Proceedings of the 43rd ACM Southeast Conference, 2003.

[126] J. J. Tony Lee. Behavioral classification. In Proceedings Eicar'06, May 2006.

[127] A. Tridgell. How samba was written http://samba.org/ftp/tridge/misc/french_cafe.txt (accessed on 03/16/09).

[128] O. University. PROTOS Test-Suite: c07-sip. `http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip`, 2005.

[129] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, pages 80–92, London, UK, 2000. Springer-Verlag.

[130] S. Vishwanathan and A. Smola. Fast kernels on strings and trees. In Proceedings of Neural Information Processing Systems, 2002.

[131] G. Wagener, R. State, and A. Dulaunoy. Malware behaviour analysis. Journal in Computer Virology, 4(4):279–287, 2008.

[132] G. Wagener, R. State, A. Dulaunoy, and T. Engel. Self adaptive high interaction honeypots driven by game theory. In Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009), 2009.

[133] L. Wang, editor. Support Vector Machines: Theory and Applications, volume 177 of Studies in Fuzziness and Soft Computing. Springer, 2005.

[134] Weidong. Discoverer: Automatic protocol reverse engineering from network traces. pages 199–212.

[135] C. Wieser, M. Laakso, and H. Schulzrinne. SIP Robustness Testing for Large-Scale Use. In SOQUA/TECOS, pages 165–178, 2004.

[136] G. Wondracek, P. M. Comparetti, C. Kruegel, and E. Kirda. Automatic network protocol analysis. In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSSâ08), 2008.

[137] Y. Wu, S. Bagchi, S. Garg, N. Singh, and T. K. Tsai. SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments. In International Conference on Dependable Systems and Networks (DSN 2004), pages 433–442. IEEE Computer Society, Jun 2004.

[138] H. Yan, K. Sripanidkulchai, H. Zhang, Z.-Y. Shae, and D. Saha. Incorporating active fingerprinting into SPIT prevention systems. In Third annual security workshop (VSW'06). ACM Press, Jun 2006.

[139] H. Yan, K. Sripanidkulchai, H. Zhang, Z. yin Shae, and D. Saha. Incorporating Active Fingerprinting into SPIT Prevention Systems. Third Annual VoIP Security Workshop, June 2006.

[140] G. Zhang, S. Ehlert, T. Magedanz, and D. Sisalem. Denial of service attack and prevention on SIP VoIP infrastructures using DNS flooding. In Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07), pages 57–66, New York, NY, USA, 2007. ACM.

[141] X. Zhu and Z. Ghahraman. Learning from labeled and unlabeled data with label propagation, 2002.