Sécurité des architectures de Convergence Fixe-Mobile

Laurent Butti
laurent.butti(@)orange-ftgroup.com

Orange Labs
Laboratoire Security and Trusted Transactions
38-40 Rue du Général Leclerc
92794 Issy-les-Moulineaux Cedex 9 - France

Résumé Le monde de la téléphonie est en pleine révolution et la Convergence Fixe-Mobile (FMC) apparaît aujourd'hui incontournable. Cette profonde mutation est entreprise par de nombreux acteurs du monde des télécommunications et repose notamment sur des normes spécifiées par le 3rd Generation Partnership Project (3GPP). Les mécanismes de sécurité présents dans ces normes sont pour la plupart issus de protocoles normalisés par l'Internet Engineering Task Force (IETF) tels que Internet Key Exchange v2 (IKEv2) [1], Extensible Authentication Protocol (EAP) [2], IPsec [3] et Transport Layer Security (TLS) [4]. Ces briques sont la clé de voûte de l'accès des utilisateurs à ces architectures. L'audit de ces architectures est nécessaire et passe par le développement d'outils de fuzzing permettant l'évaluation de la qualité des implémentations[1]. Le développement de ces outils nous a permis de découvrir certaines vulnérabilités dans des implémentations propriétaires et donc de contribuer à l'amélioration de la sécurité des architectures basées sur ces implémentations.

1 Introduction

Les réseaux radioélectriques destinés aux communications personnelles et professionnelles sont en perpétuelle mutation. C'est avec la commercialisation de Radiocom 2000 [6] à la fin des années 80 que les premières notions de téléphonie cellulaire[2] sont apparues. Au début des années 90, la technologie Global System for Mobile Communications (GSM) s'est rapidement imposée[3] et compte aujourd'hui plus de trois milliards d'utilisateurs à travers le monde. Les technologies cellulaires ont ensuite rajouté du transport de données et ont largement évolué grâce à des normes comme le General Packet Radio Service (GPRS) et l'Universal Mobile Telecommunications System (UMTS). En parallèle de cette évolution des technologies radioélectriques à courte distance de type Bluetooth [9] ou Wi-Fi [10] rajoutent une nouvelle saveur à

[1] L'adoption du terme « implémenter »par la commission générale de terminologie et de néologie a été publiée au journal officiel le 20 avril 2007 [5] (source : Wikipedia).
[2] le terme provient du fait que l'émission est réalisée sur une zone de couverture bien délimitée appelée une cellule.
[3] bien qu'entre temps nous avions cherché notre tribu avec Tatoo [7] et Bi-Bop [8].

la gamme des technologies d'accès radioélectriques. Nul doute qu'elles ne seront pas les dernières avec l'arrivée des WiMAX [11,12], Long Term Evolution (LTE) [13] et consorts[4]...

Vint enfin le désir de rassembler, de trouver une technologie d'accès commune, mais indépendante de la couche sous-jacente, ou tout du moins d'être capable de se mouvoir entre les technologies pour bénéficier en permanence du meilleur accès possible pour tous les services, que ce soit pour de la voix ou des données. Le terme Convergence Fixe-Mobile fait partie de cette mouvance, de cette volonté de fédérer les technologies d'accès.

Après une brève introduction aux réseaux cellulaires et les problématiques de sécurité intrinsèques à ces derniers, l'article présentera les principales possibilités offertes aux opérateurs de téléphonie mobile (et aux nouveaux acteurs) pour déployer des architectures de Convergence Fixe-Mobile. Nous présenterons différentes architectures contribuant à la Convergence Fixe-Mobile que sont l'Unlicensed Mobile Access (UMA), l'Interworking Wireless Local Area Network (I-WLAN) et l'IP Multimedia Subsystem (IMS). Ces architectures sont pour la plupart normalisées par le 3rd Generation Partnership Project (3GPP) qui définit alors (entre autres) les mécanismes de sécurité mis en œuvre dans ces réseaux. Nous détaillerons dans un premier temps ces mécanismes de sécurité et le niveau de sécurité attendu de ces derniers.

Enfin, nous aborderons les efforts réalisés dans notre entité pour évaluer la sécurité de ces nouvelles architectures aussi bien sur le plan fonctionnel qu'au niveau des implémentations logicielles. En particulier, nous détaillerons les outils de recherche de vulnérabilité par fuzzing qui ont été développés dans ce but ainsi que quelques résultats pratiques.

Note 1. L'ensemble Convergence Fixe-Mobile pourrait s'appeler aussi Convergence Fixe-Mobile-Internet.

Note 2. Le terme anglais « fuzzing »est difficilement traduisible, nous l'utiliserons souvent en tant que mot et verbe.

2 Introduction aux architectures de Convergence Fixe-Mobile

Le succès de la téléphonie mobile n'est plus à démontrer comme en témoignent des taux de pénétration[5] de marché supérieurs à 100% dans certains pays comme

[4] beaucoup de candidats (comme par exemple iBurst normalisé sous l'IEEE 802.20) mais peu d'élus. Et dire que des publications parlent aussi de débits de plus de 10Gb/s [14]...

[5] Le taux de pénétration est obtenu en divisant le nombre total de clients ou le nombre de clients « actifs »par la population considérée (source : ARCEP).

Singapour. En France, selon les chiffres de l'ARCEP, un chiffre de plus de 90% est avancé [15]. Cette évolution a été rapide et mondiale comme l'atteste une publication récente de l'Union Internationale des Télécommunications (ITU) [16]. Par conséquent, l'utilisateur se retrouve dans une situation très courante qui est de disposer de plusieurs numéros de téléphone, de plusieurs terminaux (mobiles ou fixes) et de plusieurs abonnements. De la même manière nous assistons à une multiplication des possibilités de communication que ce soit par de la messagerie classique ou par des outils de présence comme la messagerie instantanée. Rajoutons encore une connexion à Internet par la ligne fixe (xDSL, fibre...) et il alors compréhensible que l'utilisateur final ait un souhait de convergence, à savoir de disposer d'une solution capable de fédérer plusieurs réseaux afin d'accéder à ses services de manière transparente et ergonomique. Bien entendu, les opérateurs connaissent ces besoins et voient dans la Convergence Fixe-Mobile un moyen efficace d'agréger de nombreux abonnements historiquement différents (voix fixe, voix cellulaire, données) afin de proposer le meilleur service. C'est en partant de ce constat que les technologies de Convergence Fixe-Mobile sont apparues, elles ont pour but d'apporter la connexion aux services opérateurs de manière unifiée à l'utilisateur final.

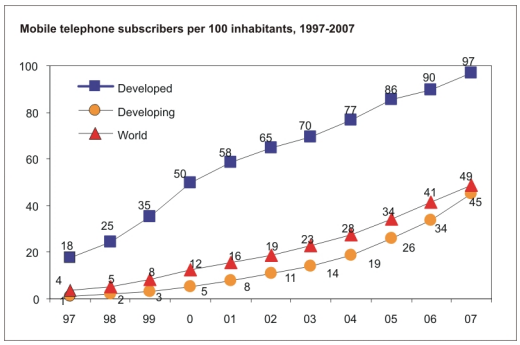

Fig. 1. Taux de pénétration de la téléphonie mobile au niveau mondial (source : ITU)

Les notions de convergence ne sont pas nouvelles et sont originellement apparues au Danemark lors de la première offre commerciale d'unification de la messagerie (mobile et fixe) en 1997. Aujourd'hui la mouvance se concentre sur la réutilisation des réseaux radioélectriques locaux Wi-Fi qui se sont largement démocratisés aussi bien dans le monde résidentiel que le monde entreprise. Certaines bandes de fréquence des réseaux Wi-Fi peuvent être non licenciées[6] comme la bande 2,4 GHz qui est

[6] bien que soumises à approbation par l'ARCEP en France.

utilisée dans les normes IEEE 802.11b, IEEE 802.11g et IEEE 802.11n. Des offres de convergence apparaissent donc et permettent la réutilisation de ces réseaux Wi-Fi en tant que nouveau moyen d'accès pour de la téléphonie mobile. Les technologies utilisées ensuite pour le transport de la voix peuvent être très différentes entre elles, mais les impératifs restent les mêmes : arriver à fédérer les accès, offrir une même qualité de service (qualité ressentie) et la possibilité du passage d'un réseau à l'autre de manière non perceptible par l'utilisateur (aussi appelé « seamless handover »). Cela permet aussi en pratique dans le cadre d'un terminal bi-mode GSM/Wi-Fi d'augmenter ostensiblement la couverture réseau dans des zones parfois difficiles à couvrir avec efficacité (comme l'intérieur des bâtiments).

3 Aperçu de la sécurité des architectures GSM

Cette partie n'a pas pour but d'être exhaustive à la vue de l'abondante littérature sur le sujet. Toutefois, il est bon de rappeler certaines problématiques de sécurité qui ont été prises en compte dans une norme qui a maintenant presque vingt ans.

3.1 Authentification

L'authentification 2G (utilisée dans les réseaux GSM) est basée sur un protocole de type *challenge response* ainsi que sur des algorithmes de cryptographie à clé secrète. La clé secrète se retrouve à la fois stockée dans le réseau de l'opérateur[7] et côté utilisateur[8] dans un Subscriber Identity Module (SIM) qui est une carte à puce. Il faut considérer que l'environnement de l'utilisateur final n'est pas un environnement de confiance donc la clé secrète est stockée dans une zone dont le but est de résister à l'extraction[9]. C'est une propriété importante qui a joué un rôle majeur dans le niveau de sécurité et donc la pérennité de ces architectures.

Une autre remarque importante relative à l'authentification GSM est que seule une authentification du client est réalisée : ceci laisse en théorie la porte ouverte[10] à des attaques de l'homme du milieu. Cependant, de part le coût élevé des stations de base GSM (Base Transceiver Station, BTS) ou de solutions commerciales dédiés à l'interception, l'attaque n'est possible qu'avec des moyens financiers assez conséquents[11].

[7] normalement peu enclin aux attaques du moins depuis l'extérieur.
[8] qui lui est bien plus propice aux attaques.
[9] que ce soit par des moyens logiques ou physiques.
[10] en particulier pour de la voie radioélectrique.
[11] bien entendu, tout est relatif...

3.2 Confidentialité des communications

En plus du mode « pas de chiffrement »A5/0, trois autres modes de chiffrement existent et présentent des niveaux de robustesse bien différents[12]. Le mode A5/2 présente des faiblesses intrinsèques[13] et n'apporte pas de sécurité en soit, par contre, le mode A5/1 est bien plus robuste. Cependant, ce dernier est tout à fait accessible pour peu que l'attaquant ait à sa disposition suffisamment de moyens financiers. Les premières publications théoriques en *ciphertext-only* datent de 2003 [17] et les premières publications d'implémentations de ces attaques datent de 2008 [18]. Enfin, le mode A5/3 [19] est quant à lui une adaptation de KASUMI [20] qui est utilisé dans les communications mobiles de troisième génération. Ce mode est de plus en plus souvent supporté dans les implémentations des téléphones GSM.

3.3 Des initiatives pratiques

L'arrivée des « Software Radio »comme le Universal Software Radio Peripheral [21] et les initiatives associées comme GNUradio [22] laissent entrebâiller la porte à des attaques sur le chiffrement réalisables avec des moyens financiers acceptables. Ces outils permettent de traiter la couche physique au niveau logiciel et donc de pouvoir (en théorie) démoduler les signaux radioélectriques. Bien entendu, cela nécessite un travail considérable et des connaissances pointues en traitement du signal mais il faut aussi matérialiser toutes les couches supérieures (et machines à état afférentes) en se basant sur les normes GSM qui sont (heureusement) publiques.

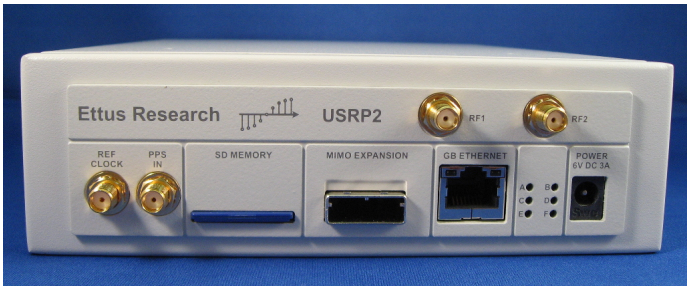

Fig. 2. Ettus USRP2 (source : Ettus Research LLC

Basé sur ce type de matériel, un nouveau projet est né récemment, le projet OpenBTS [23,24], qui propose de matérialiser l'interface Um [25] qui est l'interface

[12] en pratique, le mode A5/2 est une version « export »destinée aux pays autres que l'Europe et les États-Unis.
[13] lire volontaires.

radio entre le mobile (Mobile Station, MS) et la station de base GSM. Nul doute que toutes ces initiatives sont vouées à se démocratiser dans un futur proche.

3.4 Conclusions

Malgré certaines faiblesses universellement reconnues, les architectures GSM n'ont pas été affectées par le potentiel de fraude. La technologie est reconnue comme fiable et est considérée de confiance par ses utilisateurs. Ce standard est toujours largement utilisé à travers le monde ce qui est un réel succès pour une technologie d'une vingtaine d'années.

4 Exigences de sécurité

Nous présentons ici quelques exigences de sécurité dans le déploiement de technologies d'accès opérateur. Ces exigences permettront au lecteur de prendre conscience des difficultés auxquelles font face les opérateurs afin de maintenir un service optimum en fonction de l'hostilité de l'environnement. Bien entendu, ces exigences de sécurité peuvent être remplies par une ou un ensemble de fonctionnalités.

Les exigences peuvent être fonctionnelles comme par exemple :

- protection de l'identité des utilisateurs,
- protection des communications utilisateurs,
- garantie de l'accès au réseau si couverture,
- garantie de l'accès aux appels d'urgence,
- garantie d'une facturation fiable,
- garantie de la protection des données opérateurs.

Et (en partie) assurées par des mesures techniques :

- protection de la signalisation,
- protection du transport des données,
- authentification des utilisateurs et du réseau de l'opérateur,
- résistance des équipements de l'opérateur déposés dans des environnements physiques non sûrs

Les éléments ci-dessus ne sont pas exhaustifs, le lecteur intéressé pourra se référer au standard 3GPP TS 33.102 [26] pour une vision à la fois plus complète et plus précise.

5 Architectures de Convergence Fixe-Mobile

Nous présentons ici les différentes architectures de Convergence Fixe-Mobile en se focalisant sur les mécanismes de sécurité présents. Nous verrons que des idées et briques communes se dégagent sur lesquelles nous pourrons alors nous focaliser.

5.1 Unlicensed Mobile Access

Les spécifications UMA ont été initialement développées par un groupe d'opérateurs et de constructeurs, ces premières spécifications sont apparues en septembre 2004. Ensuite, ces spécifications ont été intégrées dans un groupe de travail du 3GPP en tant que « Generic Access to A/Gb interfaces ». En avril 2005, des spécifications plus avancées ont été publiées sous la dénomination « Generic Access Network (GAN) ». Cet acronyme souvent utilisé en normalisation représente donc la technologie UMA. Fin 2007, la norme a encore évolué pour intégrer le support des interfaces 3G qui peut alors être utilisé que ce soit en tant que terminal final (l'utilisateur) ou point d'accès (extension de couverture GSM/UMTS aussi appelée FemtoCell), ces fonctionnalités sont décrites dans la release 8 de GAN.

Les spécifications originelles UMA couvraient les besoins de « handover »entre les technologies GSM et Wi-Fi/Bluetooth. Le but du jeu était de pouvoir réaliser du « seamless handover »entre ces différentes technologies tout en maintenant un niveau de sécurité au moins équivalent[14]. Nous avons donc affaire à des terminaux bi-mode GSM/Wi-Fi qui pourront recevoir/établir des appels téléphoniques soit par le réseau 2G directement (GSM) soit en passant par le réseau Wi-Fi (tout en terminant dans le réseau 2G). Il est donc nécessaire d'avoir une infrastructure qui permette l'interconnexion entre le monde Internet et le monde 2G, ce que propose UMA. En ce qui concerne le transport des données autres que la voix, le protocole UMA dans sa version initiale est capable de transporter des communications vers le coeur de réseau GPRS. De récentes évolutions au 3GPP intègrent aussi les aspects 3G, la norme est alors appelée UMA-3G.

Sur le plan sécurité, la norme UMA repose sur le protocole IKEv2 et les protocoles d'authentification EAP-SIM ou EAP-AKA pour authentifier les utilisateurs et le « réseau ». En effet, le « réseau »est composé de plusieurs éléments intéressants à authentifier : le réseau coeur de l'opérateur (représenté par le couple Home Location Register, HLR, et l'Authentication Center, AuC) et la passerelle de sécurité frontale IKEv2 (Security Gateway, SeGW). Ensuite, grâce aux associations de sécurité préalablement négociées par IKEv2, une session IPsec permet alors d'assurer la

[14] par rapport aux attaques pratiques.

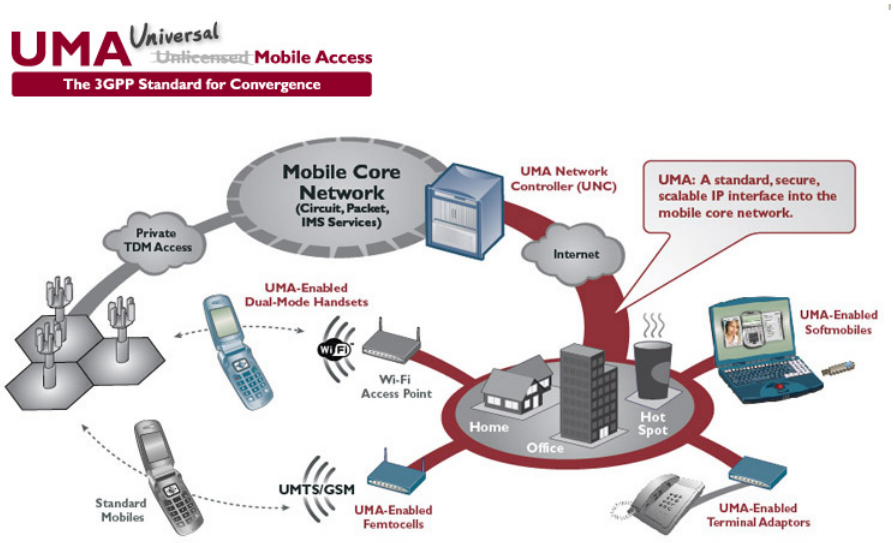

Fig. 3. Architecture Unlicensed Mobile Access (source : UMA Forum)

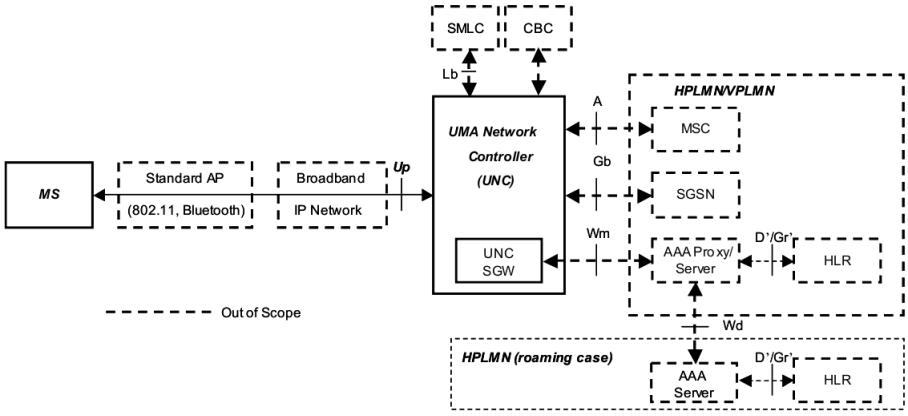

Fig. 4. Architecture Generic Access Network (source : 3GPP)

sécurité des communications transmises entre le terminal UMA et cette passerelle. Les communications UMA pourront alors transiter de manière sûre entre le terminal UMA et un équipement situé derrière[15] la passerelle IKEv2, l'UMA Network Controller (UNC). La signalisation et les communications GSM/GPRS sont alors transportées (ou gérées) via le protocole UMA à travers l'UNC vers le coeur de réseau GSM/GPRS de l'opérateur.

L'article n'étant pas dédié sur la sécurité d'UMA, mais sur la sécurité de certaines briques communes à toutes les technologies de Convergence Fixe-Mobile, nous invitons le lecteur à consulter les articles « Convergence Fixe-Mobile : UMA (Unlicensed Mobile Access) sur le devant de la scène ? » [27] et « Implications of Unlicensed Mobile Access (UMA) for GSM security » [28] pour comprendre les principaux impacts (sur le plan sécurité) à retenir sur les problématiques de l'introduction de la technologie UMA.

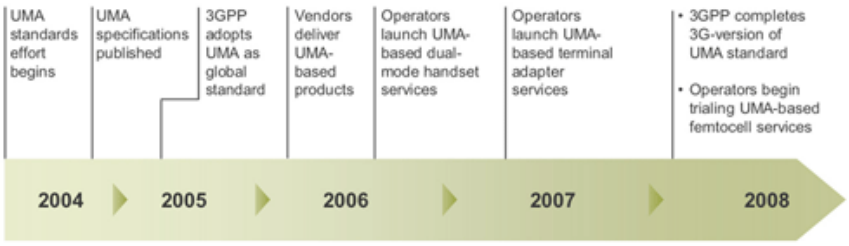

Fig. 5. Historique UMA (source : UMA Forum)

5.2 Interworking-WLAN

Les aspects inter-fonctionnement avec les réseaux WLAN ont été intégrés à partir de la 3GPP release 6 publiée en 2004. L'utilisation des protocoles d'authentification EAP-SIM et EAP-AKA est nécessaire dans les environnements orientés 3GPP qui sont destinés au monde opérateur. Les détails de l'architecture sont décrits dans le standard 3GPP TS 23.234 [29].

Pour plus de précisions, le lecteur intéressé pourra se référer à la description des fonctionnalités sécurité supportées dans I-WLAN via le standard 3GPP TS 33.234 [30].

[15] au niveau fonctionnel, car physiquement un même équipement peut faire office de passerelle de sécurité IKEv2 et de contrôleur UMA.

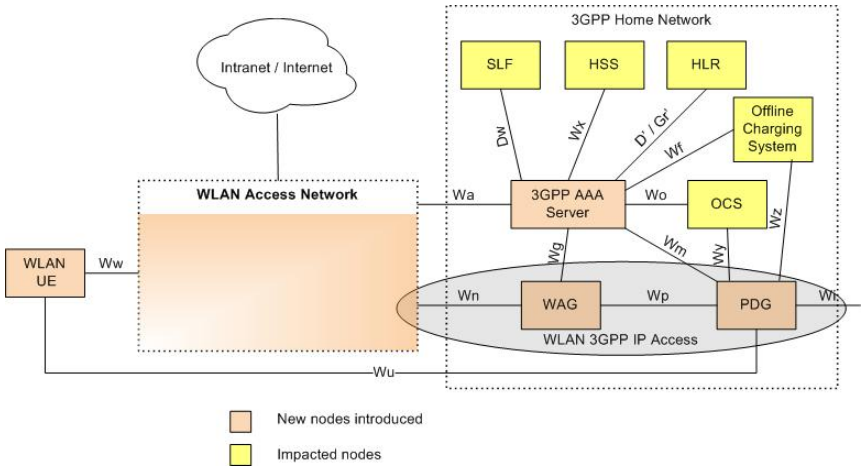

Fig. 6. Architecture Interworking-WLAN (source : 3GPP)

5.3 IP Multimedia Subsystem

La technologie IMS est originellement issue d'un forum d'industriels créée en 1999 qui avait pour but de réaliser la convergence de services mobiles/filaires voix et multimédia. Leurs travaux ont ensuite été repris par le 3GPP et intégrés dans la release 5 publiée en 2002 lorsque le protocole Session Initiation Protocol (SIP) [31] a été rajouté. Ce protocole joue un rôle clé dans IMS car il permet la signalisation des sessions multimédia. Un des principes de l'architecture IMS est de séparer la couche transport de la couche des services, et d'utiliser la couche de transport pour des fonctions de contrôle, de signalisation et de qualité de service associée à l'application désirée. Le but de l'IMS est de fournir une infrastructure unique pour tous les services multimédia quels que soient les réseaux d'accès.

Par rapport aux architectures UMA et I-WLAN, l'architecture IMS se distingue par le rôle de représentation de la couche « haute »des architectures d'accès qui seront déployées.

Sur les aspects sécurité, IMS repose à la fois sur la sécurité des couches d'accès et des mécanismes de sécurité présents pour sécuriser les transactions SIP et le transport média via RTP. Dans cet article, nous n'aborderons pas ces problématiques et nous traiterons uniquement la partie accès.

5.4 Autres alternatives

D'autres technologies peuvent contribuer ou se rattacher à la famille des architectures de Convergence Fixe-Mobile. Nous les citons dans cet article pour convaincre

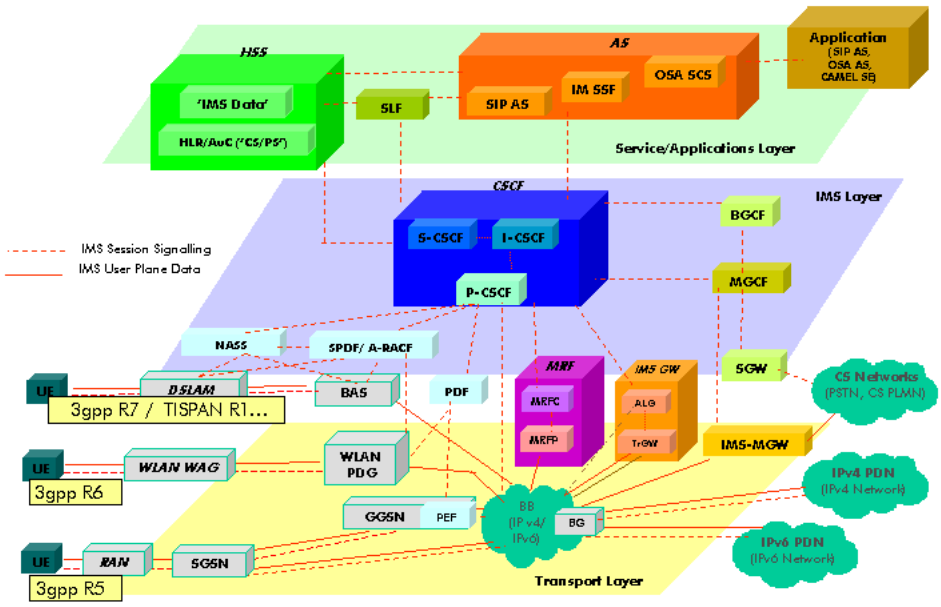

Fig. 7. Architecture IP Multimedia Subsystem (source : 3GPP)

le lecteur que la multiplication des accès rend le monde des télécommunications en pleine effervescence.

FemtoCell Cette technologie permet d'apporter de la couverture de 2G/3G dans des petites zones difficiles à couvrir (ou non couvertes) en particulier à l'intérieur de bâtiments ou en sous-sols. Bien entendu le concept est extensible à volonté et il est possible d'étendre des réseaux GPRS, WiMAX ou autres... Tout n'est qu'une question de dénomination. Une forte contrainte en sécurité est de pouvoir rattacher cet équipement au réseau coeur de l'opérateur mais depuis un réseau considéré comme n'étant pas de confiance (car présent dans des milieux résidentiels ou entreprise). Certes lors des déploiements des cellules 2G/3G les bornes pouvaient être « attaquées »physiquement mais dans le cadre des FemtoCell cela est tout de même bien plus facile car l'équipement est facilement accessible physiquement (car localisée chez le client).

Sur le plan sécurité, les solutions de raccordement de la FemtoCell au coeur de réseau de l'opérateur peuvent être multiples. Cela peut aller de la simple authentification par secret partagé avec le protocole IKEv1 à l'utilisation du protocole IKEv2 avec une authentification EAP-SIM ou EAP-AKA (et donc utilisation de crédences de type SIM ou USIM). Bien entendu, la FemtoCell a en pratique un accès réseau IP vers une passerelle de l'opérateur afin de s'y rattacher.

Nous retrouvons ici toutes les problématiques classiques en sécurité système lorsque l'équipement est positionné dans un endroit non cloisonné physiquement. Il faut donc s'assurer qu'il n'est pas possible de récupérer facilement une version « en clair »du firmware ou d'autres informations sensibles contenues dans cet équipement, typiquement lorsque cela est le cas, les crédences d'authentification.

6 La clé de voûte de la sécurité de ces architectures : l'accès

Ces architectures mettent en jeu de nombreux protocoles et cet article n'a pas la prétention de tous les analyser en profondeur sachant que la littérature est particulièrement abondante sur la plupart des protocoles mis en jeu (Wi-Fi, SIP...).

La majorité de ces architectures reposent sur un élément essentiel et commun : la technologie d'accès. En effet, elles sont indépendantes des mécanismes de sécurité de la couche liaison et reposent sur le protocole IKEv2 aidé du protocole de transport d'authentification EAP. Le protocole IKEv2 a pour but (dans le cadre de ces architectures) de transporter de manière sûre les échanges EAP et de créer des associations de sécurité (Security Association, SA) qui seront utilisées pour établir un tunnel IPsec entre le client et la passerelle de sécurité appartenant à l'opérateur.

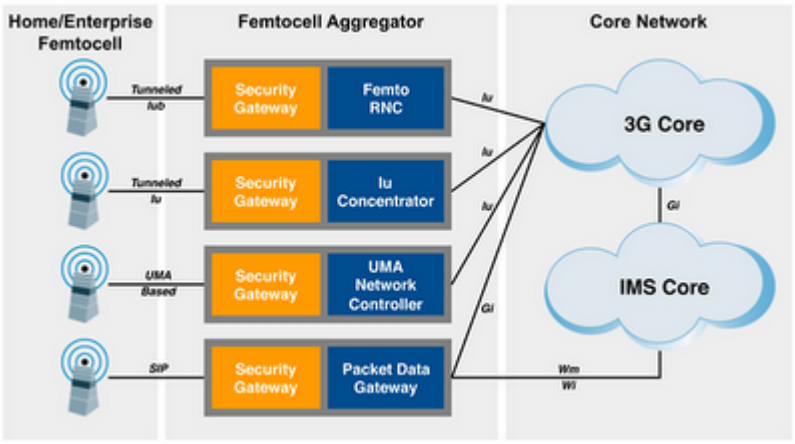

Fig. 8. Architecture FemtoCell (source : 3GPP)

L'objectif est de se reposer sur un protocole reconnu comme sûr[16] (IPsec) afin d'assurer une protection[17] des communications entre le client et le réseau de l'opérateur. Le principe étant qu'il est (en priorité) primordial de se protéger contre les attaquants non authentifiés. Bien entendu, cela ne signifie pas que les attaques issues d'utilisateurs authentifiés sont à négliger !

Nous détaillerons dans les chapitres suivants nos travaux sur la recherche de vulnérabilités dans les implémentations IKEv2 et EAP qui font partie de la surface d'attaque atteignable (accessible de tout utilisateur non authentifié).

7 Efforts réalisés sur le plan sécurité

Ce chapitre présente les principaux axes sur lesquels nous agissons pour appréhender au mieux la sécurité des architectures de Convergence Fixe-Mobile. Une attention toute particulière est faite sur l'approche audit et recherche de vulnérabilités qui sont primordiales pour s'assurer d'une amélioration du niveau de sécurité à la fois sur le plan ingénierie ou implémentation.

7.1 Suivi de standardisation

Le suivi de la standardisation et en particulier du groupe SA3 [34] du 3GPP est indispensable pour anticiper les problématiques de sécurité durant la conception des

[16] si bien utilisé...

[17] contre les attaques d'écoute passive, d'écoute active, de rejeu, d'attaques sur l'intégrité, d'attaques sur l'authenticité...

normes. Un suivi attentif permet aussi de connaître certains risques inhérents aux technologies normalisées ou en cours de normalisation afin d'être en avance de phase afin d'émettre les recommandations techniques adéquates lors du déploiement de ces technologies. Par ailleurs, aujourd'hui toutes les technologies ont tendance à interagir entre elles, comme par l'exemple la réutilisation de protocoles issus de l'IETF dans des normes IEEE, une participation à chacun de ces corps de normalisation permet alors d'appréhender au mieux les problématiques de sécurité.

7.2 Ingénierie d'architecture réseau

L'ingénierie des flux réseau joue un rôle primordial dans le niveau de sécurité de toute architecture. Ceci est encore plus vrai dans les architectures de Convergence Fixe-Mobile où les utilisateurs authentifiés ne devront avoir accès qu'à certaines ressources réseaux ou aux applications. Tout ceci ne peut être réalisé qu'après une compréhension claire des pré-requis de service teintée de principes forts sur le plan de la sécurité réseau.

Cela a aussi son importance dans l'accessibilité des failles de sécurité au niveau des implémentations. En effet, l'exemple le plus simple est de constater que de nombreuses failles de sécurité sont présentes dans des systèmes informatiques mais qu'elles ne sont pas atteignables du fait de la présence de règles de filtrage adéquates que ce soit en périphérie du système informatique vulnérable ou sur ce dernier. Ceci est capital dans toute architecture déployée, être capable de réduire un risque initialement fort grâce à une mesure préventive (et non corrective, en particulier lorsque malheureusement la correction tarde à venir...).

7.3 Audits de sécurité

Que ce soit par des évaluations de produits ou des audits sur des architectures déployées en validation il est nécessaire de relever les problèmes de sécurité le plus en amont possible. Ce n'est qu'à l'épreuve du feu que le niveau de sécurité d'un équipement ou d'une architecture augmente sensiblement. Il faut bien se rappeler que trouver des vulnérabilités permet surtout de les faire corriger avant qu'elles ne soient exploitées silencieusement (ou pas).

Lors des audits (bien que cela soit souvent la « slipfest[18] » [32]) il est souvent nécessaire de rappeler les bases classiques du durcissement de la configuration des systèmes d'exploitation et applications internes. L'audit de sécurité permet aussi d'avoir une vue « extérieure »et « objective »afin d'avoir des retours sur des points qui

[18] comme diraient certains collègues.

n'auraient pas été imaginés par les concepteurs de l'équipement ou de l'architecture. Même si l'audit n'est pas la solution miracle, il fait partie du processus d'amélioration continue[19].

7.4 Mise en place d'infrastructures de tests sécurité

En parallèle des audits de sécurité instantanés sur des équipements ou architectures, il est nécessaire de déployer des plates-formes de validation sur lesquelles les tests classiques fonctionnels seront réalisés, mais aussi les tests spécifiques sur les aspects sécurité. Ce type de démarche « en amont »réduit fortement les risques lorsqu'elle est correctement appliquée. Le principal inconvénient de cette méthode est de mobiliser plus de ressources au niveau des auditeurs sécurité car ces derniers réalisent alors un travail ressemblant à un processus de validation. Ceci est malgré tout un mal nécessaire à la vue de la foison des technologies à auditer en termes de Convergence Fixe-Mobile.

7.5 Développement de nouveaux outils

Qui dit nouveaux standards dit nouveaux outils. Il n'est pas concevable de pouvoir auditer ces nouvelles architectures sans outils appropriés. Les protocoles IKEv2 et EAP ne sont pas encore courants dans l'expérience de l'auditeur au même titre que les implémentations auditées. Raison de plus pour développer des outils permettant une évaluation de la qualité des implémentations de ces protocoles mais aussi d'outils permettant de se faire passer pour un terminal mobile. Les premiers outils permettront (éventuellement) de découvrir des erreurs d'implémentations logicielle tandis que les seconds permettront de réaliser des tests de visibilité (plus classiques) grâce à un portable tout outillé[20].

8 Recherche de vulnérabilités dans les implémentations par techniques de fuzzing

Le sujet de la sécurité des architectures de Convergence Fixe-Mobile est extrêmement large, nous nous focalisons dans cet article que sur la recherche de vulnérabilités dans les implémentations IKEv2, EAP et ses méthodes associées (EAP-SIM et EAP-AKA) par techniques de fuzzing.

[19] pour peu que les recommandations soient appliquées dans la mesure du possible.
[20] ce qui est bien plus pratique et efficace que depuis un terminal mobile.

8.1 Historique du fuzzing

Le fuzzing – qui est la contraction de « fuzz »et de « testing »– est un terme issu des travaux de Barton Miller lorsqu'il faisait tester des applications Unix à ses étudiants [35]. Ses travaux, qui datent de 1989, ont montré que de nombreuses failles de sécurité pouvaient être découvertes par des techniques simples d'injection de fautes. Pour notre culture, l'origine même du mot « fuzz »provient du bruit émis par des caractères aléatoires sur une ligne téléphonique [36]. Aujourd'hui, le fuzzing fait partie de l'arsenal classique du chercheur de failles de sécurité.

8.2 Définitions du fuzzing

Parmi de très nombreuses définitions du fuzzing, nous avons choisi de retenir les trois suivantes :

- *Wikipedia* – « Le *fuzzing* est une technique pour tester des logiciels. L'idée est d'injecter des données aléatoires dans les entrées d'un programme. Si le programme échoue (par exemple en plantant ou en générant une erreur), alors il y a des défauts à corriger. »
- *Open Web Application Security Project (OWASP)* – « Le *Fuzz testing* ou *fuzzing* est une technique de tests logiciels en *boite noire*, qui consiste à découvrir des erreurs d'implémentation logicielle en utilisant de l'injection de données mal formées, et ce de manière automatisée. »
- *Whatis.com* – « Le *Fuzz testing* ou *fuzzing* est une technique de tests logiciels utilisée pour découvrir des erreurs de développement ou des faiblesses dans des logiciels, systèmes d'exploitation ou réseaux en injectant de nombreuses entrées aléatoires dans le système afin qu'il s'arrête. Si une vulnérabilité est découverte, un outil appelé *fuzzer*, permet d'indiquer les causes potentielles de l'arrêt du système. »

Nous constatons ci-dessus que ces définitions ne sont pas forcément en accord les unes avec les autres. Par exemple, la définition de Whatis.com suggère qu'une erreur de développement entraîne forcément un arrêt de l'application testée. De la même manière, la définition de l'OWASP suggère que les données injectées sont forcément mal formées, ce qui n'est pas obligatoire en pratique pour découvrir des vulnérabilités. Il faut bien préciser ce qui est entendu par données « mal formées », est-ce par rapport à une spécification (ou à une norme) ou est-ce par rapport à l'implémentation de cette spécification ? Ceci pour rappeler que le fuzzing vise à trouver des erreurs d'implémentations logicielles et non à découvrir des vulnérabilités théoriques dans des spécifications.

Par conséquent, en s'efforçant de synthétiser au mieux les définitions ci-dessus, notre conclusion est que le fuzzing est une technique de recherche d'erreurs d'implémentation logicielle par injection de données invalides. Dans cette définition, le caractère invalide de ces données est bien entendu relatif à l'implémentation logicielle testée, i.e. trouver les tests les plus pertinents susceptibles de découvrir des erreurs d'implémentation classiques (débordement de tampon, bogue de chaîne de format...). La composante aléatoire est souvent citée dans les définitions du fuzzing, cependant elle n'est pas en pratique obligatoire pour réaliser du fuzzing...

Nous invitons le lecteur à se référer à l'article « recherche de vulnérabilités dans les drivers 802.11 par techniques de fuzzing »publié à SSTIC en 2007 [37] et au dossier « fuzzing »du magazine MISCMAG de septembre 2008 [38] pour de plus amples informations.

8.3 Pourquoi avoir choisi le fuzzing ?

La recherche de failles de sécurité se repose généralement sur trois principales techniques :

- la rétro-ingénierie qui consiste à étudier le code bas niveau et trouver des erreurs d'implémentation logicielle en ayant accès au code après compilation,
- l'audit de code source qui consiste à étudier le code avant compilation et trouver des erreurs d'implémentation logicielle,
- l'injection de données invalides (le fuzzing).

Dans notre cas, la question du choix de telle ou telle technique ne se pose même pas car nous ne disposons pas d'accès facile aux applications testées (pas de code source, pas de débogage, obligations légales ou contractuelles...). Nous sommes dans le cas de la *boite noire* où le fuzzing est réellement pertinent.

8.4 État de l'art des outils de fuzzing IKEv2 et EAP

Ce chapitre présente les outils de fuzzing commerciaux ou Open Source qui sont capables de réaliser du fuzzing d'implémentations IKEv2, EAP ou méthodes EAP. Nous avons délibérément écarté de cette section tous les fuzzers génériques qui, par exemple, opèrent par mutations de données valides.

Fuzzing IKEv2 À notre connaissance le seul outil disponible est la suite DEFENSICS de Codenomicon [39]. Il s'agit d'un fuzzer spécifique[21]. Nous allons donc étudier un domaine où peu de publications sont disponibles.

[21] développé pour fuzzer un protocole particulier et non un fuzzer par mutations qui est bien plus générique.

Fuzzing EAP et méthodes EAP À notre connaissance le seul outil disponible est la suite DEFENSICS de Codenomicon qui permet de fuzzer des implémentations EAP, EAP-MD5 [40], EAP-GTC [41] et EAP-MSCHAPV2 [42]. La contrainte semble[22] être que l'implémentation EAP fuzzée doit être un équipement supportant IEEE 802.1X (point d'accès 802.11 ou un commutateur 802.3) [50], l'implémentation EAP dans RADIUS n'étant alors pas fuzzée directement[23] (mais à travers le relais d'authentification qui est soit le point d'accès soit le commutateur).

8.5 État de l'art des vulnérabilités sur les implémentations IKEv2, EAP et méthodes EAP

Pour réaliser cette recherche, nous nous sommes basés sur la base « Common Vulnerabilities and Exposures » [43] qui référence l'ensemble des vulnérabilités publiques.

Il est bon aussi de rappeler que de nombreuses vulnérabilités sont corrigées silencieusement dans des produits commerciaux mais aussi (et de manière plus étonnante) dans des logiciels Open Source. En effet, la position de Linus Torvalds sur la gestion des failles de sécurité dans le noyau Linux avait déclenché de grandes discussions récemment [44] et permet de se rendre compte que de nombreux erreurs de programmation ne sont pas référencées en tant que vulnérabilités alors qu'ils peuvent entraîner un déni de service par un tiers, ou encore, et que d'autres vulnérabilités sont référencées en tant que vulnérabilité de type déni de service alors que l'exécution de code arbitraire aura été démontrée. Nous précisons donc ici que ne sont discutées que des vulnérabilités publiques et référencées en tant que telles, par définition, les vulnérabilités privées ne sont pas connues du grand public.

Dans les tableaux suivants nous utiliserons les termes usités dans la norme IEEE 802.1X qui définit trois entités : le supplicant (assimilé au client), l'authenticator (assimilé au relais de l'authentification) et le serveur d'authentification (serveur Authentication Authorization Accounting, AAA, qui est généralement un serveur Remote Authentication Dial-In User Service, RADIUS [61], plus rarement un serveur Diameter [62]). Ces termes sont parfaitement transposables au protocole IKEv2 car dans le cadre de l'utilisation d'authentification EAP nous avons aussi affaire à une architecture avec ces trois entités fonctionnelles.

Vulnérabilités d'implémentations IKEv2 Dans le tableau 1 nous constatons qu'une seule vulnérabilité sur des implémentations IKEv2 a été publiée. Elle est

[22] n'ayant pas eu accès à l'outil, nous basons notre analyse sur le descriptif de la solution sur leur site Web.
[23] ceci a des implications fortes car l'équipement relayant les paquets EAP effectue généralement un traitement de ces derniers et peut alors ne pas transmettre les paquets fuzzés ce qui fausse les résultats du fuzzing de l'implémentation EAP dans le serveur RADIUS.

Tab. 1. Vulnérabilités d'implémentations IKEv2

CVE-ID	Cible	Description
CVE-2008-4551	Authenticator	strongSwan 4.2.6 and earlier allows remote attackers to cause a denial of service (daemon crash) via an IKE_SA_INIT message with a large number of NULL values in a Key Exchange payload, which triggers a NULL pointer dereference for the return value of the mpz_export function in the GNU Multiprecision Library (GMP).

déclenchable au tout début des échanges IKE i.e. lors de la réception d'un paquet `IKE_SA_INIT` dont un champ nommé Key Exchange contient une valeur particulière (champ Diffie-Hellman à zéro) [45]. À noter que cette vulnérabilité a été découverte par la société Mu Dynamics, spécialisée dans la sécurité et le fuzzing, donc probablement découverte par fuzzing (cette vulnérabilité est trouvable par fuzzing par mutations, ce qui serait en adéquation avec le fait qu'ils se sont spécialisés dans le fuzzing par mutations) [46].

Tab. 2. Vulnérabilités d'implémentations EAP

CVE-ID	Cible	Description
CVE-2008-5563	Authenticator	Aruba Mobility Controller 2.4.8.x-FIPS, 2.5.x, 3.1.x, 3.2.x, 3.3.1.x, and 3.3.2.x allows remote attackers to cause a denial of service (device crash) via a malformed Extensible Authentication Protocol (EAP) frame.
CVE-2008-2441	AAA	Cisco Secure ACS 3.x before 3.3(4) Build 12 patch 7, 4.0.x, 4.1.x before 4.1(4) Build 13 Patch 11, and 4.2.x before 4.2(0) Build 124 Patch 4 does not properly handle an EAP Response packet in which the value of the length field exceeds the actual packet length, which allows remote authenticated users to cause a denial of service (CSRadius and CSAuth service crash) or possibly execute arbitrary code via a crafted RADIUS (1) EAP-Response/Identity, (2) EAP-Response/MD5, or (3) EAP-Response/TLS Message Attribute packet.
CVE-2007-5651	Authenticator	Unspecified vulnerability in the Extensible Authentication Protocol (EAP) implementation in Cisco IOS 12.3 and 12.4 on Cisco Access Points and 1310 Wireless Bridges (Wireless EAP devices), IOS 12.1 and 12.2 on Cisco switches (Wired EAP devices), and CatOS 6.x through 8.x on Cisco switches allows remote attackers to cause a denial of service (device reload) via a crafted EAP Response Identity packet.

Vulnérabilités d'implémentations EAP Dans le tableau 2 nous constatons que trois vulnérabilités ont été publiées sur des implémentations d'EAP dont une dans un serveur RADIUS commercial et deux autres sur les implémentations EAP dans

les authenticators. La vulnérabilité CVE-2007-5651 [47] a été découverte par nous-mêmes en 2007 lors de travaux préliminaires sur le fuzzing d'implémentations EAP dans les authenticators et la vulnérabilité CVE-2008-2441 [48] a été découverte par nous-mêmes en 2008 durant nos investigations sur le fuzzing d'implémentations EAP dans les serveurs RADIUS.

Vulnérabilités d'implémentations de méthodes EAP Dans le tableau 3 nous constatons que seules trois vulnérabilités (CVE-2004-1459, CVE-2006-1354 et CVE-2007-2028) semblent trouvables par fuzzing, les autres étant relatives à une mauvaise conception comme par exemple l'absence de validation de certificat lors d'authentification utilisant ces mécanismes. Par ailleurs, il est important de souligner qu'aucune vulnérabilité n'est publiée sur des implémentations d'EAP-SIM ou EAP-AKA dans des serveurs RADIUS qu'ils soient commerciaux ou Open Source.

9 Recherche de vulnérabilités dans les implémentations IKEv2

9.1 Contexte et enjeux

Ce protocole est (lorsque la passerelle est correctement configurée) le seul visible depuis tout utilisateur non authentifié. Il est donc extrêmement critique que ces implémentations soient robustes car toute faille de sécurité présente dans la passerelle IKEv2 peut entraîner un déni de service déterministe ou de l'exécution arbitraire de code à distance. La redondance des passerelles aidant, nous pouvons alors faire face à un effet domino[24] lors d'une attaque visant une implémentation particulière d'une pile IKEv2.

Certes, en cas de découverte d'une vulnérabilité sur une implémentation IKEv2, il est peu probable qu'il soit faisable en pratique d'étudier l'exploitabilité[25] dans le cas d'implémentations propriétaires où il n'est pas possible de faire autre chose qu'une analyse en *boite noire*. Cela ne préjuge en rien de l'exploitabilité, seulement, les conditions techniques rendent très difficile l'investigation pour un attaquant externe n'ayant aucun accès à l'équipement (pas de débogage...).

En ce qui concerne les failles dans les implémentations EAP et méthodes EAP, cela est différent car les serveurs RADIUS sont des logiciels fonctionnant sur des systèmes d'exploitation standards et ouverts. La présence d'une faille de sécurité sur

[24] la solution avec plusieurs implémentations d'éditeurs différents serait une option technique mais en pratique elle est rarement envisagée pour des raisons économiques ou contractuelles.
[25] dans le sens exécution de code arbitraire.

Tab. 3. Vulnérabilités d'implémentations de méthodes EAP

CVE-ID	Cible	Description
CVE-2008-1114	Supplicant	Vocera Communications wireless handsets, when using Protected Extensible Authentication Protocol (PEAP), do not validate server certificates, which allows remote wireless access points to steal hashed passwords and conduct man-in-the-middle (MITM) attacks.
CVE-2008-1113	Supplicant	Cisco Unified Wireless IP Phone 7921, when using Protected Extensible Authentication Protocol (PEAP), does not validate server certificates, which allows remote wireless access points to steal hashed passwords and conduct man-in-the-middle (MITM) attacks.
CVE-2007-2028	AAA	Memory leak in freeRADIUS 1.1.5 and earlier allows remote attackers to cause a denial of service (memory consumption) via a large number of EAP-TTLS tunnel connections using malformed Diameter format attributes, which causes the authentication request to be rejected but does not reclaim VALUE_PAIR data structures.
CVE-2007-1068	Supplicant	The (1) TTLS CHAP, (2) TTLS MSCHAP, (3) TTLS MSCHAPv2, (4) TTLS PAP, (5) MD5, (6) GTC, (7) LEAP, (8) PEAP MSCHAPv2, (9) PEAP GTC, and (10) FAST authentication methods in Cisco Secure Services Client (CSSC) 4.x, Trust Agent 1.x and 2.x, Cisco Security Agent (CSA) 5.0 and 5.1 (when a vulnerable Trust Agent has been deployed), and the Meetinghouse AEGIS SecureConnect Client store transmitted authentication credentials in plaintext log files, which allows local users to obtain sensitive information by reading these files, aka CSCsg34423.
CVE-2006-1354	AAA	Unspecified vulnerability in FreeRADIUS 1.0.0 up to 1.1.0 allows remote attackers to bypass authentication or cause a denial of service (server crash) via Insufficient input validation in the EAP-MSCHAPv2 state machine module.
CVE-2004-1459	AAA	Cisco Secure Access Control Server (ACS) 3.2, when configured as a Light Extensible Authentication Protocol (LEAP) RADIUS proxy, allows remote attackers to cause a denial of service (device crash) via certain LEAP authentication requests.
CVE-2004-1099	AAA	Cisco Secure Access Control Server for Windows (ACS Windows) and Cisco Secure Access Control Server Solution Engine (ACS Solution Engine) 3.3.1, when the EAP-TLS protocol is enabled, does not properly handle expired or untrusted certificates, which allows remote attackers to bypass authentication and gain unauthorized access via a cryptographically correct certificate with valid fields such as the username.
CVE-2003-1096	AAA	The Cisco LEAP challenge/response authentication mechanism uses passwords in a way that is susceptible to dictionary attacks, which makes it easier for remote attackers to gain privileges via brute force password guessing attacks.

ces implémentations est plus critique par rapport aux impacts si l'exploitabilité est démontrée. Par ailleurs, le canal de communication naturel avec l'attaquant semble être la réutilisation du canal utilisé pour l'authentification EAP, même s'il faut reconnaître que cela est techniquement difficile [49].

9.2 Description du protocole IKEv2

Ce protocole permet l'établissement d'associations de sécurité qui seront alors utilisées par la pile IPsec afin d'apporter la sécurité des communications entre un point (dans notre cas, le client final) et un point fixe (dans notre cas, la passerelle de sécurité, la Security Gateway) comme cela est montré dans la figure 9.

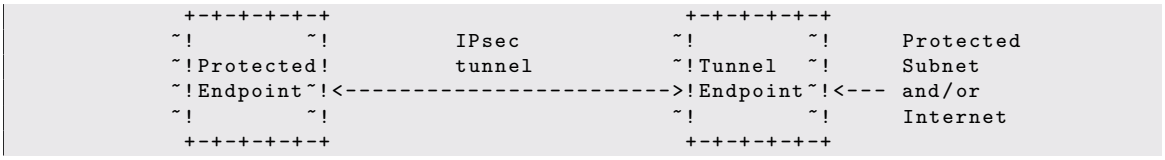

Fig. 9. Architecture Endpoint - Security Gateway IKEv2 (source : RFC 4306)

C'est ce type d'architecture qui est utilisée aussi bien dans le cadre de terminaux mobiles que de FemtoCells. Le réseau protégé étant alors la plate-forme d'accès qui sera elle-même connectée ensuite au coeur de réseau de l'opérateur mobile.

Le protocole IKEv2 suit un schéma de requête/réponse au-dessus du protocole UDP où la réponse joue un rôle d'acquittement des requêtes. Dans la figure 10, nous avons le premier échange de données entre le client (*initiator*) et la Security Gateway (*responder*). Cet échange de données est transporté dans des messages de type `IKE_SA_INIT` qui transportent `SAi1`, `KEi` et `Ni` dans le sens *initiator* vers *responder* et `SAr1`, `KEr` et `Nr` dans le sens *responder* vers *initiator*. Les champs `SA*1` transportent les informations relatives à la création de l'association de sécurité `IKE_SA`, les champs `KE*` transportent les informations relatives pour réaliser le Diffie-Hellman [52] et les champs `N*` transportent un aléa.

Avec la réussite de la première phase `IKE_SA_INIT` tout un ensemble d'algorithmes de sécurité auront été négociés ainsi qu'un ensemble de clés auront été dérivées, ce qui permettra alors de transporter de manière sécurisée les prochains contenus IKEv2. En particulier, lors de l'utilisation du protocole EAP pour authentifier le réseau et le terminal, ces échanges seront alors protégés comme cela est décrit dans la figure 11

Fig. 10. Phase initiale IKEv2 (source : RFC 4306)

via l'acronyme SK. Toute cette phase est décrite dans la section 2.16 de la RFC 4306 (IKEv2).

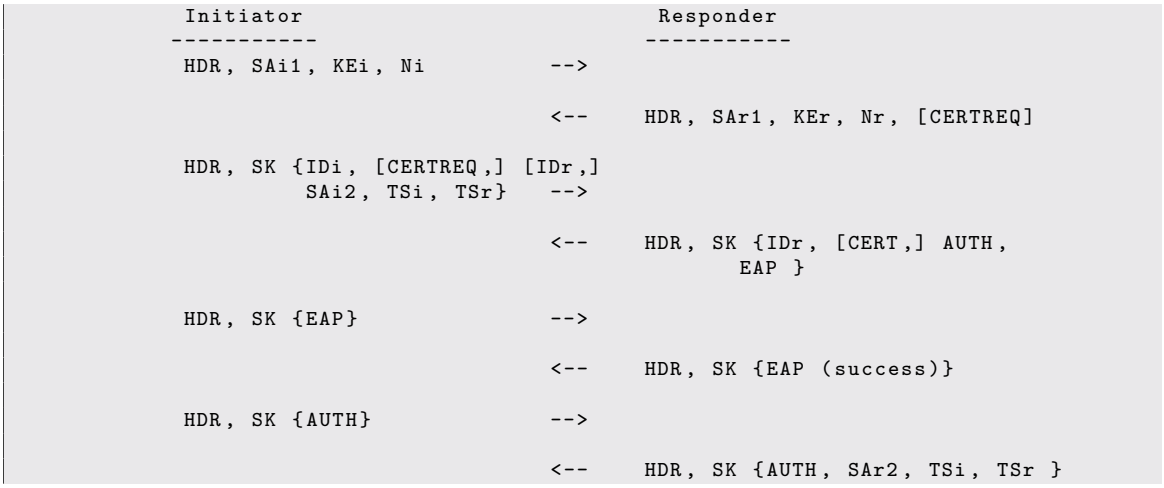

Fig. 11. Phase compète IKEv2 avec EAP (source : RFC 4306)

Le champ optionnel CERTREQ montre qu'il est possible d'authentifier la Security Gateway grâce à un système de certification où la Security Gateway présente son certificat au client qui alors sera capable de le valider si bien sûr il a toutes les informations nécessaires pour ce faire (avec entre autres le certificat de l'autorité de certification). Cette information est très importante pour l'auditeur sécurité : il vérifiera que la fonctionnalité est bien activée et qu'elle est bien utilisée (dans notre cas, que le client IKEv2 valide bien le certificat présenté par la passerelle IKEv2) car l'expérience montre que de nombreux systèmes similaires sensés vérifier la validité des certificats ne le font visiblement pas [53,54].

9.3 Choix techniques

Conditions techniques Elles définissent les choix techniques qui seront réalisés. Le protocole IKEv2 est complexe dans la génération des paquets (avec beaucoup de charges sous forme de liste chaînées) et utilise dès les premiers échanges des mécanismes cryptographiques pour chiffrer les autres messages. Or, dans le fuzzing, il est intéressant d'aller le plus loin possible dans les parties de l'implémentation qui seront utilisées lors des échanges. En conséquence, si l'objectif est d'arriver à fuzzer le contenu des messages `IKE_AUTH` alors un processus de fuzzing par mutations ne sera pas pertinent car les paquets chiffrés ne seront plus intègres si des modifications par mutations auront été effectuées (et donc le paquet sera normalement rejeté et son contenu non analysé par l'implémentation). Ces constatations permettent de s'orienter vers la génération de tests par modélisation de protocole qui semble pertinent dans le cas de IKEv2.

Une autre condition technique est l'obligation de réaliser les tests en *boite noire*. En effet, pour des raisons techniques il ne nous est pas possible d'étudier le comportement de la cible en étant en local sur le système audité (grâce par exemple à l'attachement d'un débogueur). Par conséquent, ceci a aussi un impact fort sur les choix techniques car il faudra définir un test permettant de savoir si l'implémentation se comporte de manière « non conforme »après un ou un ensemble de tests.

À noter aussi que nous avons choisi de développer un fuzzer d'implémentations serveurs IKEv2. Nous considérons que les risques sont bien moins importants sur les implémentations clientes IKEv2 car les impacts (gravité) sont plus faibles pour l'opérateur mais cela n'enlève rien à l'intérêt technique car il apparaît probable de trouver des choses intéressantes pour ceux qui auront la volonté de rechercher dans cette voie.

Base technique Tous nos développements sont basés sur des briques issues de l'outil Sulley Fuzzing Framework [55] qui est à notre avis le fuzzer Open Source le plus abouti lorsqu'il s'agit de fuzzer des protocoles réseau par modélisation de protocole. Cet outil est développé en langage Python ce qui offre à la fois une grande souplesse et puissance d'utilisation. Nous invitons (encore une fois !) le lecteur à se référer au dossier « fuzzing »du magazine MISCMAG de septembre 2008 qui présente un article complet sur le fonctionnement et l'utilisation de Sulley.

Génération des tests Nous nous reposons sur un système de génération de tests par modélisation du protocole. Il a donc fallu décrire complètement les charges contenues dans les trames `IKE_SA_INIT` et `IKE_AUTH`. Ce protocole est complexe dans

sa construction en comparaison avec d'autres protocoles bien plus simples à modéliser comme FTP ou HTTP. Dans la figure 12 nous montrons un exemple de spécification de la modélisation d'un champ Key Exchange grâce à des primitives de Sulley comme `s_short()` et `s_random()` qui permettront alors de fuzzer ces champs grâce aux heuristiques de Sulley.

```
    def key_exchange_payload(next_payload='None'):
        '''Key Exchange Payload'''

        name = 'Key Exchange'

        if s_block_start(name):
            payload_header(name, next_payload)                          #
                Payload Header
            s_short(0x0e, endian='>', fuzzable=True)                    # DH
                Group Number
            s_static('\x00\x00')                                        #
                Reserved
            s_random(PAD * 256, 0, 100, name='Key Exchange Data')       # Data
                (DH value)
        s_block_end()
```

Fig. 12. Exemple de définition d'un champ de type Key Exchange

Gestion des états Pour le premier état, l'envoi de paquets `IKE_SA_INIT` ne pose pas de problème particulier. Par contre, la principale problématique est de pouvoir créer des paquets `IKE_AUTH` valides i.e. qui utilisent l'association de sécurité qui aura été négociée dans les échanges précédents. Une première idée consiste à développer une pile IKEv2 de zéro (de préférence en Python de manière à l'intégrer directement avec Sulley) mais cette option s'avère très coûteuse en temps de développement. La deuxième approche (plus facile, mais moins intégrée) consiste à modifier un client IKEv2 Open Source de manière à lui faire faire le travail et à lui faire passer le test niveau contenu du paquet `IKE_AUTH` afin qu'il puisse le chiffrer avec la SA qu'il aura préalablement négociée. Nous avons donc développé une modification de l'outil Racoon2 [51] (implémentation IKEv2 sous *nix) qui permet de recevoir puis d'envoyer les tests `IKE_AUTH` générés par Sulley après avoir négocié la première phase IKEv2 avec la passerelle de sécurité.

Par conséquent, notre fuzzer est capable de fuzzer la première étape du protocole ainsi que la deuxième. Un fuzzer plus efficace aurait été capable d'aller fuzzer les

autres états du protocole, ce sera certainement une des prochaines améliorations de ce fuzzer.

Détection de dysfonctionnement L'architecture de fuzzing étant en *boite noire*, la seule solution est d'interroger l'implémentation IKEv2 serveur afin d'observer son comportement face à des stimuli dont la réponse est connue. Le protocole IKEv2 fonctionnant au-dessus du protocole UDP (non connecté), il est nécessaire d'envoyer un stimulus qui sera compris par l'implémentation IKEv2 auditée. Ce stimulus sera lancé avant chaque test de fuzzing pour valider que l'implémentation IKEv2 auditée est active avant le passage des tests unitaires, et après chaque test de fuzzing afin de valider que l'implémentation IKEv2 auditée est toujours active. Ceci est valable en particulier pour la phase `IKE_SA_INIT`. Concernant la phase `IKE_AUTH`, c'est plus compliqué car l'implémentation IKEv2 a des mécanismes de vérification de connections IKEv2 à moitié ouvertes (car elles ne sont pas refermées par le fuzzer). En effet, après l'envoi dans un certain laps de temps de plusieurs requêtes invalides, le serveur demande un *cookie* à chaque nouvelle requête `IKE_SA_INIT`. Le fuzzer vérifie donc que Racoon2 a réussi à envoyer une requête `IKE_SA_INIT` contenant un *cookie* valide. Une erreur est levée si cela n'est pas le cas.

9.4 Retour d'expérience

Tab. 4. Résultats fuzzing d'implémentations IKEv2

Implémentation	**IKE_SA_INIT**	**IKE_AUTH**
StrongSWAN 4.2.6	1	0
Implémentation 1 Version 1	Plusieurs	Plusieurs
Implémentation 1 Version 2	0	0
Implémentation 2	0	0
Implémentation 3	0	0
Implémentation 4	0	0
Racoon2	À faire	À faire
OpenIKEv2	À faire	À faire

Résultats Dans le tableau 4, nous présentons le nombre de tests de fuzzing déclenchant entraînant un dysfonctionnement de la part de l'implémentation IKEv2 évaluée. À noter que plusieurs tests peuvent déclencher une seule et même vulnérabilité. Aujourd'hui, seule une investigation manuelle permet de réaliser cette analyse car les moyens techniques sont limités en *boite noire*.

À noter que pour retrouver la vulnérabilité sur StrongSWAN 4.2.6, cela n'était pas possible dans la version initiale du fuzzer qui avait été développée. En effet, dans la figure 12, nous constatons l'utilisation de la primitive `s_random()` qui comme son nom l'indique renvoie une chaîne de caractères aléatoire pour chaque test de fuzzing. Or, la vulnérabilité est déclenchée par une chaîne bien spécifique, à savoir, lorsqu'elle est nulle. Il eut été possible de modifier le modèle pour retrouver cette vulnérabilité, cependant, il est apparu plus générique de modifier Sulley afin d'intégrer des valeurs particulières dans la primitive `s_random()`.

Note 3. Ces résultats sont incomplets dans la version actuelle de cet article, nous espérons pouvoir fournir des résultats plus conséquents lors de la présentation et dans la version électronique de l'article qui sera publiée sur le site de la conférence.

Limitations Un fuzzer, même évolué, présente généralement un ensemble de limitations. Les principales limitations de ce fuzzer sont :

- les champs fuzzés le sont de manière séquentielle (pas de fuzzing de plusieurs champs à fuzzer sur un même test),
- tous les états du protocole IKEv2 ne sont pas atteints,
- l'implémentation de la machine à état elle-même n'est pas fuzzée,
- la détection de dysfonctionnement est perfectible et il serait intéressant d'analyser les remontées d'erreurs renvoyées par des messages de notification IKEv2 avec le champ Notify,
- la difficulté de corréler les tests déclenchant une vulnérabilité à la vulnérabilité elle-même,
- les vulnérabilités déclenchées par une suite de tests sont difficiles à analyser et rejouer.

Malgré ces limitations, le fuzzer a permis de redécouvrir une vulnérabilité publique (CVE-2008-4551) et de trouver plusieurs autres vulnérabilités dans une implémentation propriétaire. Cette implémentation propriétaire a été corrigée afin de ne plus présenter les vulnérabilités qui avaient été découvertes précédemment.

10 Recherche de vulnérabilités dans les implémentations EAP, EAP-SIM et EAP-AKA

10.1 Contexte et enjeux

Les implémentations de ces protocoles sont aussi accessibles dans les premiers échanges à tout utilisateur non authentifié. Les erreurs d'implémentation peuvent se

retrouver à la fois sur la passerelle IKEv2 qui doit transmettre les paquets EAP dans des champs du protocole AAA (RADIUS ou Diameter) et sur le serveur AAA qui doit implémenter le protocole EAP et la (ou les) méthode(s) obligatoire(s) pour faire fonctionner l'architecture.

La encore, toute faille de sécurité présente dans un de ces éléments peut entraîner un déni de service déterministe ou de l'exécution de code arbitraire à distance. À noter que les investigations sur l'exploitabilité d'une éventuelle faille de sécurité découverte sont dans ce cas plus faciles car les serveurs RADIUS peuvent fonctionner sur des systèmes tels que des Microsoft Windows ou des *nix classiques.

10.2 Description des protocoles EAP, EAP-SIM et EAP-AKA

Le protocole EAP avait été initialement conçu pour le protocole Point-to-Point Protocol (PPP). C'est un protocole de transport d'authentification qui propose de se reposer sur des méthodes d'authentification qui peuvent être extrêmement variées : EAP-MD5, EAP-TLS [56], EAP-SIM... Ce protocole a été une première fois intégré dans la norme IEEE 802.1X en 2001 et a ensuite rejoint IEEE 802.11i [57] via une adaptation de IEEE 802.1X. Ce n'est qu'à partir de la démocratisation des réseaux Wi-Fi qu'EAP est devenu « à la mode »car il est aujourd'hui utilisé PPP, 802.1X, 802.11i, 802.16e, IKEv2, SIP...

Les méthodes EAP-SIM et EAP-AKA permettent de réutiliser les crédences d'authentification 2G et 3G indépendamment de la couche de transport sous-jacente (par exemple 802.1X ou IKEv2) car la couche intermédiaire est le protocole EAP (ce qui est visible de la partie méthodes EAP). Ceci est un grand avantage en termes de souplesse et flexibilité d'utilisation, c'est certainement une des raisons du succès de ce protocole. Les méthodes EAP-SIM et EAP-AKA permettent toutes les deux une authentification mutuelle du client (la possession d'une carte SIM/USIM et du bon code PIN) et du réseau (la possession d'un secret dans une base centrale). À noter que l'authentification GSM à l'origine ne permet pas l'authentification du réseau alors que l'authentification par la méthode EAP-SIM le permet. Ceci est un élément capital dans l'utilisation de ces technologies au-dessus de réseaux Wi-Fi (où réaliser des attaques de type homme du milieu ne requiert pas d'équipements coûteux).

Les échanges EAP-SIM sont décrits dans la figure 13 et les échanges EAP-AKA sont décrits dans la figure 14. Nous constatons dans ces échanges l'utilisation d'aléas `AT_NONCE_MT` et `AT_RAND` pour EAP-SIM qui sont utilisés pour réaliser l'authentification mutuelle, et l'utilisation d'un aléa `AT_RAND` et d'un jeton d'authentification réseau `AT_AUTN` pour EAP-AKA qui sont aussi utilisés pour l'authentification mutuelle.

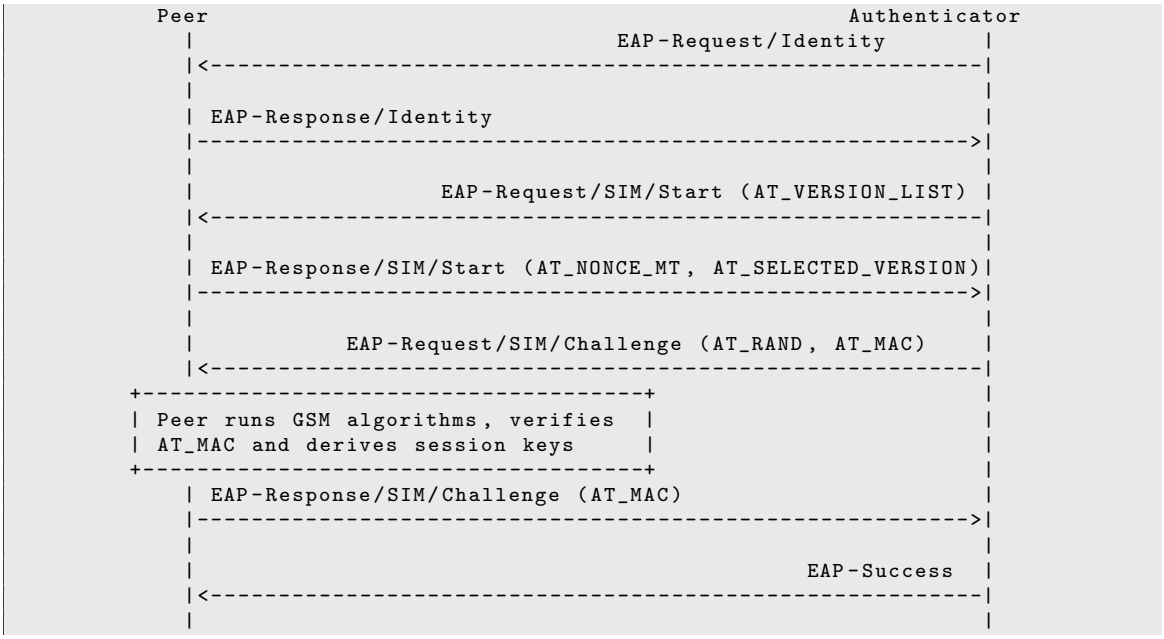

Fig. 13. Authentification EAP-SIM complète (source : RFC 4186)

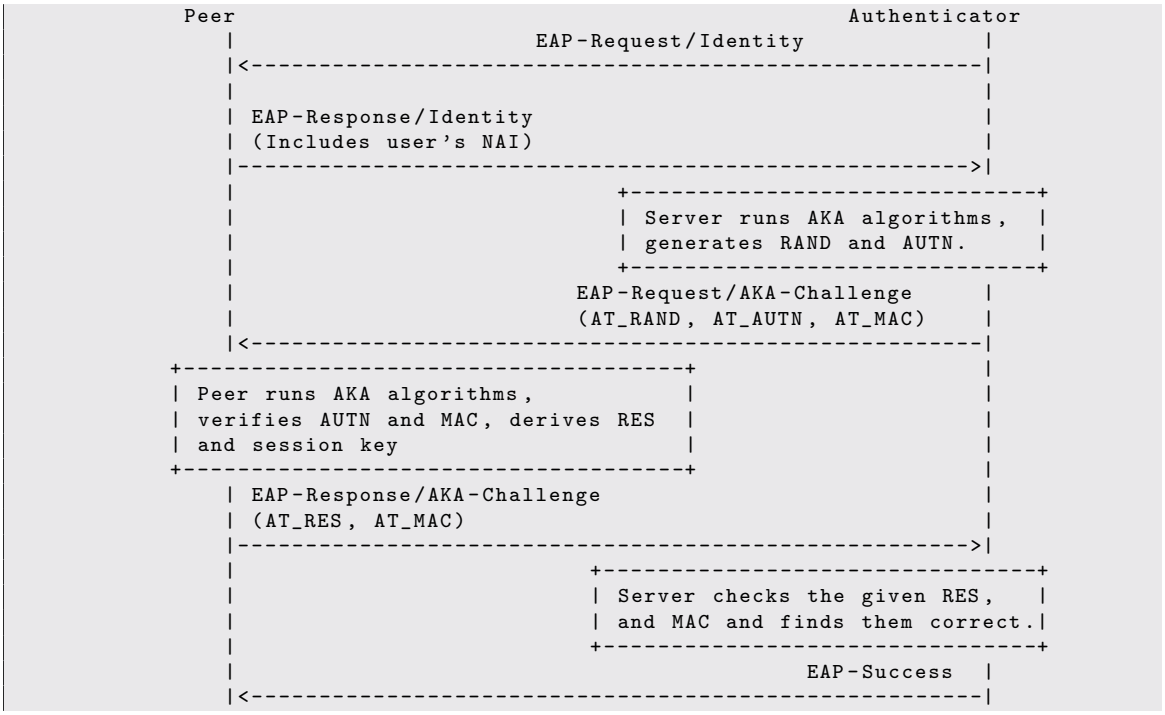

Fig. 14. Authentification EAP-AKA complète (source : RFC 4187)

10.3 Choix techniques

Conditions techniques Elles définissent les choix techniques qui seront réalisés. Dans ce cas, nous nous sommes fixés la possibilité de réaliser les tests soit en *boite noire*, soit en ayant accès à la machine hébergeant le serveur RADIUS. Par conséquent, nous avons implémenté deux méthodes différentes pour détecter des dysfonctionnements. Concernant le choix entre la génération des tests par mutations ou par modélisation de protocole, nous avons opté pour la seconde car certains champs cryptographiques sont utilisés dans les paquets que nous devons envoyer et il est fort probable que ces champs soient vérifiés avant toute analyse de la sémantique des champs EAP fuzzés.

À noter aussi que nous avons choisi de développer un fuzzer d'implémentations serveurs EAP/EAP-SIM/EAP-AKA. Nous considérons que les risques sont bien moins importants sur les implémentations clientes IKEv2 car les impacts (gravité) sont plus faibles pour l'opérateur mais cela n'enlève rien à l'intérêt technique car il apparaît probable de trouver des choses intéressantes pour ceux qui auront la volonté de rechercher dans cette voie.

Les implémentations EAP des serveurs RADIUS sont atteignables en tant que client IKEv2 (ou 802.1X dans le cas du Wi-Fi). La première idée aurait été de développer un fuzzer EAP au-dessus de protocoles comme IKEv2 et 802.1X, mais cela a un impact fort sur la précision des résultats : l'élément relais (aussi appelé l'authenticator) peut effectuer des traitements et donc ne pas transmettre les paquets EAP fuzzés vers l'implémentation EAP à fuzzer. Nous avons donc opté pour une autre solution, à savoir, agir en tant que client RADIUS directement et donc acter à la fois en tant que client EAP/EAP-SIM/EAP-AKA au-dessus de RADIUS plutôt qu'IKEv2 ou 802.1X.

Base technique De la même manière que pour la partie sur le fuzzing d'implémentations IKEv2, tous nos développements sont basés sur des briques issues de l'outil Sulley Fuzzing Framework.

Génération des tests Nous nous reposons sur un système de génération de tests par modélisation du protocole. Il a donc fallu décrire complètement les charges contenues dans les trames issues des protocoles `EAP`, `EAP-SIM` et `EAP-AKA`. Ces protocoles sont bien moins complexes que le protocole IKEv2 dans la construction des messages et présentent comme avantages (pour le développeur du fuzzer) de ne pas chiffrer le contenu des paquets. Nous avons dû aussi modéliser l'intégration des paquets EAP dans les attributs RADIUS correspondants et les intégrer au niveau RADIUS (utilisation de mécanismes cryptographiques par secret partagé entre le client et le

serveur). Dans les figures 15 et 16 nous montrons un exemple de spécification de la modélisation d'un en-tête `EAP` et `EAP-Identity` grâce à des primitives de Sulley comme `s_size()` et `s_string()` qui permettront alors de fuzzer ces champs grâce aux heuristiques de Sulley. Nous avons décrit un ensemble d'attributs EAP qui sont nécessaires par rapport à ce qui est décrit dans les RFC 4186 et RFC 4187.

```
def EAP_begin(block, code, identifier, type, subtype=None):
    """Debut de chaque bloc EAP"""

    s_byte(code, fuzzable=False)                                    # Code
    s_byte(identifier, fuzzable=False)                              # Identifier
    s_size(block, length=2, endian='>', fuzzable=True)              # Length
    s_byte(type, fuzzable=False)                                    # Type

    if subtype:
        s_byte(subtype, fuzzable=False)                             # Subtype
        s_short(0x0000, fuzzable=False)                             # Reserved
```

Fig. 15. Exemple de définition d'un champ de type en-tête EAP

```
s_initialize("Identity")
if s_block_start("no_encoder_Identity", encoder=eap.radius_encoder):
    if s_block_start("truncate", truncate=True):
        if s_block_start("Identity", truncate=True):
            EAP_begin("Identity", RESPONSE, 0, IDENTITY)            # Header EAP

            s_string(USERNAME, fuzzable=True)                       # Identity
        s_block_end()
    s_block_end()
s_block_end()
```

Fig. 16. Exemple de définition d'un champ de type EAP-Response/Identity

Gestion des états Nous nous reposons sur les standards IETF afin de décrire les états atteignables par un client qui n'aurait pas de crédences d'authentification. Tous les états sont atteignables car la vérification de l'authentification du client (par le serveur que ce soit en EAP-SIM ou EAP-AKA) est effectuée dans le dernier envoi de paquet EAP-Response. Ceci est très important pour essayer d'atteindre un maximum de parties du code de l'implémentation du serveur RADIUS.

```
s_initialize("SIM/Challenge")
if s_block_start("no_encoder_SIM/Challenge", encoder=eap.radius_encoder):
  if s_block_start("truncate", truncate=True):
    if s_block_start("SIM/Challenge"):
      EAP_begin("SIM/Challenge", RESPONSE, 1, EAP_SIM, CHALLENGE) # Header EAP

      SIM_attribute("AT_MAC", "\x00" * 18)                        # AT_MAC
      SIM_attribute("AT_IV", "\x00" * 20, reserved=True)          # AT_IV
      SIM_attribute("AT_RESULT_IND", "", reserved=True)           # AT_RESULT_IND
      SIM_attribute("AT_ENCR_DATA", "\x00" * 20, reserved=True)   # AT_ENCR_DATA
      SIM_attribute("AT_PADDING", "")                             # AT_PADDING
    s_block_end()
  s_block_end()
s_block_end()
```

Fig. 17. Exemple de définition d'un champ de type EAP-Response/SIM/Challenge

Détection de dysfonctionnement Même si nous privilégions le mode *boite noire*, le fait que les serveurs RADIUS soient utilisables sur des systèmes d'exploitation relativement « ouverts », il est possible d'utiliser les fonctions de Sulley pour attacher un débogueur (dans notre cas `PyDBG` qui fait partie de la suite PaiMei [58]) au processus qui sera audité. Nous avons donc à la fois développé une technique de détection d'erreur d'implémentation par envoi de stimulus sur la pile EAP distante (au-dessus de RADIUS) et réutilisé les fonctions de Sulley pour éventuellement attacher un débogueur lorsque cela est possible.

Le premier mode est nécessaire en *boite noire* et peut présenter un problème lorsque le processus est redémarré automatiquement en cas de plantage tandis que le deuxième mode est bien plus fiable et précis car il remontera des informations sur le plantage grâce au débogueur.

10.4 Retour d'expérience

Tab. 5. Résultats fuzzing d'implémentations EAP/EAP-SIM/EAP-AKA

Implémentation	EAP	EAP-SIM	EAP-AKA
Cisco Secure ACS 4.1	Plusieurs dizaines	N/A	N/A
Implémentation 1	0	0	À faire
Implémentation 2	0	0	À faire
FreeRADIUS	0	0	N/A

Résultats Dans le tableau 5, nous présentons le nombre de tests de fuzzing déclenchant entraînant un dysfonctionnement de la part de l'implémentation IKEv2 évaluée. À noter que plusieurs tests peuvent déclencher une seule et même vulnérabilité. Dans le cas de la vulnérabilité découverte (CVE-2008-2441) il s'agit d'une seule et même vulnérabilité déclenchée par plusieurs tests de fuzzing. Ceci a pu être identifié grâce à une analyse par les résultats de `PyDBG` et par une analyse des paquets émis (longueur avertie dans le champ longueur du paquet EAP).

Note 4. Ces résultats sont incomplets dans la version actuelle de cet article, nous espérons pouvoir fournir des résultats plus conséquents lors de la présentation et dans la version électronique de l'article qui sera publiée sur le site de la conférence.

Limitations Un fuzzer, même évolué, présente généralement un ensemble de limitations. Les principales limitations de ce fuzzer sont :

- les champs fuzzés le sont de manière séquentielle (pas de fuzzing de plusieurs champs à fuzzer sur un même test),
- tous les états du protocole EAP-AKA ne sont pas atteints (nécessitent des aspects cryptographiques),
- l'implémentation de la machine à état elle-même n'est pas fuzzée,
- la difficulté de corréler les tests déclenchant une vulnérabilité à la vulnérabilité elle-même,
- les vulnérabilités déclenchées par une suite de tests sont difficiles à analyser et rejouer.

Malgré ces limitations, le fuzzer a permis de découvrir une vulnérabilité sur une implémentation EAP d'un serveur RADIUS (CVE-2008-2441).

11 Conclusions

Nous nous sommes efforcés dans cet article de présenter quelques problématiques de sécurité des architectures de Convergence Fixe-Mobile. Nous avons choisi de nous orienter vers la recherche de vulnérabilités dans les implémentations IKEv2 et EAP/EAP-SIM/EAP-AKA dans l'espoir de trouver des vulnérabilités en amont afin de les faire corriger au plus vite par les constructeurs concernés. Les résultats montrent, que malgré la théorique profondeur des fuzzers développés, nous n'avons pas eu les résultats habituels du fuzzing. En effet, pour ceux qui se rappellent du fuzzing de drivers 802.11 ou autres (comme par exemple les lecteurs multimédia), il suffisait de chercher un peu pour trouver rapidement des vulnérabilités (qui plus est intéressantes[26]). Nous serions tenté d'affirmer que ces implémentations qui sont

[26] potentiellement exploitables.

relatives à des produits sécurité (et donc développés par des sociétés vendant des produits de sécurité) sont peut-être mieux développés que les autres (ou pas !). Sur la partie IKEv2, nous avons trouvé plusieurs vulnérabilités (au minimum deux vulnérabilités différentes) sur une seule implémentation parmi un total de cinq implémentations testées. Sur la partie EAP, nous avons trouvé une seule vulnérabilité sur quatre implémentations testées. Le bilan prouve le bon fonctionnement du fuzzer mais laisse toutefois un peu sur sa faim... Nous allons continuer à améliorer certaines parties de nos fuzzers en particulier sur les flots de connexions, la machine à états et la détection de dysfonctionnements, et dans tous les cas, seul l'avenir nous dira si nous avions raison ou pas !

12 Remerciements

Nous tenons à remercier Gabriel Campana pour ses recherches sur le fuzzing et ses développements sur le sujet, Maryline Maknavicius pour avoir joué le rôle de relecteur SSTIC, Franck Veysset et Sébastien Nguyen Ngoc pour leurs relectures attentives.

13 Aide financière

Ces travaux ont fait l'objet d'une aide financière de l'Agence Nationale de la Recherche (ANR) via le projet Future Internet Vulnerability Assessment, Monitoring and Prevention (VAMPIRE) ayant comme référence ANR-08-VERS-017 [63].

Références

1. RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, http://tools.ietf.org/rfc/rfc4306.txt, 2005.
2. RFC 3748, *Extensible Authentication Protocol (EAP)*, http://tools.ietf.org/rfc/rfc3748.txt, 2004.
3. RFC 2401, *Security Architecture for the Internet Protocol*, http://tools.ietf.org/html/rfc2401, 1998.
4. RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, http://tools.ietf.org/html/rfc5246, 2008.
5. Journal officiel du 20 avril 2007, *Implémenter*, http://www.journal-officiel.gouv.fr/verifier/getpdf.php?fic=../publication/2007/0420/joe_20070420_0093_0084.pdf.sig, 2007.
6. Radiocom 2000, http://fr.wikipedia.org/wiki/Radiocom_2000.
7. Tatoo, *Tatoo : « votre tribu garde le contact avec vous. »*, http://fr.wikipedia.org/wiki/Pager.
8. Bi-Bop, *Bi-Bop*, http://fr.wikipedia.org/wiki/Bibop.
9. Bluetooth, *Bluetooth Special Interest Group*, http://www.bluetooth.org.

10. IEEE 802.11, *Local and Metropolitan Area Networks, Specific Requirements, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, http://standards.ieee.org/getieee802/download/802.11-2007.pdf, 1997-2007.

11. IEEE 802.16-2004, *IEEE Standard for Local and metropolitan area networks Part 16 : Air Interface for Fixed Broadband Wireless Access Systems*, http://standards.ieee.org/getieee802/download/802.16-2004.pdf, 2004.

12. IEEE 802.16e-2005, *IEEE Standard for Local and metropolitan area networks Part 16 : Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands*, http://standards.ieee.org/getieee802/download/802.16e-2005.pdf, 2005.

13. LTE, Long Term Evolution, http://www.3gpp.org/article/lte.

14. IPHOBAC, *Integrated Photonic mm-Wave Functions for Broadband Connectivity*, http://www.iphobac-survey.org.

15. ARCEP, *Suivi des Indicateurs Mobiles*, http://www.arcep.fr/fileadmin/reprise/\observatoire/obs-mobile/2008/t4-2008/sim-t4-2008.pdf.

16. Union Internationale des Télécommunications, *Un nouvel indice UIT de développement des TIC : comparaison entre 154 pays*, http://www.itu.int/newsroom/press_releases/2009/07-fr.html, 2009.

17. Barkan, Elad ; Eli Biham ; Nathan Keller, *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication*, http://cryptome.org/gsm-crack-bbk.pdf, 2003.

18. David Hulton, *Intercepting GSM Traffic*, http://blog.washingtonpost.com/securityfix/shmoocon-Feb08-gsm.pdf, 2008.

19. 3GPP, *Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS ; Document 1 : A5/3 and GEA3 Specifications.*, http://www.gsmworld.com/documents/a5_3_and_gea3_specifications.pdf.

20. ETSI, *Universal Mobile Telecommunications System (UMTS) ; Specification of the 3GPP confidentiality and integrity algorithms ; Document 2 : Kasumi specification*, http://www.etsi.org/website/document/algorithms/ts_135202v070000p.pdf.

21. Ettus Research LLC, *Universal Software Radio Peripheral*, http://www.ettus.com.

22. The GNU Software Radio, *GNU Radio*, http://www.gnuradio.org.

23. Kestrel Signal Processing, Inc., *The OpenBTS Project*, http://openbts.sourceforge.net/.

24. Kestrel Signal Processing, Inc., *The OpenBTS Project Wiki*, http://gnuradio.org/trac/wiki/OpenBTS.

25. Wikipedia, *Um interface*, http://en.wikipedia.org/wiki/Um_Interface

26. 3GPP TS 33.102, *3G Security ; Security Architecture*, http://www.3gpp.org/ftp/Specs/archive/33_series/33.102/33102-820.zip, 2008.

27. Natacha Mach et Franck Veysset, *Convergence Fixe-Mobile : UMA (Unlicensed Mobile Access) sur le devant de la scène ?*, http://ed-diamond.com/feuille_misc31/index.html, 2007.

28. Sandro Grech et Pasi Eronen, *Implications of Unlicensed Mobile Access (UMA) for GSM security*, http://ieeexplore.ieee.org/iel5/10695/33755/01607554.pdf, 2005.

29. 3GPP TS 23.234, *3GPP system to Wireless Local Area Network (WLAN) interworking ; System description (Release 8)*, http://www.3gpp.org/ftp/Specs/archive/23_series/\23.234/23234-800.zip, 2008.

30. 3GPP TS 33.234, *3G Security ; Wireless Local Area Network (WLAN) interworking security (Release 8)*, http://www.3gpp.org/ftp/Specs/archive/33_series/\33.234/33234-810.zip, 2008.

31. RFC 3261, *SIP : Session Initiation Protocol*, http://tools.ietf.org/html/rfc3261, 2222.

32. Yoann Guillot et Julien Tinnès, *System Level Intrusion Prevention System Evaluation Suite and Toolkit*, http://slipfest.cr0.org, 2006.
33. RFC 2409, *The Internet Key Exchange (IKE)*, http://tools.ietf.org/rfc/rfc2409.txt, 1998.
34. 3GPP, *3GPP Service and Systems WG3 : Security*, http://www.3gpp.org/SA3.
35. Barton Miller, *Fuzz Testing of Application Reliability*, http://pages.cs.wisc.edu/~bart/fuzz, 1988-2008.
36. Gadi Evron, *the origin of the word fuzzing*, http://lists.virus.org/fuzzing-0703/msg00014.html.
37. Laurent Butti et Julien Tinnès, *Recherche de vulnérabilités dans les drivers 802.11 par techniques de fuzzing*, http://actes.sstic.org/SSTIC07/WiFi_Fuzzing/, 2007.
38. MISCMAG 39, *Dossier fuzzing*, http://ed-diamond.com/feuille_misc39/index.html, 2008.
39. Codenomicon, *DEFENSICS*, http://www.codenomicon.com.
40. RFC 3748, *Extensible Authentication Protocol (EAP), MD5-Challenge*, http://tools.ietf.org/html/rfc3748, 2004.
41. RFC 3748, *Extensible Authentication Protocol (EAP), Generic Token Card*, http://tools.ietf.org/html/rfc3748, 2004.
42. Draft eap-mschapv2, *Microsoft EAP CHAP Extensions*, http://tools.ietf.org/html/draft-kamath-pppext-eap-mschapv2-02, 2007.
43. The MITRE Corporation, *Common Vulnerabilities and Exposures*, http://cve.mitre.org.
44. Cédric Blancher, *Une faille est un bug comme les autres (ou presque)...* , http://sid.rstack.org/blog/index.php/284-une-faille-est-un-bug-comme-les-autres-ou-presque, 2008.
45. CVE-2008-4551, *fixes a potential DoS attack if a DH value of zero gets processed*, http://wiki.strongswan.org/browser/trunk/src/libstrongswan/plugins/gmp/gmp_diffie_hellman.c?rev=4345.
46. Mu Dynamics, *Mu Dynamics discovers, remediates leading Open Source VPN vulnerability : StrongSwan IKEv2 Denial-of-Service*, http://www.mudynamics.com/news/press/pr091908.html.
47. Benoît Stopin et Laurent Butti, *Unspecified vulnerability in the Extensible Authentication Protocol (EAP) implementation in Cisco IOS...* , http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5651.
48. Gabriel Campana et Laurent Butti, *Cisco Secure ACS 3.x... does not properly handle an EAP Response packet in which the value of the length field exceeds the actual packet length...* , http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-2441.
49. Cédric Blancher et Simon Maréchal, *Packin' the PMK*, http://sid.rstack.org/pres/0810_BACon_WPA2_en.pdf, 2008.
50. IEEE 802.1X-2004, *EEE Standard for Local and metropolitan area networks : Port-Based Network Access Control*, http://standards.ieee.org/getieee802/download/802.1X-2004.pdf, 2004.
51. The Racoon2 Project, *Racoon2*, http://www.racoon2.wide.ad.jp/w/.
52. Wikipedia, *Échange de clés Diffie-Hellman*, http://fr.wikipedia.org/wiki/echange_de_cles_Diffie-Hellman.
53. CVE-2008-1113, *Cisco and Vocera wireless LAN VoIP devices don't check certificates*, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1113, 2008.
54. CVE-2008-1114, *Cisco and Vocera wireless LAN VoIP devices don't check certificates*, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1114, 2008.
55. Pedram Amini, *Sulley Fuzzing Framework*, http://code.google.com/p/sulley/.

56. RFC 5216, *The EAP-TLS Authentication Protocol*, http://tools.ietf.org/html/rfc5216, 2008.

57. IEEE 802.11i, *Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6 : Medium Access Control (MAC) Security Enhancements* , http://standards.ieee.org/getieee802/download/802.11i-2004.pdf, 2004.

58. Pedram Amini, *PaiMei*, http ://code.google.com/p/paimei/.

59. RFC 4186, *Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)*, http://tools.ietf.org/rfc/rfc4186.txt, 2006.

60. RFC 4187, *Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)*, http://tools.ietf.org/rfc/rfc4187.txt, 2006.

61. RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*, http://tools.ietf.org/rfc/rfc2865.txt, 2000.

62. RFC 3588, *Diameter Base Protocol*, http://tools.ietf.org/rfc/rfc3588.txt, 2003.

63. VAMPIRE, *Future Internet Vulnerability Assessment, Monitoring and Prevention*, http://vampire.gforge.inria.fr/, 2008-2011.