UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE ED INFORMATICHE
Corso di Laurea in Informatica

Sicurezza delle reti - Parte A

Metodi di attacco e difesa

RETI DI CALCOLATORI - a.a. 2023/2024

Roberto Alfieri

La sicurezza delle reti: sommario

PARTE A

- I servizi di sicurezza
- Metodi e strumenti di Attacco
- Metodi e strumenti di Difesa

PARTE B

- Crittografia applicata e OpenSSL

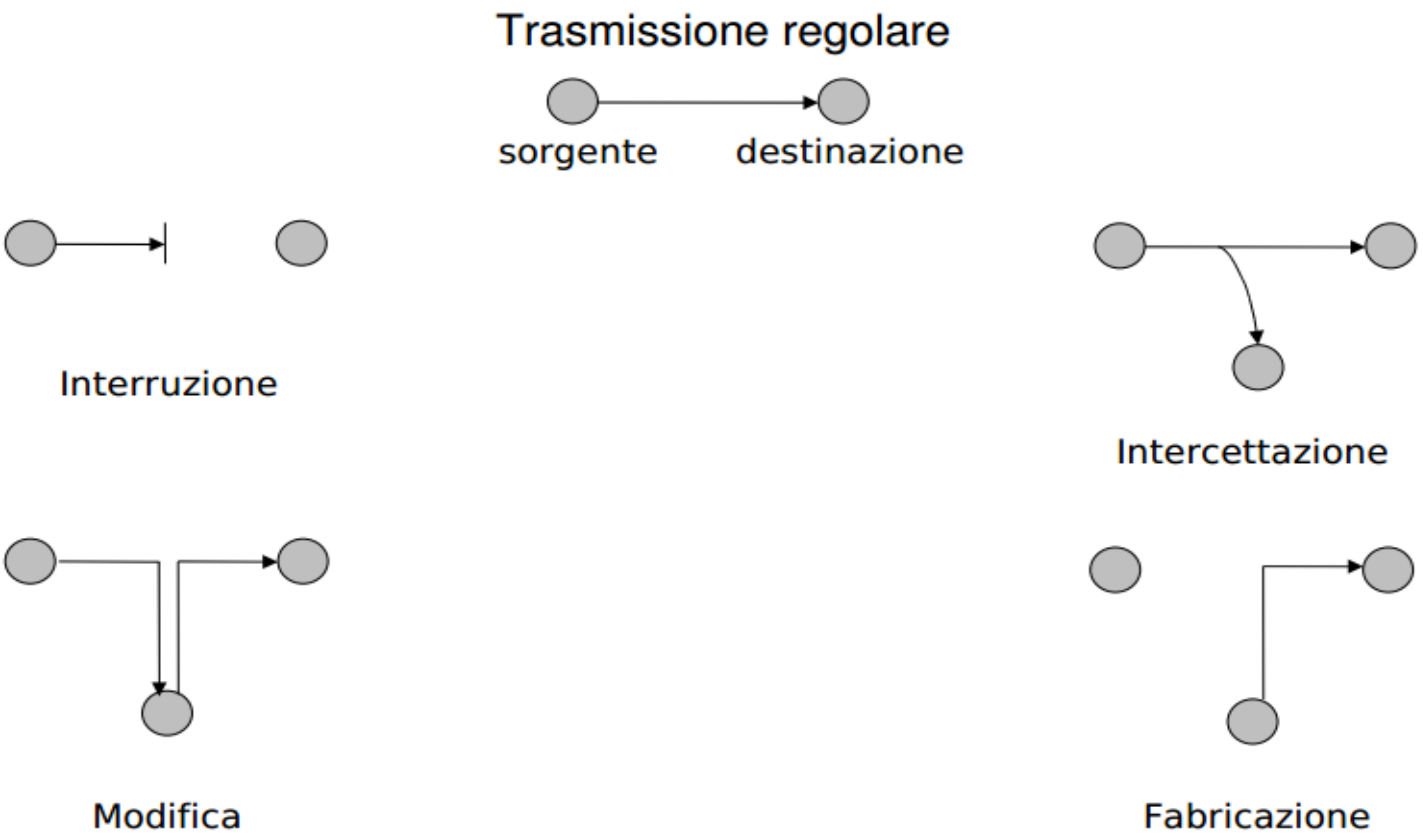
PARTE C

- Protocolli di autenticazione
- Ipsec e VPN
- Sicurezza delle reti WiFi

La sicurezza

Con il termine sicurezza informatica si intende quel ramo dell'informatica che si occupa dell'analisi delle vulnerabilità, del rischio, delle minacce e della successiva protezione dell'integrità logico-funzionale.

La gran parte delle minacce derivano dalla rete Internet, per cui la sicurezza informatica è un tema importante nello studio delle reti di calcolatori.



Politiche di sicurezza

I beni da proteggere (asset) sono principalmente i dati gestiti dalla rete informatica, ma per fare ciò bisogna anche proteggere i computer ed i vari dispositivi presenti nella rete.

La realizzazione di un sistema che garantisca una assoluta protezione da abusi è impossibile, ma è possibile attivare meccanismi di sicurezza tali da limitare e scoraggiare i tentativi.

La politica di sicurezza è ***quindi un compromesso, dettato dalle proprie necessità, tra il costo per attivarla ed il beneficio ottenuto in termini di diminuzione del rischio.***

Security services

Per proteggere un host in rete o una comunicazione occorre stabilire dei servizi di sicurezza che possono essere classificati nel seguente modo:

Autenticazione (Authentication): un servizio che consente di accertare l'identità dichiarata da una entità (origine dei dati o peer in una comunicazione) mediate la verifica di credenziali. Avviene in 2 step: Identificazione e verifica.

Autorizzazione (Authorization, Access Control): Protegge l'accesso ad una risorsa mediante l'applicazione di "Security Policy".

Confidenzialità/Riservatezza (Data Confidentiality): Impedisce l'utilizzo delle informazioni da accessi non autorizzati.

Integrità dei dati (Data Integrity): consente di garantire che i dati acceduti non sono stati modificati.

Integrità dei sistemi (System Integrity): protegge le risorse del sistema contro modifiche, distruzioni accidentali o non autorizzate.

Non ripudio (Non-Repudiation): fornisce protezione contro il ripudio nel coinvolgimento in una comunicazione.

- non ripudio della sorgente: prova chi è il mittente dei dati in una transazione
- non ripudio della destinazione: prova che i dati sono arrivati ad uno specifico destinatario

Disponibilità (Avaliability): Fornisce una protezione per garantire accessibilità di una risorsa di sistema o di rete.

Audit (**Accoutability, Traceability**): Registrazione di eventi di sistema o di rete. Consente di rintracciare, ricostruire (ed eventualmente addebitare) l'utilizzo delle risorse.

Riferimenti: RFC2828 di IETF e la raccomandazione X.800 di ITU

Metodi e strumenti di attacco ai servizi di sicurezza

Esistono diversi metodi e strumenti che un attaccante può utilizzare per violare i servizi di sicurezza a protezione di un host o di una rete.

Gli attacchi possono essere **passivi**, se l'obiettivo è la raccolta di informazioni, in modo lecito o illecito, riguardo un possibile bersaglio, oppure **attivi** quando coinvolgono l'alterazione dei dati sui dispositivi di memorizzazione o nei flussi di comunicazione.

Solitamente un attacco passivo è funzionale ad un successivo attacco attivo.

Attacco passivo

Gli attacchi passivi includono quelle attività che consentono di raccogliere dati e informazioni di un asset bersaglio senza alcuna alterazione sui dati e sui dispositivi.

L'obiettivo principale è la raccolta di informazioni sul target utili per un successivo attacco attivo.

Tipologie principali di attacco passivo:

- Scansione via rete del target e/o della rete a cui appartiene
  - Strumenti: port scanning (esempio nmap)
- Intercettazione delle comunicazioni
  - Strumenti: Sniffing (esempio tcpdump)
- Raccolta di informazioni Open Source (OSINT)
  - Strumenti: ricerca di informazioni forum, blog, siti di social network, siti di condivisione di video, wiki, record Whois e DNS di nomi di dominio registrati, metadati e file digitali, risorse web scure, dati di geolocalizzazione, indirizzi IP , i motori di ricerca delle persone e tutto ciò che può essere trovato online.

Queste attività sono il punto di partenza per un attaccante ma sono anche strumenti di difesa che possono consentire i conoscere i propri punti deboli.

Attacco passivo: Sniffing e scanning

Sniffing: Analizzare il traffico di rete utilizzando un analizzatore di protocollo.

E' necessario un accesso privilegiato all'interfaccia di rete.

Possono essere orientati ad analizzare una sequenza di pacchetti

- Esempi di strumenti: **tcpdump e wireshark**

Scanning: testare un intervallo di indirizzi IP e numeri di porta per vedere quali servizi o sistemi sono presenti ed attivi.

- Esempio di strumento : **nmap**

nmap è uno tool open-source per la network exploration e l'auditing.

https://nmap.org/man/it/

Opzioni significative: https://nmap.org/book/port-scanning-options.html

-sP (ping scan)

-sT (TCP connect() - default)

-sS (TCP syn, richiede i priv. di root)

-A rileva versione

-p seleziona le porte da testare (per default vengono scansionate le 1000 porte più popolari)

Esempi:

nmap 192.168.0.0/24

nmap -A 192.168.0.254 -p 22,8001-8060

Attacco passivo: OSINT

Open Source INTelligence si riferisce alla ricerca di informazioni tratte da fonti liberamente disponibili (non all'open source software).

Ricercare, esaminare, correlare le informazioni pubblicamente disponibili permette di ottenere informazioni sull'organizzazione, sui progetti, sui processi aziendali, sulle persone.

Le informazioni recuperabili tramite OSINT non sono utilizzabili soltanto per fini “negativi”

Possono essere utilmente impiegate per supportare decisioni strategiche, valutare campagne di marketing, verificare il “sentiment” e la reputazione online

Possono essere impiegate per migliorare la sicurezza dell'organizzazione

Modalità di OSINT

- ✓ **Manuale:** la ricerca viene effettuata direttamente dall'operatore

  Le scelte vengono fatte sul momento. Raccolta e confronto oneroso, difficilmente “scalabile” ed elevato rischio di “perdere” informazioni

- ✓ **Automatica:** la ricerca viene effettuata tramite strumenti parametrizzabili

  Deve essere effettuata una scrematura a posteriori. Potenzialmente molto efficace, facilmente “scalabile” ed integrabile, richiede enormi investimenti (HW e SW), le tecnologie sono in rapida convergenza. Rischio di falsi positivi dovuti a AI non evoluta come analista

Attacco attivo

Gli attacchi attivi sono quelli che si basano sull'alterazione dei dati oppure dei flussi con cui i dati sono trasmessi in rete.

Tipologie principali di attacco attivo:

- **Fabbricazione dell'identità** di un'altra entità (Authentication attacks)
  - Spoofing, Man in the Middle
- **Interruzione/compromissione di servizi** (Avaliability attacks)
  - Denial of Service (DoS), Distributed DoS (DDoS)
- Sfruttamento di **Vulnerabilità** nel software installato (System Integrity and Authentication attacks)
  - Applicazione degli Exploit noti
- Diffusione di Vulnerabilità intenzionali (System Integrity attacks and auth attacks)
  - Virus, Worms, BOT, Trojan
- Ingegneria Sociale
  - Phishing

Attacco attivo: Spoofing (fabbricazione)

E' un tipo di attacco informatico dove viene impiegata la falsificazione dell'identità (spoof)

Quando la falsificazione non avviene in campo informatico si parla di social engineering

User account spoofing: usare nome utente e password di un altro utente senza averne il diritto. Può avvenire utilizzando strumenti come sniffer e password crackers

I password cracker possono essere off-line come John the Ripper http://www.openwall.com/john/ , oppure on-line, come Hydra

IP Address spoofing: Si basa sul fatto che la maggior parte dei routers all'interno di una rete controllino solo l'indirizzo IP di destinazione e non quello sorgente. Finalità:

- superare le tecniche difensive basate sull'autenticazione dell'indirizzo
- Realizzare attacchi DDoS. Vedi ad esempio "NTP reflection"

MAC Address forging: il MAC address viene modificato impersonando l'indirizzo della vittima. Diversi sistemi di autenticazione/autorizzazione sono basati su MAC address.

- Autenticazione verso DHCP server
- Sessioni attive su Captive Portal (session Hijacking)

ARP Spoofing / Poisoning: Consiste nell'inviare intenzionalmente e in modo forzato risposte ARP contenenti dati inesatti. In questo modo la tabella ARP di un host conterrà dati alterati. Ettercap è un tool per attacco di tipo **man-in-the-middle**, basato su ARP poisoning.

Attacco attivo: Denial of Service (DoS)

Causano la perdita dell'utilizzo di una risorsa sovraccaricandola, ma non ne permettono l'accesso all'attaccante. Gli attacchi possono essere

- diretti (l'attaccante interagisce direttamente con la vittima)
- indiretti (l'attaccante sfrutta terze parti).

I principali attacchi sono:

- **FLOODING**
  - **Ping floods:** invio di ICMP echo request in numero maggiore a quelli gestibili dal sistema attaccato; l'aggressore invia un grosso flusso di traffico ICMP echo verso una serie di indirizzi di broadcast attribuendosi come indirizzo sorgente quello della vittima.
  - **TCP SYN Floods:** Funziona se un server alloca delle risorse dopo aver ricevuto un SYN, ma prima di aver ricevuto un messaggio ACK (vedi nmap -sS).
  - **NINVIO DI PACHETTI MALFORMATI**
  - **Ping di grandi dimensioni (ping of death**): può causare buffer overflow con conseguente blocco del servizio o, nei casi più gravi, crash del sistema.
  - **UDP bombs:** costruiti con valori illegali in certi campi. In certi sistemi operativi la ricezione di pacchetti imprevisti può causare crash
- **ATTACCHI DA PIU' HOST: DDOS (Distributed DoS)**

E' una variante di DoS realizzato utilizzando numerose macchine attaccanti che insieme costituiscono una "botnet" controllate da un una unica entità, il botmaster

  - N**TP reflection** (vedi spoofing)

Attacco attivo: sfruttamento delle vulnerabilità

La vulnerabilità può essere intesa come una componente di un sistema, in corrispondenza alla quale le misure di sicurezza sono assenti, ridotte o compromesse, il che rappresenta un punto debole del sistema e consente a un eventuale aggressore di compromettere il livello di sicurezza dell'intero sistema.

La v**ulnerabilità** si presenta ovunque ci sia un difetto di progettazione, codifica, installazione e configurazione del software. Esempi di vulnerabilità: https://csirt.gov.it

Un exploit è un frammento di codice, una sequenza di comandi, o un insieme di dati, che prendono vantaggio da una **vulnerabilità** per acquisire privilegi di accesso, eseguire codice o creare DoS su di una risorsa.

Patch : Quando viene scoperta una vulnerabilità lo sviluppatore rilascia un aggiornamento del software (**patch)**, che porta alla risoluzione della vulnerabilità di sicurezza.

Zero-day è il nome delle vulnerabilità di cui ancora non è stata rilasciata una Patch.

Zero-day exploit si riferisce a exploit di vulnerabilità zero-day.

Attacco attivo: gli exploit

I più comuni tipi di exploit prendono vantaggio da:

- **Buffer overflow (Stack o Heap):** si basa sul fatto che un programma potrebbe non controllare in anticipo la lunghezza dei dati in arrivo, ma si limita a scrivere il loro valore in un buffer di lunghezza prestabilita, confidando che l'utente (o il mittente) non immetta più dati di quanti esso ne possa contenere.

- **Cross site scripring (XSS)** è una vulnerabilità che affligge siti web dinamici che impiegano un insufficiente controllo dell'input nei form.

- **Code injection**: Questo exploit sfrutta l'inefficienza dei controlli sui dati ricevuti in input ed inserisce codice maligno (ad esempio all'interno di una query SQL, oppure attraverso Remote File Inclusion)

Spesso, quando gli exploit vengono pubblicati, la vulnerabilità viene eliminata attraverso una Patch che produce una nuova versione del software.

Vedi ad esempio: https://www.cert.garr.it/alert/security-alerts

Attacco attivo: Malware

Un qualsiasi software creato allo scopo di causare danni a un computer, ai dati degli utenti del computer, o a un sistema informatico su cui viene eseguito.
http://it.wikipedia.org/wiki/Malware

- Virus Sono parti di codice che si diffondono copiandosi all'interno di altri programmi, o in una particolare sezione del disco fisso, in modo da essere eseguiti ogni volta che il file infetto viene aperto.
- Worm questi malware non hanno bisogno di infettare altri file per diffondersi, perché modificano il sistema operativo della macchina ospite in modo da essere eseguiti automaticamente e tentare di replicarsi sfruttando per lo più Internet.
- Trojan software che oltre ad avere delle funzionalità "lecite", utili per indurre l'utente ad utilizzarli, contengono istruzioni dannose che vengono eseguite all'insaputa dell'utilizzatore
- Spyware Software che vengono usati per raccogliere informazioni dal sistema su cui sono installati e per trasmetterle ad un destinatario interessato.

Common Weakness Enumeration (CWE)

CWE https://cwe.mitre.org/ è una lista si debolezze di sicurezza mantenuta da Mitre Corporation https://www.mitre.org/ (organizzazione no-profit) in cui ogni tipologia di debolezza viene classificata attribuendo un identificativo numerico univoco.

Esempi delle debolezze più rilevanti nel 2022

- CWE-787 : Out-of-bounds Write
- CWE-79 : Cross-site Scripting
- CWE-89 : SQL Injection
- CWE-20 : Improper Input Validation
- CWE-125 : Out-of-bounds Read
- CWE-78 : OS Command Injection
- CWE-416 : Use After Free
- CWE-22 : Path traversal

Common Vulnerabilities ean Exposures (CVE)

Mitre Corporation gestisce anche il dizionario CVE (Common Vulnerabilities and Exposures) in cui sono classificate ed enumerate le singole vulnerabilità e falle di sicurezza note pubblicamente.

Esistono diversi elenchi delle vulnerabilità più sfruttate nel 2022.

Ad esempio:

https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-215a

In questo elenco la vulnerabilità più sfruttata nel 2022 è la CVE-2018-13379

Denominata «Fortinet FortiOS SSL VPN Path Traversal Vulnerability»

Appartenente al CWE-22

(Improper Limitation of a Pathname to a Restricted Directory - 'Path Traversal')

Organizzazioni come GARR-CERT diffondono regolarmente avvisi riguardo nuove vulnerabilità:

https://www.cert.garr.it/en/alert-en/security-alerts/archive/listing

CVSS

Il Common Vulnerability Scoring System (CVSS) è una norma tecnica aperta per valutare la gravità delle vulnerabilità di sicurezza di un sistema informatico.

CVSS assegna un punteggio di gravità alle vulnerabilità, consentendo a chi si occupa di rispondere all'emergenza di stabilire la priorità di risposte e risorse in base al livello di minaccia.

I punteggi vengono calcolati con una formula che dipende da diverse metriche che approssimano la facilità e l'impatto di un exploit. Il punteggio è espresso in una scala da 0 a 10, dove 10 indica il livello di vulnerabilità più grave. (Wikipedia)

Esistono diverse versioni di CVSS per calcolare lo score; le versioni più recenti sono CVSS3 e CVSS2.

Diverse organizzazioni calcolano in CVSS per le nuove vulnerabilità, tra cui NIST

Nell'esempio della vulnerabilità CVE-2018-13379 NIST ha calcolato:

CVSS3: 9.8/10 CVSS2: 9.1/10

OWASP Top10

Un'altra lista molto nota di debolezze è la Top10 che l'Open Web Application Security Project (OWASP) stila ogni anno.

La lista del 2021 è la seguente:

A01:2021 Broken Access Control
A02:2021 Cryptographic Failures
A03:2021 Injection
A04:2021 Insecure Design
A05:2021 Security Misconfiguration
A06:2021 Vulnerable and Outdated Components
A07:2021 Identification and Authentication Failures
A08:2021 Software and Data Integrity Failures
A09:2021 Security Logging and Monitoring Failures
A10:2021 Server Side Request Forgery (SSRF)

Attacco attivo: Ingegneria sociale e spam

.

- Ingegneria sociale. lo studio del comportamento individuale di una persona al fine di carpire informazioni utili. http://it.wikipedia.org/wiki/Social_engineering
  - Phishing E' un tipo di ingegneria sociale attraverso la quale un aggressore cerca di ingannare la vittima convincendola a fornire informazioni personali sensibili.

- Spam http://it.wikipedia.org/wiki/Spam Lo spamming è l'invio di messaggi indesiderati (generalmente commerciali). Può essere attuato attraverso qualunque sistema di comunicazione, ma il più usato è Internet, attraverso messaggi di posta elettronica

Il termine SPAM deriva da una scena dei Monty Python https://www.youtube.com/watch?v=zLih-WQwBSc

Metodi e strumenti di difesa

Architettura e politiche di sicurezza

- Perimetro, superficie d'attacco
- Firewall Packet filter e Firewall Proxy (system integrity), IDS, IPS (availability)
- Auditing di sicurezza (utenti, sistemi e reti)

Iniziative di comunità

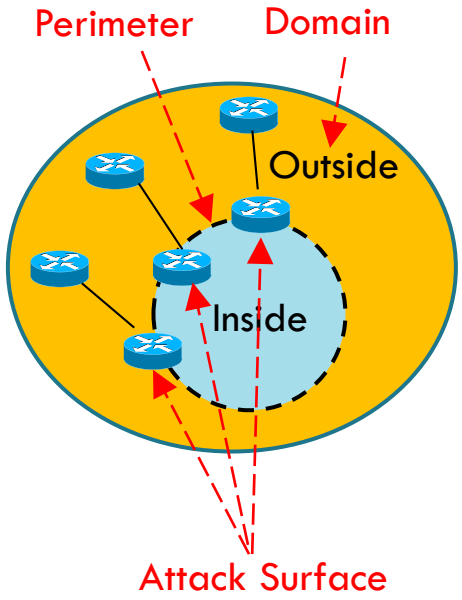
- Vulnerability Alerts e scan
- Computer Emergency Response Team (CERT)
- Normative

Rinforzo di autenticazione e riservatezza : strumenti crittografici

- Tools: crittografia simmetrica e asimmetrica, Message Digest, certificati
- Servizi: autenticazione (authentication, non repudiation), cifratura (data confidentiality), firma digitale (Data Integrity).

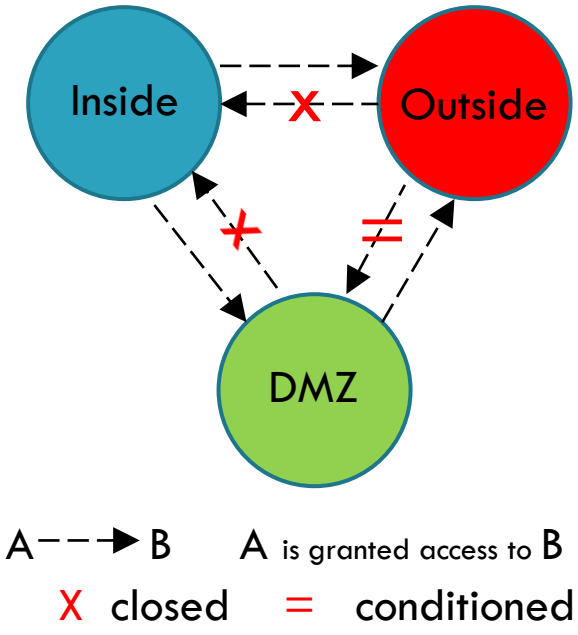
Domains, perimeter and attack surface

- A **security domain** is a set of entities/resources to be managed as a unique administration area according to a common security policy (security enforcement rules)
- A **security perimeter** is the secured boundary between the external and internal side of a security domain
  - e.g., an internal network and its public facing side, typically the Internet
  - The perimeter can be protected by several security devices
- The **attack surface** of a security domain is the sum of the different points ("attack vectors") where an unauthorized entity ("attacker") can try to enter data to or extract data or do any kind of unauthorized or hostile activity.
  - **Keeping the attack surface as small as possible** is a fundamental basic security measure

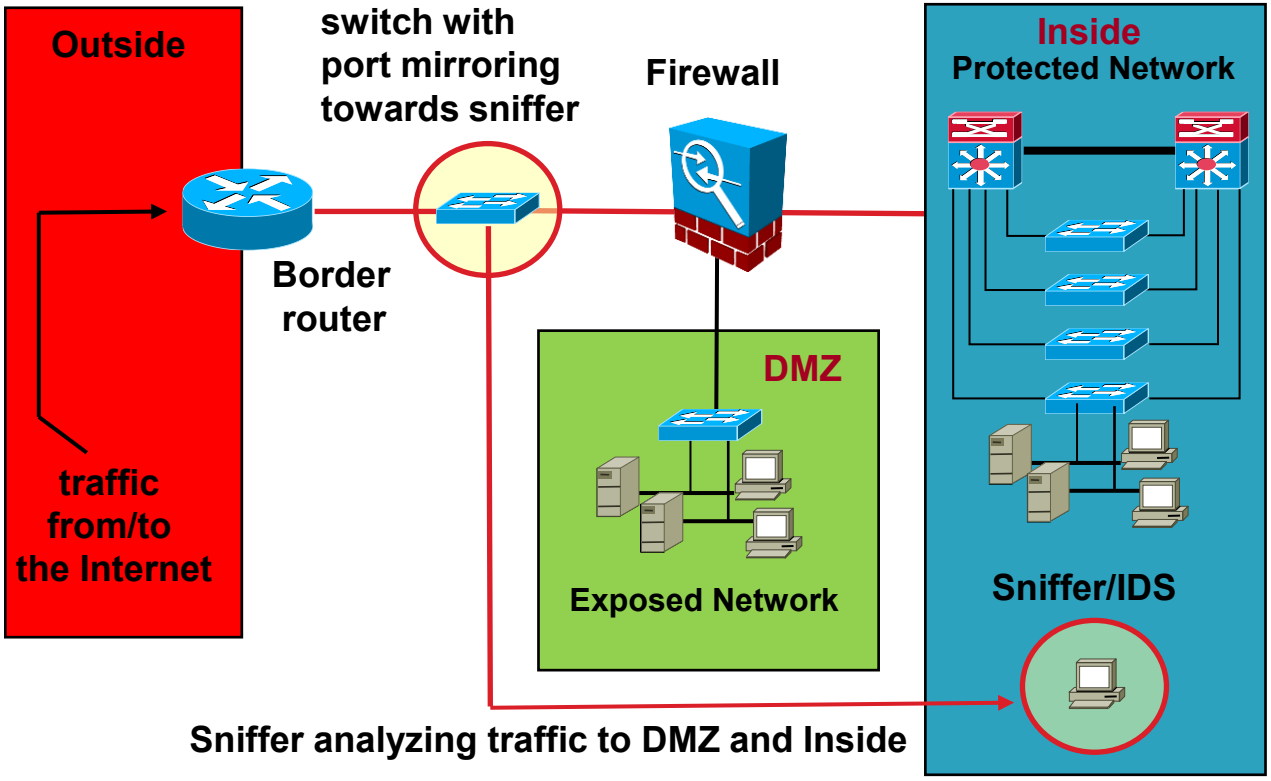


Security Domains

- Each security domain is assigned a **degree of trust** or **security level**
- Such degree defines and characterizes its visibility rules (access rights) with respect to the others
  - A domain with a higher degree of trust can have fuller visibility than those with a lower degree
  - Vice versa, visibility is blocked unless specific exceptions (filtering / visibility rules) are defined
    - DMZ and INSIDE have full visibility of OUTSIDE
    - INSIDE has full visibility of DMZ
    - Any other access is not granted



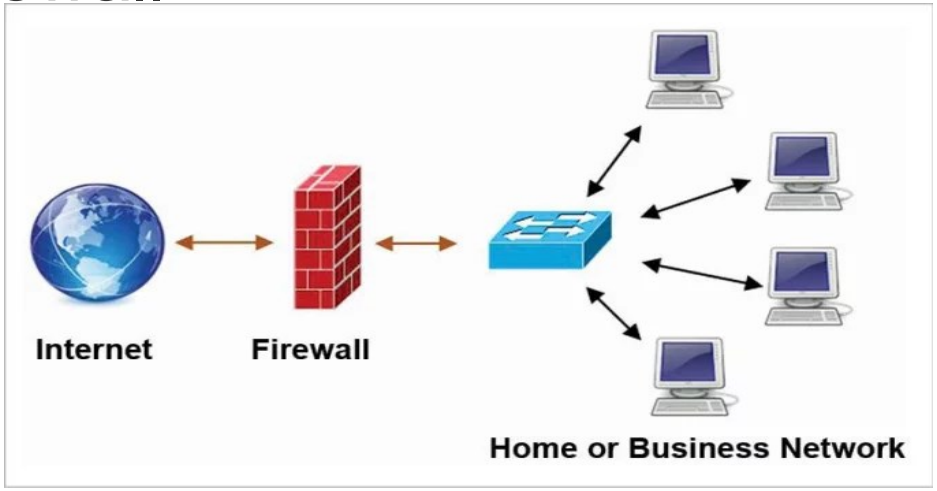
Basic security architecture



- In a common network architecture there are at least three domains:
  - **Outside** (all the world outside - the Internet): trust degree 0
  - **Inside** (the internal organization to be protected and hidden): degree of trust 100
  - **DMZ** (the set of internal machines that expose services outside): degree of trust 0 <x <100

Firewall

Per firewall si intende una entità hardware o software che si pone tra internet e la rete (o l'host) che si vuole proteggere.



Il firewall svolge una funzione di filtro, consentendo il transito solamente alle connessioni ritenute lecite mediante una opportuna “Policy”. Obiettivi principali:

- Monitorare, limitare, autenticare l'accesso alla rete da proteggere nei confronti di accessi provenienti dall'esterno (Internet).
- Monitorare, limitare, autenticare l'accesso all'esterno (Internet) da parte dell'utenza interna.

Servizi di sicurezza: system integrity, availability, audit.

I firewall possono essere di 2 tipi:

- I packet filter: agiscono ai livelli 3 e 4

- I proxy: agiscono a livello applicazione.

Firewall a filtro di pacchetti

Questo filtro analizza tutti i pacchetti in transito e applica azioni del tipo permit/deny sulla base di politiche basate sugli indirizzi IP e le porte di provenienza e/o di destinazione.

Obiettivi:

- Rendere visibili ad internet solamente i servizi di rete destinati ad un accesso pubblico (protezione dei servizi intranet e dei servizi “inconsapevoli”)
- Bloccare il traffico indesiderato (es: P2P, ..)
- Strumento per la gestione delle emergenze (bloccare un host ostile o contaminato da virus).

Agisce a livello di pacchetti IP, ma deve leggere anche i primi byte del livello 4 per leggere le porte TCP o UDP.

Può essere realizzato dai Router mediante un modulo aggiuntivo o da HW specifico.

Per Linux esiste il modulo **IPtables** che può essere applicato ad una interfaccia di rete.

IPtables

Il pacchetto software IPtables consente di applicare ACL per il Packet filtering sulle interfacce di Sistemi Linux. IPtables lavora sulle 3 tabelle filter, nat, mangle sulle quali possiamo creare catene (chains) di regole ACL.

La tabella **Filter** (tabella di default) server per il packet filter

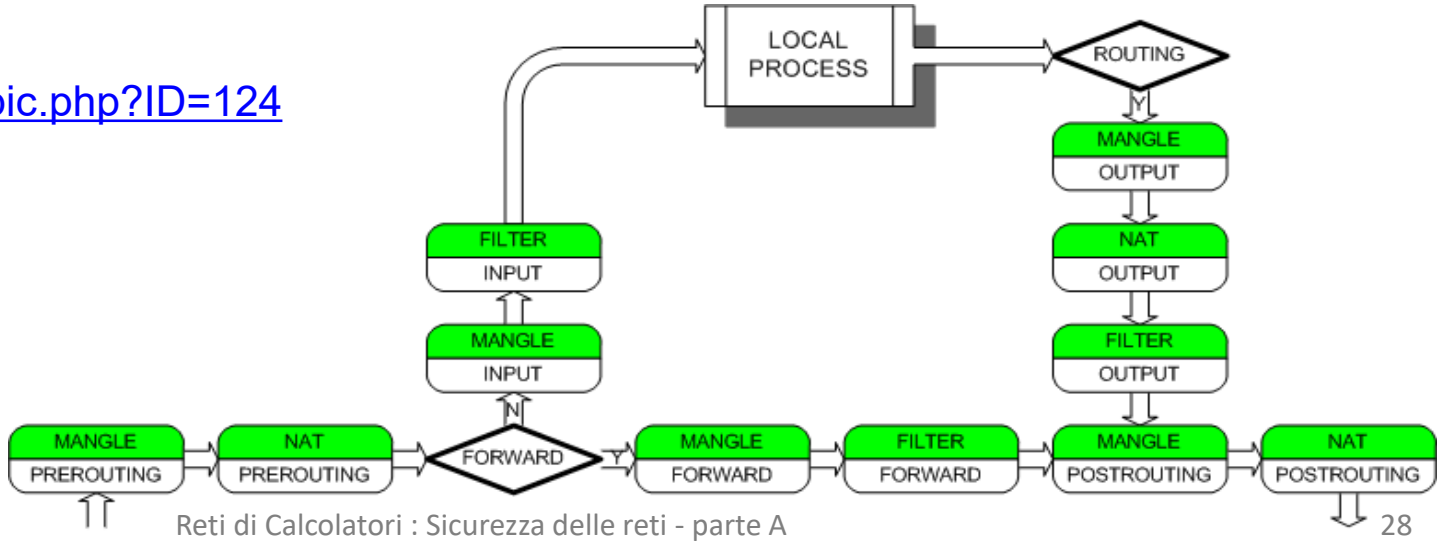
Le tabelle **NAT** (SNAT e DNAT) servono per le regole di NATting

La tabella **Mangle** server per modificare alcuni parametri nell'header pacchetto

La tabella **Filter** ha 3 catene di Default applicate su una interfaccia di rete:

- **INPUT** per il processamento dei pacchetti destinati all'host
- **OUTPUT** per i pacchetti provenienti dall'host
- **FORWARD** per i pacchetti che devono attraversare il firewall.

Riferimenti:
http://openskill.info/topic.php?ID=124



Esempi IPtables

#accetta i pacchetti entrati di connessioni gia stabilite (SYN=0)
iptables -A INPUT -p ALL -m state --state ESTABLISHED -j ACCEPT

#accetta i pacchetti entranti dei servizi interni http, 81 (vh1), ssh e 3000 (ntopng)
iptables -A INPUT -p tcp --dport http -i enp0s3 -j ACCEPT
iptables -A INPUT -p tcp –dport 81 -i enp0s3 -j ACCEPT
iptables -A INPUT -p tcp –dport 3000 -i enp0s3 -j ACCEPT
iptables -A INPUT -p tcp --dport ssh -i enp0s3 -j ACCEPT

#accetta e registra sul sistema di log i pacchetti entranti verso il servizio telnet
iptables -A INPUT -p tcp --dport 23 -i enp0s3 -j LOG --log-prefix "TELNET"

#tutto il resto viene scartato
iptables --policy INPUT DROP

iptables -nvL

Firewall Basati su Proxy

Il Proxy è un programma applicativo con funzione di tramite tra client e server.
In modo implicito o esplicito il client deve rivolgersi al proxy per poter raggiungere il server.
Occorre un Proxy specifico per ogni applicazione.

Obiettivi:

- Mettere in comunicazione client e server che non hanno visibilità diretta (ad esempio se il client è in una rete intranet)
- Migliorare le prestazioni (es: Web Caching)
- Servizi di sicurezza
  - Auditing (tracciamento delle attività)
  - Autenticazione e Autorizzazione

Esempi:

- **Proxy Web** (Vedi Squid https://it.wikipedia.org/wiki/Squid)
- **Proxy SMTP** (MTA con **antiVirus e antiSpam** posizionato all'ingresso/uscita della LAN)
- **socat** (https://linux.die.net/man/1/socat) è un tool command-line che consente di connettere due flussi TCP e può quindi essere utilizzato per realizzare un servizio Proxy.

Antivirus

L'Antivirus (AV) è un software atto a prevenire, rilevare ed eventualmente eliminare programmi dannosi. Un AV ha anche una funzione preventiva, impedendo che un virus possa entrare in un sistema ed infettarlo.

Il metodo più utilizzato per individuare virus in un file è attraverso le signatures (firma).

Il programma AV calcola la firma (signature) di un file da analizzare (e.g. Hash MD5 dell'intero file) e lo confronta con le firme di virus noti presenti all'interno di un archivio. Se la firma corrisponde il file è sicuramente un virus.

Esistono anche tecniche euristiche, di solito usate in modo complementare alle firme, che cercano di individuare virus non noti all'AV attraverso la ricerca di pattern sospetti.

Può essere installato

- sul PC: scan dei dischi dell'host e dei nuovi file salvati
- sul MailServer: scan delle mail in entrata e in uscita

The best antivirus software for Windows Home User:

https://www.av-test.org/en/antivirus/home-windows/

Tecniche Antispam

Link utili: http://it.wikipedia.org/wiki/Spam

Black list https://www.cert.garr.it/it/documentazione/articoli-tecnici/30-blacklist

Lista di server classificati spammers che viene attivata sul mail server rifiutando mail che provengono da host inclusi in questa lista. L'amministratore del mailserver può costruire manualmente una propria lista o può avvalersi di servizi in Internet che distribuiscono automaticamente le liste.

Gray-List: Si basano sul fatto che i mailer usati dagli spammer generalmente tentano l'invio di una email una sola volta: Il GrayListing consiste nel rigetto della ricezione della mail al primo tentativo, che verrà accettata ad un successivo tentativo, dopo un tempo stabilito (tipicamente 300 sec.)

White List. Liste di mittenti “Fidati” su cui non vengono effettuati controlli antispam. Include gli host accettati da Gray-list e host inseriti manualmente dall'amministratore.

Filtri Bayesiani: Sono filtri che cercano di classificare le mail in arrivo assegnando un punteggio numerico a frasi o modelli che si presentano nel messaggio. Ogni messaggio riceve quindi un punteggio compressivo (tra 0 e 1) che, dopo aver stabilito una soglia, ci consente di classificare il messaggio. Il filtro richiede un addestramento con mail spam e no-spam con cui viene creato un database di riferimento.

Esempi:

- Spamassassin http://spamassassin.apache.org/ (liste White e Black, filtri Bayesiani)

IDS (Intrusion Detection System)

IDS è un dispositivo software/Hardware per identificare accessi non autorizzati a host o LAN.

L'IDS generalmente si appoggia su un Data-Base per memorizzare le regole utilizzate per individuare le violazioni di sicurezza.

Gli IDS sono classificabili nel seguente modo:

Host IDS (HIDS): analizzano file di log e file system sull'Host.

TRIPWIRE è un esempio di HIDS. Si basa sulla differenza tra lo stato analizzato ed uno stato iniziale.

Network IDS (NIDS): analizzano il traffico di rete.

SNORT è un esempio di NIDS che può funzionare anche come sniffer / packet logger.

Quando un IDS rileva una intrusione invia una notifica all'amministratore via e-mail o con un messaggio alla console (continuous monitoring e auditing).

Intrusion Prevention System (IPS)

Gli IPS sono un'estensione degli strumenti di IDS: quando rilevano un tentativo di intrusione sono abilitati a bloccare gli accessi considerati pericolosi.

IPS può mandare un allarme (come un IDS), ma anche interagire con un firewall per eliminare pacchetti malevoli, resettare le connessioni e/o bloccare il traffico da un indirizzo IP attaccante.

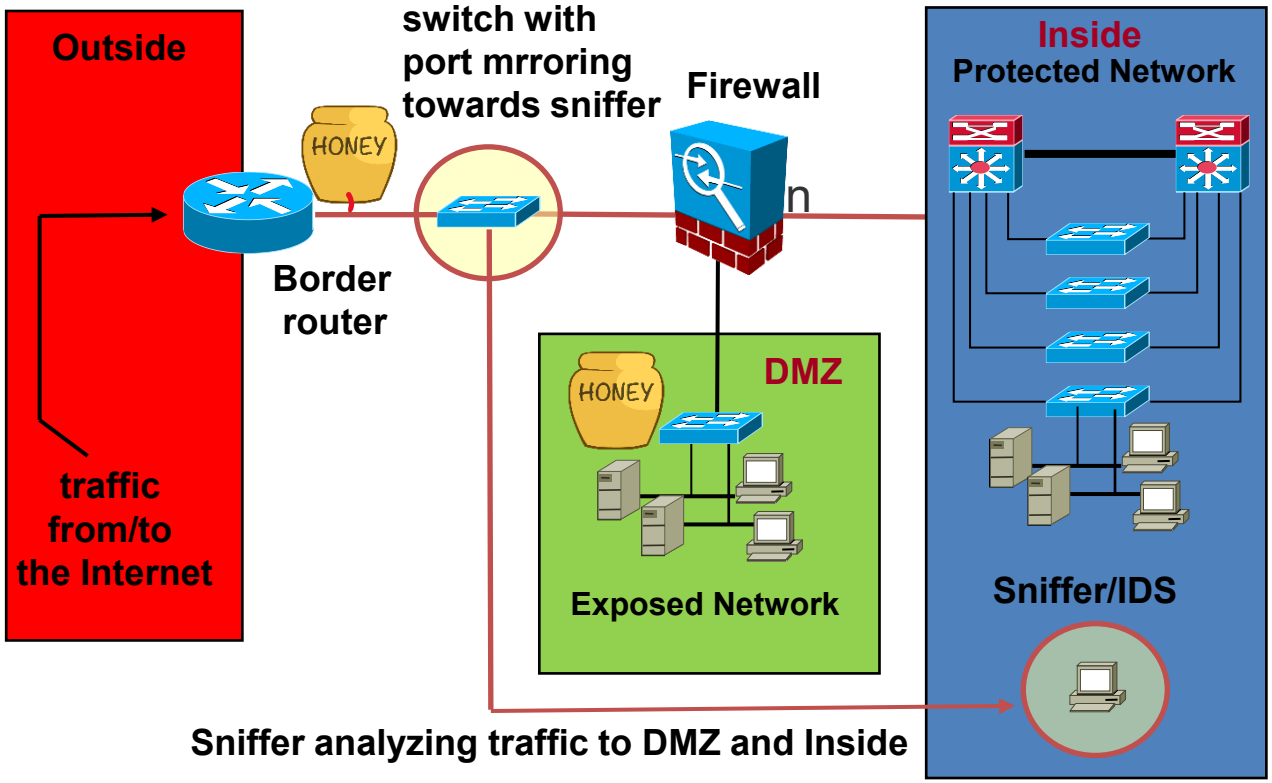
Strumenti utili: **fail2ban** http://guide.debianizzati.org/index.php/Fail2ban

Fail2ban è pensato per prevenire attacchi “brute force” via ssh bloccando temporaneamente gli indirizzi IP che provano a violare la sicurezza di un sistema.

Il programma effettua il parsing di alcuni file di log che contengono informazioni relative ad accessi falliti. Se il numero di accessi falliti supera una certa soglia l’indirizzo IP del client viene bloccato attraverso una regola di iptables.

Honeypot

Un honeypot è un sistema o componente hardware o software usato come "trappola" o "esca" a fini di protezione contro gli attacchi. Consiste in un computer o un sito che sembra essere parte della rete, ma che in realtà è ben isolato e non ha contenuti sensibili o critici.
Consente di rilevare attività malevole senza coinvolgere la rete interna.



Auditing di sicurezza informatica

Un audit di sicurezza informatica è un'analisi sistematica con l'obiettivo di verificare i controlli di sicurezza, politiche e procedure in atto e convalidare il loro efficace funzionamento in base al rischio.

Gli elementi oggetto di valutazione di un audit di sicurezza informatica sono:

- **personale aziendale**, ovvero le modalità utilizzate dai dipendenti per raccogliere, condividere e archiviare informazioni sensibili.
- I componenti del **sistema informatico** e ambiente in cui è ospitato
- **applicazioni e software** (incluse le patch di sicurezza messe a punto dagli amministratori di sistema)
- **vulnerabilità della rete** interna ed esterna

Passaggi principali dell'auditing:

- Definizione degli asset: elenco dettagliato di risorse, dati sensibili, apparecchiature informatiche e relativa valutazione del rischio informatico
- Valutazione del personale
- Monitoraggio dei sistemi e delle reti
- Identificazione delle vulnerabilità
- Risposte al rischio, aumento delle protezioni

Auditing del personale

La sicurezza dei sistemi e delle reti dipende anche dal comportamento degli utenti e degli amministratori di sistema. Occorre annotare e registrare quali dipendenti abbiano accesso alle informazioni sensibili e quanti di loro siano preparati in maniera adeguata.

Il rischio principale riguarda il furto delle credenziali per l'accesso ai sistemi aziendali.

La password è una delle forme di identificazione più semplici ed utilizzate ed è quindi uno dei principali bersagli degli attacchi.

I metodi più utilizzati sono:

1) Intercettazione. L'utilizzo di un canale non cifrato consente la cattura delle password sulla rete.
2) Furto. Alcuni utenti tendono a scriverla su un supporto magnetico per non dimenticarla.
3) Tentativi di indovinare la password (password cracker basati su dizionari)
4) Phishing.

I risultati dell'audit sul personale vengono utilizzati per programmare un **piano di formazione del personale**

Auditing di sistema

L'audit di sicurezza informatica richiede la definizione dell'elenco dei sistemi, la loro classificazione in base al rischio informatico e l'attribuzione del ruolo di amministratore.

L'amministratore di sistema deve definire un programma di audit attraverso il quale viene definita la gestione e l'analisi degli eventi, con i seguenti obiettivi:

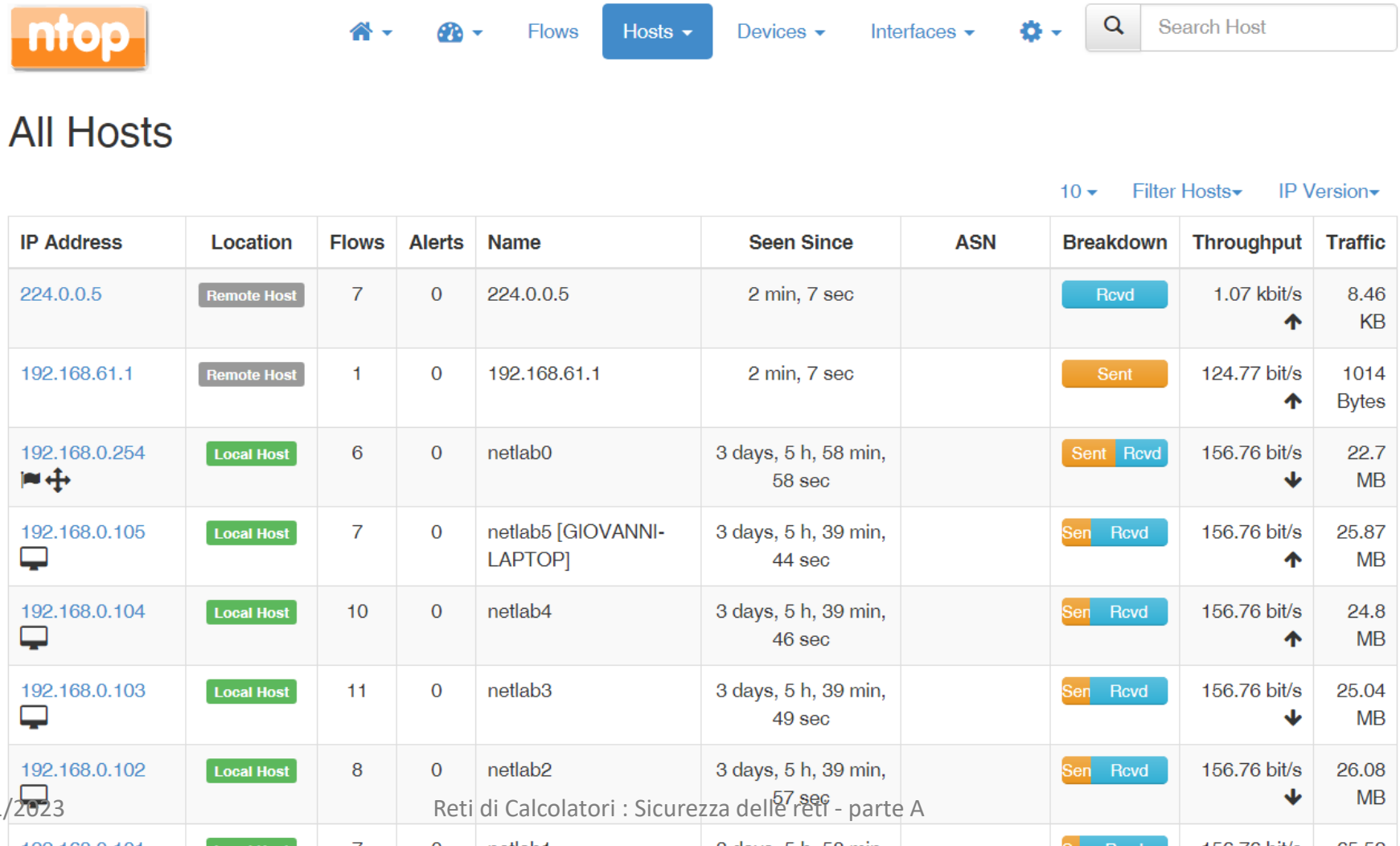
- **(Early) warning:** Individuare rapidamente eventuali attacchi in corso.
- **Trouble-shooting:** mantenere uno storico degli eventi per tracciare le attività.

rsyslog è lo strumento base per l'audit di sistema in ambiente Linux.

Consente ai processi interni di generare eventi classificati in categorie (KERN, USER, MAIL, DAEMON, AUTH, LPR, CRON, LOCAL0-7) e priorità (EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO, DEBUG). Per ogni evento è possibile definire una azione come scrivere su file (tipicamente nella directory /var/log) , inviare mail o attivare script.

Audit della rete

Consiste nella raccolta e analisi sistematica del traffico di rete e per la rilevazione in tempo reale di minacce provenienti dalla rete. Strumenti utili sono i **Network Monitor** come **ntop** , dotato di una console web, vedi figura.



IP Address	Location	Flows	Alerts	Name	Seen Since	ASN	Breakdown	Throughput	Traffic
224.0.0.5	Remote Host	7	0	224.0.0.5	2 min, 7 sec		Rcvd	1.07 kbit/s ↑	8.46 KB
192.168.61.1	Remote Host	1	0	192.168.61.1	2 min, 7 sec		Sent	124.77 bit/s ↑	1014 Bytes
192.168.0.254	Local Host	6	0	netlab0	3 days, 5 h, 58 min, 58 sec		Sent Rcvd	156.76 bit/s ↓	22.7 MB
192.168.0.105	Local Host	7	0	netlab5 [GIOVANNI-LAPTOP]	3 days, 5 h, 39 min, 44 sec		Sen Rcvd	156.76 bit/s ↑	25.87 MB
192.168.0.104	Local Host	10	0	netlab4	3 days, 5 h, 39 min, 46 sec		Sen Rcvd	156.76 bit/s ↑	24.8 MB
192.168.0.103	Local Host	11	0	netlab3	3 days, 5 h, 39 min, 49 sec		Sen Rcvd	156.76 bit/s ↓	25.04 MB
192.168.0.102	Local Host	8	0	netlab2	3 days, 5 h, 39 min, 57 sec		Sen Rcvd	156.76 bit/s ↓	26.08 MB

Auditing di rete

Identificazione delle vulnerabilità

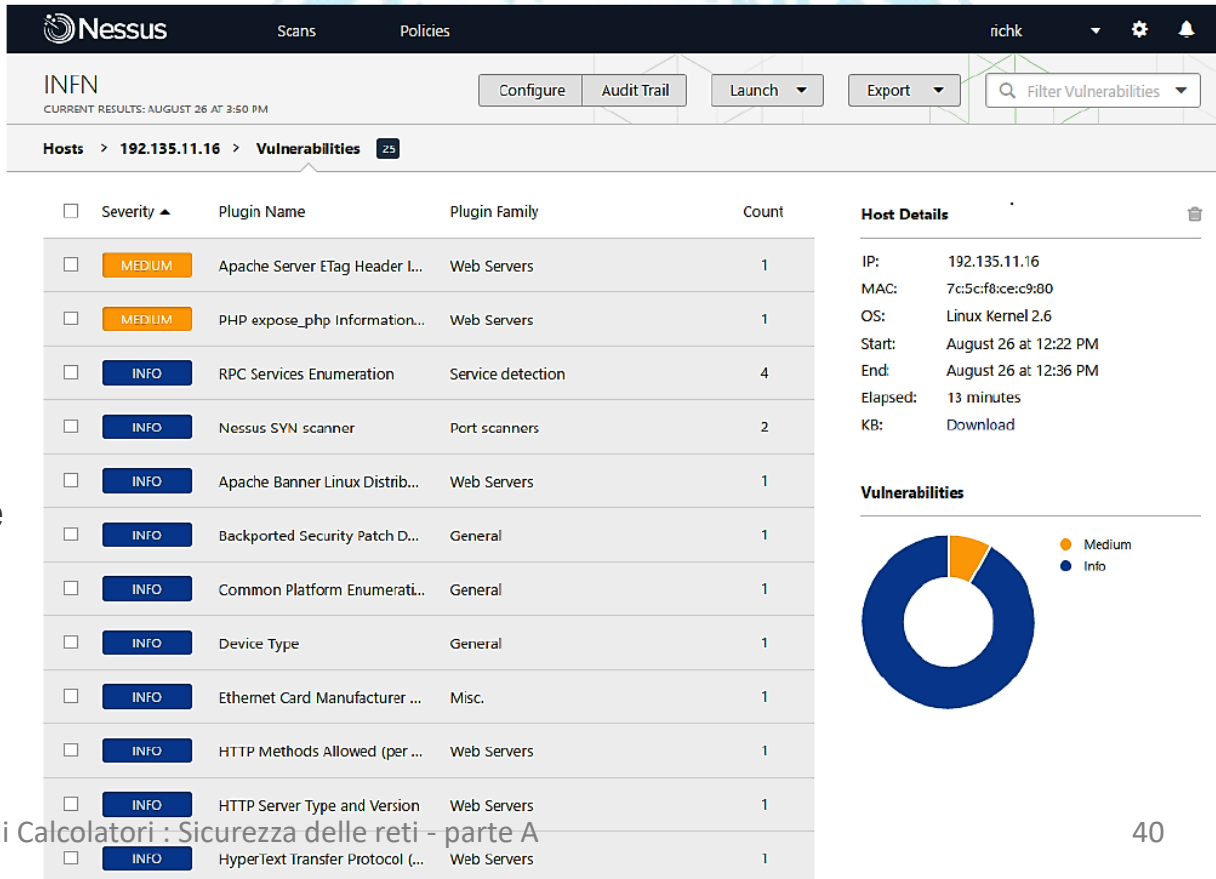
Un vulnerability scanner è un programma progettato per ricercare e mappare le debolezze di un singolo computer o degli host di una rete.

Identifica le vulnerabilità dovute a software con bugs o non aggiornato, configurazioni errate all'interno di servizi applicativi, web server, firewall router, ecc

Scanner più utilizzati: Nessus e Qualys (commerciale) - openVAS (opensource).

Funzionamento:

1 ricerca host attivi
2 port scanning per ogni host
3 ricerca vulnerabilità tramite test non invasivi
4 identificazione vulnerabilità tramite confronto con database
5 Generazione di un report



Auditing di rete

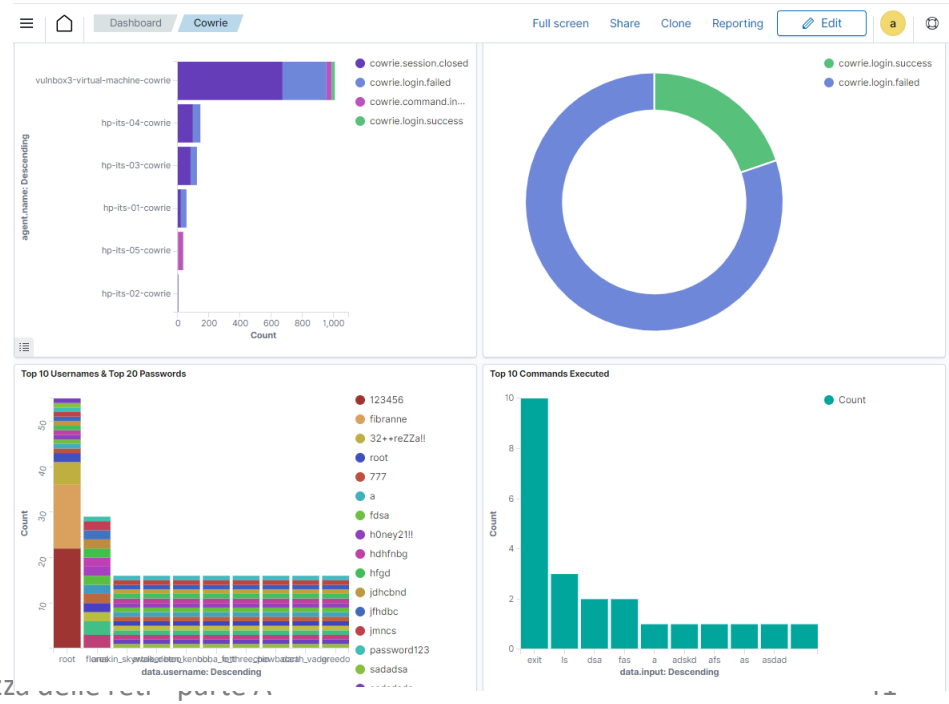
Cyber Threat Intelligence (CTI)

La Threat Intelligence, consiste nell'attività di raccolta di informazioni riguardo potenziali minacce provenienti da varie fonti; i dati vendo memorizzati, correlati e processati da processi automatici di Threat intelligence che possono generare allarmi (alert) o attivare delle azioni (Active Response).

Esempi di sorgenti delle informazioni possono essere EDR (Endpoint Detection and Response) ovvero agenti su dispositivi quali PC, , oppure Honeypot , network monitor e NIDS (Network Intrusion Detection System).

Il collettore e analizzatore dei dati è tipicamente un **SIEM** (Security Information and Event Management).

L'organizzazione delle fonti e la gestione del SIEM può essere operata da un centro operativo denominato **SOC** (Security Operation Center).



UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE ED INFORMATICHE
Corso di Laurea in Informatica

Sicurezza delle reti – Parte B

Crittografia applicata e OpenSSL

RETI DI CALCOLATORI - a.a. 2023/2024

Roberto Alfieri

La sicurezza delle reti: sommario

PARTE A

- I servizi di sicurezza
- Metodi e strumenti di attacco
- Metodi e strumenti di difesa

PARTE B

- Crittografia applicata e OpenSSL

PARTE C

- Protocolli di autenticazione
- IPsec
- VPN
- Sicurezza delle reti WiFi

Crittografia

La crittografia è lo studio dei metodi per rendere un messaggio “ofuscato” in modo da renderlo comprensibile solo a persone a cui il messaggio è destinato (servizio di sicurezza **Confidenzialità**).

L'algoritmo che esegue la cifratura o decifratura è detto **Cipher.**

La **cifratura E** (Encryption) si applica ad un **messaggio in chiaro P (Plaintext)** per ottenere un **testo cifrato C (Ciphertext). C=E(P)**

La **decifratura D** (decryption) si applica ad un **messaggio cifrato C** per ottenere il **testo in chiaro P. P=D(C)**

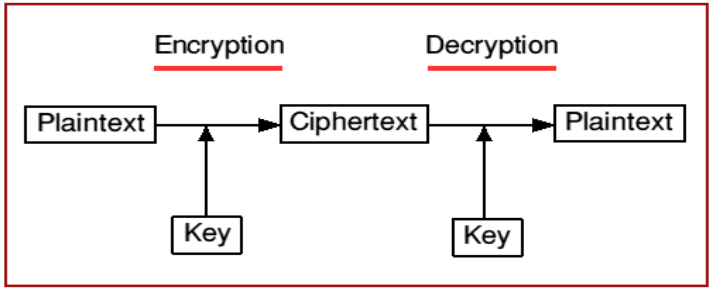
La **crittanalisi** è lo studio dei metodi per ottenere il significato di un testo cifrato violando la tecnica di cifratura del Cipher per verificarne la robustezza.

La robustezza di un Cipher non si basa sulla segretezza degli algoritmi di cifratura, ma sull'utilizzo di **chiavi segrete k** utilizzate nella fase di cifratura e decifratura.

Se gli algoritmi utilizzano la stessa chiave per cifrare e decifrare sono detti a **chiave simmetrica**, altrimenti sono detti a **chiave pubblica.**

A chiave simmetrica C=E(P,k) P=D(C,k)

A chiave pubblica C=E(P,k1) P=D(C,k2)



SSL/TLS

SSL (Secure Socket Layer) protocollo sviluppato originariamente dalla Netscape per fornire autenticazione e privacy in una connessione tra un client e un server mediante la l'uso di diverse tecnologie crittografiche.

Le principali tecnologie sono:

- Algoritmi crittografici (a chiave simmetrica o a chiave pubblica)
- Message Digest e firma digitale
- Certificati e Certification Authority
- Security Socket Layer (SSL) e Transport Layer Security (TLS)

Versioni di SSL e TLS:

- SSL v2.0 (1995): prima versione implementata da Netscape (deprecata dal 2011)
- SSL v3.0 (1996): revisione della precedente (deprecata dal 2015)
- TLS 1.0 (1999): RFC 2246. Revisione di SSL 3.0 (deprecata dal 2021)
- TLS 1.1 (2006): RFC 4346. (deprecata dal 2021)
- **TLS 1.2** (2008): RFC 5246.
- **TLS 1.3** (2018): RFC 8446 (attuale)

Vedi “SSL/TLS Strong Encryption: An Introduction” https://httpd.apache.org/docs/2.4/ssl/ssl_intro.html

Implementazioni del protocollo SSL/TLS

OPENSSL (http://www.openssl.org/)

E' una libreria Open Source basata sulla libreria SSLeay sviluppata da Eric Young e Tim Hudson.

Fino alla versione openSSL1.1 si applica la licenza originale SSLeay più restrittiva e incompatibile rispetto a GPL.

Dalla versione openSSL3.0 si applica la Apache License v2 (https://www.openssl.org/source/license)

Fornisce:

- **Un ToolKit** per l'utilizzo a linea di comando delle API.
- **La Libreria SSL** per la programmazione di canali cifrati con SSL/TLS.
- **La Libreria Crypto** per la programmazione di diversi algoritmi crittografici tra cui la cifratura a chiave simmetrica, a chiave pubblica, certificati e funzioni di Hash.

GnuTLS (http://www.gnutls.org/)

Creato per consentire alle applicazioni del Progetto GNU di usare una libreria compatibile con la GPL

Licenza GPL e LGPL

Fornisce API di programmazione (C/C++ Python PHP) e alcune utilities di supporto.

OpenSSL ToolKit

Il Toolkit consente l'utilizzo a linea di comando di tutte le operazioni crittografiche della libreria.
Sintassi:

```
openssl comando [opzioni dei comandi]
```

Ad esempio per vedere la versione di openssl in uso:

```
openssl version
```

Documentazione:

man <comando>	https://www.openssl.org/docs/manpages.html
wiki	https://wiki.openssl.org/index.php/Main_Page
OpenSSL Command-Line HOWTO:	http://www.madboa.com/geek/openssl/

Principali Comandi:

Cipher simmetrici: *enc (base64 bf des des3 idea rc4 rc5)*

Crittografia a chiave pubblica: *rsa rsautl gendsa genrsa*

Certificati X.509: *x.509 (gestione dati dei certificati), ca (per utilizzare le funzioni di Ceritification Authority), req (richiesta di certificati), verify (verifica di certificati), crl (Certificate Revocation List), crl2pkcs7 pkcs7(conversione tra diversi formati di certificati),*

Message digest *: dgst (md2, md5, rmd160, sha, sha1)*

Altro*: ciphers -v (elenco ciphers supportati), speed (benchmarking), prime (numeri primi) e rand, password, smime, s_client, s_server*

Comandi OpenSSL: base64

Viene utilizzata per codificare una sequenza binaria in una sequenza di caratteri ASCII.

La sequenza binaria è suddivisa in blocchi di 6 bit ciascuno dei quali viene trasformato in un carattere ASCII mediante un alfabeto dei 64 caratteri [a-zA-Z0-9./]

Vedi https://wiki.openssl.org/index.php/Base64

```
$ openssl   base64 -e  -in toencrypt.txt –out toencrypt.b64 (-e default)
$ openssl   base64 -d  -in toencrypt.b64 -out toencrypt.txt

$echo "encode me" | openssl base64 > encryped.b64
$openssl   base64 -d -in encrypted.b64
```

Comandi OpenSSL: prime e rand

OpenSSL contiene routine per trattare numeri primi e numeri random
(poiché questi servono per le tecniche crittografiche).

Vedi: https://wiki.openssl.org/index.php/Random_Numbers

Esempi:

```
# Testare se uno o più numeri sono primi:
$ openssl prime 11905475924546O753
> 1A6F7AC39A53511 is not prime
$ for N in $(seq 100 300); do openssl prime $N; done

# Scrive un numero random di 128 byte su stdout codificato in base64
$ openssl rand -base64 128

# Scrive un numero random di 32 bits (4 bytes) su un file
$ openssl rand -out random-data.bin 4
```

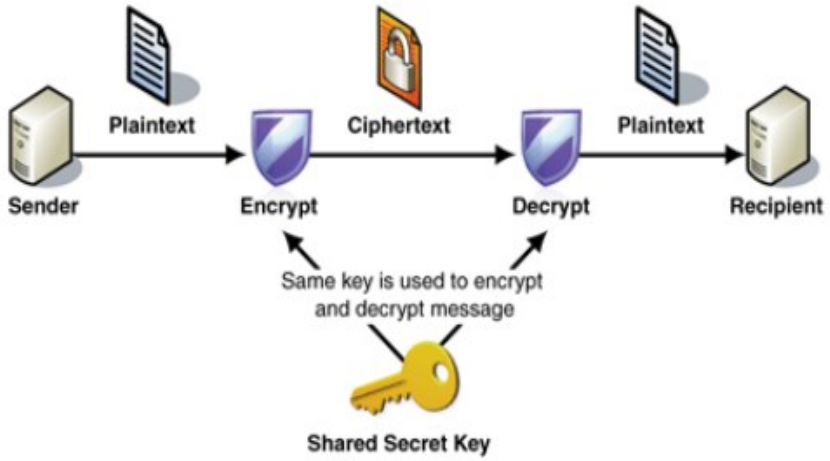
Esempio: programma in prime.c che usa le funzioni BN (BigNum) della libreria Crypto per generare numeri primi random da 1024 bit

http://didattica-linux.unipr.it/~roberto.alfieri@unipr.it/matdid/RETI/security/prime.c

Crittografia a chiave Simmetrica

E' una tecnica crittografica a chiave condivisa che consente di ottenere la cifratura di un messaggio P $C=E_K(P)$ in modo che la stessa chiave possa essere utilizzata per decifrarlo $P=D_K(C)$

Gli algoritmi E e D (Cipher) sono noti. La complessità della cifratura è data da K la cui lunghezza determina l'ampiezza dello spazio delle possibili codifiche di P e di conseguenza la robustezza della cifratura.

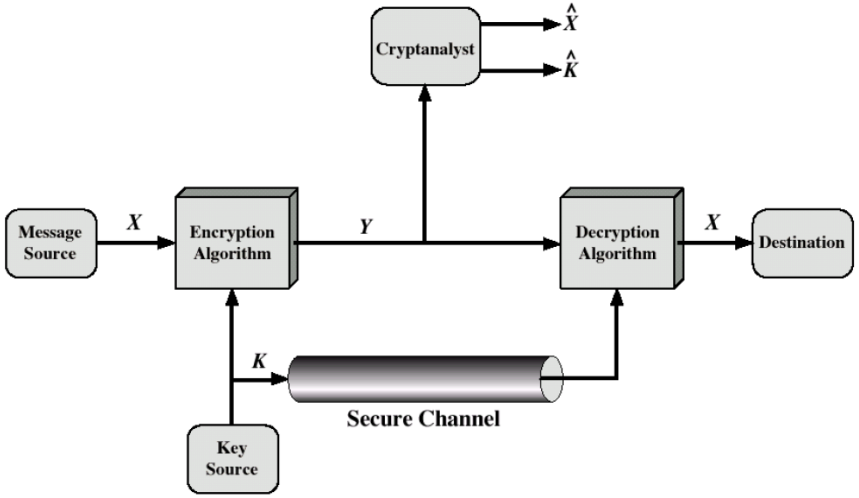


Algoritmi di cifratura molto performanti e semplici da implementare.

Crittografia a chiave Simmetrica: criticità

1) E' necessario un canale (o un metodo) sicuro per **distribuire la chiave tra le due parti.**

Generalmente lo scambio avviene attraverso algoritmi a chiave pubblica, più complessi sia da implementare che da eseguire, ma che permettono questo scambio in modo sicuro.



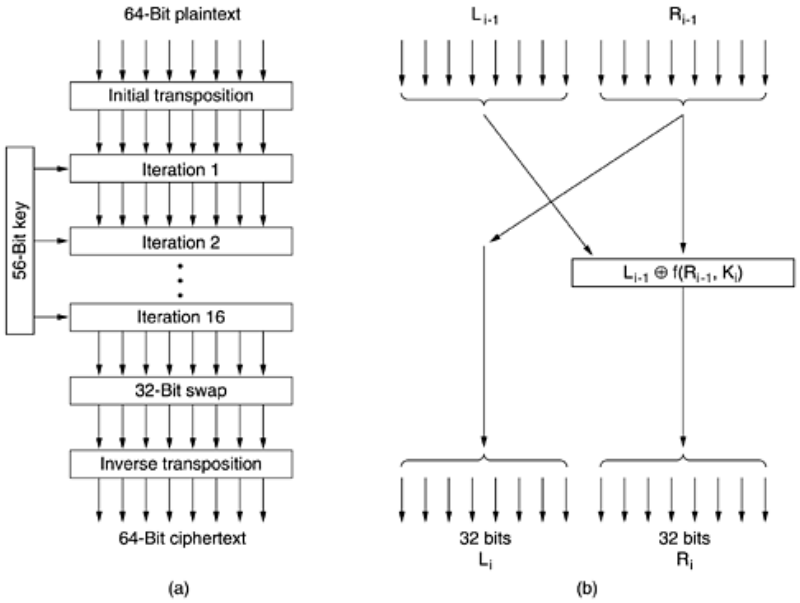
2) Il cipher può comunque essere compromesso con **un attacco a forza bruta**, ovvero testando tutte le possibili chiavi del cipher.

La protezione da questo tipo di attacco si ottiene aumentando la lunghezza del **numero di bit della chiave** e conseguentemente lo spazio delle possibili chiavi da testare, in modo che il costo per forzare il cipher ecceda il valore del dato cifrato.

Cipher Simmetrici

OpenSSL supporta tutti i principali Cipher Simmetrici, tra cui:

DES: (Data Encryption Standard) http://it.wikipedia.org/wiki/Data_Encryption_Standard
sviluppato dall'IBM e adottato dal governo USA nel 1977. Lavora **su blocchi** del messaggio di 64 bit, con chiave di 64 bit di cui 8 sono di parità dispari, pertanto la chiave è di 56 bit.

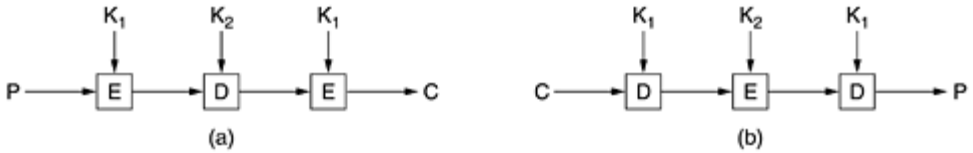


(a) struttura generale (b) dettaglio di una iterazione

La lunghezza della chiave è troppo breve e l'algoritmo può essere forzato in poche ore.

Cipher Simmetrici

TripleDES: E' l'algoritmo DES applicato 3 volte. Può utilizzare 1, 2 o 3 chiavi DES. Esempio con 2 chiavi EDE (K1 e K2, totale 112 bit):



AES: Nel 1997 il NIST (National Institute of Standard and Technology) promosse un concorso per la realizzazione di un nuovo Cipher Simmetrico che sarebbe diventato uno standard del Governo USA con il nome di AES (Advanced Encryption Standard).

L'algoritmo vincitore fu Rijndael (V. Rijmen e J. Daemen).

L'algoritmo supporta chiavi da 128, 192 o 256 bit su blocchi da 128 bit.

RC4: (Rivest 1987) è un algoritmo utilizzato ampiamente in protocolli quali il WEP (WiFi) e SSL.

Comandi OpenSSL: enc

Questo comando serve per cifrare/decifrare utilizzando uno tra i cipher simmetrici supportati che sono circa 50, tra cui

-des -des3 (3 chiavi EDE) -aes128 -aes192 -aes256 -rc4

Vedi la pagina Wiki: https://wiki.openssl.org/index.php/Enc

Esempio: DES3

```
$ openssl enc -e -a -des3 -in myfile      -out myfile.des3
$ openssl enc -d -a -des3 -in myfile.des3 -out myfile
( -a    determina l'output codificato BASE64 )
```

Message Digest

Il **Message Digest** (MD)

è una sequenza di bit di lunghezza limitata e fissa associata ad un messaggio (P) e ne rappresenta la “firma” (o “impronta”).

Il MD **non è invertibile,** ovvero non è possibile risalire al messaggio originale.

Se il massaggio cambia anche di un solo bit il MD diventa completamente diverso.

Il MD viene calcolato applicando al messaggio una **funzione di Hash: MD=H(P)**

L'algoritmo deve essere “**Collision Free**” , ovvero deve evitare (o minimizzare) la possibilità che 2 messaggi generino lo stesso MD. Per questo il MD non può essere troppo breve.

Applicazioni Principali:

- Verificare l'integrità di messaggi o file. Il messaggio P viene spedito assieme al MD; chi li riceve ricalcola i MD e lo compara con quello ricevuto.

- Verifica della password. Viene memorizzato il MD della password in chiaro MD=H(clear_passw). Per verificare la password bisogna confrontare H(proposed_clear_passw) con MD.

Comandi OpenSSL: dgst

Principali Digest:

MD5 è un algoritmo di Hashing a 128 bit realizzato da R. Rivest nel 1991 (RFC1321).

RIPEMD è una famiglia di algoritmi sviluppati come alternativa Europea a MD5
Il più importante e' **ripemd160** http://en.wikipedia.org/wiki/RIPEMD

SHA (Sechure Hash Algorithm) indica una famiglia di 5 diverse funzioni sviluppate da NSA come standard Federale del governo USA:
- sha1 (160 bit)
- sha2 (**sha224** (224 bit) – **sha256** (256bit) – **sha384** (384 bit) **- sha512** (512bit))

Il comando **dgst** serve ad applicare una tra le funzioni di Hash supportate (man dgst):

Esempi:

```
openssl dgst -md5 -hex myfile  (-md5 default, vedi anche md5sum)
openssl sha1 -hex myfile
```

HMAC

Se 2 parti A e B condividono una chiave simmetrica possono autenticare i messaggi che si scambiano utilizzando lo schema HMAC (Hashed Message Authentication Code) che è una funzione di Hash applicata al Messaggio e alla Chiave condivisa:

Firma= HMAC (K,M)

- Il mittente che deve inviare M calcola HMAC(K,M) che invia assieme ad M
- Il destinatario riceve M e HMAC(K,M), quindi verifica la firma ricalcolando l'HMAC

Questo schema è utilizzato in diversi protocolli crittografici, tra cui IPsec.

```
echo "foo" | openssl dgst  -sha256 -hmac 123
#-hmac key : create a hashed MAC using “key”.
```

Password

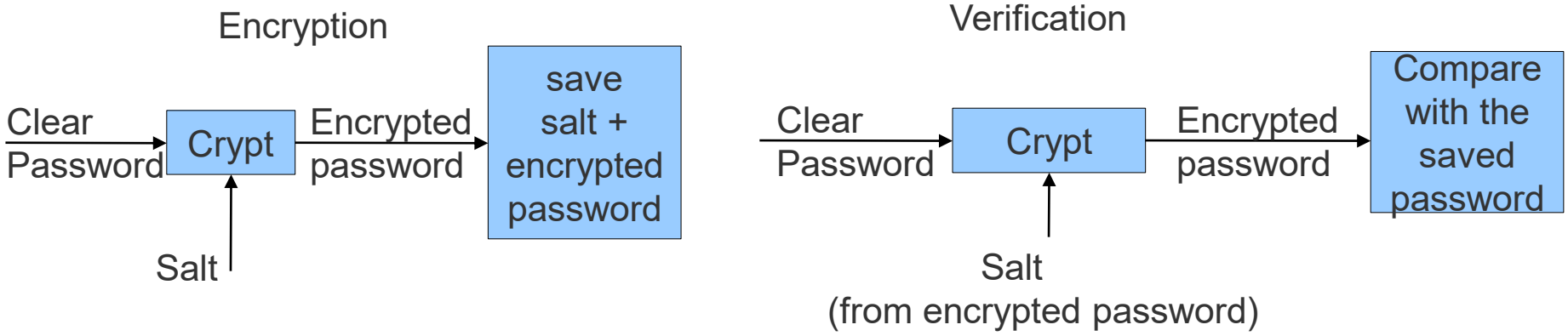
La password è una sequenza alfanumerica che l'utente deve inserire per accedere ad una risorsa protetta. E' quindi un segreto condiviso tra utente e risorsa, che normalmente la risorsa memorizza in formato cifrato.

La cifratura non e' invertibile, quindi per la verifica occorre cifrare la password da verificare e confrontarla con la password cifrata memorizzata dalla risorsa.

La cifratura avviene utilizzando cipher simmetrici, o funzioni di Hash.

Spesso si utilizza usa stinga casuale, detta «salt», per complicare gli attacchi al dizionario.

Il comando **passwd** per generare le password Unix/Linux si basa sulla funzione **crypt**().



Password con crypt()

crypt() in glibc:

Crypt itera 25 volte l'algoritmo DES per cifrare un messaggio costante (tipicamente una sequenza di 0) utilizzando una chiave simmetrica derivata dalla password in chiaro inserita dall'utente + una stringa casuale noti di 12 bit “salt”. Il salt viene scritto in chiaro all'inizio della password cifrata con codifica base64 (2 caratteri).

La modifica di DES (25 iterazioni + salt) rende la cifratura non invertibile.

La verifica della password consiste nel confronto tra i messaggi codificati:

crypt viene ripetuto con la password da verificare e il salt prelevato dalla password cifrata.

Il fatto che la cifratura non è invertibile consente di utilizzare gli algoritmi di Hashing come tecniche alternative di codifica, implementate in glibc2.

crypt in glibc2: aggiunge la possibilità di cifrare con MD5.

L'hash MD5 è di 128 bit

L'output è una stringa composta al più da 34 byte di cui:

-la prima parte è il salt: 1<max 8 char>$

-la seconda parte è una sequenza di 22 caratteri (128 bit di MD5 / 6 di BASE64) contenenti MD5(<salt><clear_password>)

Comandi OpenSSL: passwd

Con openssl è possibile generare le password utilizzando il crypt di glibc (default) oppure un Message Digest:
MD5 con l'opzione -1, SHA-256 opzione -5, SHA-512 opzione -6

Esempi:

```
$ openssl passwd mysecret
QvpTKPjqpBD9.
(i primi 2 caratteri Qv sono il salt generato random e utilizzato
   assieme a mysecret per derivare la chiave di cifratura)

$ openssl passwd -salt Qv mysecret (riproduce la stessa cifratura)

$ openssl passwd -1 mysecret
$1$NvOCDeMO$5keqaA/5i/O7kpEXArm0L/
( NvOCDeMO è il salt casuale, le restanti 22 cifre BASE64 sono l'hash
   MD5 di 128 bit)
```

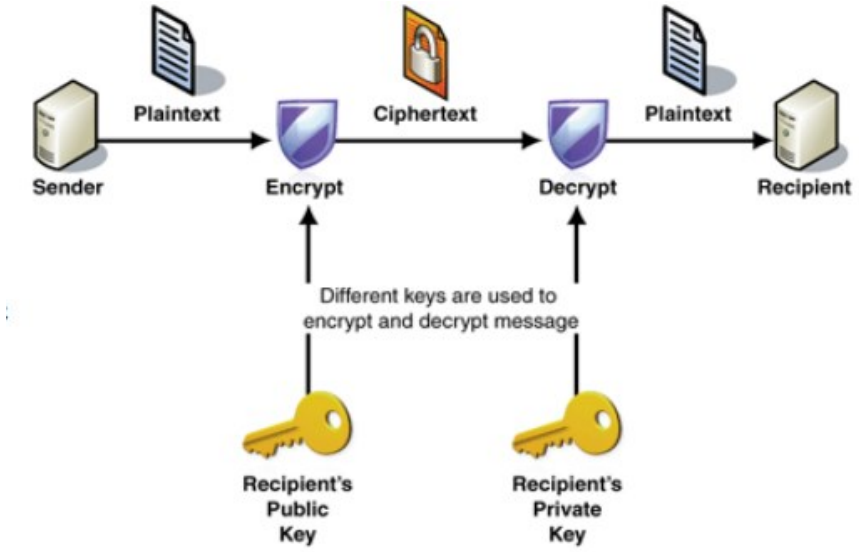
Le password di Apache:
https://httpd.apache.org/docs/2.4/misc/password_encryptions.html

Algoritmi a chiave pubblica

La cifratura a chiave simmetrica ha una grave debolezza nella condivisione della chiave: il trasferimento della chiave la espone ad intercettazione.

Diffie ed Hellmann (Stanford) nel 1976 proposero una tecnica nuova di crittografia, basata sull'aritmetica modulare, in cui vengono utilizzate due chiavi K_e e K_d distinte per la codifica $C=K_e(P)$ la decodifica $P= K_d(C)$.

Assegnando ad una chiave il ruolo di “chiave privata” e all'altra il ruolo di “chiave pubblica” si supera la debolezza della chiave condivisa.



Algoritmi a chiave pubblica

Una importante proprietà di questo algoritmo è: $K_d(K_e (P)) = K_e(K_d (P)) = P$

Questo consente di poter applicare le chiave in 2 modi diversi ottenendo 2 diverse funzioni:

1) Applicando prima la chiave pubblica si ottiene la **Privacy (crypt/decrypt).**

A deve inviare un messaggio P riservato a B su di un canale insicuro.

B possiede una coppia di chiave asimmetriche B_e (pubblica) e B_d (privata).

A cifra P con la chiave pubblica di B: $C=B_e(P)$. Solo B può decifrarlo $P= B_d(C)$

2) Applicando prima la chiave privata si ottiene **l'Autenticazione e Integrità (sign/verify)**

A deve inviare un messaggio P attraverso un canale insicuro. Tutti lo possono leggere, ma chi lo riceve deve essere sicuro che è stato inviato da A.

A possiede una coppia di chiave asimmetriche A_e (pubblica) e A_d (privata).

A cifra P con la propria chiave privata: $C= A_d (P)$. Chiunque può applicare la chiave pubblica di A: $P=A_e(C)$. La decifratura funziona solo se P è stato cifrato da A.

RSA

Rivest, Shamir e Adleman del MIT implementarono nel 1978 l'algoritmo che ha preso il loro nome (RSA) e che è attualmente il più utilizzato nelle applicazioni crittografiche a chiave pubblica.

1) si scelgono 2 numeri primi casuali p e q, sufficientemente grandi.
2) si calcolano m=pq (chiamato modulo) z=(p-1)(q-1)
3) si sceglie un numero e<(p-1)(q-1) coprimo con (p-1)(q-1) #(senza divisori comuni)
4) si calcola un numero d tale che e*d = 1 mod z (il resto della divisione (e*d) /z deve essere 1)

$C = P^e \bmod m$ (e,m) è la chiave pubblica

$P = C^d \bmod m$ (d,m) è la chiave privata

Le 2 chiavi hanno una parte comune **m** (M**odulo**) che è tipicamente di 1024 o 2048 bit e una parte specifica **e, d (Esponenti)** di circa 20 bit.

Per violare la chiave privata occorre determinare **d**. Questo può essere fatto solo per "forza bruta" fattorizzando **m** che è il prodotto di 2 numeri primi.

L' operazione richiederebbe un tempo enormemente grande anche sul più veloce dei computer.

La cifratura $C = P^e \bmod m$ limita la dimensione massima di P (deve essere $P < m$)

Comandi OpenSSL: RSA

OpenSSL 1.x supporta RSA con i seguenti comandi: genrsa, rsa e rsautl.
Nota: In openSSL 3.x i comandi sono genpkey, pkey e pkeyutl

genrsa (man genrsa) consente di creare una coppia di chiavi RSA:

```
openssl genrsa -out rsa_key.pem 2048
# aggiungere -des3 per cifrare la chiave privata con un pass-phrase
```

rsa (man rsa) consente di processare le chiavi rsa. Esempi:

`openssl rsa -in rsa_key.pem -text` (mostra il contenuto)

`openssl rsa -in rsa_key.pem -pubout -out rsa_pub.pem` (estrae la chiave pubblica)

rsautl viene utilizzato per cifrare/decifrare firmare/verificare con chiavi RSA.

Esempio encrypt/dectypt:

```
openssl rsautl -encrypt -pubin -inkey rsa_pub.pem -in text.txt     -out
   encrypted.txt
openssl rsautl -decrypt        -inkey rsa_key.pem -in encrypted.txt -out text.txt
```

Esempio sign/verify:

```
openssl rsautl -sign           -inkey rsa_key.pem -in text.txt    -out signed.txt
openssl rsautl -verify -pubin  -inkey rsa_pub.pem -in signed.txt  -out text.txt
```

Comandi OpenSSL: DIGEST firmato con RSA

Le chiavi RSA possono essere utilizzate per firmare il Digest di un messaggio:

Il seguente comando crea il digest di file.txt utilizzando SHA1, quindi lo firma con la chiave privata:

```
openssl dgst -sha1 -sign rsa_key.pem -out file.sha1_sign   file.txt
```

Per la verifica occorre il messaggio originale, il digest firmato e la chiave pubblica di chi ha firmato:

```
openssl dgst -sha1 -verify rsa_pub.pem -signature file.sha1_sign file.txt
> Verified OK
```

Algoritmi a chiave pubblica: Certificazione dell'identità

Se io genero una coppia di chiavi, uso la chiave privata per cifrare un messaggio e pubblico il messaggio cifrato, chiunque può verificare con la mia chiave pubblica che il messaggio l'ho cifrato io, ma questo non garantisce nulla riguardo la mia identità.

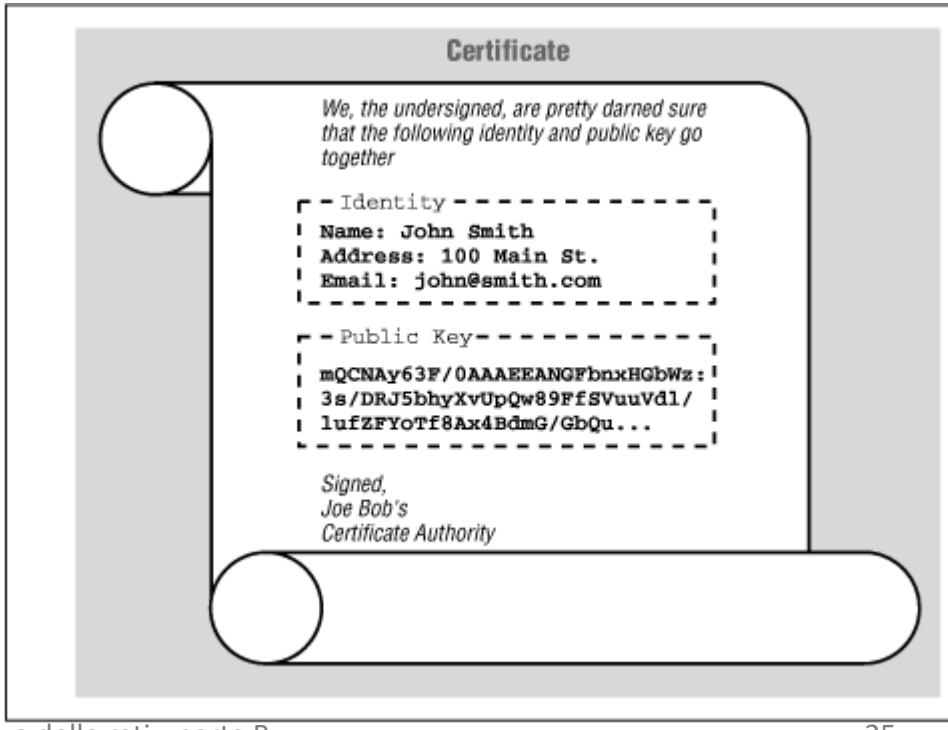
Per associare in modo certo una chiave pubblica a una persona (o host o software) si utilizza il Certificato, ovvero l'insieme della chiave pubblica e dei dati del proprietario firmati (cifrati con la chiave privata) da una **Autorità di Certificazione** che garantisce l'autenticità dei dati contenuti nel **Certificato.**

Applicazioni principali**:**

Posta elettronica (identità del mittente)

Connessioni Web (identità del server e del client)

Software (identità dello sviluppatore)



Certificati X.509

X.509 è lo standard Internazionale emanato da ITU per il formato dei Certificati.

La versione v.3 di X.509 è utilizzata da SSL/TLS.

Questo standard stabilisce quali informazioni possono comporre un certificato; i principali campi sono:

Version: numero della versione di X.509 (v1, v2 o v3)

Serial Number: numero univoco di emissione da parte della CA

Signature Algorithm: Algoritmo usato per firmare il certificato

Issuer: Distinguished Name DN della CA che ha emesso il certificato

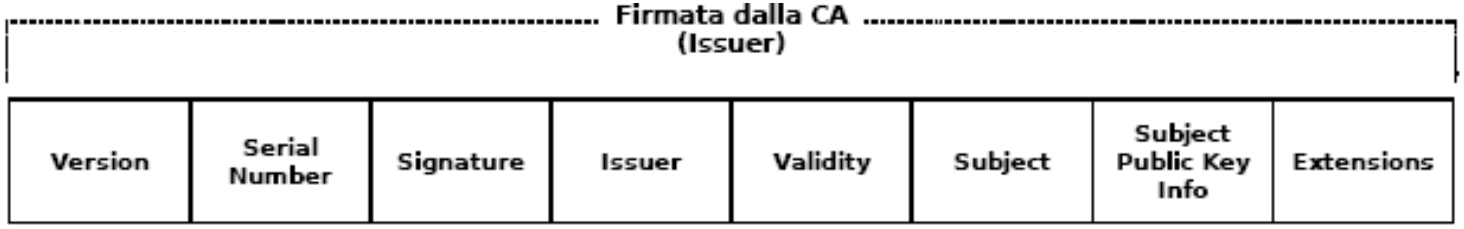
Validity: Inizio e Fine del periodo di Validità.

Subject: Distinguished Name DN del proprietario del Certificato.

Subject Public Key Info : Chiave pubblica (Modulo + Esponente) e algoritmo utilizzato

X509v3 Extensions: Estensioni opzionali (solo v3).

Signature: Firma da parte della CA (MD del Certificato, cifrato con la chiave privata della CA).



La Certification Authority

La CA è un ente che firma le richieste di certificato da parte di una comunità di utenti/host/software garantendone l'identità.

La CA possiede una propria coppia di chiavi e autofirma la propria richiesta (self-signed).

Per identificare univocamente i certificati esiste un name-space gerarchico di certificati, in cui ogni nodo ha un attributo e un valore.

I principali attributi sono: O (Organization), OU (Org. Unit), C (Country), CN (CommonName) , ecc

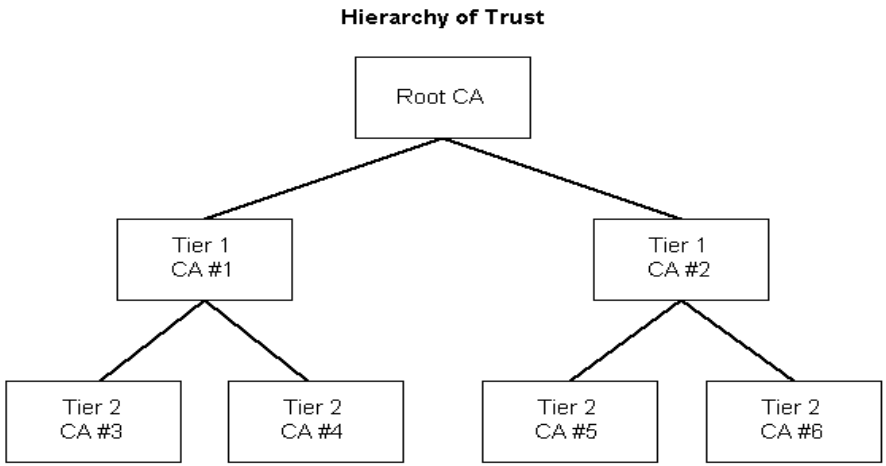
Ogni certificato possiede quindi un FQDN (Subject). Esempio:

C=IT, O=UniprScienze, OU=Staff, CN=roberto alfieri/emailAddress=roberto.alfieri@unipr.it

La CA possiede un BaseDN ed è vincolata ad emettere Certificati all'interno del suo BaseDN.

Una CA può firmare certificati anche per altre CA, poiché il namespace gerarchico consente di organizzare diverse CA all'interno dello stesso albero.

Ogni CA deve gestire la lista dei Certificati Revocati (CRL) che va aggiornata regolarmente.



Istanze di Certification Authority

Alcune CA rilasciano certificati a pagamento e sono già inserite e riconosciute dai più diffusi client Web e SMTP, come ad esempio Comodo, Entrust, GeoTrust, GlobalSign, Cybertrust, Verisign e DigiCert (vedi le impostazioni del Web Browser).

Generalmente una Organizzazione/Ente/Impresa può creare una CA per le certificazioni riconosciute al suo interno

Il GARR fornisce un Certification Service per tutti gli enti afferenti: https://www.servizi.garr.it/cs

GARR partecipa al Trusted Certificate Service (TCS) promosso da Géant a favore delle reti della ricerca europee.

Tramite questo servizio offre gratuitamente alla propria comunità certificati digitali x.509 emessi da una delle maggiori Certification Authority commerciali: Sectigo Limited, riconosciuta automaticamente dalla quasi totalità dei browser web esistenti.

Comandi OpenSSL: ca

OpenSSL consente di creare una propria Certification Authority:

Questo comando genera chiave e certificato autofirmato:

```
openssl req -config openssl.cnf -newkey rsa:512 -days 1000 –nodes   \
                -keyout cakey.pem  -out cacert.pem -x509 -new
```

Con il comando **ca** vengono gestite le operazioni della Certification Authority.
Esempi:

- Firma di una richiesta di Certificato:

```
  openssl ca -in req.pem -out newcert.pem
                        -config /var/www/html/myCA/openssl.cnf
```

- Generazione della Certificate Revocation List:

```
     openssl ca   -gencrl   -out crl.pem
```

Esempio di Certificato della CA (MyCA)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=IT, O=UniprScienze, OU=UniprScienzeCA, CN=UniprScienze/emailAddress=roberto.alfieri@fis.unipr.it
        Validity
            Not Before: Jun  4 20:21:12 2010 GMT
            Not After : Jun  3 20:21:12 2015 GMT
        Subject: C=IT, O=UniprScienze, OU=UniprScienzeCA, CN=UniprScienze/emailAddress=roberto.alfieri@fis.unipr.it
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:cc:89:ba:73:31:b2:b3:e8:74:d9:30:b8:93:02:
                    51:b1:12:8f:f5:e7:f9:2f:96:68:15:e6:4a:19:d8:
                    66:6a:e9:74:66:3e:9f:6f:02:25:ef:3e:5f:09:c3:
                    63:70:e7:40:63:53:a8:75:3c:b7:a8:cb:68:de:4e:
                    dd:c2:89:c9:6d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7A:4A:AE:FA:83:9D:C8:DC:E7:74:0A:B9:17:3A:CB:38:3A:31:CA:A7
            X509v3 Authority Key Identifier:
                keyid:7A:4A:AE:FA:83:9D:C8:DC:E7:74:0A:B9:17:3A:CB:38:3A:31:CA:A7
```

Comandi OpenSSL: req

Questo comando serve per generare una richiesta di certificato.

La chiave privata può essere fornita (-key) o può essere generata dal comando.

Con l'opzione -nodes non viene cifrata la chiave privata. Questo serve per i certificati host per i quali non è possibile digitare una Passphrase.

Il file di configurazione (openssl.cnf) contiene le informazioni relative al proprietario del certificato, alcune delle quali vengono richieste interattivamente.

Esempi:

Generazione della coppia di chiavi (chiave privata e richiesta di certificato):

```
openssl req -new -nodes -out hostreq.pem -keyout hostkey.pem -config
  openssl.cnf
```

Verifica il contenuto della richiesta:

```
openssl req -in req.pem -text
```

Certificati self-signed

Con l'opzione -x509 non viene generata una richiesta, ma un certificato self-signed

Se il file di configurazione (-config) non viene fornito i dati del certificato vengono richiesti interattivamente.

Generazione del certificato self-signed per un server:

```
openssl req -new -nodes -out hostcert.pem -keyout hostkey.pem -x509
   Generating a 1024 bit RSA private key  ......
   writing new private key to 'keyfile.pem'
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [GB]: IT
State or Province Name (full name) [Berkshire]: Parma
Locality Name (eg, city) [Newbury]: Parma
Organization Name (eg, company) [My Company Ltd]: UNIPR
Organizational Unit Name (eg, section) []: LPR
Common Name (eg, your name or your server's hostname) []: lpr1.fis.unipr.it
Email Address []: roberto.alfieri@unipr.it
```

Formati dei certificati X.509 : DER e PEM

I certificati X.509 possono essere rappresentati in diversi possibili formati.
I principali sono: DER, PEM e PKCS12.

DER è un formato binario utilizzato in ambiente Windows e Java con estensioni .DER o .CER.

PEM è testuale (base64) ed è utilizzato prevalentemente in ambiente Unix.
Può contenere Certificati, richieste di Certificati, chiavi private.
Gli oggetti contenuti sono delimitati da stringhe del tipo:

----- BEGIN CERTIFICATE --- ---END CERTIFICATE ----
---BEGIN RSA PRIVATE KEY--- ---END RSA PRIVATE KEY ---
-----BEGIN CERTIFICATE REQUEST--- ---END CERTIFICATE REQUEST--

Comandi OpenSSL: x509

Visualizzare i campi e cambiare formato (tra PEM e DER) di un certificato:

Esempi:

```
openssl x509 -in cert.pem -noout -text
openssl x509 -in cert.pem -noout -serial
openssl x509 -in cert.pem -noout -subject
openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
```

https://www.openssl.org/docs/man1.1.1/man1/x509.html

Formati dei certificati X.509 : PKCS

PKCS (Public Key Cryptography Standards) è un gruppo si Standard creati da “RSA Data Security” con lo scopo di creare formati di interoperabilità.

Sono stati creati 15 standard (da PKCS#1 a PKCS#15) ma i formati prevalentemente utilizzati sono:

PKCS#12: Personal Information Exchange Syntax Standard.

Nato come evoluzione di PFX, definisce un formato per immagazzinare chiave privata e certificato in un file protetto con password (cifratura con chiave simmetrica). L'estensione è p12.

PKCS#7: Cryptographic Message Syntax Information.

E' un formato per rappresentare messaggi cifrati o firmati ed è utilizzato da S/MIME per l'invio di e-mail cifrate e/o firmate. Recepito da IETF (RFC 2986).

PKCS#11: Cryptographic Token Interface

Definisce le API per utilizzare i Token Crittografici, come le SmartCard e le chiavi USB

Comandi OpenSSL: pkcs12

Questo comando serve per gestire i file in formato PKCS12.
https://www.openssl.org/docs/manmaster/man1/openssl-pkcs12.html

Esempi:
Genera il file PKCS12 dai file PEM:

```
openssl pkcs12 -export -out cert.p12 -inkey userkey.pem -in usercert.pem
```

Dal formato PKCS12 a PEM (separatamente chiave e certificato):

```
#estrai la chiave da cert.p12 (aggiungere -nodes per i server)
openssl pkcs12 -nocerts -in cert.p12 -out userkey.pem
#estrai il certificato da cert.p12
openssl pkcs12 -clcerts -nokeys -in cert.p12 -out usercert.pem
#visualizza il certificato in chiaro
openssl x509 -in usercert.pem -text
```

Dal formato PEM a PKCS12 (con cypher AES256-CBC):

```
openssl pkcs12 -export -inkey userkey.pem -in usercert.pem -out test.p12
   -certpbe AES-256-CBC -keypbe AES-256-CBC
```

Comandi OpenSSL: s/mime

MIME (Multipurpose Internet Mail Extensions)

è una estensione del protocollo di posta Elettronica per poter includere in un unico messaggio più documenti (Allegati), codificati in ASCII Standard (vedi la Posta Elettronica).

MIME introduce un header in cui è possibile inserire campi che descrivono il contenuto, tra cui:

Content-Type : Tipo di dato contenuto (esempio image/JPEG)

Content-transfer-encoding: Codifica del contenuto (esempio BASE64)

S/MIME (Secure/MIME)

include in questo schema la possibilità di

cifrare/decifrare il contenuto, utilizzando un algoritmo a chiave simmetrica.

La chiave simmetrica viene cifrata con la chiave pubblica del destinatario e inviata insieme al messaggio stesso.

(Content-Type del messaggio cifrato: application/x-pkcs7-mime)

firmare/verificare un messaggio allegando la Firma

(Content-Type della firma: application/x-pkcs7-signature)

Consultare le opzioni: man smime

https://www.openssl.org/docs/manmaster/man1/openssl-smime.html

Cifrare/Decifrare con s/mime

Per inviare un messaggio cifrato al destinatario la chiave simmetrica viene cifrata con la chiave pubblica del destinatario e inviata assieme al messaggio stesso. Il destinatario legge la chiave simmetrica utilizzando la propria chiave privata, quindi decripta il messaggio con la chiave simmetrica.

Il Mittente codifica il messaggio con seguente comando

```
openssl smime -encrypt -text -in message.txt  \
          -out encrypted-message.txt  destination-user-certificate.pem
```

viene generato un file Mime con una intestazione del tipo:

MIME-Version: 1.0

Content-Type: application/x-pkcs7-mime; name="smime.p7m"

Content-Transfer-Encoding: base64

Il Destinatario lo decodifica con il seguente comando:

```
openssl  smime  -decrypt -text  -in encrypted-message.txt  \
                    -out decrypted-message.txt  -inkey userkey.pem
```

Firmare/Verificare con s/mime

La firma consiste nel cifrare con la chiave privata del mittente il Digest del Messaggio, che verrà inviato insieme al messaggio stesso.

- **Garanzia dell'identità del Signer**: Consiste nella decifratura della Firma con la chiave pubblica del Signer.
- **Garanzia di Integrità del Messaggio**: Il messaggio è integro se il Digest decifrato nella firma e il Digest ricalcolato sono uguali.

Per firmare un messaggio:

```
openssl smime -sign -text -in message.txt  -out signed-message.txt \
                          -signer usercert.pem  -inkey userkey.pem
```

Il certificato del Signer è incluso nel messaggio.

Con questo comando il Destinatario estrae il Certificato del Signer:

```
openssl smime -pk7out -in signed-message.txt | openssl pkcs7 -print_certs
```

Con questo comando il destinatario verifica il certificato del Signer tra le CA di cui si fida (-CAfile o -CApath) , quindi usa il Certificato per decifrare la firma ed estrarre il MD, infine confronta il MD ricevuto con quello calcolato.

```
openssl  smime  -verify  -text -in signed-message.txt  -CAfile CAcert.pem
```

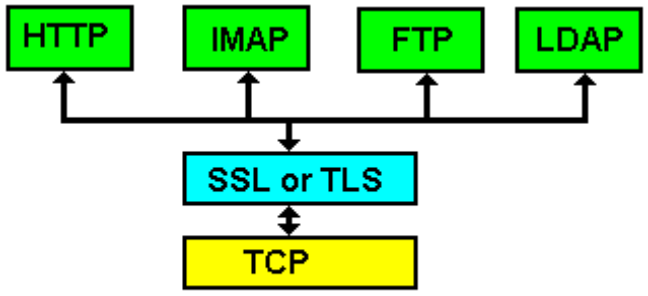
Crittografia nella comunicazione

Gli algoritmi crittografici simmetrici e a chiave pubblica vengono utilizzati per costruire protocolli di rete con l'obiettivo di cifrare la comunicazione, fornendo diversi servizi di sicurezza quali la **Confidenzialità, l'Autenticazione e il Non Ripudio.**

Il principale protocollo di questo tipo è **SSL** (Secure Socket Layer), poi diventato **TLS** (Transport Security Layer).

SSL /TLS è strutturato come un layer che si pone tra il trasporto (TCP) e l'applicazione.

Molte applicazioni di rete (ad esempio HTTP, IMAP, LDAP) sono state adattate per appoggiarsi su SSL (anziché TCP) per usufruire dei servizi di sicurezza del protocollo.



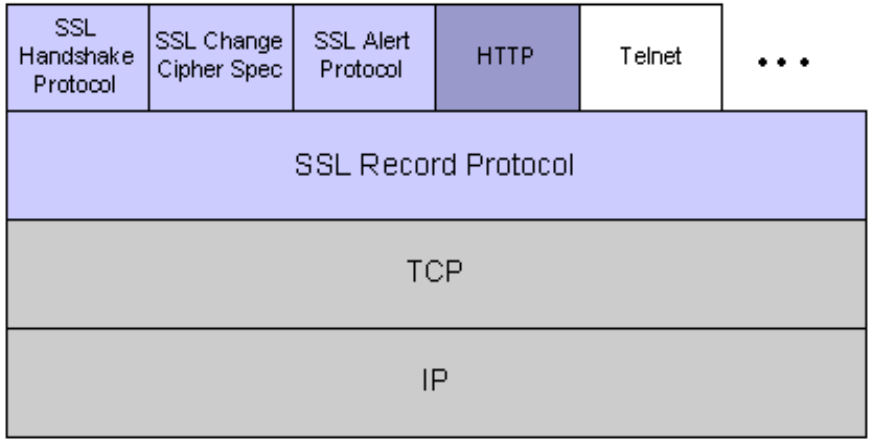
SSL/TLS

Il protocollo SSL mette in sicurezza una connessione Client-Server introducendo un **Layer di Sicurezza al livello di Trasporto (TLS)**.

In questo modo l'applicativo non vede più le API di TCP e deve quindi riscrivere l'interfaccia con il livello di Trasporto utilizzando le API SSL/TLS.

Vengono utilizzati 2 protocolli principali:

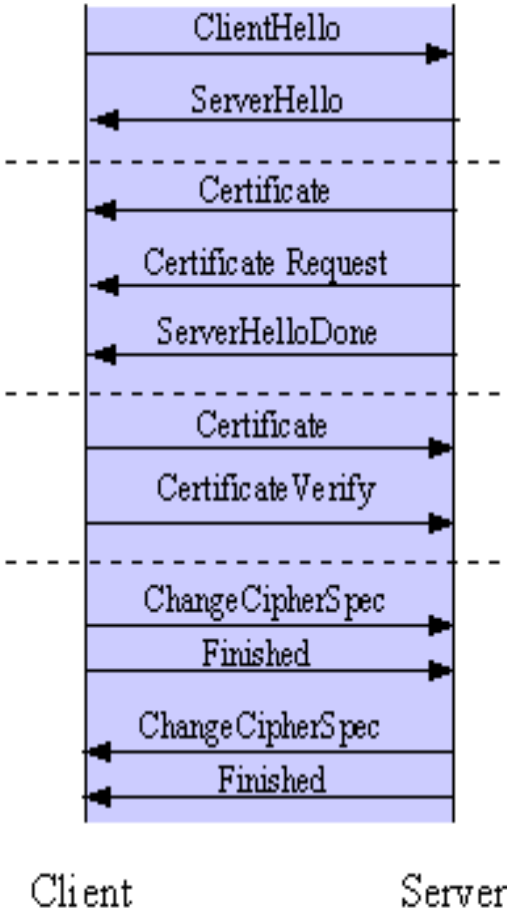
- Il Record Layer Protocol è utilizzato per trasmettere e ricevere dati (cifratura e decifratura) mediante un Cipher Simmetrico.
- L'Handshake Protocol interviene solo all'inizio (o per ripristinare una sessione interrotta). Negozia i parametri per lo scambio dei dati (scambio dei Certificati e condivisione di una chiave di Sessione per il Cipher Simmetrico).



SSL Handshake Protocol

1. Il client manda SSL version e il Cipher Suite da utilizzare
2. Il server manda il proprio SSL versione e il cipher suite
3. Il server manda il proprio certificato
4. Il server opzionalmente richiede il certificato del client
5. Il client verifica il certificato del server e procede solo se è ok
6. Il client genera il pre-master secret
7. Il client cripta la chiave con il cert. del server e la invia
8. Se il sever ha richiesto l'autenticazione del client,
   Il client cifra un challenge e lo invia al server
9. Se il server non riesce a decifrare termina la sessione.
10. Client e server usano il pre-master secret per
    Generare la chiave di sessione condivisa.
11. Il client manda un messaggio di conclusione dell'handshake
12. Il server manda un messaggio di conclusione dell'handshake

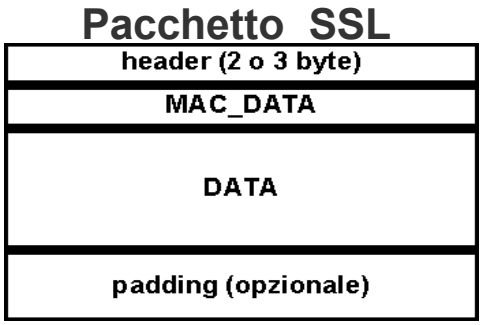
In qualsiasi momento entrambi possono **Renegoziare** la connessione, nel qual caso la procedura è ripetuta.

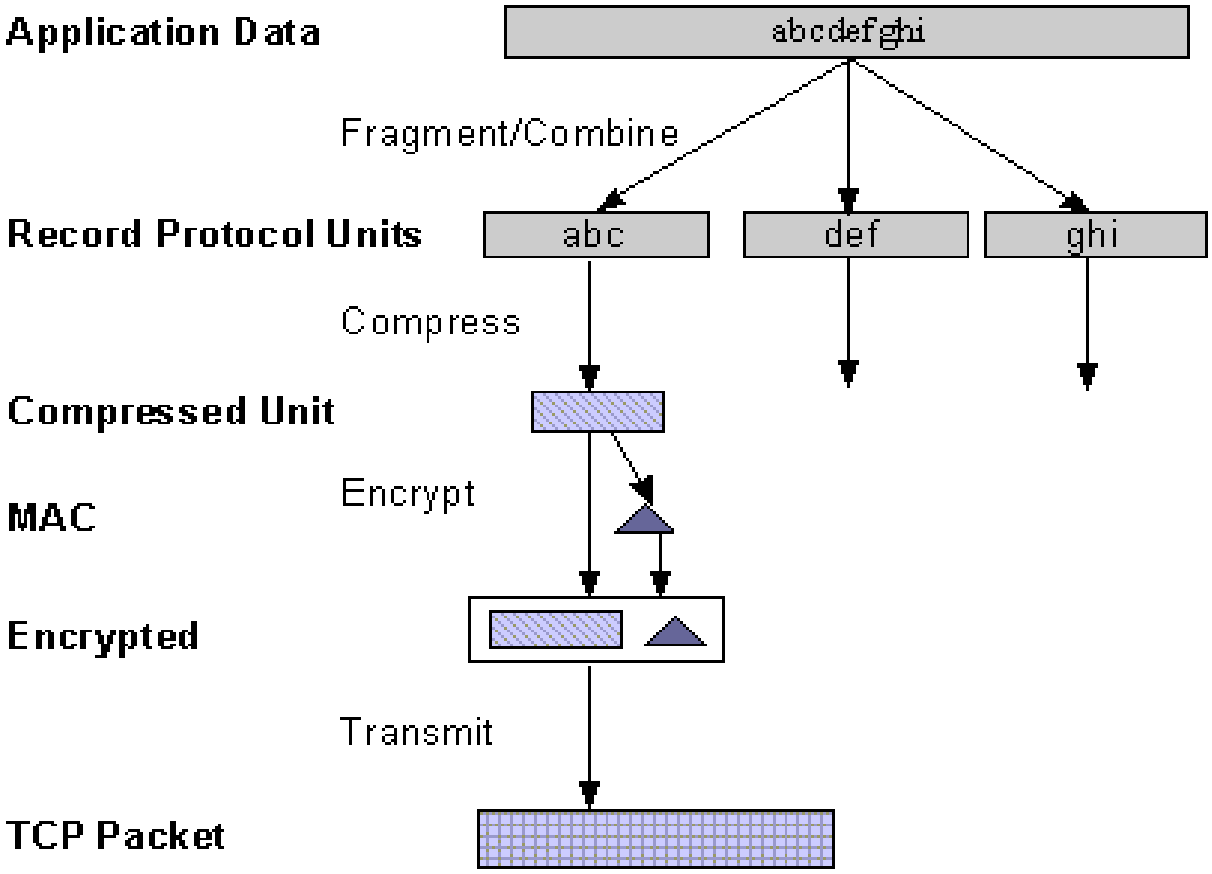


SSL Record Protocol

Al termine dell'Handshake Protocol inizia l'SSL Record Protocol:

1. Frammentazione del flusso
2. Compressione dei frammenti
3. Calcolo del Digest
4. Cifratura





SSL Cipher suites

Una sessione SSL/TLS richiede l'utilizzo di un set di strumenti crittografici per

- Scambio chiavi (Kx)
- Autenticazione (Au)
- Cifratura del canale (Enc)
- Digest (Mac)

Un cipher suite è un insieme di strumenti che implementano le funzioni necessarie.
https://www.ssl.com/it/guida/tls-conformit%C3%A0-agli-standard/

Il seguente comando elenca le cipher suite supportate in openssl:

```
openssl ciphers -v
```

Applicazioni SSL e StartTLS

Alcune versioni di applicazioni comuni incorporano SSL/TLS attivando connessioni cifrate in ascolto su porte diverse e che possono quindi coesistere con le versioni non cifrate. Le principali applicazioni SSL sono:

- **https** (HTTP over SSL) 443/tcp
- **pop3s** (POP3 over SSL) 995/tcp
- **imaps** (IMAP over SSL) 993/tcp
- **smtps** (SMTP over SSL) 465/tcp
- **ldaps** (LDAP over SSL) 636/tcp

STARTTLS è una estensione della comunicazione in chiaro, che offre l'opzione di passare ad una connessione cifrata (SSL o TLS) anziché passare ad una connessione cifrata separata.

IMAP, SMTP , POP e LDAP hanno implementazioni con supporto StartTLS.

Comandi OpenSSL: s_client s_server

s_client (info: man s_client) è un client in grado di attivare una connessione con un server ssl/tls. Visualizza i dati del protocollo ssl handshake (protocollo tls utilizzato, cipher, la chiave di sessione, il certificato del server, subject, ecc)

Esempio: **`openssl s_client -connect www.unipr.it:443`**

Il carattere R forza la rinegoziazione, Q forza la disconnessione
Una volta connessi è possibile digitare manualmente i comandi da inviare al server (ad esempio, se il server è www : "GET / HTTP/1.0")

s_server (info: man s_server) svolge le stesse funzioni lato server. Esempio:

`openssl s_server -accept 443 -www -cert hostcert.pem -key hostkey.pem`

Con l'opzione www viene emulato un web server che risponde al browser con i dati salienti della connessione SSL.

Programmazione openSSL

La libreria **SSL** fornisce le API necessarie per la programmazione in C di canali cifrati.

https://developer.ibm.com/tutorials/l-openssl/

La cifratura avviene grazie all'astrazione di I/O, denominata BIO (Basic I/O abstraction), che consente di creare un canale cifrato o in chiaro

Esempio in C:

TLS_client.c https://wiki.openssl.org/index.php/SSL/TLS_Client

Simple_TLS_Server.c https://wiki.openssl.org/index.php/Simple_TLS_Server

Python consente di creare connessioni SSL/TLS, sia client che server, attraverso diverse librerie che implementano un wrapper di TCP basato su openSSL. Le principali sono:

- **TLS/SSL wrapper for socket objects** https://docs.python.org/2/library/ssl.html
- **pyOpenSSL** https://pyopenssl.org/en/stable/

Il modulo httplib (python 2) supporta sia HTTP, con httplib.HTTPConnection(), che HTTPS, grazie al metodo httplib.HTTPSConnection()

In Python 3 il modulo httplib è stato rinominato http.client

UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE ED INFORMATICHE
Corso di Laurea in Informatica

Sicurezza delle reti – Parte C

Protocolli

RETI DI CALCOLATORI - a.a. 2023/2024

Roberto Alfieri

La sicurezza delle reti: sommario

PARTE A

- I servizi di sicurezza
- Metodi e strumenti di attacco
- Strumenti di Difesa

PARTE B

- Crittografia applicata e OpenSSL

PARTE C

- Protocolli di Autenticazione
- IPsec e VPN

Protocolli di Autenticazione

Riferimenti: *http://it.wikiversity.org/wiki/Protocolli_di_autenticazione*

Autenticazione (Authentication): un servizio di sicurezza che consente di accertare l'identità dichiarata da una entità mediate la verifica di credenziali.

L'autenticazione può avvenire:

- tra una persona fisica e un host o dispositivo (es. bancomat)
- in una comunicazione di rete (l'origine dei dati o i peer di una comunicazione) mediante un opportuno protocollo

L'autenticazione può essere mutua oppure no, dipende dalle situazioni.

Ad esempio è mutua quando consulto la posta elettronica:

- Il server si deve autenticare con me per dimostrarmi di essere il server che gestisce la mia posta.
- Io devo dimostrare al server che sono il titolare della mailbox.

Le tecniche possono basarsi su

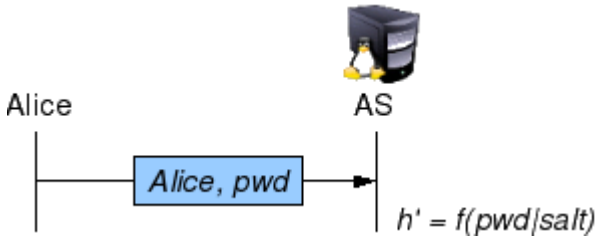
- La conoscenza di un segreto (password, PIN, ..)
- tecniche crittografiche
- Caratteristiche biometriche (se l'autenticazione avviene tra una persona ed un host locale): timbro voce, impronta digitale, fondo dell'occhio, ecc

Autenticazione tramite password: PAP

PAP (Password Authentication Protocol)

- presume che il canale sia sicuro (non intercettabile)
- Il client manda al server il proprio Nome e la Password
- Il server cerca in una tabella il nome utente e verifica la correttezza della password applicando una funzione di trasformazione

  (che consente di evitare che il server memorizzi la password in chiaro)
- Ancora in uso in PPP anche se deprecato.



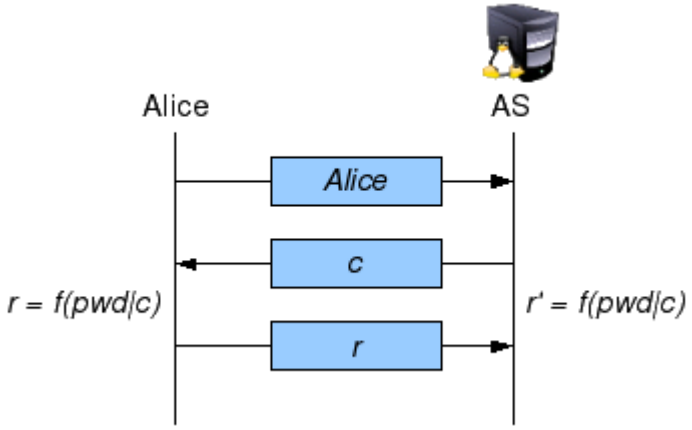
Autenticazione tramite password: CHAP

CHAP (Challenge Handshake Authentication Protocol)

- Usato in PPP
- Windows usa una variante di CHAP detta MS-CHAP
- Il server invia un numero casuale c (challenge) utilizzato dal client come salt
- La funzione di trasformazione r=f(pwd,c) è calcolata sia dal server che dal client
- L'implementazione standard di CHAP usa MD5: r=MD5(pwd, c)

Vantaggi: La password non viene scambiata tra client e server

Problemi: Il DB delle password deve essere salvato in chiaro. Attacco al dizionario



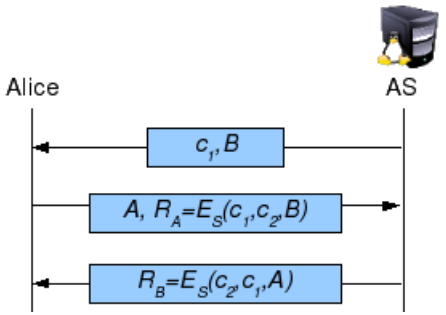
Autenticazione con challenge e chiave simmetrica

Le 2 parti A e B , che condividono una chiave simmetrica S, si inventano ciascuna un numero casuale, detto Challenge (c1 e c2)

Il server invia la propria identità (B) e il proprio Challenge (c1)

Il client risponde inviando la propria identità (A) e la cifratura di c1, c2 e B.

Il server chiude il protocollo inviando la cifratura di c1, c2 e A.



Vantaggi: I messaggi cifrati non sono esposti ad attacco al dizionario.

Problemi:

- Se abbiamo N nodi ogni nodo deve conoscere N-1 chiavi.
- La condivisione di una chiave simmetrica si espone ad intercettazione.

Autenticazione con KDC

Il modello del “Centro di Distribuzione delle Chiavi” (KDC, Key Distribution Center) si applica ad una comunità di N entità (persone/host/servizi) che devono autenticarsi reciprocamente.

In questo schema ogni utente ha una singola chiave condivisa con il KDC.

Esempio: A deve comunicare con B

A condivide con KDC la chiave Ka, B condivide con KDC la chiave Kb

A sceglie una chiave di sessione Ks, invia a KDC la chiave e B, in modo cifrato.

KDC decifra Ks e la invia a B

```
A  ->   A, Ka(B,Ks)   ->  KDC
                          KDC    ->    Kb(A, Ks)   -> B
```

Vantaggi: Singola chiave Ka per comunicare con N entità.

Problemi: A deve inserire la chiave Ka per ogni connessione.

Autenticazione Kerberos

Kerberos è un protocollo di Autenticazione (progettato al MIT) che implementa il modello del “Centro di Distribuzione delle Chiavi”.

E' ampiamente diffuso soprattutto negli USA sia su Linux che Windows

Un sistema Kerberos gestisce una comunità di utenti (REALM) in cui ogni utente ha una singola chiave condivisa Ka con il KDC, ma il KDC si compone di 2 server:

AS (Authentication Server) Gestisce il LOGIN

TGS (Ticket Granting Server) Gestisce la sessione

La password di A (Ka) viene usata una sola volta per tutte le autenticazioni della sessione (Single Sign On - SSO) e rimane sul computer del client solo per pochi millisecondi.

La chiave di sessione che A presenta a B serve solo a dimostrare l'identità di A (autenticazione). B deciderà cosa consentire di fare ad A (autorizzazione)

Autenticazione Kerberos

Innanzitutto A chiede all'AS la chiave di sessione Ks (Login sul REALM):

A -> A -> AS

A <- Ka(Ks), Ktgs(A,Ks) <- AS

Ktgs(A,Ks) contiene A e Ks cifrate con la chiave segreta del TGS

Quando A deve comunicare con B chiede al TGS un Ticket Kab da usare con B:

A -> Ktgs(A,Ks), B, Ks(t1) -> TGS

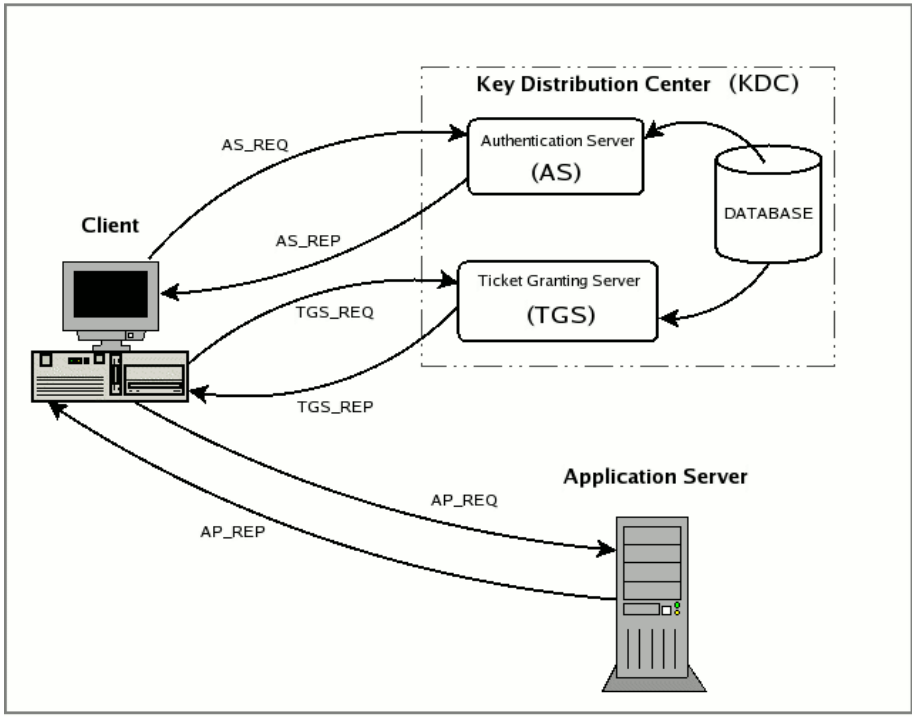
A <- Ks(B, Kab), Kb(A,Kab) <- TGS

Quindi A si rivolge a B comunicandogli la chiave di sessione Kab:

A -> Kb(A,Kab), Kab(t1) -> B

A <- Kab(t2) <- B

I timestamp t1 e t2 impediscono che qualcuno possa intercettare i messaggi e replicarli con un mittente falsificato (spoofed).



Scambio chiavi di Diffie-Hellman

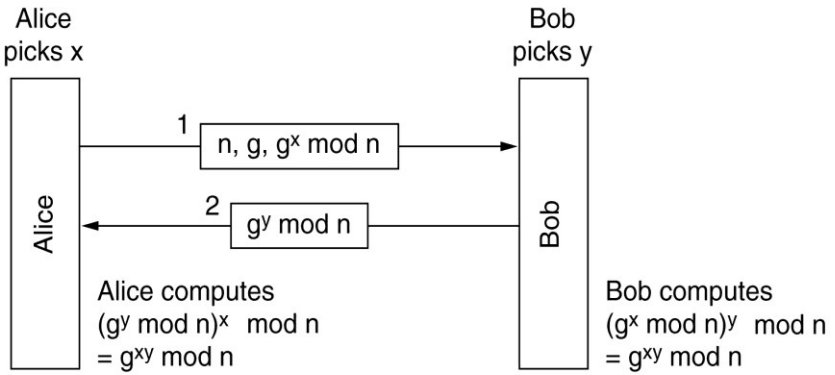
Consente a 2 entità che non hanno avuto contatti in precedenza di stabilire in modo sicuro una chiave simmetrica condivisa.

Servizi di sicurezza: **confidenzialità senza autenticazione.**

A e B devono condividere 2 numeri grandi n e g che possono scambiarsi in chiaro.
(n e (n-1)/2 sono primi, g=f(n) opportunamente calcolato)

1) A sceglie X grande che mantiene segreto
quindi invia A -> n, g, g^x mod n -> B

2) B sceglie Y grande che mantiene segreto
quindi invia A <- g^y mod n <- B



3) B calcola $(g^x \bmod n)^y \bmod n = g^{xy} \bmod n$
A calcola $(g^y \bmod n)^x \bmod n = g^{xy} \bmod n$

g^{xy} mod n
è la chiave condivisa di sessione

Autenticazione con PKI

L'utilizzo di una PKI (Public Key Infrastructure) ha il vantaggio di non richiedere preventivamente chiavi condivise.

I nodi A e B hanno una coppia di chiavi A-> (Ea,Da) B-> (Eb,Db).

Le chiavi Ea ed Eb sono pubbliche.

A invia la propria Identità e un Challenge Ra a B, cifrati con la chiave pubblica di B.

A -> Eb(A,Ra) -> B

B decifra il messaggio, sceglie una chiave di sessione Ks e la invia ad A:

A <- Ea (Ra, Rb, Ks) <- B

A risponde con il Challenge di B cifrato con la chiave di sessione Ks:

A -> Ks(Rb) -> B

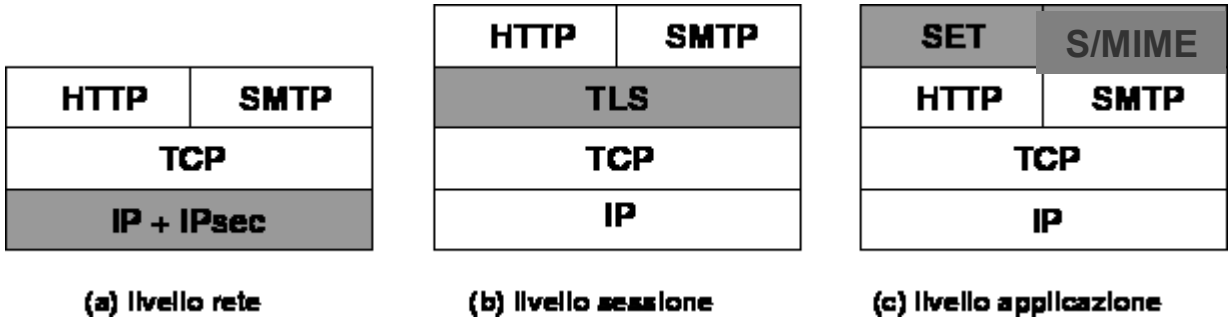
Servizi di sicurezza:

- **la chiave di sessione Ks è condivisa (confidenzialità)**
- **gli host hanno verificato l'identità reciproca (autenticazione).**

Protocolli per la riservatezza

La cifratura di una comunicazione può avvenire a diversi livelli:

- Alcune applicazioni cifrate si appoggiano sull'applicazione in chiaro. Il payload viene cifrato e quindi veicolato da applicativo non cifrato (vedi ad es. S/MIME su SMTP)
- Il protocollo **SSL/TLS** fornisce un Layer intermedio tra TCP e applicazione che consente di cifrare le applicazioni. Questo richiede la riscrittura delle applicazioni che devono interfacciarsi al layer SSL anziché TCP.



- IPsec è un Layer di cifratura che viene posizionato a livello rete, rendendo la cifratura trasparente al livello delle applicazioni, che non devono essere modificate

IPsec è integrato in IPv6 (Extension Header 50 e 51), mentre è opzionale in IP4.

Attualmente l'uso predominante di IPsec è la creazione di Reti Virtuali Private (VPN)

Protocolli IPsec

Una “connessione” IPsec chiamata SA (Security Association), è una connessione Simplex e ha un Identificatore di sicurezza associato.

Per una connessione Duplex è necessario attivare un SA per ciascuna direzione.

Ogni pacchetto Ipsec include nell’intestazione un indice (Security Parameter Index – SPI) che consente al ricevente individuare la SA e quindi di reperire la chiave di decifratura.

IPsec, analogamente a SSL, è formato da

- Un protocollo per lo scambio delle chiavi necessarie per la cifratura del canale:
  - **IKE** (Internet Key Exchange)
- Due protocolli alternativi per la cifratura dei dati sul canale:
  - **AH** (Authentication Header). Gestisce integrità, ma non confidenzialità.
  - **ESP** (Encapsultating Security Payload). Anche confidenzialità (cifratura payload)

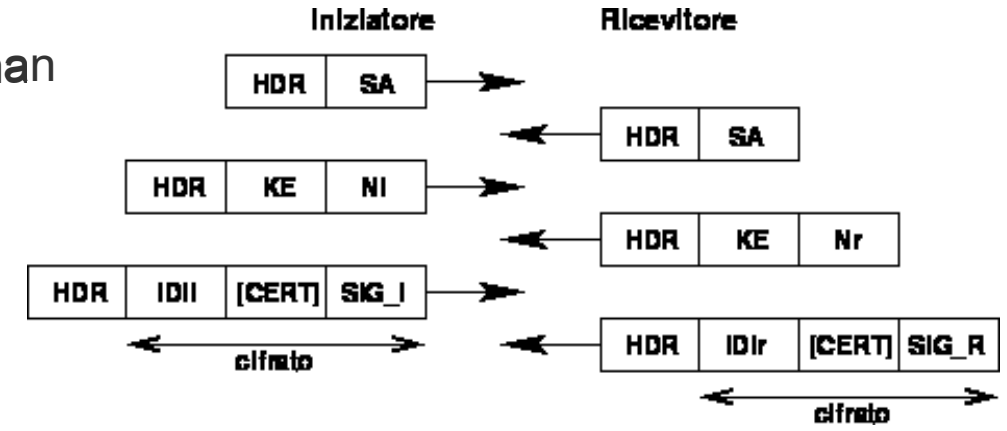
IKE (Internet Key Exchange)

IKE è utilizzato per stabilire una SA.

E' a livello applicazione e usa UDP come trasporto sulla porta 500.

L'obiettivo è stabilire una Shared Session Secret da cui poi derivare la chiave per cifrare la SA.

Viene utilizzato l'algoritmo di Diffie-Hellman



Ogni host IPsec gestisce un Security Association Database che include l'elenco delle SA attive.

Ogni elemento del DB indicizzato dal Security Parameter Index (SPI) include:

- l'indirizzo di destinazione
- servizi di sicurezza (AH, ESP)
- Algoritmi simmetrici usati per cifrare i dati (3DES, AES, ..) e le chiavi associate
- altri parametri quali l'IPsec lifetime

Transport mode e Tunnel mode

Sia AH che ESP possono funzionare il modalità Transport o Tunnel.

Transport mode

- connessione host-to-host
- usato dagli end-point non dai gateway
- viene cifrato solo il payload dei datagrammi IP, non l'header
- computazionalmente leggero
- ogni host deve avere tutto il software necessario ad implementare IPsec
- si aggiunge solo l'header IPsec; mittente e destinazione si vedono

Tunnel mode

- connessione Gateway-to-Gateway
- viene cifrato tutto il pacchetto IP originale
- utilizzato per realizzare le VPN
- computazionalmente oneroso
- solo i Gateway devono avere il software IPsec
- si hanno punti di centralizzazione quindi single point of failure
- si ha un doppio incapsulamento, aggiungendo l'header del gateway e l'header IPsec

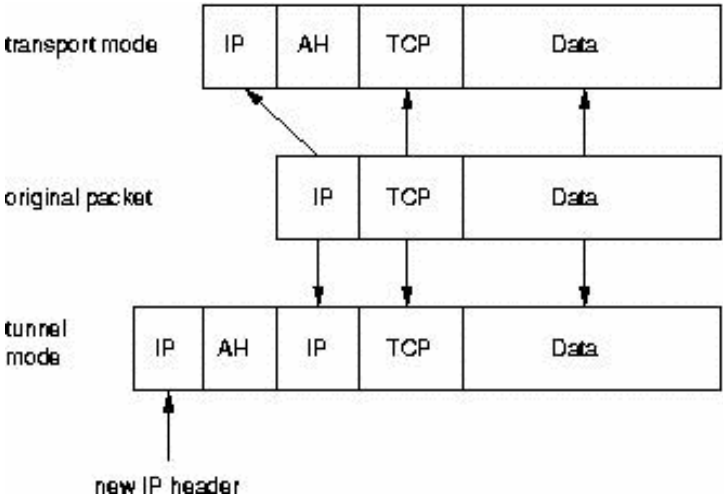
AH (Authentication Header)

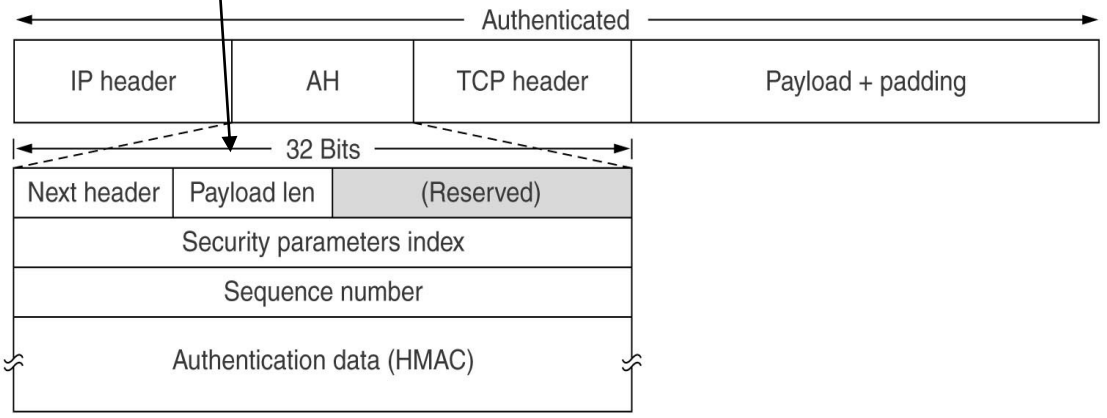
AH gestisce integrità del pacchetto, ma non la confidenzialità: non ha la cifratura.

Il protocollo determina una intestazione di 24 Byte che contiene l'HMAC del Datagramma IP (Header+payload)

L'intestazione che può essere inserita

- nelle estensioni dei protocolli IPv4 e IPv6 (**Transport Mode)**
- nell'estensione di un nuova intestazione IP che come payload incapsula il pacchetto IP originale **(Tunnel Mode)**





ESP (Encapsulating Security Payload)

ESP, rispetto a HA, aggiunge la confidenzialità poiché il payload viene cifrato.

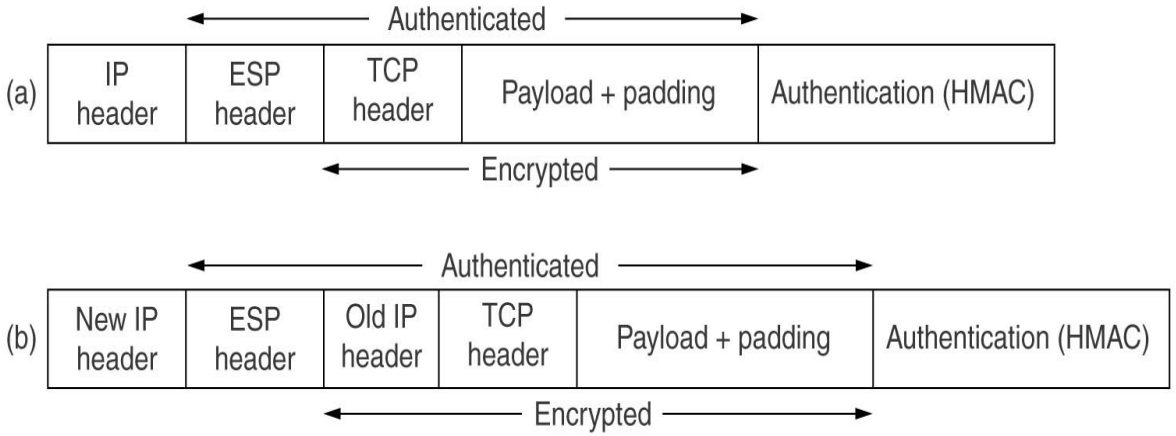
Il campo **HMAC (**diversamente da AH)

- non copre l'Header IP
- è accodato al payload cifrato. Viene calcolato mentre il pacchetto sta uscendo

Cifratura:

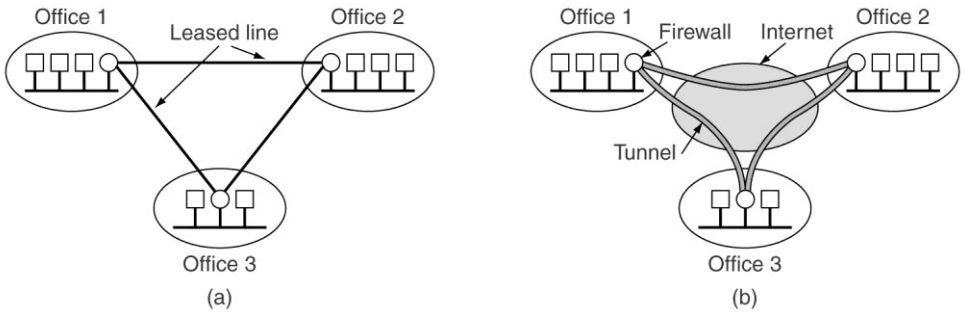
a) **Transport**: viene cifrata la trama di trasporto (TCP Header + Payload)

b) **Tunnel**: viene cifrato il pacchetto IP (old IP header+TCP header+Payload)



VPN (Virtual Private Network)

Una Virtual Private Network o VPN è una rete privata instaurata tra soggetti che utilizzano un mezzo di trasmissione pubblico e condiviso come ATM o, più frequentemente, Internet.



L'utilizzo tipico in Internet è

- tra 2 o più LAN remote
- tra una LAN e un singolo host (e.g. Una persona che si trova all'esterno della propria struttura e vuole connettere il proprio portatile come se fosse all'interno.)

In entrambi i casi viene generato un tunnel protetto tra 2 gateway.

I protocolli più utilizzati per realizzare il tunnel cifrati sono:

- IPsec, SSL/TLS, PPTP (Point-to-Point Tunnelling Protocol di Microsoft)

Use Cases: Forticlient (in uso in UNIPR) https://forticlient.com/techspec