



# Internet Security

Prof. Bella Giampaolo

AA 2023/24



REALIZZATO DA  
Giulio Pedicone

REALIZZATO DA  
Vincenzo Villanova



# Indice

<b>Introduzione.....</b>	<b>8</b>
Motivazioni: app sicure – pagamenti online .....	8
Motivazioni: browsing sicuro.....	8
Motivazioni: pagamenti innovativi e sicuri.....	8
Ambiti di sicurezza informatica.....	9
<b>Parte 1: Esempi reali e falsi miti .....</b>	<b>10</b>
Trojan Horse .....	10
Strumenti generali di sicurezza.....	13
Limiti della crittografia.....	14
Limiti dell'uso di password .....	14
Limiti di un Firewall.....	15
Definizione di Sicurezza per punti.....	16
Rischi base per la sicurezza .....	16
Rischi digitali per la sicurezza .....	17
Il gioco della sicurezza (Attaccare e difendere) .....	18
Port Scanning .....	19
<b>Parte 2: Proprietà, Attacchi e Attaccanti .....</b>	<b>20</b>
Attacchi.....	20
Attacco reale: pirateria digitale.....	21
Proprietà di Sicurezza .....	21
Definizione di segretezza (Confidenzialità)	22
Definizione di Autenticazione .....	23
Integrità (coerenza) .....	23
Privatezza (Privacy) .....	24
Anonimato.....	24
Proprietà di sicurezza di livello 2 .....	25
Non ripudio .....	25
Autenticazione anonimato e non ripudio.....	26
WATA: Written Authenticated Though Anonymous.....	28
WATA 2 .....	29
Disponibilità (non Denied of Service).....	31
Proprietà di sicurezza di livello 3 .....	31
Controllo d'accesso.....	31

Esempio di una politica di sicurezza .....	32
Elementi di una politica di sicurezza.....	32
Modalità e relazioni fra loro.....	33
Inconsistenze di una politica.....	33
Politiche in pratica: MAC.....	34
Politiche in pratica: RBAC.....	34
Meccanismi implementativi: ACM.....	34
Meccanismi implementativi: ACL e CAL.....	35
Tassonomia di attaccanti.....	35
Modelli di attaccante .....	36
Modello di attaccante: Dolev-Yao.....	36
Modello di attaccante: General Attacker .....	36
<b>Parte 3: Autenticazione.....</b>	<b>37</b>
Ambientazioni di autenticazione .....	37
Autenticazione utente-computer .....	37
Autenticazione basata su conoscenza.....	37
Guessing: Attacchi .....	38
Guessing: Contromisure .....	38
Norme per una buona password.....	39
Mantenere una password.....	39
CTSS: Compatible Time Sharing System.....	39
Autenticazione basata sul possesso .....	40
Smart Token .....	40
Funzionamento di uno Smart Token.....	41
Discussione.....	41
Autenticazione basata su biometria .....	41
Funzionamento .....	42
<b>Parte 4: Cenni di crittografia .....</b>	<b>43</b>
Crittografia: Concetti di base.....	43
Crittosistema.....	44
In rete.....	44
Segretezza di una chiave.....	45
Crittografia simmetrica .....	45
Crittografia simmetrica in rete .....	45
Limiti della crittografia simmetrica .....	47
Crittografia asimmetrica.....	47

Crittografia asimmetrica in rete.....	49
Limite della crittografia asimmetrica .....	50
Riepilogo crittografia simmetrica ed asimmetrica .....	50
Crittosistema sicuro.....	51
Hash crittografico .....	51
Esempio: SHA 1.....	52
CTSS + Hashing .....	52
Salting.....	53
UNIX: Aggiunta di una password.....	53
UNIX: Verifica di una password.....	54
Freshness .....	55
<b>Parte 5: Protocolli di sicurezza basilari.....</b>	<b>57</b>
Protocolli basilari per la freshness .....	57
Dalla crittografia alla sicurezza.....	57
Sintassi dei messaggi crittografici.....	58
Sintassi di un protocollo di sicurezza.....	58
Protocolli basilari per la segretezza .....	59
Protocolli basilari per l'autenticazione .....	59
Combinare segretezza ed autenticazione .....	60
Come ottenere l'integrità.....	61
Firma digitale.....	62
Creare e verificare una firma.....	62
Protocolli basilari per l'integrità.....	63
Combinare tutte le proprietà di livello 1.....	63
Protocollo Diffie-Hellmann.....	64
Protocollo Diffie-Hellmann – Attacco .....	65
Diffie-Hellmann – Attacco al microscopio .....	66
Come irrobustire Diffie-Hellmann.....	66
RSA Key Exchange.....	67
P.I.I.: Personal Identifiable Information.....	67
Certificazione: Crittografia asimmetrica .....	68
Certificato X.509.....	69
Public Key Infrastructure .....	70
Chain Of Trust.....	71
Segretezza VS Fiducia.....	72
Problema: Certificati Self-Issued .....	73

Lista di Revoca dei Certificati (CRL).....	75
<b>Parte 6: Protocolli di sicurezza storici.....</b>	<b>76</b>
Protocollo Needham-Schröder.....	76
Scenario di attacco .....	79
Potenziali soluzioni.....	80
Conseguenze dell'attacco.....	81
Vendetta nel modello General Attacker.....	81
Principi di disegno: Explicitness .....	82
Riepilogo del protocollo Needham-Schröder.....	82
Protocollo di Woo-Lam .....	83
Attacco su Woo-Lam.....	85
Soluzione all'attacco su Woo-Lam.....	86
<b>Parte 7: IPSEC.....</b>	<b>87</b>
Protocolli di sicurezza .....	87
Alternative alla crittografia .....	87
Sicurezza a quale livello? .....	88
Sicurezza a livello di rete.....	88
Sicurezza a livello di trasporto .....	89
Sicurezza a livello applicazione .....	89
IPSec .....	90
Security Association (SA).....	90
Security Association Database (SAD) .....	91
Authentication Header (AH) .....	91
AH - Formato.....	92
IPSec funzionamento di base .....	93
Prevenzione dei replay attack.....	93
Protezione mediante AH: Trasporto vs Tunnel .....	94
AH in IPv4 ed in IPv6.....	95
Usi di AH: Trasporto vs Tunnel.....	96
AH in Tunnel .....	96
Encapsulaing Security Payload (ESP).....	97
ESP – Formato .....	98
ESP in IPv4.....	99
Come rendere la rete domestica più sicura? .....	100
Tipo 1 di combinazioni SA: sicurezza punto-punto.....	100
Tipo 2 di combinazioni SA: sicurezza fra intermediari.....	101

Tipo 3 di combinazioni SA: tipo 1 & tipo 2 .....	101
Tipo 4 di combinazioni SA: tipo 1 & sicurezza punto intermedio .....	102
Internet Key Exchange (IKE) .....	102
<b>Parte 8: Intrusion Detection .....</b>	<b>103</b>
Intrusione versus Malware.....	103
Intrusione versus DoS .....	103
Negazione del servizio DoS .....	104
Dos vs DDoS .....	104
Come reagire?.....	105
Anti-DoS: Cookie Transformation.....	105
Intrusione .....	106
Tecniche di intrusione.....	106
Intrusion Detection VS Intrusion Management.....	107
Intrusion Detection System .....	107
Definire i comportamenti.....	108
Record di auditing.....	108
Record di Denning.....	109
Tecniche di rilevamento .....	110
Rilevamento statistico – Modelli .....	111
Rilevamento a regole.....	112
Tecniche di protezione .....	112
<b>Parte 9: Software nocivo (Malware).....</b>	<b>113</b>
Bug VS Software Nocivo .....	113
Software Nocivo - Caratteristiche .....	114
Software Nocivo - Tassonomia.....	115
Trapdoor (Backdoor).....	116
Bomba Logica.....	116
Trojan .....	117
Zombie.....	117
Worm .....	118
Worm Morris .....	118
Attacco Known-Ciphertext .....	119
Buffer Overflow mediante finger .....	120
Backdoor in Sendmail.....	121
Virus.....	122
Macroviruss.....	123

Software Nocivo – Distinzione Delicata.....	123
Struttura di un semplice virus.....	124
Struttura di un virus che comprime .....	125
Esempio di virus: Brain .....	126
Rimozione di Software Nocivo .....	127
Struttura ideale di un antivirus.....	128
<b>Parte 10: WWW Security Protocols .....</b>	<b>129</b>
Protocollo di sicurezza per il web .....	129
SSL.....	130
SSL vs TLS.....	131
I protocolli in SSL.....	132
SSL Record Protocol.....	132
Record Header .....	133
MAC di SSL.....	134
SSL Change Cipher Spec Protocol.....	134
SSL Alert Protocol.....	135
SSL Handshake Protocol.....	136
Messaggi SSL.....	136
Fase 1: Stabilimento della connessione.....	137
Client Hello.....	138
Server Hello .....	138
Cipher Suite .....	139
Fase 2: Autenticazione del server e scambio delle chiavi .....	140
Fase 3: Autenticazione del client e scambio delle chiavi.....	141
Fase 4: Fine.....	142
Master Secret.....	143
Key Blocks .....	143
Ridurre la latenza di rete.....	144
TLS .....	145
HMAC .....	146
P_HASH.....	147
Pseudo Random Function.....	148
Evoluzione di TLS.....	149
<b>Parte 11 (Lab): Difesa della Rete .....</b>	<b>150</b>
Red, Purple, Blue Team.....	150
Intrusione .....	150

Linee di difesa tipiche di una rete.....	151
Firewall.....	151
Vulnerabilità.....	152
Debolezza (Weakness).....	152
Exploit.....	152
Firewall: Funzionalità principali e secondarie .....	152
Tipologie di Firewall.....	153
Systems Protected Firewall .....	153
Network Placement Firewall.....	154
Form Factors Firewall .....	155
Packet Filtering Firewall.....	156
Stateful Inspection Firewall.....	157
Firewall Packet Inspection.....	157
Stateful Inspection Firewall: Limiti.....	158
Circuit Level Gateway .....	159
Proxy Firewall / Application Firewall.....	160
Web Application Firewall.....	161
Next generation Firewall.....	162
Netfilter .....	163
Iptables.....	164
Nftables.....	166
Snort .....	171
Protezione data dai sistemi di difesa .....	175
Incident Response .....	176
Tipologie di Incidenti Comuni .....	176
Possibili impatti di un incidente.....	177
Log di sistema.....	178
SIEM.....	178
Esempi di Alert.....	179
IBM QRadar.....	179
Wazuh.....	180
SOC.....	181
Greynoise .....	181

# Introduzione

## Motivazioni: app sicure – pagamenti online

Effettuare **pagamenti online** inserendo il numero della propria carta di credito online è un enorme **atto di fede**. Questo comporta il **rischio** di **esporre** le **informazioni finanziarie** personali a potenziali minacce.

### **Security VS Usability**

La facilità d'uso ci spinge ad effettuare **atti di fede**. Questo perché, se da un lato la **sicurezza** richiede **procedure rigorose** per **proteggere** i **dati** sensibili, dall'altro eccessiva complessità **potrebbe scoraggiare** gli **utenti**. Trovare un equilibrio tra sicurezza e usabilità è essenziale per garantire pagamenti sicuri e senza intoppi online.

### **Trust Anchor**

Il **lucchetto verde** visualizzato su un **sito web** o un'applicazione durante il processo di pagamento è un esempio di trust anchor. La presenza di un trust anchor **aiuta** gli **utenti** a **sentirsi più sicuri** nel fornire le proprie informazioni finanziarie, in quanto **indica** che il **sito è autenticato** e che le **comunicazioni** sono **crittografate**.

## Motivazioni: browsing sicuro

Gli **aggiornamenti software** sono **fondamentali** misure di sicurezza perché **correggono** le **vulnerabilità** di sicurezza esistenti nel software, fornendo patch e miglioramenti che mantengono i sistemi al passo con le minacce emergenti. **Senza** di **essi**, i **dispositivi** rimarrebbero **esposti** a **rischi** significativi di **violazioni** della **sicurezza** e di **compromissione** dei **dati**.

## Motivazioni: pagamenti innovativi e sicuri

**Distributed Public Ledger** (Bollettino Pubblico Distribuito)

**Annota** tutte le **transazioni** di **compravendita**, o di qualunque tipo in altri casi.

**Innumerevoli problemi** di **sicurezza** risolvibili tramite crittografia.

# Ambiti di sicurezza informatica

## Programmazione:

- **Null Pointer Dereference:** Si verifica quando un programma tenta di accedere o dereferenziare un puntatore che punta a NULL, causando spesso un crash del programma o comportamenti imprevisti.
- **Buffer Overflow:** Accade quando un programma tenta di scrivere più dati in un buffer di quelli che può contenere, potenzialmente sovrascrivendo altre parti della memoria e portando a vulnerabilità di sicurezza.

## Sistemi operativi:

- **Protezione memoria:** Meccanismi implementati dai sistemi operativi per proteggere le regioni di memoria da accessi non autorizzati, impedendo a un processo di accedere direttamente alla memoria di un altro processo.
- **Controllo d'accesso:** Il controllo d'accesso regola l'accesso alle risorse del sistema operativo da parte degli utenti e dei processi, garantendo che solo gli utenti autorizzati possano eseguire determinate azioni o accedere a determinati file.

## Servizi innovativi:

- **Moneta elettronica:** Sistemi di pagamento elettronici che consentono a individui e aziende di effettuare transazioni finanziarie online in modo sicuro e conveniente.
- **Autenticazione audio:** Tecniche di autenticazione basate sull'analisi delle caratteristiche vocali per verificare l'identità degli utenti.

## Database:

- **Linking Attack:** Un attacco che sfrutta le relazioni tra diverse entità in un database per ottenere informazioni sensibili o compromettere la sicurezza del sistema.
- **K-Anonymity:** Una tecnica di anonimizzazione dei dati che garantisce che ogni individuo in un dataset sia anonimo rispetto ad almeno K altri individui nel dataset.

## Reti ...

## Introduzione alle Regole (Policy)

Le policy rappresentano le scelte progettuali per regolare il comportamento degli utenti e dei sistemi stessi. Le policy possono riguardare diversi aspetti, come l'accesso ai dati, l'utilizzo delle risorse informatiche, la sicurezza delle informazioni e così via.

Un evento non viene considerato un attacco informatico se non viola le policy stabilite

# Parte 1: Esempi reali e falsi miti

## Trojan Horse

Il **Trojan Horse**, o Cavallo di Troia, è un tipo di **software** che, **ingannevolmente**, si presenta come innocuo o addirittura benefico per l'utente, ma in realtà **nasconde intenzioni malevoli**. Questo tipo di malware può essere progettato per svolgere una vasta gamma di azioni dannose, come il **furto** di **dati** sensibili, il **monitoraggio** delle attività **dell'utente**, il **danneggiamento** del **sistema** o la **creazione** di **backdoor** per futuri attacchi.

### Esempio non più funzionante:

```
cp /bin/sh /tmp/.xxsh  
chmod u+s,o+x /tmp/.xxsh  
rm ./ls  
ls $*
```

### Funzionamento del trojan (Inserimento della vulnerabilità):

1. **Copia della shell**: Il codice **copia** la **shell** di sistema (/bin/sh) **in una cartella nascosta** chiamata /tmp/.xxsh. Questa **mossa** è **ingannevole poiché** il punto prima del nome della cartella la **rende invisibile** alla maggior parte dei comandi di visualizzazione delle directory, come ls senza l'opzione -a.
2. **Modifica dei permessi**: Successivamente, il codice **modifica i permessi** della **shell copiata** nella cartella nascosta. Imposta il bit SUID (Set User ID) al proprietario della shell e concede i permessi di esecuzione a tutti gli utenti. Questa azione è pericolosa in quanto **concede** agli **utenti** l'accesso ai **privilegi del proprietario** della shell, potenzialmente consentendo loro di eseguire comandi con privilegi elevati.
3. **Autocancellazione del trojan**: Il codice **si autoelimina**, rimuovendo se stesso dal sistema. Questo rende il Trojan meno facile da rilevare dopo l'esecuzione.
4. **Stealthiness**: Infine, per mantenere la sua operazione nascosta, **il Trojan esegue il comando ls** per far credere all'utente che tutto funzioni normalmente. Questo può ingannare gli utenti, facendo loro credere che il sistema sia intatto e privo di eventuali intrusioni.

## Approfondimento su bit SUID

Il bit SUID (Set User ID) è un attributo speciale nei permessi di un file eseguibile in sistemi UNIX e UNIX-like. Quando il bit SUID è impostato su un file eseguibile, il programma viene eseguito con i privilegi dell'utente che è proprietario del file anziché con i privilegi dell'utente che lo ha avviato. Questo significa che il programma ha temporaneamente accesso a risorse di sistema che normalmente sarebbero inaccessibili all'utente che lo sta eseguendo.

```
$ ls -l /usr/bin/passwd  
-rwsr-xr-x 1 root root [...]
```

Nell'esempio fornito, il comando ls -l /usr/bin/passwd mostra un esempio di file eseguibile, passwd, con il bit SUID impostato. Questo file è di proprietà di root, il superutente del sistema, e viene eseguito con i suoi privilegi quando viene avviato da qualsiasi utente sul sistema.

Senza l'uso del bit SUID, una possibile soluzione per consentire agli utenti di cambiare le proprie password sarebbe la duplicazione del programma passwd per ogni utente, o la delega di privilegi speciali a singoli utenti. Tuttavia, queste soluzioni sarebbero meno efficienti in termini di sicurezza e manutenzione del sistema rispetto all'utilizzo del bit SUID su un singolo eseguibile come passwd.

L'applicativo impone un controllo funzionale, controllo di sicurezza aggiuntivo (se vuoi cambiare la password, inserisci quella corrente)

## Abuso di sistema VS Violazione di sistema

Il Trojan visto in precedenza non rompe il sistema e non viene violata alcuna policy, abbiamo solo abusato il sistema (abbiamo sfruttato una vulnerabilità del sistema).

In questo scenario si presuppone che il trojan venga eseguito subito.

L'attacco funzionava perchè fino a qualche anno fa il trojan era già eseguibile.

## Cartella Download

La cartella "Download" nei sistemi operativi ha uno scopo pratico di fornire agli utenti un luogo centralizzato per trovare i file scaricati dalla rete.

Tuttavia nasce per motivi di sicurezza, la sicurezza dei file scaricati dipende dai permessi impostati sulla cartella stessa e dai permessi dei file specifici.

### Perchè viene eseguito il trojan e non il comando ls di sistema?

Se il trojan sta nella cartella Download come fa la vittima ad eseguire questo trojan? Perchè in realtà fino a qualche anno fa la cartella download non esisteva e tutto veniva salvato nella home. Gli applicativi erano file standalone, l'applicativo non aveva un installer e si lavorava esclusivamente sulla home directory.

Si presuppone quindi che il file trojan si trovi quindi nella cartella home (predefinita) dell'utente. Quindi quando si eseguiva ls andava in esecuzione il trojan e non l'ls di sistema.

Possibile soluzione ma non quella corretta:

Link simbolico: ln -s

Non rappresenta una soluzione corretta per questo scenario poiché non influisce sull'ordine di esecuzione dei comandi o sulla rimozione del file ls.

### Variabile PATH

La variabile PATH è una componente delle variabili di ambiente del sistema che il SO utilizza per determinare i percorsi in cui cercare i programmi eseguibili. Questa informazione è essenziale per localizzare i comandi che gli utenti digitano nella shell.

```
$PATH  
usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
# Serie di percorsi dove il sistema ricerca i comandi da eseguire ognuno separata  
da:
```

Quando si esegue un comando, il sistema consulta una serie di percorsi specificati nella variabile PATH fino a trovare il comando desiderato.

In passato, nella variabile PATH di Linux, era presente il punto (.), che indicava la possibilità di eseguire comandi direttamente dalla directory home dell'utente, ed era solitamente posizionato all'inizio della lista di percorsi da seguire.

Tuttavia, questa configurazione era vulnerabile ad attacchi, quindi una mitigazione è stata rimuovere il punto (.) da PATH.

**Nota bene: Questa soluzione non risolveva completamente il problema poiché impediva agli utenti tradizionali di eseguire i comandi dalla propria home.**

Una soluzione più efficace è stata rimuovere la priorità dalla cartella in cui vengono scaricati i file. Questo aiuta a proteggere il sistema da possibili attacchi, in quanto i file scaricati non hanno più la priorità di esecuzione rispetto ai comandi di sistema.

Attualmente, per motivi di sicurezza, nella variabile PATH non troviamo neanche la cartella ./Downloads. Questa è una precauzione aggiuntiva adottata per proteggere il sistema da potenziali minacce.

### Modello di attacco

Sono le ipotesi di base per il funzionamento dell'attacco.

- **Sfruttamento di una vulnerabilità:** Consiste nell'identificare e sfruttare una vulnerabilità nel sistema bersaglio.
- **Catena di attacco (o kill-chain di attacco):** Dopo aver sfruttato la vulnerabilità, l'attaccante procede attraverso una serie di fasi o azioni coordinate per raggiungere il proprio obiettivo. La catena di attacco è progettata per massimizzare l'efficacia dell'attacco e minimizzare le possibilità di rilevamento da parte del sistema di difesa.

### Dove possiamo trovare trojan?

- Pubblicità su YouTube
- Software di sicurezza (vulnerabilità più o meno deliberate, non ci si può fidare in principio degli antivirus, nemmeno di quello preinstallato)
- DDoS: Distributed Denied of Service, indisponibilità del servizio

## Strumenti generali di sicurezza

- **Cifratura:** Utilizzata per **proteggere i dati** attraverso la cifratura **simmetrica** (stessa chiave per cifrare e decifrare) e **asimmetrica** (due chiavi, pubblica e privata).
- **Policy di Sicurezza:** Definiscono **regole** per distinguere comportamenti leciti da tentativi di attacco, stabilendo il **comportamento accettabile** del **sistema**.
- **Autenticazione:** Processo di **verifica dell'identità** tramite password, PIN, smart card, biometria, etc., per garantire l'accesso autorizzato alle risorse.
- **Programmi di Protezione:** Software come antivirus, IDS (Intrusion Detection Systems), e firewall, che **proteggono i sistemi** da malware, monitorano il traffico di rete e filtrano accessi non autorizzati.
- **Protocolli di Sicurezza:** SSH e SSL assicurano la protezione dei dati durante il transito su reti non sicure, utilizzando crittografia e autenticazione.
- **Fattore Umano:** **Sensibilizzazione** degli **utenti** attraverso **informazione** e **formazione** riguardo alle pratiche di sicurezza, riducendo il rischio di attacchi basati sull'ingegneria sociale.

## Limiti della crittografia

La **crittografia** è una **disciplina** precisa, una **scienza esatta** come branca della matematica. Si **basa** su presupposti matematici chiari e incontestabili, come **l'impossibilità** di **violare RSA** a 2048 bit in tempo polinomiale e la **dimostrazione** della **sicurezza** dello **XOR** come metodo di codifica.

Tuttavia, la sicurezza associata alla crittografia è tutt'altro che esatta. Il suo utilizzo per ottenere sicurezza in pratica non offre le stesse garanzie. Ci sono **limiti computazionali** delle smartcard, anche se questo sta diventando sempre meno un problema. **L'utilizzo** è anche **limitato da leggi nazionali** e ci sono **limiti umani** di memoria e facilità d'errore.

Un esempio tangibile è rappresentato dalla '**sindrome del post-it**', in cui gli **utenti attaccano** pezzetti di carta con le **password** scritte **sopra** ai loro **monitor**. L'autenticazione a due fattori è resa obbligatoria solo per i sistemi bancari in base alla legislazione europea, ma nel contesto della posta elettronica, l'autenticazione a due fattori non è di default, nonostante l'utilizzo di un sistema di single sign-on.

In realtà, la sicurezza è tutt'altro che una scienza esatta. Nonostante i fondamenti matematici solidi, la sicurezza in pratica è influenzata da una serie di fattori imprevedibili e variabili

## Limiti dell'uso di password

1. **Scelta di una buona password:** Creare password complesse e difficili da indovinare.
2. **Mnemonicità:** Bilanciare la complessità con la facilità di memorizzazione.
3. **Attacco dizionario:** Proteggere contro attacchi che utilizzano elenchi di parole comuni.
4. **Attacchi statistici:** Difendersi da attacchi che sfruttano combinazioni di caratteri statisticamente probabili.
5. **Periodicità:** Gestire la necessità di cambiare periodicamente le password.
6. **Riutilizzo:** Evitare l'uso della stessa password su più servizi.
7. **Numerosità:** Gestire l'alto numero di password necessarie per vari account.

Nella realtà, le password più comuni sono sorprendentemente semplici, come "love" e le sue varianti, evidenziando una resistenza a creare password sicure e complesse.

### Sicurezza e generazione di password:

L'uso di generatori di password solleva una questione critica: la sicurezza potrebbe essere compromessa se chi genera la password ne conosce il contenuto, richiedendo un notevole atto di fiducia.

### Linee guida del NIST

- 2004: Il NIST ha stabilito criteri per la creazione di password robuste, politiche che sono state ampiamente accettate.
- 2016-2017: Il NIST ha introdotto nuove linee guida che non sono ancora state pienamente adottate. Queste nuove norme mirano a risolvere i conflitti tra requisiti di sicurezza e usabilità, proponendo password semplici e numeriche facili da ricordare, limitando però il numero di tentativi di accesso per prevenire attacchi di enumerazione (brute force).

### Mitigazione degli attacchi

Un metodo efficace per contrastare gli attacchi brute force è l'implementazione di un controllo a soglia, che limita il numero di tentativi di inserimento della password prima che vengano attivate misure di sicurezza.

## Limiti di un Firewall

Un firewall, sia hardware che software, controlla e gestisce il traffico in ingresso verso una rete o un sistema. Tuttavia, presenta alcuni limiti che richiedono una riflessione sulla politica da adottare:

- **Politiche di filtraggio:** Le politiche di filtraggio devono essere attentamente progettate per consentire il traffico solo da sorgenti attendibili e verso destinazioni autorizzate.
- **Filtraggio del traffico in uscita:** È cruciale considerare anche il filtraggio del traffico in uscita. Ad esempio, il rischio legato all'invio di dati sensibili da parte di malware tramite una connessione di tipo reverse shell deve essere affrontato. Filtrare il traffico in uscita può aiutare a mitigare questo rischio.
- **Bind shell vs reverse shell:** Una bind shell ascolta su una porta specifica e attende che un utente si connetta ad essa, mentre una reverse shell si connette a una porta specifica su un sistema remoto. Conoscere la differenza tra questi due tipi di connessioni può migliorare l'efficacia delle politiche di filtraggio e la sicurezza complessiva del sistema.

## Definizione di Sicurezza per punti

La sicurezza informatica può essere definita in modo generale come il processo volto a prevenire gli attacchi e garantire il corretto funzionamento del sistema.

- **Non un prodotto ma un processo:** È importante comprendere che la sicurezza informatica non è un prodotto fisico come una rete di computer o un sistema operativo, ma piuttosto un processo continuo che coinvolge la formazione delle persone, l'implementazione di aggiornamenti e la conduzione di sessioni di valutazione della vulnerabilità e dei test di penetrazione (VAPT).
- **Anello più debole di una catena:** È essenziale riconoscere che la sicurezza informatica rappresenta il punto più vulnerabile di un sistema. La sicurezza di un sistema è tanto solida quanto il suo anello più debole.
- **Espresso da che cosa:** La sicurezza informatica si basa su ciò che si vuole proteggere e su quali minacce sono più rilevanti. È fondamentale contestualizzare il livello di sicurezza in base al tipo di attaccante o al modello di rischio.
- **Sempre soggetta all'analisi costi/benefici dell'attaccante:** La valutazione della sicurezza informatica è sempre soggetta a un'analisi costi/benefici. È importante bilanciare il costo degli investimenti in sicurezza con i potenziali danni derivanti da una violazione della sicurezza.
- **Si realizza in pratica mediante livelli di sicurezza:** È cruciale perseguire sempre un livello elevato di sicurezza, poiché un sistema potrebbe essere protetto da attacchi condotti da hacker meno esperti ma risultare vulnerabile a quelli orchestrati da individui più competenti e sofisticati.

## Rischi base per la sicurezza

1. **Complessità del singolo sistema da mettere in Sicurezza:** Un problema fondamentale riguarda la necessità dei browser di comunicare in modo chiaro e accurato al loro utente il livello effettivo di sicurezza dei siti web visitati.
2. **Combinazione di Punti:** È essenziale combinare i sistemi con attenzione per garantire una protezione efficace.
3. **Predisposizione ai Bug:** È cruciale comprendere le proprietà inattese di un software, definizione fondamentale per gestire i rischi.
4. **Proprietà Emergenti:** Le nuove e moderne caratteristiche introdotte nei software possono portare a nuovi problemi di sicurezza. L'evoluzione del software genera nuove funzionalità e, di conseguenza, nuove esigenze di

sicurezza connesse a tali funzionalità (conforme a ISO 27005: Sorgenti di Rischio).

5. **Interazione con l'uomo:** Il fattore umano rimane sempre cruciale in termini di sicurezza. L'uomo rappresenta l'anello più debole della catena.

## Rischi digitali per la sicurezza

- **Automatizzazione dell'Offensiva:** La possibilità di automatizzare gli attacchi rappresenta una minaccia significativa. Gli attaccanti possono sfruttare strumenti e script per eseguire attacchi in modo efficiente e senza intervento umano.
- **Assenza di Distanza e Attacchi da Remoto:** La mancanza di distanza geografica non costituisce più una barriera per gli attacchi. È possibile eseguire attacchi da remoto, aumentando il punteggio di vulnerabilità secondo il CVSS (Common Vulnerability Scoring System), che valuta la gravità di una vulnerabilità su una scala da 1 a 10.
- **Mercato delle Vulnerabilità:** Esiste un mercato dinamico per le vulnerabilità informatiche. Gli individui etici che scoprono vulnerabilità spesso le segnalano a database internazionali dedicati, ma vi è anche un mercato in cui le vulnerabilità possono essere acquistate e vendute illegalmente.
- **Difficoltà nella Reazione:** La risposta agli attacchi informatici può essere estremamente complessa e difficile. Gli attaccanti spesso operano con grande velocità e astuzia, mentre le organizzazioni devono affrontare sfide tecniche, legali e di comunicazione nel tentativo di mitigare gli effetti degli attacchi e ripristinare la sicurezza del sistema.

## Il gioco della sicurezza (Attaccare e difendere)

Nel processo di sicurezza informatica, si segue un ciclo/gioco continuo di attacco e difesa:

1. **Ricerca delle Vulnerabilità**: Gli attaccanti cercano e identificano le debolezze nei sistemi informatici.
2. **Identificazione e Aggiornamento del Sistema**: Le vulnerabilità individuate vengono valutate e corrette mediante aggiornamenti e modifiche al sistema.
3. **Aggiornamento e Difesa**: Il sistema viene rafforzato con patch di sicurezza e contromisure per proteggerlo dagli attacchi.
4. **Monitoraggio e Risposta agli Incidenti**: Il sistema viene costantemente monitorato per individuare attività sospette e reagire prontamente in caso di violazioni della sicurezza.
5. **Ripetizione del Ciclo**: Il ciclo di attacco e difesa è continuo e richiede un impegno costante per mantenere la sicurezza del sistema.

Questo processo richiede una continua vigilanza e adattamento per contrastare le minacce informatiche in evoluzione.

### Metodologia d'attacco

- 1 Studiare il sistema target (*port scanning*)
- 2 Cercarne (in rete?) potenziali punti deboli (*deamon flaw*)
- 3 Disegnare (o scaricare!) eseguibili per verificare i punti deboli (*exploits*)
- 4 Goto 1

### Metodologia di difesa

- 1 Installare strumenti di difesa (*firewall, IDS*)
- 2 Aggiornare il sistema (*updates, security patch*)
- 3 Monitorare il sistema
- 4 Goto 1

## Port Scanning

Il **port scanning** è un **processo** mediante il quale **vengono esaminate** sequenzialmente le **porte** di un **sistema** al fine di **rilevare eventuali aperture** e **ottenere informazioni** sui servizi o sui sistemi operativi in esecuzione dietro di esse.

**Scansione Stealth:** Si tratta di una tipologia di scansione che **mira a non essere rilevata** dal sistema bersaglio, con l'obiettivo di sfuggire alla rilevazione da parte dei sistemi di Intrusion Detection.

Originariamente concepito come strumento diagnostico, il port scanning è diventato uno strumento principale utilizzato dai pen-tester, anche se può essere sfruttato per scopi malevoli, simile all'uso iniziale della polvere da sparo per i fuochi d'artificio che oggi è associata a usi più pericolosi.

Esempio di porte aperte (Output di nmap): Nmap non mostra tutte le 65535 porte disponibili su un sistema, ma soltanto quelle effettivamente esposte e accessibili dall'esterno.

```
ftp 21/tcp File Transfer
telnet 23/tcp Telnet
smtp 25/tcp Simple Mail Transfer
finger 79/tcp Finger
login 513/tcp remote login(rlogind)
shell 514/tcp rlogin style (rshd)
printer 515/tcp spooler (lpd)
```

## Funzioni hash

Le funzioni hash sono utilizzate per registrare un segreto su memoria di massa attraverso la generazione di un valore univoco chiamato hash. Questo hash funge da impronta digitale del segreto, consentendo di proteggerlo senza memorizzarlo direttamente. Quando si verifica l'autenticità o l'integrità del segreto, il valore hash memorizzato viene confrontato con quello calcolato, garantendo la sicurezza dei dati e riducendo il rischio di accesso non autorizzato o manipolazione.

# Parte 2: Proprietà, Attacchi e Attaccanti

## Attacchi

Gli **attacchi informatici** possono essere **classificati** in base agli obiettivi che persegono. Alcuni esempi includono:

- **Accesso non autorizzato al sistema**
- **Accesso al sistema a nome di un'altra persona**
- **Ottenimento di privilegi elevati all'interno del sistema**
- **Impersonificazione di un utente autorizzato**
- **Furto di dati sensibili**
- **Attacco di negazione del servizio (DoS)**
- **Mancato recapito dei dati o dei servizi**

Se un attaccante ha accesso fisico al dispositivo, può seguire i seguenti passaggi:

- **Attacco fondamentale:** Prova ad accedere al sistema.
  - ✓ **Contromisura:** Autenticazione tramite PIN, Password o Biometria.
- **Attacco successivo 1:** Cerca di utilizzare le funzionalità del sistema (ad esempio, l'accesso alla casella di posta elettronica).
  - ✓ **Contromisura:** Autenticazione per accedere alle funzionalità.
- **Attacco successivo 2:** Cerca di acquisire dati sensibili.
  - ✓ **Contromisura:** Crittografia dei dati sensibili.

Nella pratica, molte di queste **contromisure** possono **fallire**. Ad esempio, i browser memorizzano le password, esiste una forte resistenza generale all'adozione della crittografia (anche se esistono strumenti come BitLocker di Windows per crittografare i dischi).

### Attacco reale: furto (perdita) di dispositivi

Pratica: contromisure fallite!

- ① **Attacco fondamentale:** accesso al sistema
  - Contromisura: autenticazione al sistema
    - Laptop: riambientazione memoria di massa, boot da dispositivo esterno
    - Smartphone: spesso disabilitata per ragioni di usabilità
- ② **Attacco successivo 1:** uso funzionalità di sistema
  - Contromisura: autenticazione alla funzionalità
    - Laptop: browser registra password
    - Smartphone: app registra password
- ③ **Attacco successivo 2:** acquisizione di dati sensibili
  - Contromisura: crittografia
    - Laptop: riluttanza verso la codifica, quindi accesso a password in chiaro, copia di documento, di dati bancari, etc.
    - Smartphone: idem (?)

## Attacco reale: pirateria digitale

La **pirateria digitale** è un problema serio che riguarda vari aspetti:

- **Furto di proprietà intellettuale:** La natura digitale dei contenuti permette una copiabilità infinita, rendendo difficile proteggere la proprietà intellettuale. Le contromisure possono includere il watermarking, o l'uso di digital stores che offrono un'esperienza utente superiore rispetto al download illegale.
- **Furto di identità:** Il furto di identità può avvenire quando qualcuno riesce a fare login con le credenziali di un altro utente. Le contromisure possono includere l'autenticazione a più fattori, che richiede all'utente di verificare la propria identità in più modi prima di ottenere l'accesso.
- **Furto di marchio:** Il furto di marchio può avvenire quando un logo o un marchio viene utilizzato senza permesso. Le contromisure possono includere l'applicazione delle leggi sul diritto d'autore e dei marchi.

## Proprietà di Sicurezza

- **Impossibilità di Conoscere Tutte le Proprietà di Sicurezza:** Poiché la sicurezza informatica è un campo in continua evoluzione e complesso, è impossibile conoscere tutte le sue proprietà in modo esaustivo.
- **Pilastri Fondamentali (ISO 27001):** Segretezza, Autenticazione e Integrità sono considerati pilastri fondamentali della sicurezza dell'informazione secondo lo standard ISO 27001.
- **CIA (Confidentiality Integrity Availability):** Questi tre principi rappresentano la triade classica della sicurezza informatica, enfatizzando la riservatezza, l'integrità e la disponibilità dei dati.
- **Ulteriori Proprietà a Livello Più Alto:**
  - **Livello 2:** Non Ripudio, Disponibilità.
  - **Livello 3:** Valuta Elettronica, Equilibrio tra Privacy e Legge, Controllo di Accesso e Livelli di Segretezza.

## Definizione di segretezza (Confidenzialità)

**Definizione:** La segretezza si riferisce all'informazione che deve rimanere privata e non deve essere divulgata a entità non autorizzate.

**Password:** Una password rappresenta un segreto condiviso tra due parti, come un utente e un server di autenticazione. È fondamentale che entrambe le parti conoscano il segreto per poter autenticare correttamente l'utente.

### **Misure di Sicurezza:**

- **Crittografia:** Consiste nel convertire i dati in un formato illeggibile a meno che non si disponga di una chiave di decrittazione. Questo garantisce che anche se i dati sono intercettati, non possono essere compresi senza la chiave corretta.
- **Steganografia:** È un'alternativa alla crittografia che non modifica l'aspetto esterno dei dati, ma li nasconde all'interno di altri dati. Ad esempio, si possono modificare i bit meno significativi di un'immagine per memorizzare informazioni senza che queste siano visibili all'occhio umano.

**Esempio:** Modifica dei bit meno significativi di un'immagine per nascondere informazioni al suo interno, che rimangono impercettibili all'occhio umano.

**Misure di Sicurezza Più Robuste:** Alcuni esempi includono l'utilizzo di algoritmi crittografici robusti come AES256 e altri protocolli di sicurezza avanzati.

### Differenza tra codifica e cifratura

- **Codifica:**

Trasforma i dati da un formato all'altro senza l'uso di una chiave segreta. Scopo principale è la conversione dei dati in un formato specifico. Non è una misura di sicurezza.

- **Cifratura:**

Utilizza una chiave segreta per trasformare i dati in modo che siano comprensibili solo a chi possiede la chiave. Scopo principale è garantire la confidenzialità dei dati. È una misura di sicurezza che protegge i dati da accessi non autorizzati.

## Vulnerabilità

Il firewall non riesce a bloccare il file contenente il trojan. Tuttavia, è possibile correggere la vulnerabilità aggiornando il software di lettura dei file in modo che ignori lo script inserito. Questa azione, conosciuta come patching, permette di risolvere la vulnerabilità nel software e proteggere il sistema dagli attacchi.

## Definizione di Autenticazione

**L'autenticazione** è il **processo** di **verifica dell'identità** delle entità, confermando che esse siano effettivamente chi dichiarano di essere. Involge la validazione dell'identità di individui, amici o siti web. Ad esempio, i siti web devono autenticare gli utenti per consentire l'accesso ai servizi offerti. Questo concetto è ampiamente applicato nei browser web e nelle applicazioni mobili basate su browser.

Alcune misure:

- **Conoscenza** (password, PIN)
- **Possesso** (smartcard, smart token)
- **Biometria** (impronte, iride)

## Integrità (coerenza)

L'integrità, come proprietà fondamentale di sicurezza, implica che le informazioni non possono essere modificate da entità non autorizzate.

**Checksum:** Si tratta di un controllo di integrità che verifica la presenza di errori all'interno di un pacchetto dati attraverso una semplice operazione di somma. Tuttavia, non costituisce una misura di sicurezza in quanto non impedisce a entità non autorizzate di modificare i dati.

**Firma Elettronica:** Mentre qualsiasi messaggio che viaggia attraverso la rete può essere alterato da nodi intermediari e anche un messaggio cifrato può essere soggetto a manipolazioni, la firma digitale o elettronica non può essere alterata. Questa firma fornisce un'ulteriore garanzia di integrità, rendendo i messaggi immutabili e autenticati.

## Privatezza (Privacy)

La privacy è il diritto alla segretezza e alla riservatezza delle proprie informazioni personali.

- **Misure di Sicurezza:** Una misura di sicurezza fondamentale per proteggere la privacy è quella di non fornire il consenso alle politiche che richiedono la condivisione o la divulgazione dei dati personali.
- **Diritti del GDPR:** Secondo il Regolamento Generale sulla Protezione dei Dati (GDPR), in caso di cambio di fornitore di servizi, è possibile richiedere all'ente di cancellare tutti i propri dati personali entro 72 ore dalla richiesta.
- **Privatezza come Diritto di Segretezza:** La privatezza viene spesso intesa come il diritto alla segretezza delle proprie informazioni. Ad esempio, una password è considerata segreta, ma non necessariamente privata, poiché può essere condivisa con altre persone, mentre la privacy implica che queste informazioni siano riservate e non accessibili a terzi senza autorizzazione.

## Anonimato

L'anonimato è il diritto di non rivelare la propria identità ad altre entità, consentendo così di nascondere il proprio identificativo.

**TOR (The Onion Router):** TOR è un sistema di routing che offre servizi di anonimato su Internet. Utilizza un sistema a cipolla, rendendo difficile il tracciamento dell'indirizzo IP attraverso la rete tramite IP Traceback, grazie a difficoltà strutturali e topologiche che complicano l'autenticazione.

**Server Anonimizzatore:** Questi server permettono la comunicazione tra utente e servizio, mantenendo anonima la comunicazione tra il server anonimizzatore e il destinatario. Tuttavia, la comunicazione tra l'utente e il server anonimizzatore non è protetta.

**VPN (Virtual Private Network):** A differenza di TLS che protegge i dati in transito, una VPN fornisce all'utente un indirizzo IP locale al chiamato, ma non garantisce l'anonimato.

**Navigazione in Incognito:** Questa modalità non elimina i cookie preesistenti, le credenziali o i cookie registrati temporaneamente. Al di fuori del proprio dispositivo, la navigazione in incognito non offre anonimato, in quanto le attività possono essere tracciate come in una sessione di navigazione normale.

## Proprietà di sicurezza di livello 2



## Non ripudio

**Definizione:** L'entità non può negare la propria partecipazione ad una transazione con uno specifico ruolo. La non-ripudio è una caratteristica di sicurezza che implica la mancanza di fiducia reciproca tra le entità coinvolte, impedendo loro di negare la propria partecipazione o azione nei confronti dell'altra parte.

### **Scenario Tipico: Compravendita**

Nel contesto di una compravendita, la non-ripudio garantisce che entrambe le parti non possano negare di aver compiuto o accettato una transazione.

La **Posta Elettronica Certificata (PEC)** è una tecnologia utilizzata per garantire la non-ripudio nella comunicazione via email. Viene impiegata in contesti legali, come i concorsi pubblici, in quanto equivale a una raccomandata con ricevuta di ritorno. Tuttavia, entrambe le parti coinvolte devono essere dotate di un indirizzo PEC affinché questa garanzia sia valida.

# Autenticazione anonimato e non ripudio

- 1 autenticazione  $\xrightarrow{\text{(ovvio)}}$   $\neg$  anonimato
- 2 anonimato  $\xrightarrow{\text{(ovvio)}}$   $\neg$  autenticazione
  - autenticazione  $\overset{(1,2)}{\equiv} \neg$  anonimato
- 3 autenticazione  $\xrightarrow{??}$  non-ripudio
- 4 non-ripudio  $\xrightarrow{\text{(ovvio)}}$  autenticazione
- 5 anonimato  $\xrightarrow{(2,contra(4))}$   $\neg$  non-ripudio
- 6 non-ripudio  $\xrightarrow{(4,1)}$   $\neg$  anonimato

- L'autenticazione esclude l'anonymato.
- L'anonymato esclude l'autenticazione.
- L'autenticazione garantisce il non ripudio?
  - Quando un utente si autentica su un servizio e le azioni vengono registrate in un log, questo non necessariamente garantisce il non ripudio. I log, simili a checksum, possono essere manipolati durante un attacco se non garantita l'integrità. Pertanto, l'autenticazione da sola non assicura la non ripudiabilità.
- Il non ripudio richiede l'autenticazione.
- L'anonymato implica mancanza di autenticazione, la mancanza di autenticazione implica mancanza di non ripudiabilità. Queste relazioni sono transitive.
- Il non ripudio richiede autenticazione; l'autenticazione, a sua volta, richiede anonymato per transitività.

## Cerimonia di Sicurezza

Protocollo di sicurezza che coinvolge attori umani (quando si effettua un esame universitario siamo parte di una cerimonia)

Anonymato miglior prerequisito per una valutazione equa ad un esame universitario. Come facciamo ad autenticare e rendere anonimo allo stesso tempo uno studente all'esame universitario?

## Separation Of Duty (Atto di fede)

Proprietà che richiede la suddivisione dei compiti o delle responsabilità tra più soggetti. Ad esempio, durante gli esami universitari, una persona potrebbe essere responsabile dell'autenticazione dei candidati, mentre un'altra persona si occupa della valutazione degli esami scritti.

## **Public Auditability**

L'auditabilità pubblica si riferisce all'autenticazione che avviene sotto gli occhi dell'individuo o di un pubblico. Ad esempio, durante un esame universitario, potrebbe essere utilizzato un talloncino contenente il nome dello studente, sigillato all'interno di una busta e poi chiuso con una firma a cavallo della busta per garantire l'integrità fisica e fornire evidenza di eventuali manomissioni (tamper evidence).

Successivamente, l'esame scritto e la busta contenente il talloncino vengono inseriti in un'altra busta che viene aperta solo durante la fase di correzione dell'esame. Tuttavia, non c'è garanzia che il nome contenuto nel talloncino non venga visualizzato prima della correzione del compito, sollevando dubbi sulla sicurezza del processo di auditabilità pubblica.

## **Conciliare Autenticazione e Anonimato**

Anche se si implementa una efficace separazione dei compiti, non risolviamo completamente il dilemma della conciliazione tra autenticazione ed anonimato. Anche un piccolo segno distintivo nell'esame scritto potrebbe compromettere il sistema, permettendo all'esaminatore di identificare il candidato.

L'autenticazione del candidato è un processo separato dall'anonimato richiesto durante l'esame. Tuttavia, l'anonimato può essere infranto quando viene associato il voto all'identità del candidato.

Pertanto, cercare di combinare autenticazione ed anonimato in un'unica soluzione risulta impossibile.

## **Threat Model / Modello di Attaccante**

Nel contesto della sicurezza, è essenziale definire il modello delle minacce o degli attaccanti.

# WATA: Written Authenticated Though Anonymous

Il processo di WATA, acronimo di "Written Authenticated Though Anonymous", prevede due principali aspetti:

- **L'autenticazione dello studente**, per la protezione dell'integrità dell'esame e per la validità dei risultati da parte del docente.
- **L'anonimato del compito**, per garantire la privacy dello studente e ridurre il rischio di influenze esterne sulla valutazione.

Durante l'esame, uno studente riceve un talloncino che lo identifica (token), il quale viene successivamente portato a casa dall'utente al termine del compito. Tuttavia, c'è il rischio che lo studente possa manipolare il talloncino al fine di disassociare il codice a barre dalla propria identità.

Per mitigare questo rischio, il token necessita di una misura di integrità simile a quella presente sulle banconote. Nel caso di WATA, tale integrità è garantita da una firma a cavallo apposta dal docente, la quale non deve fuoriuscire dal token, che non può essere compromessa o rimossa dal talloncino. Questa firma non è immune agli attacchi, ma rende più complesso per lo studente alterare l'associazione tra il talloncino e la propria identità.

## **Anonimato: Sinergia tra Misure di Sicurezza**

Per garantire l'anonimato durante gli esami, è fondamentale adottare una combinazione di misure di sicurezza:

- **Randomizzazione dei compiti:** Assegnare i compiti in modo casuale ai candidati per evitare qualsiasi schema o prevedibilità.
- **Copertura del barcode e altre informazioni scritte:** Durante il processo di autenticazione, è importante coprire il barcode e qualsiasi altra informazione che possa rivelare l'identità dello studente.

Name : \_\_\_\_\_  
Surname : \_\_\_\_\_  
ENRL Number : \_\_\_\_\_ / \_\_\_\_\_  
Date : \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_  
 Signature \_\_\_\_\_

Scissors icon: A pair of scissors is positioned above a dashed line, indicating where to cut the paper.

Barcode: There are two barcodes on the page, one near the top right and one at the bottom right.

Question 1: 1) How Did He/She invent the hot water?  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

## WATA 2

Protocollo cyber-fisico garantisce una sicurezza integrata sia materiale (cartacea) che digitale, consentendo all'esaminato di rimanere anonimo durante la fase di valutazione da parte del docente.

Il protocollo prevede una suddivisione dell'esame in quattro fasi:

- 1. Preparazione**
- 2. Test** (effettuazione dell'esame)
- 3. Valutazione**
- 4. Notifica del voto** allo studente

La presenza dell'invigilator non risolve i problemi di sicurezza quando esaminatore e invigilator sono la stessa persona. Il protocollo è caratterizzato da numerosi aspetti funzionali e si basa su un database contenente 1000 domande, le quali vengono randomizzate e assegnate casualmente a ciascun foglio di esame, con la possibilità di scegliere il numero di domande per compito.

### **Fase 1: Preparazione dell'esame**

1. Estrazione casuale delle domande dal database.
2. Generazione casuale di un ID alfanumerico di lunghezza n.
3. Creazione di una voce nel database contenente l'ID del test e inizializzazione della cella per il voto.
4. Emissione di:
  - a. Primo ID del test (token).
  - b. Form di autorizzazione (token).
  - c. ID del test (sul foglio di esame).
  - d. Domande d'esame.
  - e. Spazi vuoti per le risposte dello studente.
5. Apposizione della firma di integrità da parte del docente sopra il codice a barre e il nome e cognome (bio\_sign).

### **Fase 2: Effettuazione dell'esame**

1. Consegnare formale dei fogli da parte dell'esaminatore all'invigilator.
2. Assegnazione casuale dei compiti da parte dell'invigilator.
3. Compilazione da parte dello studente del modulo di autenticazione (token), firmando nella sezione apposita.
4. Fornisce le informazioni appena scritte all'invigilator per la fase di autenticazione un documento di riconoscimento, nascondendo gli elementi peculiari del compito come ad esempio la prima domanda.

5. Verifica da parte dell'invigilator della validità del documento, dell'iscrizione dello studente nella lista dei registrati all'esame e della coerenza tra i dati del documento di identità e quelli nel token di autenticazione.
6. Risposta dello studente alle domande del compito, completando il test.
7. Rimozione del token dal compito per mantenere l'anonimato, consegna dell'esame all'invigilator per la fase di valutazione.

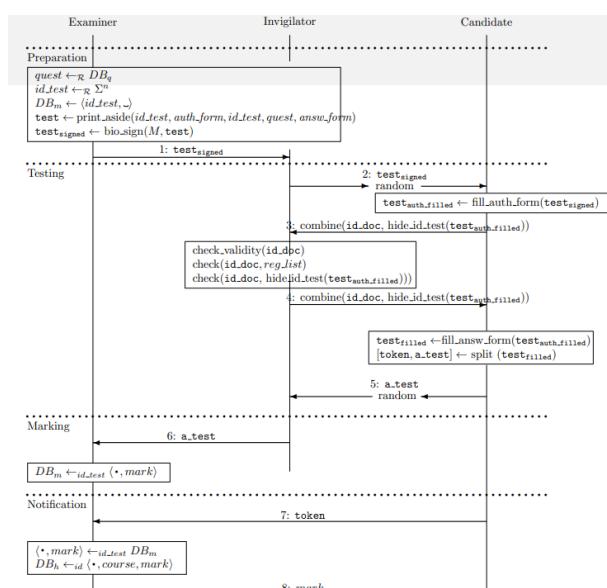
### Fase 3: Valutazione dell'esame

1. Consegnare formale dei test all'esaminatore da parte dell'invigilator.
2. Correzione da parte del docente degli esami scritti.
3. Registrazione del voto nel database.

### Fase 4: Notifica della valutazione

1. Lo studente o un suo delegato si presenta al dipartimento (irrilevante perché il voto va comunque all'autore del compito) con il token tagliato durante la fase 2 per la notifica del voto.
2. Registrazione formale, in un ulteriore database (storico - h), delle informazioni del corso, dell'identità dello studente e del voto assegnato. Attualmente questo database è Smartedu.

Nella fase finale, c'è il rischio che solo il docente, attraverso il database, possa vedere il voto dell'esame e manipolarlo, ad esempio dicendo allo studente di essere stato bocciato o di aver ottenuto un voto basso, anche se il punteggio effettivo del compito de-anonimizzato è più alto. Inoltre, potrebbe omettere elementi dall'esame durante la valutazione e tenerli separatamente. Per prevenire questo, la comunicazione del voto dovrebbe essere pubblica, ad esempio tramite screen-sharing, per evitare discrepanze tra il voto comunicato e quello effettivamente assegnato. Inoltre, c'è un problema di integrazione tra il protocollo WATA e il database finale di valutazione (GOMP) per renderlo pubblicamente verificabile.



## Disponibilità (non Denied of Service)

**Definizione:** L'affidabilità del servizio è una caratteristica di sicurezza fondamentale, ed è essenziale garantire che il sistema sia disponibile e funzionante in ogni momento.

Il rifiuto del servizio è considerato un attacco DoS (Denial of Service), come ad esempio una fork bomb che genera ripetutamente processi figlio, sovraccaricando il sistema e impedendone il corretto funzionamento.

### **Misure di mitigazione del problema:**

- Autenticazione e restrizione dell'accesso agli utenti.
- Aumento della complessità dell'accesso al sistema, impegnando il chiamante computazionalmente e trasferendo il carico al chiamante per metterlo in attesa.

## Proprietà di sicurezza di livello 3



## Controllo d'accesso

**Definizione:** Ogni utente ha accesso solamente alle risorse e ai servizi per i quali è autorizzato.

### **Esempi:**

- Autenticazione dell'utente.
- Definizione delle policy di sicurezza.
- Implementazione delle politiche (liste di controllo degli accessi (ACL), utenti limitati)

## Esempio di una politica di sicurezza

Policy di sicurezza per l'accesso a dei file:

1. Un utente ha il permesso di leggere un qualunque file pubblico
2. Un utente ha il permesso di scrivere solo su file pubblici di sua proprietà
3. Un utente ha il divieto di fare il downgrade di un file, ovvero prendere una versione più vecchia di un file
4. L'utente ha l'obbligo di cambiare la propria password quando scade
5. Un utente segreto ha il permesso di leggere su un qualunque file non pubblico, come se fosse un utente root
6. Un utente segreto ha il permesso di scrivere su qualunque file non pubblico
7. Un amministratore ha il permesso di sostituire un qualsiasi file con una versione più obsoleta, il nome utente segreto è solo evocativo
8. Un utente che non cambia la sua password scaduta ha il divieto di compiere qualunque operazione
9. Un utente che non cambia la password non ha discrezione di cambiarla

## Elementi di una politica di sicurezza

- **Ruoli**
  - Utente, utente segreto, sistemista, utente negligente
- **Utente**
  - Qualunque entità che ricopra un certo ruolo
- **Operazioni**
  - Leggere, scrivere, downgrade, cambio password
- **Modalità** (Verbi servili (volere, dovere, potere), nessuna volontà ma obblighi)
  - Obbligo, permesso, divieto, discrezionalità

L'utenza prende il requisito autorizzativo di un ruolo

## Modalità e relazioni fra loro

x = operazione

- **Modalità base**
  - Obbligatorio(x)
- **Modalità derivate**
  - Vietato(x)
  - Permesso(x)
  - Discrezionale(x)
- **Loro definizioni**
  - Vietato = Obbligatorio (not x)
  - Permesso = Non obbligatorio (not x)
  - Discrezionale = Non obbligatorio(x)

**Esempio** x = fumare

- **Vietato fumare** = Obbligatorio non fumare
- **Permesso di fumare** = Non è obbligatorio non fumare
- **Discrezione nel fumare** = Non è obbligatorio fumare

## Inconsistenze di una politica

Le inconsistenze possono essere di due tipi:

- **Contraddizione:** Obbligatorio(x) <-> Non obbligatorio(x)
- **Dilemma:** Obbligatorio(x) <-> Obbligatorio(not x)

**Esempio di contraddizione:**

Obbligatorio fare l'esame <-> Non obbligatorio fare l'esame (a livello di modalità)

**Esempio di dilemma:**

Obbligatorio fare l'esame <-> Obbligatorio non fare l'esame (a livello di operazione)

## Politiche in pratica: MAC

Il Controllo d'Accesso Obbligatorio (MAC) stabilisce una politica riguardante la scrittura delle policy ed è fondato sull'obbligo, con radici nell'ambito militare.

**Esempio:** Il sistema richiede all'utente di cambiare la password del sistema ogni sei mesi. È importante notare che i sistemi Windows e Linux che utilizziamo attualmente non implementano il Controllo d'Accesso Obbligatorio (MAC).

## Politiche in pratica: RBAC

Il Controllo d'Accesso Basato sui Ruoli (RBAC) si fonda su politiche non obbligatorie e si basa sulla definizione di ruoli utente, come amministratori e utenti base. In questo contesto, il controllo degli accessi è determinato dal ruolo dell'utente.

Il concetto chiave è rappresentato dalle definizioni di Permesso e Divieto:

- Permesso <-> Non è vietato (x)
- Vietato <-> Non è permesso (x)

Questa politica è ampiamente utilizzata nei sistemi operativi attuali.

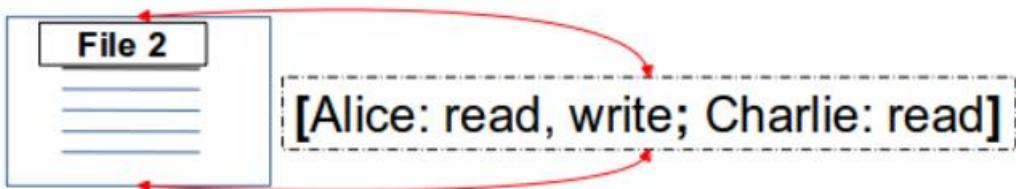
## Meccanismi implementativi: ACM

Questa matrice rappresenta lo stato dei permessi all'interno di un sistema. Le righe corrispondono ai soggetti (ad esempio, gli utenti) e le colonne corrispondono agli oggetti (ad esempio, i file). Una cella specifica nella matrice, indicata come  $ACM[s, o]$ , mostra i permessi che il soggetto 's' ha sull'oggetto 'o'. La dimensione della matrice è proporzionale alla dimensione del sistema. In Linux, un soggetto 's' può essere un gruppo di utenti.

	File1	File2	File3	Program1
Alice	<i>read, write</i>	<i>read</i>		<i>execute</i>
Bob	<i>read</i>		<i>read, write</i>	
Charlie		<i>read</i>		<i>read, execute</i>

## Meccanismi implementativi: ACL e CAL

**Lista di Controllo degli Accessi (ACL):** Questa è essenzialmente ogni colonna dell'ACM, registrata con l'oggetto specifico. Mostra tutti i permessi che vari soggetti hanno su un particolare oggetto.



**Lista delle Capacità (Capability List):** Questa è ogni riga dell'ACM, registrata con il soggetto specifico. Mostra tutti i permessi che un particolare soggetto ha su vari oggetti.



## Tassonomia di attaccanti

Si possono classificare per moventi: Ricchezza, Informazioni sensibili, Potere, Gloria, Divertimento.

- Vari moventi
  - Ricchezza
  - Informazioni sensibili
  - Potere
  - Gloria
  - Divertimento
  - : :

- Varie classificazioni
  - Diritti
  - Risorse
  - Esperienza
  - Rischio accettato
  - : :

1. Hacker (cracker), 2. Attaccanti interni,
3. Spionaggio industriale, 4. Servizi segreti,
5. Organizzazioni criminali/terroristiche, 6. Difesa

## Modelli di attaccante

**Definizione:** Un modello di attaccante specifica le capacità offensive di un preciso attaccante

Cambiare il modello potrebbe cambiare il valore di verità di un'affermazione di sicurezza e fare un'analisi caso peggiore richiede un modello massimamente offensivo.

### Modello di attaccante: Dolev-Yao

Il Modello Dolev-Yao, denominato dai suoi creatori, è un modello altamente aggressivo progettato per valutare la sicurezza dei nostri prodotti. È utilizzato per analizzare la sicurezza delle reti e rappresenta un tipo di attacco. Nella sua implementazione, viene considerato uno scenario in cui due servizi segreti, identificati come 007, operano rispettivamente a Londra e a New York, comunicando tra loro in modo sicuro (è importante spiegare in che modo viene definita la sicurezza durante l'esame).

L'obiettivo è creare un modello estremamente robusto che possa rappresentare le situazioni più critiche, ipotizzando che **tutti gli attori nel mondo siano coinvolti nel tentativo di compromettere questa comunicazione** tra gli agenti segreti 007.

In questo contesto, **l'intera rete è considerata ostile** e aggressiva al massimo livello.

L'aggressività massima indica che l'attaccante, che è rappresentato dalla rete stessa, ha il controllo completo su tutte le operazioni.

### Modello di attaccante: General Attacker

Nel contesto dei modem 56k, la linea telefonica e quella di rete erano integrate, il che significava che una volta connessi a Internet, la linea telefonica diventava inutilizzabile. Con l'ISDN, invece, la linea telefonica e quella di rete erano separate. Il modello proposto da Dolev-Yao non è più considerato realistico poiché ogni attaccante ha i propri interessi e obiettivi. Nello specifico scenario di trasferimento di un messaggio tra Alice e Bob nel contesto del modello Dolev-Yao, si presume che l'attaccante intermedio, ovvero la rete stessa, possa intercettare il messaggio. Tuttavia, nel modello di Attaccante Generico, un attaccante potrebbe sfruttare un attacco senza informare nessuno. Chiunque può essere attaccante senza interesse a colludere con altri, al peggio con capacità di totale controllo della rete, ma senza violare la crittografia.

# Parte 3: Autenticazione

## Ambientazioni di autenticazione

- **Utente-computer:** Accedere ad un sistema tramite password o biometria
- **Computer-computer:** un computer vuole accedere ad un altro indipendentemente dal suo utente (IP, MAC)
- **Computer-utente:** L'utente è convinto ed ha evidenza dell'identità del sito
- **Utente-utente:** meno nota

## Autenticazione utente-computer

- **Basata su conoscenza:** password, passphrase, PIN
- **Basata su possesso:** smart card, smart token
- **Basata su biometria:** impronta, iride, tono vocale

## Autenticazione basata su conoscenza

Chiunque conosca la password di un utente per un sistema può impersonare in toto quell'utente col sistema, Semplice ed economica da implementare

Si corrono i rischi di:

- **Guessing:** riuscire ad indovinare la password (brute force)
- **Snooping:** Sbirciare la password quando viene inserita
- **Spoofing:** Scoprire la password tramite fake login (trojan)
- **Sniffing:** Intercettare la password durante la fase di trasmissione (Wireshark)

## Guessing: Attacchi

1. Provo inizialmente con password brevi, tipiche e relative all'utente
2. Attacco dizionario: Vengono provate tutte le parole di un dizionario, arricchite con doppie parole, (0 al posto di o o 1 al posto di i)
3. Attacco di forza bruta: Vengono provate tutte le parole costruibili in un dato vocabolario (alfanumerico, simboli e caratteri speciali) di lunghezza via via crescente, tutte le combinazioni di lunghezza 1, poi lunghezza 2... complessità esponenziale

Controllo a soglia coerente con le linee guida NIST del 2016, tecnologia già implementata nei bancomat.

## Guessing: Contromisure

1. **Controllo sulla password:** Lunghezza della password e presenza di lettere, simboli e caratteri speciali
2. **Controllo sul numero di inserimenti:** Controllo a soglia
3. **Uso di CAPTCHA:** Completely Automated Public Turing Test To Tell Computers and Human Apart (Test di Turing Completamente Automatico e Pubblico per Distinguere Computer e Umani.)

Alan Turing fu il primo a teorizzare l'intelligenza artificiale e il test di Turing è stato superato da una macchina solo recentemente. Un esempio pratico è la visualizzazione di parole distorte che devono essere riconosciute.

Le CAPTCHA cercano di portare il test di Turing a un livello grafico. Un algoritmo recente di Google, che doveva riconoscere i numeri civici per oscurarli, è stato in grado di superare il 99% delle CAPTCHA alfanumeriche.

Google ReCaptcha è una tecnologia completamente rivoluzionaria che richiede all'utente di dimostrare di essere umano. L'algoritmo dietro a questo sistema non è noto pubblicamente, ma si suppone che tenga traccia dei movimenti del mouse.

## Norme per una buona password

Bilanciare complessità e mnemonicità, questo implica:

- Non usare una parola del dizionario
- Usare almeno 8 caratteri
- Non usare la stessa password per autenticazioni diverse

## Mantenere una password

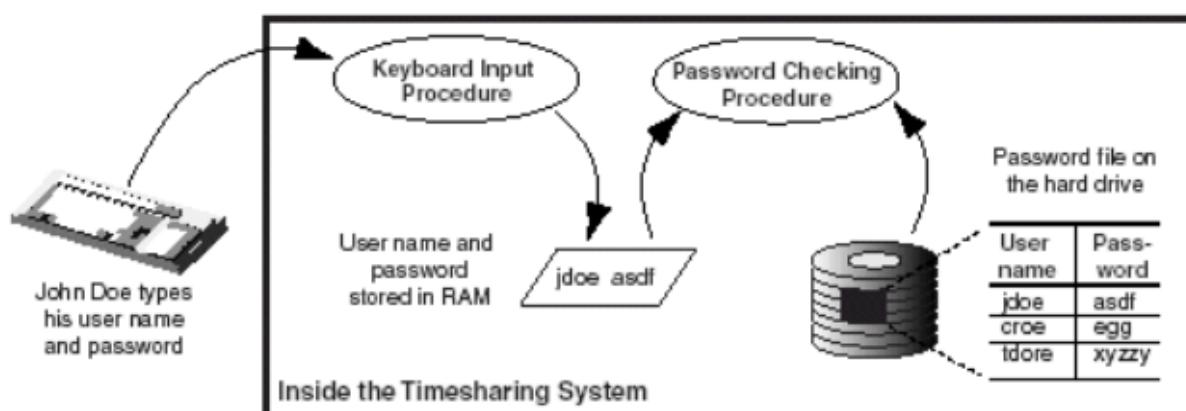
La password va mantenuta in qualche modo sul sistema al quale garantisce l'autenticazione utente.

**Soluzione:** memorizzare in un database ciascuna password in qualche forma

## CTSS: Compatible Time Sharing System

Password memorizzate in chiaro su un file di sistema protetto da politica di sicurezza.

Si associano in una tabella username e password. Molte vulnerabilità e parecchi attacchi registrati in passato.



## Autenticazione basata sul possesso

Nell'autenticazione basata sul possesso, l'identità dell'utente è verificata tramite il possesso di un oggetto magnetico o elettronico, come una smart card o un token.

Questo oggetto può contenere informazioni sensibili memorizzate in modo completamente leggibile o tramite un'interfaccia funzionale coerente.

Tuttavia, ci sono dei limiti:

- Il processo di autenticazione riconosce l'oggetto, non direttamente l'utente.
- Il rischio di smarrimento dell'oggetto è più elevato rispetto alla dimenticanza di una password.

Per affrontare questi limiti, una soluzione comune è **'l'autenticazione a due fattori'**, come nel caso dei bancomat. È necessario possedere la carta (oggetto) e conoscere il PIN per utilizzarla, aumentando così il livello di sicurezza.

## Smart Token

Lo Smart Token è un dispositivo utilizzato per l'autenticazione che si basa su un PIN e genera un One-Time Password (OTP) accettato una sola volta dal server. È comunemente impiegato per l'autenticazione su siti web, richiedendo la conoscenza di una password oltre all'OTP generato dal dispositivo.

Esistono due tipologie principali di token:

- **A pulsante:** Genera un OTP premendo un pulsante. Tuttavia, se il token viene smarrito, chiunque lo trovi potrebbe leggere l'OTP.
- **A PIN:** Richiede l'inserimento di un PIN prima di generare l'OTP, rendendo il passaggio dell'autenticazione attraverso lo smart token più robusto.

Nel primo caso, un sito potrebbe richiedere una password di lunghezza 12 e fornire uno smart token a pulsante. Nel secondo caso, potrebbe richiedere una password di soli 8 caratteri e richiedere un PIN per l'accesso al token. Le due tecnologie sono praticamente equivalenti in termini di sicurezza.

Un'alternativa moderna agli smart token è rappresentata dalle app mobili che offrono autenticazione biometrica. Tuttavia, c'è una differenza significativa tra gli smart token e le app: nel primo caso, il software è eseguito su un sistema isolato nel dispositivo, mentre nel secondo caso lo smartphone è connesso al mondo esterno, aumentando potenzialmente le vulnerabilità.

## Funzionamento di uno Smart Token

Lo smart token funziona utilizzando una chiave segreta (seme) memorizzata dalla fabbrica. Prende informazioni esterne come il PIN e l'ora per generare tramite un algoritmo di funzione pseudo-casuale (PRF) una one-time password (OTP). La OTP viene visualizzata sul display e rinnovata ogni 30-90 secondi. Minimizza i rischi di guessing e sniffing. Lo smart token e il server condividono un algoritmo comune e hanno orologi sincronizzati per garantire che la password generata sia accettata.

## Discussione

È importante non confondere la conoscenza con il possesso: il possesso di un oggetto può spesso essere sostituito da un'informazione che rappresenta quell'oggetto, come nel caso delle stampanti 3D. Il fatto che un token generi una OTP rappresenta quindi un miglioramento rispetto ai tradizionali sistemi di autenticazione come il bancomat.

## Autenticazione basata su biometria

L'autenticazione basata su biometria si basa sul possesso di caratteristiche univoche del corpo per confermare l'identità dell'utente. Queste caratteristiche possono essere:

- **Fisiche:** Come impronte digitali, forma della mano, impronta della retina o del viso.
- **Comportamentali:** Come la firma, il timbro di voce, la grafia o la dinamica della digitazione.

Anche se tecnicamente meno precisa dei primi due metodi, l'autenticazione basata su biometria è comunque affidabile poiché non esistono due campioni biometrici uguali.

# Funzionamento

## • Fase iniziale di campionamento

- Esecuzione di più misurazioni sulla caratteristica d'interesse
- Definizione di un template che media le misurazioni e rappresenta la caratteristica

## • Autenticazione

- Decisa dal confronto fra caratteristica appena misurata e template
- Successo se i due corrispondono a meno di una tolleranza prestabilita

Questo metodo si basa sulla misurazione della distanza tra la firma biometrica dell'utente e quella registrata nel sistema.

Today's Biometric Signature from Cathy:	Cathy's Stored Biometric Pattern:	Tim's Stored Biometric Pattern:
389	390	284
416	418	570
501	502	534
468	471	501
353	355	399

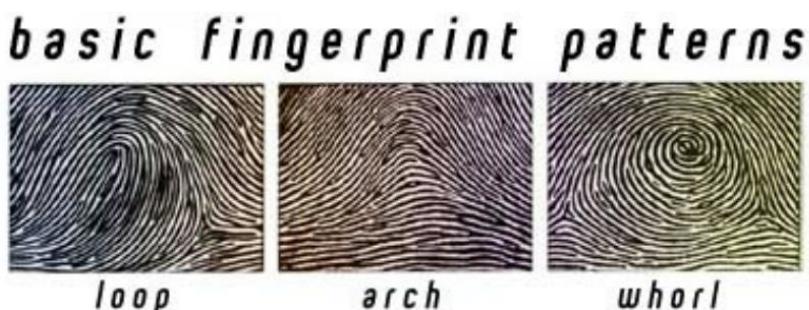
Distance = 4 from that signature      Distance = 199 from that signature

Stored Biometric Pattern  $\equiv$  Template

Le impronte digitali sono un metodo di autenticazione storico, ottenute tradizionalmente con inchiostro, e si suddividono in tre categorie principali: Loop, Arch e Whorl.

### ■ Tre schemi preminanti

- **Loop** (60%), linee circolari che escono verso l'esterno
- **Arch** (5%), linee poco circolari che escono
- **Whorl** (35%), linee circolari che non escono



# Parte 4: Cenni di crittografia

## Crittografia: Concetti di base

La crittografia è l'arte di trasformare le informazioni da una rappresentazione a un'altra, mantenendo la stessa semantica ma con una sintassi differente. Questa pratica ha radici antiche, risalenti all'antica Grecia, ma ha subito sviluppi cruciali negli anni '70. Prima di questo periodo, la cifratura era principalmente simmetrica, ovvero si utilizzava una sola chiave. Tuttavia, negli anni '70 ha preso forma la cifratura asimmetrica. Ad esempio, l'ENIGMA utilizzava un metodo di cifratura simmetrica con una chiave variabile. Questa tecnologia è ampiamente impiegata su Internet, come nell'utilizzo di HTTPS al posto di HTTP o di SSH invece di Telnet.

Cifrare è simile a tradurre in un'altra lingua: si modifica la sintassi (rappresentazione) ma non la semantica:

- **Encrypt:** Ciao -> Hello
- **Decrypt:** Hello -> Ciao

Le funzioni di crittografia sono pubbliche e accessibili a tutti.

La differenza tra cifratura e traduzione sta nella presenza di un parametro chiamato chiave crittografica. Normalmente, dalla stringa cifrata non è possibile estrarre l'informazione contenuta.

Un esempio di cifrario è il Cifrario di Cesare (spostamento alfabetico):

- **Encrypt(m, k):** CIAO -> DLBP (dove m = messaggio, k = chiave = 1)
- **Decrypt(m, k):** DLBP -> CIAO

Il compito della cifratura è nascondere la semantica di un messaggio, ma non la trasmissione del messaggio stesso, il che è significativo in termini di privacy.

Romperne la crittografia significa acquisire la semantica senza conoscere la chiave di cifratura (senza dover eseguire il processo di decifratura). Con un po' di fortuna, è anche possibile scoprire la chiave di cifratura, ma è un evento molto raro.

## Crittosistema

Un crittosistema è costituito da una coppia di algoritmi:

- Uno per crittografare e produrre un crittostesto ( $\varepsilon$ ).
- Uno per decriptare e produrre un testo in chiaro ( $\mathcal{D}$ ).

Ciascun algoritmo richiede due input: un testo e una chiave.

- Dato un testo  $m$  e una chiave  $k$ , il testo viene codificato come  $\varepsilon(m, k)$ , indicato come  $m_k$ .
- Dato un crittostesto  $m_k$  e una chiave  $k'$ , il crittostesto viene decodificato come  $\mathcal{D}(m_k, k')$ , che produce  $m$  se specifiche condizioni legano  $k$  con  $k'$ .
- Non è necessario che  $k$  sia uguale a  $k'$ .

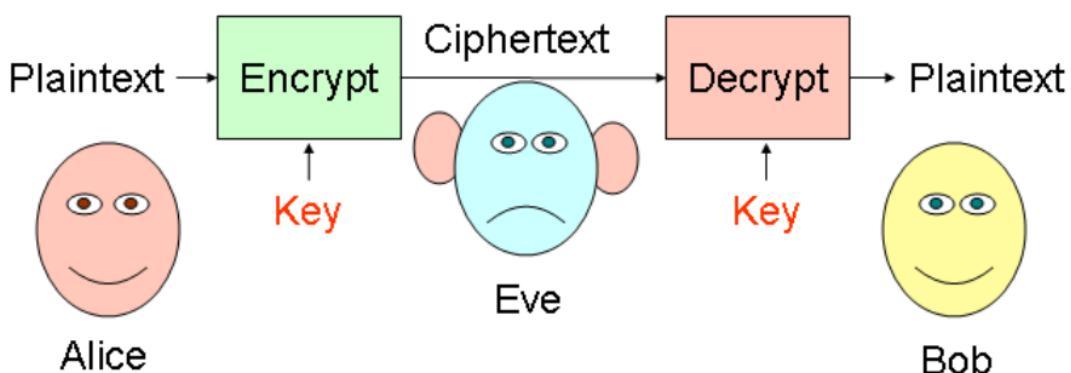
Quando si ha un crittostesto  $m_k$ , non è essenziale che si possiedano le caratteristiche per decifrare il messaggio, ovvero applicare l'algoritmo di decrittazione. Decifrare con successo significa scoprire il testo in chiaro di un messaggio cifrato.

La chiave può essere la stessa usata per crittografare (in algoritmi simmetrici) oppure la sua inversa (in cripto algoritmi moderni, ovvero asimmetrici).

L'associazione tra testo in chiaro e crittostesto dipende da un certo sistema.

## In rete

Se Alice vuole inviare un messaggio cifrato a Bob e solo Alice e Bob conoscono la chiave o le chiavi crittografiche solo loro potranno comunicare, Eve non conoscendo la chiave crittografica può solo provare a decifrarlo, anche tramite brute force, ma non c'è la certezza di riuscire ad effettuare ciò con successo. Il crittosistema va considerato come pubblico. Le chiavi crittografiche tipicamente vanno mantenute segrete!



## Segretezza di una chiave

Se la lunghezza del segreto è  $k$ , ci sono  $2^k$  possibili combinazioni per generare la chiave di cifratura, e la probabilità di indovinarla è  $\frac{1}{2^k}$ .

È quindi fondamentale avere un valore  $k$  elevato, spesso di 256 bit (come nell'AES 256).

La probabilità di rivelazione della chiave di un algoritmo crittografico non supera mai  $\frac{1}{2^k}$ .

## Crittografia simmetrica

La crittografia simmetrica comporta la decifrazione del testo utilizzando la stessa chiave impiegata per cifrare il messaggio.

È possibile decifrare correttamente il messaggio solo quando  $k' = k$ .

### **Equazione di correttezza:**

- $\mathcal{D}(\mathcal{E}(m, k), k) = m$ : La decrittazione utilizzando  $k$  restituisce  $m$ .
- $\forall k_1. k_1 \neq k \rightarrow \mathcal{D}(\mathcal{E}(m, k), k_1) \neq m$ : Se  $k' \neq k$ , non otteniamo il messaggio originale  $m$ .

Se un aggressore tenta di decifrare utilizzando chiavi casuali, non otterrà il messaggio corretto.

### **Esempi:**

- Cifrario di Cesare, DES, 3DES
- Lunghezza tipica della chiave: 128/256 bit
- Velocità di esecuzione

## Crittografia simmetrica in rete

Fissato un agente A (macchina, utente), esso sia munito di chiave simmetrica, indicata come  $k_a$ .

$k_a$  è detta chiave a lungo termine perché il suo intervallo di validità è molto lungo, indipendente dalla specifica sessione di comunicazione.

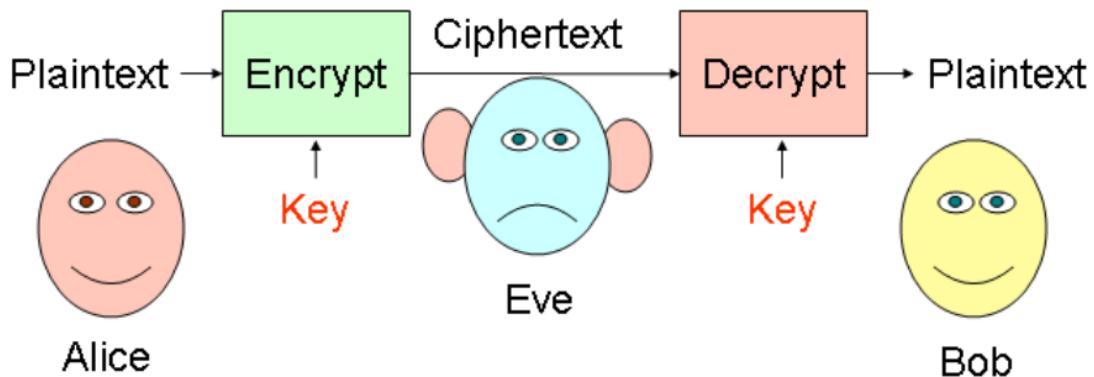
Una chiave a lungo termine, dal punto di vista della sua validità, non è un segreto che cambia frequentemente, come il numero della nostra carta di credito o la nostra password che viene cambiata ogni circa 6 mesi.

Un segreto che rimane valido per molto tempo è di grande importanza; più è lungo il periodo di validità, maggiore è l'importanza del segreto.

Immagina che Alice abbia una chiave a lungo termine e voglia inviare un messaggio in modo confidenziale, quindi lo cifra con questa chiave. Anche se Eve intercettasse il criptotesto, non riuscirebbe a decifrarlo con successo, garantendo così la confidenzialità. Quando Bob riceverà il criptotesto, proverà a decifrarlo ma dovrà conoscere la chiave.

Supponiamo che esista un modo sicuro per trasmettere la chiave in modo confidenziale. Tuttavia, non è necessario che Alice venga a conoscenza della chiave di Bob. Non c'è una connessione diretta tra la chiave cifrata e la password, e non è una buona pratica trasferire la chiave a lungo termine. La differenza principale sta nella lunghezza: le password sono generalmente corte (circa 8 caratteri, 64 bit), mentre le chiavi di cifratura sono molto più lunghe (256 bit).

Una chiave a breve termine è condivisa tra Alice e Bob solo per una sessione specifica. Il suo valore è contingente e contestualizzato, quindi è necessario un modo sicuro per trasmetterla. Non possiamo inviarla in chiaro, altrimenti sarebbe facilmente intercettabile, compromettendo così la sicurezza della chiave di cifratura.



## Limiti della crittografia simmetrica

- **Limite 1:** A non vuole rivelare  $k_a$  a B; quindi, questo non può decriptare
  - Sia  $k_{ab}$  una chiave dedicata specificatamente a questa sessione fra A e B
  - $k_{ab}$  è detta chiave a breve termine o chiave di sessione
  - Può essere condivisa fra i due agenti
- **Limite 2:** servirebbe una chiave di sessione per ogni coppia di agenti che vogliono comunicare fra loro
- **Limite 3:** come condividere  $k_{ab}$  fra A e B pur mantenendola confidenziale?

Se Alice non vuole divulgare la sua chiave di cifratura a Bob: si può inviare una chiave a breve termine ( $k_{ab}$ ) che può essere condivisa tra i due utenti.

**Scalabilità della soluzione:** Se Alice desidera comunicare con Bob, necessita di una chiave di sessione specifica; se invece vuole comunicare con Charlie, avrà bisogno di un'altra chiave di sessione, diversa da quella precedente. Ogni coppia di utenti richiede una chiave di sessione unica.

**Problema:** Come condividere la chiave di sessione in modo sicuro e confidenziale?

## Crittografia asimmetrica

### **Definizione:**

- Ogni chiave  $k$  ha una sua inversa denotata  $k^{-1}$
- Ciascuna chiave non si può ricavare dall'altra, pertanto la coppia va generata monolicamente
- L'unico modo per estrarre il testo in chiaro da un crittotesto è decodificare quest'ultimo con l'inversa della chiave usata per costruirlo

### **Equazione di correttezza:**

- $\mathcal{D}(\mathcal{E}(m, k), k^{-1}) = m$ : La decrittazione utilizzando  $k^{-1}$  restituisce m.
- $\forall k_1. k_1 \neq k^{-1} \rightarrow \mathcal{D}(\mathcal{E}(m, k), k_1) \neq m$ : Se  $k' \neq k^{-1}$ , non otteniamo il messaggio originale m.

## Come si generano le chiavi di cifratura?

Il generatore di chiavi produce simultaneamente la coppia di chiavi. Una metà è tenuta privatamente (chiave privata), mentre l'altra metà è resa pubblica (chiave pubblica).

## Esempi di crittosistemi asimmetrici:

DSA, RSA con chiave di 1024 bit. La chiave è molto più lunga rispetto a quella simmetrica, rendendola più robusta ma anche richiedendo più risorse computazionali.

Le operazioni di encryption asimmetriche sono computazionalmente più costose rispetto a quelle simmetriche. Di conseguenza, è utile utilizzare sistemi ibridi che sfruttino entrambe le tecnologie, simmetrica ed asimmetrica.

Nella crittografia asimmetrica, si presuppone che ogni dispositivo disponga di una coppia di chiavi, una pubblica e una privata.

## Esempio di uso di RSA semplificato:

Si pubblichi:

- $n = 33$ ;

Si prendano:

- $k = 3$
- $k^{-1} = 7$

Siano:

- $\varepsilon(x, e) = x^e \bmod n$
- $\mathcal{D}(x, d) = x^d \bmod n$

Allora, preso  $m = 7$ , si ha  $\varepsilon(m, k) = 13$  quindi  $\mathcal{D}(\varepsilon(m, k), k^{-1}) = 7$

## Crittografia asimmetrica in rete

Fissato un agente A (macchina, utente), esso sia munito di coppia di chiavi asimmetriche, indicate come  $k_a$  e  $k_a^{-1}$  a lungo termine:

- $k_a$  è resa nota a tutti ed è detta **chiave pubblica** di A
- $k_a^{-1}$  è il segreto di A ed è detta **chiave privata** di A

Se si utilizza la chiave privata per cifrare, è necessario utilizzare quella pubblica per decifrare.

Quando Alice decide di cifrare un messaggio, ha a disposizione tre chiavi:

- La sua chiave pubblica.
- La chiave pubblica di Bob.
- La sua chiave privata.

Alice rende nota a tutti la sua chiave pubblica e mantiene segreta la sua chiave privata per cifrare il messaggio. Bob riesce a decifrare il messaggio utilizzando la chiave pubblica di Alice.

Tuttavia, poiché la chiave è pubblica, può essere utilizzata anche da un attaccante senza problemi.

Se l'**obiettivo della sicurezza** è l'**autenticazione** e non la segretezza, Alice cifra con la sua **chiave privata** e tutti possono decifrare utilizzando la chiave pubblica di Alice. Poiché la **confidenzialità del messaggio non è importante** e si può dedurre che la chiave privata appartiene ad Alice, si riesce ad autenticarla con successo.

Se l'**obiettivo della sicurezza** è la **segretezza**, allora Alice sceglierà la **chiave pubblica** di Bob per cifrare. È necessario conoscere l'inversa della chiave pubblica per poterla decifrare correttamente.

Rimosso il problema della condivisione del segreto iniziale!

Ma se ne crea un altro...

## Limite della crittografia asimmetrica

Esistono anche problemi con la crittografia asimmetrica, come possiamo essere certi che la chiave pubblica di Alice appartenga effettivamente ad Alice?

Se Alice cifra con la sua chiave privata e Bob decifra il messaggio con una chiave pubblica di un soggetto diverso da Alice, otterrà un messaggio leggibile e comprensibile. Questo significa che Bob autenticherà X invece di Alice, quindi è necessaria una certificazione.

Un altro esempio è quando Alice intende inviare un messaggio confidenziale a Bob utilizzando la chiave pubblica di ciphatura di Bob. Tuttavia, Alice potrebbe erroneamente utilizzare la chiave pubblica di un soggetto diverso da Bob e consegnare il messaggio a un altro utente.

**Certificazione:** Come associare correttamente una chiave pubblica al suo legittimo proprietario, ovvero al legittimo proprietario della metà privata.

Cosa succede se questa associazione non funziona? Esempio:

- A intenda spedire  $m$  a B in maniera confidenziale
- A intende quindi spedire  $m_{k_b}$
- A pertanto prende la chiave  $k$  ma erroneamente crede che sia  $k = k_b$  mentre risulta  $k = k_c$
- A costruisce  $m_k$  e lo spedisce a B
- B, ricevuto  $m_k$ , lo decripta ma ottiene  $m'$  con  $m \neq m'$
- C, intercettato  $m_k$ , può decriptarlo con  $k_c^{-1}$  ottenendo  $m$

## Riepilogo crittografia simmetrica ed asimmetrica

- **Crittografia simmetrica:** Per condividere un messaggio in modo sicuro, si utilizza un sistema di scambio di chiavi protetto o si trasmette la chiave simmetrica in modo sicuro prima di inviare il messaggio. Questo assicura la segretezza del messaggio.
- **Crittografia asimmetrica:** Per verificare il proprietario di una chiave pubblica, è necessaria una certificazione, che può essere ottenuta da un'autorità di certificazione affidabile. Questo processo conferisce autenticità alla chiave pubblica del destinatario.

Utilizzando queste tecniche, è possibile ottenere le proprietà di sicurezza di livello 1 (Segretezza, Autenticazione ed Integrità)

# Crittosistema sicuro

## Definizione:

- Sia calcolato  $\varepsilon(m, k)$  per ogni testo  $m$  e chiave  $k$ ;
- Sia calcolato  $D(\varepsilon(m, k), k_1) = n$  per ogni chiave  $k_1$  tale che  $k_1 \neq k$  se il crittosistema è simmetrico, o  $k_1 \neq k^{-1}$  se il crittosistema è asimmetrico;
- Allora l'accesso a  $n$  non aumenti significativamente la probabilità di un attaccante di indovinare  $m$  o sue porzioni

Un crittosistema si dice sicuro se la probabilità di scoprire la chiave crittografica non migliora nonostante la pubblicità dell'algoritmo.

# Hash crittografico

Una funzione crittografica di hash è un algoritmo matematico che converte dati di lunghezza arbitraria in una stringa binaria di dimensione fissa, chiamata valore di hash o message digest.

È progettata per essere unidirezionale, rendendo difficile invertirla. Deve essere unica per ogni messaggio, deterministica (lo stesso input produce sempre lo stesso output), e veloce da calcolare.

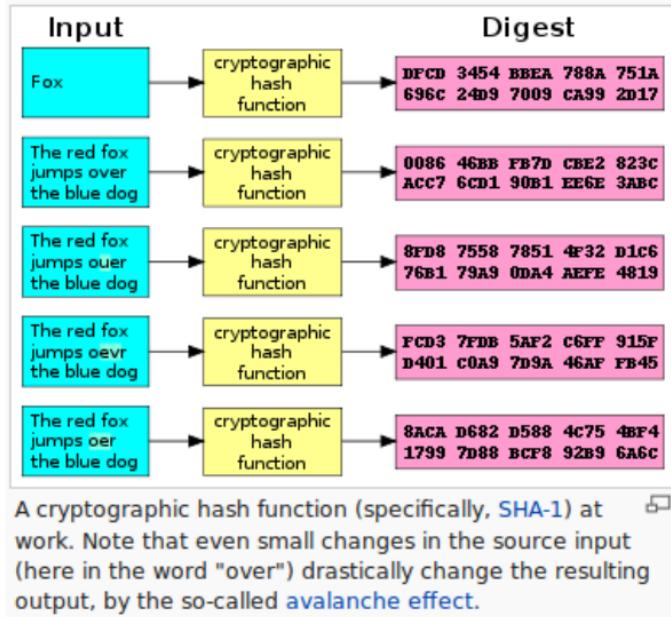
Le sue proprietà la rendono adatta per firme digitali, codici di autenticazione dei messaggi e altre forme di autenticazione, oltre che per la rilevazione di impronte digitali, dati duplicati e checksum per la corruzione dati.

Caratteristiche delle funzioni di hash:

- Il calcolo dell'hash è molto semplice.
- È impossibile generare il messaggio originale partendo dall'hash.
- Anche cambiando un solo bit nel messaggio, l'hash cambierà completamente, generando codici totalmente diversi.

Le funzioni di hashing e di cifratura sono diverse perché tramite l'hashing non è possibile rigenerare il messaggio originale dall'hash, mentre con la cifratura si possono utilizzare due parametri di input per l'operazione.

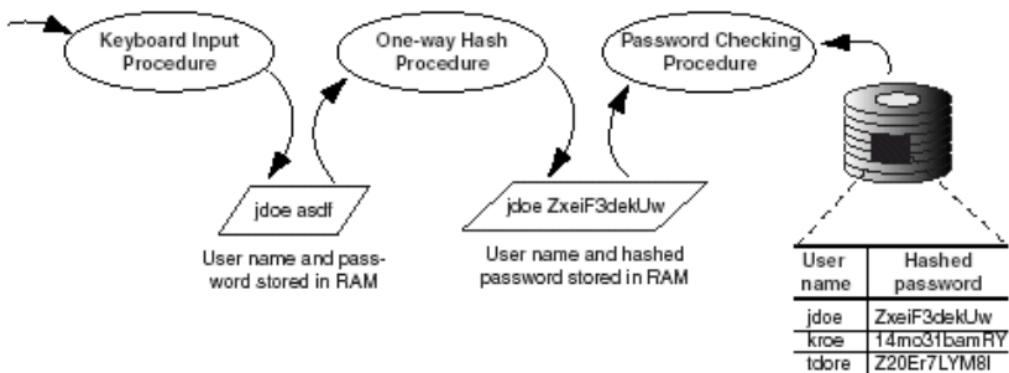
## Esempio: SHA1



## CTSS + Hashing

Negli anni '60, al MIT, il Compatible Time Sharing System (CTSS) è stato introdotto. Le password erano memorizzate in chiaro su file di sistema, ma erano protette da politiche di sicurezza. Si trattava di un'idea circolare: il controllo dell'accesso era basato sull'autenticazione, e viceversa. Tuttavia, si registrarono numerosi attacchi.

Per affrontare questa vulnerabilità, venne proposto di potenziare CTSS con una funzione di hash, come sviluppato presso l'Università di Cambridge nel 1967. In questo sistema, **il file delle password memorizzava l'hash di ciascuna password**, migliorando così la sicurezza complessiva del sistema.



## Salting

Il salt è un messaggio casuale, inizialmente di 12 bit ma oggi considerato insufficiente, che viene aggiunto a una password per proteggerla dagli attacchi dizionario prima di applicare l'hash.

Unix salvava le password in memoria generando un salt per ciascuna password da memorizzare. Utilizzava la password per codificare una stringa di zeri insieme al salt tramite la funzione di encryption crypt(3) basata su DES.

L'uso di una funzione di encryption per generare una funzione hash, come crypt(3), è un esempio storico. Criptava una stringa di zeri estesa con 12 bit di salt randomico, funzionando come una discreta funzione hash per il suo tempo.

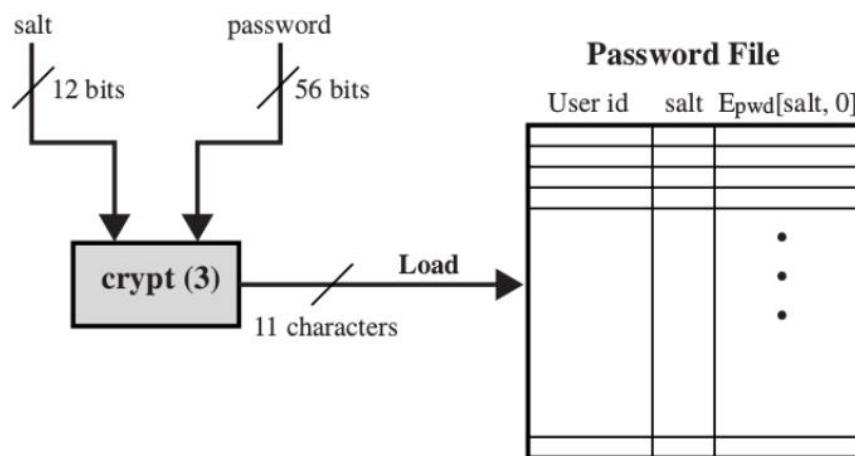
Questo approccio sfrutta in modo astuto un algoritmo di encryption che soddisfa le caratteristiche di una funzione hash.

Il salt complica gli attacchi, mentre il pepper, sebbene fisso, allunga l'input e aumenta la complessità dell'hash, rendendo più difficile la pre-elaborazione degli input possibili.

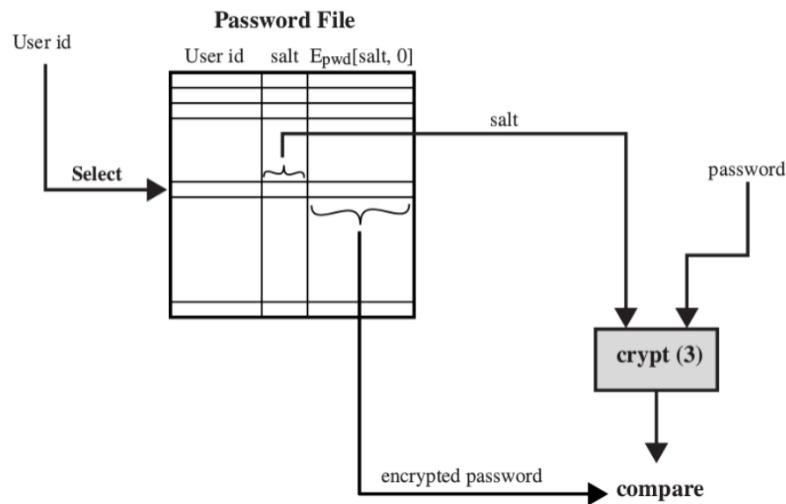
## UNIX: Aggiunta di una password

Si ottengono 56 bit utilizzando i 7 bit meno significativi dei primi 8 caratteri della password. Questi 56 bit vengono utilizzati come chiave per criptare il salt e una stringa di zeri. La password non è più memorizzata come testo in chiaro, ma viene utilizzato il segreto da proteggere come chiave anziché come testo in chiaro.

Non viene utilizzato l'hashing.



## UNIX: Verifica di una password



Una misura che fornisce segretezza ed autenticazione è l'hash crittografico, come crypt3, che considera solo i primi 8 caratteri della password (è consigliabile utilizzare password più lunghe). Questo metodo è definito tramite una primitiva crittografica.

Le tabelle rainbow sono una pre-tabulazione degli hash ottimizzata in termini di spazio, utilizzata per invertire gli hash.

Durante la verifica di una password, il programma di autenticazione cerca nella tabella il salt associato all'utente e, insieme alla password, calcola l'hash crittografico. Se il risultato corrisponde a quanto registrato nella tabella, l'utente viene autenticato. Tuttavia, finora non abbiamo menzionato nulla riguardo all'integrità.

### Tabella Rainbow

Le tabelle rainbow sono un metodo di ottimizzazione per gli attacchi di forza bruta contro le funzioni di hash, mentre l'attacco per invertire l'hash sfrutta queste tabelle precalcolate per trovare corrispondenze tra hash target e valori originali.

## Freshness

Attributo, caratterizzazione, di almeno due proprietà fondamentali della sicurezza: confidenzialità ed autenticazione.

Più è recente un segreto maggiore sarà l'affidabilità, importante quindi cambiare spesso la password.

La freshness dell'autenticazione è fondamentale.

### **Autenticazione bancaria:**

1. John si autentica all'accesso
2. John effettua una operazione finanziariamente sensibile (operazione dispositivo) tramite 2FA (Normativa PSD2) ulteriore misura di sicurezza
3. La banca onora la richiesta 2

### **Esempio:**

1. John si autentica alle 10:00
2. John effettua una richiesta di bonifico alle 10:15, la banca dovrebbe concedere questa operazione?

Come facciamo a capire che la richiesta di bonifico provenga dallo stesso client che si è autenticato alle 10:00?

### **Cosa fa la banca alla richiesta di autenticazione dell'utente?**

- 1a) Controllo dell'identità di John
- 1b) Se il controllo va a buon fine si crea un nuovo ID di sessione
- 1c) Inserimento dell'ID di sessione all'interno dei cookie di sessione
- 1d) Invio del cookie al client
- 1e) Memorizzazione del cookie di sessione

Ogni qualvolta si voglia effettuare un'operazione dispositivo si invia il **session cookie**, viene verificato ed autorizzato il movimento.

Session cookie fondamentale dal punto di vista della segretezza, esiste un protocollo (TLS) che gestisce l'autenticazione ed i cookie di sessione. Chiunque sia in possesso della session ID può operare a nome di chi si è autenticato.

Perché la session ID? Perché altrimenti l'utente dovrebbe autenticarsi ad ogni operazione effettuata.

**Nonce: Number Only Once**, numero random utilizzato a livello HTTP che viene generato solo una volta, la session ID è un nonce.

### Che caratteristiche deve avere il Session ID?

La session ID è fondamentale per la sicurezza, poiché permette agli utenti di eseguire operazioni senza doversi autenticare ad ogni passaggio. Deve essere unico nel tempo e sufficientemente grande da evitare attacchi brute force.

Un attacco che sfrutta la mancanza di freshness è il **replay attack**, in cui un attaccante registra e riproduce una sequenza di azioni per ottenere un vantaggio illegittimo.

# Parte 5: Protocolli di sicurezza basilari

## Protocolli basilari per la freshness

### **Timestamp**

Chi desidera garantire la freschezza dei dati inserisce un timestamp, associandolo in modo affidabile al messaggio target. Tuttavia, c'è il rischio che possa falsificarlo. È fondamentale che gli orologi distribuiti siano sincronizzati, altrimenti chiunque potrebbe attaccare il sistema.

### **Nonce**

Chi desidera ricevere una garanzia di freschezza pubblica una nonce e attende traffico che la menzioni. Tale traffico sarà sicuramente successivo all'istante di creazione della nonce, anche se potrebbe riferirsi a materiale più vecchio. In questo caso, non è necessario che gli orologi siano sincronizzati.

## Dalla crittografia alla sicurezza

Per ottenere le tre proprietà fondamentali della sicurezza - segretezza, autenticazione ed integrità - i protocolli di sicurezza utilizzano modelli crittografici come il modello Dolev-Yao, che analizzano i casi peggiori di attacco, assumendo che la crittografia funzioni.

Per quanto riguarda l'autenticazione e l'integrità, si utilizza il timestamp, un marcitore temporale che è utile per verificare la freshness, calcolando la differenza tra l'orario corrente e l'orario di generazione del timestamp. Tuttavia, è fondamentale considerare che il timestamp diventa critico e sensibile se alterato nel traffico o falsificato dal mittente. Per garantire l'affidabilità del timestamp, è necessario un orologio sincronizzato, ad esempio tramite il Network Timing Protocol (NTP), che assicura la sincronizzazione dell'orologio.

# Sintassi dei messaggi crittografici

## Atomici

- Nomi di agenti:  $A, B, C, \dots$
- Chiavi crittografiche:  
 $k_a, k_b, \dots, k_a^{-1}, k_b^{-1}, \dots$   
 $k_{ab}, k_{ac}, \dots$
- Nonce:  $N_a, N_b, \dots$
- Timestamp:  $T_a, T_b, \dots$
- Digest:  $h(m), h(n), \dots$
- Etichette: "Trasferisci £100 dal conto di..."

## Composti

- Concatenazioni:  $m, m'$ 
  - ciascuno può essere crittotesto
- Crittotesti:  $m_k$ 
  - il testo in chiaro può essere concatenazione

## Problema

Si può autenticare un messaggio concatenato?

**Come autenticare un messaggio concatenato?** Posso io ricevente avere una garanzia di autenticazione sul messaggio, posso derivare la proprietà che il messaggio sia autentico? NO

## Esempio di messaggio concatenato:

Alice riceve un messaggio concatenato  $(m,n)$  che non è possibile autenticare, perchè nell'invio del messaggio tra Alice e Bob l'attaccante Charlie potrebbe modificare  $n$  con  $n'$ .

Alice ---  $(m,n)$  ---> Bob

**Attacco:** Alice ---  $(m,n)$  --- Charlie ---  $(m,n')$  ---> Bob

# Sintassi di un protocollo di sicurezza

L'etichetta del mittente diventa irrilevante, es: IP mittente nel pacchetto può essere modificato per effettuare un attacco, non è affidabile e non garantisce una misura di autenticazione. A garantire l'autenticazione è il traffico stesso tramite crittografia, sappiamo che la crittografia garantisce autenticazione.

1.  $A \rightarrow B : A, N_a$
2.  $B \rightarrow A : N_a k_b^{-1}$

## Protocolli basilari per la segretezza

### Crittografia simmetrica

Supponiamo che esista una chiave di sessione condivisa tra A e B e solo da loro. Il messaggio è confidenziale solo in virtù di questa condizione. A invierà a B un messaggio cifrato con la chiave  $K_{ab}$  a B.

$$A \longrightarrow B : m_{k_{ab}}$$

### Crittografia asimmetrica

- B deve avere una chiave privata valida e certificata, sicura e non scaduta
- A deve verificare le chiavi pubbliche  $K_b$  sia di B

$$A \longrightarrow B : m_{k_b}$$

A prende la chiave pubblica del destinatario B per decifrarlo serve la chiave privata di B

## Protocolli basilari per l'autenticazione

### Crittografia simmetrica

- Chiave  $K_{ab}$  condivisa solo tra A e B
- B deve verificare l'assunzione precedente

$$A \longrightarrow B : "Sono io!"_{k_{ab}}$$

Nell'assunzione 1 non è specificato chi deve sapere il contenuto, nell'assunzione 2 si specifica che deve essere solo B a sapere il contenuto, l'assunzione 2 permette a B di rilevare il mittente del messaggio e quindi di autenticarlo.

### Crittografia asimmetrica

- A deve avere una chiave privata valida
- B deve verificare che la chiave pubblica  $K_a$  sia di A

$$A \rightarrow B : "Sono io!"_{k_a^{-1}}$$

Serve anche questa volta certificazione, chi decifra decide che il mittente è il proprietario della chiave pubblica. Obiettivo è l'autenticazione, non mi interessa il contenuto del messaggio, posso scrivere ciò che voglio.

## Combinare segretezza ed autenticazione

### Crittografia simmetrica

- Chiave  $K_{ab}$  condivisa solo tra A e B
- B deve verificare l'assunzione precedente

$$A \rightarrow B : m_{k_{ab}}$$

### Crittografia asimmetrica

- B deve avere una chiave privata valida e certificata, sicura e non scaduta
- A deve verificare le chiave pubblica  $K_b$  sia di B
- A deve avere una chiave privata valida e certificata, sicura e non scaduta
- B deve verificare che la chiave pubblica  $K_a$  sia di A

$$1. A \rightarrow B : \{m_{k_a^{-1}}\}_{k_b} \text{ o } 1. A \rightarrow B : \{m_{k_b}\}_{k_a^{-1}}$$

# Come ottenere l'integrità

Qualunque messaggio sulla rete potrebbe essere alterato, in particolare anche un messaggio protetto per segretezza e autenticazione.

L'integrità è sempre a rischio, utilizziamo le funzioni hash con la caratteristica che l'entropia dell'input sia magnificata nell'entropia dell'output.

## **Utilizzo del checksum sui download**

Quando scarichiamo un file e ci viene fornito l'hash bisogna verificarlo per ottenere la proprietà di integrità. Questo protocollo ha delle vulnerabilità l'attaccante potrebbe cambiare  $m$  in  $m'$ , ricalcolare l'hash di  $m'$  ed inviare il tutto al destinatario.

Quando scarichiamo un file ( $m, h(m)$ ) Alice invia a Bob **soltamente  $m$** .

$h(m)$  è un semplice testo visualizzato a video, se l'attaccante cambia  $m$  in  $m'$  deve anche violare il sito modificando  $h(m)$  in  $h(m')$  per far credere a Bob di aver scaricato il file corretto.

## **Come impedire che l'attaccante riesca a cambiare $h(m)$ in $h(m')$ ?**

Possiamo cifrare l'hash con la chiave privata del mittente. Questo protocollo prende il nome di **firma digitale**.

$$A \longrightarrow B : m, h(m)_{k_a^{-1}}$$

Ricevuto il messaggio, **B**:

- Applica la funzione hash alla prima componente
- Decodifica la seconda componente
- Confronta i due risultati
- Ottiene garanzia di integrità se e solo se essi combaciano

## **Come è possibile ottenere certificazione con la cifratura simmetrica?**

A  $\rightarrow$  B:  $m, h(m, K_{ab})$  protocollo MAC Message Authentication Code, si aggiunge autenticazione all'hash. Inserire dentro l'hash funziona come cifrare l'hash con la chiave.

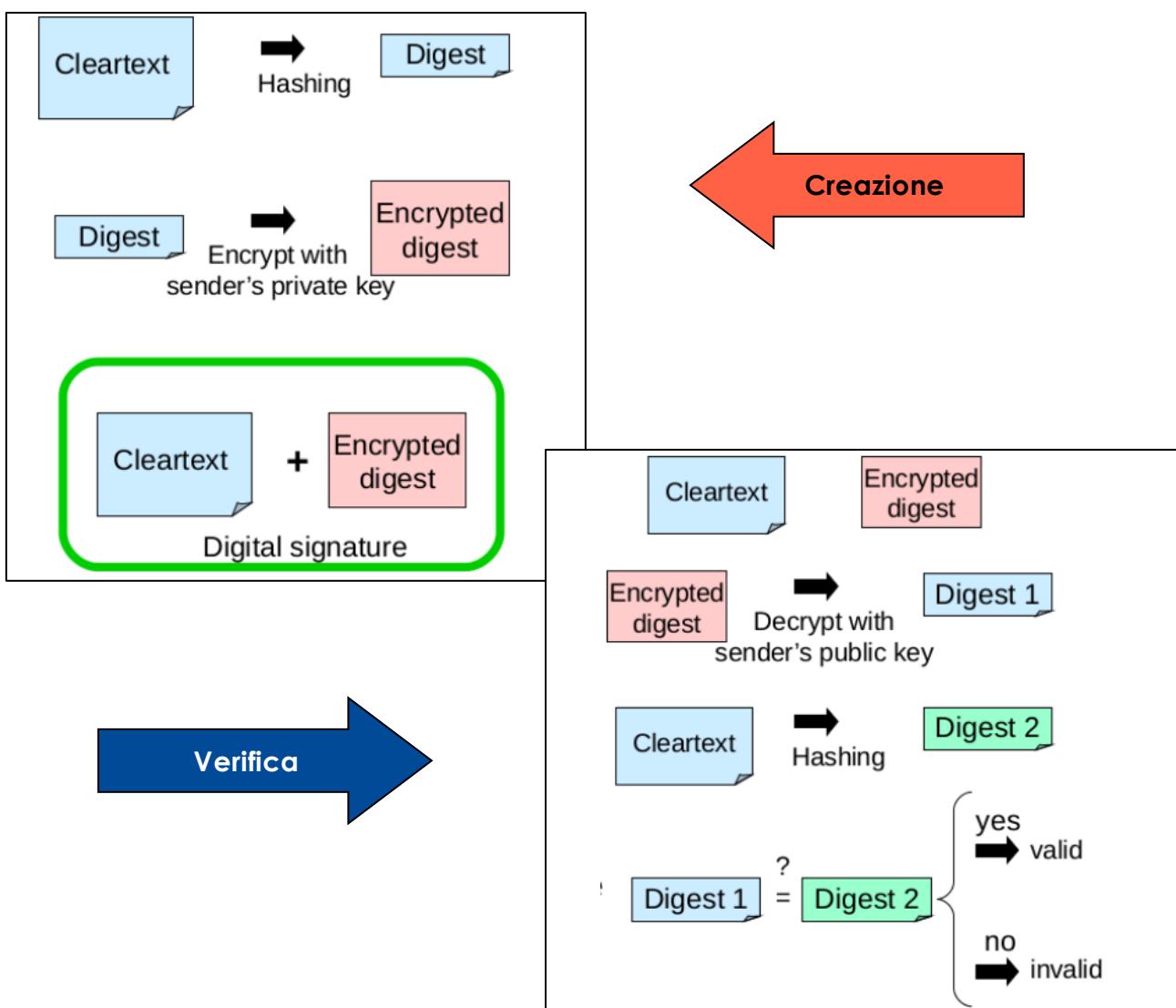
# Firma digitale

Assicura sia l'integrità che l'autenticazione. Non esiste una soluzione che garantisca l'integrità senza l'autenticazione. Sfruttiamo anche la proprietà di autenticazione, cifrando con la chiave privata del mittente.

Effettuare una firma digitale significa utilizzare un paio di algoritmi, uno per **creare la firma** e uno per **verificarla**. Si invia il documento in chiaro insieme alla cifratura del digest. Questo processo combina autenticazione e integrità, ma non confidenzialità e integrità.

Nessuno può firmare per conto di un altro, ma tutti possono verificare la firma digitale, poiché tutti conoscono la chiave pubblica del mittente.

## Creare e verificare una firma



## Protocolli basilari per l'integrità

### Crittografia simmetrica

- Omesso

### Crittografia asimmetrica

- A deve avere una chiave privata valida e certificata, sicura e non scaduta
- B deve verificare che la chiave pubblica  $K_a$  sia di A

$$A \longrightarrow B : sign_A(m)$$

$$sign_A(m) = m, h(m)_{k_a^{-1}}$$

Stessi prerequisiti dell'autenticazione di A con B

## Combinare tutte le proprietà di livello 1

- Segretezza del messaggio m per A e B,
- autenticazione di A con B,
- integrità di m nella trasmissione da A a B

### Crittografia simmetrica

- Omesso

### Crittografia asimmetrica

- B deve avere una chiave privata valida e certificata, sicura e non scaduta
- A deve verificare che la chiave pubblica  $K_b$  sia di B
- A deve avere una chiave privata valida e certificata, sicura e non scaduta
- B deve verificare che la chiave pubblica  $K_a$  sia di A

$$A \longrightarrow B : sign_A(m_{k_b})$$

La firma digitale risolve il problema delle falsificazioni. La firma tradizionale offre solo un livello di autenticazione ed è facilmente falsificabile.

Per quanto riguarda la segretezza del messaggio ( $m$ ) tra A e B, l'autenticazione di A con B e l'integrità di  $m$  durante la trasmissione da A a B, è necessario considerare che tutti i protocolli basilari presentati sono vulnerabili agli attacchi di ripetizione e non incorporano meccanismi di freschezza.

Prima di esaminare protocolli più avanzati, manteniamo due promesse fondamentali:

- Utilizzo della crittografia **simmetrica**: condivisione di un **segreto** iniziale.
- Utilizzo della crittografia **asimmetrica**: **certificazione**.

Per ottenere ciò, verranno impiegati protocolli dedicati:

- Per la crittografia **simmetrica**: **Diffie-Hellmann** (DH) o **RSA** Key Exchange.
- Per la crittografia **asimmetrica**: **Public-Key Infrastructure** (PKI).

## Protocollo Diffie-Hellmann

1. Alice e Bob concordano due parametri pubblici molto grandi e correlati tra loro:  $\alpha$  (alpha) e  $\beta$  (beta)
2. Alice genera un numero casuale privato,  $X_a$ 
  - a. Successivamente calcola  $Y_a = \alpha^{X_a} \text{ mod } \beta$  (pubblico)
3. Bob genera un numero casuale privato,  $X_b$ 
  - a. Successivamente calcola  $Y_b = \alpha^{X_b} \text{ mod } \beta$  (pubblico)
4. Alice e Bob eseguono lo scambio delle chiavi  $Y_a$  e  $Y_b$
5. Bob alla ricezione di  $Y_a$  calcola  $Y_a^{X_b} \text{ mod } \beta$
6. Alice alla ricezione di  $Y_b$  calcola  $Y_b^{X_a} \text{ mod } \beta$

Si nota che  $Y_b^{X_a} \text{ mod } \beta = Y_a^{X_b} \text{ mod } \beta$  sono quindi lo stesso numero che rappresentano quindi la chiave di sessione tra Alice e Bob che chiameremo con  $K_{ab}$ .

Le Y sono pubbliche mentre le X sono private.

### **Questo protocollo è legato alla crittografia asimmetrica?**

No, ma la crittografia asimmetrica si ispira ad esso.

**Questo schema presenta delle vulnerabilità? Esiste un'operazione che consenta a chiunque intercetti le Y di derivare le X?**

L'attaccante può intercettare  $Y_a$  e  $Y_b$ , per calcolare  $K_{ab}$  basterebbe calcolare  $X_a$  e  $X_b$ .

Per calcolare l'esponente, sarebbe necessario applicare l'operazione:  $\log_\alpha(\alpha^{X_a})$  il che è uguale a  $X_a$ .

Tuttavia, essendo applicata anche l'operazione modulo, bisognerebbe effettuare l'operazione di **logaritmo discreto**, che risulta essere **intrattabile** e onerosa **per numeri alpha e beta molto grandi**.

In pratica, dato  $Y_a$  non è possibile ricavare  $X_a$  in tempi ragionevoli.

### **Non vi è autenticazione tra Alice e Bob**

Sebbene la confidenzialità sia assicurata, l'assenza di autenticazione non garantisce la confidenzialità, poiché non possiamo essere certi di comunicare con Bob ma potremmo trovarci a comunicare con Charlie. Questa mancanza apre le porte ad un attacco Man In The Middle.

## Protocollo Diffie-Hellmann – Attacco

L'attaccante Charlie ha una sua coppia di chiavi  $X_c$  e  $Y_c$

1. Alice invia a Bob  $Y_a$
2. Charlie intercetta  $Y_a$  ed invia a Bob  $Y_c$
3. Bob invia ad Alice  $Y_b$
4. Charlie intercetta  $Y_b$  ed invia ad Alice  $Y_c$
5. Bob alla ricezione di  $Y_c$  calcola  $Y_c^{X_b} \text{ mod } \beta$
6. Alice alla ricezione di  $Y_c$  calcola  $Y_c^{X_a} \text{ mod } \beta$

Da questo ne deriva che:

- Alice conosce  $K_1 = Y_c^{X_a} \text{ mod } \beta$
- Bob conosce  $K_2 = Y_c^{X_b} \text{ mod } \beta$
- La chiave di sessione fallisce poiché  $K_1 \neq K_2$

Charlie, conosce invece sia  $K_1$  che  $K_2$  che calcola come:

- $K_1 = Y_a^{X_c} \text{ mod } \beta$
- $K_2 = Y_b^{X_c} \text{ mod } \beta$

## Diffie-Hellmann – Attacco al microscopio

Nei primi 4 punti dell'attacco Charlie, per via della mancanza di autenticazione fa un duplice attacco di autenticazione costituito dai passi 2 e 4.

Nel calcolo di  $K_1$  e  $K_2$ , l'attaccante Charlie fa un duplice attacco di segretezza utilizzando indebitamente  $Y_a$  e  $Y_b$  in quanto questa versione non prevede alcuna tecnica di segretezza

Se l'attaccante avesse semplicemente inviato le sue proprie  $Y_c$  senza conoscere  $Y_a$  e  $Y_b$ , non sarebbe stato in grado di completare l'attacco e di calcolare  $K_1$  e  $K_2$ .

## Come irrobustire Diffie-Hellmann

Il primo ricorso ufficiale alla crittografia asimmetrica è una delle mosse per rafforzare Diffie-Hellman. Una delle due versioni di Diffie-Hellman è utilizzata in IPsec. L'approccio congiunto tra crittografia asimmetrica e simmetrica, che è più veloce ed efficiente, è un'altra strategia di irrobustimento.

Per **prevenire l'attacco all'autenticazione**, si può utilizzare la codifica con la chiave privata del mittente. Anche se l'attaccante riuscisse a intercettare  $Y_a$  e  $Y_b$ , non sarebbe in grado di fornire ai destinatari la propria  $Y_c$ , poiché dovrebbe cifrarla con la chiave del mittente, che l'attaccante non possiede.

1.  $A \rightarrow B : \{Y_a\}_{k_a^{-1}}$
2.  $B \rightarrow A : \{Y_b\}_{k_b^{-1}}$

Per **proteggere contro l'attacco alla confidenzialità**, si può adottare la cifratura con la chiave pubblica del destinatario. In questo modo, anche se l'attaccante intercetta i valori  $Y_a$  e  $Y_b$ , non sarà in grado di decifrarli senza la chiave privata del destinatario.

1.  $A \rightarrow B : \{Y_a\}_{k_b}$
2.  $B \rightarrow A : \{Y_b\}_{k_a}$

## RSA Key Exchange

Alternativa a Diffie-Hellmann basata su crittografia asimmetrica. Nel mondo asimmetrico esiste ancora il problema della certificazione. Il mittente inventa la chiave di sessione randomicamente e la manda in una busta digitale. Anch'esso manca di autenticazione ma L'attaccante non può calcolare la chiave di sessione. Affinché il protocollo funzioni, A deve innanzitutto procurarsi il certificato (della chiave pubblica) di B.

$$1. \quad A \rightarrow B : \{k_{ab}\}_{k_b}$$

## P.I.I.: Personal Identifiable Information

Il nome e il cognome non identificano univocamente una persona; sono necessarie ulteriori informazioni, come il codice fiscale. Nella carta di identità, un elemento che attesta l'autenticità è il numero di carta di identità, che autentica anche il produttore della carta di identità.

Gli elementi fondamentali in una carta di identità sono almeno tre:

- Anagrafica
- Fotografia
- Produttore

Queste tre informazioni devono essere prodotte per garantire l'integrità e per essere riconoscibili e non sostituibili.

A e  $K_a$  rappresentano l'anagrafica e la chiave privata di A, mentre CA (Autorità di Certificazione) è l'ente che emette il certificato.

Il compito della CA è quello di mettere insieme, tramite cifratura,  $\{A, K_a\}$ , cifrati con la chiave inversa di CA ( $K_{CA}^{-1}$ ). Questo avviene attraverso una firma digitale primitiva crittografica, che rappresenta una misura che combina autenticazione ed integrità.

Esiste una catena di certificati e, di conseguenza, una catena di atti di fiducia che culmina con la root, che nel caso della carta di identità è il Ministero degli Interni.

I certificati vengono firmati sempre dall'ente che si trova più in alto nella catena, fino ad arrivare al livello 0, in cui il certificato viene firmato dallo stesso ente che lo produce.

**RCA:** Root Certification Authority. Sul certificato radice, facciamo un atto di fede.

## Certificazione: Crittografia asimmetrica

Un certificato è paragonabile a una carta d'identità e possiede caratteristiche di grande valore, tra cui la fiducia. Un'autorità di certificazione vende fiducia, e dietro le CA ci sono spesso consorzi di banche. Il valore di un certificato deriva dall'autorevolezza di chi lo emette e dalla libertà di chi lo riceve nel decidere se accettarlo o meno. L'origine dell'albero delle certificazioni è autocertificante, quindi rappresenta un atto di fede. I fornitori di identità digitale sono regolamentati dalla legge italiana.

Gli URL hanno bisogno di questi certificati, e ci sono problematiche relative alla certificazione dei wildcard per coprire anche i sottodomini. Ad esempio,

### **Se unict.it è certificato, questo implica che anche dmi.unict.it lo sia?**

Non necessariamente, poiché non esiste un'implicazione logica diretta.

TLS è un protocollo crittografico che utilizza una combinazione di cifratura simmetrica e asimmetrica, e richiede la certificazione, ma la radice della certificazione richiede un atto di fede.

### **Come fa il browser a sapere chi è l'autorità di certificazione radice?**

Ogni browser possiede un database di certificati TLS. Se il browser possiede la root, può certificare l'intera catena di certificati. Ogni browser fornisce un insieme diverso di certificati di root, che possono variare.

La **certificazione** serve ad **associare una chiave pubblica al suo legittimo proprietario**, ovvero colui che possiede la sua metà privata. L'associazione è realizzata da un **certificato digitale** firmato digitalmente da una autorità di certificazione (CA).

Un valido certificato si compone di almeno tre elementi essenziali:

- Due elementi congiunti
- Un terzo elemento che identifica l'autorità di certificazione (la quale deve essere autenticata tramite firma digitale).

La certificazione è intrinsecamente una catena di fiducia che si estende attraverso un numero potenzialmente infinito di passaggi; in ultima analisi, giungiamo a una **radice autocertificata**. Limitare il numero di autorità di radice, rendendole invalidabili e non espandibili, costituisce una precauzione essenziale che rinforza l'intera catena di fiducia.

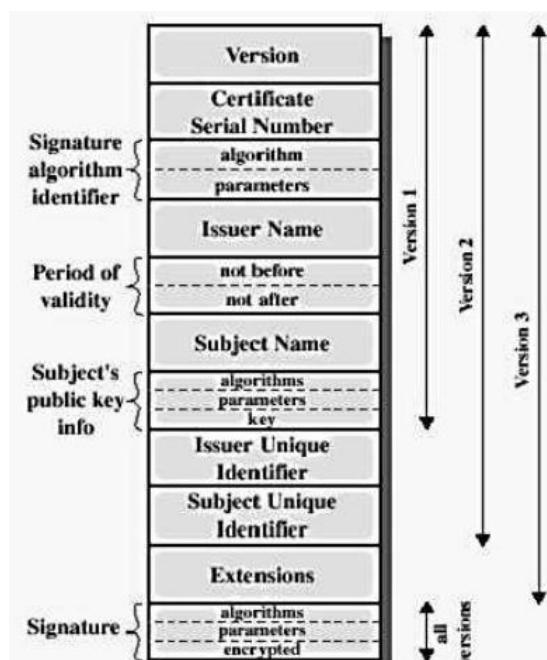
Le radici dei certificati risiedono nel nostro browser; solo le catene di certificati che si collegano a una delle radici memorizzate nel browser saranno considerate valide. L'obiettivo primario è autenticare un URL, e per farlo si utilizza TLS, un protocollo crittografico a chiave pubblica. Questo richiede un certificato valido.

L'URL dovrebbe essere facilmente memorizzabile e indicativo; l'autenticazione richiede una dichiarazione di identità che consente all'utente di verificare l'affermazione fatta. Senza autenticazione, i contenuti di una pagina web non possono essere considerati affidabili.

### **Ma dove risiede la misura dell'autenticazione?**

I browser autenticano un sito tramite il simbolo del lucchetto verde, e l'utente deve fidarsi della genuinità del browser. La conferma dell'autenticazione viene comunicata dall'utente al browser attraverso il lucchetto chiuso, affermando che l'URL corrisponde a ciò che dichiara e che il sito è autentico. Tuttavia, l'autenticazione non garantisce la qualità; alcuni siti potrebbero essere compromessi da vulnerabilità, rendendoli potenzialmente pericolosi.

## Certificato X.509



Il formato ufficiale dei certificati X.509 comprende:

- Versione
- Numero di certificato
- Identificativo della firma
- Nome del fornitore del certificato
- Periodo di validità (con possibilità di revoca in caso di compromissione della chiave privata o altri eventi, con una relativa fiducia nel certificato)
- Nome del soggetto (A)
- Informazioni pubbliche ( $K_a$ )
- Identificativo dell'emittente (CA)
- Identificativo del soggetto (A)
- Estensioni
- Firma (che può comprendere sia la coppia di chiavi che solo la seconda componente, ovvero la cifratura del digest della parte chiave)

La firma, essenziale per la verifica dell'autenticità del certificato, può includere sia la coppia di chiavi che solo la cifratura del digest della parte chiave.

## Public Key Infrastructure

Una infrastruttura a chiave pubblica o sistema di certificazione consiste nella gerarchia di autorità e nelle tecnologie che esse usano. La Public-Key Infrastructure (PKI) è una struttura ad albero composta da certificazioni.

Ad esempio, il rettore certifica le firme dei direttori di dipartimento, i direttori di dipartimento a loro volta certificano i propri docenti.

Quando si deve certificare una "foglia" (ossia un'entità finale, come un singolo docente), non è necessario certificare l'intero albero, ma solo il percorso a ritroso dalla foglia fino alla radice. Questo processo richiede uno sforzo con complessità logaritmica.

Per comunicare con Alice, a Bob serve il certificato di Alice.

Schematizzando, esso ha forma:  $\{\dots A \dots K_a \dots\} K_{ca}^{-1}$

Quindi A ha bisogno del certificato della CA.

A sua volta, questo `e firmato da un'autorità superiore, ed ha forma:

$$\{\dots CA \dots K_{ca} \dots\} K_{ca(n-1)}^{-1}$$

Pertanto, ciascuna autorità è certificata da una superiore, fino alla root certification authority (RCA) o primary certification authority.

## Chain Of Trust

La catena di fiducia di A è la lista completa dei certificati (fino a quello di root) che permettono, insieme alla chiave pubblica della RCA, di verificare il certificato di A. Per comunicare con Alice, Bob deve risolvere la catena di fiducia.

Sia  $CA(n)$  l'autorità foglia; sia  $RCA = CA(0)$

- $\{\dots A \dots k_a \dots\}_{k_{ca(n)}^{-1}}$
- $\{\dots CA(n) \dots k_{ca(n)} \dots\}_{k_{ca(n-1)}^{-1}}$
- $\vdots$
- $\{\dots CA(1) \dots k_{ca(1)} \dots\}_{k_{rca}^{-1}}$
- $\{\dots RCA \dots k_{rca} \dots\}_{k_{rca}^{-1}}$

Il certificato di root (RCA) è l'unica entità autocertificata, e lo store dei certificati di root è vulnerabile e suscettibile agli attacchi.

Se un aggressore riesce maliziosamente ad inserire un certificato di root:

$$\{bad_{RCA}, K_{bad_{RCA}}\}K_{bad_{RCA}}^{-1}$$

Questo equivale a convertire la fiducia in un attacco diretto.

A sua volta,  $bad_{RCA}$  potrebbe emettere certificati per gli URL di sua proprietà, generando una nuova coppia di chiavi e firmando con la chiave di root. Fare un attacco del genere era relativamente semplice qualche anno fa, poiché l'importazione non richiedeva autorizzazioni di amministratore. Oggi, il metodo di alzare i privilegi rende l'attacco più complesso, ma rimangono possibili vulnerabilità pratiche (bug) che potrebbero consentire l'attacco.

La trust base del browser è estensibile per scopi funzionali, quindi è possibile aggiungere certificati.

**Qual è la differenza tra un certificato di root di Verisign e uno homemade?**  
Sintatticamente sono identici.

**Come vengono gestiti i sottodomini?**

Usando i "wildcards", il fornitore offre un numero limitato di certificati che coprono tutti i suoi sottodomini. La perdita di fiducia si propaga verso il basso lungo la catena.

## Segretezza VS Fiducia

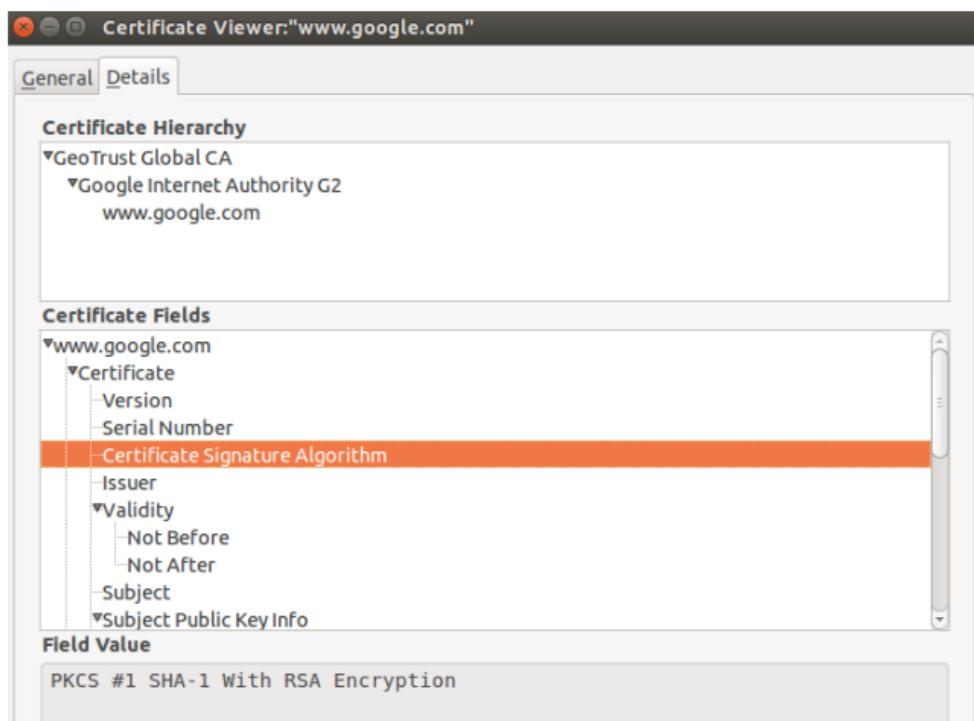
Se l'attaccante si impossessasse della chiave privata di una certa CA:

- Non si impossesserebbe di alcun'altra chiave privata automaticamente
- Automaticamente tutti perderebbero fiducia in qualunque chiave pubblica la cui certificazione necessiti della chiave pubblica della CA attaccata. L'attaccante potrebbe creare delle false certificazioni aventi chiavi pubbliche con certificati falsi.

La perdita di segretezza non si propaga; la perdita di trust si propaga verso i livelli inferiori.

Se ipotizziamo che un certificato sia compromesso, ad esempio se un'autorità di certificazione perde la propria chiave privata, ciò non implica automaticamente che lo stesso problema si propaghi ai suoi sottoalberi, ovvero ai certificati emessi da essa.

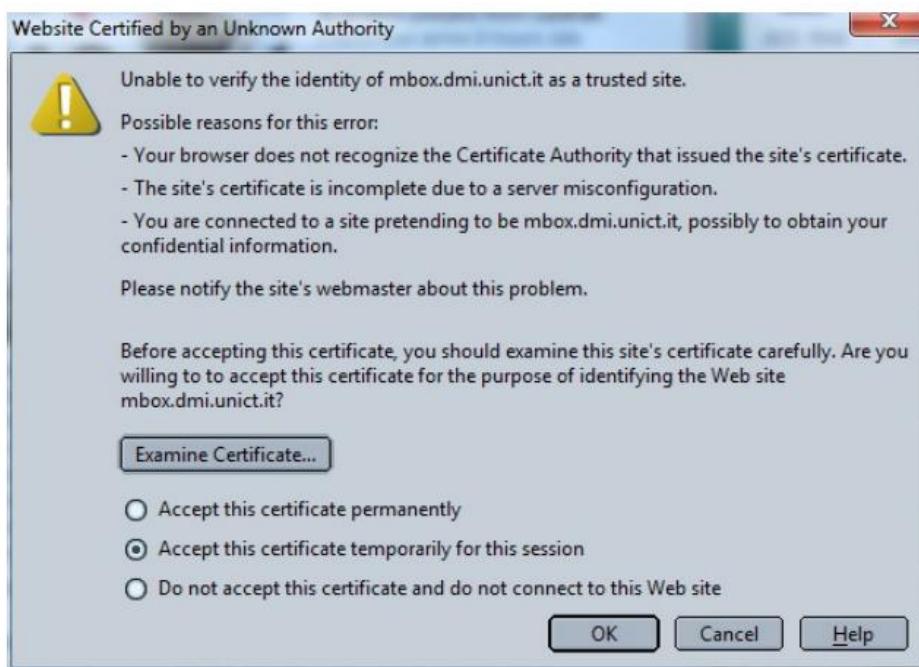
I problemi di sicurezza, come la perdita della chiave privata, non si estendono oltre la propria entità; tuttavia, i problemi di fiducia si propagano. Questo significa che, se un'autorità di certificazione perde la fiducia a causa di una compromissione, anche i certificati emessi da essa rischiano di non essere più considerati affidabili, causando una diffusione del problema lungo la catena di fiducia.



## Problema: Certificati Self-Issued

Quando il browser non riesce a verificare l'identità di un sito web, l'utente ha tre opzioni:

- Non accettare il certificato e non connettersi al sito web.
- Accettare temporaneamente il certificato, rendendolo valido solo per la sessione corrente.
- Accettare il certificato in modo permanente.



Se si accetta il certificato temporaneamente, si espone a numerose possibilità di attacchi di "man-in-the-middle". Ad esempio, ogni volta che si tenta di effettuare l'accesso, le richieste possono essere intercettate sulla rete locale e reindirizzate verso un sito fasullo non certificato. Questo crea un rischio concreto di spoofing.

Accettare il certificato in modo permanente, d'altra parte, equivale a fare un atto di fede nei confronti del sito web, poiché si assume che tutte le future connessioni saranno sicure.

In termini di impatto, entrambe le opzioni sono equivalenti, ma la verosimiglianza di un potenziale rischio è differente.

## Questo scenario è sempre meno comune per diverse ragioni:

- **Costi ridotti della certificazione:** Grazie a servizi come Let's Encrypt, che offre certificati gratuiti, e alla diminuzione dei costi complessivi della certificazione, ottenere un certificato è diventato più accessibile economicamente.
- **Facilità di ottenimento dei certificati:** Let's Encrypt, supportato da un consorzio di autorità di certificazione, ha semplificato notevolmente il processo di ottenimento dei certificati, rendendolo disponibile per tutti i tipi di siti web, compresi i sottodomini.
- **Tipologie di certificati:** Anche se Let's Encrypt offre principalmente certificati di tipo Domain Validated (DV), che richiedono solo una validazione del dominio, ci sono altre tipologie di certificati come quelli di tipo Extended Validation (EV) e Organization Validated (OV) che offrono livelli di validazione più elevati, specialmente per siti sensibili come le banche.
- **Differenze visive nei browser:** Alcuni browser forniscono indicazioni visive diverse per i diversi tipi di certificati, come le barre degli indirizzi colorate o i segni di sicurezza, che aiutano gli utenti a identificare i siti affidabili.
- **Certificate Transparency:** Questa tecnologia aiuta a garantire che i certificati siano autentici e rende più difficile per gli aggressori utilizzare certificati fraudolenti.
- **Rischi associati ai cambi di certificato:** Il cambio repentino di certificato può essere considerato sospetto e potrebbe essere un segno di un possibile attacco in corso.

In generale, l'aumentata disponibilità e accessibilità dei certificati, insieme a migliori pratiche di sicurezza e tecnologie aggiuntive come Certificate Transparency, hanno contribuito a rendere il web più sicuro e affidabile per gli utenti.

## Lista di Revoca dei Certificati (CRL)

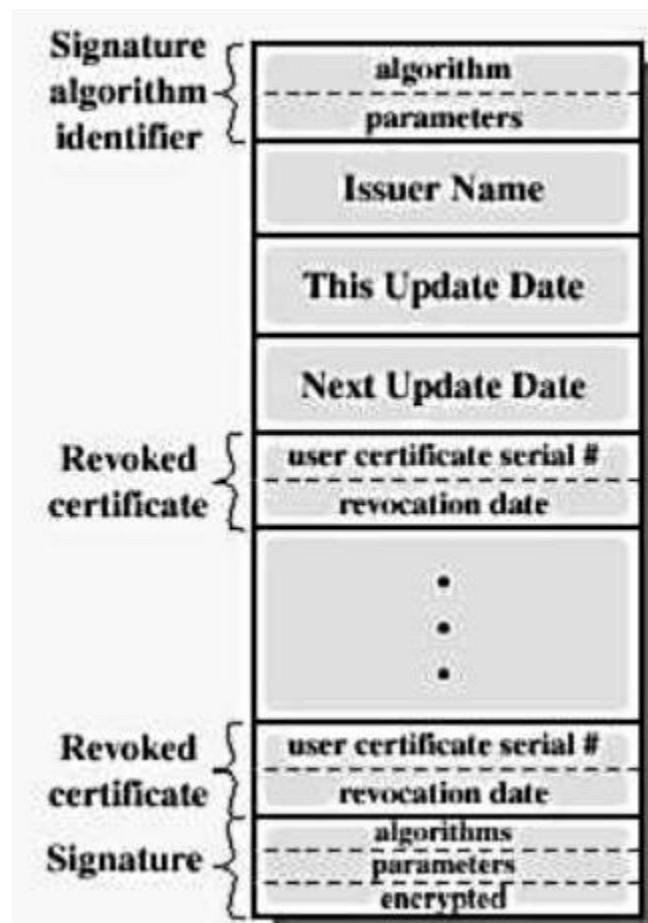
La necessità di revocare un certificato può sorgere in vari contesti, come la perdita della chiave privata o il cambio del Subject Identifier.

Revocare un certificato è un'operazione altrettanto sensibile quanto rilasciarlo. Richiede la stessa autorità e solo l'entità che ha emesso il certificato può revocarlo.

Una Certificate Revocation List (CRL) è una lista di certificati revocati. Essenzialmente, una CRL è un oggetto statico, una sequenza di bit, un messaggio.

Se il browser non venisse informato della CRL, potrebbe validare certificati che sono stati revocati. Tuttavia, attualmente non esiste un protocollo definito per la distribuzione delle CRL.

In passato, le CRL venivano scaricate da entità come Verisign e integrate nel browser. Oggi, esiste un protocollo chiamato Online Certificate Status Protocol (OCSP), che funziona verificando lo stato di validità di un certificato direttamente presso l'ente di certificazione ogni volta che un utente cerca di accedere a un URL. Questo processo aiuta a garantire che i certificati revocati non siano considerati validi durante le connessioni web.



# Parte 6: Protocolli di sicurezza storici

## Protocollo Needham-Schröder

I protocolli di sicurezza spesso mancano di reciprocità, il che può rappresentare un limite. Nel protocollo di autenticazione esaminato, l'autenticazione avviene tramite la chiave privata del mittente.

Per implementare un meccanismo di sfida e risposta (Challenge-Response), utilizziamo la chiave pubblica del destinatario.

### **Protocollo di autenticazione di Bob con Alice con aggiunta di freshness**

1. Alice invia a Bob un messaggio cifrato con la chiave pubblica di Bob.
2. Bob decifra il messaggio utilizzando la sua chiave privata e risponde ad Alice con il messaggio in chiaro  $m$ .

Il focus qui è sull'**autenticazione, non sulla confidenzialità** del messaggio.

Tuttavia, il messaggio in chiaro letto da Alice non garantisce che stia comunicando effettivamente con Bob. Per questo motivo,  $m$  deve essere non prevedibile e quindi rappresentare una Nonce.

Alice genera un numero casuale  $N_a$  e lo cifra con la chiave pubblica di Bob, ottenendo  $\{N_a\}K_b$ .

Bob riceve il messaggio da Alice, lo decifra con la sua chiave privata e lo restituisce ad Alice in chiaro. In questo contesto di Challenge-Response, la confidenzialità della nonce non è necessaria.

---

Per autenticare Bob con Alice utilizzando un meccanismo di challenge-response con l'aggiunta di freshness, si seguono i seguenti passaggi:

- **Invio della challenge da parte di Alice:** Alice invia a Bob una nonce cifrata con la chiave pubblica di Bob,  $\{N_a\}K_b$ .
- **Risposta di Bob:** Bob riceve il messaggio cifrato da Alice e lo decifra con la sua chiave privata, ottenendo la nonce  $N_a$ . Successivamente, Bob invia la nonce in chiaro ad Alice.

In questo protocollo, chiunque desideri confermare l'autenticità di Bob deve includere la nonce nella sua risposta e riceverla indietro intatta. Questo garantisce la "freshness" dei dati scambiati durante il processo di autenticazione.

## Come mutualizzare la garanzia di freshness? Come autenticare Alice con Bob?

Per mutualizzare la garanzia di freshness e autenticare Alice con Bob, si può seguire questo protocollo:

- **Invio della challenge da parte di Bob:** Bob invia ad Alice una nonce cifrata con la chiave pubblica di Alice,  $\{Nb\}K_a$ .
- **Risposta di Alice:** Alice riceve il messaggio cifrato da Bob e lo decifra con la sua chiave privata, ottenendo la nonce Nb.

In questo modo, entrambi Bob e Alice possono essere certi che l'altro abbia ricevuto e utilizzato una nuova nonce, garantendo così la freschezza dei dati scambiati e mutua autenticazione.

## Come ottimizzare questo protocollo?

Possiamo unificare i 2 step centrali dove Bob invia informazione ad Alice, il protocollo passa da:

1.  $A \rightarrow B:\{Na\}K_b$ 
  - Alice invia a Bob una nonce cifrata con la chiave pubblica di Bob
2.  $B \rightarrow A:Na$ 
  - Bob invia ad Alice il nonce Na decifrato con la sua chiave privata
3.  $B \rightarrow A:\{Nb\}K_a$ 
  - Bob invia ad Alice una nonce cifrata con la chiave pubblica di Alice
4.  $A \rightarrow B:Nb$ 
  - Alice invia a Bob il nonce Nb decifrato con la sua chiave privata

A:

1.  $A \rightarrow B:\{Na\}K_b$ 
  - Alice invia a Bob una nonce cifrata con la chiave pubblica di Bob
2.  $B \rightarrow A:Na,\{Nb\}K_a$ 
  - Bob invia ad Alice il nonce Na decifrato con la sua chiave privata
  - Bob invia ad Alice una nonce cifrata con la chiave pubblica di Alice
3.  $A \rightarrow B:Nb$ 
  - Alice invia a Bob il nonce Nb decifrato con la sua chiave privata

Questa versione è sufficiente per l'autenticazione con freshness, possiamo però aggiungere un ulteriore livello di sicurezza cifrando tutte le nonce.

- 1.**  $A \rightarrow B: \{Na\}K_b$ 
  - Alice invia a Bob una nonce cifrata con la chiave pubblica di Bob
- 2.**  $B \rightarrow A: \{Na, Nb\}K_a$ 
  - Bob invia ad Alice le nonce Na ed Nb cifrate con la chiave pubblica di Alice
- 3.**  $A \rightarrow B: \{Nb\}K_b$ 
  - Alice invia a Bob il nonce cifrato con la chiave pubblica di Bob

Quando Bob riceve da Alice il primo nonce, non sa a chi replicare il messaggio è necessario quindi aggiungere nel messaggio un riferimento al mittente, il protocollo diventa quindi:

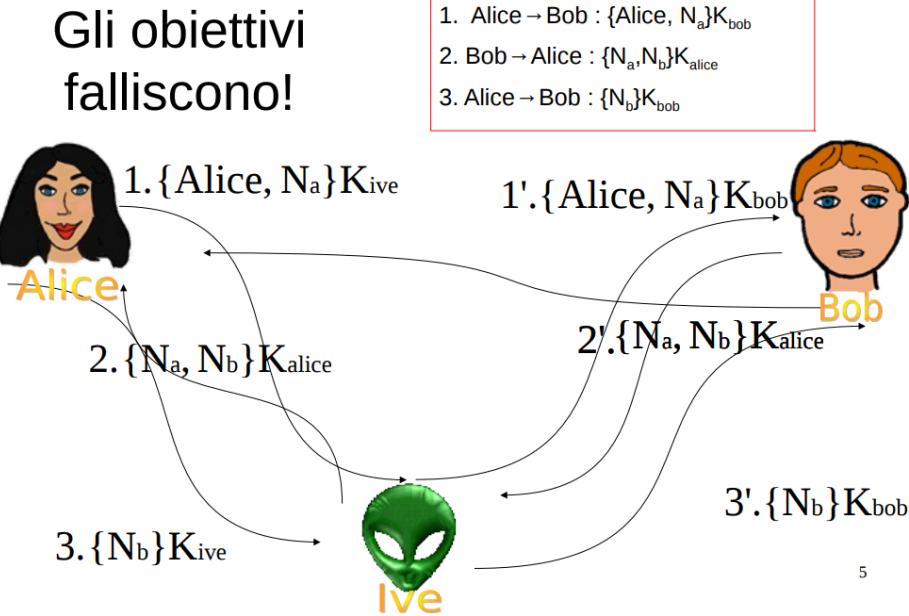
- 1.**  $A \rightarrow B: \{Alice, Na\}K_b$ 
  - Alice invia a Bob una nonce cifrata con la chiave pubblica di Bob insieme ad un riferimento a se stessa
- 2.**  $B \rightarrow A: \{Na, Nb\}K_a$ 
  - Bob invia ad Alice le nonce Na ed Nb cifrate con la chiave pubblica di Alice
- 3.**  $A \rightarrow B: \{Nb\}K_b$ 
  - Alice invia a Bob il nonce cifrato con la chiave pubblica di Bob



3

- Il protocollo soddisfa il requisito di autenticazione tra Alice e Bob? Sì
- Il protocollo soddisfa il requisito di autenticazione tra Bob e Alice? Sì
- Il protocollo soddisfa la confidenzialità della coppia  $\{Na, Nb\}$ ? Sì perché le nonce sono sempre cifrate.

## Scenario di attacco



Nel tentativo di comunicare con Ive, Alice invia  $\{Alice, Na\}K_{ive}$ .

Tuttavia, l'attaccante Dolev-Yao si finge Ive e intercetta il messaggio, ottenendo così la nonce Na di Alice. L'attaccante, impersonando Ive, manipola il flusso di comunicazione coinvolgendo Bob.

Ive, fingendosi Alice, riutilizza la nonce di Alice e, accanto ad essa, inserisce 'Alice', cifrando il tutto con la chiave pubblica di Bob. Invia quindi:  $\{Alice, Na\}K_{bob}$

Successivamente, Bob invia ad Ive la coppia  $\{Na, Nb\}K_{alice}$ .

Ive, non potendo decifrare il messaggio poiché non possiede la chiave privata di Alice, lo reinoltra ad Alice.

Alice riceve  $\{Na, Nb\}K_{alice}$  e nota che la nonce Na è tornata indietro, soddisfacendo così la challenge-response e autenticando erroneamente Ive senza rilevare alcuna anomalia nel protocollo.

Successivamente, Alice estrae la seconda nonce Nb e la rimanda a Ive, appena autenticato, cifrandola con la chiave pubblica di Ive:  $\{Nb\}K_{ive}$ .

In questo modo, Ive ottiene la nonce Nb cifrandola con la chiave pubblica di Bob, precedentemente nota a Ive, mentre Bob, rivedendo la sua stessa nonce Nb, non rileva anomalie e ritiene di aver autenticato con successo Alice, ignaro del fatto che il suo interlocutore è Ive.

# Potenziali soluzioni

Possiamo apportare queste modifiche al protocollo originale:

## Primo fix

1. Alice invia a Bob:  $\{\{Alice, Na\}K_{alice}^{-1}\}K_{bob}$

Mettiamo una cifratura con chiave privata nel messaggio 1 è ancora possibile effettuare l'attacco **non è quindi un fix corretto**.

## Secondo fix

1. Alice invia a Bob:  $\{\{Alice, Na\}K_{alice}^{-1}\}K_{bob}$
2. Bob invia ad Alice:  $\{\{Na, Nb\}K_{bob}^{-1}\}K_{alice}$

Possibile soluzione Alice si accorgerebbe di non star parlando con l'interlocutore corretto.

## Terzo Fix

2. Bob invia ad Alice:  $\{\{Na, Nb\}K_{bob}^{-1}\}K_{alice}$

Possibile semplificare il tutto modificando solo il secondo passaggio del protocollo

## Quarto Fix

2. Bob invia ad Alice:  $\{Na, Nb, Bob\}K_{alice}$

Possibile anche rimuovere il secondo livello di encryption, Alice non riesce ad identificare Ivo ed Ivo non può spacciarsi per Alice agli occhi di Bob. Il senso del fix è quello di esplicitare il mittente del messaggio all'interno del crittotesto come accade esattamente alla passo 1. Da ciò ne deriva che esplicitare l'identità del mittente è rilevante

## Quinto Fix

1. Alice invia a Bob:  $\{Alice, Bob, Na\}K_{bob}$

Nel messaggio 1 non ci sono vulnerabilità, modificare quello non porterebbe alcuna soluzione all'attacco.

Il problema dell'attacco sta al passaggio 3 che è il sintomo del problema ed è quindi necessario fixare il messaggio 3, modificando il messaggio 2 informando Alice se eventualmente tutti i controlli non sono andati a buon fine.

## Conseguenze dell'attacco

Le conseguenze di questo attacco sono significative. L'attaccante riesce a scoprire la chiave di sessione autenticandosi come Alice presso Bob, il che significa che l'attaccante può impersonare Alice e accedere ai servizi offerti da Bob utilizzando la chiave di sessione rubata. Questo è particolarmente critico nel contesto di un'applicazione web, dove il client si autentica al server tramite un cookie di sessione che contiene le nonces Na e Nb. Utilizzando questa chiave di sessione compromessa, l'attaccante potrebbe anche cifrare i messaggi, aumentando il potenziale danno dell'attacco.

Sebbene abbiamo discusso delle soluzioni per proteggere la nonce Nb, potrebbe sembrare che la protezione della nonce Na non sia stata considerata. Tuttavia, la considerazione principale è che, poiché Alice condivide la nonce Na con Ive per la sessione corrente, le sorti di Na sono meno rilevanti. Bob acquisisce Na in modo involontario a causa dell'attività fraudolenta di Ive; quindi, la policy associata a Na è che è stata concepita da Alice per condividerla con Ive.

Cambiarlo a un modello di attaccante più generale, come un attaccante generico, rende il quesito sulla confidenzialità di Na più rilevante. In questo caso, il fatto che Bob abbia acquisito la nonce in modo indesiderato potrebbe sollevare preoccupazioni sulla sicurezza dei dati e sulla privacy degli utenti.

In sintesi, mentre la protezione di Nb è prioritaria, non bisogna trascurare la protezione di Na, specialmente considerando gli scenari in cui la sicurezza dei dati e la privacy degli utenti sono in gioco.

## Vendetta nel modello General Attacker

Nel modello di attaccante generico, Bob, scoprendo l'importanza della nonce Na, potrebbe pianificare una vendetta contro Ive. Supponendo che Alice sia una banca, Bob potrebbe eseguire un attacco come segue:

Bob si impone come Ive, utilizzando la chiave di sessione rubata {Na, Nb}, e invia a Alice la seguente richiesta:

{Na, Nb, "trasferisci 20000€ dal conto di Ive al conto di Bob"}Kalice

In questo scenario, Alice (la banca) riceve la richiesta e la interpreta come legittima poiché soddisfa la challenge con la nonce Nb. Di conseguenza, esegue il trasferimento di denaro dal conto di Ive al conto di Bob, credendo che la richiesta provenga da Ive stesso. Questo tipo di attacco rappresenta un grave rischio per la sicurezza e la fiducia nell'ambiente online. Dimostra l'importanza di proteggere sia la nonce Na che la nonce Nb e sottolinea la necessità di implementare misure di sicurezza robuste per prevenire l'usurpazione delle identità e le frodi finanziarie.

## Principi di disegno: Explicitness

**Definizione:** Se le identità del mittente e del ricevente sono significative per il messaggio, allora è prudente menzionarle esplicitamente.

È preferibile specificare chiaramente le identità del mittente e del destinatario quando queste informazioni sono rilevanti.

### **Ma quando sono rilevanti?**

In modo sorprendente, includere il nome "Alice" nel messaggio 1 potrebbe non essere essenziale. Eliminare il riferimento ad "Alice" dal messaggio 1 potrebbe sembrare insignificante dal punto di vista della sicurezza, poiché chiunque potrebbe inserire "Alice" in quel messaggio. L'identità del mittente potrebbe essere dedotta dal livello di trasporto.

Seguire il principio di esplicitazione non è da solo garanzia di successo del protocollo, ma rappresenta un principio di prudenza fondamentale.

## Riepilogo del protocollo Needham-Schröder

Il protocollo di Needham-Schröder, un'implementazione di cifratura asimmetrica, è un metodo di sicurezza basilare che non trova impiego nelle autenticazioni dove è richiesto un alto livello di sicurezza, come nei siti bancari. Esiste uno scenario in cui la proprietà di autenticazione fallisce e la confidenzialità della chiave di autenticazione è compromessa. Questo tipo di attacco è stato scoperto 15 anni dopo la creazione del protocollo. Se consideriamo l'utilizzo di questo protocollo con un modello di attaccante più moderno rispetto al 2003 (General Attacker), possiamo giustificare il motivo per cui il protocollo non era stato progettato per affrontare un modello di attaccante come il Dolev-Yao, che è emerso solo 5 anni dopo la sua creazione. Inoltre, il protocollo non era stato concepito per gestire situazioni in cui Alice inizia una sessione con l'attaccante. Era stato progettato per agenti che si fidavano reciprocamente.

## Protocollo di Woo-Lam

Protocollo a cifratura simmetrica, unidirezionale e di tipo challenge-response, il problema è quello della condivisione della chiave iniziale risolvibile con Diffie-Hellman, una misura tipica è la presenza di un garante delle comunicazioni (TTP: Trusted Third Party). Se tra Alice e Bob è presente un TTP incorruttibile e fidato è possibile agevolare i due interlocutori, una delle ipotesi di lavoro è che il TTP conosca tutte le chiavi a lungo termine degli agenti. Questo equivale a dire che Alice e Bob sono registrati su un server che si chiama TTP che conosce entrambe le loro password. L'obiettivo del protocollo è autenticare unidirezionalmente Alice con Bob, non necessariamente viceversa, a differenza del protocollo precedente dove l'autenticazione è mutua.

- $A \rightarrow B: A$ 
  - Alice prova ad autenticarsi con Bob e lo interessa in maniera molto essenzialista, rappresenta un "Hello Message" inviando a Bob un riferimento a se stessa, ma chiunque potrebbe dire a Bob di essere Alice.
- $B \rightarrow A: Nb$ 
  - Bob invia ad Alice un challenge, una nonce.
- $A \rightarrow B: \{Nb\}K_a$ 
  - Alice cifra con la sua chiave la nonce appena ricevuta e la invia a Bob.
- $B \rightarrow TTP: \{\{A, Nb\}K_a\}K_b$ 
  - Bob riceve da Alice  $\{Nb\}K_a$  e non se ne fa nulla in quanto non è in grado di decifrarlo, lo inoltra quindi al TTP insieme al riferimento al passo 1. Questa query è cruciale, se nel mezzo Dolev-Yao dovesse scombinare questo abbinamento fallirebbe il protocollo, è necessario quindi cifrare entrambi con la chiave di Bob  $K_b$ . L'integrità non è garantita, il TTP decifra con la chiave di Bob ed otterrà una coppia dove la prima componente è leggibile.
- $TTP \rightarrow B: \{Nb\}K_b$ 
  - TTP ha ricevuto il messaggio 4 che è un crittotesto, guarda il livello di trasporto e capisce che viene da Bob, ma se il protocollo viene manomesso TTP prenderà la chiave sbagliata, non viene violata né l'autenticazione, né la confidenzialità, ci sarà solo una interruzione di servizio (DDOS) in quanto il messaggio non sarà leggibile, la sessione avrà un aborto. Se nulla di tutto ciò accade, dal suo database il TTP accede alla chiave di Bob, decifra, cerca la chiave di Alice nel suo database, decifra la componente ed estrae la nonce Nb che invia nuovamente a Bob cifrandola con la chiave di Bob. A questo punto Bob valuterà se autenticare o meno Alice in base alla nonce ricevuta.

### Perchè il messaggio 5 di Woo-Lam è cifrato?

Il messaggio 5 nel protocollo di Woo-Lam è cifrato per garantire l'autenticità del mittente, ovvero il Trusted Third Party (TTP), e per proteggere la confidenzialità della nonce  $N_b$ . Questo è importante perché Bob riceverà il messaggio cifrato con la sua chiave privata, che potrebbe essere decifrato solo da qualcuno che conosce la sua chiave a lungo termine, il che nel contesto del protocollo può essere solo il TTP.

Uno scenario di attacco sarebbe indurre Bob a credere che Alice sia online quando in realtà è offline, evidenzia l'importanza di garantire l'autenticità e l'integrità delle comunicazioni nel contesto del protocollo di autenticazione. La cifratura del messaggio 5 con la chiave di Bob contribuisce a mitigare questo tipo di rischio, proteggendo Bob da potenziali tentativi di inganno.

1. A → B : A
2. B → A :  $N_b$
3. A → B :  $\{N_b\}K_a$
4. B → TTP :  $\{A, \{N_b\}K_a\}K_b$
5. TTP → B :  $\{N_b\}K_b$

## Attacco su Woo-Lam

Alice è offline e l'attaccante si spaccia per lei. Questo è possibile tramite l'ausilio di due sessioni, Bob farà abort in una sessione ed Ok nell'altra quella dove Charlie si impersonifica in Alice.

- **Sessione 1:**  $C \rightarrow B: A$ 
  - L'attaccante Charlie dice a Bob di essere Alice.
- **Sessione 2:**  $C \rightarrow B: C$ 
  - L'attaccante Charlie dice a Bob di essere Charlie.
- **Sessione 1:**  $B \rightarrow A: N_b$ 
  - Bob genera una nonce  $N_b$  e la invia ad Alice.
- **Sessione 2:**  $B \rightarrow A: N'_b$ 
  - Bob genera una nonce  $N'_b$  e la invia a Charlie.

Ma ricordiamo che Alice e Charlie sono la stessa persona ma con 2 sessioni differenti.

- **Sessione 1:**  $C \rightarrow B: \{N_b\}K_c$ 
  - Charlie invia a Bob la nonce  $N_b$  cifrata con la sua chiave  $N_c$
- **Sessione 2:**  $C \rightarrow B: \{N_b\}K_c$ 
  - Charlie invia a Bob la nonce  $N_b$  cifrata con la sua chiave  $N_c$

Invia la stessa nonce o, meglio, lo stesso challenge ad ambedue le sessioni.

- **Sessione 1:**  $B \rightarrow TTP: \{\{A, N_b\}K_c\}K_b$ 
  - Bob prende il challenge cifrato e lo accoppia ad Alice cifrando il tutto con la propria chiave a lungo termine e lo invia al TTP.
- **Sessione 2:**  $B \rightarrow TTP: \{\{C, N_b\}K_c\}K_b$ 
  - Bob prende il challenge cifrato e lo accoppia a Charlie cifrando il tutto con la propria chiave a lungo termine e lo invia al TTP.
- **Sessione 1:**  $TTP \rightarrow B: \{N_b''\}K_b$ 
  - Il TTP prova a decifrare e non ottiene nulla di significativo, una nonce che chiamiamo  $N_b''$  e rispedisce a Bob cifrando con la chiave di Bob.
- **Sessione 2:**  $TTP \rightarrow B: \{N_b\}K_b$ 
  - Il TTP decifra il messaggio, questa volta l'associazione è corretta ed ottiene  $N_b$  e rispedisce a Bob cifrando con la chiave di Bob.

Bob rivedendo indietro  $N_b$  autentica Alice ma che in realtà è Charlie non riuscendo a distinguere le due sessioni avviate.

- Bob vede indietro Nb
- Pertanto, autentica l'utente cui l'ha associata, ossia Alice
- Alice potrebbe perfino essere off-line
- Bob non distingue la sessione!

1. C → B : A

1'. C → B : C

2. B → A : N<sub>b</sub>

2'. B → C : N<sub>b</sub>'

3. C → B : {N<sub>b</sub>}K<sub>c</sub>

3'. C → B : {N<sub>b</sub>}K<sub>c</sub>

4. B → TTP : {A, {N<sub>b</sub>}K<sub>c</sub>}K<sub>b</sub>

4'. B → TTP : {C, {N<sub>b</sub>}K<sub>c</sub>}K<sub>b</sub>

5. TTP → B : {N<sub>b</sub>''}K<sub>b</sub>

5'. TTP → B : {N<sub>b</sub>}K<sub>b</sub>

## Soluzione all'attacco su Woo-Lam

Chi fa l'errore? Bob, non il TTP è necessario quindi operare solo sul passaggio 5.

Basterebbe informare B nel messaggio 5 dell'identità dell'agente la cui chiave a lungo termine è stata usata per decifrare il challenge privato, aggiungiamo A al messaggio 5 dentro l'encryption per non sbagliarsi sulle sue conclusioni.

Nel protocollo modificato i passi delle rispettive sessioni sarebbero così riformulati;

- **Sessione 1:** TTP → B:{Nb'',A}K<sub>b</sub>
- **Sessione 2:** TTP → B:{Nb,C}K<sub>b</sub>

Adesso Bob effettuerà l'abort di entrambe le sessioni non riconoscendo nella prima la nonce Nb'' e nella seconda riconosce la nonce Nb ma non c'è l'associazione tra la nonce ricevuta e Charlie che non è il suo interlocutore originale.

# Parte 7: IPSEC

## Protocolli di sicurezza

La sicurezza informatica si colloca sia a livello della comunicazione che al di sopra di essa, e si basa su misure fondamentali come:

- **Crittografia**
- **Steganografia**

La crittografia trasforma i dati in modo che siano illeggibili per chi non possiede la chiave di decifratura, mentre la steganografia nasconde informazioni all'interno di altri dati, solitamente modificando i bit meno significativi di un'immagine o di un altro file. Tuttavia, questo approccio non è particolarmente robusto dal punto di vista della sicurezza, poiché nascondere informazioni in un luogo ovvio aumenta il rischio di scoperta da parte di terze parti.

## Alternative alla crittografia

### **Chaffing & Winnowing: Mescolare e Separare**

Inventata da Ron Rivest, il co-creatore di RSA, questa tecnica dimostra che la sicurezza non è raggiungibile solo attraverso la crittografia.

Il funzionamento è il seguente: invece di cifrare direttamente il messaggio, viene applicato un MAC (Message Authentication Code), una sorta di impronta digitale, al segreto. Quando il destinatario riceve il MAC, lo confronta con quello che dovrebbe essere sulla base della coppia ricevuta.

Contemporaneamente, vengono inviate al destinatario una serie di coppie fittizie, come se fossero messaggi subliminali inseriti all'interno della comunicazione reale. Un potenziale ascoltatore vede un'enorme quantità di traffico e presume che tutte le coppie siano valide, mentre solo il destinatario può identificare la coppia corretta grazie al controllo di integrità che va a buon fine solo con la coppia giusta. Tuttavia, con questo schema, un attaccante può ancora vedere il segreto, ma non riesce a capire quale sia tra tutte le altre coppie fittizie. Se il segreto è una password, l'attaccante può tentare di indovinarla provando tutte le possibili combinazioni. Mentre con la crittografia l'informazione viene trasformata in modo che non sia deducibile sintatticamente (confidenzialità tramite non deducibilità), il metodo Chaffing & Winnowing garantisce la confidenzialità attraverso l'indistinguibilità tra le informazioni genuine e quelle fittizie.

## Sicurezza a quale livello?

Per comunicare mediante un protocollo di sicurezza, tutte le macchine devono eseguire un client per quel protocollo. Se ne può fare a meno se garantiamo sicurezza ad un livello più basso. Tutte le “vecchie” applicazioni distribuite diventerebbero automaticamente “sicure”.

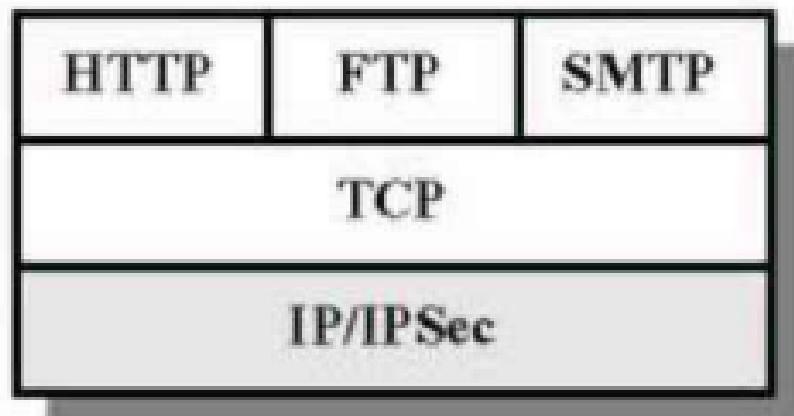
La sicurezza può essere integrata a vari livelli nello stack TCP/IP. Al livello inferiore, diventa più trasparente e rigida, mentre al livello superiore richiede una gestione dedicata, offrendo così maggiore flessibilità.

IPSEC rappresenta un protocollo di sicurezza operante a livello IP. Tuttavia, come può essere realizzata l'autenticazione, la confidenzialità e l'integrità a livello IP? La soluzione consiste nella crittografia dei pacchetti IP.

## Sicurezza a livello di rete

La sicurezza a livello di rete presenta vantaggi e svantaggi distinti:

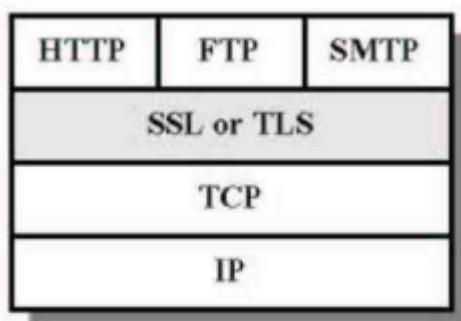
- **Pro:**
  - Le applicazioni e gli utenti possono rimanere all'oscuro delle procedure di sicurezza.
- **Contro:**
  - Tuttavia, può appesantire significativamente le comunicazioni, favorendo attacchi di tipo DoS.
  - Potrebbe richiedere modifiche al sistema operativo.



## Sicurezza a livello di trasporto

La sicurezza a livello di rete presenta vantaggi e svantaggi distinti:

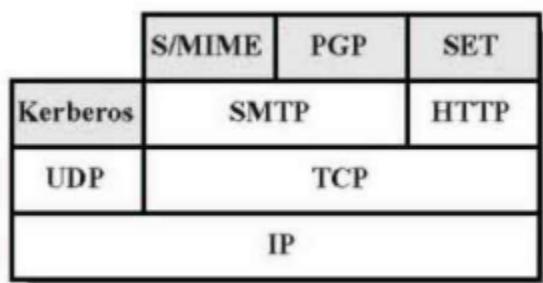
- **Pro:**
  - Essenzialmente, può essere implementata o a livello di trasporto o direttamente nelle applicazioni.
  - I principali browser sono dotati di client TLS integrato.
- **Contro:**
  - Naturalmente, richiede modifiche (anche minime) al livello che la riceve.



## Sicurezza a livello applicazione

Sicurezza a livello di applicazione presenta vantaggi e svantaggi distinti:

- **Pro:**
  - Consente una sicurezza personalizzabile, gestita direttamente dalle singole applicazioni.
- **Contro:**
  - Tuttavia, progettare la sicurezza a questo livello può essere ingannevole e comportare rischi imprevisti.



## IPSec

IPsec è una suite di tre protocolli fondamentali:

- **IKE (Internet Key Exchange)**: Implementa il protocollo Diffie-Hellman per avviare la sicurezza delle comunicazioni.
- **AH (Authentication Header)**: Fornisce autenticazione e integrità dei dati.
- **ESP (Encapsulated Security Payload)**: Garantisce la confidenzialità dei dati.

Mentre è obbligatorio per IPv6, per IPv4 è facoltativo.

Non è necessario che tutti i nodi di una rete siano compatibili con IPsec; è sufficiente che siano compatibili con IP, a differenza di IPv4 dove ogni nodo deve essere conforme a IP. In IPsec, è sufficiente che i nodi mittente e destinatario siano conformi a IPsec.

## Security Association (SA)

La Security Association (SA) rappresenta una regola fondamentale o policy di IPsec.

Si tratta di una relazione unidirezionale tra mittente e destinatario; sono necessarie due SA per una comunicazione bidirezionale.

L'identificatore della policy è chiamato SPI (Security Parameter Index).

Una SA specifica il tipo di sicurezza utilizzato (AH o ESP) e comprende tre campi principali:

- **SPI**: Security Parameter Index, che identifica la policy.
- **Indirizzo IP di destinazione**.
- **Identificatore del protocollo** (AH e/o ESP).

## Security Association Database (SAD)

Ogni nodo della rete deve gestire un Database delle Security Association (SAD) con tutte le SA attive su quel nodo.

Ogni voce nel database contiene tutti gli elementi della SA e include ulteriori informazioni:

- **Informazioni AH:** dettagli aggiuntivi sull'autenticazione.
- **Informazioni ESP:** dettagli aggiuntivi sulla confidenzialità.
- **Durata della SA** (Lifetime).

## Authentication Header (AH)

Authentication Header (AH) è un frammento aggiunto a ciascun pacchetto IP per supportare l'autenticazione e l'integrità dei pacchetti.

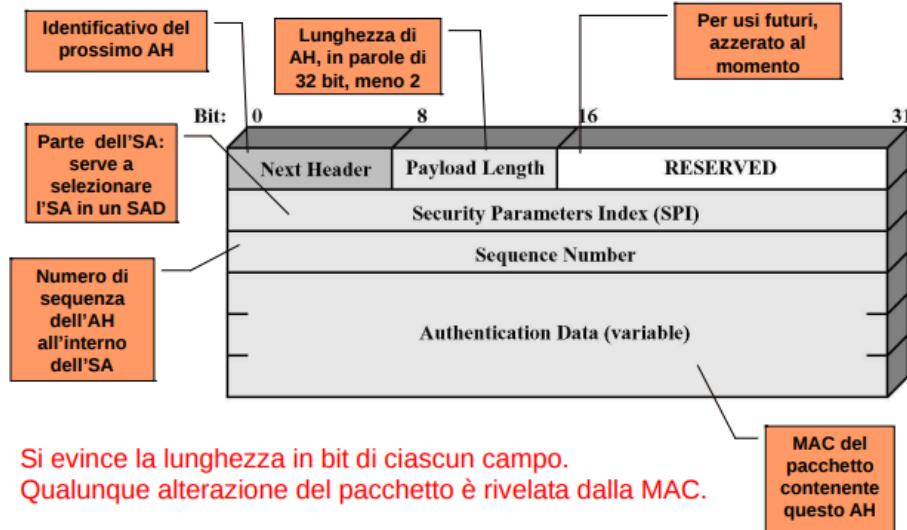
Per poter calcolare il Message Authentication Code (MAC), mittente e ricevente devono aver concordato in precedenza una chiave IKE.

AH autentica l'intero pacchetto, escludendo i campi variabili dell'header IP che potrebbero essere modificati dai nodi intermedi, come il tipo di servizio, i flag, l'offset del frammento, il time to live, e così via.

Questo protocollo previene sia gli attacchi di replay che l'IP spoofing. Il replay attack viene contrastato utilizzando un meccanismo di freshness come attributo della proprietà di autenticazione. Per contrastare l'IP spoofing, viene utilizzato il MAC, che fornisce sia autenticazione che integrità.

## AH - Formato

### AH – formato



- **Next Header:** Identifica il prossimo header dopo l'AH.
- **Payload Length:** Lunghezza del payload (dati) protetto da AH.
- **SPI (Security Parameters Index):** Identificatore che seleziona la Security Association (SA) nel Security Association Database (SAD).
- **Sequence Number:** Numero di sequenza dell'AH all'interno della SA.
- **MAC (Message Authentication Code):** Codice di autenticazione del messaggio per il pacchetto contenente l'AH.

Si evince la lunghezza in bit di ciascun campo. Qualunque alterazione del pacchetto è rivelata dalla MAC.

## IPSec funzionamento di base

Il funzionamento di base di IPSec segue questi passaggi:

- Ogni nodo mittente seleziona una Security Association (SA) utilizzando il proprio Security Policy Database (SPD).
- L'Identificatore della Security Association (SPI) dell'SA scelta viene incluso nell'Authentication Header (AH) di ciascun pacchetto.
- Quando un nodo riceve un pacchetto con l'AH, estrae l'SPI dall'header per selezionare la corrispondente SA dal proprio Security Association Database (SAD).

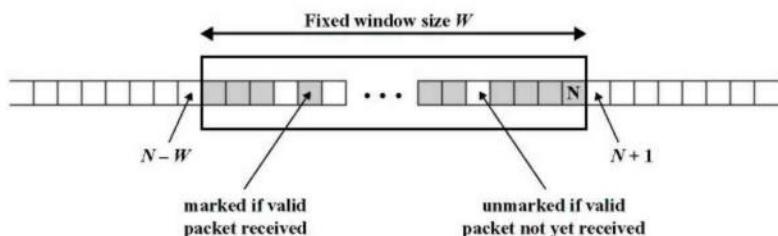
Il Security Policy Database (SPD) è una componente di IPSec che definisce le politiche per la gestione del traffico di rete. Decide quali pacchetti devono essere protetti da IPSec e come, basandosi su regole predefinite. In sostanza, l'SPD è il set di istruzioni che determina se e come un pacchetto di dati deve essere sicuro o meno.

## Prevenzione dei replay attack

I pacchetti, una volta autenticati, sono protetti dal rischio di essere replicati grazie all'utilizzo del campo Sequence Number nell'Authentication Header (AH).

Il mittente assegna un numero sequenziale compreso tra 0 e  $2^{32}-1$ . Se necessita di ulteriori pacchetti, deve negoziare una nuova Security Association (SA).

- Il ricevente deve rifiutare i pacchetti che sono vecchi, ripetuti o falsificati. Per fare ciò, accetta una "finestra" di dimensione  $W$  (tipicamente  $W=64$ ) di numeri di sequenza, dove  $N$  rappresenta il massimo numero nella finestra.
- Quando un pacchetto arriva, se il suo numero rientra nella finestra e non è già stato ricevuto, viene autenticato tramite il Message Authentication Code (MAC), e la posizione relativa viene segnata.
- Se il numero del pacchetto è maggiore di  $N$ , viene estesa la finestra verso destra fino a quel numero.
- Se il numero del pacchetto è inferiore o uguale a  $N-W$ , oppure il pacchetto è già presente o non è autenticato tramite MAC, viene segnalata un'anomalia.



## Questo meccanismo a finestra protegge da attacchi Dolev-Yao?

**Esempio reale:** Tick and Cross di studenti prenotati all'esame. Se due persone con lo stesso nome si presentano all'esame solo il primo di loro potrà effettuarlo. Non è un meccanismo di sicurezza fin quando non è possibile per uno studente cambiare il suo nome.

Nella pratica non è una misura di sicurezza dal momento in cui l'IP non venga modificato.

Tuttavia, va notato che questo meccanismo, da solo, non è una misura di sicurezza completa, in quanto dipende dalla corretta gestione del MAC. Il MAC permette di verificare che il pacchetto sia recente e che sia stato autenticato in modo integro, rendendo il meccanismo a finestra affidabile. Se il MAC è valido, ciò significa che il payload non è stato alterato e che il pacchetto è recente.

Pertanto, il sistema MAC garantisce l'integrità e l'autenticità dei dati, rendendo il meccanismo a finestra un metodo efficace per prevenire i replay attack quando utilizzato correttamente.

## Protezione mediante AH: Trasporto vs Tunnel

La protezione tramite Authentication Header (AH) può avvenire in due modalità: trasporto e tunnel.

- **Modalità trasporto:** Si applica solo ai dati del pacchetto e ai campi non variabili dell'header IP, fornendo protezione contro l'IP spoofing. In IPv6, questa modalità estende la protezione anche alle estensioni dell'header.
- **Modalità tunnel:** Protegge l'intero pacchetto IP (dati e header IP) e i campi non variabili dell'indirizzo IP esterno, garantendo così la sicurezza contro l'IP spoofing. Anche in IPv6, questa modalità estende la protezione alle estensioni dell'header.

Con IPSec è possibile creare un tunnel che funge da VPN. Trasformando un tunnel di rete in IPSec, si ottiene una VPN.

Nella modalità tunnel, se un pacchetto attraversa un nodo non compatibile con IPSec, il nodo lo inoltrerà senza interpretarne il contenuto, basandosi solo sull'header. Nella modalità trasporto, invece, un nodo non compatibile con IPSec non saprebbe come gestire il pacchetto, poiché non sarebbe in grado di comprendere il suo trattamento.

## AH in IPv4 ed in IPv6



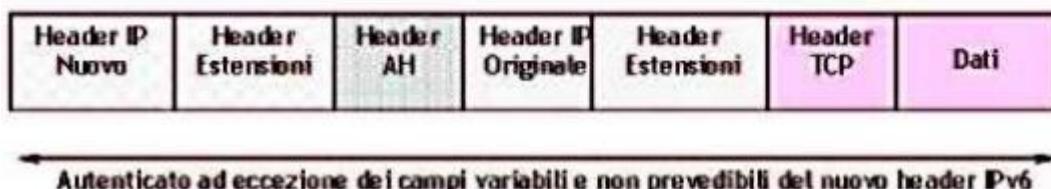
modalità  
trasporto



modalità  
tunnel



modalità  
trasporto



modalità  
tunnel

## Usi di AH: Trasporto vs Tunnel

	<b>Modalità trasporto</b>	<b>Modalità tunnel</b>
<b>Requisiti</b>	Mittente e ricevente devono usare IPSec	Mittente e ricevente possono non usare IPSec
<b>Garanzie</b>	<b>Autenticazione punto-punto (end-to-end)</b>	<b>Autenticazione intermedia (attraverso router/firewall)</b>

## AH in Tunnel

Nel contesto di un tunnel AH, consideriamo il NAT come una misura necessaria, spesso adottata per garantire la sicurezza. Ciò è dovuto al fatto che in molte situazioni non è possibile comunicare direttamente con un dispositivo su Internet; quindi, si crea una rete di collegamenti in cui i pacchetti attraversano altri dispositivi, eseguendo il NAT.

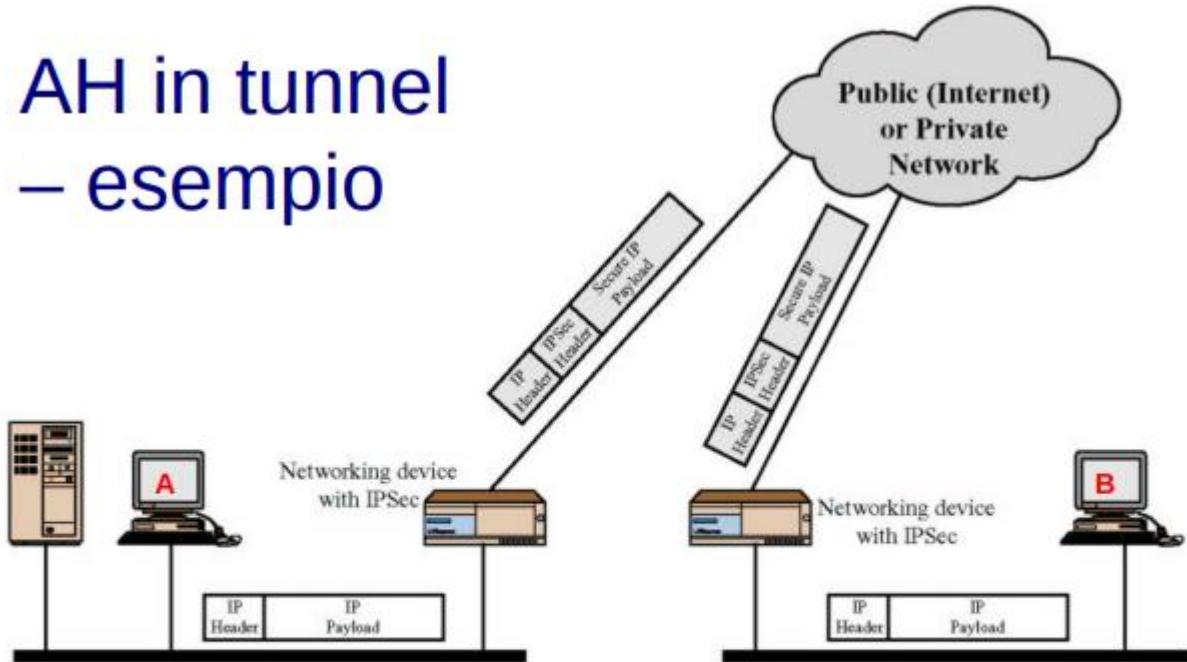
Questa situazione può essere paragonata a una triangolazione: sebbene il collegamento tra il client mittente e il server possa essere sicuro, il collegamento tra il server e il destinatario potrebbe non esserlo. Un esempio pratico può essere il controllo remoto di una lampada Tapo.

Tuttavia, se si utilizza la crittografia come nel caso di un tunnel AH, nessuno potrà violare le proprietà di sicurezza durante il passaggio attraverso la rete intermedia.

Un esempio di utilizzo di AH in modalità tunnel potrebbe essere il seguente:

- Il nodo A su una rete NA genera un pacchetto IP classico destinato al nodo B su una rete NB.
- Il router o il firewall di NA incapsula questo pacchetto in un pacchetto IPSec in modalità tunnel e lo inoltra al router o al firewall di NB.
- Questo dispositivo di NB estrae ed autentica il pacchetto IP originale; quindi, lo inoltra al nodo B sulla rete NB.

# AH in tunnel – esempio



A e B si scambiano pacchetti autenticati.  
I router/firewall intermedi incapsulano il traffico usando AH in modalità tunnel.

## Encapsulating Security Payload (ESP)

ESP (Encapsulating Security Payload) fornisce confidenzialità dei dati e, optionalmente, autenticazione.

- **Confidenzialità del contenuto:** ESP cifra il contenuto del pacchetto, garantendo che sia inaccessibile a chiunque non sia il destinatario autorizzato.
- **Confidenzialità del flusso di traffico:** ESP protegge anche il flusso di traffico da analisi da parte di intermediari, fornendo così un ulteriore livello di sicurezza.
- **Autenticazione opzionale:** ESP può anche autenticare i dati, se configurato di conseguenza.

ESP può essere utilizzato da solo o in sequenza insieme ad AH (Authentication Header).

ESP è considerato un altro header a sé stante, nonostante possa fornire funzionalità simili ad AH come l'autenticazione. Storicamente, AH e ESP sono stati sviluppati come concetti separati. Tuttavia, un'estensione di ESP permette anche l'autenticazione, portando ad una sovrapposizione di funzionalità tra i due protocolli.

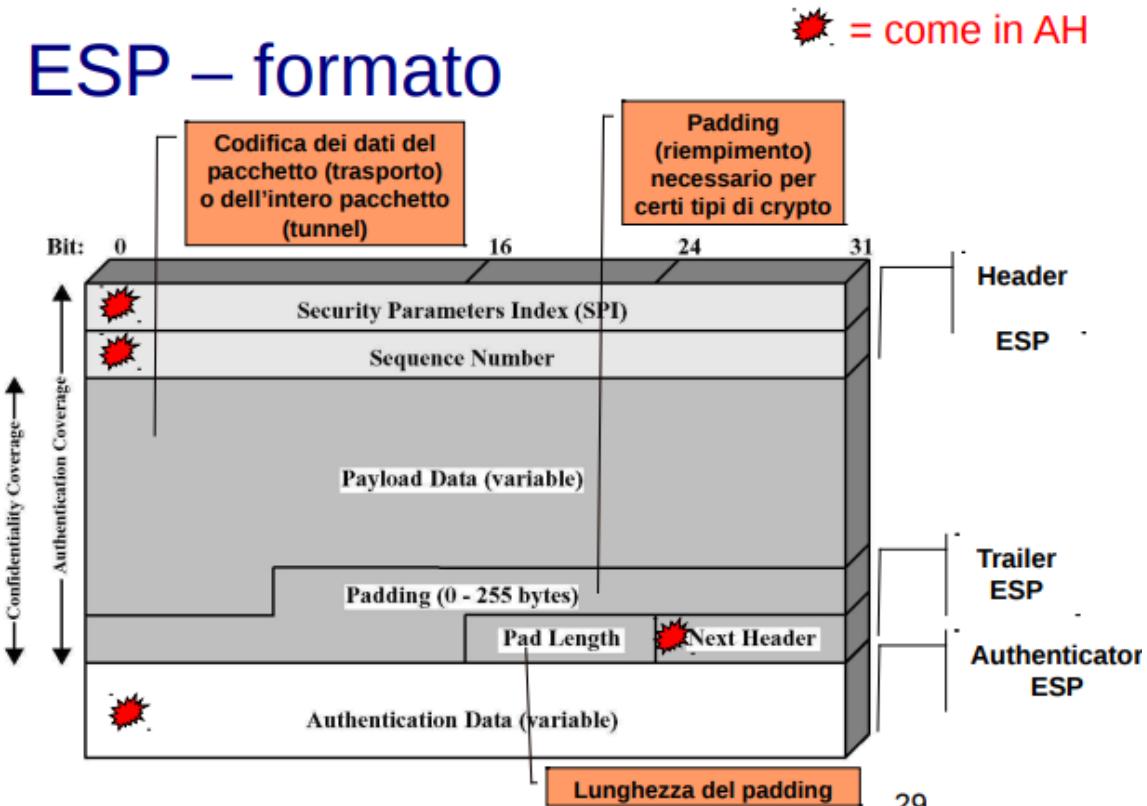
In sintesi, ESP è utilizzato principalmente per garantire la confidenzialità dei dati, ma offre anche la possibilità di autenticazione, fornendo così una solida protezione per le comunicazioni.

## ESP – Formato

ESP, l'Encapsulating Security Payload, protegge il contenuto dei pacchetti, ma non tutti gli elementi sono criptati. Ad esempio, l'indice di SPI (Security Parameters Index) è trasportato in chiaro, rendendolo soggetto ad analisi statistica. Tuttavia, ESP include diverse componenti:

- **SPI (Security Parameters Index):** Identifica i parametri di sicurezza associati a un flusso di dati.
- **Numero di sequenza:** Aiuta a garantire la freschezza dei dati e fornisce meccanismi anti-replay.
- **Payload:** Contiene i dati effettivi del pacchetto.
- **Padding:** Porzioni di dati riservate utilizzate per allineare i dati o per scopi di sicurezza.
- **Lunghezza del padding:** Indica la lunghezza del padding aggiunto al pacchetto.
- **Next Header:** Specifica il tipo di dati contenuti nel payload, consentendo al destinatario di interpretarlo correttamente.

Anche se ESP protegge il contenuto dei pacchetti, alcuni elementi rimangono visibili e potrebbero essere soggetti ad analisi, come nel caso dell'SPI. Pertanto, è importante considerare questo aspetto quando si valuta la sicurezza delle comunicazioni protette da ESP.

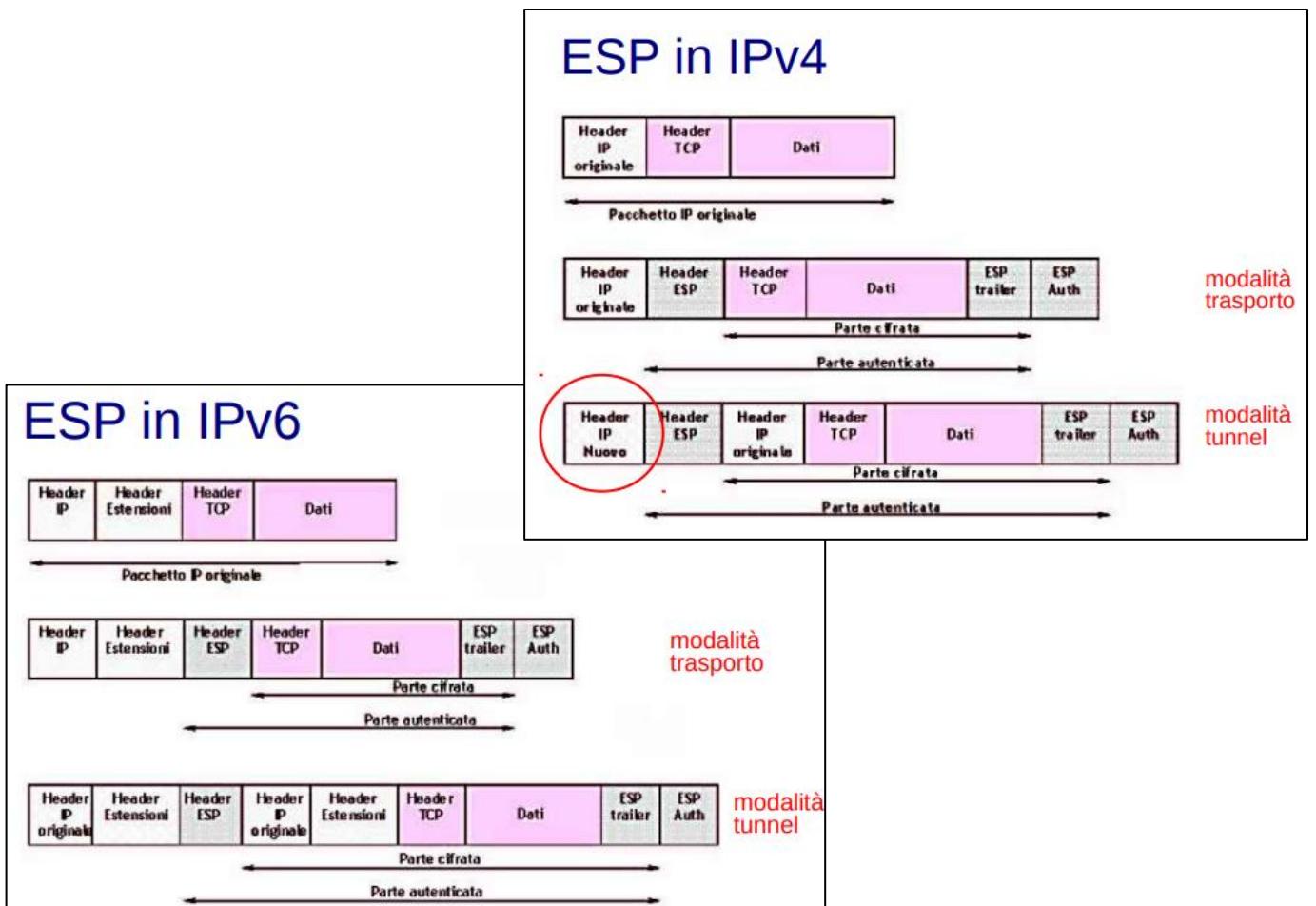


## ESP in IPv4

In modalità trasporto, ESP cifra l'header TCP, il payload e l'ESP trailer, ma non copre altri elementi come l'IP originale. Di conseguenza, l'indirizzo IP di destinazione rimane visibile e non è criptato, lasciando l'interlocutore esposto. Tuttavia, cifrare l'header IP potrebbe compromettere il funzionamento del protocollo IP.

Per ovviare a questa limitazione, possiamo optare per l'utilizzo di ESP in modalità tunnel. In questo caso, l'header IP originale viene cifrato e l'intero pacchetto, compreso l'header IP cifrato, viene incapsulato in un nuovo header IP. Un dispositivo intermedio, come un server anonimizzatore compatibile con IPSec, può fungere da punto di ingresso sicuro per il traffico.

Con la modalità tunnel, la crittografia avviene tra le reti, ad esempio tra due "nuvolette", piuttosto che direttamente tra i dispositivi finali. Questo approccio semplifica la gestione, poiché ogni tunnel può essere gestito singolarmente, rispetto alla gestione di ogni singolo dispositivo. Anche se non c'è una cifratura end-to-end, questa soluzione fornisce comunque un livello di sicurezza significativo per il traffico tra le reti coinvolte.



# Come rendere la rete domestica più sicura?

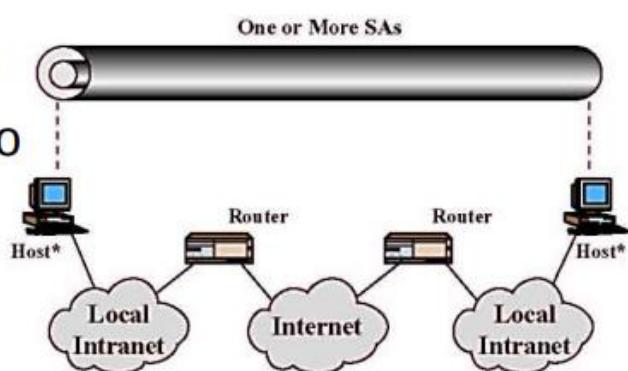
Per rendere la rete domestica più sicura, è possibile adottare diversi approcci:

1. **Attivare una VPN tramite il modem:** Se il software del modem lo consente, attivare una VPN direttamente sul modem può fornire una maggiore protezione per le comunicazioni in entrata e in uscita dalla rete domestica. Una VPN crittografa il traffico Internet, nascondendo così i dati sensibili dagli occhi indiscreti.
2. **Utilizzare servizi di NAT forniti dagli ISP:** Molti Internet Service Provider (ISP) offrono servizi di Network Address Translation (NAT), che forniscono una forma di protezione aggiuntiva per i dispositivi all'interno della rete domestica. Il NAT nasconde gli indirizzi IP locali dei dispositivi dietro un unico indirizzo IP pubblico.
3. **Considerare l'utilizzo di indirizzi IP pubblici:** Anche se gli indirizzi IP pubblici sono spesso considerati come un extra o offerti solo a un costo aggiuntivo, possono aumentare la sicurezza della rete domestica, specialmente se si desidera consentire l'accesso remoto ai dispositivi interni.
4. **Attivare una VPN tra il modem e il fornitore del modem:** Una VPN tra il modem e il provider del modem può creare un ulteriore strato di sicurezza, garantendo che il traffico sia critografato lungo tutto il percorso tra la rete domestica e il provider di servizi Internet.

Adottando queste misure, è possibile aumentare significativamente la sicurezza della propria rete domestica, proteggendo i dati personali e impedendo l'accesso non autorizzato alla rete.

## Tipo 1 di combinazioni SA: sicurezza punto-punto

1. AH in trasporto
2. ESP in trasporto
3. 1 & 2



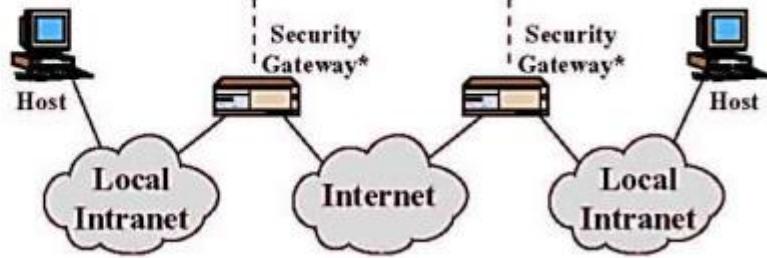
I nodi intermediari non devono per forza essere IPSEC-compliant

## Tipo 2 di combinazioni SA: sicurezza fra intermediari

1. AH in tunnel



2. ESP in tunnel



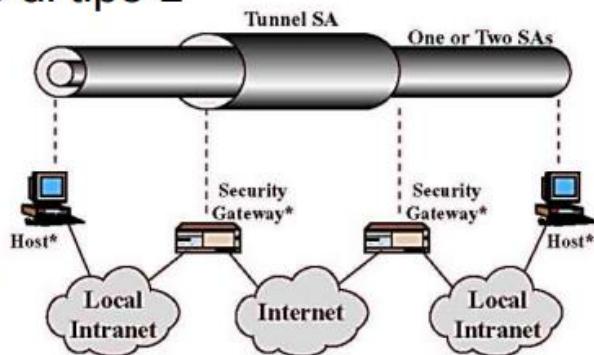
Gli host non devono per forza essere IPSEC-compliant e gli attacchi avverranno su internet.

## Tipo 3 di combinazioni SA: tipo 1 & tipo 2

1. Combinazione di tipo 1  
punto-punto

&

combinazione  
di tipo 2 fra  
intermediari



Combinazione delle tipologie appena viste

- Combinazione di tipo 1 punto-punto
- Combinazione di tipo 2 fra intermediari

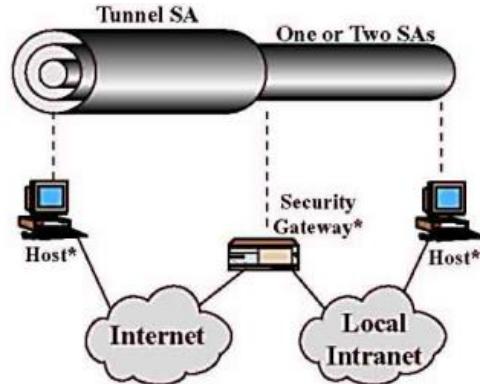
Aggiunta di un tunnel tra i gateway, se ci fosse un problema sui gateway ci sarebbe il tunnel interno a protezione della comunicazione.

## Tipo 4 di combinazioni SA: tipo 1 & sicurezza punto intermedio

1. Combinazione di tipo 1 punto-punto

&

(AH in tunnel  
punto-intermed. |  
ESP in tunnel punto-intermed.)



- Combinazione di tipo 1 punto-punto
- AH in tunnel o ESP in tunnel

Creo un tunnel tra l'host ed il gateway, i nodi devono essere IPSEC

## Internet Key Exchange (IKE)

Protocollo per la generazione delle chiavi da utilizzare con AH ed ESP.

IKE si fonda su una variante arricchita di Diffie-Hellman, integrata con ISAKMP. L'uso della cifratura asimmetrica favorisce la rapida e efficiente condivisione delle chiavi simmetriche. IPSEC, tramite il protocollo ISAKMP, elimina la necessità di avere chiavi aggiornate continuamente.

# Parte 8: Intrusion Detection

## Intrusione versus Malware

- Un intruso è un processo utente reale con determinati privilegi.
- I virus raramente possono essere considerati tipi particolari di intrusi. Un virus non è paragonabile a un batterio.
- Un worm, più facilmente, può essere considerato un tipo di intruso. Come un batterio, è autonomo.
- Un intruso non è né un virus né un worm, ma può installarne uno. Si fa riferimento ai processi intrusi come evidenziato dal comando "top" in Linux.

## Intrusione versus DoS

### **DoS:**

- Effetti temporanei
- Immediatamente pubblico
- Blocco delle risorse

### **Intrusione:**

- Effetti generalmente permanenti
- Spesso non reso pubblico
- Uso illecito di risorse

Gli attacchi DoS bloccano una macchina, gli intrusi sfruttano il sistema si parla di stealth e non valutabile.

<b>DoS</b>	<b>Intrusioni</b>
<ul style="list-style-type: none"><li>■ Effetti temporanei</li><li>■ Immediatamente pubblico</li><li>■ Blocco delle risorse</li></ul>	<ul style="list-style-type: none"><li>■ Effetti generalm. permanenti</li><li>■ Spesso non reso pubblico</li><li>■ Uso illecito di risorse</li></ul>

## Negazione del servizio DoS

**Definizione:** La negazione del servizio (DoS) è un tipo di attacco informatico che mira a impedire la normale operatività di un sistema, rendendolo inaccessibile agli utenti legittimi. Esistono diverse tipologie di attacchi DoS:

- **cDoS (Consumo di risorse computazionali):** Questo tipo di attacco sfrutta il consumo eccessivo di risorse computazionali del sistema bersaglio, come la CPU, impedendo così il normale funzionamento dei servizi.
- **mDoS (Consumo della memoria):** In questo caso, l'attaccante satura la memoria del sistema bersaglio, impedendo l'accesso ai dati e ai servizi.
- **bDoS (Consumo della banda di trasmissione):** Questo tipo di attacco sovraccarica la banda di trasmissione del sistema, rendendo difficile o impossibile per gli utenti legittimi accedere ai servizi.

Inoltre, c'è il concetto di DDoS (Distributed Denial of Service), che è una variante più sofisticata del DoS, in cui l'attacco è orchestrato da più macchine collegate in rete, spesso sfruttando una botnet. Questo rende l'attacco più potente e difficile da mitigare. Gli attacchi DoS sono indipendenti dall'autenticazione e possono essere eseguiti anche su sistemi autenticati. L'obiettivo principale è sovraccaricare il sistema o esaurire le sue risorse critiche, rendendolo inutilizzabile per gli utenti legittimi. Nel contesto dell'Internet of Things (IoT), la sicurezza riveste un ruolo cruciale, poiché i dispositivi connessi potrebbero essere vulnerabili agli attacchi DoS e potrebbero essere utilizzati come parte di una botnet per lanciare attacchi DDoS su larga scala.

## Dos vs DDoS

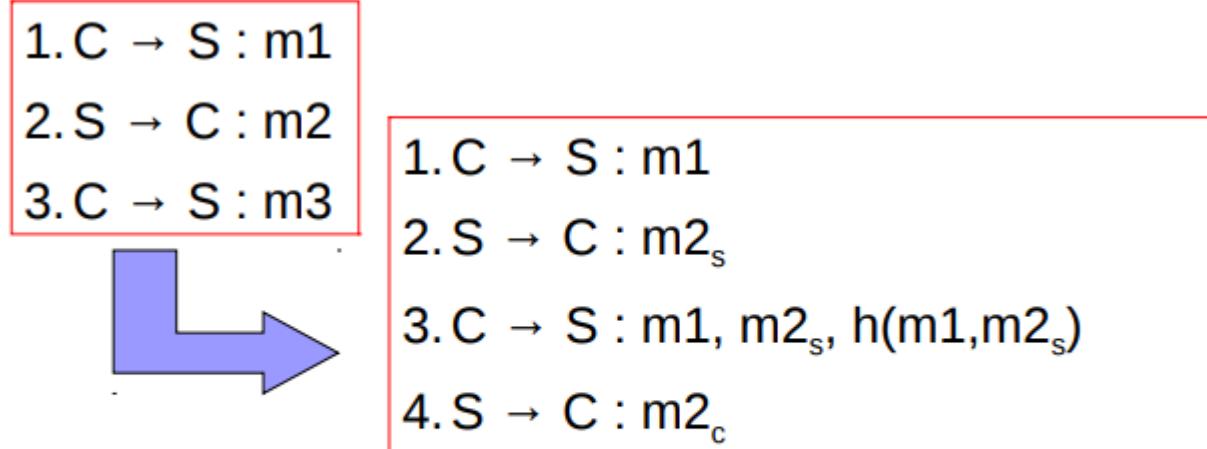
DoS è un attacco che sfrutta un router con funzionalità di broadcast. L'attaccante invia un pacchetto PING utilizzando l'indirizzo IP della vittima. Questo pacchetto viene poi riflesso da tutte le macchine nella sottorete, creando un attacco DDoS che sovraccarica il sistema bersaglio. In un'altra variante, l'attaccante istruisce un certo numero di "zombie" (macchine compromesse) a contattare un particolare indirizzo IP, il che porta alla saturazione della macchina dietro quell'IP.

## Come reagire?

Come affrontare un attacco DoS è ancora una sfida complessa oggi. Esistono firewall dotati di moduli antidos che possono filtrare il traffico, mentre un'alternativa strategica è l'implementazione della trasformazione dei cookie. Ma quale dovrebbe essere la risposta?

- Per un attacco DoS, una strategia è quella di chiudere le connessioni provenienti dalla rete attaccante.
- Ma per un attacco DDoS, chiudere tutte le connessioni può essere difficile e sconveniente.
- Una soluzione potrebbe essere il filtraggio del numero di connessioni accettate, ad esempio tramite htaccess con un firewall.
- Inoltre, una euristica di prevenzione potrebbe consistere nel complicare computazionalmente l'accesso, ad esempio tramite la trasformazione dei cookie (transformation cookie).

## Anti-DoS: Cookie Transformation



Creazione del messaggio computazionalmente complicato rimandata  
fino a quando è chiaro che il client sia pronto a ricevere all'IP del passo 1

Se il protocollo è semplice, l'attaccante può sfruttarlo per inviare più richieste contemporanee, causando confusione al server che potrebbe non essere in grado di gestirle tutte.

Una strategia per contrastare questo tipo di attacco è la trasformazione dei cookie. Il server, anziché rispondere direttamente con il messaggio m2, invia inizialmente un messaggio più semplice, m2s, sotto forma di cookie. Il protocollo richiede quindi al client di inviare lo storico dei messaggi m1 e m2s, insieme all'hash tra m1 e m2s. La verifica dell'hash è un'operazione veloce e consente al server di capire se il client è attivo dall'altra parte. Solo a questo punto, il server risponderà, poiché ha confermato l'attività del client.

Se il client non fosse attivo e cercasse di eseguire un attacco del tipo fork bomb, creando un loop malevolo di richieste, non sarebbe in grado di mantenere lo storico dei messaggi e calcolare l'hash richiesto. Di conseguenza, non avrebbe la possibilità di completare l'operazione richiesta, fornendo una protezione aggiuntiva contro gli attacchi DoS.

## Intrusione

**Definizione:** L'intrusione si verifica quando un individuo ottiene in modo illecito privilegi superiori a quelli legittimamente posseduti.

- Questo può avvenire attraverso l'acquisizione di accesso al sistema e l'estensione dei propri privilegi.
- Non è necessariamente limitato alle intrusioni dalla rete, ma può coinvolgere anche l'inserimento di software dannoso tramite dispositivi come floppy disk o pen drive.
- Spesso coinvolge la violazione dei meccanismi di autenticazione e controllo degli accessi.

In sintesi, l'intrusione comporta l'ottenimento di privilegi non autorizzati, violando le regole di accesso e autenticazione del sistema.

## Tecniche di intrusione

- **Violazioni di password:** tecniche standard e non-standard
- **Intercettazione di informazioni sensibili:** scovarle (non solo password) se transitano in rete senza crittografia oppure violare protocolli di sicurezza
- **Uso di combinazioni di SW nocivo:** sullo stile del worm Morris

# Intrusion Detection VS Intrusion Management

- **Intrusion Detection (rilevamento dell'intrusione):** È il processo di individuare attività sospette o non autorizzate all'interno di un sistema o di una rete. L'obiettivo principale è rilevare l'intrusione il prima possibile per prendere provvedimenti e prevenire danni maggiori.
- **Intrusion Management (gestione dell'intrusione):** Una volta individuato un intruso, questo processo mira ad espellerlo dal sistema o dalla rete il più rapidamente possibile per limitare i danni che può causare. L'obiettivo è minimizzare l'impatto negativo sull'integrità, la disponibilità e la riservatezza dei dati e dei servizi.

## Intrusion Detection System

Un IDS registra e valuta i log, analizzandoli per individuare attività sospette. L'approccio consiste nel registrare tutti i dettagli, il che si rivela prezioso in caso di problemi, consentendo un'auditing efficace dei log (come il comando "history" in Linux). In caso di attacco, l'IDS è responsabile di analizzare i log.

Questo sistema può operare sia in tempo reale che in modalità post-mortem, consentendo di individuare intrusioni nel sistema in tempo reale. Un Security Operations Center (SOC) è un insieme di tecnologie che consente di esternalizzare la gestione della rilevazione e prevenzione degli attacchi. Queste tecnologie, alimentate dall'intelligenza artificiale, offrono opportunità di lavoro per gli analisti SOC. Un insieme di tecnologie IDS costituisce una tecnologia nota come SIEM / SOAR (Security Incident and Event Management), che comprende sistemi di rilevamento delle intrusioni. Un IDS può essere implementato a livello di nodo sulla nostra macchina o come un sistema di rete. Il suo obiettivo è riconoscere i processi non autorizzati.

Per definire quali processi siano considerati non autorizzati, si possono adottare diverse strategie. Ad esempio, si può utilizzare una whitelist dei processi su una distro standard: i processi non presenti nella whitelist vengono considerati alieni. Tuttavia, questa soluzione statica potrebbe non essere sempre efficace. In alternativa, è possibile adottare un approccio dinamico che studia il comportamento del processo, analizzandolo per identificare eventuali comportamenti dannosi. In sintesi, un IDS registra il comportamento del sistema, analizzando i log attraverso tecniche di pattern matching e operando in tempo reale. Un IDS distribuito può coinvolgere più processi o macchine per la registrazione, l'analisi e la presa di decisioni.

## Definire i comportamenti

Il principale ostacolo per un Sistema di Rilevamento delle Intrusioni (IDS) è identificare i comportamenti dell'intruso in contrasto con quelli dell'utente legittimo. Questo può essere affrontato considerando il comportamento medio nel tempo e utilizzando tecniche di machine learning.

Tuttavia, l'obiettivo è trovare il miglior compromesso tra scambiare un utente legittimo per un intruso (falso positivo) e non rilevare un intruso come tale (falso negativo).

## Record di auditing

Ci servono i log, le entry di un log si chiamano record

- **Record Nativi:**
  - Tutti i sistemi operativi includono funzionalità per la raccolta di informazioni generiche sulle attività degli utenti.
  - Questi possono includere la storia dei comandi della shell.
  - Sebbene questa sia una soluzione semplice, non sempre fornisce informazioni dettagliate e utili.
- **Record Specifici per il Rilevamento:**
  - Si tratta di software aggiuntivo progettato per raccogliere esclusivamente informazioni specifiche per il rilevamento delle intrusioni.
  - Questo tipo di record può evidenziare deviazioni dal modello statistico o comportamenti sospetti.
  - Anche se può appesantire il sistema, fornisce informazioni mirate e utili per il rilevamento delle intrusioni.

Gli accessi vanno loggati e per legge ci vuole in certi casi.

## Record di Denning

I record di Denning sono un esempio di record specifici per il rilevamento delle intrusioni, ciascuno contenente sei campi distinti:

1. **Soggetto:** Rappresenta l'utente o il processo che esegue azioni tramite comandi specifici. Possono essere raggruppati in classi di accesso, noti anche come ruoli.
2. **Azione:** Indica l'operazione che il soggetto svolge sul sistema.
3. **Oggetto:** Rappresenta l'entità su cui viene svolta l'azione. È importante notare che un soggetto può anche essere un oggetto, consentendo un'analisi più dettagliata delle interazioni nel sistema.
4. **Eccezione:** Specifica se è stata prodotta un'eccezione in risposta all'azione e, in tal caso, quale eccezione è stata generata.
5. **Utilizzo delle risorse:** Questo campo fornisce informazioni dettagliate sulle risorse utilizzate durante l'azione, come il numero di righe stampate o visualizzate, il numero di settori letti o scritti, il tempo CPU impiegato e le unità di I/O utilizzate.
6. **Timestamp:** Serve come identificatore temporale che indica il momento in cui è iniziata l'azione.

Questi record offrono un quadro completo delle attività nel sistema, consentendo un'analisi approfondita per il rilevamento e la gestione delle intrusioni.

## Record di Denning – esempio

- Il comando  
`cp ~giamp/slides.pdf ~barba`
- Genera i 3 record relativi alle 3 azioni

giamp	execute	/bin/cp	0	CPU = 00002	11058721678
giamp	read	-giamp/slides.pdf	0	SECTORS = 5	11058721679
giamp	write	-barba	-1	SECTORS = 0	11058721680

anomalia

109

# Tecniche di rilevamento

- **Indipendenti dal Passato (Standard):**
  - Queste tecniche si basano su criteri predefiniti che non tengono conto delle precedenti attività nel sistema.
  - Un esempio è l'imposizione di un limite massimo di tentativi di inserimento della password, ad esempio massimo 3 fallimenti consentiti.
- **Dipendenti dal Passato:**
  - Queste tecniche si basano sull'osservazione o campionamento del funzionamento o comportamento del sistema in passato.
  - Si assume che il comportamento futuro sarà simile a quello passato.

Le tecniche di rilevamento possono essere suddivise in:

- **Rilevamento Statistico:**
  - Si tratta di creare un modello statistico del comportamento di un utente legittimo.
  - Quando un utente o un processo si discosta in modo significativo da questo modello, viene identificato come un potenziale intruso.
  - Questo metodo è particolarmente efficace per individuare attaccanti esterni che possono comportarsi in modo diverso dagli utenti legittimi.
- **Rilevamento a Regole:**
  - In questo approccio, vengono definite regole per il comportamento di un intruso.
  - Quando un'attività corrisponde a una di queste regole, viene identificata come potenziale intrusione.
  - Questo metodo è particolarmente efficace nel rilevare attaccanti interni che potrebbero avere una conoscenza approfondita del sistema e delle sue regole di funzionamento.

Queste tecniche non sono intelligenti, con nuove tecniche di machine learning è possibile prevedere eventuali cambi di programma.

- **Rilevamento statistico:** Creo un modello statistico per il comportamento di un utente legittimo. Se l'intruso si discosta sufficientemente dal processo allora lo flaggo come intruso
- **Rilevamento a regole:** Fare più di n tentativi in una certa fascia oraria vieni flaggato come intruso (due regole in questo caso)

## Rilevamento statistico – Modelli

- 1. Media e Deviazione Standard:** Questo modello calcola la media e la deviazione standard di un parametro nel corso del tempo. Questi valori forniscono un'indicazione del comportamento medio del parametro e della sua variabilità.
- 2. Modello Operativo:** Questo modello definisce un limite di accettabilità per un parametro e segnala un'eventuale intrusione quando questo limite viene superato. È un approccio basato su soglie di allarme.
- 3. Multivariazione:** In questo modello, vengono considerate più variabili simultaneamente per identificare modelli complessi di comportamento anomalo.
- 4. Processo di Markov:** Questo modello utilizza la teoria dei processi stocastici per modellare il comportamento del sistema nel tempo e prevedere le transizioni tra diversi stati.
- 5. Serie Temporale:** Questo modello analizza le variazioni di un parametro nel tempo, consentendo di individuare tendenze, stagionalità e comportamenti anomali nel tempo.

Ad esempio, il tempo trascorso dall'ultimo login potrebbe essere utilizzato come parametro per il rilevamento statistico. Un account inattivo da troppo tempo che effettua il login potrebbe indicare un account compromesso o non più utilizzato, noto anche come "account death".

## Rilevamento statistico – esempi

Parametro	Modello	Tipo di intrusione rilevata
<b>Login and Session Activity</b>		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off-hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.

## Rilevamento a regole

Il rilevamento a regole si basa su un vasto database di regole predefinite, ciascuna progettata per identificare modelli specifici di comportamento sospetto o attività anomale nel sistema.

Esempi noti di sistemi basati su regole includono Snort e ntop. Snort, ad esempio, utilizza un linguaggio di regole flessibile e potente, che permette agli amministratori di definire regole personalizzate per il rilevamento di attacchi e intrusioni.

Ogni regola di Snort, in sostanza, è simile a scrivere una riga di controllo di iptables, fornendo un metodo efficace per identificare e rispondere a varie tipologie di minacce informatiche.

## Tecniche di protezione

- **Limitazione dell'Automazione:** Questa tecnica mira a limitare il numero di tentativi di accesso automatico per prevenire attacchi di forza bruta o di tentativi ripetuti. Ad esempio, il sistema può essere configurato per ripristinare la connessione dopo un numero specifico di inserimenti di password errate (come 3), o il telefono cellulare può bloccare la SIM dopo un certo numero di tentativi di inserimento del PIN errati.
- **Utilizzo di Esche:** (Honey Pot) Le esche sono risorse fintizie progettate per attirare gli attaccanti e distoglierli dalle risorse più importanti o per farli rimanere più a lungo nel sistema, consentendo al sistema di identificarli e intraprendere azioni preventive. Ad esempio, possono essere creati URL come "transazioni.grossasocieta.com" o "conticorrenti.banca.com", che sembrano legittimi ma conducono a risorse monitorate.
- **Utilizzo di Trappole:** Questa tecnica prevede il massiccio monitoraggio di risorse fintizie o esche, e appena l'attaccante accede a una di esse, viene attivata una trappola che permette di rilevare e rispondere all'attacco in tempo reale.

# Parte 9: Software nocivo (Malware)

## Bug VS Software Nocivo

**Bug e Vulnerabilità:** Un bug è una proprietà inattesa del software. Anche se non necessariamente nocivo, un bug può essere sfruttato da altri software. Ad esempio, un'applicazione utente su Windows 2000 potrebbe sovrascrivere il modulo di sicurezza in memoria. I bug sono solitamente preterintenzionali, ovvero non introdotti deliberatamente.

**Software Nocivo:** Il software nocivo (malware) è scritto con l'esplicito scopo di violare la sicurezza di un sistema. Non è detto che sfrutti dei bug per farlo. Ad esempio, un virus può infettare un sistema anche se non ci sono vulnerabilità evidenti. È possibile scaricare un allegato contenente un eseguibile (.exe, .bat) che rappresenta il malware, il quale sfrutta il sistema ospitante indipendentemente dalla presenza di bug.

Un bug preterintenzionale non rende necessariamente nocivo il software che lo ospita. Tuttavia, la presenza di malware implica sempre un'intenzione malevola.

**Nesso tra Attività Offensiva e Malware:** Il legame tra attività offensiva e malware non è sempre evidente. Ad esempio, un penetration tester etico può impiantare malware per testare la sicurezza di un sistema. Questo è un caso di sfruttamento di vulnerabilità, noto come exploitation. Un sistema può avere vulnerabilità, come un buffer overflow, che possono essere scoperte tramite hacking etico.

**Trojan:** I trojan sono software nocivi che spesso vengono utilizzati dai penetration tester. Queste attività sfruttano vulnerabilità del sistema. Tuttavia, un sistema solido in termini di vulnerabilità può comunque essere compromesso tramite malware se l'utente esegue un file infetto.

**Difesa contro il Malware:** Microsoft Defender è un servizio che include antivirus, IDS (Intrusion Detection System) e firewall in un unico pacchetto. Questo tipo di software, sebbene chiuso e monolitico, rappresenta una difesa importante contro il malware.

**Affrontare le Vulnerabilità:** Lo scenario malware richiede un sistema antimalware o un security center. Tuttavia, affrontare le vulnerabilità by design è una sfida diversa. Gli antivirus non rilevano sempre le vulnerabilità, e spesso non lo fanno. La soluzione è l'aggiornamento del software, che richiede una fiducia costante ed un atto di fede nel processo di sviluppo e distribuzione degli aggiornamenti.

## Approfondimento sulle Vulnerabilità

### Definizione:

- Le vulnerabilità sono specifiche istanze di problemi di sicurezza su determinati sistemi.

### Gestione delle Vulnerabilità:

- **Repository Pubblici:** Gli hacker etici pubblicano le vulnerabilità scoperte in repository pubblici.
- **Vulnerability Response:** Prima della pubblicazione, le vulnerabilità devono essere risolte, dando al venditore il tempo necessario per proteggere il sistema.
- **Aggiornamenti Software:** Risolvono le vulnerabilità note (n-day) presenti nei repository pubblici.

### Vulnerabilità Zero Day:

- Una vulnerabilità zero day è una falla di sicurezza non ancora documentata.
- Gli attaccanti sfruttano queste vulnerabilità prima che siano conosciute e risolte.
- I penetration tester possono scoprire vulnerabilità zero day durante le loro attività offensive.

### Esempio di Exploit:

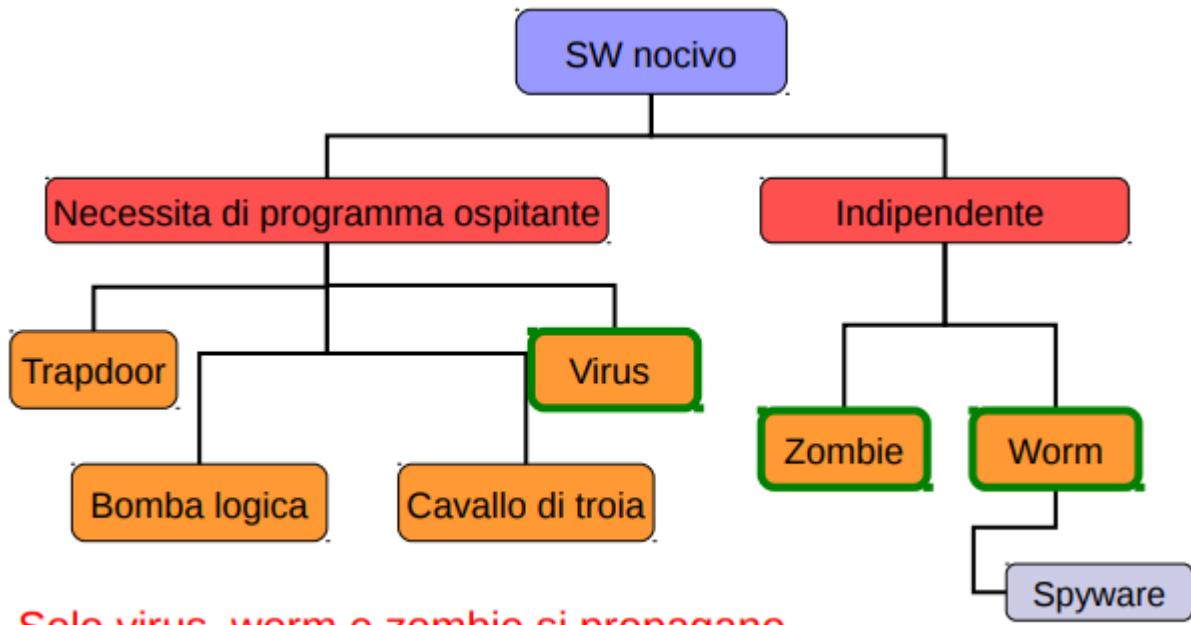
- Zippando un file eseguibile con una password e allegandolo a una mail, è possibile evitare che il sistema di sicurezza lo riconosca immediatamente.
- Alcuni gestori di posta elettronica, come Google, possono comunque bloccare questi file per proteggere gli utenti.

## Software Nocivo - Caratteristiche

- **Carico:** specifica violazione di sicurezza
  - Sempre presente
- **Propagazione:** meccanismo di trasmissione fra le macchine
  - Non sempre presente

La pericolosità dipende dal contesto: un carico elevato può distruggere un sistema critico, mentre un'alta propagazione può causare epidemie su larga scala. Entrambi rappresentano seri rischi.

# Software Nocivo - Tassonomia



Si suddividono in software che necessitano un programma ospitante e quelli indipendenti.

## Necessitano di un programma ospitante:

- Trapdoor
- Virus
- Bomba logica
- Trojan

## Indipendenti:

- Zombie: Potenzialmente tutti i dispositivi IoT se vulnerabili possono essere trasformati in zombie
- Worm (Spyware)

Si propagano solo virus, zombie e worm.

## Trapdoor (Backdoor)

### **Definizione:**

Un punto segreto di accesso a un programma che bypassa le procedure di sicurezza normalmente previste.

### **Caratteristiche:**

- Oltrepassa le misure di sicurezza previste, rappresentando una vulnerabilità.
- Originariamente concepita per semplificare il beta-testing.
- La conoscenza della trapdoor è necessaria per il suo utilizzo.
- Tipicamente implementata per comodità in passato, spesso per motivi di testing e basata sulla "security by obscurity".

## Bomba Logica

### **Definizione:**

Frammento di codice all'interno di un programma non nocivo che "esplode" quando si verificano certe condizioni

### **Caratteristiche:**

- Un particolare tipo di malware.
- Si attiva e si manifesta al raggiungimento di una condizione specifica (ad esempio, un certo numero di utenti collegati simultaneamente a una piattaforma).

# Trojan

## **Definizione:**

Un programma utile o apparentemente utile che, durante l'esecuzione, compie violazioni di sicurezza.

## **Caratteristiche:**

- **Numerosi Usi:**
  - Rinominare defrag.exe come format.exe.
  - Un utente che esegue clear potrebbe ritrovarsi tutti i propri file con permessi modificati (r--rwxrwx).
- **Metodi di Inserimento:**
  - Spesso inserito nel sistema tramite trapdoor, backdoor o tecniche di ingegneria sociale.
- **Funzionamento:**
  - Presenta un'interfaccia apparentemente innocua, ma contiene un payload dannoso.
  - Esempio: Creazione di un link simbolico tra ls e rm, ingannando l'utente a eseguire un comando dannoso.

# Zombie

## **Definizione:**

Programma nocivo che sfrutta una macchina remota già violata per lanciare nuovi attacchi, difficilmente riconducibili all'autore del malware.

## **Caratteristiche:**

- **Diffusione:**
  - Estremamente diffusi oggi su Internet.
- **Utilizzo:**
  - Tipicamente usati per attacchi di negazione del servizio (DoS).
- **Funzionamento:**
  - Una macchina violata viene identificata come zombie, così come il software che utilizza la macchina per lanciare attacchi.
  - La natura distribuita degli attacchi rende difficile tracciare l'entità del malware e il suo autore.

# Worm

## Definizione:

Programma nocivo che infetta macchine remote, ognuna delle quali infetta a sua volta altre macchine.

## Caratteristiche:

- **Trasmissione:**
  - Più dannoso di uno zombie perché si trasmette autonomamente da macchina a macchina.
  - Non richiede intervento umano per propagarsi.
- **Esempi:**
  - Il worm di Morris infettò 6000 macchine in poche ore.
  - Un virus che si propaga via email ha le caratteristiche di un worm.
- **Pericolosità:**
  - I worm sono pericolosi perché si diffondono rapidamente tra più dispositivi, aumentando esponenzialmente l'infezione.

# Worm Morris

## Contesto Storico:

Il worm Morris, anche conosciuto come worm Internet, è stato uno dei primi worm informatici a diffondersi su una vasta rete di macchine Unix. Il suo creatore, Robert Morris, è stato multato e condannato a servizi civili.

## Caratteristiche del Worm:

- **Attacco e Propagazione:**
  - Sfruttava tre vulnerabilità note in Unix per infettare macchine remote.
  - Utilizzava tecniche come l'attacco known-ciphertext su file di password, buffer overflow su programmi come finger e backdoor nel programma sendmail.
  - Una volta violata la macchina, il worm scaricava un codice C di 99 linee che si auto-compilava ed eseguiva, propagando così ulteriormente il worm.

### **Problemi e Impatto:**

- **Bug e Malfunzionamenti:**
  - A causa di bug nel worm, alcune versioni non terminavano correttamente e si replicavano continuamente sulla stessa macchina.
- **Impatto Sulle Macchine:**
  - Questo comportava un serio calo delle prestazioni delle macchine infettate.
- **Disconnessione dalla Rete:**
  - Migliaia di macchine sono state disconnesse dalla rete per proteggersi o per proteggere altre macchine.

### **Conclusioni:**

Il worm Morris è stato un esempio pionieristico di malware che ha dimostrato la vulnerabilità delle reti informatiche e l'importanza di implementare soluzioni di sicurezza robuste per prevenire tali attacchi.

## Attacco Known-Ciphertext

### **Definizione:**

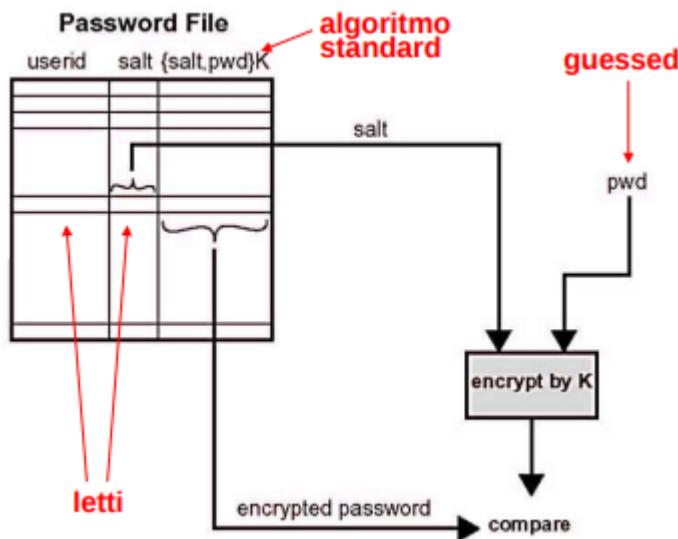
L'attacco known-ciphertext è un tipo di attacco crittografico che sfrutta informazioni note sul testo cifrato per ottenere informazioni sul testo in chiaro o sulla chiave di ciphertura.

### **Descrizione dell'Attacco:**

- **Vulnerabilità nel File di Password:**
  - Sebbene il file di password sia criptato, le sue tre colonne (contenenti nomi utente, hash delle password e altri dati) sono leggibili da tutti.
  - Questo è dovuto a un errore nell'implementazione della politica di sicurezza, che rende visibili informazioni sensibili.
- **Metodi Utilizzati da Morris:**
  - Morris ha utilizzato un attacco dizionario con una lista di parole note.
  - Prova a codificare possibili candidati, partendo dai nomi utente e loro permutazioni.
  - Se non ha successo, utilizza una lista di candidati statistici.
  - Se ancora senza successo, prova tutte le parole nella directory locale /local.
  - Se riesce ad ottenere una corrispondenza, può effettuare il login su una macchina remota per eseguire ulteriori azioni.

## Conclusioni:

Questo tipo di attacco sfrutta la debolezza nella gestione dei file di password, permettendo agli attaccanti di ottenere accesso non autorizzato alle risorse di sistema. L'attacco di Morris è stato uno dei primi esempi di utilizzo di questa tecnica per diffondere malware e compromettere sistemi Unix.



## Buffer Overflow mediante finger

### Descrizione della Vulnerabilità:

Il buffer overflow è una vulnerabilità che si verifica quando un programma scrive dati oltre i limiti di un buffer allocato in memoria, causando la sovrascrittura di dati adiacenti o addirittura del puntatore di ritorno della funzione.

### Scenario di Attacco:

- Il worm lanciava il comando finger, e il servizio fingerd rispondeva dalla macchina remota.
- La vulnerabilità si sfruttava tramite un buffer overflow nel buffer di input di fingerd, che andava oltre il suo limite e sovrascriveva l'indirizzo di ritorno.
- Prima di terminare, fingerd eseguiva la shellcode caricata nel buffer, la quale scaricava e eseguiva le 99 linee di codice del worm.

### Risultato dell'Attacco:

Questa vulnerabilità permetteva al worm di ottenere l'esecuzione remota di codice sulla macchina bersaglio, aprendo la porta per l'infezione e la propagazione del malware.

# Backdoor in Sendmail

## **Descrizione della Vulnerabilità:**

Il backdoor in Sendmail è una falla di sicurezza che permetteva di utilizzare Sendmail come un interprete di comandi, aprendo la porta per l'esecuzione remota di codice.

## **Scenario di Attacco:**

- Sendmail è un demone di sistema eseguito in background, in attesa di indirizzi email da gestire.
- Il backdoor consisteva nel far passare Sendmail in "modalità di debug", in cui non solo gestiva le email ma eseguiva anche comandi arbitrari.

## **Utilizzo da Parte di Morris:**

- Robert Morris sfruttò questa trapdoor presente in Sendmail su macchine remote.
- Utilizzando questa vulnerabilità, Morris era in grado di inviare comandi al Sendmail remoto, incluso il comando per scaricare e eseguire le 99 linee di codice del suo worm.

## **Risultato dell'Attacco:**

Questo backdoor ha permesso a Morris di ottenere un accesso non autorizzato e remoto alle macchine bersaglio, facilitando la propagazione del suo worm e causando un impatto significativo sulla sicurezza e sulla stabilità dei sistemi interessati.

# Virus

## Definizione:

Un virus è un programma nocivo che infetta altri programmi non nocivi, sfruttandoli per propagarsi.

## Modalità di Violazione:

- **Aggiunta di Codice Indesiderato:**
  - Tipicamente, un virus aggiunge pezzi di codice indesiderato ai programmi ospiti, compromettendoli.
- **Tipologie:**
  - Virus per Applicativi o per Dati.
  - Virus per il Settore di Boot.

## Caratteristiche:

- **Difficoltà di Individuazione:**
  - I virus possono essere difficili da individuare, specialmente se sono polimorfi (cambiano forma) o criptati (nascondono le proprie tracce).
- **Modifiche dannose:**
  - Modificando casualmente i bit di un'eseguibile, un virus può comprometterlo, rendendolo non funzionante.

## Identificazione e Riconoscimento:

- **Firma del Virus:**
  - Gli antivirus utilizzano le firme dei virus per riconoscerli. Queste firme devono essere aggiornate regolarmente per rilevare le nuove varianti.
- **Differenza dalla Firma Digitale:**
  - La firma del virus è diversa dalla firma digitale e viene utilizzata esclusivamente per il riconoscimento dei virus.

## Diffusione e Adattamento:

I virus possono essere ovunque e sono in grado di adattarsi, cambiando forma o firma per eludere la rilevazione.

# Macrovirus

## **Definizione:**

I macrovirus sono virus scritti come macro di un'altra applicazione, spesso utilizzati per infettare documenti di Microsoft Office.

## **Caratteristiche:**

- **Utilizzo delle Macro in MS Office:**
  - L'uso delle macro in MS Office può essere una vulnerabilità, in quanto le macro possono essere utilizzate per eseguire codice dannoso.
- **Livelli di Eseguibilità delle Macro:**
  - Le macro possono essere configurate per essere eseguite all'apertura del documento.
  - Inizialmente, i macrovirus possono infettare senza dare sintomi evidenti.

## **Rischio e Vulnerabilità:**

- **Nascita di Nuove Categorie di Virus:**
  - Le macro di Office hanno aperto la porta a una nuova categoria di virus che sfruttano le macro per propagarsi.
- **Filtri e Vulnerabilità:**
  - I filtri di sicurezza possono presentare vulnerabilità che gli attaccanti possono sfruttare per nascondere eseguibili maligni all'interno di documenti infetti.

# Software Nocivo – Distinzione Delicata

	<b>Tipo</b>	<b>Caratteristiche essenziali</b>
<b>Non replic.</b>	Trapdoor	Concede accesso non-autorizzato
<b>Replicano</b>	Bomba logica	Si attiva solo su specifiche condizioni
	Cavallo di troia	Ha funzionalità illecite inattese
	Virus	Attacca un programma e si propaga attraverso qualunque mezzo, anche fisico
	Zombie	Usa macchina già violata per attaccare
	Worm	Viola macchine “ricorsivamente” attraverso la rete

# Struttura di un semplice virus

## Struttura di un semplice virus

**Firma del virus:** codice usato per discernere file da infettare da quelli già infetti

Infetta un file ancora "pulito"

Esegue ulteriore danno – per es. una bomba logica

Restituisce il controllo al programma ospitante

```
program V :=  
    {goto main;  
     1234567;  
  
     subroutine infect-executable :=  
     {loop:  
      file := get-random-executable-file;  
      if (first-line-of-file = 1234567)  
          then goto loop  
      else prepend V to file; }  
  
     subroutine do-damage :=  
     {whatever damage is to be done}  
  
     subroutine trigger-pulled :=  
     {return true if some condition holds}  
  
     main:  
         main-program :=  
             {infect-executable;  
              if trigger-pulled then do-damage;  
              goto next;}  
  
     next:
```

### Firma del Virus:

- Una stringa usata per identificare il virus all'interno dei file.
- Di solito si trova all'inizio del file bersaglio.

### Funzionamento:

- **Riconoscimento della Firma:**
  - Il virus si riconosce tramite la sua firma.
  - Se la firma è presente nel file target (ad esempio, nella prima riga), il file viene considerato già infetto e viene saltato.
  - Altrimenti, il virus procede con l'infezione.
- **Infettare il File:**
  - Il virus infetta un file "pulito" aggiungendo la sua firma e il proprio codice al suo interno.
  - Esecuzione del Danno Aggiuntivo:
  - Dopo l'infezione, il virus può eseguire ulteriori danni al sistema, come attivare una bomba logica.
- **Restituire il Controllo:**
  - Infine, il virus restituisce il controllo al programma ospitante, permettendo al file infetto di funzionare come previsto, ma con il codice dannoso del virus aggiunto.

# Struttura di un virus che comprime

```
program CV :=  
  
1) Comprime il      {goto main;  
nuovo file da      01234567;  
infettare  
  
2) Appende il virus subroutine infect-executable :=  
all'inizio          {loop:  
                        file := get-random-executable-file;  
                        if (first-line-of-file = 01234567) then goto loop;  
  
3) Decomprime il      (1) compress file;  
resto del file       (2) prepend CV to file;  
ospite                }  
  
4) E lo esegue        main:  main-program :=  
                           {if ask-permission then infect-executable;  
                            (3)  uncompressed rest-of-file;  
                            (4)  run uncompressed file;}  
                           }
```

## Funzionamento del Virus:

- **Compressione del Nuovo File:**
  - Il virus comprime il nuovo file da infettare.
- **Inserimento del Virus:**
  - Appendendo il proprio codice, il virus infetto viene aggiunto all'inizio del file compresso.
- **Decompressione del File Ospite:**
  - Il virus decomprime il resto del file ospite dopo il suo codice.
- **Esecuzione del File:**
  - Il file viene eseguito con il virus attivo.

## Strategie di Rilevamento:

- **Dimensioni dei File:**
  - Il virus potrebbe essere rilevato controllando le dimensioni del file, tuttavia, se il virus comprime il file in modo che la dimensione del file compresso più il virus sia simile a quella del file originale, questa strategia di rilevamento potrebbe non essere efficace.

## **Nuove Strategie di Difesa:**

- È necessario sviluppare nuove strategie di difesa che vadano oltre il semplice controllo delle dimensioni dei file, ad esempio:
  - Analisi del comportamento del file per rilevare azioni sospette.
  - Analisi delle firme del virus all'interno del file compresso.
  - Utilizzo di tecniche di machine learning per identificare pattern di comportamento maligno.
  - Implementazione di controlli di integrità per verificare se il file è stato alterato.

## Esempio di virus: Brain

### **Caratteristiche di Base:**

Il virus Brain attacca sistemi Microsoft, rinominando il volume del disco in "BRAIN" e in alcune versioni cancellando frammenti del disco rigido. Si propaga naturalmente.

### **Carico del Virus:**

- Il virus viene caricato in memoria alta, ed esegue una chiamata di sistema per resettare il limite di memoria alta al di sotto dell'area che lo contiene.
- Modifica il vettore degli interrupt in modo che l'indirizzo della procedura per gestire le letture da disco punti al proprio codice. In questa prima versione, non causa danni.

### **Propagazione del Virus:**

- Gestendo le letture, il virus controlla se i byte 5° e 6° dei dati letti contengono la sua firma (valore esadecimale 1234).
  - Se la firma non è presente, infetta 6 settori casuali.
  - Se la firma è presente, si propaga con i dati.

### **Carico Maggiore del Virus:**

- Marca i settori infetti come "danneggiati" per impedire al sistema operativo di utilizzarli.
- Continua ad infettare settori finché ce ne sono di non infetti sul disco, riducendo lo spazio disponibile sul disco continuamente.

# Rimozione di Software Nocivo

## **1. Antivirus:**

- **Metodo Comune ed Evolutivo:**

- L'antivirus è il metodo più comune e in continua evoluzione per rimuovere software nocivo.
- Tuttavia, è limitato dalla necessità di aggiornamenti costanti per rilevare le nuove minacce.

## **2. Sistemi Immuni:**

- **Prototipo IBM degli Anni '90:**

- Sviluppato negli anni '90 da IBM, il prototipo dei sistemi immuni sfrutta tecniche simili a quelle degli antivirus.
- Si basa su un'intera struttura distribuita di prevenzione per rilevare e rimuovere il software nocivo.

## **3. Software Sentinella:**

- **Integrato con il Sistema Operativo:**

- Il software sentinella è integrato direttamente nel sistema operativo.
- Blocca l'esecuzione di codice nocivo e protegge il sistema da potenziali minacce.

## **Ruolo delle Difese:**

- L'antivirus rappresenta solo una delle difese disponibili contro i virus.
- Gli "Immune System" e i software sentinella sono altre aree importanti della difesa, lavorando in sinergia per proteggere i sistemi da software nocivi.
- Inoltre, i sistemi operativi stessi spesso incorporano meccanismi di sicurezza per controllare il codice eseguito e prevenire l'esecuzione di software nocivo.

# Struttura ideale di un antivirus

## **Scansione dei File:**

L'antivirus dovrebbe scansionare i file uno dopo l'altro, controllando la firma dei virus presenti nelle prime righe dei file. Nel caso dei virus polimorfi, si potrebbero utilizzare pattern noti per cercare firme in diverse parti del file.

## **Automazione della Scansione:**

Per rendere l'operazione meno onerosa per il sistema, l'antivirus dovrebbe automatizzare la scansione, decidendo autonomamente la periodicità e la profondità della scansione. Attualmente, molti antivirus sono residenti in memoria e utilizzano le capacità computazionali in modo efficiente.

## **Approccio alle Generazioni di Virus:**

- **Prima Generazione:** Confronto con un DB
- **Seconda Generazione:** Utilizza euristiche e strategie per riconoscere i virus.
- **Terza Generazione:** Ricerca azioni illecite all'interno dei file.
- **Quarta Generazione:** Utilizza un insieme di tecniche di controllo di accesso, inclusi sistemi di prevenzione delle intrusioni (IPS).

## **Rilevamento dei File:**

L'antivirus può percepire l'arrivo di un nuovo file tramite il cambiamento delle dimensioni dei file presenti nel sistema, dovuto all'inserimento di firme al loro interno. Questo può attivare la scansione dei nuovi file per rilevare eventuali minacce.

# Parte 10: WWW Security Protocols

## Protocollo di sicurezza per il web

Il WWW è uno specifico servizio per il web, questi protocolli si possono trovare dietro altri tipi di servizi S/MINE 3D-Secure ed SET

Il pagamento è un obiettivo funzionale sensibile dal punto di vista della sicurezza.

3D-Secure mette in sicurezza la transazione tra Customer e Vendor, il vendor invia la merce in modo (anche) digitale ed il pagamento arriverà all'interno dello stesso tunnel in quanto la voglio mettere in sicurezza. Dietro il 3D-Secure ci sta qualcosa di più rispetto a ciò che sta dietro TLS/SSL che costruiscono un tunnel sicuro tra 2 agenti. SSL è omologo ad IPSEC solo che sta ad un altro livello.

### **Come realizzare un tunnel sicuro**

TLS prevede segretezza, autenticazione ed integrità.

### **Come mettere in sicurezza un pagamento digitale**

Mettere in sicurezza il pagamento vuol dire che il vendor prima di elargire il servizio verifica il pagamento. Se non fa questo non è il tunnel di connessione sicuro ad essere il problema ma il customer stesso.

Il sistema di pagamento annota le transazioni effettuate Tra Client e Vendor c'è però un terzo agente che è la banca. Non è possibile quindi costruire un semplice tunnel tra Client e Vendor. Come strumenti di pagamento usiamo le carte di debito/credito che permetta al vendor di ricevere il denaro. Come strumento di annotazione del pagamento si usa il numero della carta di credito, il customer trasferisce questo numero al gestore del pagamento. Inizialmente non era così il numero veniva trasmesso direttamente al vendor senza alcun controllo aggiuntivo (batch processing) che successivamente tramite i numeri di carta di credito acquisiti andrà in banca a richiedere i soldi.

Facilmente attaccabile, al customer bastava inviare un numero di carta di credito fasullo ma ben formato e riceveva comunque il servizio richiesto.

Il pagamento digitale per essere sicuro deve avere una verifica sulla sensatezza del numero della carta di credito acquisito. 3D-SECURE ed SET sono più complessi di TLS/SSL perché mettono in sicurezza l'intero pagamento a differenza di TLS/SSL. SSL nasce come un prodotto proprietario TLS è la sua versione open-source, usa strumenti standard contrariamente a quanto faceva SSL.

# SSL

Protocollo di sicurezza più utilizzato in rete fra livello trasporto e livello applicazione  
Ma sia SSL che TLS possono essere implementati ad ambedue i livelli

$$\text{HTTPS} = \text{HTTP} + \text{SSL}$$

## Cosa SSL garantisce

- **Segretezza:** con l'uso di codifica simmetrica
- **Integrità:** con l'uso di codifica asimmetrica e hashing
- **Autenticazione (utente):** con l'uso di firma elettronica e certificazione per il server e, optionalmente per il client

## Come scoprire se uso SSL

- Identificato da HTTPS, FTPS, SMTSP.
- Viceversa, se SSL è abilitato, può essere usato indirizzando <https://...>
- Su una connessione HTTP, l'URL diventa automaticamente <https://...>
- Usa porta 443 invece di 80 come HTTP

## Come ottenere SSL

Pre-installato nei moderni browser, insieme a un insieme di certificati di RCA. Va abilitato sul server durante il processo di configurazione.

## HTTPS

Se faccio scansione dei pacchetti mi accorgo dell'utilizzo della porta 443 a differenza della porta 80 utilizzata da HTTP, cifra contenuti payload form cookie ed header HTTP. Lo strumento per la gestione per l'autenticazione sul web sono i cookie che gestiscono l'ID di sessione. Un particolare header HTTP è HSTS, introdotto in una versione recente di TLS per contrastare un particolare tipo di attacco.

## Attacco di stripping su SSL

Attacco di downgrade delle scelte di sicurezza, ad esempio l'utente richiede HTTPS l'attaccante riesce a strippeare ed indirizzare l'utente su HTTP (per ragioni di legacy il sito potrebbe non essere compatibile) Un altro esempio è l'utilizzo di un dominio squatted o fasullo e richiede possesso del DNS, requisiti dell'attacco più importanti e meno realistici. Lo stripping ad oggi è più difficile i browser implementano sempre più il supporto ad HSTS che è una strategia per risolvere questo problema. Per mitigare il rischio della seconda tipologia di SSL stripping le aziende acquistano domini simili al proprio, attacco di natura socio tecnologica, se l'utente fosse attento si accorgerebbe che in realtà sta accedendo ad un sito errato.

- **Version 1:** prevented by HSTS
  - User wants `https://www.securesite.com`
  - MitM downgrades response to  
`http://www.securesite.com`
- **Version 2:** not prevented by HSTS
  - User wants `https://www.securesite.com`
  - MitM downgrades response to  
`http://www.securesitee.com`

## SSL vs TLS

SSL (Secure Sockets Layer) è stato sviluppato nei primi anni '90. Netscape, un famoso browser dell'epoca, fu uno dei primi a implementare SSL. Nel 1995, venne introdotto SSL 3.0, ma il suo RFC (Request for Comments), una descrizione tecnica e specifica di un prodotto open source, venne pubblicato solo molti anni dopo. Originariamente, SSL era un prodotto proprietario. Nel 1999, con la nascita di TLS (Transport Layer Security), si cercò di standardizzare il protocollo SSL.

In sintesi, TLS è la versione standardizzata di SSL.

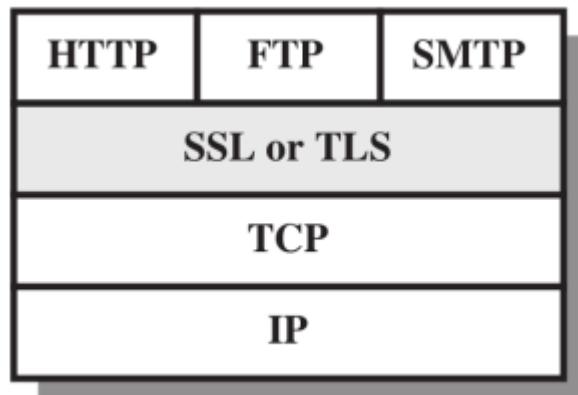
Netscape lancia SSL:

- **SSL 1.0:** violato durante la presentazione
- **SSL 2.0:** vulnerabile ad attacchi Man-in-the-Middle (MiM)
- **SSL 3.0:** versione più recente, introdotta a metà degli anni '90

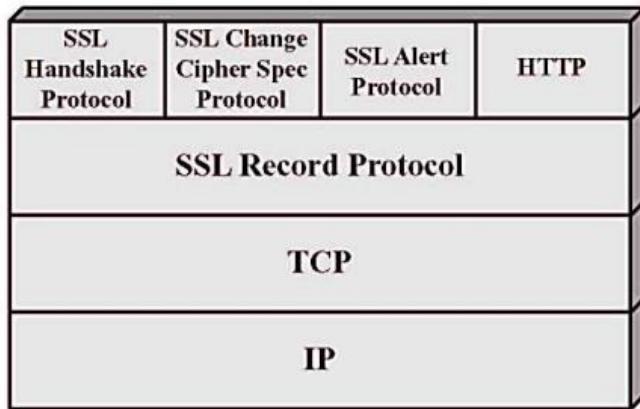
L'IETF (Internet Engineering Task Force) forma un gruppo di lavoro su TLS

TLS 1.0 può essere considerato come una sorta di SSL 3.1

Attualmente, TLS è uno standard Internet



## I protocolli in SSL

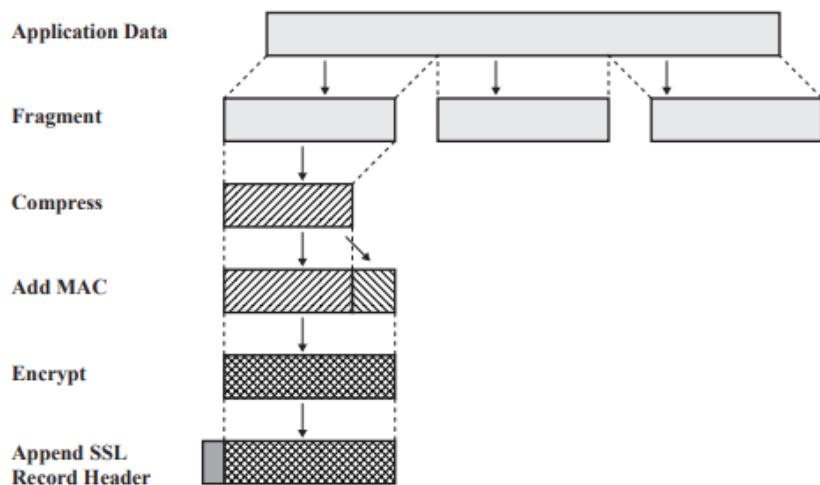


SSL non è un singolo protocollo, ma un insieme di protocolli:

- Protocollo di **Handshake**
- Protocollo per **cambiare le specifiche crittografiche**
- Protocollo di **Alert**
- Protocollo di **Record**

Questi protocolli operano a livelli diversi. Alla base di questi quattro protocolli c'è un protocollo principale di sicurezza che distribuisce la chiave condivisa: in IPSEC è chiamato IKE, mentre in SSL è il protocollo di Handshake. Il protocollo di Record utilizza la chiave per creare il substrato di sicurezza per tutte le applicazioni che utilizzano HTTPS, costruendo il tunnel crittografico. Questo protocollo impiega il materiale crittografico per garantire un livello di sicurezza adeguato all'applicazione.

## SSL Record Protocol

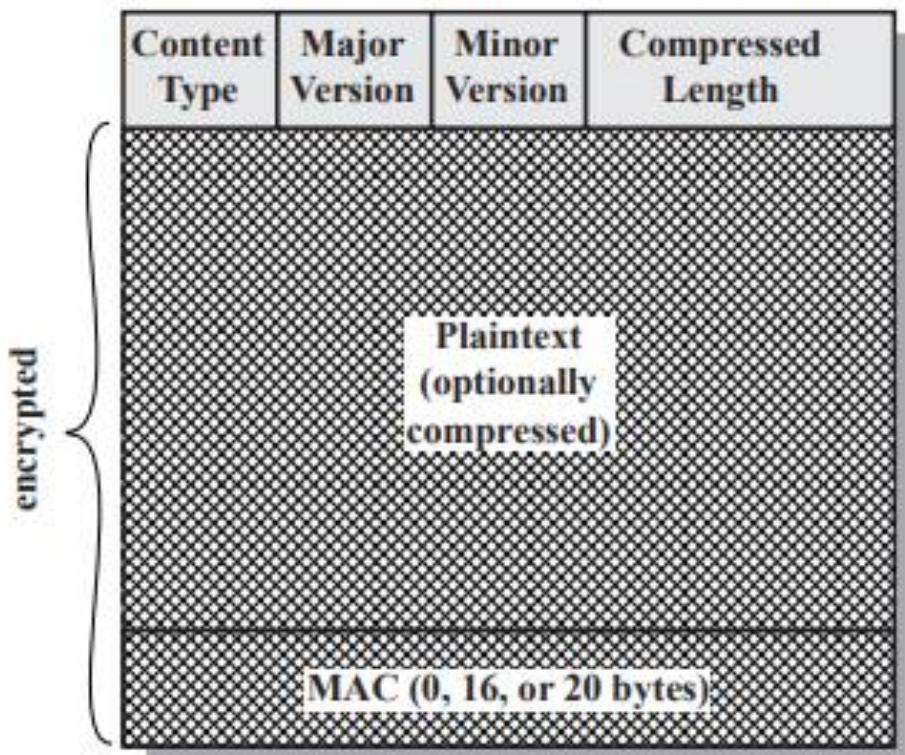


## Record Header

I dati dell'applicazione vengono frammentati e compressi, a cui viene aggiunto un MAC (Message Authentication Code), una primitiva crittografica che utilizza una chiave simmetrica per garantire autenticità e integrità. Il MAC può essere rappresentato come  $m,h(m,k)$  e funziona come una firma digitale quando la chiave è simmetrica. Una volta aggiunto il MAC, l'intero frammento viene cifrato. Infine, a ogni frammento viene aggiunto un header che contiene le seguenti informazioni:

- **Content Type:** Indica a quale protocollo si riferisce il frammento.
- **Major Version:** Numero di versione principale del protocollo.
- **Minor Version:** Numero di versione secondaria del protocollo.
- **Compressed Length:** Lunghezza del frammento compresso, utile per la successiva decompressione.

Content type	Major version	Minor version	Compressed length
--------------	---------------	---------------	-------------------



## MAC di SSL

L'indentazione evidenzia che si tratta di un hash annidato, il cui scopo era aumentare la difficoltà di inversione dell'hash. Non esistono tabelle rainbow per hash annidati.

- **MAC\_Write\_Secret:** Nome SSL per la chiave di sessione
- **SSLCompressed.fragment:** Input per il MAC
- **SSLCompressed.type:** Tipo di frammento SSL
- **pad\_1, pad\_2:** Padding, utilizzato per garantire l'applicabilità della funzione hash

Il processo prevede la concatenazione della chiave, l'aggiunta del padding e l'applicazione dell'hash. Questo ciclo viene ripetuto per aumentare la sicurezza.

```
hash(MAC_write_secret || pad_2 ||  
      hash(MAC_write_secret || pad_1 || seq_num ||  
            SSLCompressed.type || SSLCompressed.length ||  
            SSLCompressed.fragment))  
  
where  
|| = concatenation  
MAC_write_secret = shared secret key  
hash = cryptographic hash algorithm; either  
       MD5 or SHA-1  
pad_1 = the byte 0x36 (0011 0110) repeated  
        48 times (384 bits) for MD5 and 40  
        times (320 bits) for SHA-1  
pad_2 = the byte 0x5C (0101 1100) repeated 48  
        times for MD5 and 40 times for SHA-1  
seq_num = the sequence number for this message  
SSLCompressed.type = the higher-level protocol used to process  
                     this fragment  
SSLCompressed.length = the length of the compressed fragment  
SSLCompressed.fragment = the compressed fragment (if compression  
                         is not used, this is the plaintext fragment)
```

## SSL Change Cipher Spec Protocol

Implementa in modo semplice l'accordo tra client e server. Consiste in un messaggio con un singolo byte di valore 1. Considerato parte del protocollo SSL HP. Causa il salvataggio dello stato pendente come stato corrente. Aggiorna il campo Cipher Suite per la connessione attuale.

Attraverso un singolo byte, negoziamo le preferenze crittografiche con il protocollo di handshake e con questo "schiocco di dita" le mettiamo in atto, sincronizzandoci grazie a questo protocollo. Lo stato corrente, noto come Cipher Suite, rappresenta l'insieme delle scelte crittografiche. Le scelte crittografiche sono numerose, ma abbiamo più di una chiave di sessione: una per il MAC e una per la crittografia. Il protocollo di handshake distribuisce la chiave di sessione e stabilisce la Cipher Suite.

## SSL Alert Protocol

Il protocollo di alert viene implementato anche sopra SSL Record Protocol, garantendo così la protezione dei messaggi di alert. Gli alert, se modificati, possono essere sfruttati dagli hacker, quindi è importante metterli in sicurezza.

Ogni messaggio di alert è composto da **due byte**:

- Uno per il livello
- Uno per il codice.

Un alert di tipo **fatal** causa la chiusura della connessione:

- **Esempi:** unexpected\_message, bad\_record\_mac, decompression\_failure

Un alert di tipo **warning** riguarda la validità o la scadenza dei certificati:

- **Esempi:** unsupported\_cert, cert\_revoked, cert\_expired

In caso di alert fatal, il flag `is_resumable` viene impostato a 0, indicando che la connessione SSL non può essere ripristinata.

### **Connessione Nuova e Connessione Ripristinabile**

- **Connessione Nuova:** Richiede di ripetere tutto il protocollo dall'inizio, con un impatto significativo a livello computazionale.
- **Connessione Ripristinabile:** Viene effettuata per motivi di efficienza, riducendo la latenza e migliorando le prestazioni complessive.

Ripristinare una connessione è cruciale per mantenere un alto livello di efficienza e ridurre i tempi di latenza durante le comunicazioni sicure.

# SSL Handshake Protocol

Protocollo di sicurezza più complesso che garantisce le tre proprietà fondamentali della sicurezza: riservatezza (cifratura simmetrica), autenticazione (cifratura asimmetrica) e integrità (hashing). L'obiettivo principale è stabilire un Master Secret, ottenuto tramite l'algoritmo Diffie-Hellman, dal quale vengono derivati ulteriori segreti.

Il protocollo viene eseguito prima della trasmissione di qualsiasi dato dell'applicazione.

## **Formato del Messaggio:**

- **Tipo:** Numero che identifica uno dei dieci messaggi
- **Lunghezza:** Lunghezza in byte del messaggio
- **Contenuto:** Campi del messaggio (es. parametri)

Il protocollo di sicurezza garantisce che tutte le comunicazioni avvengano in modo sicuro, proteggendo i dati trasmessi attraverso l'uso combinato di tecniche crittografiche simmetriche e asimmetriche, nonché di algoritmi di hashing per mantenere l'integrità dei messaggi.

1 byte	3 bytes	$\geq 0$ bytes
Type	Length	Content

# Messaggi SSL

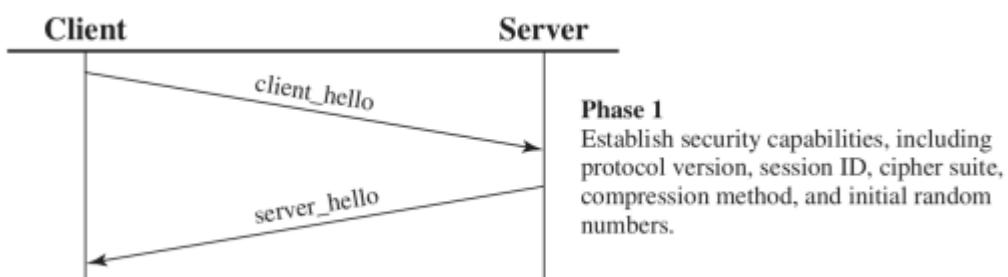
Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

I messaggi hello sono tra i più importanti nel protocollo di handshake e consentono di distribuire le chiavi tramite Diffie-Hellman o RSA.

- **hello\_request:** (S -> C), Messaggio dal contenuto nullo, facoltativo
- **client\_hello:** (C -> S), contiene versione, numero casuale, ID di sessione, cipher suite e metodo di compressione
- **server\_hello:** (S -> C), contiene gli stessi parametri del client\_hello
- **certificate:** Catena di certificati X.509, utilizzata per permettere al browser (client) di autenticare l'URL tramite crittografia asimmetrica
- **server\_key\_exchange:** Il server invia il primo messaggio di scambio della chiave di sessione, contenente parametri e firma del messaggio
- **certificate\_request:** Messaggio opzionale che il server invia al client per richiedere l'autenticazione del client
- **server\_done:** Il server segnala di aver terminato la fase di configurazione
- **certificate\_verify:** Il client verifica la firma del server
- **client\_key\_exchange:** Il client invia il secondo parametro di Diffie-Hellman
- **finished:** Valore di hash che conclude il protocollo di handshake

Questi messaggi garantiscono l'istituzione sicura di una connessione crittografata, facilitando lo scambio di chiavi e l'autenticazione reciproca tra client e server.

## Fase 1: Stabilimento della connessione



Fase cruciale del protocollo, si inviano il numero del protocollo, l'ID della sessione, cipher suite, metodo di compressione, e numeri random meglio conosciuti come nonce.

## Client Hello

Durante la fase di Client Hello, il client stabilisce le capacità di sicurezza, includendo la versione del protocollo, l'ID della sessione, la cipher suite, il metodo di compressione e i numeri casuali (nonce). Ecco i campi inviati dal client:

- **Version:** La versione del protocollo più alta supportata.
- **Random:** Un numero casuale (client\_hello\_random) per prevenire attacchi di replay.
- **Session ID:** Zero per una nuova sessione, oppure un ID di sessione per ripristinare una sessione precedente.
- **Cipher Suite:** Lista di coppie del tipo Cipher Spec (tipi di cifratura o hashing) e algoritmo di scambio chiavi.
- **Compression Method:** Lista dei metodi di compressione supportati.

## Server Hello

Durante la fase di Server Hello, il server conferma o rifiuta le proposte del client, inviando i seguenti campi:

- **Version:** La versione del protocollo più bassa tra quella suggerita dal client e la più alta supportata dal server.
- **Random:** Un numero casuale (server\_hello\_random) per prevenire attacchi di replay.
- **Session ID:** Se il client ha inviato un ID di sessione non zero, il server verifica la presenza di tale ID e può reinviarlo per ripristinare la sessione o generare un nuovo ID per una nuova sessione.
- **Cipher Suite:** La specifica di cifratura e l'algoritmo di scambio chiavi scelti dal server.
- **Compression Method:** Il singolo metodo di compressione scelto.

# Cipher Suite

Una cipher suite è composta da due elementi principali: specifiche di cifratura e algoritmi o protocolli di scambio delle chiavi.

## Specifiche di Cifratura

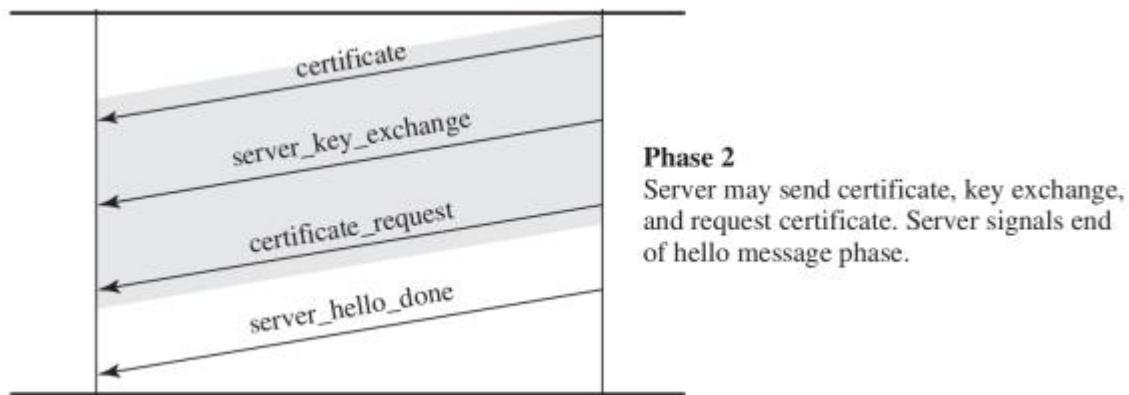
- **Algoritmo di Encryption:** L'algoritmo utilizzato per la cifratura dei dati.
- **Algoritmo MAC:** L'algoritmo usato per il Message Authentication Code, che include una funzione hash.
- **Tipo di Cifratura:** Può essere a flusso (stream) o a blocchi (block), cioè bit a bit o in chunk.
- **Is Exportable:** Un flag che indica se la cifratura è esportabile dagli Stati Uniti.
- **Dimensione dell'Hash:** Può essere di 16 byte per MD5 o 20 byte per SHA-1.
- **Materiale della Chiave:** Una sequenza di byte utilizzata per generare altre chiavi, fungendo da generatore per nuovo materiale crittografico.
- **Initialization Vector (IV):** Il vettore di inizializzazione utilizzato nel crittosistema CBC (Cipher Block Chaining).

## Protocollo di Scambio delle Chiavi

- **Scambio delle Chiavi RSA:** La chiave di sessione viene cifrata con la chiave pubblica del server.
- **Diffie-Hellman Anonimo:** Versione tradizionale del protocollo, vulnerabile agli attacchi man-in-the-middle (MITM).
- **Diffie-Hellman Effimero:** Versione in cui i parametri pubblici sono autenticati tramite firma digitale, migliorando la sicurezza.
- **Diffie-Hellman con Parametri Fissi:** I parametri pubblici sono fissati e derivati computazionalmente dai certificati di chiave pubblica del client e del server.
- **Forteza:** Un protocollo ora deprecato.

CipherSuite	Key Exchange	Cipher	Hash
<i>SSL_NULL_WITH_NULL_NULL</i>	NULL	NULL	NULL
<i>SSL_RSA_WITH_NULL_MD5</i>	RSA	NULL	MD5
<i>SSL_RSA_WITH_NULL_SHA</i>	RSA	NULL	SHA
<i>SSL_RSA_EXPORT_WITH_RC4_40_MD5</i>	RSA_EXPORT	RC4_40	MD5
<i>SSL_RSA_WITH_RC4_128</i>	RSA	RC4_128	MD5
<i>SSL_RSA_WITH_RC4_128_SHA</i>	RSA	RC4_128	SHA
<i>SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5</i>	RSA_EXPORT	RC2_CBC_40	MD5
<i>SSL_RSA_WITH_IDEA_CBC_SHA</i>	RSA	IDEA_CBC	SHA
<i>SSL_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	RSA_EXPORT	DES40_CBC	SHA
<i>SSL_RSA_WITH_DES_CBC_SHA</i>	RSA	DES_CBC	SHA
<i>SSL_RSA_WITH_3DES_EDE_CBC_SHA</i>	RSA	3DES_EDE_CBC	SHA
<i>SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA</i>	DH_DSS_EXPORT	DES40_CBC	SHA
<i>SSL_DH_DSS_WITH_DES_CBC_SHA</i>	DH_DSS	DES_CBC	SHA
<i>SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA</i>	DH_DSS	3DES_EDE_CBC	SHA
<i>SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	DH_RSA_EXPORT	DES40_CBC	SHA
<i>SSL_DH_RSA_WITH_DES_CBC_SHA</i>	DH_RSA	DES_CBC	SHA
<i>SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA</i>	DH_RSA	3DES_EDE_CBC	SHA
<i>SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA</i>	DHE_DSS_EXPORT	DES40_CBC	SHA
<i>SSL_DHE_DSS_WITH_DES_CBC_SHA</i>	DHE_DSS	DES_CBC	SHA
<i>SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA</i>	DHE_DSS	3DES_EDE_CBC	SHA
<i>SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	DHE_RSA_EXPORT	DES40_CBC	SHA
<i>SSL_DHE_RSA_WITH_DES_CBC_SHA</i>	DHE_RSA	DES_CBC	SHA
<i>SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA</i>	DHE_RSA	3DES_EDE_CBC	SHA
<i>SSL_DH_anon_EXPORT_WITH_RC4_40_MD5</i>	DH_anon_EXPORT	RC4_40	MD5
<i>SSL_DH_anon_WITH_RC4_128</i>	DH_anon	RC4_128	MD5
<i>SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA</i>	DH_anon	DES40_CBC	SHA
<i>SSL_DH_anon_WITH_DES_CBC_SHA</i>	DH_anon	DES_CBC	SHA
<i>SSL_DH_anon_WITH_3DES_EDE_CBC_SHA</i>	DH_anon	3DES_EDE_CBC	SHA

## Fase 2: Autenticazione del server e scambio delle chiavi



Durante questa fase, il server autentica sé stesso e inizia lo scambio delle chiavi. I messaggi principali inviati dal server includono il certificato, lo scambio delle chiavi e la richiesta di certificato al client. La fase si conclude con il messaggio di "Server Hello Done", che segnala la fine della fase dei messaggi hello.

### Messaggi Inviati dal Server:

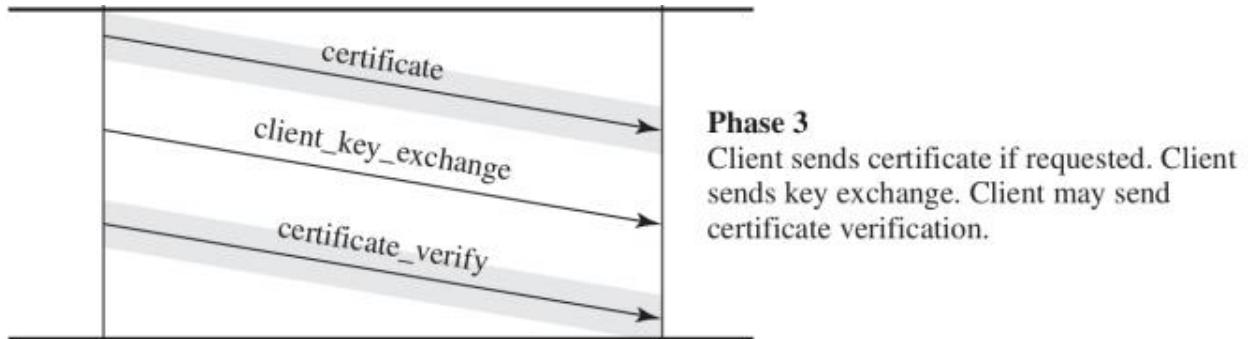
- **Certificate:** Invia la catena di certificati X.509 per l'autenticazione.
- **Server Key Exchange:** Invia metà delle informazioni necessarie per l'algoritmo di scambio delle chiavi.
- **Certificate Request:** Richiede al client di autenticarsi (opzionale).
- **Server Hello Done:** Informa il client che i parametri inviati sono accettati e la fase dei messaggi hello è conclusa.

### Firme Digitali

Le firme digitali sono utilizzate per garantire l'integrità e l'autenticità dei messaggi. Trasportano i campi Random per garantire la freschezza delle comunicazioni e prevenire attacchi di replay. I metodi di firma includono:

- **DSS (Digital Signature Standard):** Utilizza SHA-1 per creare la firma digitale.
- **RSA:** Utilizza una concatenazione di MD5 e SHA-1 per creare la firma digitale.

## Fase 3: Autenticazione del client e scambio delle chiavi



Durante questa fase, il client procede con l'autenticazione e lo scambio delle chiavi. Ecco i passaggi principali:

### Messaggi Inviati dal Client:

- **Certificate:** Il client invia i propri certificati se richiesto dal server.
- **Client Key Exchange:** Il client invia il Pre-Master Secret, una sequenza di 48 byte.
- **Certificate Verify:** Il client firma tutto il traffico visto finora, chiedendo al server di verificare esplicitamente la firma del client.

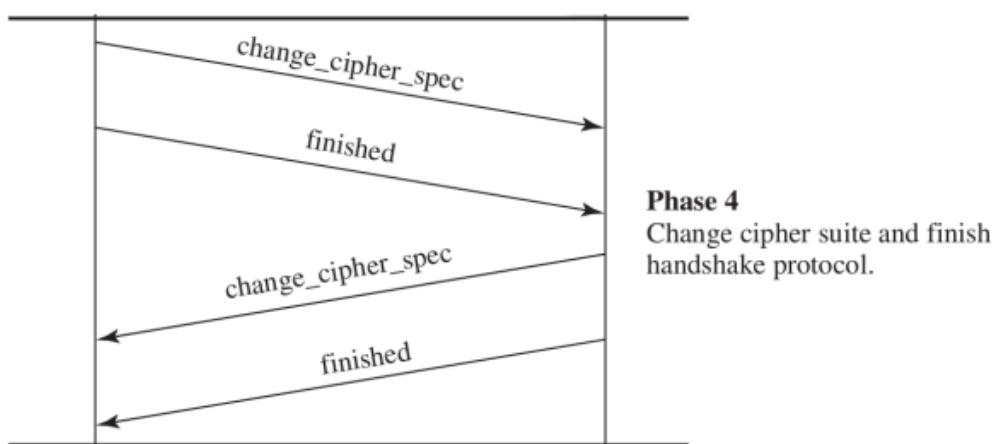
### Dettagli dei Passaggi:

- **Verifica dei Certificati:** I certificati inviati dal client vengono verificati tramite un digest che garantisce l'integrità e l'autenticazione, supportato dalla firma digitale.
- **Pre-Master Secret:** Conosciuto anche come "seme pre-master", questa chiave di sessione è calcolata e utilizzata per derivare ulteriori segreti crittografici. Il primo segreto calcolato sarà il seme per i calcoli successivi.
- **Master Secret:** Il Master Secret è determinato computazionalmente dal Pre-Master Secret. Questo passaggio assicura che il client e il server possano generare in modo indipendente lo stesso materiale crittografico per la sessione.

## Utilizzo dei PRF (Pseudo Random Function)

Come nell'OTP (One-Time Password), i PRF (Pseudo Random Function) sono funzioni basate sugli hash che generano chiavi pseudo-randomiche. Queste funzioni sono essenziali per garantire la sicurezza delle chiavi generate durante lo scambio di chiavi crittografiche. Utilizzando un calcolo offline, si ottiene lo stesso materiale crittografico, garantendo che le chiavi generate siano sicure e prevedibili quanto le chiavi iniziali. Questo processo assicura che la comunicazione tra client e server avvenga in modo sicuro e che entrambe le parti possano confermare l'autenticità reciproca prima di procedere con la fase successiva della connessione sicura.

## Fase 4: Fine



- **Change Cipher Spec:** È un singolo byte (valore 1) che segnala l'inizio dell'uso della cipher suite negoziata. Questo passaggio determina quale suite crittografica verrà utilizzata per la comunicazione.
- **Finished Messages:** Sono messaggi finali scambiati tra client e server. Contengono un digest crittografico del traffico scambiato utilizzando la nuova cipher suite. Questo digest serve a verificare che entrambi i partecipanti siano in grado di crittografare e decrittografare correttamente i dati utilizzando la stessa suite crittografica.

Tutti i passaggi dell'handshake, inclusi il cambio della cipher suite e lo scambio dei messaggi "Finished", sono protetti dal protocollo di record SSL/TLS. Questo protocollo utilizza la suite crittografica concordata per cifrare i dati e garantire la sicurezza (confidenzialità, integrità, autenticità) della comunicazione tra client e server.

## Master Secret

```
master_secret = MD5(pre_master_secret || SHA('A' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('BB' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('CCC' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random))
```

Il processo riguarda la generazione di chiavi crittografiche utilizzando le funzioni hash MD5 e SHA. In questo contesto, il pre-master secret (PMS) funge da seme, mentre i valori casuali e le stringhe specifiche sono utilizzati come sale. La chiave finale, di dimensioni pari a 48 byte, è ottenuta tramite tre applicazioni consecutive della funzione MD5. In dettaglio, MD5 prende come parametri il PMS concatenato con l'output di una funzione SHA interna. Quest'ultima utilizza come input il PMS, due numeri casuali (nonce) e una stringa inizialmente impostata su 'A'. Successivamente, la stringa è modificata in 'BB' e infine in 'CCC', permettendo di ottenere ulteriori 16 byte di output per ciascuna iterazione. L'obiettivo di questo processo è quello di generare segreti aggiuntivi a partire dal primo segreto condiviso, il PMS. L'uso del PMS come seme, combinato con valori casuali e stringhe come sale, assicura la diversificazione e la sicurezza della chiave crittografica risultante.

## Key Blocks

```
key_block = MD5(master_secret || SHA('A' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('BB' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('CCC' || master_secret ||
    ServerHello.random || ClientHello.random)) || ...
```

Il processo di generazione dei blocchi di chiavi continua su entrambe le estremità fino alla costruzione della Write Key e del Write MAC Secret. Seguendo un procedimento simile a quello precedentemente descritto, il parametro del pre-master secret è sostituito dal master secret. In questo caso, il master secret viene utilizzato come seme, condito con le nonce per garantire la freschezza, insieme al meccanismo delle stringhe. Il master secret, combinato con i valori casuali e le stringhe, viene sottoposto a iterazioni continue del processo descritto, generando ulteriori bit necessari per la costruzione delle chiavi di scrittura e dei segreti MAC di scrittura. Questo processo può continuare indefinitamente o fino a quando non si raggiunge il numero di bit richiesti per le operazioni crittografiche necessarie.

## Ridurre la latenza di rete

Ridurre la latenza di rete è un aspetto cruciale, poiché non si possono sprecare millisecondi. Una sessione rappresenta un'associazione tra un client e un server, creata tramite SSL/TLS, che definisce i parametri crittografici da utilizzare su più connessioni per migliorare l'efficienza. Una connessione, invece, è un trasporto secondo il modello OSI, peer-to-peer e transitorio, associato a una singola sessione.

L'idea è di salvare lo stato di una sessione e riprenderla su una nuova connessione utilizzando materiale pre-concordato. Lo stato della sessione include:

- **ID della Sessione:** scelto arbitrariamente dal server.
- **Certificato del Peer:** certificato X509.3.0 del peer.
- **Metodo di Compressione:** specifica dell'algoritmo di compressione.
- **Specifiche di Cifratura:** algoritmi di crittografia dei dati e MAC.
- **Master Secret:** segreto condiviso di 48 byte.
- **Is Resumable:** flag che indica se la sessione è ripristinabile.

Il ripristino della sessione avviene salvando lo stato della sessione alla fine della Fase 4 del protocollo SSL/TLS. Durante una nuova connessione, il client propone l'ID della sessione da riprendere nel messaggio "client hello". Il server verifica nel proprio database degli ID delle sessioni e decide se riprendere la sessione. Se la sessione viene ripresa, si saltano le Fasi 2 e 3, eseguendo solo le Fasi 1 e 4.

Durante il ripristino della sessione, si calcolano nuovi blocchi di chiavi partendo dal Master Secret salvato nello stato della sessione, ma utilizzando nuovi valori casuali (nonce) per garantire la freschezza. Questo meccanismo di ripristino è utilizzato per motivi di efficienza: permette di riutilizzare la suite crittografica di una sessione precedente, riducendo così il tempo necessario per stabilire nuove connessioni.

In sintesi, il ripristino della sessione consiste nel riutilizzare il Master Secret con nuove nonce per generare nuovi blocchi di chiavi, mantenendo però la sicurezza e l'efficienza della comunicazione crittografica.

## TLS

TLS (Transport Layer Security) rappresenta un tentativo di standardizzare SSL, un protocollo nato inizialmente come iniziativa commerciale privata e che oggi ha oltre 25 anni di storia. TLS è stato introdotto come la versione standardizzata di SSL per migliorare e modernizzare le pratiche di sicurezza nelle comunicazioni su Internet.

Una delle principali caratteristiche di TLS è l'uso del MAC (Message Authentication Code) tramite lo standard HMAC (Hash-based Message Authentication Code). HMAC è un meccanismo di autenticazione dei messaggi basato su hash che si è evoluto parallelamente allo sviluppo di TLS.

La prima versione di TLS, versione 1.0, è stata pubblicata nel 1999 come RFC 2246. Questa versione è sostanzialmente una versione standardizzata di SSL 3.0, con alcune differenze e miglioramenti:

- **HMAC:** Utilizzo di HMAC per l'integrità e l'autenticità dei messaggi.
- **Pseudo Random Function (PRF):** Introduzione di una funzione pseudocasuale per migliorare la sicurezza delle chiavi.
- **Cipher Suites Aggiuntive:** Inclusione di suite di cifratura aggiuntive per offrire più opzioni di crittografia.
- **Avvisi Aggiuntivi:** Introduzione di nuovi avvisi per gestire meglio gli errori e le condizioni di sicurezza.

In sintesi, TLS è un'evoluzione di SSL che introduce standard moderni per garantire la sicurezza delle comunicazioni, con particolare attenzione all'integrità e all'autenticità tramite HMAC, miglioramenti nelle funzioni di generazione delle chiavi e una maggiore varietà di suite di cifratura.

# HMAC

$$\text{HMAC}_K(M) = \text{H} [ (K^+ \oplus \text{opad}) \parallel \text{H} [ (K^+ \oplus \text{ipad}) \parallel M ] ]$$

where

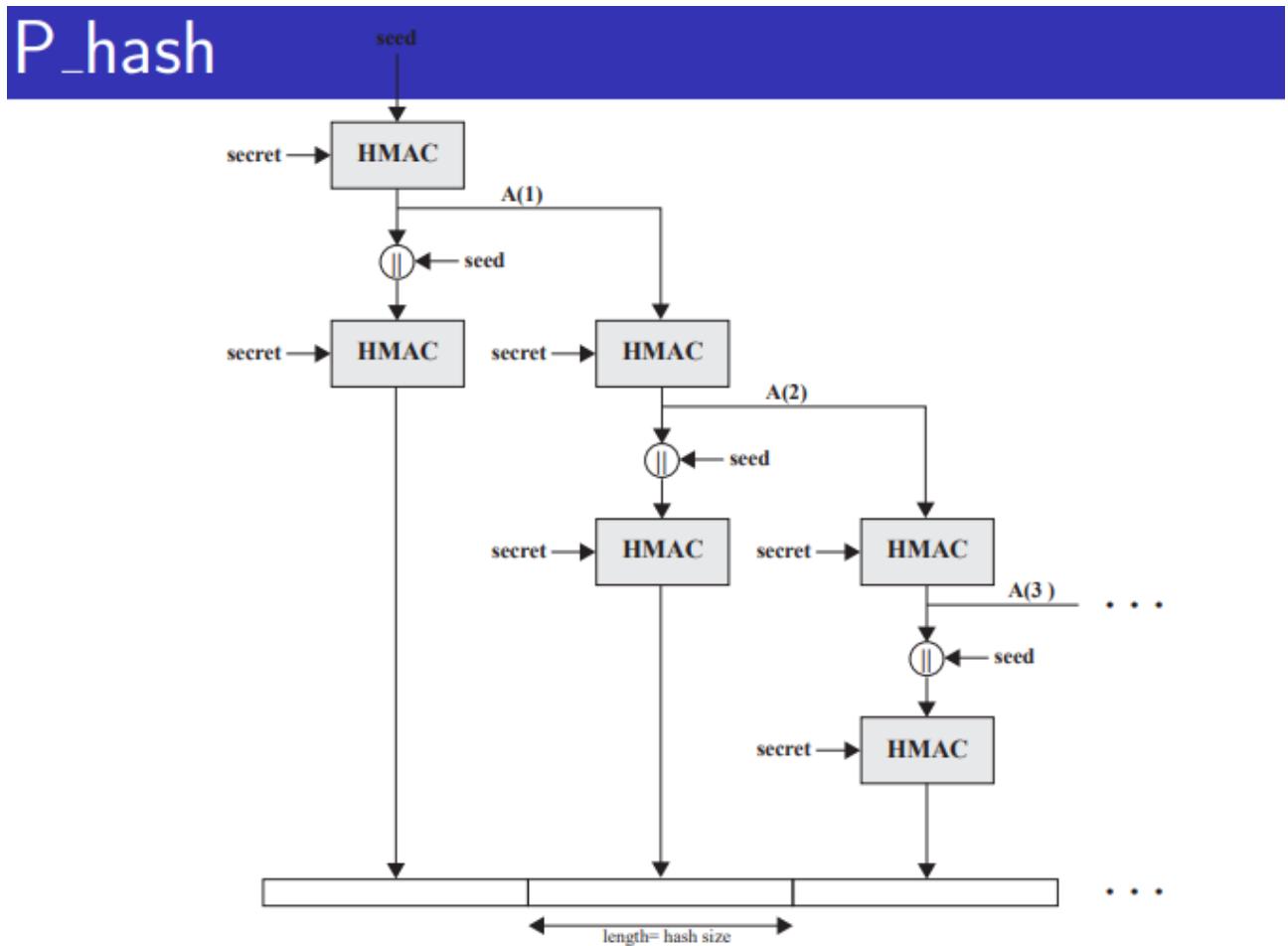
- $H$  = embedded hash function (for **TLS**, either MD5 or SHA-1)
- $M$  = message input to HMAC
- $K^+$  = secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)
- ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)
- opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

HMAC (Hash-based Message Authentication Code) utilizza due parametri principali: un messaggio  $M$  e una chiave  $K$ . La chiave  $K$  viene elaborata in modo specifico per adattarsi alla lunghezza del blocco della funzione hash scelta.

- 1. Preparazione della Chiave:** La chiave  $K$  viene allungata con zeri a sinistra fino a raggiungere la lunghezza del blocco richiesto dalla funzione hash.
- 2. Concatenazione e Padding:** La chiave allungata viene combinata tramite XOR con due diversi padding di bit. Questo processo crea due valori intermedi.
- 3. Annidamento delle Funzioni Hash:** Viene applicata una funzione hash specifica  $H$  ai valori concatenati e pad-dati, seguendo il metodo definito nello standard HMAC (RFC 2104).

HMAC utilizza quindi  $M$  come seme e  $K$  come segreto, applicando una funzione hash scelta  $H$  per garantire l'integrità e l'autenticità del messaggio.

## P\_HASH



P\_HASH è una rappresentazione grafica del key block, dove l'applicazione interna utilizza un seme e un segreto per creare un HMAC. Questo HMAC viene quindi concatenato al seme originale per formare un nuovo input per un altro HMAC.

Questo processo è simile al calcolo del key block in TLS, dove l'uso di HMAC per combinare seed e segreto assicura la generazione di dati crittograficamente sicuri e variabili, essenziali per la sicurezza e l'integrità delle comunicazioni.

## Pseudo Random Function

La Pseudo Random Function (PRF) è una funzione che genera output pseudo-casuali controllati tramite iterazioni HMAC, utilizzando un segreto, un'etichetta (label) e un seme concatenato di valori casuali.

Nel dettaglio, la formula della PRF è la seguente:

$$\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P\_MD5}(S1, \text{label} \parallel \text{seed}) \oplus \\ \text{P\_SHA-1}(S2, \text{label} \parallel \text{seed})$$

Dove:

- **secret** è il segreto diviso in due parti: S1 e S2.
- **label** specifica se l'output è il "master secret" o il "key block".
- **seed** è la concatenazione di due valori casuali.

Il processo operativo comprende:

1. **Divisione del Segreto:** Il segreto viene suddiviso in S1 e S2.
2. **P\_MD5 e P\_SHA\_1:**
  - a. P\_MD5 utilizza S1 come segreto e calcola un HMAC con label || seed utilizzando MD5.
  - b. P\_SHA\_1 utilizza S2 come segreto e calcola un HMAC con label || seed utilizzando SHA-1.
3. **Operazione di XOR:** I risultati di P\_MD5 e P\_SHA\_1 vengono combinati tramite XOR per produrre l'output finale della PRF.

La PRF è progettata per generare rapidamente nuove stringhe di bit, garantendo freschezza e sicurezza. Utilizza un segreto iniziale per arricchire un seme di numeri casuali, fondamentale per la creazione di nuovo materiale crittografico o per la derivazione di segreti come il master secret nei protocolli di sicurezza come TLS.

## Evoluzione di TLS

TLS ha subito un'evoluzione significativa nel corso degli anni per migliorare la sicurezza e l'affidabilità delle comunicazioni su Internet. Introdotta nel 1999 con TLS 1.0, questa versione iniziale ha rappresentato una standardizzazione dell'originale SSL 3.0, mirando a fornire un protocollo crittografico robusto per la protezione delle informazioni trasmesse online.

Successivamente, nel 2006, TLS 1.1 ha introdotto il concetto di estensioni, permettendo l'aggiunta di nuovi parametri ai messaggi di negoziazione ("hello") per migliorare ed estendere le funzionalità del protocollo.

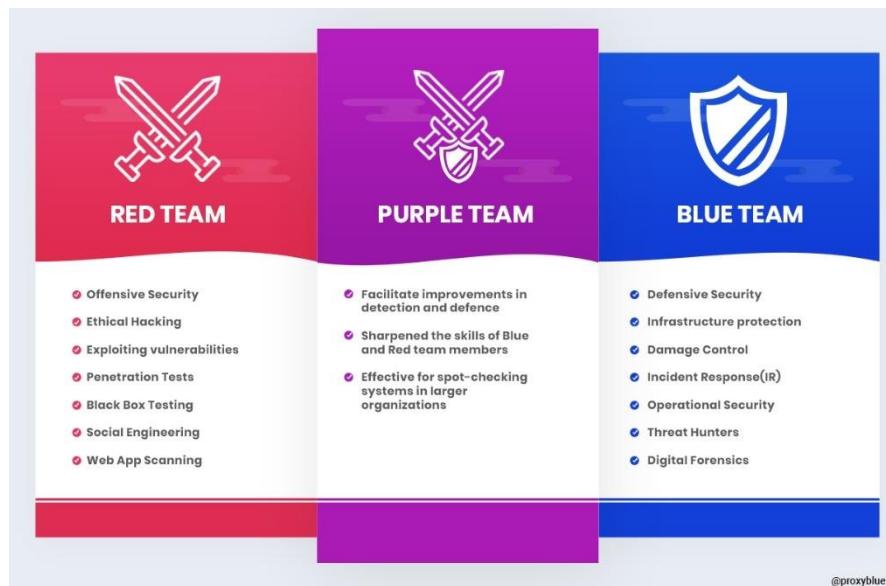
Nel 2008 è stata rilasciata TLS 1.2, che ha rappresentato una revisione significativa rispetto alla versione precedente. Questa versione ha affrontato le vulnerabilità scoperte in MD5 e SHA-1, optando per l'uso di algoritmi più sicuri come SHA-256. Inoltre, TLS 1.2 ha migliorato la capacità di client e server di specificare gli algoritmi crittografici accettati, riducendo così le vulnerabilità e migliorando la sicurezza complessiva del protocollo.

Negli anni successivi, vari attacchi contro le implementazioni di TLS hanno evidenziato la necessità continua di migliorare la sicurezza. Nel 2018, TLS 1.3 è stato approvato come standard Internet, introducendo ulteriori miglioramenti per ridurre la latenza, migliorare le prestazioni e rafforzare la sicurezza delle comunicazioni online. TLS 1.3 ha deprecato protocolli e algoritmi crittografici più deboli, continuando ad adattarsi alle nuove minacce e alle esigenze crescenti di sicurezza informatica.

In sintesi, l'evoluzione di TLS riflette un impegno costante nel migliorare la sicurezza delle comunicazioni su Internet, adottando tecnologie avanzate e rispondendo alle sfide emergenti nel panorama della sicurezza informatica.

# Parte 11 (Lab): Difesa della Rete

## Red, Purple, Blue Team



Il **Red Team** si occupa di sicurezza offensiva, hacking etico, sfruttamento delle vulnerabilità, test di penetrazione, black box testing, ingegneria sociale e scansione delle applicazioni web.

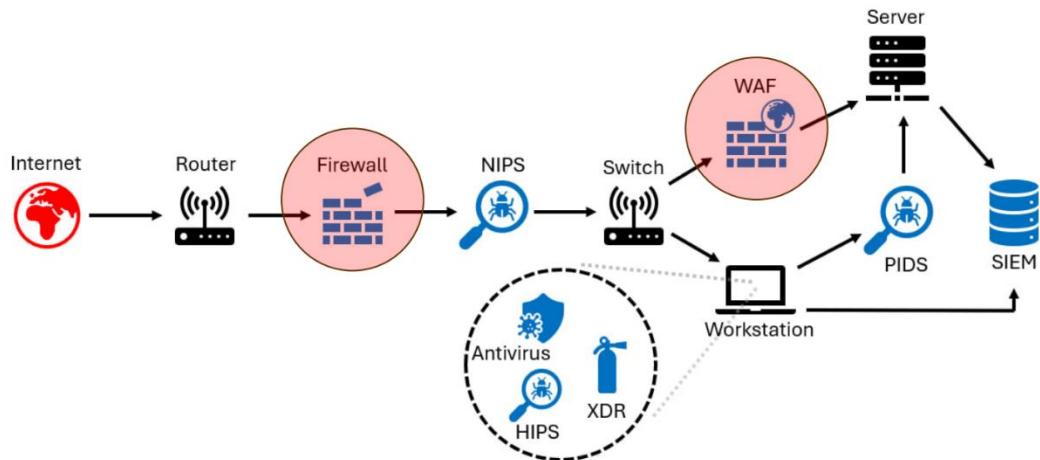
Il **Purple Team** facilita i miglioramenti nella rilevazione e nella difesa, affinando le competenze dei membri dei team Blue e Red. È particolarmente efficace per controlli a campione nei sistemi di grandi organizzazioni.

Il **Blue Team** si occupa di sicurezza difensiva, protezione dell'infrastruttura, gestione dei danni, risposta agli incidenti, sicurezza operativa, caccia alle minacce e analisi forense digitale.

## Intrusione

**Definizione:** Ottenere illecitamente privilegi superiori a quelli posseduti illecitamente.

## Linee di difesa tipiche di una rete



## Firewall

Il firewall era originariamente concepito come un "dispositivo che controlla il flusso di traffico tra reti con diverse impostazioni di sicurezza," ad esempio:

- Tra una LAN e la rete Internet
- Tra due sottoreti interne
- Tra una rete trusted e una untrusted

I firewall odierni si sono evoluti da questa concezione tradizionale.

### Requisiti del Firewall

- **Controllo totale del traffico:** Tutto il traffico tra reti deve passare attraverso il firewall.
- **Regole di sicurezza:** Ogni traffico deve essere regolato da specifiche regole di sicurezza.
- **Regole per gruppi:** Ad esempio, "ammetti tutti i pacchetti TCP e UDP in entrata e uscita" copre diversi tipi di pacchetti.
- **Configurazione sicura:** Errori nella configurazione possono compromettere la sicurezza.
- **Best practices:** Devono essere seguite per una gestione sicura del firewall.
- **Vulnerabilità:** Il firewall stesso può avere vulnerabilità da monitorare e mitigare.

I firewall moderni sono più avanzati ma mantengono la funzione di base di controllo del traffico tra reti diverse.

## Vulnerabilità

**Definizione:** Violazione di una delle tre proprietà fondamentali della sicurezza (Confidenzialità, Integrità, Disponibilità) all'interno di un dato software o hardware.

Una vulnerabilità non (ancora?) pubblica e non patchata viene chiamata 0-day.

## Debolezza (Weakness)

**Definizione:** Problema di sicurezza teorico, non collegato ad uno specifico software o hardware.

## Exploit

**Definizione:** Una qualsiasi risorsa che sfrutta una vulnerabilità.

Se utilizzato per sfruttare una vulnerabilità 0-day, viene chiamato exploit 0-day. Può avere pressoché qualsiasi forma, come eseguibile, stringa, immagine, audio, pagina web, archivio, codice, e molto altro ancora.

## Firewall: Funzionalità principali e secondarie

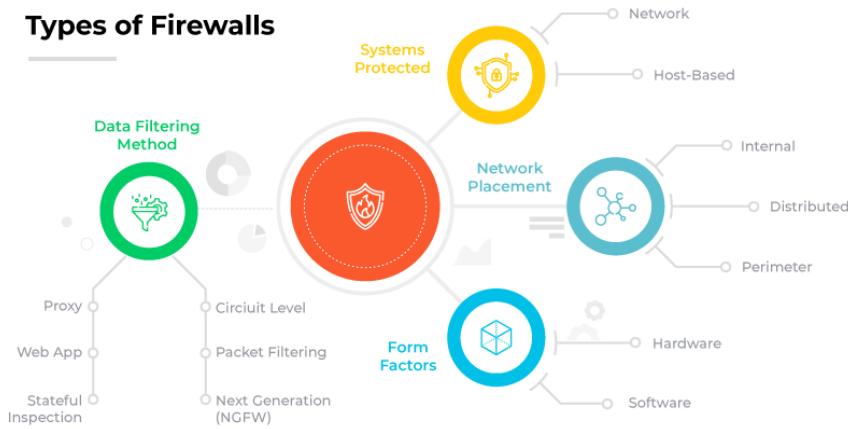
### **Funzionalità Principali**

- **Protezione dei servizi:** L'amministratore può esporre alcune porte solo a utenti di una determinata rete.
- **Monitoraggio e filtraggio del traffico:** L'amministratore può filtrare il traffico anche su porte aperte, analizzando il contenuto dei pacchetti.
- **Filtraggio dei dati:** Ad esempio, bloccare allegati email specifici.

### **Funzionalità Secondarie**

- **Creazione di VPN:** Utilizzando IPSec in modalità tunnel.
- **Fornitura di indirizzi IP:** Il firewall può operare come server o client DHCP (Dynamic Host Configuration Protocol).
- **Mappatura di indirizzi locali in indirizzi Internet:** Utilizzando NAT (Network Address Translator).

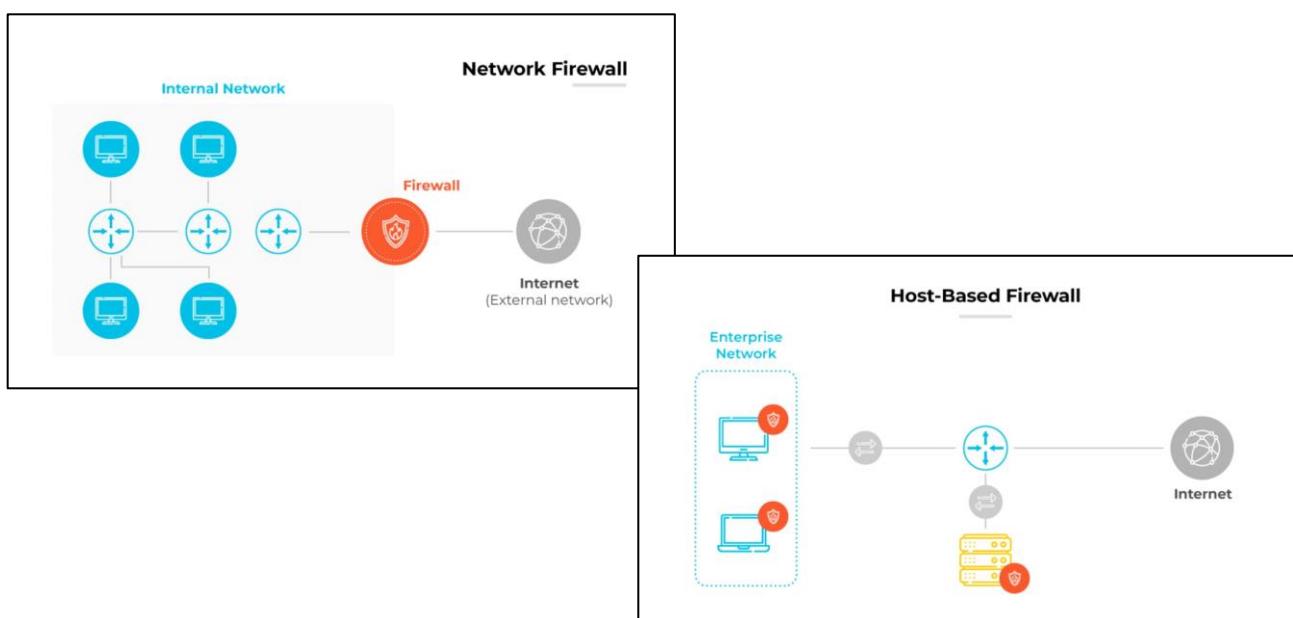
# Tipologie di Firewall



## Systems Protected Firewall

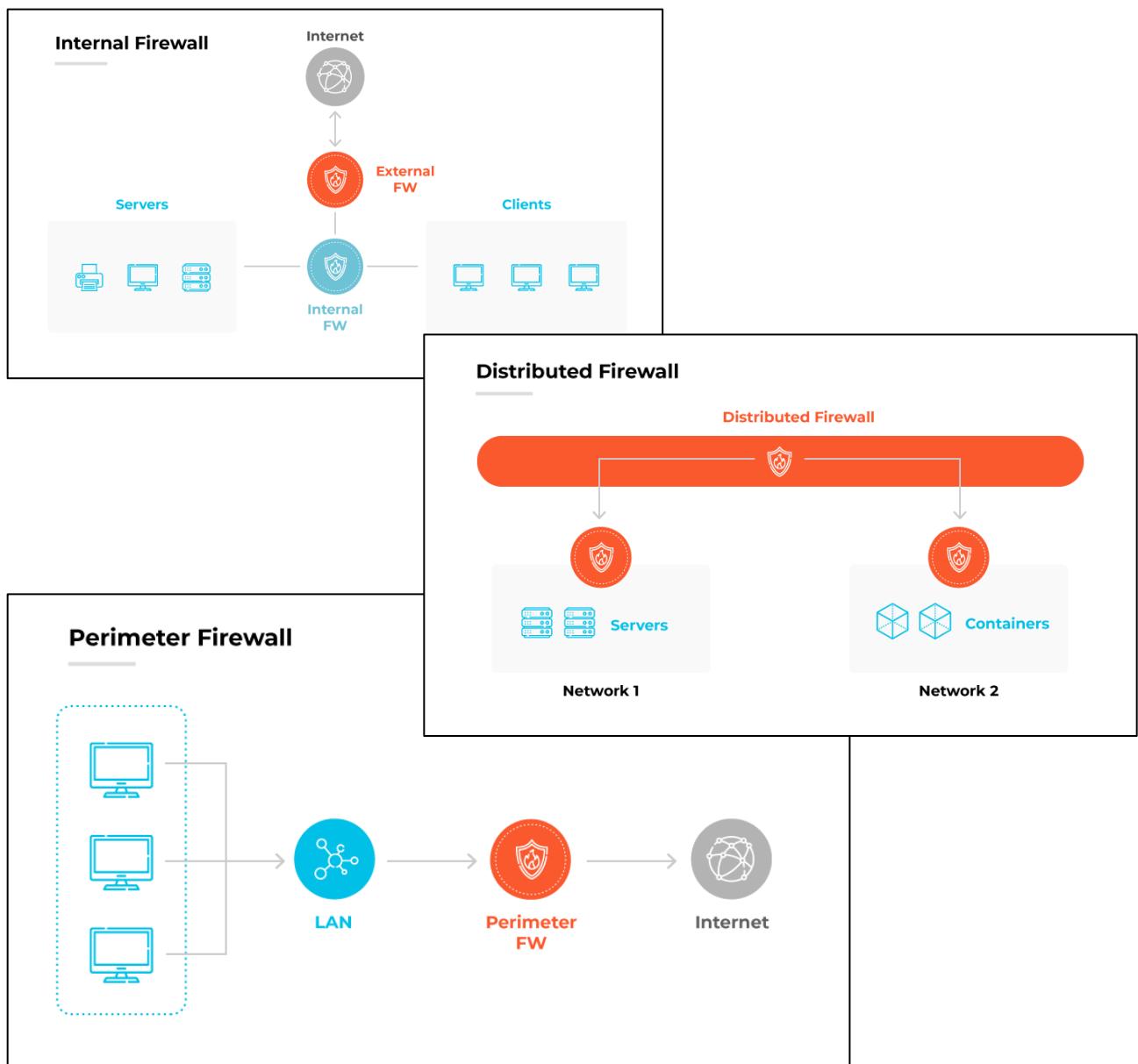
Non tutti i firewall proteggono una rete, una distinzione chiave rispetto alla definizione tradizionale.

- **Network Firewall:** Si posiziona tra una rete e un'altra, monitorando la validità del traffico in entrata e in uscita secondo un determinato set di regole. Serve a proteggere una o più reti e a mantenerne l'integrità.
- **Host-Based Firewall:** Esamina il traffico in entrata e in uscita del dispositivo su cui è installato, basandosi su un set di regole specifiche. Anche se la rete viene violata, può fornire una seconda linea di difesa per il dispositivo.



# Network Placement Firewall

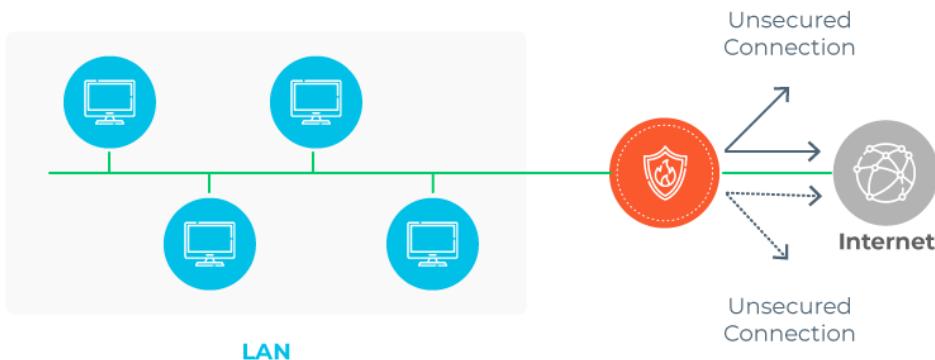
- **Internal Firewall:** Analizza il traffico scambiato tra dispositivi della stessa rete, proteggendo dalle minacce interne come insider threats o script che girano su macchine già violate.
- **Distributed Firewall:** Non si trova su una specifica macchina, ma è installato su più macchine contemporaneamente e può operare su più reti. Questo permette alle organizzazioni con molte reti e sottoreti di gestire in modo scalabile la protezione di ciascuna di esse.
- **Perimeter Firewall:** Il classico firewall situato tra una LAN e la rete Internet. Analizza il traffico in entrata e in uscita per difendere la LAN dalle minacce esterne.



# Form Factors Firewall

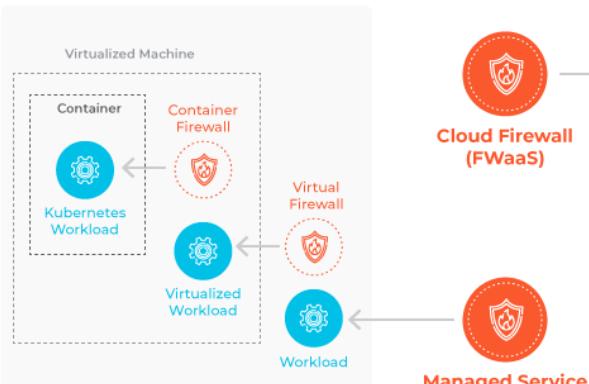
- **Hardware Firewall:** Dispositivo fisico che esegue operazioni di analisi e filtraggio del traffico. Viene fisicamente interposto tra una rete e l'altra, ad esempio, collegandolo al router e permettendo ai dispositivi di accedere a Internet solo attraverso il firewall.
- **Software Firewall:** Esegue le stesse funzioni del firewall hardware, ma viene installato su una macchina come un normale software. È utile in scenari in cui non è possibile o è difficile usare firewall fisici, come nel cloud o nei container. Essendo software, le sue funzionalità possono essere estese nel tempo.

## Hardware Firewall

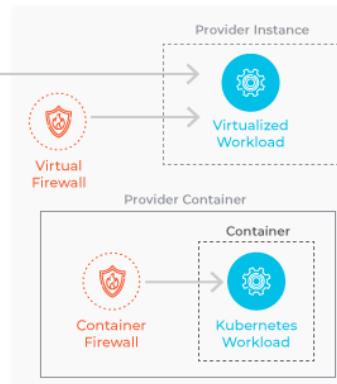


## Software Firewall

### Private Cloud



### Public Cloud



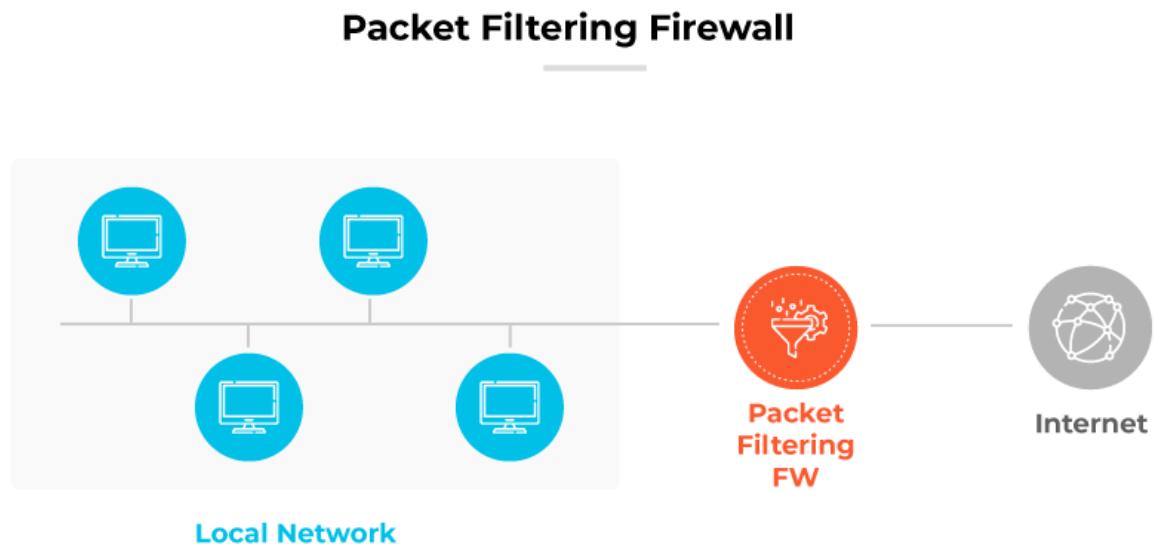
# Packet Filtering Firewall

Opera al livello 3 del modello ISO/OSI, ovvero il livello di rete. L'amministratore definisce delle regole (allow/deny) sul traffico della suite TCP/IP (TCP, UDP, ICMP, IGMP). I parametri del pacchetto che possono essere presi in considerazione per sviluppare le regole includono:

- Indirizzo IP di origine
- Indirizzo IP di destinazione
- Numero porta di origine
- Numero porta di destinazione
- Protocollo utilizzato

## Packet-Filtering Firewall: Limiti

- Si limita a far passare i pacchetti corrispondenti alle regole "allow", droppando gli altri.
- È stateless, ovvero non ha memoria dei pacchetti ricevuti precedentemente.
- Le informazioni di origine sul pacchetto, nonché altri campi, possono essere modificate dall'attaccante, permettendo di bypassare le regole del firewall.
- Vulnerabile ad attacchi di routing, in cui l'attaccante fa passare il pacchetto da un host che sa essere "allowed".



## Stateful Inspection Firewall

Operando ai livelli 3 e 4 del modello ISO/OSI (livello di rete e trasporto), lo Stateful Inspection Firewall (SIF) va oltre le funzionalità del packet-filtering firewall (PFF):

- **Funzionalità:** Oltre a quanto fatto dal PFF, il SIF memorizza i pacchetti processati precedentemente e tiene traccia delle connessioni aperte.
- **Analisi dei pacchetti:** Il SIF analizza gli header dei pacchetti per verificare se possono essere malevoli (packet inspection). Questo avviene verificando se tali header sono coerenti con altri pacchetti che sono già passati in precedenza dal firewall.
- **Esempio di utilizzo:** Durante un port scan, lo SIF potrebbe identificare tutto il traffico dell'attaccante come sospetto e bloccarlo. Il PFF, invece, si limiterebbe a seguire le regole specificate dall'amministratore, permettendo il completamento del port scan.

Il Stateful Inspection Firewall offre una maggiore sicurezza rispetto al packet-filtering firewall grazie alla sua capacità di comprendere lo stato delle connessioni e di analizzare i pacchetti in modo più dettagliato, riducendo così il rischio di attacchi malevoli che potrebbero sfuggire al firewall stateless.

## Firewall Packet Inspection

- **Packet Inspection:** Consiste nell'analizzare il contenuto degli header dei pacchetti, come gli indirizzi IP di sorgente e destinazione, le porte di sorgente e destinazione, e il protocollo utilizzato.
- **Deep Packet Inspection (DPI):** Oltre all'analisi degli header, include anche l'analisi del contenuto effettivo dei pacchetti. Il firewall decide quindi, in base all'analisi completa effettuata, se consentire o bloccare il pacchetto. Esistono due principali tecniche di analisi:
  - **Protocol Anomaly:** Di default, non viene permesso il passaggio di nessun pacchetto ("default deny"). Solo i pacchetti che corrispondono a un determinato set di regole possono passare.
  - **Pattern/Signature Matching:** Di default, viene permesso il passaggio di tutti i pacchetti ("default permit"). I pacchetti che corrispondono a un determinato set di regole (come pattern o firme di attacco noti) vengono bloccati.

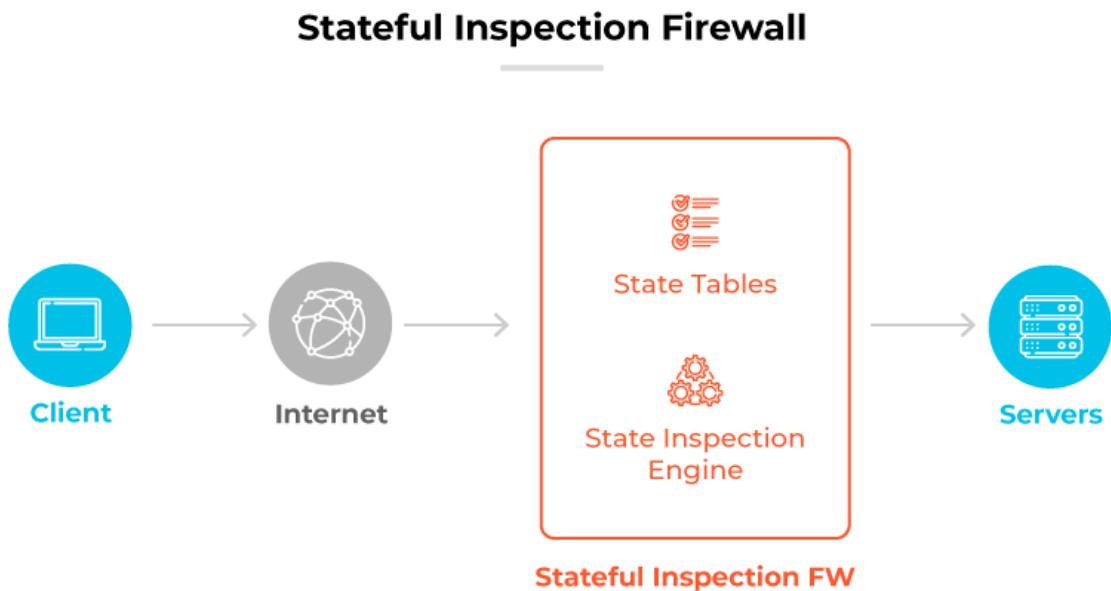
La Deep Packet Inspection è più avanzata rispetto alla semplice Packet Inspection perché permette al firewall di analizzare il contenuto effettivo dei pacchetti, aumentando così la precisione nella rilevazione di attività malevole e migliorando la sicurezza complessiva del sistema protetto.

## Stateful Inspection Firewall: Limiti

I Stateful Inspection Firewall (SIF) possono presentare alcuni limiti che includono:

- **Regole non sempre chiare e variabili nel tempo:** Le regole di filtraggio possono cambiare nel tempo o non essere definite in modo chiaro, il che può portare a rischi di falsi positivi (FP) e falsi negativi (FN). Questo significa che il firewall potrebbe erroneamente bloccare traffico legittimo (FP) o permettere traffico malevolo (FN).
- **Esempio di FP:** Un SIF potrebbe identificare l'invio di pacchetti SYN, ACK e/o FIN come traffico malevolo e non completare eventuali handshake o trasferimenti di dati, bloccando così connessioni legittime.
- **Anomalie:** Sia i Stateful Inspection Firewall che i Packet-Filtering Firewall (PFF) possono essere soggetti ad anomalie, come discusso da Cuppens et al. nel loro studio del 2012 intitolato "Handling Stateful Firewall Anomalies". Queste anomalie possono influenzare il comportamento previsto del firewall, compromettendo la sua capacità di rilevare e mitigare minacce in modo affidabile.

È fondamentale che gli amministratori di rete monitorino attentamente le regole e il comportamento dei firewall per mitigare questi rischi e mantenere una sicurezza efficace della rete.



# Circuit Level Gateway

Il Circuit Level Gateway opera al livello 5 del modello ISO/OSI, il livello di sessione. È comunemente utilizzato per creare VPN e verifica se i pacchetti inviati sono parte legittima di una sessione attraverso i seguenti passaggi:

- **Verifica della genuinità degli handshake TCP e degli attori coinvolti:** Verifica che gli handshake TCP siano autentici e che gli attori coinvolti siano autorizzati.
- **Creazione di un circuito virtuale per la durata della sessione:** Una volta stabilita la legittimità della sessione, crea un circuito virtuale che gestisce il flusso di pacchetti per tutta la durata della sessione.
- **Filtraggio dei pacchetti:** Filtra i pacchetti in base agli indirizzi e alle porte specificate, garantendo che siano validi nel contesto della sessione.

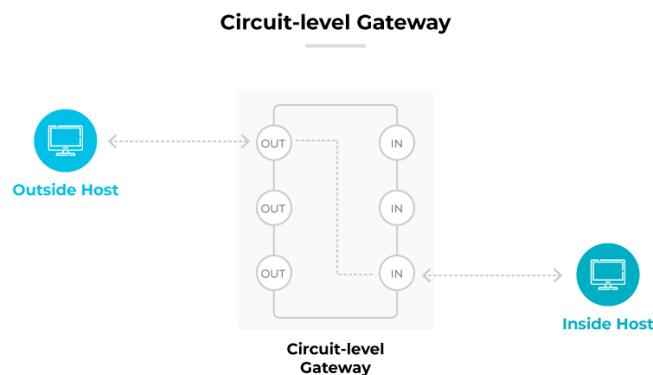
## Firewall Circuit Level Gateway

### Pro:

- **Anonimato dei computer interni:** Ogni computer esterno alla rete vede il traffico arrivare dal gateway, mantenendo un buon livello di anonimato per i computer all'interno della rete.
- **Facile configurazione, prestazioni di rete migliorate:** La configurazione è semplice e, in alcuni casi, il Circuit Level Gateway può addirittura migliorare le prestazioni della rete.

### Contro:

- **Limitazioni nei filtri:** Se un pacchetto è valido all'interno della sessione, non è possibile scartarlo perché non è possibile impostare altri tipi di filtri una volta che la sessione è stata stabilita.
- **Rischio di trasmissione di contenuti malevoli:** Questo permette a qualsiasi tipo di contenuto, come malware o payload malevoli, di passare indisturbato attraverso il canale una volta che la sessione è stata stabilita.

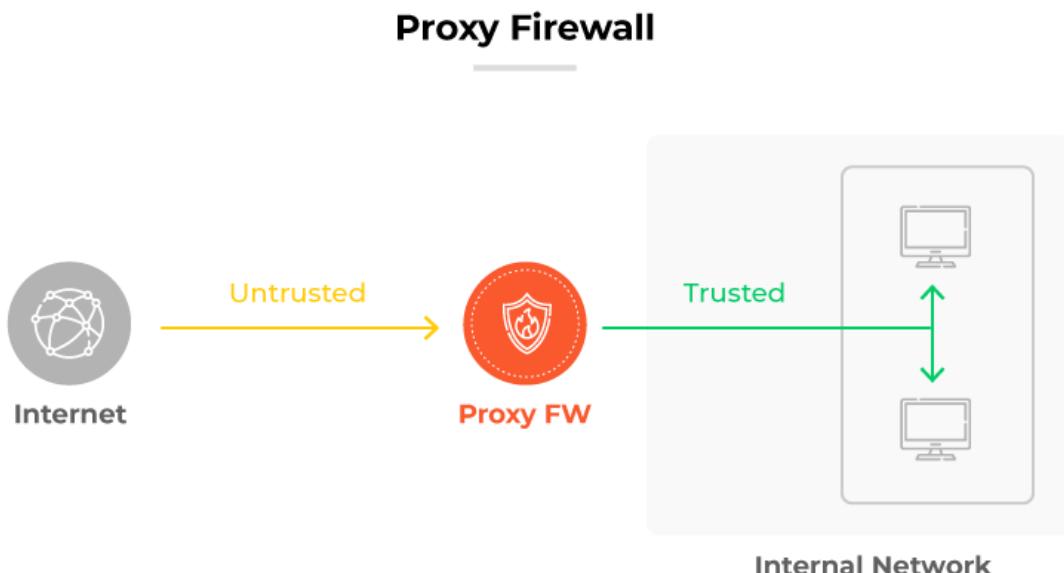


## Proxy Firewall / Application Firewall

Il Proxy Firewall, o Application Firewall, opera al livello 7 del modello ISO/OSI, il livello applicativo. Funziona come un proxy, agendo come un Man-In-The-Middle (MITM) tra i client interni alla rete e i server esterni. Ecco le sue caratteristiche principali:

- **Funzionamento:** Il Proxy Firewall si interpone tra i client interni e i server esterni. Tutte le richieste dei client passano attraverso il proxy, che può esaminare, filtrare e modificare le richieste e le risposte.
- **Anonimato e protezione:** Il Proxy Firewall dispone di un proprio indirizzo IP visibile esternamente. Protegge l'identità dei client interni perché i server esterni vedranno le richieste provenire dal proxy invece che direttamente dai client interni.
- **Deep Packet Inspection:** Può analizzare il contenuto di ogni pacchetto prima di decidere se inoltrarlo o bloccarlo. Questo tipo di analisi avanzata, conosciuta come Deep Packet Inspection (DPI), permette al proxy firewall di rilevare e prevenire minacce avanzate, come malware e attacchi basati sul contenuto dei pacchetti.

Il Proxy Firewall è particolarmente utile per garantire sicurezza a livello applicativo, proteggere l'identità dei client interni e applicare politiche di sicurezza avanzate basate sul contenuto dei dati trasferiti attraverso la rete.



# Web Application Firewall

Il Web Application Firewall (WAF) opera al livello 7 del modello ISO/OSI, il livello applicativo. Funziona come un reverse proxy, agendo come un Man-In-The-Middle (MITM) tra i client esterni alla rete e i server interni. Ecco le sue caratteristiche principali:

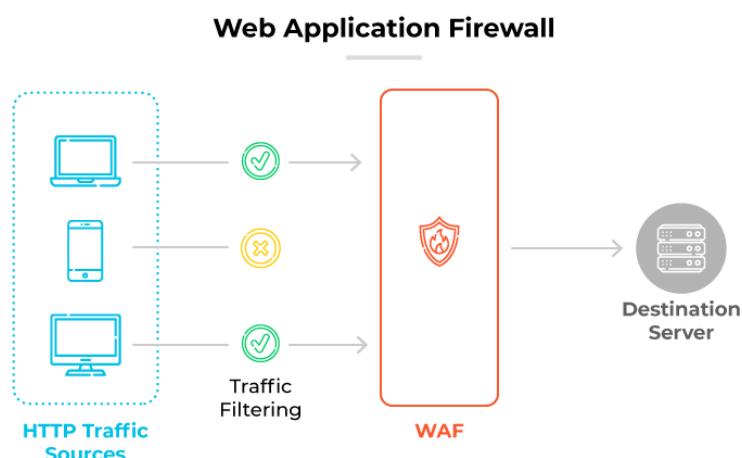
- **Funzionamento:** Il WAF fa da intermediario tra i client esterni e i server interni. Tutte le richieste dei client passano attraverso il WAF, che può esaminare, filtrare e modificare le richieste e le risposte per proteggere le applicazioni web dai vari tipi di attacchi.
- **Protezione delle applicazioni web:** L'obiettivo principale del WAF è proteggere le applicazioni web dai vari tipi di attacchi, come SQL injection, cross-site scripting (XSS), command injection e altri. Il WAF utilizza solitamente la Deep Packet Inspection per esaminare il contenuto dei pacchetti in cerca di payload noti che potrebbero sfruttare vulnerabilità comuni.
- **Protezione dell'identità dei server interni:** Il WAF protegge l'identità dei server interni esponendo solo se stesso come punto di contatto per i client esterni. Questo significa che i client esterni vedranno solo il WAF e non i server interni effettivi dietro di esso.

## Esempi di payload:

Il WAF può rilevare e bloccare payload noti utilizzati per attacchi comuni, come:

- `' or 1=1 --+` tipico payload per SQL Injection che cerca di bypassare i controlli di login.
- `;cat /etc/passwd`, tipico payload per OS Command Injection che può essere utilizzato come Proof of Concept (PoC).

Il WAF è essenziale per proteggere le applicazioni web contro attacchi sofisticati e per mantenere la sicurezza delle informazioni gestite dai server interni.

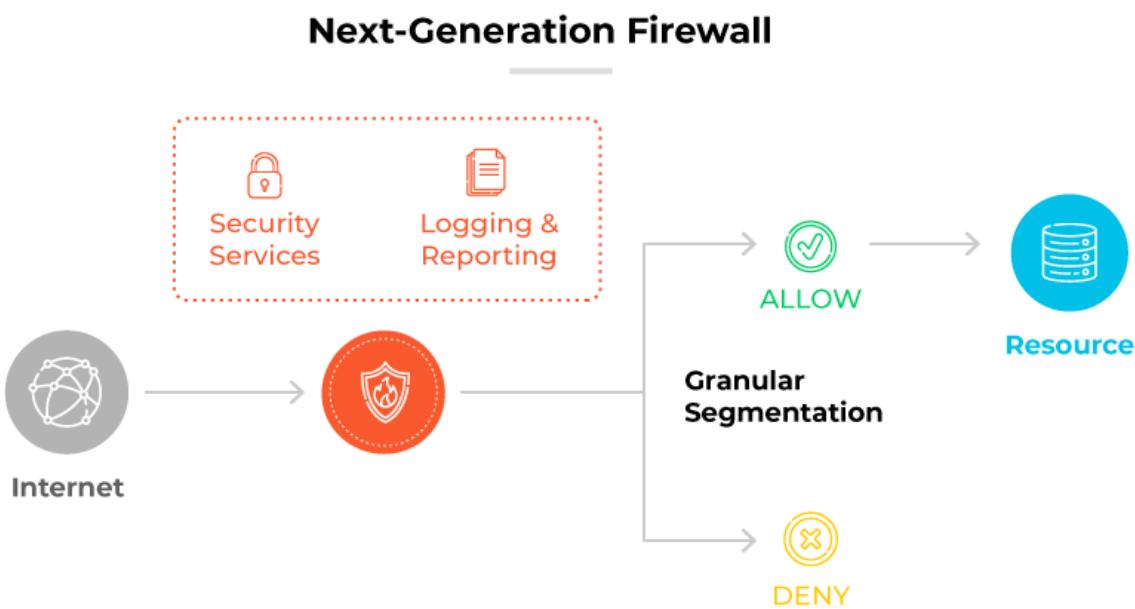


## Next generation Firewall

I Next Generation Firewall (NGFW) rappresentano una evoluzione avanzata dei firewall tradizionali, combinando diverse funzionalità in un'unica soluzione integrata. A differenza dei firewall precedenti, non esiste una definizione precisa e unificata di NGFW nella letteratura, ma generalmente:

- **Funzionalità integrate:** Un NGFW cerca di combinare le caratteristiche di diversi tipi di firewall visti fino ad ora, come packet filtering, deep packet inspection (inclusa TLS inspection), protezione dalle minacce, sistema di prevenzione intrusioni (IPS), e altre funzionalità avanzate di sicurezza.
- **Flessibilità e ampiezza di funzionalità:** NGFW sono progettati per affrontare una vasta gamma di minacce e vulnerabilità moderni, offrendo una protezione più completa e efficace rispetto ai firewall tradizionali.
- **Esempio pratico:** Il firewall FortiGate della serie 7000F, come menzionato nel link fornito, rappresenta un esempio di NGFW avanzato. Combina funzionalità di base come il packet filtering con capacità avanzate come l'ispezione TLS, la protezione dalle minacce e altre tecnologie integrate per migliorare la sicurezza della rete.

In sintesi, i NGFW sono concepiti per adattarsi alle crescenti esigenze di sicurezza delle reti moderne, integrando più strati di protezione in un'unica soluzione per affrontare le minacce in modo più efficace e efficiente.



# Netfilter

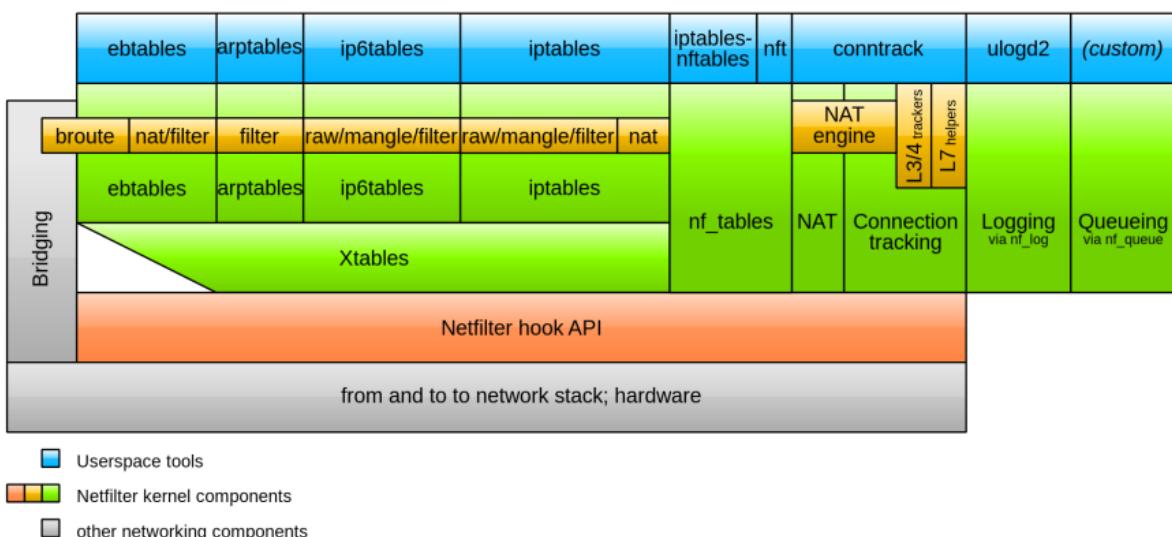
Netfilter è un framework di filtraggio dei pacchetti fornito dal kernel Linux. Gestisce il filtraggio dei pacchetti attraverso cinque hook nel networking stack, ai quali è possibile collegare funzioni personalizzate. Ecco una panoramica dei hook principali di Netfilter:

## Hooks di Netfilter

- **NF\_IP\_PRE\_ROUTING:** Questo hook aggancia i pacchetti in entrata subito dopo il loro ingresso nel networking stack. È il primo punto in cui è possibile intervenire nel processo di routing dei pacchetti in arrivo.
- **NF\_IP\_LOCAL\_IN:** Aggancia i pacchetti in entrata destinati al sistema locale. Questo hook viene attivato quando i pacchetti sono destinati alla stessa macchina su cui il filtro è attivo.
- **NF\_IP\_FORWARD:** Aggancia i pacchetti in entrata che devono essere inoltrati a un altro host. Questo hook viene attivato quando il sistema agisce come router e deve inoltrare pacchetti provenienti da una rete a un'altra.
- **NF\_IP\_LOCAL\_OUT:** Aggancia i pacchetti in uscita creati dal sistema locale, subito dopo il loro ingresso nel networking stack. Questo hook è il primo punto in cui è possibile applicare filtri ai pacchetti generati localmente prima che vengano effettivamente trasmessi sulla rete.
- **NF\_IP\_POST\_ROUTING:** Questo hook aggancia i pacchetti in uscita (sia quelli generati localmente che quelli inoltrati) subito prima che vengano effettivamente inviati. È l'ultimo punto in cui è possibile intervenire prima che i pacchetti lascino il sistema.

## *Netfilter components*

Jan Engelhardt, last updated 2014-02-28 (initial: 2008-06-17)



# Iptables

Iptables è un firewall su Linux che interagisce direttamente con il framework di filtraggio dei pacchetti Netfilter. Ecco una panoramica dettagliata delle caratteristiche e delle funzionalità di Iptables:

## Caratteristiche principali

- **Sistema Protetto:** Principalmente host-based, ma può essere utilizzato anche in scenari di perimeter a seconda della configurazione della macchina su cui è eseguito.
- **Posizionamento nella Rete:** Prevalentemente interno, ma può essere configurato anche come perimeter firewall se posizionato strategicamente.
- **Form Factor:** Software, installato direttamente sulla macchina Linux come un modulo del kernel.
- **Metodo di Filtraggio dei Dati:** Combina Packet Filtering con Stateful Inspection attraverso il modulo Conntrack di Netfilter.

## Tabelle di Iptables

Iptables organizza le regole di filtraggio dei pacchetti all'interno di diverse tabelle, ognuna delle quali ha uno scopo specifico:

- **Filter:** La tabella più utilizzata, contiene le regole per decidere se un pacchetto deve essere inoltrato alla sua destinazione o droppato.
- **NAT:** Contiene le regole per il network address translation (NAT), ovvero per modificare gli indirizzi IP di sorgente e destinazione dei pacchetti.
- **Mangle:** Contiene le regole per l'alterazione degli header dei pacchetti, come il TTL (Time To Live).
- **Raw:** Utilizzata per configurare eccezioni nel connection tracking, permettendo di specificare pacchetti che non devono essere analizzati mediante stateful inspection.
- **Security:** Utilizzata per definire parametri aggiuntivi come SEC MARK e CONSEC MARK per marcare pacchetti e sessioni con contesti di sicurezza aggiuntivi, utili per il Mandatory Access Control di SELinux.

## Chains di Iptables

Le regole di Iptables sono organizzate in catene (chains) che sono collegate agli hook di Netfilter:

- **PREROUTING:** Agganciata all'hook NF\_IP\_PRE\_ROUTING, per i pacchetti in entrata prima della decisione di routing.
- **INPUT:** Agganciata all'hook NF\_IP\_LOCAL\_IN, per i pacchetti in entrata destinati al sistema locale.
- **FORWARD:** Agganciata all'hook NF\_IP\_FORWARD, per i pacchetti in transito attraverso il sistema.
- **OUTPUT:** Agganciata all'hook NF\_IP\_LOCAL\_OUT, per i pacchetti in uscita generati localmente.
- **POSTROUTING:** Agganciata all'hook NF\_IP\_POST\_ROUTING, per i pacchetti in uscita dopo la decisione di routing.

Le catene predefinite possono essere estese con catene personalizzate, che devono essere raggiunte attraverso un jump da una delle catene predefinite.

## Conntrack (Connection Tracking)

- Modulo di Iptables che implementa la stateful inspection, permettendo di filtrare pacchetti in base al loro stato.
- Gli stati possibili per i pacchetti sono: NEW, ESTABLISHED, RELATED e INVALID, consentendo una gestione dettagliata delle connessioni di rete.

## Problemi di Iptables

- **Sovrabbondanza di tabelle e catene:** Tutte le tabelle e le catene predefinite sono sempre create, anche se non vengono utilizzate, il che può influire sulle prestazioni.
- **Gestione di IPv6:** Richiede l'uso di un tool separato chiamato Ip6tables per IPv6, il che può complicare la gestione e introdurre possibilità di errori umani.
- **Aggiornamento del kernel per nuovi protocolli:** L'introduzione di nuovi protocolli richiede un aggiornamento del kernel, potenzialmente creando incompatibilità con altri strumenti e servizi sulla macchina.

Questi problemi hanno portato allo sviluppo di Nftables nel 2014, che rappresenta il successore di Iptables all'interno di Netfilter, cercando di risolvere queste e altre problematiche.

# Nftables

Nftables rappresenta l'evoluzione di Iptables all'interno del framework Netfilter di Linux, introducendo miglioramenti significativi nella gestione delle regole di filtraggio dei pacchetti. Ecco un'analisi dettagliata delle sue caratteristiche principali:

## Fondamenti di Nftables

- **Sintassi e Creazione di Tabelle e Catene:** A differenza di Iptables, Nftables non crea tavole o catene di default. Gli amministratori devono creare esplicitamente le tavole e le catene necessarie e agganciarle agli hook desiderati nel networking stack.
- **Hook Aggiuntivo:** Introduce l'hook "Ingress", che permette di agganciare le catene ai pacchetti subito dopo il rilascio dal Network Interface Controller (NIC), prima ancora del hook "Prerouting".

## Tabelle di Nftables

- **Tipi di Tabelle:**
  - **Filter:** Simile alla tabella "Filter" di Iptables, contiene le regole per il filtraggio dei pacchetti.
  - **Route:** Simile al vecchio "Mangle" di Iptables, utilizzato per il routing dei pacchetti in uscita.
  - **NAT:** Contiene le regole per il Network Address Translation, per modificare gli indirizzi IP di sorgente e destinazione dei pacchetti.
- **Famiglie di Tabelle:**
  - **ip, ip6:** Gestisce i pacchetti IPv4 e IPv6 rispettivamente.
  - **bridge:** Utilizzato per il filtraggio dei pacchetti in bridge networking.
  - **arp:** Gestisce i pacchetti ARP.
  - **inet:** Combinazione di ip e ip6.
  - **netdev:** Introduzione significativa, permette di agganciare i pacchetti su una specifica interfaccia di rete e utilizzare l'hook "Ingress", utile per esempio nel filtraggio di attacchi DDoS.

## Catene di Nftables

- **Base Chain:** Agganciata direttamente agli hook di Netfilter, con un sistema di priorità per determinare l'ordine di esecuzione delle catene agganciate allo stesso hook.
- **Regular Chain:** Simile alle catene definite dall'utente in Iptables, si attivano solo quando vi è un jump esplicito da un'altra catena.

## Conclusioni

Nftables introduce una maggiore flessibilità e capacità di gestione rispetto a Iptables, con una sintassi migliorata e nuove funzionalità come l'hook "Ingress" e la famiglia di tabelle "netdev". Pur mantenendo i concetti di base di Iptables, Nftables offre una nuova prospettiva sulla gestione avanzata delle regole di filtraggio dei pacchetti in Linux.

# IDS vs IPS

## Sistema di Rilevamento delle Intrusioni (IDS)

- **Posizionamento nella Infrastruttura di Rete:**
  - **Out-of-Band:** IDS è di solito posizionato al di fuori della linea diretta di comunicazione. Monitora il traffico di rete passivamente senza interferire con esso.
- **Tipo di Sistema:**
  - **Passivo:** IDS è un sistema passivo che analizza il traffico di rete e genera avvisi basati su attività sospette rilevate. Non blocca attivamente o impedisce gli attacchi.
- **Meccanismi di Rilevamento:**
  - **Rilevamento basato su Firma:** IDS utilizza il rilevamento basato su firma per identificare modelli noti o firme di attacchi nel traffico di rete.
  - **Firme di fronte agli exploit:** Queste firme rilevano tecniche specifiche utilizzate dagli attaccanti.
  - **Firme di fronte alle vulnerabilità:** Queste firme rilevano tentativi di sfruttare vulnerabilità nel software o nei sistemi.

## Sistema di Prevenzione delle Intrusioni (IPS)

- **Posizionamento nella Infrastruttura di Rete:**
  - **Inline:** IPS è posizionato direttamente nel percorso del traffico di rete, consentendogli di bloccare o prevenire attivamente minacce rilevate.
- **Tipo di Sistema:**
  - **Attivo:** IPS può operare in modalità attiva, dove non solo rileva ma risponde automaticamente e blocca attività sospette in tempo reale.
  - **Passivo:** Alcuni IPS possono anche operare passivamente, simile agli IDS, monitorando e notificando le minacce senza bloccarle attivamente.
- **Mecanismi di Rilevamento:**
  - **Rilevamento basato su Firma:** Come gli IDS, IPS utilizza il rilevamento basato su firma per identificare modelli noti o firme di attacchi.
  - **Rilevamento basato su Anomalia:** Alcuni IPS avanzati possono incorporare anche il rilevamento basato su anomalie per rilevare deviazioni dal comportamento normale della rete.

	Intrusion Prevention System	IDS Deployment
<b>Placement in Network Infrastructure</b>	Part of the direct line of communication (inline)	Outside direct line of communication (out-of-band)
<b>System Type</b>	Active (monitor & automatically defend) and/or passive	Passive (monitor & notify)
<b>Detection Mechanisms</b>	1. Statistical anomaly-based detection 2. Signature detection: - Exploit-facing signatures - Vulnerability-facing signatures	1. Signature detection: - Exploit-facing signatures

# Tipi di IDS

## **Network-based (NIDS)**

Monitora il traffico in entrata e in uscita da una determinata sottorete. Analizza il traffico di rete per rilevare attività sospette o malevole all'interno della rete.

## **Host-based (HIDS)**

Viene installato su uno specifico endpoint (ad esempio, un computer) e monitora solo il suo traffico. Esamina anche file di log e altre attività del sistema per identificare comportamenti anomali.

## **Protocol-based (PIDS)**

Solitamente installato sui server, analizza i messaggi scambiati mediante uno o più protocolli di rete per identificare anomalie o stati teoricamente irraggiungibili. Aiuta a proteggere la comunicazione a livello di protocollo di rete.

## **Application Protocol-Based (APIDS)**

Analizza i messaggi scambiati mediante uno o più protocolli a livello di applicazione. Ad esempio, il Tabular Data Stream Protocol viene utilizzato per interrogare database MS-SQL. Questo tipo di IDS è specificamente progettato per monitorare il traffico relativo ad applicazioni specifiche.

## Tipi di IPS

### **Network-based (NIPS)**

Come il NIDS, viene posizionato dentro una sottorete e monitora il traffico in entrata e in uscita. Può bloccare eventuale traffico identificato come sospetto.

### **Host-based (HIPS)**

Come l'HIDS, viene installato su un endpoint e monitora il suo traffico, eventualmente bloccandolo. Offre protezione specifica per il singolo dispositivo su cui è installato.

### **Wireless-based (WIPS)**

Si connette a una rete wireless e cerca di identificare dispositivi non autorizzati connessi, con l'obiettivo di deautenticarli e prevenire accessi non autorizzati.

### **Network Behaviour Analysis (NBA)**

Posizionato all'interno di una sottorete per identificare traffico insolito. È in grado di rivelare la presenza di malware con funzionalità di rete o lo sfruttamento di vulnerabilità 0-day, analizzando i pattern di comportamento del traffico di rete.

## IDS vs IPS (In pratica)

Oggi è raro trovare un IDS che non includa funzionalità IPS, sebbene possano essere disattivate di default. Questo evidenzia un'evoluzione nella gestione delle minacce informatiche, con una crescente enfasi sulla prevenzione delle intrusioni integrata fin dalla progettazione dei sistemi (security by design).

# Snort

Snort è un IPS basato su rete (NIPS) sviluppato da Martin Roesch nel 1998, successivamente acquisito da Cisco nel 2013. È un software open-source con regole guidate dalla community e aggiornamenti frequenti, inclusa la versione più recente, Snort 3, rilasciata nel 2021.

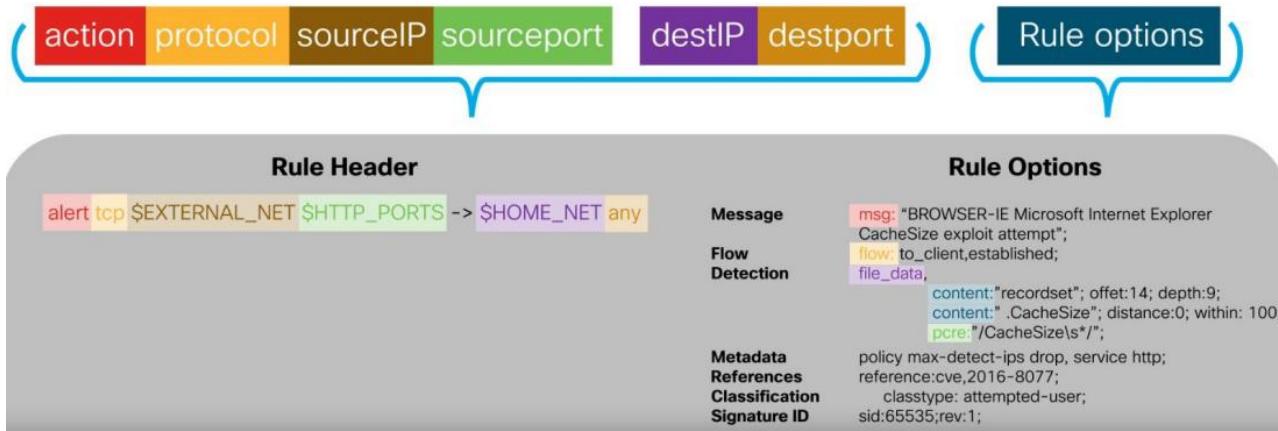
## Vantaggi e Svantaggi di Snort

### Pro:

- Snort 3 è multi-threaded, migliorando l'efficienza rispetto alle versioni precedenti.
- Gestione efficiente della RAM.
- Elevata precisione operativa in condizioni normali.
- Facile segnalazione di falsi positivi tramite il sito web dedicato.

### Contro:

- Elevato tasso di falsi positivi.
- Durante attacchi DoS o in scenari ad alto throughput, potrebbe perdere pacchetti, portando a falsi negativi.



### Azioni di Base di Snort:

- **Alert:** Genera un alert e logga il pacchetto.
- **Block:** Blocca il pacchetto e i pacchetti successivi del flusso, più log.
- **Drop:** Blocca il pacchetto, più log.
- **Log:** Logga il pacchetto.
- **Pass:** Ignora il pacchetto senza loggarlo.

### Risposte Attive di Snort:

- **React:** Invia una risposta al client e interrompe la sessione.
- **Reject:** Interrompe la sessione con un TCP reset o un ICMP unreachable.
- **Rewrite:** Sovrascrive i contenuti del pacchetto in base alle regole definite.

### Database delle Regole di Snort:

- Le regole di Snort sono disponibili online e sono descritte dettagliatamente.
- Esistono regole per rilevare tentativi di accesso a file sensibili come `/etc/passwd`.
- È possibile scaricare tutte le regole create dalla community.

Questi elementi rendono Snort una solida opzione per la sicurezza delle reti, nonostante le sfide con i falsi positivi e la gestione del throughput in situazioni di stress.

## IPS o WAF

La scelta tra IPS (Intrusion Prevention System) e WAF (Web Application Firewall) dipende dalle esigenze di sicurezza specifiche e dall'infrastruttura IT dell'organizzazione:

- **IPS** è posizionato al livello 3 (rete) e 4 (trasporto) del modello OSI, proteggendo contro intrusioni di rete e malware, ma non decripta il traffico cifrato (TLS) per motivi di prestazioni.
- **WAF** è posizionato al livello 7 (applicativo) e protegge le applicazioni web da attacchi come SQL injection e XSS, decriptando il traffico cifrato per analizzarne il contenuto.

Mentre IPS e WAF possono avere sovrapposizioni nelle funzionalità, sono progettati per scopi diversi e possono essere complementari quando implementati insieme per una sicurezza a strati.

## Suricata

Suricata è un sistema di Network-based IPS (NIPS) rilasciato nel 2010, diventato rapidamente popolare come Snort. È open-source e supportato da una comunità attiva di circa 200 contributori.

### Vantaggi e svantaggi di Suricata:

- **Multi-threaded:** Supporta il threading multipli, migliorando l'efficienza nell'elaborazione dei pacchetti.
- **Consumo di RAM:** Richiede più memoria rispetto a Snort, ma è comunque efficiente nelle prestazioni.
- **Perdita di pacchetti/alert:** Riduce al minimo la perdita di pacchetti o alert anche in condizioni operative stressanti.
- **Precisione:** Meno preciso di Snort ma con meno falsi positivi, il che implica un potenziale incremento di falsi negativi.
- **NSM (Network Security Monitoring):** Funge anche da sistema di monitoraggio della sicurezza di rete, registrando e interpretando tutto il traffico in formato JSON, facilmente integrabile con altri strumenti come Elastic e Tulip.
- **Scripting Lua:** Supporta lo scripting Lua per personalizzare le regole di detection, offrendo maggiore flessibilità.

### Regole in Suricata:

Le regole in Suricata sono simili a quelle di Snort ma con alcune differenze nelle azioni e nella sintassi. Utilizzano un linguaggio specifico per specificare condizioni di alert basate su protocolli di rete e altri parametri.

Suricata si distingue per la sua capacità di gestire grandi volumi di traffico, minimizzando la perdita di informazioni critiche e supportando una vasta gamma di configurazioni personalizzabili attraverso l'uso di Lua e altri strumenti di scripting.

# Endpoint Detection and Response (EDR)

**Definizione:** EDR è un software progettato per proteggere gli endpoint aziendali (come laptop, smartphone) da varie minacce, superando i limiti degli antivirus tradizionali. Gartner ha riconosciuto EDR nel 2013 come parte fondamentale della difesa cibernetica moderna.

## **Funzionalità di EDR:**

- **Raccolta dei dati dagli endpoint:** Monitora e registra tutte le attività e gli eventi sugli endpoint, come processi attivi, modifiche ai file, siti web visitati, ecc.
- **Analisi dei dati per il rilevamento delle minacce:** Utilizza algoritmi di machine learning per identificare comportamenti sospetti o indicatori di compromissione (IOC) e indicatori di attacco (IOA).
- **Risposta automatizzata alle minacce:** Una volta rilevata una minaccia, EDR può sollevare alert, isolare dispositivi, bloccare funzionalità, impedire l'esecuzione di codice malevolo e scansionare altri endpoint per minacce simili.
- **Risoluzione delle minacce e supporto per la ricerca:** Fornisce supporto per analisi forensi, identifica file infetti, vulnerabilità sfruttate, credenziali compromesse e suggerisce modifiche alle regole di Access Control per prevenire futuri attacchi.
- **Interfacciamento con MITRE ATT&CK:** Alcuni EDR si interfacciano con MITRE ATT&CK per comprendere meglio le tecniche di attacco utilizzate dagli aggressori reali.

## **Esempi e Considerazioni:**

- **Soluzioni commerciali:** Fornite da aziende leader in sicurezza come Kaspersky, Checkpoint, Sophos, offrono una vasta gamma di funzionalità EDR.
- **EDR open source:** Esistono anche soluzioni EDR open source come OpenEDR, ma possono essere meno mantenute o meno frequentemente aggiornate.

EDR rappresenta una risposta avanzata e automatizzata alle minacce informatiche sugli endpoint, sfruttando la raccolta di dati dettagliata e l'analisi intelligente per proteggere le organizzazioni dalle minacce moderne.

## eXtended Detection and Response (XDR)

**Definizione:** XDR è una soluzione di sicurezza che estende le funzionalità di EDR per proteggere tutti gli asset aziendali, inclusi quelli on-premises e in cloud. Lo scopo è fornire una protezione olistica e migliorare la capacità di rilevamento e risposta alle minacce su tutta l'infrastruttura IT.

### **Funzionalità:**

- **Monitoraggio Continuo:** Rete, e-mail, applicazioni, server (anche in cloud) e altri sistemi di protezione.
- **Raccolta Dati di Telemetria:** Collezione e analisi dei dati provenienti da diversi sistemi aziendali per una visione integrata della sicurezza.

## Protezione data dai sistemi di difesa

### **"Se utilizzo Firewall, IPS, Antivirus e XDR sono al sicuro?"**

Purtroppo, la risposta è "no". È fondamentale comprendere che il Blue Team (difensori) reagisce agli attacchi del Red Team (attaccanti) e raramente può prevenire attacchi non ancora ideati.

### **Approccio Best Effort:**

- Il Blue Team opera quasi sempre in modalità best effort.
- Ogni giorno emergono nuove problematiche di sicurezza, come vulnerabilità appena scoperte, campagne malware inedite e nuovi tentativi di intrusione.

Nonostante l'adozione di Firewall, IPS, Antivirus e XDR, un incidente di sicurezza è sempre possibile. La sicurezza informatica è dinamica e richiede vigilanza continua e adattamento costante alle nuove minacce.

# Incident Response

Incident Response è definito da IBM come i processi e le tecnologie di un'organizzazione mirati a individuare e rispondere alle minacce informatiche, intrusioni o attacchi cibernetici. Anche gli standard come ISO/IEC 27001 richiedono procedure specifiche per gestire tali incidenti.

## Tipologie di Incidenti Comuni

### **Ransomware**

- Malware che blocca l'utilizzo del sistema operativo del computer vittima, chiedendo un riscatto per restituire le funzionalità bloccate.
- I ransomware odierni criptano il contenuto dei dischi collegati alla macchina violata. Pagando il riscatto, si ottiene la chiave per decriptarli (sebbene non garantito).

### **Phishing**

- Messaggi, in qualunque formato, che mirano a manipolare il comportamento del destinatario per fargli eseguire azioni dannose.
- Azioni tipiche includono scaricare e aprire file (potenzialmente malevoli), effettuare bonifici, e cliccare su link che reindirizzano a domini malevoli.
- Spesso il mittente si finge un parente o una figura autorevole. Quando il messaggio è attentamente personalizzato, si parla di spear-phishing.

### **Social Engineering**

Insieme di tecniche che hanno lo scopo di manipolare il comportamento delle persone e fare loro eseguire azioni che porteranno alla compromissione della sicurezza dei loro asset personali o aziendali.

## Tipologie di Social Engineering:

- **Phishing**
- **Baiting:** Ad esempio, la truffa del principe nigeriano o CD-ROM gratuiti.
- **Tailgating:** L'attaccante segue qualcuno in un'area riservata fisicamente o sfrutta una pausa pranzo senza bloccare lo schermo digitalmente.
- **Pretexting:** Creare un falso pretesto e offrirsi di risolverlo, come le pubblicità che propongono falsi antivirus.
- **Quid pro quo:** Offrire un premio in cambio di un'azione, come il milionesimo visitatore che deve installare un software.
- **Scareware:** Software che usa la paura per manipolare, come pagine web che accusano l'utente di un crimine e chiedono di installare software per verificare l'innocenza.
- **Watering hole attack:** Iniezione di codice malevolo in un sito web o software tramite vulnerabilità per rubare credenziali o eseguire codice malevolo.

## Importanza del Social Engineering

Nonostante si pensi che il Social Engineering influenzi solo una minoranza non abituata alla tecnologia, i dati mostrano un quadro diverso. Ad esempio, il Cyber Security Breaches Survey 2024 del governo UK evidenzia l'ampio impatto di queste tecniche.

## Possibili impatti di un incidente

Gli incidenti di sicurezza informatica possono causare significativi impatti negativi per le aziende. Per esempio, una violazione di un account email aziendale può aumentare il rischio di phishing, soprattutto se l'email proviene da un indirizzo compromesso di un collega o del capo. Inoltre, vulnerabilità condivise tra diversi sistemi possono facilitare l'accesso laterale degli attaccanti, rendendo complicato il recupero e la pulizia delle macchine colpite. Talvolta, la migliore soluzione è ripristinare completamente i sistemi violati, utilizzando backup sicuri o formattazioni. Altri gravi impatti includono la possibile esfiltrazione e vendita di dati sensibili nelle darknet, con conseguenti danni d'immagine, multe, denunce e perdite finanziarie. In caso di attacchi ransomware, dove i dati sono criptati e resi inaccessibili, è spesso necessario ripristinare i sistemi, anche se in alcuni paesi, come in Italia, pagare il riscatto è illegale. Affrontare questi incidenti richiede competenze avanzate in forensics e una strategia di risposta agli incidenti ben strutturata. La sicurezza informatica deve essere considerata una priorità strategica per ogni organizzazione, con investimenti adeguati in prevenzione, rilevamento e risposta agli incidenti.

## Log di sistema

I log di sistema registrano ogni azione effettuata da utenti e sistemi all'interno di un sistema operativo o di applicazioni. Tuttavia, in caso di violazione di sicurezza, questi log potrebbero non essere affidabili. Gli attaccanti potrebbero cancellarli o modificarli per eludere la rilevazione o incolpare altri. Pertanto, è essenziale disporre di strumenti che garantiscano l'integrità dei log, consentendo una corretta analisi forense e un'accurata ricostruzione degli eventi avvenuti durante l'incidente.

## SIEM

Il Security Information and Event Monitoring (SIEM) è un sistema critico nella cybersecurity, concepito per collezionare, gestire e analizzare i log di sicurezza provenienti da diverse fonti all'interno di una rete aziendale. Questo strumento, originariamente definito nel 2005 da Gartner combinando le funzionalità di Security Information Monitoring (SIM) e Security Event Monitoring (SEM), opera solitamente su una macchina separata dedicata.

I requisiti fondamentali per un corretto funzionamento di un SIEM includono la separazione fisica dalla rete principale, il ricevimento immediato dei log appena generati dalle macchine, e l'utilizzo di metodi di crittografia sia per i log in transito che quelli salvati. È essenziale garantire l'integrità dei file di log utilizzando hash crittografici o HMAC (Hash-based Message Authentication Code).

L'integrità dei log è cruciale perché fornisce una prova affidabile degli eventi accaduti e aiuta a identificare e rispondere prontamente a potenziali violazioni. Gli hash crittografici o i MAC vengono utilizzati per verificare che i log non siano stati alterati dopo la loro generazione. Questo meccanismo permette di rilevare qualsiasi tentativo di manipolazione da parte di un attaccante, che dovrebbe compromettere sia la macchina che genera i log sia il sistema SIEM stesso per nascondere le sue tracce.

Questa struttura garantisce che il SIEM possa rilevare le intrusioni prima che gli attaccanti riescano a compromettere gravemente il sistema, fornendo al Blue Team la possibilità di rispondere tempestivamente agli incidenti di sicurezza.

## Esempi di Alert

Alcuni esempi di regole personalizzate per avvisi sulle condizioni degli eventi comprendono regole di autenticazione dell'utente, rilevamento degli attacchi e rilevamento delle infezioni.

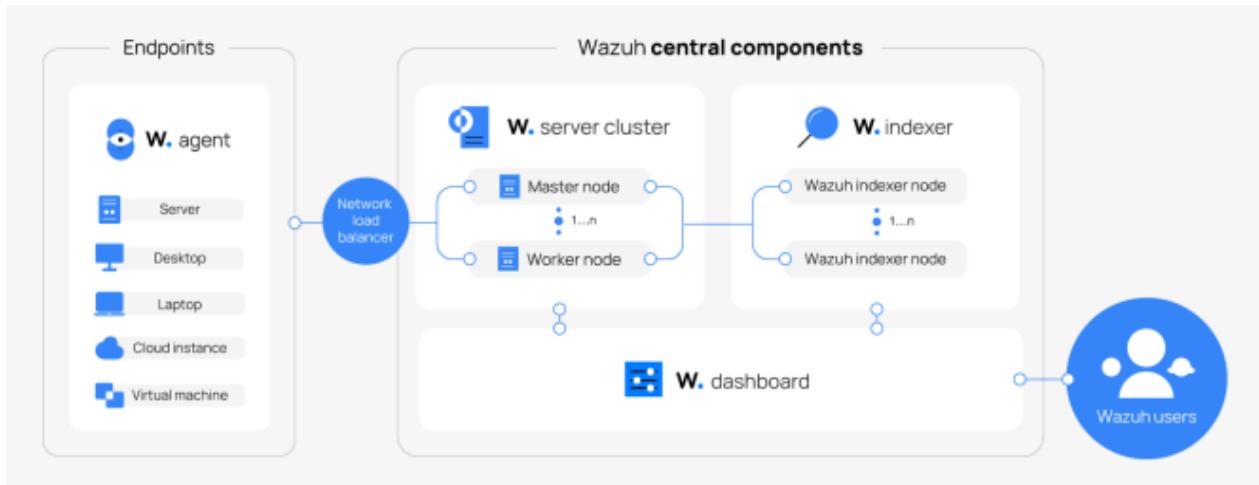
Regola	Traguardo	Grilletto	Originari degli eventi
Ripeti origine di accesso all'attacco	Avviso tempestivo per attacchi di forza bruta, indovinare le password e applicazioni configurate in modo errato.	Avviso su 3 o più accessi non riusciti in 1 minuto da un singolo host.	Active Directory, Syslog (host Unix, switch, router, VPN), RADIUS, TACACS, applicazioni monitorate.
Ripeti attacco firewall	Allerta precoce per scansioni, propagazione di worm, ecc.	Avvisa su 15 o più eventi di caduta/rifiuto/negazione del firewall da un singolo indirizzo IP in un minuto.	Firewall, router e switch.
Ripeti il sistema di prevenzione delle intrusioni di rete	Allerta precoce per scansioni, propagazione di worm, ecc.	Avviso su 7 o più avvisi IDS da un singolo indirizzo IP in un minuto	Dispositivi di rilevamento e prevenzione delle intrusioni di rete
Ripeti il sistema di prevenzione delle intrusioni dell'host di attacco	Trovare gli host che potrebbero essere infetti o compromessi (che mostrano comportamenti di infezione)	Avviso su 3 o più eventi da un singolo indirizzo IP in 10 minuti	Avvisi del sistema di prevenzione delle intrusioni dell'host
Rilevamento/rimozione di virus	Avvisa quando viene rilevato un virus, uno spyware o un altro malware su un host	Avvisa quando un singolo host rileva un malware identificabile	Antivirus, HIPS, rilevatori di anomalie comportamentali di rete/sistema
Virus o spyware rilevati ma non riusciti a pulire	Avvisa quando è trascorsa >1 ora da quando il malware è stato rilevato, su una fonte, senza che il virus corrispondente sia stato rimosso correttamente	Avvisa quando un singolo host non riesce a pulire automaticamente il malware entro 1 ora dal rilevamento	Firewall, NIPS, Anti-Virus, HIPS, eventi di accesso non riusciti

## IBM QRadar

IBM QRadar è una suite di strumenti modulari sviluppata da IBM per la sicurezza informatica. Include SIEM per la gestione degli eventi e delle informazioni di sicurezza, EDR per il rilevamento e la risposta agli incidenti sugli endpoint, e Log Insights per la visualizzazione dei log tramite una piattaforma cloud. Inoltre, offre SOAR per l'automazione e l'orchestrazione della risposta alle minacce. La versione gratuita, QRadar Community Edition, supporta solo SIEM e gestisce fino a 100 eventi al secondo.

# Wazuh

Wazuh è una piattaforma gratuita che unisce funzionalità di SIEM e XDR, lanciata nel 2017. È composta da quattro moduli essenziali: l'Indexer per l'archiviazione degli alert, il Server che gestisce gli agent, il Dashboard per l'interfaccia utente e l'Agent installabile sugli endpoint. Questa soluzione è focalizzata sul monitoraggio avanzato e sulla risposta agli incidenti, rendendola adatta per la sicurezza informatica a livello aziendale.



## Wazuh Regole

```
<rule id="5700" level="0" noalert="1">
  <decoded_as>sshd</decoded_as>
  <description>SSHD messages grouped.</description>
</rule>

<rule id="5716" level="5">
  <if_sid>5700</if_sid>
  <match>^Failed|~error: PAM: Authentication</match>
  <description>sshd: authentication failed.</description>
  <mitre>
    <id>T1110.001</id>
    <id>T1021.004</id>
  </mitre>
  <group>authentication_failed,gdpr_IV_35.7.d,gdpr_IV_32.2,g
    ,tsc_CC7.3,</group>
</rule>

<rule id="5763" level="10" frequency="8" timeframe="120" ignore="60">
  <if_matched_sid>5760</if_matched_sid>
  <same_source_ip/>
  <description>sshd: brute force trying to get access to the system. Authentication failed.</de
  <mitre>
    <id>T1110</id>
  </mitre>
  <group>authentication_failures,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_SI.4,n
    .1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
```

## SOC

Il SOC (Security Operations Center) è il team responsabile del monitoraggio continuo e della risposta agli incidenti all'interno di un'organizzazione. Gestito dal CISO (Chief Information Security Officer) o da un SOC Manager, il SOC utilizza una serie di strumenti per monitorare l'infrastruttura 24/7 e reagire prontamente quando necessario. Può essere interno all'azienda o gestito da consulenti esterni.

Oltre al monitoraggio degli strumenti di sicurezza, i compiti del SOC includono la gestione dell'inventario degli asset, la verifica della protezione degli asset, l'implementazione di aggiornamenti di sistema e delle regole di sicurezza, la verifica dei backup e la gestione delle procedure di Incident Response, che include test regolari per simulare situazioni di attacco.

I membri del SOC partecipano a corsi di aggiornamento per rimanere al passo con le nuove minacce e le soluzioni di difesa, utilizzano strumenti di Threat Intelligence e talvolta si occupano anche della conformità normativa, verificando che le soluzioni di sicurezza rispettino le normative applicabili come PCI DSS, GDPR e NIS2.

## Greynoise

Greynoise è una rete di honeypot distribuita su Internet che simula la presenza di servizi vulnerabili, noti come "Personae". Il suo obiettivo principale è raccogliere informazioni sugli attaccanti e sulle loro metodologie di attacco monitorando le interazioni con questi servizi falsi.

Questi dati sono estremamente utili per i Security Operations Center (SOC) poiché consentono di valutare se gli alert generati sono effettivamente riconducibili a attacchi reali o se potrebbero trattarsi di falsi positivi. Questa analisi aiuta il SOC a priorizzare le minacce e adottare risposte appropriate in tempi rapidi.

Greynoise offre anche la possibilità di interrogare il suo database gratuitamente attraverso il loro sito web, fornendo agli operatori di sicurezza accesso a informazioni cruciali per migliorare la capacità di difesa dell'organizzazione contro le minacce cibernetiche.