Sicurezza dei Sistemi e delle Reti

Indice

Capitolo 1

Introduzione

1.1 Terminologia

Un **sistema** può essere visto come (anche una combinazione di):

- *hw*
- *sw*
- persone che lavorano con *hw* e *sw*
- clienti

Un **attore** può essere:

- una *persona* che *interagisce* con il sistema
- un *dispositivo* che *interagisce* con il sistema
- un *ruolo* (cliente)
- un *ruolo complesso* (Alice che finge di essere Bob)

Una rete è una configurazione di individui interconnessi. Una **rete di computer** può essere vista sotto due punti di vista:

- ***Fisico:*** una infrastruttura *hw* che connette diversi dispositivi
- ***Logico:*** un sistema che facilita lo scambio di informazioni tra applicazioni che non condividono uno spazio di memoria

1.2 Sicurezza

La sicurezza può essere intesa come il **raggiungimento di un obiettivo in presenza di un attacco**; è difficile da assicurare perché l'obiettivo è *negativo*:

- *dimostrare che Alice può accedere ad un file è facile*
- *dimostrare che nessuno oltre ad Alice può accedervi è molto più difficile*

Di norma si raggiunge con un processo **iterativo**:

- si cerca di trovare l'*anello debole* nel sistema
- si adottano delle *contromisure*
- si continua a fare *analisi* in cerca di nuove vulnerabilità

Il concetto di *sicurezza perfetta* non è raggiungibile; per discutere di sicurezza si deve definire:

- **Politica di sicurezza:** definizione di regole di sicurezza che il sistema deve rispettare
- **Modello di minaccia:** assunzioni su cosa possa fare l'avversario per penetrare nel sistema; devo comprendere la potenza dell'avversario
- **Meccanismi:** *sw* o *hw* che cercano di assicuare che la politica sia rispettata, finché l'attaccante segue il modello di minaccia

Le reti di computer sono sistemi insicuri: abbiamo un sistema complesso (*computer*) in un sistema complesso (*rete*) $\rightarrow$ è difficile prevedere da quale punto arriveranno gli attacchi e quali vettori verranno sfruttati.
Ad oggi, le motivazioni dietro agli attacchi sono principalmente:

- economiche
- politiche / militari
- attivismo

Capitolo 2

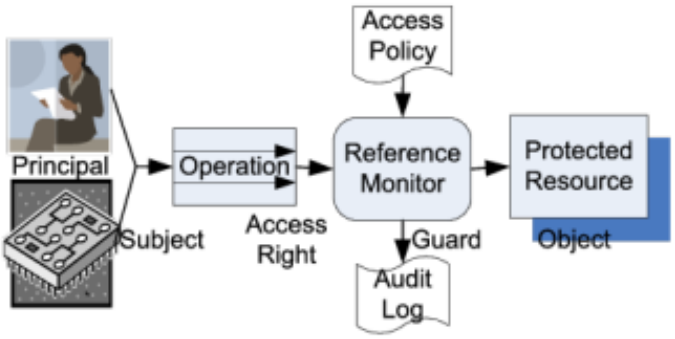
Standard e Concetti base

2.1 Standard

Ci sono diverse organizzazioni che si occupano di standard:

- NIST (*National Institute of Standars and Technology*)
- ISOC (*Internet Society*)
- ITU-T (*International Telecommunicatin Union*)
- ISO (*International Organization for Standardization*)
  - 27001: documento a cui fare riferimento per costruire un sistema di gestione della sicurezza delle informazioni che possa essere certificato da un ente indipendente
  - 27002: non è certificabile, è una raccolta di *best practices* per soddisfarre i requisiti della 27001

2.2 Reference Monitor Model - RMM



Il **reference monitor** è un **sistema dotato di una politica di controllo degli accessi**. Si occupa di:

- **autenticare chi vuole accedere**
- **autorizzare** o meno le operazioni richieste in base ai permessi
- fare **audit** → tenere un log delle azioni compiute

2.3 Definizioni

- La **sicurezza informatica** è l'insieme di strumenti, politiche, linee guida ... che possono essere utilizzate per proteggere l'ambiente e le risorse dell'organizzazione e degli utenti del cyberspazio
- I **beni dell'organizzazione e degli utenti** comprendono i dispositivi informatici connessi, personale, infrastrutture e la totalità delle informazioni trasmesse e/o archiviate nel cyberspazio
- Gli **obiettivi** generali di sicurezza comprendono *disponibilità, integrità e confidenzialità*
- *Sottoinsiemi* della sicurezza informatica:
  - **Sicurezza delle informazioni:** conservazione della CIA delle informazioni
  - **Sicurezza delle reti:** protezione delle reti e del loro servizio da modifiche non autorizzate e garanzia che la rete svolga correttamente le sue funzioni critiche

2.4 Sfide della sicurezza informatica

- Non è semplice; può avere requisiti semplici ma **meccanismi di implementazione complessi**
- Nello sviluppo di un meccanismo di sicurezza, si deve sempre **considerare potenziali attacchi**
- Le procedure utilizzate per fornire particolari servizi possono essere **controintuitive poiché complesse**
- Bisogna decidere **dove utilizzare i meccanismi di sicurezza**, sia a livello logico che a livello fisico
- I meccanismi di sicurezza in genere coinvolgono **più di un algoritmo o protocollo**
- Una battaglia **continua** tra attaccante e difensore

2.5 Principi fondamentali di progettazione della sicurezza

- **Fail-safe default:** nel caso in cui il sistema vada in default, deve rimanere in uno *stato protetto*
- **Economia di meccanismo:** i meccanismi devono essere il più semplice possibile
- **Mediazione completa:** *tutti* gli accessi devono essere controllati per assicurarsi che siano consentiti; solitamente accade che solo la prima interazione è controllata
- **Design aperto:** la sicurezza non deve dipendere dalla segretezza della sua progettazione o implementazione
- **Seperazione dei privilegi:** un sistema non dovrebbe concedere l'autorizzazione in base a *una singola* condizione
- **Minimi privilegi:** devono essere concessi il minor numero possibile di privilegi ad ogni soggetto; eventuali permessi addizionali devono essere concesso per il tempo minimo possibile
- **Accettabilità psicologica:** i meccanismi di sicurezza non dovrebbero rendere l'accesso ad una risorsa più difficile
- **Isolamento**
- **Incapsulamento**
- **Modularità**
- **Stratificazione (*layering*)**
- **Minima sorpresa:** evitare che l'utente si trovi davanti a situazioni inaspettate che potrebbero portarlo a seguire comportamenti scorretti

2.6 Superificie di attacco

Una superficie di attacco è costituita dalle **vulnerabilità raggiungibili e sfruttabili** in un sistema, come ad esempio:

- porte aperte verso l'esterno
- interfacce web
- dipendente con accesso a dati sensibili
- ...

→ è necessario **ridurre al minimo** la superificie di attacco

2.6.1 Categorie di attacco

- Superificie di attacco di **rete**: sono incluse vulnerabilità del protocollo di rete, che possono portare a DoS, interruzione dei collegamenti di comunicazioni ed altri attacchi intrusivi
- Superficie di attacco **software**: vulnerabilità nel codice delle applicazioni; un focus particolare è il software per server web
- Superificie di attacco **umano**: vulnerabilità create dal personale o da estranei, come *social engeneering*, errore umano o intrusi

2.7 Albero di attacco

Un albero di attacco è un modo ti **rappresentare le possibilità di attacco**, e quindi di progettare le **contromisure**.

2.8 Tipi di attacco

- **Passivi:** non alterano le informazioni in transito; lo scopo è ottenere informazioni sui messaggi trasmessi
- **Attivi:** modificano il flusso delle informazioni
  - *Attacco di replay:* l'attaccante osserva le informazioni e le riutilizza in un secondo momento per creare una nuova sessione di comunicazione
    $\rightarrow$ ci si tutela con *numeri casuali* e *timestamp* per, rispettivamente, controllare che i messaggi non siano già stati scambiati o che siano ancora validi
  - *DoS e DDoS*
  - ...

2.9 Implementazione della sicurezza

Quattro linee d'azione complementari:

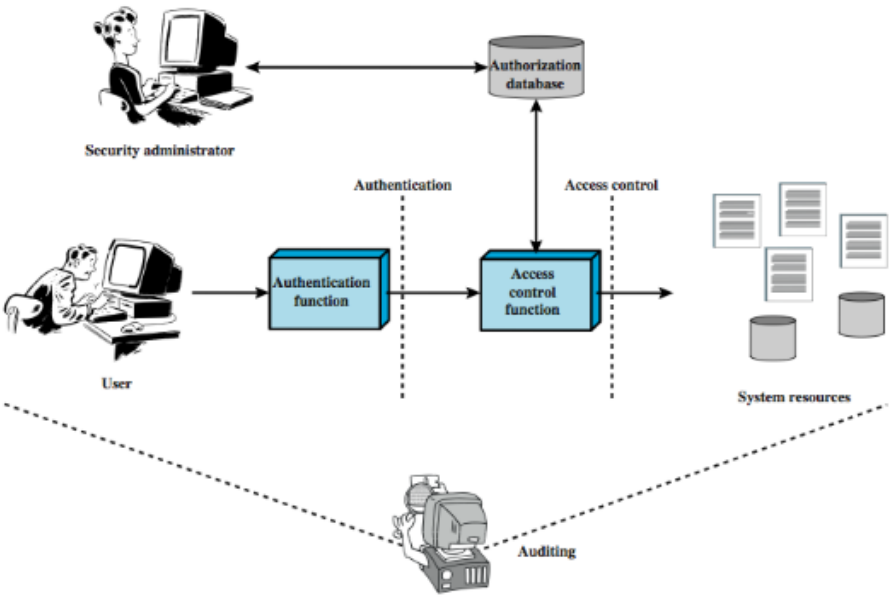
- **Prevenzione**
- **Rilevamento**
- **Risposta** in modo da fermare un attacco e prevenire ulteriori danni
- **Ripristino** con sistemi di backup in caso l'integrità dei dati sia compromessa

Capitolo 3

Controllo degli accessi

Il controllo degli accessi è un elemento centrale nella sicurezza informatica. È il **meccanismo** che definisce una **politica di sicurezza** per la quale si decide quali utenti possano accedere o meno ad una risorsa.
Si basa su tre principi fondamentali:

- **Autenticazione:** verifica che le credenziali fornite siano valide
- **Autorizzazione:** concessione di un permesso ad un'entità affinché possa accedere ad una risorsa del sistema
- **Auditing:** verifica delle attività e dei registri di sistema



3.1 Requisiti di sicurezza

3.1.1 Requisiti di sicurezza di base

- Limitare l'accesso al sistema informativo agli **utenti aurotizzati**, ai processi che agiscono per conto degli utenti autorizzati o ai dispositivi
- Limitare l'accesso al sistema informativo alle tipologie di funzioni che gli utenti autorizzati possono eseguire

3.1.2 Requisiti di sicurezza derivati

- **separare i doveri** dei singoli individui per ridurre il rischio di attività malevole
- utilizzare **account non privilegiati** quando si accede a funzioni non di sicurezza
- **impedire agli utenti non privilegiati** di eseguire funzioni privilegiate e controllare l'esecuzione di tali funzioni
- limitare i **tentativi di accesso** non riusciti
- utilizzare il **blocco della sessione** per nascondere l'accesso ai dati dopo un periodo di inattività
- fornire **avvisi di privacy** e sicurezza secondo le norme vigenti
- **terminare automaticamente** la sessione dopo una determinata azione
- **monitorare e controllare** le sessioni di **accesso remoto**
- usare sistemi **crittografici** per garantire la riservatezza delle sessioni per accessi da remoto
- **instradare l'accesso remoto** tramite punti di controllo degli accessi
- **autorizzare** l'esecuzione remota di **comandi privilegiati**
- **autorizzare l'accesso wireless** prima di consentire tali connessioni

3.2 Elementi di Access Control

- **Soggetto:** entità che può accedere agli oggetti (ad esempio, un processo che rappresenta l'utente)
- **Oggetto:** risorsa ad accesso controllato (file, directory, ...)
- **Diritto di accesso:** modo in cui un soggetto accede ad un oggetto (lettura, scrittura, ...)

3.3 Politiche di controllo degli accessi

- **DAC:** controlla l'accesso in base all'identità del soggetto e alle autorizzazioni che indicano che è/non è consentito fare ai richiedenti
- **MAC:** controlla l'accesso in base al confronto tra delle specifiche etichette di sicurezza applicate agli oggetti e le autorizzazioni di sicurezza
- **RBAC:** controlla l'accesso in base al ruolo che l'utente ha nel sistema e alle regole che stabiliscono quali accessi sono consentiti a quali ruoli
- **ABAC:** controlla l'accesso in base agli attributi dell'utente

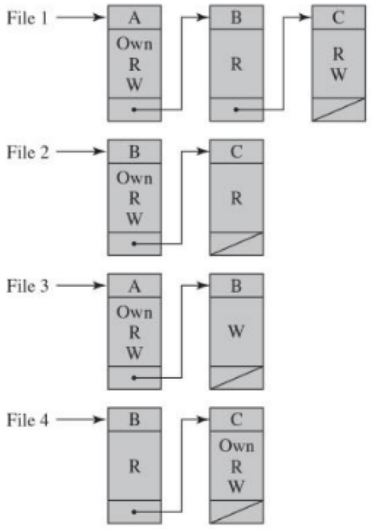
3.3.1 DAC

Il controllo dell'accesso viene fatto sull'**identità del soggetto richiedente** e delle **regole di accesso**. Definito *discrezionale* perché un'entità potrebbe avere i privilegi di accessi che le permettono, a sua volta, di concedere l'accesso ad un'altra entità.
Si può rappresentare mediante una matrice di accesso, dove:

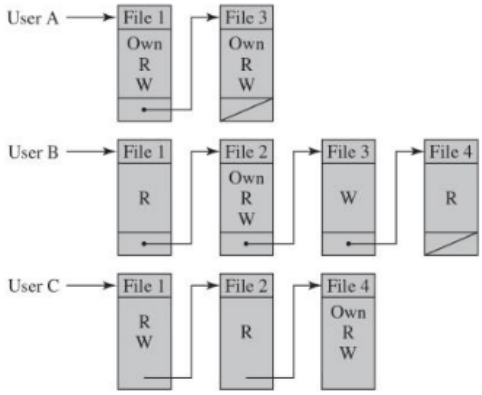
- elenca i soggetti in una dimensione (colonne)
- elenca gli oggetti nell'altra dimensione (righe)
- ogni cella specifica i diritti di accesso di quel soggetto a quel determinato oggetto

	OBJECTS			
	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Questa matrice ha il problema di essere *sparsa*, ovvero molto grande e con molte celle vuote
→ viene trasformata in una serie di liste per risorse ed utenti



(b) Access control lists for files of part (a)



(c) Capability lists for files of part (a)

3.3.2 MAC

Controlla l'accesso in base al confronto tra:

- **Etichette di sicurezza** che indicano quanto sono sensibili le risorse
- **Autorizzazioni di sicurezza** che indicano quali entità del sistema sono idonee ad accedere a quali risorse

Questa politica è definita obbligatoria (*mandatory*) perché un'entità che ha accesso ad una risorsa non può estendere il permesso ad un'altra; può farlo solo l'amministratore di sistema.
I sistemi MAC si dividono in:

- ***Multilevel security systems:*** consiste in una struttura verticale di livelli di sicurezza; agli utenti viene assegnato un livello e possono accedere solo a risorse con livello uguale o inferiore
- ***Multilateral security systems:*** l'accesso viene assegnato in base a segmenti che formano gruppi costituiti da livelli di sicurezza e parole in codice

  → si ottiene una struttura orizzontale, che contiene livelli di sicurezza verticali aggiuntivi

Vantaggi e svantaggi

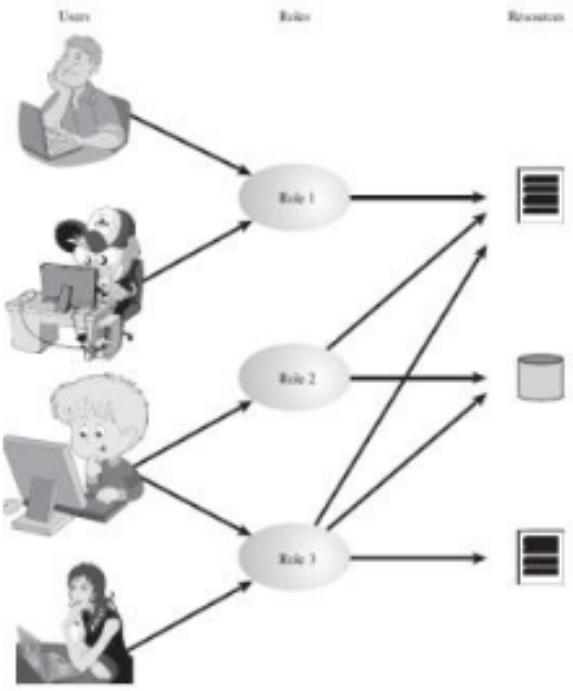
- Il MAC è uno dei sistemi di accesso più sicuri, poiché è praticamente a prova di manomissione

  - gli utenti non possono fare modifiche
  - il controllo è automatizzato
  - i dati non possono essere modificati senza apposita autorizzazione
- ...tuttavia
  - richiede una pianificazione dettagliata e lavoro amministrativo
  - controllare e aggiornare i diritti di accesso
  - manutenzione per aggiunta di nuovi dati o utenti e relative modifiche
  - → elevato carico di lavoro per l'amministratore

3.3.3 RBAC

Ci sono quattro tipi di entità:

- **Utente:** una persona che ha accesso al sistema; ogni individuo ha un ID associato
- **Ruolo:** inteso come una funziona lavorativa all'interno dell'organizzazione
- **Autorizzazione:** approvazione di una modalità di accesso ad uno o più oggetti
- **Sessione:** mappatura tra un utente e un sottoinsieme dei ruoli a cui è assegnato

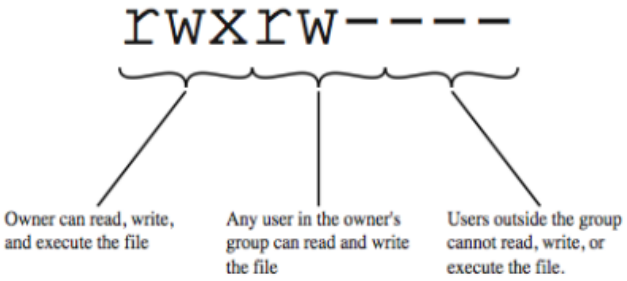


3.4 Unix security model

In Linux ci sono tre entità da considerare:

- **Soggetto:** può essere un utente o un processo
- **Oggetto:** file, cartelle, ...
- **Operazioni consentite:** lettura, scrittura, esecuzione

In Unix, ogni utente ha associato un id univoco, detto **UID**; può appartenere a gruppi di utenti, anch'essi identificati da un id univoco detto **GID**. Tutti gli utenti appartenenti ad un gruppo possono condividere tra loro oggetti.
Ad ogni file è assegnato un unico utente proprietario e un unico gruppo proprietario. L'autorizzazione viene concessa mediante una ACL che identifica le operazioni che i soggetti possono fare.



3.4.1 Processi in Linux

Ogni processo è isolato dagli altri e non possono accedere alla memoria altrui. Ogni processo viene eseguito con le autorizzazione dell'UID dell'utente che lo sta eseguendo.
Nel momento della creazione, ad ogni processo sono assegnati tre ID (inizialmente tutti uguali all'UID):

- **Effective UID:** determina le autorizzazioni per il processo
- **Real UID:** determina l'utente che ha avviato il processo
- **Saved UID:** EUID prima di eventuali modifiche

L'utente *root* può cambiare EUID/RUID/SUID a valori arbitrari; utenti non privilegiati possono cambiare EUID solo a RUID o SUID

3.4.2 Unix file access control

Le modifiche agli ID sono apportate mediante i comandi *setUID* e *setGID*; questa modifica permette ai programmi non privilegiati di accedere a risorse generalmente non accessibili.

Le directory possono aver impostato uno ***sticky bit***: specifica che solo il proprietario di un file nella cartella può apportare una modifica a quel file
Il ***superuser*** è esente dalle consuete restrizioni di controllo degli accessi, ha accesso a tutto il sistema.

3.5 Windows security architecture

L'architettura di sicurezza di Windows è basata su più entità:

- ***Security Reference Model (SRM):*** componente che in modalità kernel esegue controlli delle autorizzazioni e manipola i privilegi degli utenti
- ***Local Security Authority (LSA):*** risiede in un processo utente, è responsabile dell'applicazione della politica di sicurezza locale, tra cui:
  - criteri per le password, come complessità e tempi di scadenza
  - politica di controllo → specifica quali operazioni su quali oggetti vadano controllate
  - impostazioni dei privilegi
- ***Security Account Manager (SAM):*** è un database che archivia i dati degli account e le informazioni di sicurezza su entità locali e gruppi
- ***Active Directory (AD):*** implementa il protocollo LDAP (*Lightweight Directory Access Protocol*)

3.5.1 Windows security model

Windows ha un complesso sistema di controllo dell'accesso; ogni oggetto ha ACL per permettere autorizzazioni granulari ad utenti e/o gruppi di utenti.

3.5.2 Security descriptor

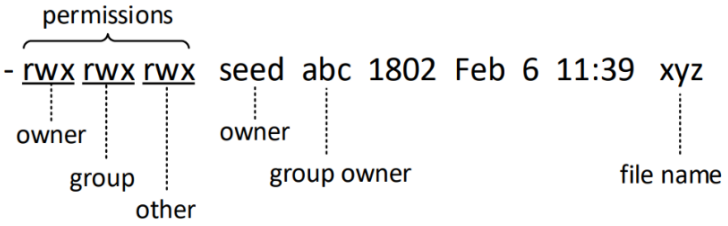
Ogni oggetto ha un *security descriptor* che contiene:

- ***security identifier* (SID)** per il possessore e il gruppo primario dell'oggetto (SID è associato univocamente ad ogni utente)
- ***discretionary ACL* (DACL):** diritti di accesso per gli utenti e i gruppi
- ***system ACL* (SACL):** tipi di accesso che generano log

Ad ogni processo viene inoltre associato un insieme di **token**, che prende il nome di ***security context***. Nel momento in cui un processo vuole accedere ad un oggetto, presenta il suo insieme di token e il sistema controlla se il security context abbia o meno accesso a tale risorsa in base al *security descriptor* dell'oggetto.

3.6 Set-UID

In Linux, ad ogni utente viene assegnato un ID univoco, memorizzato in `/etc/password`
Tramite i gruppi si rappresentano i gruppi di utenti che hanno le stesse autorizzazioni.
Le autorizzazioni su file vengono espresse nel seguente modo:



Per **set-UID** si intende **consentire all'utente di eseguire un programma con i privilegi del proprietario del programma**. Quando viene eseguito un normale programma, $RUID = EUID$, con entrambi che equivalgono all'id dell'utente che esegue il programma.
Quando viene eseguito un set-uid, RUID è uguale all'id dell'utente, ma EUID è uguale all'id del proprietario del programma

→ se il programma è di proprietà di *root*, viene eseguito con i *privilegi di root*

3.6.1 Superificie di attacco dei programmi Set-UID

- **Attacchi tramite input utente:** potrebbe essere inserito del codice malevolo, ad esempio tramite:
  - buffer overflow
  - inserimento di stringhe formattate appositamente
  - comando `chsh` → permette di cambiare sheòò tra quelli presenti in `/etc/password`

  → problema: è impossibile disinfettare gli input dell'utente
- **Attacchi tramite input di sistema:** cambiare i riferimenti simbolici a file privilegiati
- **Attacchi tramite variabili d'ambiente:** le variabili d'ambiente possono essere impostate prima dell'esecuzione di un programma e influenzarne il comportamento
- **Perdita di capacità:** i programmi privilegiati potrebbero declassarsi durante l'esecuzione

- **Invocazione di programmi:** invocazione di comandi esterni dall'interno di un programma
  - agli utenti viene chiesto di fornire dati in input al comando; se non vengono fatti gli opportuni controlli, i dati di input dell'utente potrebbero essere trasformati nel nome di un comando
- **Attacchi tramite dynamic linker:** se il programma prevede linking dinamico di librerie, anche il contenuto diventa rilevante; se manomesse, potrebbero portare a comportamenti anomali del codice

Capitolo 4

Politiche di sicurezza

La *gestione della sicurezza* è un *processo formale* per rispondere alle domande:

- quali sono i beni da proteggere
- quali sono le possibili minacce
- come si possono contrastare le minacce

Questo processo ha natura iterativa, ed è contenuto nella ISO 31000; in questa norma viene descritto un *modello per la gestione della sicurezza delle informazioni* che comprende le seguenti fasi:

- **Plan:**
  - stabilire politiche, processi e procedure di sicurezza
  - eseguire la valutazione del rischio
  - sviluppare un piano di trattamento del rischio
- **Do:**
  - implementare il piano di trattamento del rischio
- **Check:**
  - monitorare e mantenere il piano di trattamento del rischio
- **Act:**
  - mantenere e migliorare la gestione dei rischi
  - risposta ad incidenti, analisi di vulnerabilità e riprendere il ciclo iterativamente

4.1 Definizioni

- **Rischio:** esprime la possibilità che un attacco causi danni ad una organizzazione
- **Risorsa:** tutto ciò che necessita di essere protetto
  - *hw*
  - *sw*
  - *reputazione*

  La valutazione di una risorsa viene fatta in base:
  - ai costi da sostenere per sostituire la risorsa nel caso non sia più disponibile
  - perdita di incassi in caso di attacco
- **Vulnerabilità:** punti deboli che possono essere sfruttati per causare danni al sistema; possono essere classificate come:
  - critico
  - moderato
  - basso

Capitolo 5

Malware

Una definizione *informale* può essere quella di programma **malevolo**, solitamente inserito di nascosto in un sistema, che ha lo scopo di compremettere la riservatezza, l'integrità o la disponibilità del sistema stesso.
I malware sono classificati in base a:

- **Propagazione** (sw, rete, social engeneering, . . .)
- **Azioni sui dati** colpiti (corruzione, furto, crittografia, . . .)
- **Attack kit** (strumenti già pronti per attaccare)
- **Attori e/o motivazioni** dell'attacco

Name	Description
Advanced Persistent Threat (APT)	Crimini informatici diretti contro obiettivi aziendali e politici, utilizzando un'ampia varietà di tecnologie di intrusione e malware, applicati in modo persistente ed efficace a obiettivi specifici per un periodo prolungato, spesso attribuiti a organizzazioni sponsorizzate dallo stato.
Adware	Pubblicità integrata nel software. Può provocare annunci pop-up o il reindirizzamento di un browser a un sito commerciale.
Attack kit	Insieme di strumenti per generare automaticamente nuovo malware utilizzando una varietà di meccanismi di propagazione e carico utile forniti.
Auto-rooter	Strumenti di hacker dannosi utilizzati per penetrare in nuove macchine da remoto.
Backdoor (trapdoor)	Qualsiasi meccanismo che eluda un normale controllo di sicurezza; potrebbe consentire l'accesso non autorizzato alle funzionalità di un programma o a un sistema compromesso.
Downloaders	Codice che installa altri elementi su un computer sotto attacco. Normalmente è incluso nel codice malware inserito prima in un sistema compromesso per poi importare un pacchetto malware più grande.
Drive-by-download	Un attacco che utilizza codice su un sito Web compromesso che sfrutta una vulnerabilità del browser per attaccare un sistema client quando viene visualizzato il sito.
Exploits	Codice specifico per una singola vulnerabilità o un insieme di vulnerabilità

Name	Description
Flooders (DoS client)	Utilizzato per generare un grande volume di dati per attaccare i sistemi informatici in rete, eseguendo una qualche forma di negazione del servizio (DoS) attacco.
Keyloggers	Cattura le sequenze di tasti su un sistema compromesso.
Logic bomb	Codice inserito nel malware da un intruso. Una bomba logica rimane dormiente finché non viene soddisfatta una condizione predefinita; il codice quindi attiva un carico utile.
Macro virus	Un tipo di virus che utilizza codice macro o script, generalmente incorporato in un documento o modello di documento e attivato quando il documento viene visualizzato o modificato, per essere eseguito e replicarsi in altri documenti simili.
Mobile code	Software (ad esempio, script e macro) che può essere distribuito senza modifiche a un insieme eterogeneo di piattaforme ed eseguito con la stessa semantica.
Rootkit	Insieme di strumenti hacker utilizzati dopo che l'aggressore è entrato in un sistema informatico e ha ottenuto l'accesso a livello di root.
Spammer programs	Utilizzato per inviare grandi volumi di posta elettronica indesiderata.
Spyware	Software che raccoglie informazioni da un computer e le trasmette a un altro sistema monitorando sequenze di tasti, dati sullo schermo e/o traffico di rete; o eseguendo la scansione dei file sul sistema alla ricerca di informazioni sensibili.

Name	Description
Trojan horse	Un programma per computer che sembra avere una funzione utile, ma ha anche una funzione nascosta e potenzialmente dannosa che elude i meccanismi di sicurezza, a volte sfruttando le autorizzazioni legittime di un'entità di sistema che lo invoca.
Virus	Malware che, una volta eseguito, tenta di replicarsi in un altro codice macchina o script eseguibile; quando ha successo, si dice che il codice è infetto. Quando viene eseguito il codice infetto, viene eseguito anche il virus.
Worm	Un programma per computer che può essere eseguito in modo indipendente e può propagare una versione completa e funzionante di se stesso su altri host di una rete, sfruttando le vulnerabilità del software nel sistema di destinazione o utilizzando credenziali di autorizzazione acquisite.
Zombie, bot	Programma installato su un computer infetto che viene attivato per lanciare attacchi su altri computer.

5.1 Trojan

È un programma che ha un effetto evidente e atteso dall'utente, che ha però anche un effetto **nascosto** che viola le politiche di sicurezza e che viene condotto senza l'autorizzazione dell'utente

5.2 Virus

È un codice che può replicarsi modificando altri file o programmi per inserire codice in grado di replicarsi a sua volta; questa **proprietà di replicazione** è ciò distingue i virus dagli altri tipi di malware. Non svolge nessuna azione evidente, ma cerca di rimanere nell'ombra.

La replica richiede un certo tipo di assistenza da parte dell'utente, come ad esempio cliccare su un allegato.
Un virus è composto da tre parti:

- **Meccanismo di infezione**
- **Trigger:** evento che determina quando il payload viene attivato
- **Payload:** cosa fa il virus (oltre a diffondersi)

I virus attraversano quattro fasi:

1. **Dormiente:** il virus è inattivo in attesa di essere attivato
2. **Scatenante:** il virus viene attivato
3. **Propagazione:** il virus inserisce una copia di sé stesso in certe parti del sistema; ogni programma infetto conterrà ora un altro virus che entrerà a sua volta in fase di propagazione
4. **Esecutiva:** la funzione viene eseguita

5.2.1 Vettori di infezione

I principali vettori di infezione sono:

- **Boot sector** di dispositivi esterni; il codice è inserito nel boot sector e viene eseguito in fase di avvio
- **Eseguibili**
- **File macro:** il virus si attacca ai documenti per propagarsi

5.2.2 Classificazione dei virus

È possibile classificare i virus in base alle **tecniche usate per superare i controlli di sistema:**

- **Cifratura del virus:** crea una chiave per *"crittografarsi"*; quando viene chiamato un programma infetto, con tale chiave viene decifrato il virus. Per evitare pattern di bit, durante la propagazione la chiave viene cambiata
- **Stealth virus:** si nasconde dal rilevamento da parte dell'antivirus, tramite mutazione o compressione del codice
- **Polymorphic virus:** durante la replica crea copie che svolgono la stessa funzione ma che hanno pattern di bit diversi
- **Metamorphic virus:** si riscrive completamente ad ogni iterazione per aumentare la difficoltà di rilevamento
- **Compression virus:** comprimono il file eseguibile in modo che la versione infetta abbia la stessa dimensione di quella originale

5.3 Worm

I worm sono programmi *stand alone* (a differenza dei virus che devono essere attivati da un qualche evento) in grado di replicarsi.
Le fasi di esecuzione sono:

- **Probing:** cerca informazioni sulla macchina
- **Expoloitation:** sfrutta le informazioni raccolte per trovare vulnerabilità
- **Replicazione**
- **Attacco** (payload)

5.4 Drive-by-download

Sfruttano **vulnerabilità del browser** per installare codice malevolo ad insaputa dell'utente nel momento in cui visita la pagina web dell'attaccante.

5.5 Clickjacking

L'attaccante intercetta un *click* dell'utente per costringerlo a fare delle cose contro la sua volontà.

5.6 Zombie e botnet

Lo *zombie* è una singola macchina, mentre la *botnet* è un insieme di macchine zombie controllate da una singola entità; vengono usate per fare DDoS, phishing, spamming, ...

5.7 Rootkit

È un insieme di programmi installati su un sistema per mantenere l'accesso ad un sistema, ad esempio, con privilegi di amministratore, nascondendo le prove della sua presenza e aggirando i meccanismi di controllo.
Permettono di fare attacchi anche con scarse conoscenze tecniche.

5.8 Scareware

Software che hanno lo scopo di diffondere shock, ansia e/o la percezione di una minaccia; sono un attacco di *social engeneering*.

5.9 Ransomware

Software che tiene in ostaggio il sistema per richiedere un riscatto all'utente, spesso tramite cifratura.

5.10 Vulnerabilità zero-day

Si intende un'exploit non ancora nota e che non ha quindi una contromisura.

5.11 Spear phishing

Viene **studiato nel dettaglio il bersaglio**, in modo tale da fare del phishing più mirato ed efficace.

5.12 Spyware

Malware che raccoglie piccole informazioni alla volta sugli utenti a loro insaputa, come ad esempio un *keylogger*.

5.13 APT - Advanced Persistent Threats

- ***Advanced:*** è un'applicazione con un ampia varietà di tecnologie di intrusione e malware
- ***Persistent:*** attacchi per un periodo di tempo prolungato verso il target
- ***Target:*** target selezionati in modo accurato

Le fasi principali di un attacco tramite APT sono:

- **Ricognizione:** si sceglie una vittima e la si studia
- **Weaponization:** si mette un trojan che permette accesso remoto in un payload consegnabile (email, USB, web)
- **Sfruttamento:** il codice malevolo viene attivato per portare a termine il suo scopo

5.14 Approcci alle contromisure per i malware

Le principali contromisure da adottare sono:

- assicurare di **aggiornare** tutti i sistemi
- ridurre al **minimo le vulnerabilità**
- impostare adeguati **controlli** di accesso alle applicazioni e ai dati
- **limitare** il numero di file a cui un utente può accedere, e che quindi può potenzialmente infettare

Le azioni da prendere nel caso il sistema venga attaccato sono:

- **Rilevamento:** accertarsi della presenza del virus; può essere fatto con:
  - programmi anti-virus, 4 generazioni:
    - **scanner semplici** → ricercano malware noti
    - **scanner euristici** → utilizzano regole euristiche
    - **trap di attività:** viene cercato il malware in base alle sue azioni piuttosto che in base alla sua struttura → l'analisi è dinamica
    - **protezione completa:** più tecniche usate insieme
  - meccanismi di protezione perimetrale nei firewall
- **Identificazione:** individuare lo specifico malware
- **Rimozione** di tutte le tracce dal sistema

Capitolo 6

Autenticazione

Processo per determinare se un utente, un'applicazione o un processo che agisce per conto di un utente **è effettivamente chi o cosa dichiara di essere**; si fa un confronto tra le credenziali inserite e quelle memorizzate nel sistema.

6.1 Principi di autenticazione

- **Identità digitale:** è la rappresentazione unica di un soggetto, consiste in un attributo o un insieme di attributi che descrivono univocamente un soggetto
- **Prova dell'identità:** stabilisce che un soggetto è chi afferma di essere ad un determinato livello di certezza
- **Autenticazione digitale:** il processo di determinazione della validità di uno o più autenticatori utilizzati per rivendicare un'identità digitale

6.2 Requisiti di sicurezza

- **Di base:**
  - **identificare** gli utenti del sistema informativo e i processi che agiscono per loro conto
  - autenticare (o **verificare**) le identità di tali utenti
- **Derivati:**
  - utilizzare l'**autenticazione a più fattori**
  - impiegare meccanismi resistenti ad **attacchi di replay**
  - impedire il riutilizzo degli *id* per un certo intervallo di tempo
  - disabilitare gli *id* dopo un periodo di inattività

- applicare una **complessità minima** della password
- proibire il **riutilizzo** delle password
- memorizzare e trasmettere solo password **criptate**
- **oscurare il feedback** delle informazioni di autenticazione

6.3 Metodi di autenticazione

- **qualcosa che uno sa:** password, ...
- **qualcosa che uno ha:** token → smartcard, chiave fisica, ...
- **qualcosa che l'individuo è:** tratto biometrico
- **qualcosa che l'individuo fa:** biometria dinamica

6.4 Password

Lo schema standard *id-password* è soggetto a diverse vulnerabilità:

- canale tra client e server
- compromissione del client
- compromissione del server
- social engeneering
- password deboli

Esistono diversi tipi di attacco alle password:

- attacchi a **dizionario**
- **forza bruta**
- uso della **stessa password** per più servizi

6.4.1 Memorizzazione delle password

Nello vecchio sistema UNIX veniva memorizzato nel file `/etc/passwd` l'hash delle password insieme al suo id.
Oggi, la password viene salvata come **hash** nel file `/etc/shadow`, che è **leggibile solo da *root***.
Per proteggersi da attacchi di forza bruta, è possibile usare un **salt**: invece che memorizzare l'hash della password, ad esso viene concatenato all'inizio e/o alla fine un *numero casuale* di lunghezza arbitraria; vengono usati salt fino a 48 bit.

6.4.2 Cracking delle password

- attacchi di dizionario
- tabelle *rainbow*: contengono hash precalcolati di diversi dizionari, con già inseriti dei salt

La maggioranza degli attacchi **sfrutta password deboli**; la soluzione è costringere gli utenti ad usare **password complesse**.

6.4.3 Utilizzo delle password su canali non sicuri

Esistono tre approcci:

- **One-time-password**
- **Challenge response:** chi si autentica dimostra di conoscere un segreto (challenge); di norma vengono usati anche timestamp per evitare attacchi di replay
- **Zero knowledge proof:** chi si autentica fornisce una prova di conoscere la password senza fornirla esplicitamente

6.5 Altri metodi di autenticazione

- **Smart token:** dispositivi dotati di **capacità computazionale**, possono fare diversi tipi di controlli
- **Smart card:** forniscono informazioni una volta inserite nel sensore, non hanno sistemi di calcolo
- **Autenticazione biometrica**
- **Autenticazione reciproca:** permettono ad entrambe le parti di accertarsi reciprocamente della propria identità

6.6 Problemi di sicurezza dell'autenticazione

- **Intercettazione**, ad esempio con un attacco che implica la vicinanza fisica di utente ed attaccante
- **Attacchi host:** diretti al file dell'utente in cui sono contenute le password/token/...
- **Replay:** l'avversario ripete una risposta dell'utente acquisita in precedenza, senza che il server sia in grado di distinguerla; si può contrastare con tre approcci:
  - allegare un **numero di sequenza** a ciascun messaggio
  - **timestamp** per verificare la validità
  - **challenge-response:** quando una parte invia un messaggio, deve contenere il *nonce* del messaggio precedente
- **Attacchi client:** l'avversario cerca di autenticarsi senza accedere all'host remoto
- **DoS:** tenta di disabilitare un servizio

Capitolo 7

Modello ISO-OSI

Ad oggi internet è il più grande sistema ingegneristico mai creato dall'umanità, che comprende miliardi di utenti e macchine.
Le operazioni chiave nella rete sono:

- invio e consegna di pacchetti
- instradamento dei pacchetti
- individuazione degli indirizzi

7.1 Protocolli

Sono un **insieme formale di regole** durante un'interazione; descrivono come i computer trasmettono i dati attraverso una rete, come il formato e l'ordine dei messaggi, oltre alle azioni intraprese sulla trasmissione e/o ricezione di essi.

7.2 Layering

Per far fronte alla complessità, i protocolli sono organizzati in uno *stack* di livelli
→ i dati vengono passati dal livello più alto a quello più basso
Questo facilita la manutenzione e l'aggiornamento del sistema.

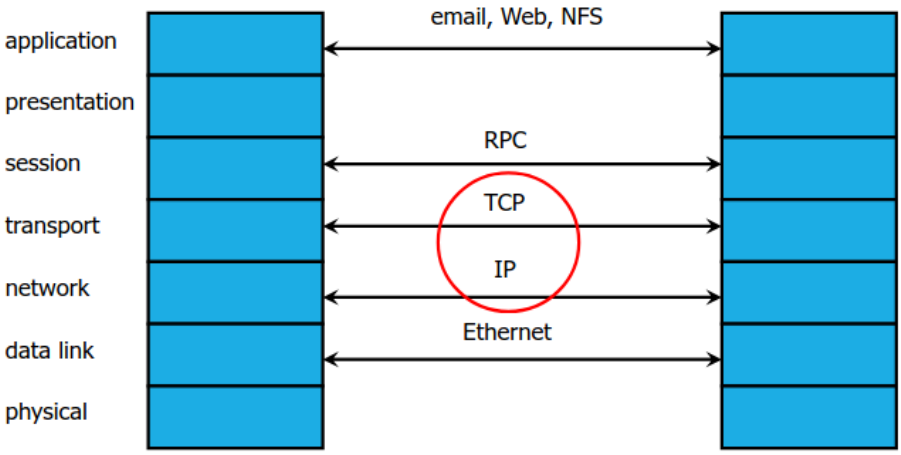


Figura 7.1: OSI stack

7.3 Principio end-to-end

Le funzionalità specifiche dell'applicazione risiedono nei nodi finali di comunicazioni della rete, piuttostoc che nei nodi intermedi come *gateway* e *router*; l'*intelligenza* sta ai vertici della rete.

7.4 Tipi di indirizzi in Internet

- **MAC** nel livello di accesso alla rete
  - associata alla scheda con interfaccia di rete
  - 48/64 bit
- **IP** per il livello di rete
  - 32bit per IPv4, 64bit per IPv6
  - Ad esempio, 128.3.23.3
- **IP + porte** per il livello di trasporto
  - Ad esempio, 128.3.23.3:80
- **Dominio** per livello di applicazione/umano
  - Ad esempio, www.inter.it

7.4.1 Routing e traduzione degli indirizzi

- Routing tra indirizzi IP ed indirizzi MAC
  - protocollo di risoluzione degli indirizzi (ARP) per IPv4
  - Neighbour Discovery Protocol (NDP) per IPv6
- Routing con indirizzi IP
  - TCP, UDP, IP per instradare pacchetti
  - Border Gateway Protocol per gli aggiornamenti delle tabelle di routing
- Traduzione da indirizzi IP a nomi di dominio
  - DNS

7.5 Interfaccia di rete

Un computer può avere più interfaccie di rete; i pacchetti vengono trasmessi tra le interfacce di rete.
Impara l'indirizzo MAC di ogni computer ad esso connesso ed inoltra i frame solo al computer di destinazione.

7.6 Netmask

La netmask è una sequenza di 32 bit che identifica quali bit sono comuni negli indirizzi IP all'interno di una LAN (sottorete).

Ad esempio, 159.132.30.0/24 $\rightarrow$ 24 bit per la sottorete

7.7 Tipi di minacce nella rete

- Confidenzialità (*packet sniffing*)
- Integrità (*session hijacking*)
- Disponibilità (DoS)
- *Translation poisoning, routing*

7.8 ARP (Address Resolution Protocol)

Il protocollo ARP connette il livello di rete con il livello dati, **convertendo un indirizzo IP in un indirizzo MAC**.
Ogni nodo mantiene una tabella (*ARP cache*) in cui ci sono le associazioni già note; altrimenti, si chiede a tutti i nodi della rete locale chi ha un certo indirizzo IP.

→ ARP funziona inviando n messaggi broadcast e memorizzando nella cache le risposte, per utilizzi futuri:

- *ARP request* (effettuata in broadcast)
  - `who has <IP1> tell <IP2>`
- *ARP reply*
  - `<IP1> is <MAC>`

7.8.1 ARP spoofing - Cache poisoning

Viene fatta una *assunzione di trust* nella LAN, dato che:

- le richieste non vengono tracciate
- gli annunci non sono autenticati
- le macchine si fidano una dell'altra

→ una macchina malevola puà ingannare le altre
Dato che una *cache ARP* si aggiorna ogni volta che riceve una risposta ARP anche se non ha inviato alcuna richiesta ... è possibile *avvelenare* una cache inviando delle risposte *"gratuite"*!

7.9 MAC address e spoofing

Le schede di rete sono indentificate da un numero seriale, che viene utilizzato come indirizzo all'interno della LAN.
Con i privilegi adeguati, è quasi sempre possibile (e facile) cambiare il numero MAC usato nella produzione dei frame

→ conoscendo il MAC di una macchina assente, è immediato impersonarla

7.10 MAC address flooding

Switch

Uno switch è un dispositivo che opera a livello di collegamento (link); ha più porte, ciascuna collegata ad un computer. Quando i frame arrivano sulle porte dello switch, gli indirizzi MAC di orgine vengono appresi dall'intestazione del pacchetto e registrati in una tabella.

- Se lo switch conosce già l'indirizzo MAC, inoltra il frame alla porta dell'indirizzo MAC
- Se non esiste, lo switch funge da hub e inoltra il frame su ogni altra porta dello switch, mentre memorizza il MAC per la prossima volta

Attacco

L'attacco di flooding viene spesso chiamato anche come attacco di overflow della tabella degli indirizzi MAC, dato che ha dimensioni limitate.
Il MAC flooding **invia un intero gruppo di indirizzi MAC di origine falsi**, fino a quando la tabella è satura e non può più salvare l'indirizzo MAC

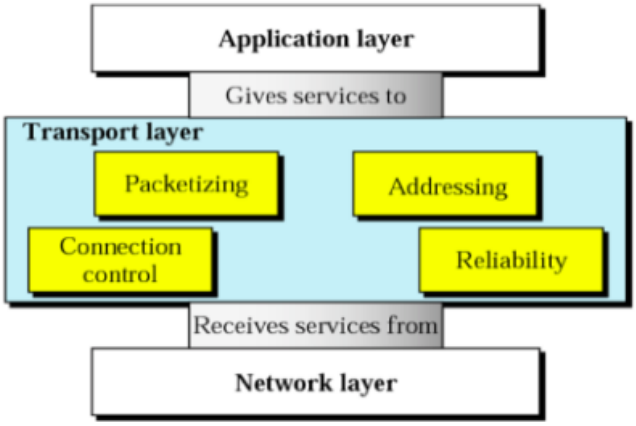
→ lo switch comincerà a trasmettere tutti i pacchetti ricevuti a tutte le altre macchine sulla rete
I tool di attacco generano circa 160.000 voci MAC al minuto.

Capitolo 8

Attacchi TCP/IP

8.1 Livello di trasporto

La responsabilità principale del livello di trasporto è la consegna dei dati da processo a processo. I singoli processi in esecuzione su un dato host sono identificati dai loro numeri di porta.



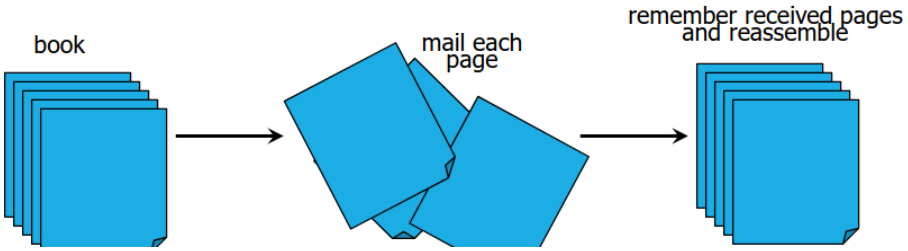
I due protocolli del livello di trasporto utilizzati in internet sono TCP e UDP.

8.1.1 Porte

Un numero di porta è un intero a 16 bit che assume un valore comprso tra 0 e 65535. I numeri di porta tra 0 e 1023 (i cosidetti *numeri di porta noti*) sono riservati ad applicazioni come HTTP, ...

8.2 TCP (Transmission Control Protocol)

- **Mittente:** suddivide i dati in pacchetti; ad ogni pacchetto è associato un numero di sequenza
- **Ricevitore:** riassembla i pacchetti nell'ordine corretto; i pacchetti persi vengono rispediti



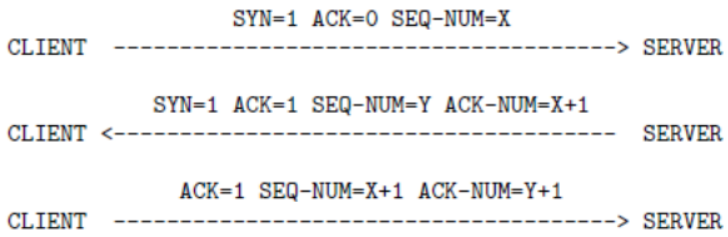
8.2.1 Flag TCP

- **SYN:** richiesta di connessione, primo pacchetto della comunicazione
- **FIN:** intenzione del mittente di terminare la sessione
- **ACK:** conferma del pacchetto precedente
- **RST:** reset della sessione
- **URG:** dati urgenti che vengono inviati con precedenza sugli altri (es. CTRL+C)

8.2.2 TCP handshake

Le connessioni TCP vengono stabilite tramite un handshake a tre vie:

- il server ha generalmente un listener passivo, in attesa di una richiesta di connessione
- il client richiede una connessione inviando un pacchetto SYN
- il server risponde inviando un pacchetto SYN/ACK, indicando un riconoscimento per la connessione
- il client risponde inviando un ACK al server, stabilendo così la connessione



8.2.3 Problemi intrinsechi

- **Non c'è autenticazione** fra le parti; un utente malevolo potrebbe intromettersi nella connessione fintanto che usa un *sequence number* corretto e gli indirizzi IP sono corretti
- i **controlli di integrità** sono banali

8.3 Spoofing

8.3.1 TCP spoofing

Ogni connessione TCP ha uno stato associato:

- numero di sequenza
- numero di porta

→ è facile da indovinare, dato che si usano numeri di porta standard e numeri di sequenza prevedibili
È dunque possibile iniettare pacchetti in connessioni esistenti:

- se l'attaccante conosce il numero di sequenza iniziale e la quantità di traffico, può indovinare il probabile numero corrente
- altrimenti, dato che la maggior parte dei sistemi accetta grandi finestre di numeri di sequenza per gestire le perdite di pacchetti, invia un flusso di pacchetti con probabili numeri di sequenza

8.3.2 IP spoofing

Le autenticazioni locali su indirizzi IP sono insicure, specialmente all'interno di rete locali; l'header IP è falsificabile senza particolari difficoltà.

- **Blind spoofing**

  Attacca da qualsiasi fonte; cerca di impersonare un host qualunque di internet, non facente parte della sottorete in cui si trova.

  Blind perché l'attaccante non ha nessuna possibilità di vedere i pacchetti mandati in risposta ai pacchetti *"spoofati"* che ha spedito; questi ultimi

saranno infatti spediti al vero host che l'attaccante sta impersonando, impedendogli quindi di venire a conoscenza di *ack number* e *sequence number* corretti per continuare la connessione.

- **Non-blind spoofing**

  L'attaccante si trova nella stessa rete della vittima, quindi sarà in grado di intercettare i pacchetti in risposta a quelli fraudolenti che ha inviato (tramite sniffer), e quindi di ottenere il sequence number corretto.

I passaggi di IP spoofing sono:

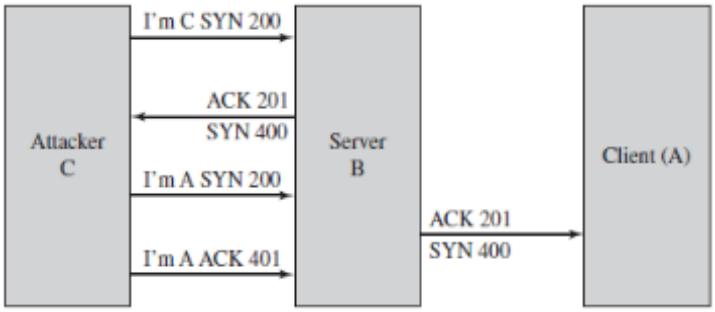
1. viene **scelta una vittima** e cercato un host considerato come fidato dalla vittima
2. si **disabilita l'host fidato** (ad esempio tramite SYN flooding)
3. a questo punto si **contatta la vittima** e si ottiene un numero di sequenza valido, per provare a predire il prossimo numero di sequenza
4. l'attaccante manda un pacchetto SYN utilizzando come IP mittente quello della macchina fidata. Anche se l'hacker non riceve il SYN/ACK, può sempre rispondere in modo che la vittima pensi di aver stabilito correttamente la connessione

La disabilitazione dell'host fidato è parte fondamentale, in quanto alla ricezione del pacchetto SYN/ACK potrebbe interrompere la connessione tramite un pacchetto RST.

8.4 TCP session hijacking

L'obiettivo è aprire una connessione con il server **impersonando la vittima**;

- L'attaccante apre una connessione autentica con il suo bersaglio (B), ricevendo un numero di sequenza
- L'attaccante (C) si spaccia per il client (A)
  - invia un pacchetto con l'indirizzo del client (A) nel campo dell'indirizzo sorgente e un numero di sequenza coerente con ciò che B si aspetta
- Se l'attaccante indovina il numero di sequenza, il server presume di avere una connessione con A
  - L'attaccante non può vedere l'output di questa sessione, ma potrebbe eseguire comandi con i privilegi di A sul server



8.5 ACK storm

Gli attacchi ACK storm si basano sul fatto che, quando si riceve un pacchetto con ACK più grande di quello inviato dal client ricevente, il client deve inviare l'ultimo pacchetto inviato all'altro lato e scartare il pacchetto ricevuto.
L'attacco prevede di:

- prelevare un pacchetto da una connessione TCP tra un client e un server
- generare due pacchetti, ciascuno inviato a una parte con l'indirizzo IP dell'altra
- inviare i pacchetti contemporaneamente

→ la connessione entra in un ciclo infinito di ACK.
Come contromisura a questo attacco, si è modificato il protocollo facendo in modo che il pacchetto viene scartato senza dover rimandare la risposta.

8.6 Attacco DoS

Lo scopo è quello di impedire alla vittima di rispondere alle richieste; esistono due tipi di DoS:

- **DoS bug:** sfrutta difetti di progettazione della vittima
- **DoS flood:** vengono sfruttate delle bot-net per inondare di richieste la vittima
  - un modo è quello di usare **ACK reflection attack:**
    L'attaccante invia del traffico con l'indirizzo IP della vittima ad un server intermedio (detto *reflector*), che si occuperà di mandare le risposte alla vittima
    → si può mandare una serie di SYN al reflector che inonderà la vittima di SYN/ACK; la vittima vedrà l'IP del reflector e non quello dell'attaccante

8.6.1 SYN flood

Prevede che l'attaccante **continui ad inviare richieste SYN al server** senza poi rispondere ai SYN/ACK ricevuti in risposta; in questo modo, il server dovrà dedicare risorse per rispondere ed attendere inutilmente che le connessione TCP aperte terminino l'handshake. L'idea è quella di impedire al server di poter rispondere ad ulteriori richieste di connessione legittime.

8.6.2 Contromisure ad attacco DoS

Ci sono quattro approcci principali:

- **aumentare le dimensioni del *backlog*** (coda in cui il server memorizza le connessioni iniziate che non hanno ancora terminato l'handshake)
- ridurre il **SYN-Received timer**, ovvero il tempo per cui una connesisone rimane nel backlog prima che venga scartata
- **SYN cookies:** il server inserisce le informazioni che di norma si salverebbero nel backlog all'interno di un cookie che viene inserito nella risposta SYN/ACK inviata al client. Solo nel momento in cui in server riceve risposta allora aprirà la connessione e le dedicherà risorse.

  Il cookie deve essere firmato e non modificabile; viene calcolato nel seguente modo

  - i primi **5 bit** sono dati da $tmod32$, con t **timestamp**
  - i seguenti **3 bit** sono la codifica del numero di entry che il server avrebbe salvato nel backlog
  - gli ultimi **24 bit** sono il risultato di una funzione di hash, che prende in input:
    - input IP server
    - IP client
    - porta del server
    - porta client
    - timestamp

  Nel momento in cui il client manda l'ACK in risposta con allegato il cookie, la **verifica** ha i seguenti passi:

  1. viene verificato il timestamp per vedere se il pacchetto sia ancora valido
  2. vien ricomputata la funzione di hash per vedere se il SYN sia ancora valido
  3. vengono decodificati i 3 bit corrispondenti alla entry della coda di backlog

- **cancellazione di una entry a caso** per far posto a una nuova
- uso di un **prolexic proxy:** si mette un proxy tra server e client; sarà il proxy a gestire l'handshake, ed inoltrerà al server solo le richieste SYN di cui si riceve anche un ACK

8.7 UDP (User Datagram Protocol)

UDP è un protocollo di trasporto minimale che si limita ad inviare le informazioni sulla rete senza preoccuparsi che queste arrivino effettivamente a destinazione; non c'è nessun controllo del flusso e nessun riconoscimento.
Viene usato ad esempio per applicazioni di streaming.

8.7.1 DDoS - NTP amplification

Il protocollo NTP è usato per sincronizzare i *clock* tra macchine diverse tramite UDP.
L'attacco consiste nello sfruttare il comando `monlist`, abilitato di default nelle vecchie versioni del protocollo, che permette di ottenere gli ultimi 600 IP che hanno fatto domanda al server NTP; successivamente si inonda di traffico UDP la vittima facendo IP spoofing.
Passaggi:

- L'attaccante usa una botnet per inviare pacchetti UDP con IP mittente della vittima al server NTP, con i quali richiede l'esecuzione del comando `monlist`
- ogni pacchetto effettua una richiesta al server NTP che risponde all'IP della vittima
- la rete della vittima viene inondata di traffico, circa 20 volte superiore a quello delle richieste inviate dall'attaccante (la richiesta del comando `monlist` è circa 200 volte più piccola dell'output del comando stesso)

L'attacco viene definito di *amplificazione* perché da una banda piuttosto piccola è possibile ricavare grosse quantità di dati con cui effettuare un DDoS tramite la *reflection* del server NTP.

8.8 ICMP (Internet Control Message Protocol)

È un protocollo che permette la comunicazione tra host e router, per scambiare informazioni relative allo stato della rete (host irraggiungibili, echo delle richieste/risposte . . .).

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

8.8.1 Attacco *smurf* DDoS

L'attacco prevede l'utilizzo di una **richiesta ICMP echo** (ping) e l'**IP spoofing** per inviare alla vittima una quantità di dati di risposte echo che rendono inservibili altre richieste.
Passaggi:

- l'attaccante usa come mittente in una richiesta ICMP echo l'IP della vittima all'indirizzo di broadcast della rete

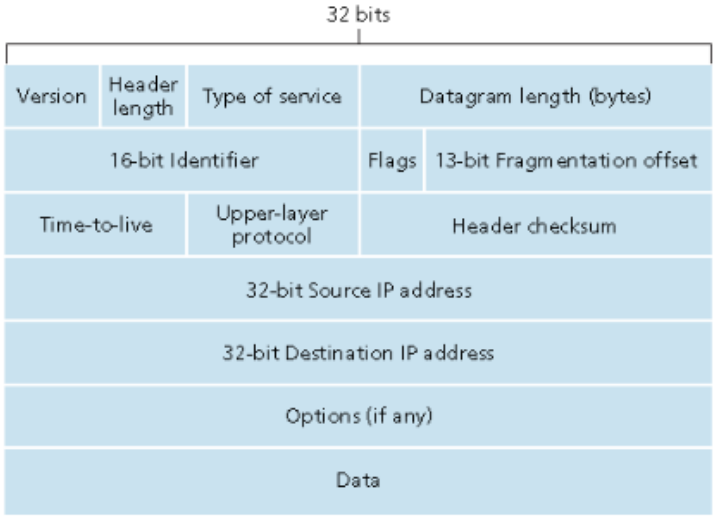
  → la richiesta viene inoltrata a tutti gli host della rete

- tutti gli host rispondono all'IP della vittima, causando il DDoS

La **contromisura** può essere scartare tutti i pacchetti che hanno come destinazione l'indirizzo di broadcast.

8.9 IP

I datagrammi IP sono composti da diverse sezioni:



Durante il trasporto dei datagrammi IP è possibile che sia richiesta la loro frammentazione, per poi ricomporli lato ricevente.

8.9.1 Teardrop attack

In questo attacco, l'attaccante invia una serie di pacchetti che **vengono frammentati, ma non possono essere ricostruiti:** si inviano frame che dicono di avere posizioni di partenza e lunghezze in byte che si sovrappongono tra loro; come ad esempio:

- *posizione 0 e lunghezza 60 btye*
- *posizione 30 e lunghezza 90 btye*
- *posizione 40 e lunghezza 120 btye*

In questo caso viene sollevato un errore che manda in crash il sistema opeativo del server (DoS).
La **contromisura** consiste nello scartare tutti i frame che si riferiscono a pacchetti IP che risultano malformati.

8.9.2 Ping of Death

Quando si usa il comando *ping* di ICMP, bisogna rispettare una dimensione massima di 2^16, ovvero la dimensione massima di un pacchetto.
L'attaccante può mandare un insieme di frame riferiti ad un pacchetto IP, la cui ricomposizione supera il limite imposto; in questo caso si verificherà un buffer overflow che manderà in crash il sistema operativo.
Come nel caso precedente, la **contromisura** consiste nel controllare i frame e scartare quelli non validi.

8.10 DNS amplification DDoS

L'attacco consiste nell'**inviare una query DNS con IP mittente spoofato**, in modo che il DNS resolver risponda alla vittima. Il DoS è dato dal fatto che è possibile ricavare una query DNS che generi una risposta fino a 50 volte maggiore

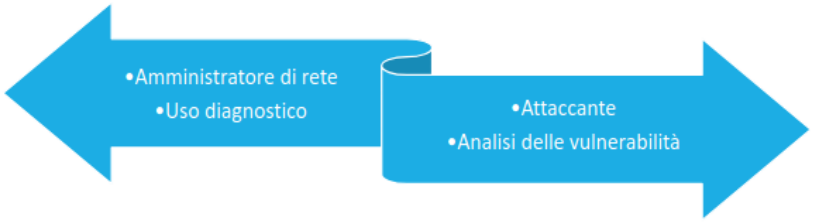
$\rightarrow$ la vittima non è in grado di gestire il traffico della risposta (DoS)

Come in tutti gli attacchi di *amplificazione*, la banda richiesta dall'attaccante è molto inferiore a quella con cui viene travolta la vittima.

Capitolo 9

Network e Port Scanning

Dal punto di vista dell'**amministratore** di rete, la scansione è un'**attività preliminare fondamentale** per conoscere quali dispositivi siano collegati e configurazione abbiano; l'amministratore può conoscere come sia composto il sistema informatico da difendere.
Dal punto di vista dell'**attaccante**, accedere a queste informazioni permette di sapere a quali **vulnerabilità** sia esposta la rete (sistemi operativi in uso, sw non aggiornato, porte aperte sfruttabili, . . .).



Le **attività principali** portate avanti dall'attaccante sono:

- la **mappatura della rete** (network scanning, vedo quali macchine sono attive sulla rete)
- riconoscere i servizi UDP e TCP **disponibili** (port scanning)
- riconoscere i *sistemi di filtraggio* usati per l'utente
- determinare i *sistemi operativi* usati

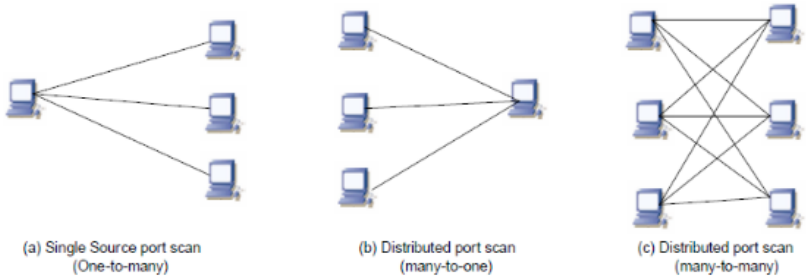
Sia *network scanning* che *port scanning* sono rilevabili e sono **illegali** (salvo autorizzazione del proprietario della rete/macchina).
Generalmente, salvo alcuni servizi che devono essere esposti verso l'esterno (come HTTP/HTTPS nel caso di un server web), il firewall blocca questo tipo di richieste.

9.1 Tipologie di scansione

Le scansioni vengono classificate in base al modo in cui sono compiute, al loro scopo e al numero di host coinvolti.

9.1.1 Numero di host coinvolti

- un singolo host interroga diverse macchine
- tanti host fanno scanning di una sola macchina (si cerca di lasciare meno tracce)
- tante macchine fanno scanning di tanti host



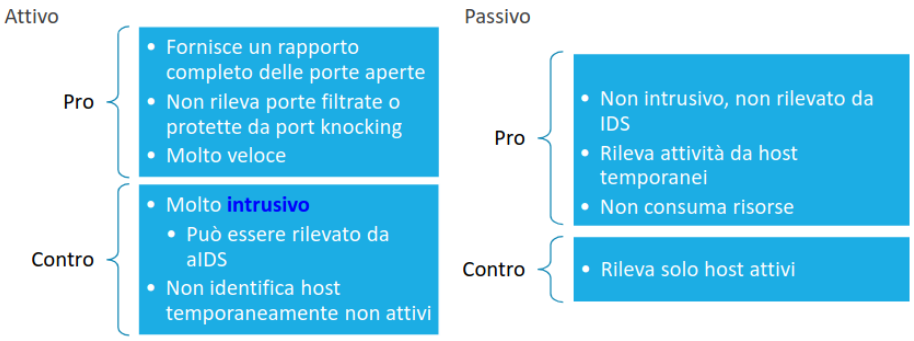
9.1.2 Natura della scansione

È possibile parlare di scanning:

- **verticale:** più porte di una singola macchina vengono scansionate
- **Orizzontale:** la stessa porta di più macchine viene controllata
- **Ibrido:** combinazione delle precedenti

Per quanto riguarda la **natura della scansione**, si può classificare in:

- **Scanning attivo:** si mandano pacchetti alla macchina da scansionare, che possono essere generici o relativi a un particolare protocollo o sw
- **Scanning passivo:** ci si limita a fare sniffing sulla rete e dedurre come sia configurata la macchina



9.1.3 Scopo

- Si parla di ***wide range scanning*** quando si analizza un dato insieme di indirizzi
- Si parla di ***target-specific scanning*** quando la scansione riguarda un'entità specifica

9.2 Risultati (port scanning)

I risultati di uno scan su una porta possono portare alle seguenti considerazioni:

- **Aperta:** il target ha risposto, indicando che un servizio è in ascolto su quella porta (*connection accepted*)
- **Chiusa:** il target ha risposto rifiutando la connessione (*connection denied*)
- **Bloccata/Filtrata:** un sistema di sicurezza perimetrale (come un firewall) ha bloccato l'accesso alla porta, impedendo di individuare uno stato (*connection dropped/filtered*)

9.3 Attacchi tramite protocolli noti

- **ARP scan:** tramite il protocollo ARP è possibile scoprire tutti i dispostivi attivi su una rete locale; l'attacco consiste nel mandare una serie di richieste ARP in broadcast al fine di collezionare tutti gli IP
- **ICMP:** vengono mandati una serie di pacchetti ICMP *echo request* (ping) per poi rimanere in attesa della risposta (ICMP *echo reply*); spesso questo approccio non è percorribile in quanto i ping vengono spesso bloccati dagli amministratori di rete.

  Per aggirare questa contromisura, è possibile usare ICMP *timestamp*

- **TCP:** sono possibili due approcci che sfruttano questo protocollo:
  - **TCP SYN ping:** si manda un TCP SYN; nel caso in cui si riceve risposta SYN-ACK, il servizio è attivo su quella porta
  - **TCP ACK ping:** si manda un pacchetto ACK; nel caso in cui si riceve risposta RST, il servizio è attivo su quella porta
- **UDP ping:** si manda un pacchetto UDP vuoto; se l'host risponde, sappiamo che è attivo su quella porta
- **IP ping:** si mandano pacchetti con protocolli non abilitati (nell'header IP è prevista l'indicazione del protocollo) e si verifica se l'host risponde (analogo a UDP ping)

9.3.1 TCP connect scan

È un attacco che fa uso della *syscall* `connect()` per connettersi ad una socket esposta sulla rete in stato di listen.

Open scan

Le risposte ad un SYN possono essere:

- SYN-ACK: porta aperta
- RST: porta chiusa
- *nessuna rispsota* : porta filtrata da firewall
- ICMP unreachable: porta filtrata da un firewall

Half-open scan

Nel caso in cui si riceva un SYN-ACK, si restituisce un RST per **terminare la connessione** TCP rimasta aperta.
Questo approccio è meno intruviso e quindi meno rilevabile.

9.3.2 Stealth scan

Gli attacchi di *stealth scan* possono essere fatti in vari modi:

- **SYN-ACK scan:** le richieste SYN sono spesso individuate e filtrate; si può invece mandare un pacchetto SYN-ACK e dedurre informazioni dalla risposta
  - se la risposta è RST, la porta è chiusa
  - se non si riceve risposta, la porta è aperta (si possono avere falsi positivi, ad esempio per *packet loss*)

- **ACK scan:** prevede di mandare un pacchetto ACK direttamente (non si mira a vedere se una porta è aperta):
  - se la risposta è RST, la porta *non è filtrata*
  - se non si riceve risposta o *ICMP unreachable*, la porta *è filtrata*

  → questo attacco mira ad **individuare la presenza di un firewall**

- **Window scan:** è simile ad un ACK scan, si basa siò *window size* presente nel pacchetto TCP
  - RST e $window \neq 0 \rightarrow$ porta aperta
  - RST e $window = 0 \rightarrow$ porta chiusa
  - nessuna riposta o *ICMP unreachable* $\rightarrow$ porta filtrata

- **FIN, XMAS, NULL scan:** in questo attacco si punta ad impostare a 0/1 alcuni flag dell'header TCP; nello specifico:
  - FIN
  - URG
  - PSH

  Gli attacchi operano in questo modo:
  - **FIN:** mette a 1 solo il flag FIN
  - **NULL:** mette a 0 tutti i flag
  - **XMAS:** *spegne* e *accende* in maniera casuale tutti i flag (come le luci di natale)

  In base alla rispota, si può capire se la porta è aperta, chiusa oppure filtrata. Questo attacco permette di scavalcare alcuni firewall ceh filtrano solo SYN e ACK.

- **TCP fragmentation scan:** invia dei frammenti di pacchetti; aumenta la difficoltà di detection ma è lento, non affidabile e potrebbe mandare in crush il firewall (si viene individuati)

- **Idle scan:** non si inviano i pacchetti direttamente alla vittima, ma si utilizza un intermediario; in questo modo si riesce a non apparire nella connessione con il server

- **FTP bounce scan:** simile all'*idle scan* (server FTP agisce da zombie), usa il protocollo FTP (File Transfer Protocol); se il server FTP è in modalità attiva, con il comando PORT si può indicare su quale IP e porta ricevere la risposta (si usano quelli della vittima).

  Se il server non riesce a collegarsi, darà un errore sulla connessione FTP all'attaccante.

  Ha il vantaggio di essere *stealth*, ma funziona solo con TCP, è lento e lascia tracce sul server FTP

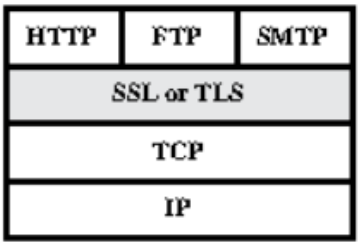
9.4 OS fingerprint

È un attacco in cui l'obiettivo è **identificare il sistema operativo** usato dalla vittima. Si guardano le eventuali risposte e da esse si cerca di capire qual è il sistema operativo (ci sono delle picolle differenze).
Ci si può difendere usando dei firewall o cambiando la firma del TCP/IP stack, usando quella di un altro sistema operativo per ingannare l'attaccante.

Capitolo 10

SSL, TLS e Certificati

SSL (Secure Sockets Layer) e TLS (Transport Layer Security) sono lo stesso tipo di protocolli con algoritmi crittografici diversi; assicurano che la comunicazione tra due host avvenga in un **canale sicuro**, facendo rispettare **integrità** e **autenticità**.



(b) Transport level

Vengono impiegati in ogni browser web, sistemi di pagamento, ...

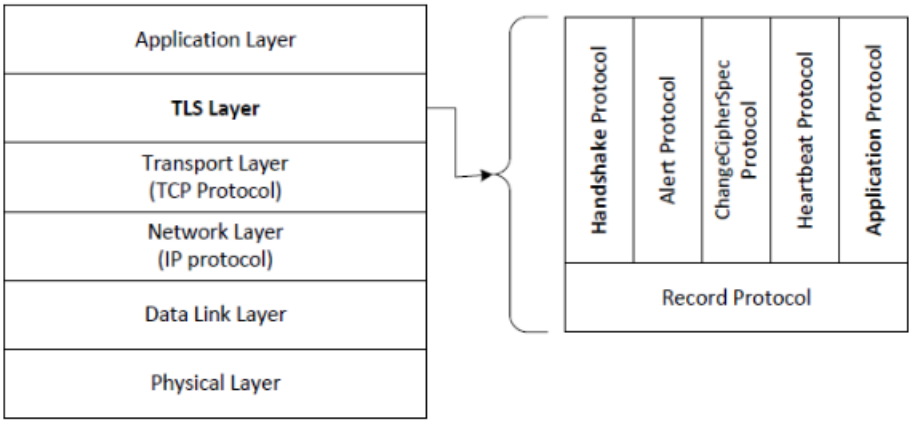
10.1 SSL *basics*

SSL consiste in diversi protocolli:

- **Handshake protocol:** usa chiave pubbliche per stabilire le chiavi segrete condivise tra client e server
- **Record protocol:** fornisce dei servizi di sicurezza ai protocolli di livello più alto; usa le chiavi segrete per proteggere la confidenzialità, integrità e autenticità dei dati
- si *negozia la versione del protocollo e gli algoritmi crittografici da usare*
- *autentica server e client* (opzionale); vengono usati dei certificati digitali per verificare le identità (spesso solo il server è autenticato)

10.2 TLS *basics*

TLS si pone tra il livello applicativo e quello di trasporto: i dati non protetti vengono forniti a TLS, che li trasforma in dati criptati per poi fornirli al livello sottostante (trasporto).



In TLS ci sono *due entità* principali:

- **Connessione:** un trasporto che fornisce un tipo di servizio adeguate; in TLS, tali connessioni sono relazioni *peer-to-peer* (i nodi sono equivalenti)

  Ogni connessione è transitoria ed è associata ad una sessione

- **Sessione:** è un'associazione tra client e server creata dal protocollo di handshake; ogni sessione definise dei parametri crittografici di sicurezza che possono essere condivisi tra più connessioni

  → vengono utilizzate per evitare la costosa negoziazione di nuovi parametri di sicurezza per ogni connessione

Tra qualsiasi coppia di parti (ad esempio applicazioni come HTTP su client e server) potrebbero esserci più connessioni protette (in teoria anche più sessione, ma non accade nella pratica).

10.2.1 Componenti di TLS

Alert Protocol

Notifica situazioni anomale o segnala eventuali problemi; viene usato per trasmettere allarmi relativi al protocollo SSL/TLS all'altra entità in comunicazione. Ogni messaggio è compsoto da **2 byte**:

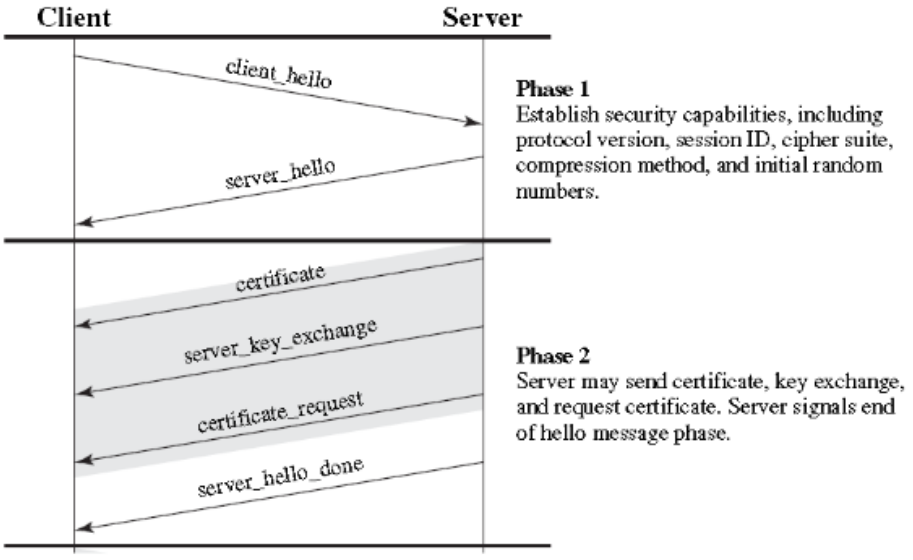
- il *primo byte* assume il valore di **warning** o **fatal** che indica la gravità del messaggio identificato dal secondo byte; i messaggi *fatal* sono irreversibili e la sessione viene interrota

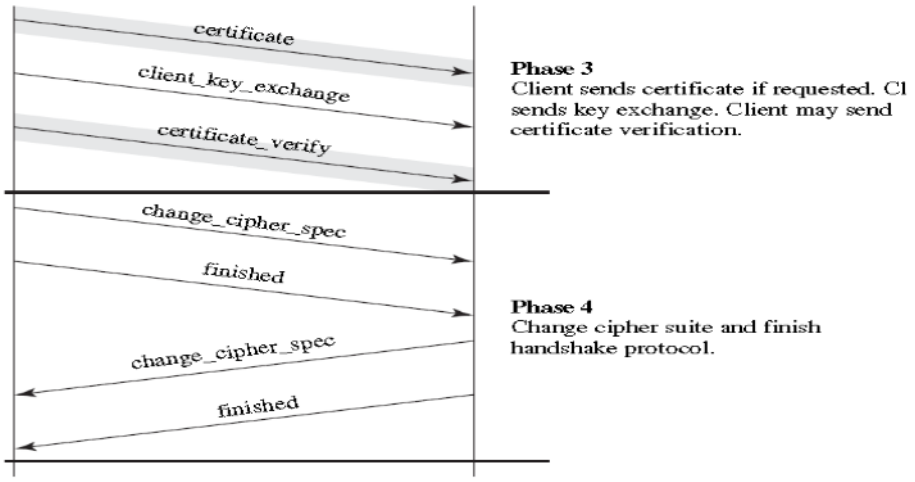
- il *secondo byte* contiene un codice che indica l'**avviso specifico**

Handshake protocol

Permette alle parti di negoziare i diversi algoritmi necessari per la sicurezza della comunicazione, consente l'eventuale autenticazione e lo scambio delle chiavi.
È fatto da una sequenza di messaggi, ciascuno dei quali ha i seguenti campi:

- **Tipo** (1 byte): indica uno dei 10 tipi di messaggi disponibili
  - `hello_request`
  - `client_hello`
  - `server_hello`
  - `certificate_request`
  - `certificate`
  - `certificate_verify`
  - `server_key_exchange`
  - `clinet_key_exchange`
  - `server_done`
  - `finished`
- **Lunghezza** (3 byte)
- **Contenuto** (byte): i parametri associati al messaggio





La **generazione** e **scambio delle chiavi** viene fatta in tre passaggi:

- *Segreto pre-master*: viene crittografato un numero casuale con la chiave pubblica del server
- *Segreto master*: viene generato usando i *nonce* scambiati durante l'handshake e il segreto pre-master
- *Chiave di sessione*: viene usato il segreto principale per generare una sequenza di byte (le chiavi, dipendono dall'algortimo di cifratura)

Change chiper spec protocol

Impone l'esecuzione di un nuovo handshake per rinegoziare i parametri di sicurezza e ripetere l'autenticazione.
È costituito da un singolo byte impostato al valore 1.

Record protocol

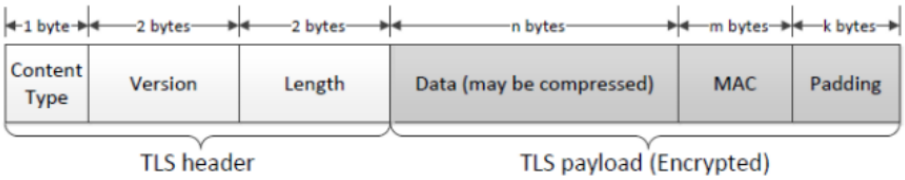
Fornisce *due servizi* per le connessioni TLS:

- **confidenzialità:** usa la chiave segreta condivisa durante l'handshake
- **integrità dei messaggi:** la stessa chiave segreta viene usata per il MAC (Message Authentication Code)

I passagi che segue il protocollo sono:

1. prende in input un messaggio da trasmettere
2. frammenta l'input in blocchi
3. applica il MAC al frammento (lo concatena in coda)

4. cifra quanto ottenuto
5. aggiunge un'intestazione e trasmette in un segmento TCP; l'header contiene:
   - *content-type* (alert, handshake, ...)
   - *versione*
   - *lunghezza del payload*



10.2.2 Stato di sessioni e connessioni

Lo stato di una sessione è definito da:

- **identificatore**
- **certificato peer**
- **metodo di compressione** (da TLS 1.3 è vietato comprimere)
- **cipher spec:** algoritmo di crittografia usato e una funzione di hash da usare per il MAC
- **master secret**
- `isResumable`**:** una flag che indica se è possibile avviare nuove connessioni dalla sessione

Lo stato di una connessione è definito da:

- **server e client randomness** (sequenza di byte)
- **server write MAC secret**
- **client write MAC secret**
- **chiave di scrittura server**
- **chiave di scrittura client**
- **numeri di sequenza**

10.2.3 Costo di una sessione

Sia lato client che lato server:

- generare valori casuali
- verifica dei certificati
- cifrare/decifrare i valori casuali con la chiave opportuna
- calcolare chiave con hash

→ ogni sessone ha un costo elevato; tanto richieste di connessione potrebbero portare ad un DoS

10.3 SSH (Secure Shell)

SSH è un protocollo di rete crittografico che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica.
SSH è organizzato come tre protocolli che tipicamente vengono eseguiti su TCP:

- **Transport Layer Protocol:** fornisce l'autenticazione del server, la riservatezza e l'integrità dei dati
- **Protocollo di autenticazione utente** per autenticare l'utente sul server
- **Protocollo di connessione**

Sia TLS che SSH creano un tunnel per il trasporto di dati sicuro; hanno però alcune differenze relative alla sua realizzazione, come i formati usati; inoltre, SSH ha dei protocolli per gestire ciò che accade all'interno del tunnel.

10.4 Vulnerabilità

10.4.1 SSL *version rollback*

Il server viene ingannato pensando che sta comunicando con un client che supporta solo la versione SSL 2.0; questo è problematico perché:

- le *cipher spec* non sono autenticate
- i messaggi dell'handshake non sono protetti
- non supporta catene di certificazione
- ...

10.4.2 Man In The Middle TLS *downgrade*

Un utente malintenzionato che ha il controllo del traffico può simulare questo attacco ancora oggi.

Per gestire la connessione tra due dispositivi su una rete locale, bisogna fare in modo di reindirizzare il traffico attraverso l'attaccante, **manipolando la cache di ARP** (Address Resolution Protocol).

10.5 HTTPS

HTTPS è la versione sicura di HTTP; usa un canale sicuro SSL/TLS per inviare dei messaggi HTTP.
Bisogna fare attenzione che un attaccante non si introduca nel passaggio automatico tra HTTP e HTTPS: l'attaccante potrebbe redirizionare il traffico, intercettando i pacchetti della vittima e modificando la richiesta.
La contromisura adottata è che le interazioni devono avvenire interamente in HTTPS; il browser si rifiuta di usare HTTP.

10.6 Certificati

I certificati in ambito web vengono rilasciati da un'ente accreditato a:

- browser
- siti web

→ prima di chiedere la pagina di un sito web, il browser richiede al web server il suo certificato; solamente dopo aver verificato chiede la pagina.
Un possibile attacco è quando un attaccante chiede un certificato legittimo e ne crea una **copia falsa**, da usare ad esempio in un attacco di phishing.
Diventa fondamentale poter **revocare certificati**, ad esempio a causa di:

- chiave privata compromessa
- l'utente ha smesso di pagare per avere una certificazione, o non vuole più averla
- certificato compromesso