Une attaque peut en cacher une autre

François Morris
DSI / CNRS
1 place Aristide Briand
92195 MEUDON

Résumé

À travers l'étude d'incidents réels mais anonymisés, nous décrirons comment répondre à un incident et procéder à l'investigation numérique.

Afin d'illustrer notre propos, nous nous appuierons sur un exemple, en l'occurrence, une défiguration qui exploitait une faille bien connue et relativement ancienne du CMS SPIP.

L'analyse révèlera qu'il y a eu deux attaquants différents, le deuxième lançant son attaque dès que la défiguration a été publiée par le premier.

Nous souhaitons montrer que dans bien des cas l'analyse ne demande pas des moyens importants ni n'exige un haut niveau d'expertise. Une simple recherche dans les journaux peut suffire.

Nous verrons que derrière ce qui peut sembler une défiguration relativement bénigne, l'attaquant a compromis totalement le système en y cachant des portes dérobées lui permettant d'y revenir à sa guise et d'avoir un contrôle total.

Nous décrirons les outils installés par l'attaquant et comment on peut les détecter.

Une étude de la vulnérabilité montrera que celle-ci est triviale à exploiter.

Nous verrons comment une configuration trop laxiste du serveur, a permis à l'attaquant s'étant introduit sur un premier site, de modifier les autres hébergés sur la même machine.

Nous donnerons quelques bonnes pratiques pour limiter les risques de survenue de tels incidents, de l'application des correctifs de sécurité au durcissement des configurations.

Nous donnerons une évaluation des impacts et conséquences de l'incident. Nous aborderons aussi l'aspect juridique avec le dépôt de plainte.

Tout au long de l'exposé nous montrerons qu'au-delà de ce cas particulier on retrouve de façon générale les mêmes éléments dans les nombreux incidents dont nous avons eu connaissance.

.

Mots-clefs

Investigation numérique, réponse à incident, faille, vulnérabilité, exploit, CMS, risque, sécurité, marqueur, IOC

1 Avertissement

Cette description du *modus operandi* d'attaquants est uniquement destinée à montrer un exemple de réponse à incident et à favoriser la mise en place de mesures de protection. En aucun cas elle n'est une incitation à la commission d'une attaque qui tomberait sous le coup des articles 323-1 et suivants du code pénal[1]. Les informations ont été rendues anonymes. Bien que les attaquants revendiquent leur action, nous

[1] http://www.legifrance.gouv.fr/affichCode.do?idSectionTA=LEGISCTA000006149839&cidTexte=LEGITEXT000006070719

n'avons pas voulu les citer, ne serait-ce que pour éviter qu'ils se reconnaissent et améliorent leur méthode.

2 Pierre

Nous avons été alertés de la défiguration d'un site web utilisant le CMS SPIP. La page défigurée contenait des revendications par un certain Pierre[2] qui défend une cause politique. Une recherche sur zone-h[3] a permis de retrouver la page défigurée et l'heure à laquelle elle a été recopiée par zone-h. Cela a permis d'encadrer les recherches.

Ce Pierre revendique sur zone-h de nombreuses défigurations (221 au 03/02/2015 depuis le 31/12/2013).

Face à un incident, il faut toujours garder toutes les hypothèses ouvertes et ne pas avoir de préjugé. Cependant connaissant les attaques à la mode, nous avons tout de suite pensé à l'exploitation de la faille CVE-2013-2118[4]. Cette faille est triviale à exploiter. Elle permet, sans authentification préalable, de créer un compte ayant les privilèges de l'administrateur de SPIP et de recevoir par courriel le mot de passe associé. L'attaque peut facilement être automatisée à l'aide d'un script[5] (une trentaine de lignes en python suffisent). Nous verrons que l'attaquant ne l'a pas fait et s'est contenté d'utiliser un navigateur.

L'exploitation de la faille est d'une facilité déconcertante car il suffit de se connecter à l'URL */spip.php?page=identifiants&mode=0minirezo* et de remplir le formulaire pour recevoir par courriel le mot de passe (cf. figure 1).

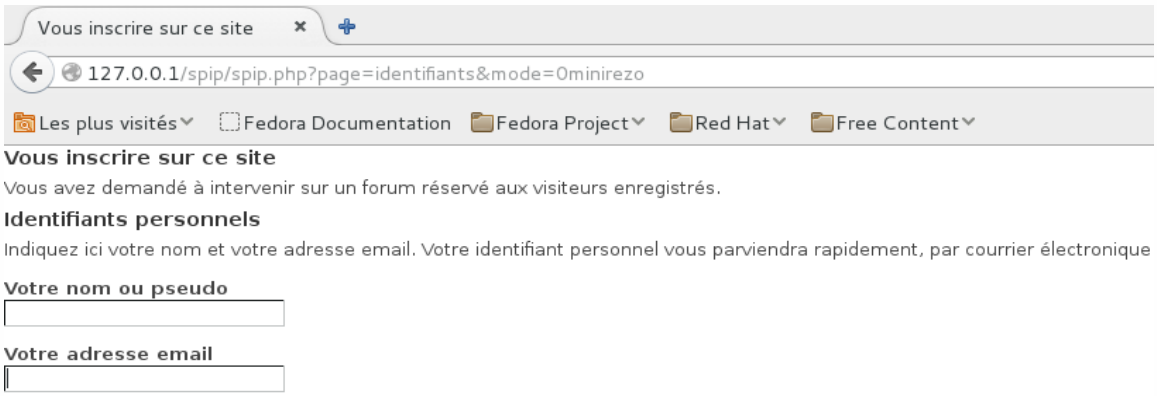

Figure 1 - Formulaire d'inscription

Aussi avons-nous commencé par rechercher[6] dans les journaux les requêtes contenant la chaîne *0minirezo* qui correspond au mode administrateur de SPIP. Évidemment certaines requêtes peuvent être parfaitement légitimes.

Dans les extraits de journaux qui suivent, outre l'anonymisation, des enregistrements seront supprimés et certains champs comme le User-agent seront abrégés, afin d'en faciliter la lecture.

En retenant les occurrences les plus anciennes de ***0minirezo*** on trouve :

```
10.1.2.20 - - [24/Jan/2015:01:36:14 +0100] "GET
/spip.php?page=identifiants&mode=0minirezo HTTP/1.1" 200 4200 "-"
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like
```

[2] Les noms ont été changés
[3] http://www.zone-h.org
[4] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-2118
[5] On peut en trouver facilement un sur Internet.
[6] Nul besoin d'outils élaborés, un simple grep ou zgrep a suffi.

```
Gecko) Chrome/32.0.1700.76 Safari/537.36"
10.1.2.20 - - [24/Jan/2015:01:36:14 +0100] "GET
/prive/spip_style.css HTTP/1.1" 200 3110
"http://www.site_defigure.fr/spip.php?page=identifiants&mode=0mini
rezo" "Mozilla/5.0 (Windows NT 5.1)"
[…]
10.1.2.20 - - [24/Jan/2015:01:36:26 +0100] "POST
/spip.php?page=identifiants&mode=0minirezo HTTP/1.1" 200 271
"http://www.site_defigure.fr/spip.php?page=identifiants&mode=0mini
rezo" "Mozilla/5.0"
```

Cela semble bien la signature de l'attaque décrite ci-dessus. D'après une recherche Google *l'User-Agent* correspondrait à un navigateur Chrome sous Windows XP. Il faut rester prudent car il n'y a rien de plus facile que d'usurper un *User-Agent*.

L'envoi du message contenant le mot de passe est confirmé par le fichier <répertoire du site>/tmp/mail.log qui contient le journal des messages envoyé par SPIP.

```
Jan 24 01:36:26 10.1.2.20 (pid 32341) mail pierre@fai_1
[UMR nnnn - xxx] Identifiants personnels
From: pierre@fai_1 (pierre at fai_1)
Reply-To: pierre@fai_1 (pierre at fai_1)
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
MIME-Version: 1.0
```

Une consultation de la base whois indique que l'adresse 10.1.2.20 est attribuée à un opérateur situé à l'étranger. Nous effectuons alors un nouvelle recherche dans les journaux avec comme critère l'adresse IP utilisée ci-dessus.

```
10.1.2.20 - - [24/Jan/2015:01:36:00 +0100] "GET
/spip.php?article1120 HTTP/1.1" 200 18150
"http://www.bing.com/search?q=madagascar+spip.php&go=Valider&qs=ds
&first=161&FORM=PERE4" "Mozilla/5.0"
[...]
10.1.2.20 - - [24/Jan/2015:01:36:14 +0100] "GET
/spip.php?page=identifiants&mode=0minirezo HTTP/1.1" 200 4200 "-"
"Mozilla/5.0"
[...]
10.1.2.20 - - [24/Jan/2015:01:36:26 +0100] "POST
/spip.php?page=identifiants&mode=0minirezo HTTP/1.1" 200 271
"http://www.site_defigure.fr/spip.php?page=identifiants&mode=0mini
rezo" "Mozilla/5.0"
[...]
10.1.2.20 - - [24/Jan/2015:01:36:35 +0100] "GET / HTTP/1.1" 200
24971 "-" "Mozilla/5.0"
```

Il est intéressant de noter que la première requête a pour *Referer* le moteur de recherche *Bing*. L'attaquant cherchait manifestement un site SPIP d'où la présence de spip.php dans sa recherche.

La deuxième requête n'a pas de *Referer* ce qui suggère que l'attaquant a entré directement l'URL dans le navigateur. L'attaquant poursuit sa navigation sur le site.

Il faut attendre que l'attaquant récupère son mot de passe et se connecte à nouveau. Comme nous soupçonnons que l'attaquant ne va pas se contenter de lire des pages mais va vouloir déposer du contenu nous recherchons les requêtes POST postérieures au début de l'attaque. En effet le remplissage d'un formulaire ou l'envoi d'un fichier passent nécessairement par des requêtes POST. La première trouvée est une bonne candidate, surtout que l'adresse IP correspond au même FAI que précédemment.

```
10.1.17.47 - - [24/Jan/2015:12:48:39 +0100] "POST
/spip.php?page=login&url=%2Fecrire%2F HTTP/1.1" 302 266
"http://site_defigure.fr/spip.php?page=login&url=%2Fecrire%2F"
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/39.0.2171.71 Safari/537.36"
```

En effet dans SPIP, la présence d'*ecrire* dans l'URL indique que l'on ajoute ou modifie du contenu dans le CMS. La présence de *login* suggère que l'on se trouve à une étape d'authentification ce qui est cohérent avec le fait qu'une modification ne peut être effectuée que par un utilisateur authentifié. De plus l'adresse IP appartient au même FAI que précédemment. Il faut noter que l'*User-Agent* n'est plus le même. L'attaquant a-t-il changé de machine ? A-t-il fait entre temps une mise à jour de son navigateur, ou bien est-ce une autre personne (ce qui pourrait suggérer que l'on est en présence d'un réseau) ? C'est difficile à dire. On va alors rechercher dans les journaux tout ce qui concerne cette adresse 10.1.17.47.

En parcourant les requêtes provenant de cette adresse IP, on en trouve une concernant l'authentification (présence de la variable *var_login*).

```
10.1.17.47 - - [24/Jan/2015:12:48:37 +0100] "GET
/spip.php?page=informer_auteur&var_login=pierre&var_compteur=14221
00015902 HTTP/1.1" 200 134
"http://site_defigure.fr/spip.php?page=login&url=%2Fecrire%2F"
"Mozilla/5.0"
```

La présence de l'identifiant *pierre* est très intéressante puisque la défiguration a été revendiquée sur zone-h par un certain *Pierre* qui défend une cause politique. Une recherche Google *'Hacked by Pierre' site:www.site-defigure.fr* permet de retrouver des dizaines de pages défigurées.

Nous nous intéressons alors aux requêtes POST provenant de cette IP.

```
10.1.17.47 - - [24/Jan/2015:12:52:28 +0100] "POST
/ecrire/?exec=articles_edit HTTP/1.1" 302 326
"http://site_defigure.fr/ecrire/?exec=articles_edit&id_article=111
6" "Mozilla/5.0"
10.1.17.47 - - [24/Jan/2015:12:52:28 +0100] "POST
/ecrire/?exec=configuration HTTP/1.1" 302 336
"http://site_defigure.fr/ecrire/?exec=configuration" "Mozilla/5.0"
```

Sans être un spécialiste de SPIP, il est évident qu'il s'agit de modifier du contenu, des articles ou même la configuration. Ce n'est pas parce que les requêtes POST se sont terminées avec un code 302, qu'elles n'ont pas été effectuées. En effet un code 302 (déplacé de façon temporaire) signifie seulement que le navigateur doit se connecter à l'URL précisée dans la réponse.

Nous n'avons pas trouvé de traces permettant de montrer que l'attaquant ait fait plus que modifier du contenu en utilisant les fonctionnalités de SPIP. En particulier, il n'a pas réussi à modifier la page d'accueil, ce qui explique pourquoi la revendication ne concernait qu'une page située dans le répertoire de téléchargement (/IMG). C'est d'ailleurs général pour les défigurations de cet attaquant qui d'après zone-h affectent assez rarement la page d'accueil.

Le *modus operandi* de Pierre ne semble pas montrer un haut niveau d'expertise. Il exploite une vulnérabilité bien connue de SPIP pour se créer un compte avec des privilèges élevés ce qui lui permet de déposer sur le site ses pages de revendication. Il est probable qu'il n'utilise aucun outil spécifique, uniquement son navigateur.

3 CVE-2013-2118

Revenons sur la vulnérabilité exploitée. Elle a fait l'objet du CVE-2013-2118 et un correctif critique a été publié le 23/05/2013[7] Si on regarde la correction effectuée dans le code de SPIP (cf. figure 2), on constate qu'il a été ajoutée une vérification.

```
1544 1544       default:
1545            if ($mode AND $mode == addslashes($mode))
1546              return $mode;
     1545       if ($mode AND $mode == addslashes($mode)){
     1546            include_spip("inc/autoriser");
     1547            return autoriser("inscrireauteur",$mode)?$mode:"";
```

Figure 2 - Correction effectuée dans SPIP

Le code source[8] de la fonction modifiée montre qu'aucun contrôle n'était effectué lorsque l'on utilisait le mode *0minirezo* qui correspond à celui de l'administrateur.

```
// Utile pour le formulaire d'inscription.
// Si un mode est fourni, verifier que la configuration l'accepte.
// Si mode inconnu laisser faire, c'est une extension non std
function tester_config($id, $mode='') {
  $s = array_search($mode, $GLOBALS['liste_des_statuts']);
  switch ($s) {
  case 'info_redacteurs' :
    return (($GLOBALS['meta']['accepter_inscriptions'] == 'oui') ?
$mode : '');
  case 'info_visiteurs' :
    return (($GLOBALS['meta']['accepter_visiteurs'] == 'oui' OR
       $GLOBALS['meta']['forums_publics'] == 'abo') ? $mode : '');
  default:
    if ($mode AND $mode == addslashes($mode)){
      return $mode
    }
    if ($GLOBALS['meta']["accepter_inscriptions"] == "oui")
      return $GLOBALS['liste_des_statuts']['info_redacteurs'];
     if ($GLOBALS['meta']["accepter_visiteurs"] == "oui")
      return $GLOBALS['liste des statuts']['info visiteurs'];
    return '';
  }
}
```

[7] http://contrib.spip.net/SPIP-3-0-9-2-1-22-2-0-23-corrections-de-bug-et-faille?lang=fr
[8] https://core.spip.net/projects/spip/repository/revisions/20541/entry/branches/spip-2.1/ecrire/inc/filtres.php

Le formulaire prévu pour se créer un compte visiteur pouvait donc être facilement détourné pour se créer un compte administrateur. On en déduit aisément qu'il était possible de se faire envoyer un mot de passe sans contrôle préalable.

Cette vulnérabilité correspond à *Violation de gestion d'authentification et de session*, le deuxième risque du top 10 de l'OWASP[9].

4 Paul

Nous nous étions arrêtés au premier *0minirezo* trouvé. Il y en a d'autres comme

```
10.2.216.65 - - [24/Jan/2015:13:36:38 +0100] "POST
/spip.php?page=identifiants&mode=0minirezo HTTP/1.1" 200 271
"http://site_defigure.fr/spip.php?page=identifiants&mode=0minirezo
" "Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101
Firefox/31.0"
```

Nous effectuons alors une recherche sur l'adresse IP 10.2.216.65 qui, coïncidence ou pas, appartient au même opérateur étranger que celui utilisé par Pierre et surprise la première requête est

```
10.2.216.65 - - [24/Jan/2015:13:36:20 +0100] "GET / HTTP/1.1" 200
24652 "http://www.zone-h.com/mirror/id/nnnnnnnn" "Mozilla/5.0
(Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.
```

Le *referer* (lien vers la page à partir de laquelle la requête a été faite) est justement la copie effectuée par zone-h de la page défigurée par Pierre. On peut penser qu'un autre pirate s'est dit que ce site était vulnérable et qu'il pouvait l'attaquer à son tour. Il est aussi possible que cela soit le même attaquant ou un membre de son réseau qui se soit dit « la défiguration c'est bien joli maintenant passons aux choses sérieuses avec une attaque plus élaborée ayant de plus grands impacts ».

C'est une constatation très générale, dans la grande majorité des cas d'intrusion que nous avons pu approfondir nous avons trouvé les traces de plusieurs attaquants voire de luttes entre attaquants.

L'analyse a montré que Paul s'est contenté de déposer des portes dérobées sur cette machine mais n'a pas effectué de défiguration. Par la suite nous étudierons le *modus operandi* de Paul sur une autre attaque plus intéressante et de plus grande envergure que nous avons eu aussi à analyser. L'analyse s'est déroulée de la même façon que pour Pierre, aussi nous ne la détaillerons pas. Il s'agit toujours de recherche dans les journaux et les fichiers modifiés aux dates concernées.

Paul a utilisé exactement la même méthode que Pierre pour obtenir un accès administrateur à SPIP.

La première trace d'une intrusion de Paul remonte au 01/01/2015 mais elle peut être plus ancienne car nous ne disposons pas hélas de journaux antérieurs. C'est pourquoi il faut rappeler la recommandation de conserver des journaux suffisamment complets sur une année. Il semble qu'à cette date Paul se soit contenté de vérifier que le site était vulnérable afin d'y revenir le jour voulu.

L'analyse d'autres défigurations revendiquées par Paul a montré que celui-ci utilisait souvent, comme cette fois-ci, l'adresse paul2@fai_2 associée aussi à l'identifiant jean. Une recherche Google montre que cette adresse paul2@fai_2 est utilisée depuis longtemps puisque on la retrouve sur pastebin.com dans ce qui semble une liste d'adresses volées datant de 2012. Manifestement l'attaquant ne cherche pas à se dissimuler en utilisant des adresses de courriel éphémères (jetables).

[9] http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013%20-%20French.pdf

SPIP permet d'installer des *plugin*. La documentation[10] précise : *Depuis SPIP 2.0, il est possible d'activer l'installation automatique de plugins. Pour cela, il faut créer un sous-dossier plugins/auto/, et donner au serveur les droits d'écriture dans ce sous-dossier*. Si on est administrateur SPIP, ce qui est le cas pour l'attaquant, on a la possibilité via un formulaire de spécifier l'URL d'un plugin à télécharger puis de l'installer. Il suffit à l'attaquant de mettre son code malfaisant dans un plugin. Dans le cas présent le plugin porte le nom de *vst_v02-2* et son répertoire contient plusieurs *webshell*[11] dont *pat.php* auquel Paul se connecte peu après l'installation.

Nous avons obtenu une version en clair du *webshell* qui était obscurci en le soumettant au site unPHP[12]. Sa lecture comme son exécution dans un bac à sable ne laissent aucun doute sur son usage malveillant (cf. figures 3 et 4).

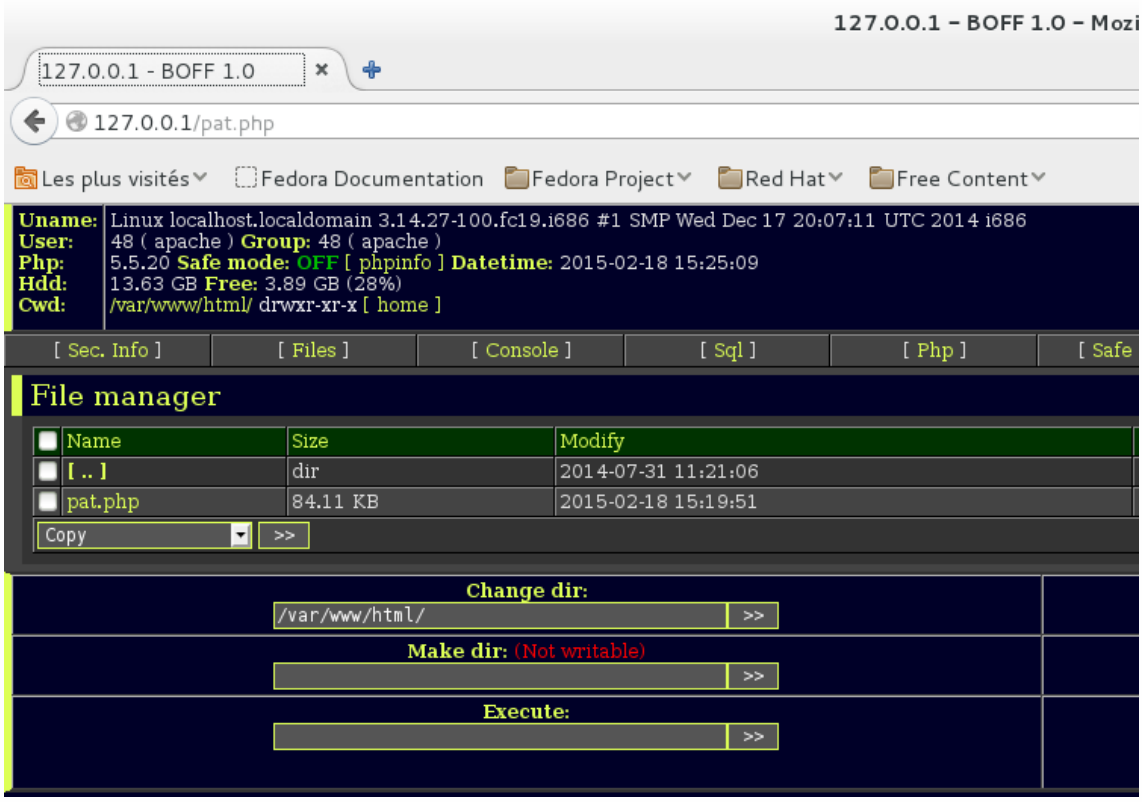

Figure 3 - Aperçu du webshell

[10] http://www.spip.net/fr_article3396.html
[11] http://korben.info/un-detecteur-de-web-shell-de-hackers-pour-votre-serveur.html
[12] http://www.unphp.net/

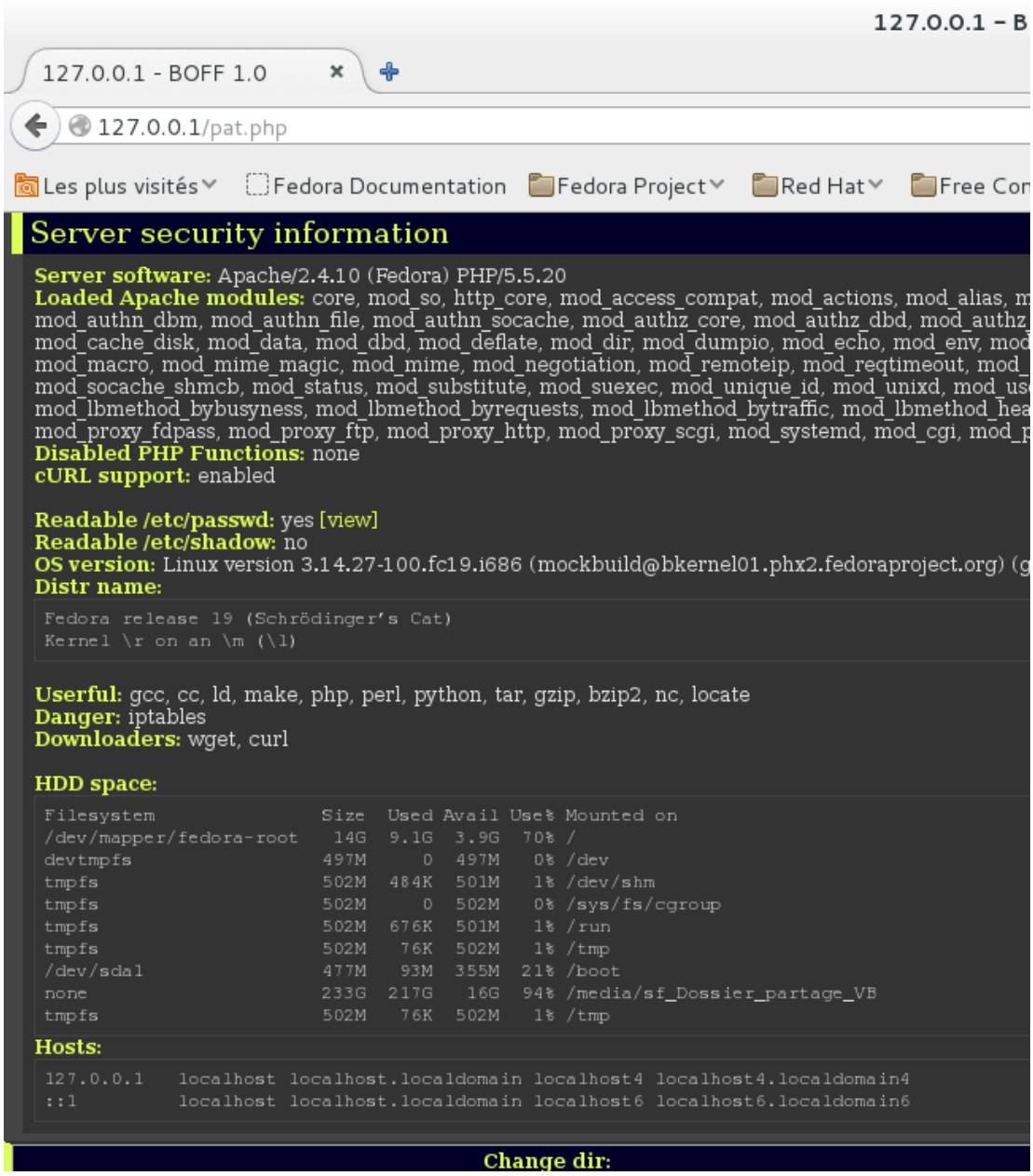

Figure 4 - Quelques cibles du webshell

Paul à partir de son *webshell* pouvait lire, écrire ou modifier des fichiers sous l'identité du serveur web. Or les droits qui lui étaient attribués permettaient uniquement de traverser le répertoire contenant les racines des documents (*DocumentRoot*) des différents sites hébergés sur la machine. Ils ne permettaient pas d'en lister le contenu.

Paul a utilisé un autre *webshell pro.php* pour créer un lien symbolique vers la racine du disque. On retrouve la commande utilisée dans le *webshell* mis au clair.

```
symlink("/","pro/root")
```

Peu après une trace de la requête HTTP

```
GET /plugins/auto/vst_v02-2/pro/root
```

avec un code de retour de 200 montre qu'il a pu lister le contenu du répertoire racine du disque. Des requêtes analogues lui ont permis de découvrir l'organisation du disque et en particulier quels étaient les répertoires des autres sites hébergés sur le même serveur. Nous constatons que l'attaquant a systématiquement lu les fichiers de configurations d'Apache pour les différents sites, qu'aussitôt après il se connectait au *webshell pat.php*, qu'au même moment était créé son fichier de revendication *mkd.html* et que peu après il y avait une requête pour accéder à ce fichier. Il est évident que ce fichier de revendication a été créé à l'aide du *webshell*.

Cela supposait que la configuration d'Apache permette de suivre les liens symboliques. Au cas où, le *webshell pro.php* créait, en même temps que le lien symbolique sous le même répertoire que celui-ci, un fichier *.htaccess*. Les commentaires ont été ajoutés par nous.

```
Options All # tout est permis comme suivre les liens symboliques
DirectoryIndex Sux.html # pas de Sux.html => liste le répertoire
AddType text/plain .php # les .php sont vus comme du texte
AddHandler server-parse .php # affiché et non pas exécuté
AddType text/plain .html
AddHandler txt .html
Require None # aucune autorisation n'est exigée
Satisfy Any  # tout est permis
```

En fonction des options de configuration d'Apache, il n'est pas certain que toutes ces directives soient prises en compte.

On trouve encore d'autres *webshell* de différentes origines, ce qui montre que note pirate s'était constitué une boîte à outils qu'il utilise en fonction de ses besoins.

Il y a aussi un autre *webshell k2.php* qui n'est pas offusqué et pourrait être si on en croit les commentaires être écrit par notre attaquant Paul (cf. figure 5).

Figure 5 - Autre webshell

On a aussi *p.php* qui est un *webshell* PHP cherchant à se masquer sous la forme d'une image GIF.

```
GIF89aP;
<?GIF89aP;
error_reporting(0); //If there is an error, we'll show it, k?
$password = ""; // You can put a md5 string here too, for
plaintext passwords: max 31 chars.
```

Ou encore le fichier *domain.cgi* qui est un *webshell* écrit en Perl. On peut imaginer les possibilités d'un tel code malfaisant qui, de plus, a l'avantage, pour l'attaquant, de court-circuiter toutes les restrictions que l'on a pu mettre sur PHP. Les quelques commentaires extraits du programme donnent une idée de ses fonctionnalités.

```
# user wants to run a command
# user wants to save a file
# user wants to upload a file
# user wants to back connect or bind port
# user wants to brute force
# user wants to download a file
# user wants to view log file
# user wants to view list user/domain
# user wants to logout
```

L'enregistrement dans le journal montre un accès à l'URL de ce code. Cependant si le serveur n'a pas retourné d'erreur (code 200), la longueur 49707 de la réponse correspond juste à la taille du fichier. Cela montre que le script n'a pas été exécuté mais seulement affiché. Échec donc pour l'attaquant.

```
10.3.58.96 - - [24/Jan/2015:14:29:25 +0100] "GET
/plugins/auto/thickbox2/domain.cgi HTTP/1.1" 200 49707 "-" "Mozilla/5.0"
```

On a trouvé sur la première machine le fichier *thickbox2.zip*

Les accès à ce fichier sont intéressants

```
10.3.58.96 - - [24/Jan/2015:13:54:04 +0100] "GET
/plugins/auto/thickbox2/thickbox2.zip HTTP/1.1" 200 31767 "-" "Mozilla/5.0"
10.9.176.10 - - [24/Jan/2015:13:54:31 +0100] "GET
/plugins/auto/thickbox2/thickbox2.zip HTTP/1.0" 200 31767
"http://www.autre_site.fr/web/" "SPIP-2.1.1 (http://www.spip.net)"
```

On constate que juste après son dépôt ce fichier est téléchargé depuis l'adresse de l'attaquant certainement pour vérification mais aussi depuis une autre adresse 10.9.176.10 avec un *User-Agent* qui indique qu'il s'agit non pas d'un navigateur mais du logiciel SPIP. Cela correspond très probablement à une tentative d'installation par Paul de ses logiciels malfaisants sur cette autre machine.

Nous avons trouvé une copie du fichier */etc/passwd* et dans les journaux des traces de consultation de divers fichiers de configuration. Ce qui prouve que l'attaquant avait peut-être d'autres visées qu'une simple défiguration. Tous les mots de passe, ceux des utilisateurs du système, ceux permettant de

contrôler l'accès aux CMS, ceux utilisés par les CMS ou applications web pour se connecter à la base de données[13] doivent être considérés comme définitivement compromis.

N'ayant pas effectué de recherches sur d'éventuels fichiers effacés nous ignorons ce que l'attaquant a installé puis supprimé.

5 Marqueurs

Les marqueurs ou IOC (*Indicator Of Compromission*) sont des éléments que l'on retrouve sur une machine victime d'une attaque. L'analyse de l'attaque va permettre de déterminer les marqueurs qui sont spécifiques de celle-ci. Ainsi manuellement (*grep*, *find*, etc.) ou à l'aide d'outils automatiques comme YARA[14] ou OpenIOC[15], il va être possible de vérifier si une machine a été victime de l'attaque.

L'analyse précédente nous a permis de déterminer un certain nombre de ces marqueurs. Il s'agit des adresses de courriel ou des identifiants utilisés par l'attaquant, celui-ci ayant eu le bon goût de ne pas en changer. Il y a aussi *0minirezo* dans les requêtes HTTP qui est certes parfois légitime mais est caractéristique de la vulnérabilité utilisée. La présence de liens symboliques pointant en dehors de l'arborescence du site et notamment sur la racine du système n'est sans doute pas spécifique de cette attaque mais est toujours éminemment suspecte. Lorsque l'on a des fichiers PHP qui contiennent nombre d'instructions comme *eval*, *base64_decode*, de longues chaînes codées en hexadécimal ou en base64, il est fort probable qu'il s'agisse d'un programme qui cherche à se masquer et donc vraisemblablement malfaisant. Les noms de fichiers et répertoires utilisés par l'attaquant ne sont pas forcément très discriminant, nous avons constaté que sur deux machines à l'exception de *pat.php*, il n'avait pas utilisé les mêmes.

6 Mesures de protection

Comment se protéger de telles attaques. Il faut respecter le principe d'une défense en profondeur où chaque mesure mise ne place ne sera pas forcément totalement efficace et pourra éventuellement être contournée mais où l'ensemble sera suffisamment résistant pour décourager la plupart des attaquants.

La mise à jour régulière des systèmes et différentes applications s'impose. Dans le cas présent, si le correctif disponible depuis plus d'un an avait été appliqué, l'attaquant n'aurait pu s'introduire.

Ne pas ouvrir le back office à l'ensemble de l'Internet. L'administration du CMS et la mise à jour de son contenu devraient être effectués depuis des adresses IP appartenant au réseau interne ou au moins contrôlées. Si la contribution externe s'avère réellement indispensable après étude du risque, l'utilisation d'un VPN permet de s'affranchir d'éventuels obstacles. Il est aussi possible de mettre un contrôle d'accès pour l'administration ou la modification du contenu du CMS au niveau du serveur Apache et pas uniquement au niveau du CMS. Cela peut se faire en restreignant les URL utilisés à ces fins à certaines adresses IP ou bien à des utilisateurs authentifiés (par certificat client par exemple) au niveau d'Apache. Les failles sont dans les CMS rarement dans le mécanisme d'authentification du serveur Apache. Avec SPIP on peut contrôler l'accès au répertoire */ecrire.*

Interdire les fonctions dangereuses de PHP comme *symlink* aurait pu empêcher l'attaquant de se créer son lien symbolique qui lui a permis de remonter à la racine du système. Cela n'a un sens que si on interdit aussi toutes les fonctions permettant d'exécuter des commandes ou un *shell* comme *system*, *exec*, *passtru*, *popen*, *shell_exec*. Plus largement, une attention particulière doit être portée au contenu de la

[13] Généralement en clair dans un fichier de configuration.
[14] http://plusvic.github.io/yara/
[15] http://www.openioc.org/

configuration PHP (php.ini) dans son ensemble (des outils peuvent aider[16]). L'OWASP publie une synthèse de bonnes pratiques pour améliorer la sécurité de l'environnement d'exécution de PHP[17].

Définir la directive *open_basedir* pour chaque site de telle sorte que PHP ne puisse accéder à des fichiers en dehors de l'arborescence du site.

Configurer le serveur Apache pour lui interdire de suivre les liens symboliques. L'option *FollowSymlinks* ne devrait pas être activée. Ce n'est pas absolu car des situations de compétition (*race conditions*) peuvent éventuellement permettre de passer outre mais cela complique la tâche de l'attaquant.

Positionner les utilisateurs, groupes et droits d'accès afin de limiter au maximum les répertoires et fichiers auquel le serveur web peut accéder. Dans le cas présent une simple interdiction de lister un répertoire tout en laissant la possibilité de le traverser a sérieusement compliqué le travail de l'attaquant.

Interdire qu'un fichier *.htaccess* puisse redéfinir certains paramètres critiques d'Apache comme FollowSymlinks ou le *handler* associé à une extension de fichier.

Interdire la redéfinition de certains paramètres de PHP dans *.htaccess* ou par le script lui-même.

Peut-être mettre en œuvre des solutions pour durcir le système comme SELinux. Ce dernier n'est certainement pas simple à mettre en œuvre mais il permet de bien contrôler l'accès aux ressources comme les fichiers.

Envisager des solutions de virtualisation pour isoler les sites entre eux. Cela peut aller d'une virtualisation complète à la VMware, Xen ou KVM à des solutions plus légères comme des conteneurs Virtuezzo, OpenVZ, LXC (Linux containers), Jail sous BSD. Docker pourrait être utilisé pour faciliter le déploiement.

Mettre en place des outils de consolidation, d'indexation et d'analyse de journaux comme Splunk[18] ou ELK[19], permettrait de répondre plus efficacement aux incidents.

7 Plainte

L'action des attaquants est clairement répréhensible et il est normal de porter plainte. En effet l'accès frauduleux à un STAD[20], l'introduction et la modification frauduleuses de données sont des faits visés par les articles 323-1[21] et 323-3[22] du code pénal. Cela n'exclut pas que d'autres qualifications puissent être retenues. Sachant que les procédures sont souvent longues et que nombre de plaintes sont classées sans suite, il convient de déterminer l'opportunité d'une action en justice. Dans tous les cas il faut respecter les procédures de son organisme et s'adresser à son RSSI et à son service juridique[23]. Il est très important d'acquérir et de préserver, dès la découverte de l'incident, le maximum d'éléments pouvant servir de preuves comme les journaux, les images des disques.

Conclusions

[16] https://github.com/psecio/iniscan
[17] https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet#php.ini
[18] http://fr.splunk.com/
[19] http://www.elasticsearch.org/overview/
[20] Système de traitement automatisé de données – expression issue de la loi Godfrain datant de 1988 sur la fraude informatique.
[21] Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 60 000 € d'amende.
Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 100 000 € d'amende.
[22] Le fait d'introduire frauduleusement des données dans un système de traitement automatisé, d'extraire, de détenir, de reproduire, de transmettre, de supprimer ou de modifier frauduleusement les données qu'il contient est puni de cinq ans d'emprisonnement et de 75 000 euros d'amende.
[23] Il faut noter, pour les structures dépendant du MENESR, qu'il ne faut pas hésiter à solliciter le RSSI adjoint de celui-ci qui a une bonne expérience des dépôts de plaintes et de bonnes relations avec les services de police.

Une attaque peut en cacher une autre. Aucune attaque réussie sur un site web ne doit jamais être prise à la légère. En particulier une défiguration n'est pas anodine. L'attaquant ne s'est généralement pas contenté d'afficher sa page, il a aussi très souvent installé une porte dérobée pour faire d'autres attaques et revenir à sa guise sur la machine. Et si ce n'est pas lui, ce sera un autre.

Dans la plupart des sites web compromis que nous avons eu l'occasion d'analyser un peu en profondeur nous avons constaté les traces de plusieurs attaques, la première datant souvent de très longtemps (plusieurs mois voire années) et le dernier attaquant qui se fait détecter n'est pas forcément le meilleur et celui qui a fait le plus dégâts, c'est le moins discret.

Face à une défiguration qui est par essence très visible, il faut se poser la question de savoir s'il ne s'agit pas d'un rideau de fumée destinée à cacher une attaque beaucoup grave.

En présence d'une attaque sur un site web, il n'y a pas d'alternative, il faut le réinstaller à partir de zéro, avec un système à jour. Il est extrêmement difficile de retrouver toutes les portes dérobées qu'a pu laisser l'attaquant et ceux qui l'ont précédé. Il faut aussi changer tous les mots de passe. L'information de la chaîne fonctionnelle SSI adéquate est nécessaire pour caractériser le côté isolé ou massif de l'attaque.