SICUREZZA INFORMATICA

NB. TCP è affidabile non sicuro!

Una delle cose da capire subito riguardo alla sicurezza informatica è che un sistema non è mai sicuro al 100%. Il lavoro all'interno di questo settore diventa quindi quello di limitare al massimo le vulnerabilità presenti nel sistema.

Per un neofita come me, va subito specificato che c'è difesa e difesa. Bisogna conoscere quanto più profondamente possibile ciò che si vuole difendere per poter adattare una strategia forte ed efficace. Per esempio, c'è estrema differenza nel difendere l'accesso ad un computer che ad uno specifico suo contenuto.

Un sistema è sicuro quando sono presenti 3 componenti fondamentali:

- confidenzialità:
  - riservatezza dei dati: solo persone o processi intesi possono accedere a determinati dati;
  - privacy: l'utente controllo o influenza quali informazioni possono essere collezionate e memorizzate (tu sei proprietario dei tuoi dati);
- integrità: controllo che i dati non vengano alterati, con possibilità di saperlo se necessario;
- disponibilità: si deve garantire che il sistema sia accessibile quando richiesto;
- autenticità: ciascun utente deve poter verificare l'autenticità delle informazioni e si richiede di poter verificare se una informazione è stata manipolata;
- tracciabilità: le azioni di un'entità devono essere tracciate in modo univoco in modo tale da supportare la non-ripudiabilità e l'isolamento delle responsabilità. Ad esempio, nessun utente deve poter ripudiare o negare in tempi successivi messaggi da lui spediti o firmati.

Come extra aggiungiamo il possesso, che richiede di proteggere le informazioni dal prelievo, dalla copia o dal controllo non autorizzati.

Una minaccia è una possibile violazione della sicurezza (sinonimo di vulnerabilità, quindi una possibilità), mentre la violazione effettiva è chiamata attacco. Gli attacchi possono essere:

- attivi: tentativi di alterare le risorse o modificare il funzionamento dei sistemi;
- Passivi: tentativi di carpire informazioni e utilizzarle senza intaccare le risorse;
- Interni: iniziati da un'entità interna al sistema;

- Esterni: iniziati da un'entità esterna, tipicamente attraverso la rete.

L’attacco ad un sistema lavora invece su diverse basi, in particolare:
- divulgazione: contrario di confidenzialità, si tratta di attaccare la riservatezza di determinati dati ad accesso limitato e/o privati;
- alterazione: si tratta dell’opposto dell’integrità, ossia contraffare o modificare un dato;
- distruzione: rappresenta l’opposto della disponibilità;

Lavorare contro questo tipo di attacchi, è perciò l’equivalente che lavorare sul rendere un sistema sicuro per come lo abbiamo definito. Tuttavia, proteggere confidenzialità e integrità può restringere eccessivamente dal punto di vista della disponibilità. Al contrario, eccessiva disponibilità espone il sistema al rischio complementare delle altre due componenti. La chiave è come sempre l’equilibrio tra queste 3 parti.

Un meccanismo di sicurezza è un metodo (strumento/ procedura) per garantire una politica di sicurezza. Data una politica che distingue le azioni “sicure” da quelle “non sicure”, i meccanismi di sicurezza devono prevenire, scoprire o recuperare da un attacco.

Nella prevenzione il meccanismo deve rendere impossibile l’attacco. Spesso sono soluzioni pesanti ed interferiscono con il sistema al punto da renderlo scomodo da usare; un esempio è la richiesta di password come modo di autenticazione.

Nella scoperta, il meccanismo è in grado di scoprire che un attacco è in corso. E’ utile quando non è possibile prevenire l’attacco, ma può servire anche a valutare le misure preventive. Si usa solitamente un monitoraggio delle risorse del sistema, cercando eventuali tracce di attacchi.

Nel recupero da un attacco, si può fare in due modi:
- fermare l’attacco e recuperare/ricostruire la situazione pre-attacco, ad esempio attraverso copie di backup;
- continuare a far funzionare il sistema correttamente durante l’attacco (fault-tolerant).

Nonostante anni di ricerca, è difficile progettare sistemi che prevengono completamente le falle nella sicurezza. Tuttavia, insiemi di pratiche e regole sono state codificate, analogamente a quanto succede per l'ingegneria del software: aspetti economici dei meccanismi, fail-safe default (comportamenti non specificati devono prevedere un default sicuro), progettazione aperta, tracciabilità delle operazioni (qualsiasi operazione può essere ricostruita e il sistema ripristinato), separazione dei privilegi (dei

diversi utenti con diverse permessi in particolare, per prevenire la privilege escalation), separazione delle funzionalità (un sistema compromesso non dovrebbe compromettere gli altri), isolamento dei sottosistemi, modularità, ecc ecc.

Nell'aggiunta di un meccanismo specifico vengono quindi valutati i seguenti aspetti:

- analisi costi-benefici della sicurezza;
- analisi dei rischi (valutare le probabilità di subire attacchi e i danni che possono causare);
- aspetti legali (ad esempio uso della crittografia negli USA) e morali;
- problemi organizzativi (ad esempio la sicurezza non "produce" nuova ricchezza, riduce solo le perdite);
- aspetti comportamentali delle persone coinvolte

Meccanismi specifici – Legati ad uno specifico livello OSI

- Crittografia
  - Trasformazione dei dati in un formato non intellegibile
- Firma digitale e Integrità dei dati
  - Usata per provare la sorgente e l'integrità di dati o messaggi
- Autenticazione e Controllo degli accessi
  - Gestione dei diritti degli utenti rispetto le risorse

Meccanismi generali

- Rilevamento degli eventi
- Gestione degli Audit
- Recovery

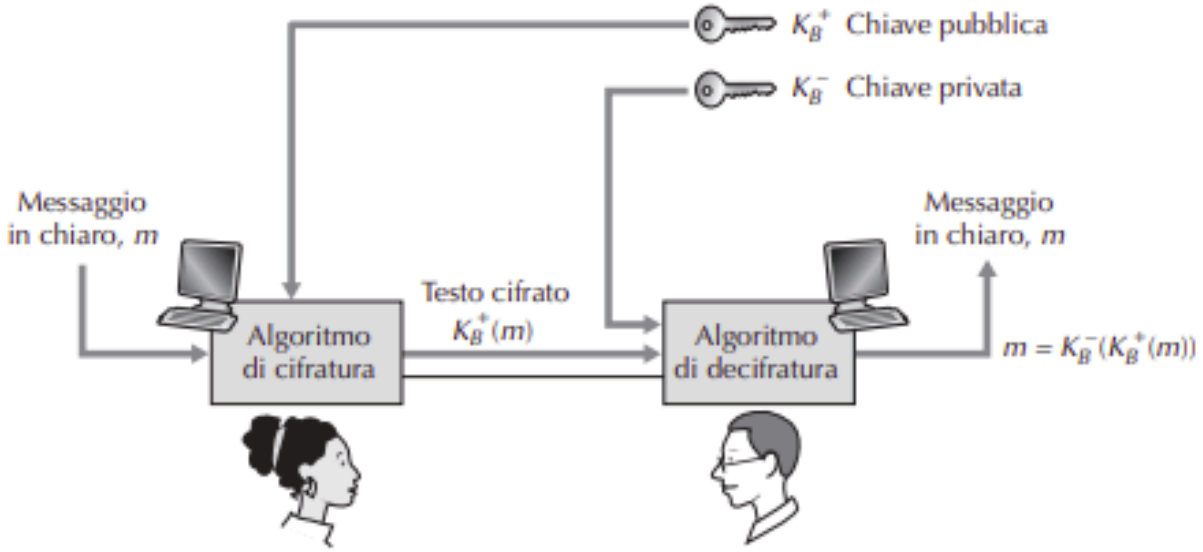
CRITTOGRAFIA

La crittografia è la scienza che si occupa di proteggere l'informazione rendendola sicura, in modo che un utente non autorizzato che ne entri in possesso non sia in grado di comprenderla. La crittoanalisi è invece la scienza che cerca di aggirare o superare le protezioni crittografiche, accedendo alle informazioni protette. L'insieme di crittografia e crittoanalisi è detto crittologia.

Definiamo come algoritmo crittografico una funzione che prende in ingresso un messaggio e un parametro detto chiave, e produce in uscita un messaggio trasformato. Nello specifico, la cifratura trasforma un testo in chiaro (plaintext o cleartext) in testo cifrato (ciphertext), mentre la decifratura inverte il processo, da testo cifrato a testo in chiaro.

Se le chiavi di cifratura e decifratura sono uguali, l'algoritmo è detto simmetrico, e per tanto la chiave deve essere segreta. Se invece le chiavi sono diverse, l'algoritmo è detto asimmetrico, per cui una chiave è pubblica, mentre l'altra privata (segreta).

Per gli algoritmi simmetrici, i primi ad essere nati, esiste quindi il problema della distribuzione delle chiavi. Serve un canale di comunicazione sicuro per trasmettere la chiave da un utilizzatore all'altro (altrimenti chi deve decifrare non può farlo). Nel 1976 vengono quindi inventate le chiavi asimmetriche, in cui il dato viene cifrato con la chiave pubblica del destinatario, che potrà decifrarlo con la propria chiave privata.



Non è più necessario incontrarsi per scambiare chiavi. La stessa chiave (pubblica) può essere usata da più utenti. Tuttavia è richiesto che la generazione di una coppia di chiavi pubblica/ privata sia semplice, così come deve essere semplice l'operazione di cifratura e

decifratura se si è a conoscenza della relativa chiave; deve inoltre essere computazionalmente impraticabile ricavare la chiave privata da quella pubblica, e computazionalmente impraticabile ricavare il testo in chiaro avendo il testo cifrato e la chiave pubblica.

Non deve essere possibile (facilmente) ottenere il corrispondente testo in chiaro dato un testo cifrato senza conoscere la chiave di decifratura, oppure, dato un testo cifrato e il corrispondente testo in chiaro, ottenere la chiave di decifratura. In generale, nessun algoritmo crittografico è assolutamente sicuro, quindi si dice che è computazionalmente sicuro in almeno uno dei seguenti casi:

1. il costo necessario a violarlo è superiore al valore dell'informazione cifrata;
2. il tempo necessario a violarlo è superiore al tempo di vita utile dell'informazione cifrata.

La crittoanalisi tenta di ricostruire il testo in chiaro senza conoscere la chiave di decifratura. L'attacco più banale in questo senso è quello definito "a forza bruta", ossia tentare di decifrare il messaggio provando tutte le chiavi possibili. E' applicabile a qualunque algoritmo, ma la sua praticabilità dipende dal numero di chiavi possibili. È comunque necessario avere informazioni sul formato del testo in chiaro, per riconoscerlo quando si trova la chiave giusta.

Principio di Kerckhoffs: nel valutare la sicurezza di un algoritmo crittografico si assume che il crittoanalista conosca tutti i dettagli dell'algoritmo. La segretezza deve risiedere nella chiave, non nell'algoritmo (peso questa roba, da tatuaggio).

La crittografia simmetrica, altrimenti detta crittografia a chiave segreta, utilizza come detto una chiave comune e il medesimo algoritmo crittografico per la codifica e la decodifica dei messaggi. Due utenti che desiderano comunicare devono quindi accordarsi su di un algoritmo e su di una chiave comune, opportunamente scambiata su un canale sicuro.

Uno dei cifrari più classici e conosciuti è il cifrario di Cesare che sostituisce ogni lettera del testo in chiaro con quella che si trova K posizioni più avanti nell'alfabeto; K è quindi la chiave. Le chiavi possibili sono solamente 20. Ecco un esempio:

Esempio: K = 3

- In chiaro: A B C D E F G H I L M N O P Q R S T U V Z
- Cifrate: D E F G H I L M N O P Q R S T U V Z A B C
- Esempio di messaggio in chiaro / cifrato: CIAO / FNDR

Un costrutto di crittografia più avanzato è per esempio la cifratura monoalfabetica, che consiste nel sostituire ogni lettera con un'altra dell'alfabeto (permutazione), secondo un certo alfabeto che costituisce la chiave. Le chiavi possibili con questo algoritmo sono pari al numero di permutazioni possibili, quindi 21!, ovvero circa 5,1 x 10^19.

Esempio:

- In chiaro: A B C D E F G H I L M N O P Q R S T U V Z
- Cifrate: M Z N C B V L A H S G D F Q P E O R I T U
- Esempio di messaggio in chiaro / cifrato: CIAO / NHMF

Analisi delle frequenze:

L'analisi delle frequenze nella decifratura si basa sull'idea che, in ogni lingua, certe lettere (o combinazioni di lettere) compaiono più spesso di altre (per esempio la lettera 'e' in italiano è molto comune). Quando hai un testo cifrato con un cifrario a sostituzione semplice (dove ogni lettera è sempre sostituita dalla stessa altra lettera), puoi:

1. calcolare le frequenze delle lettere nel testo cifrato;
2. confrontarle con le frequenze medie delle lettere nella lingua originale (es. italiano);
3. dedurre le sostituzioni: Se la lettera più frequente nel cifrato è 'X', e la lettera più frequente in italiano è 'E', è molto probabile che 'X' stia per 'E'. Si procede così per le altre lettere, affinando le ipotesi;

In pratica, si cercano i pattern statistici del linguaggio per "rompere" il codice. Funziona benissimo con i cifrari monoalfabetici, ma diventa molto più difficile con quelli polialfabetici o più complessi.

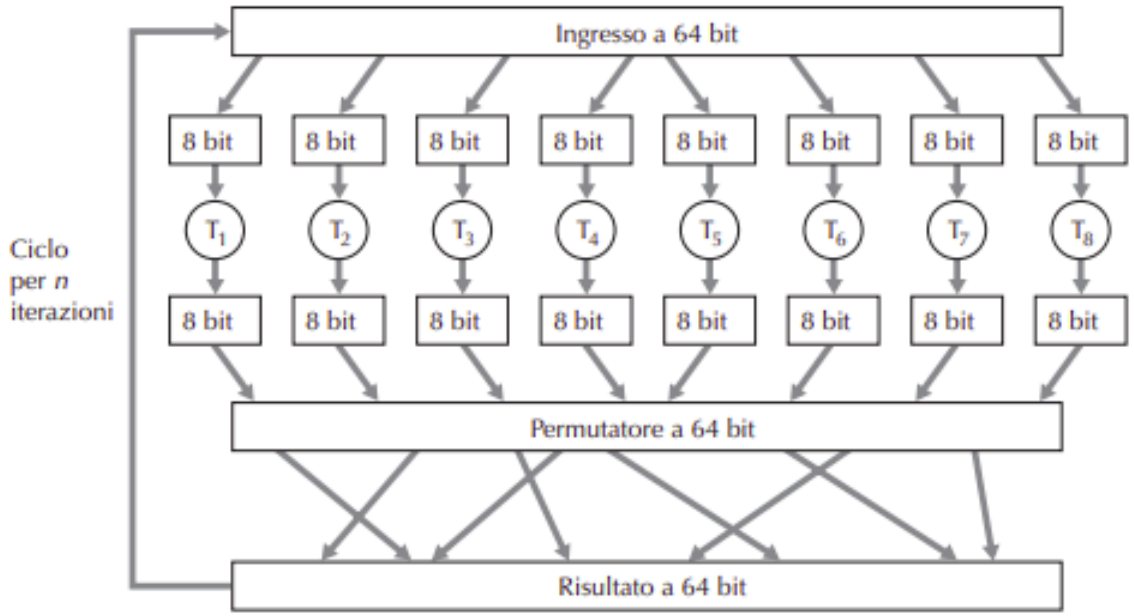
Cifratura a blocchi:

La cifratura a blocchi è un tipo di algoritmo crittografico simmetrico (quindi stessa chiave per cifratura che per decifratura). Il concetto fondamentale di un cifrario a blocchi è che opera su blocchi di dati di dimensione fissa. Non cifra il testo un carattere alla volta, ma prende un "pezzo" (un blocco, appunto) di testo in chiaro di una dimensione predefinita e lo trasforma in un blocco di testo cifrato della stessa dimensione. Come Funziona (in modo molto semplificato):

1. suddivisione in Blocchi: il testo in chiaro originale (che potrebbe essere un messaggio, un file, qualsiasi dato) viene diviso in blocchi di dimensioni uguali. Ad esempio, se il cifrario opera su blocchi di 128 bit, il testo in chiaro viene suddiviso in segmenti da 128 bit;

2. cifratura blocco per blocco: ogni singolo blocco di testo in chiaro viene processato dall'algoritmo usando la chiave segreta. L'algoritmo esegue una serie complessa di trasformazioni matematiche e logiche (come sostituzioni, permutazioni, operazioni XOR) sul blocco;
3. output cifrato: il risultato di queste operazioni è un blocco di testo cifrato della stessa dimensione originale. Questo processo viene ripetuto per tutti i blocchi;
4. decifratura: per decifrare, il processo viene invertito. Ogni blocco di testo cifrato viene passato all'algoritmo insieme alla stessa chiave segreta, e l'algoritmo esegue le operazioni inverse per riottenere il blocco di testo in chiaro originale.

Le permutazioni possono essere combinate tra loro per creare schemi più complessi.



Di algoritmi che implementano questo metodo ci ricordiamo del DES (Data Encryption Standard), il più noto algoritmo crittografico simmetrico moderno, nato negli anni ‘70 a seguito di un progetto di IBM, e adottato ufficialmente nel ‘77 come standard dal governo americano. Esso utilizza chiavi di 56 bit, infatti è da considerarsi ormai obsoleto. Un esempio invece molto più recente e estremamente valido è invece l'AES (Advanced Encryption Standard), lo standard attuale usato quasi ovunque, dalla protezione dei dati sul tuo computer alle comunicazioni online. L'AES lavora su blocchi da 128 bit e può usare chiavi di diverse lunghezze, rendendolo estremamente robusto.

Per gli algoritmi di cifratura a chiave asimmetrica invece, l’esempio più importante è l’RSA, che si basa sulla difficoltà di scomporre un numero in fattori primi. La chiave in RSA ha di solito dimensioni di almeno 210 bit (ossia oltre 300 cifre decimali). Un attacco a forza bruta

contro RSA non consiste nel provare tutte le chiavi possibili, ma nel fattorizzare il prodotto di due numeri primi. Per capire come avviene la cifratura e la decifratura con RSA ci si deve avvalere della matematica a modulo.

Scelti due primi p,q si calcola

- $n = p*q$
- $z = (p-1)*(q-1)$
- un numero $1 < e < n$ relativamente primo a z
- un numero d tale che $(e*d-1)$ sia multiplo di z

Chiave pubblica → (n,e) Per cifrare m → $c = m^e \bmod n$

Chiave privata → (n,d) Per decifrare c → $m = c^d \bmod n$

Algoritmo RSA: esempio

- ❑ p = 5, q = 7
- ❑ Segue:
  - n = p*q = 35
  - z = (p-1)*(q-1) = 24
  - e = 5 (relativamente primo a 24; andavano bene anche 7, 9, 11, …)
  - d = 29 (infatti 5*29 - 1 = 144 → multiplo di 24)

- ❑ Messaggio da inviare: la parola "love"
  - Supponiamo di rappresentare le lettere con numeri da 1 a 26 (incluse le lettere x,y,w, …)

Lettere in chiaro	m: rappresentazione numerica	m^e	Testo cifrato $c = m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Testo cifrato c	c^d	$m = c^d \bmod n$	Lettere in chiaro
17	481968572106750915091411825223071697	12	l
15	12783403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	10000000000000000000000000000000	5	e

Come possiamo notare gli svantaggi principali delle chiavi asimmetriche sono che richiedono molte risorse computazionali. Parliamo di 100-1000 volte più lenti degli algoritmi simmetrici. Vengono infatti utilizzati per scambiarsi una chiave di sessione, che verrà poi usata con un algoritmo simmetrico sicuro e computazionalmente più efficiente. Con RSA ciò che viene cifrato con la chiave pubblica si può decifrare con la chiave privata, ma vale anche l'inverso: ciò che è cifrato con la chiave privata si può decifrare con la chiave pubblica.

N.B: nelle procedure di cifratura NON interessa minimamente che le parti coinvolte siano autenticate, ossia non ci si pone il problema se chi sta ricevendo il mio 'messaggio' sia effettivamente chi volevo lo ricevesse. Di questo se ne occupa l'...

INTEGRITA'

La firma digitale è la controparte elettronica e crittografica di una firma autografa tradizionale. Il suo scopo principale non è tanto garantire la segretezza di un documento (quella è la cifratura), quanto assicurare l'autenticità (chi ha firmato) e l'integrità (il documento non è stato alterato dopo la firma) del messaggio o del documento digitale. Una caratteristica fondamentale della firma digitale è la sua non ripudiabilità: una volta che un'entità ha firmato digitalmente un documento, non può in seguito negare di averlo fatto. Questo la rende legalmente valida in molti contesti.

La firma digitale sfrutta algoritmi a chiave asimmetrica, come RSA, ma in un modo che potremmo definire "inverso" rispetto alla cifratura standard:

- algoritmo di firma: la persona che vuole firmare (il mittente) usa la propria chiave privata per "firmare" il documento. Nel contesto di RSA, questo significa applicare l'operazione di decifratura (che normalmente si farebbe con la chiave privata per decifrare un messaggio destinato a sé) al messaggio o, più comunemente, al suo hash;
- algoritmo di verifica: chiunque voglia verificare la firma (il destinatario) usa la chiave pubblica del mittente. Applica l'operazione di cifratura (che normalmente si farebbe con la chiave pubblica per cifrare un messaggio) alla firma ricevuta. Se il risultato corrisponde al messaggio originale (o al suo hash), la firma è valida.

In pratica, la chiave privata garantisce l'identità del firmatario (solo lui la possiede e può "decifrare" in questo modo), mentre la chiave pubblica permette a chiunque di verificare quella "decifratura".

Il problema dell'onerosità:

Firmare un intero documento digitale significa applicare l'algoritmo di firma (l'operazione di "decifratura" RSA) all'intero contenuto del doc. Questo è un processo oneroso e lento dal punto di vista computazionale, specialmente per doc di grandi dimensioni. La soluzione è combinare RSA con funzioni hash: per superare questo limite, si adotta un approccio più efficiente e standardizzato:

- il doc originale non viene firmato direttamente. Invece, si calcola il suo hash crittografico. Una funzione hash genera una "impronta digitale" univoca e di lunghezza fissa del doc, molto più corta del doc stesso. Anche una minima modifica al doc originale produce un hash completamente diverso.
- questo hash viene poi firmato digitalmente utilizzando la chiave privata del mittente. L'operazione RSA avviene quindi su una stringa molto più corta, rendendola computazionalmente efficiente.
- per la verifica, il destinatario riceve il doc, la firma digitale e la chiave pubblica del mittente. Calcola da solo l'hash del doc ricevuto e, contemporaneamente, "decifra" la firma digitale utilizzando la chiave pubblica del mittente. Se i due hash (quello calcolato e quello estratto) corrispondono, la firma è valida e il documento è integro.

Quindi l'hash ha le seguenti proprietà:

- mappa stringhe di lunghezza arbitraria in stringhe di lunghezza fissa: questa è la proprietà che abbiamo appena descritto. Indipendentemente da quanto è lungo il tuo messaggio (anche un solo bit o terabyte di dati), l'hash risultante avrà sempre la stessa dimensione;
- la funzione deve essere "one-way" (unidirezionale): questo significa che è facile calcolare l'hash a partire da un dato input, ma è impossibile fare il percorso inverso, cioè risalire all'input originale (il messaggio) conoscendo solo l'hash. È come schiacciare della frutta per fare un succo: è facile fare il succo, ma impossibile rifare il frutto dal succo. Questa proprietà è cruciale per la sicurezza: se qualcuno avesse l'hash della tua firma, non potrebbe ricostruire il doc originale o la tua chiave privata;
- la funzione deve essere "collision-resistant" (resistente alle collisioni): questa è forse la proprietà più importante per la sicurezza. Significa che è estremamente difficile (o computazionalmente impraticabile) trovare due input diversi che producano lo stesso identico output (lo stesso hash). Se fosse facile trovare una "collisione", un attaccante potrebbe creare un documento falso con lo stesso hash di un documento originale firmato, ingannando il sistema di verifica. Un buon algoritmo di hash rende questo evento così improbabile da essere considerato impossibile per scopi pratici.

Algoritmi di Hash Comuni:

- MD5 (Message Digest 5): produce un hash di 128 bit. È stato molto popolare in passato, ma oggi è considerato obsoleto e insicuro. Sono state scoperte delle tecniche per creare collisioni, il che lo rende inadatto per la firma digitale o qualsiasi applicazione che richieda robusta sicurezza. Lo si trova ancora per verifiche di integrità non critiche (es. verificare il download di un file);
- SHA-1 (Secure Hash Algorithm 1): produce un hash di 160 bit. Anche SHA-1 è stato ampiamente utilizzato, ma, come MD5, è stato deprecato e considerato obsoleto per la maggior parte degli usi crittografici. Le tecniche di attacco alle collisioni sono diventate sufficientemente pratiche da renderlo insicuro per la firma digitale e i certificati SSL/TLS;
- SHA-2 (Secure Hash Algorithm 2): questa è una famiglia di algoritmi che include diverse varianti a seconda della lunghezza dell'hash prodotto, le più comuni sono SHA-256, SHA-384 e SHA-512. Sono gli algoritmi più utilizzati e considerati sicuri per la maggior parte delle applicazioni crittografiche attuali, inclusa la firma digitale;
- SHA-3 (Secure Hash Algorithm 3): è la famiglia di algoritmi più recente, selezionata tramite un'altra competizione del NIST (come per AES). Offre lunghezze di hash simili a SHA-2 (256, 384, 512 bit) ed è stata sviluppata con un'architettura interna diversa per fornire un'alternativa in caso di future vulnerabilità scoperte in SHA-2, o per contesti specifici. È anch'esso considerato sicuro e moderno.

AUTENTICAZIONE

L'autenticazione risponde a una domanda fondamentale: "Chi sei?" È il processo con cui un sistema verifica l'identità di un'entità, sia essa una persona (un utente), un dispositivo, o un'altra applicazione. Pensiamo ai sistemi tipici client-server, come quando cerchi di accedere a un sito web o a un'applicazione. Tu sei il "client" e il sito/applicazione è il "server". Per poterti dare accesso alle tue informazioni o funzionalità, il server deve prima sapere con certezza chi sei tu. È importante distinguere l'Autenticazione dall'Autorizzazione:

- Autenticazione: è il processo di verifica dell'identità. Il sistema risponde alla domanda "Sei davvero tu, Tizio Caio?".
- Autorizzazione: Una volta che la tua identità è stata verificata (autenticazione completata), l'autorizzazione stabilisce cosa puoi fare all'interno del sistema.

L'autenticazione è, in sostanza, un processo di verifica di un'affermazione. Quando provi ad accedere (come in un login), tu affermi la tua identità. Il sistema, attraverso il processo di autenticazione, verifica se questa affermazione è vera. Per capire meglio i meccanismi di

autenticazione, possiamo immaginare che il sistema si ponga tre domande fondamentali su di te:

1. Chi sei? (La tua identità dichiarata: username, ID, ecc.);
2. Come lo sai? (La prova che tu sei davvero quella persona: password, PIN, ecc.);
3. Chi te lo ha detto? (La fonte di fiducia che garantisce la tua prova: un certificato, un token, ecc.).

Quest'ultima è una semplificazione, ma ci porta ai tre tipi fondamentali di fattori di autenticazione che sono universalmente riconosciuti e su cui si basano tutti i sistemi moderni:

- Qualcosa che sai: questa è la forma più comune. Esempi includono password, PIN, risposte a domande di sicurezza. È una conoscenza che solo l'utente e il sistema dovrebbero condividere. E' vulnerabile ad attacchi a forza bruta e attacchi a dizionario;
- Qualcosa che hai: si riferisce a un oggetto fisico in tuo possesso. Esempi sono una smart card, un token USB, il tuo smartphone (per ricevere codici via SMS o app di autenticazione), una chiave fisica;
- Qualcosa che sei: riguarda le tue caratteristiche biometriche uniche. Esempi sono l'impronta digitale, la scansione dell'iride, il riconoscimento facciale o vocale.

Spesso, i sistemi di sicurezza più robusti usano una combinazione di questi fattori (autenticazione a due o più fattori) per aumentare notevolmente la sicurezza.

PLUS: metodo salt, è un valore casuale e unico aggiunto a ogni password prima che venga crittografata (hashing). Serve a rendere gli hash univoci, anche per password uguali, e a proteggere da attacchi come le rainbow table. I valori salt non sono cifrati o hashati, ma contenuti in una tabella apposita libera e pulita in backend. Usando il TIMESTAMP per il valore salt, si può usare anche l'informazione per generare periodicamente la richiesta di rinnovo psw.

Le tipologie di autenticazione viste finora si possono usare senza modifiche per l'autenticazione locale (dentro rete privata, oppure anche solo sbloccare il pc). Se l'autenticazione avviene da remoto (autenticazione diretta) ci sono altri problemi. Per esempio, un intruso potrebbe registrare e replicare le informazioni.

Quando molti sistemi/applicazioni condividono gli stessi utenti, si ricorre spesso a sistemi di autenticazione indiretta. Le informazioni sugli utenti vengono centralizzate sul sistema di autenticazione, e gli altri sistemi si appoggiano su di esso. Due esempi di sistemi di autenticazione indiretta sono:

- RADIUS (Remote Authentication Dial In User Service), nato per l’accesso remoto dial-up (tramite linea telefonica e ISP);
- Kerberos, usato per l’autenticazione e il single sign-on (SSO) tra applicazioni all’interno di un “dominio” amministrativo; ad esempio all’interno di un’azienda, come univr;

CA:

La sua funzione è quella di legare in modo verificabile una chiave pubblica a un'identità specifica (persona, azienda, server), tramite il seguente processo:

1. un'entità chiede alla CA di certificare la propria chiave pubblica;
2. la CA verifica rigorosamente l'identità del richiedente;
3. se la verifica è positiva, la CA emette un Certificato Digitale.

Il Certificato Digitale contiene:

- la chiave pubblica dell'entità;
- l'identità dell'entità;
- la firma digitale della CA stessa (con la propria chiave privata), che è l'attestazione di fiducia.

Lo standard più comune per questi certificati è X.509.

Campi essenziali di un certificato X.509

Nome campo	Descrizione
Versione	Numero di versione della specifica X.509
Numero seriale	Identificatore unico del certificato fornito dalla CA
Firma	Specifica l’algoritmo utilizzato dalla CA per firmare il certificato
Nome dell’emittente	Identificativo della CA che rilascia il certificato, in formato DN [RFC 4514]
Periodo di validità	Inizio e fine del periodo di validità del certificato
Nome del soggetto	Identificativo dell’entità la cui chiave pubblica è associata al certificato (in formato DN)
Chiave pubblica del soggetto	Chiave pubblica del soggetto e indicazioni dell’algoritmo da utilizzare

Un certificato digitale contiene diverse informazioni cruciali:

1. Chiave Pubblica del Soggetto: La chiave pubblica dell'entità a cui il certificato si riferisce (es. la chiave pubblica di un sito web).

2. Identità del Soggetto: Le informazioni che identificano univocamente il titolare della chiave (es. nome del dominio per un sito web, nome dell'organizzazione, indirizzo email).
3. Identità dell'Emittente (CA): Il nome dell'Autorità di Certificazione che ha rilasciato il certificato.
4. Periodo di Validità: Le date di inizio e fine validità del certificato. Dopo la data di fine, il certificato non è più considerato valido.
5. Firma Digitale della CA: La firma della CA sul certificato stesso. Questa è la prova che la CA garantisce la corrispondenza tra la chiave pubblica e l'identità dichiarata.

Un certificato digitale ha una data di scadenza, ma può anche essere revocato (annullato) prima di tale data. Questo accade se la fiducia associata al certificato viene meno. Le ragioni più comuni per la revoca sono:

- la chiave privata corrispondente è stata compromessa (rubata o esposta);
- l'identità del titolare è cambiata;
- c'è stato un errore nell'emissione del certificato.

AUTORIZZAZIONE

L'autorizzazione è il processo di decidere se un'entità (un utente, un processo, ecc.) può accedere a una specifica risorsa (un file, un database, una funzione) o eseguire una determinata azione (leggere, scrivere, eseguire, eliminare).

Per gestire queste decisioni, i sistemi utilizzano dei modelli che descrivono come i permessi sono assegnati e controllati:

1. Matrice di Controllo Accessi (Access Control Matrix):
   - è un modello concettuale, come una tabella gigante;
   - le righe rappresentano i soggetti (gli utenti o i processi);
   - le colonne rappresentano gli oggetti (le risorse, come file, directory);
   - ogni cella all'incrocio di una riga e una colonna contiene i permessi che quel soggetto ha su quell'oggetto (es. "leggi", "scrivi", "esegui");
   - è un modello utile per capire la logica, ma poco pratico da implementare direttamente per grandi sistemi.
2. Liste di Controllo Accessi (Access Control List - ACL):
   - è una delle implementazioni più comuni della matrice di controllo accessi;
   - per ogni oggetto (risorsa), c'è una lista allegata che specifica quali soggetti (utenti o gruppi) hanno quali permessi su quell'oggetto;

    - è efficiente per rispondere a domande come: "Chi può accedere a questo file?".
3. Liste di Capacità (Capability List):
    - l'opposto dell'ACL;
    - per ogni soggetto (utente), c'è una lista allegata che specifica a quali oggetti quel soggetto può accedere e con quali permessi;
    - è efficiente per rispondere a domande come: "Cosa può fare questo utente?".

Politiche di Controllo Accessi: DAC vs. MAC

Questi modelli possono essere applicati secondo diverse politiche:

1. Controllo Accessi Discrezionale (DAC - Discretionary Access Control):
    - Chi decide: il proprietario della risorsa è colui che decide chi può accedere e con quali permessi. È il modello più comune nei sistemi operativi come Windows o Linux, dove il proprietario di un file può impostare i permessi per altri utenti o gruppi;
    - Flessibilità: è molto flessibile, poiché il proprietario ha il controllo diretto;
    - Problema: non permette di controllare la diffusione dell'informazione. Se un utente ha il permesso di leggere un file, può copiarlo e condividerlo con chiunque, anche con chi non avrebbe i permessi originali. Questo è un limite in ambienti dove il controllo sulla diffusione è critico.
2. Controllo Accessi Obbligatorio (MAC - Mandatory Access Control):
    - Chi decide: la politica di accesso è determinata centralmente dal sistema (o da un'autorità di sicurezza), non dai singoli proprietari delle risorse;
    - Contesto: utilizzato principalmente in ambiti di sicurezza molto elevata, come il settore militare o governativo, dove la classificazione delle informazioni è fondamentale.
    - Come funziona: si basa su una classificazione degli oggetti (es. Top Secret, Secret, Confidential) e dei soggetti (livello di abilitazione). Il sistema forza il rispetto di regole rigide, come nel famoso modello di Bell-LaPadula:
        - "No read up" (Non leggere in su): un soggetto non può leggere informazioni classificate a un livello di sicurezza superiore al proprio;
        - "No write down" (Non scrivere in giù): un soggetto non può scrivere informazioni a un livello di sicurezza inferiore al proprio (per evitare di "declassare" informazioni sensibili).
    - Confronto: è meno flessibile del DAC ma molto più robusto nel garantire la protezione delle informazioni classificate.

Approcci Moderni: Ruoli e Gruppi

Per semplificare la gestione dei permessi in sistemi complessi, si usano:

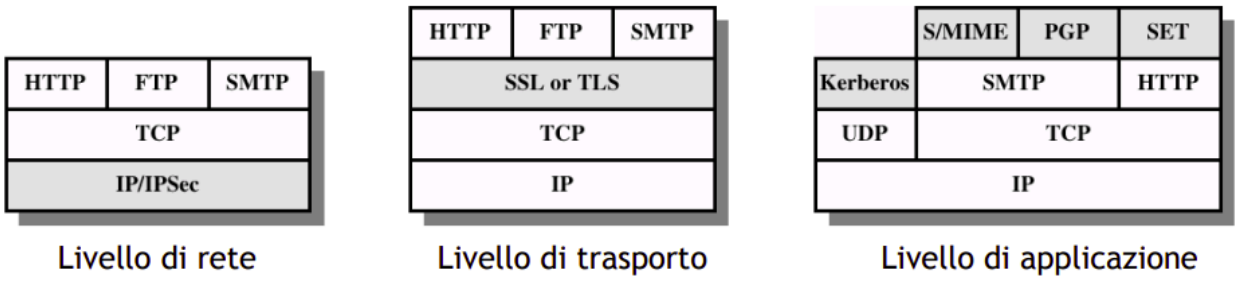
- Controllo Accessi Basato sui Ruoli (RBAC - Role-Based Access Control):
  - invece di associare i permessi direttamente agli utenti, i permessi vengono associati a dei ruoli (es. "Amministratore", "Docente", "Studente", "Impiegato");
  - gli utenti vengono poi assegnati a uno o più ruoli;
  - vantaggi: Semplifica enormemente la gestione (cambiando i permessi di un ruolo, si aggiornano tutti gli utenti con quel ruolo) e riduce gli errori.
- Gruppi:
  - i gruppi permettono di gestire insiemi di soggetti (utenti) o oggetti (risorse) in modo omogeneo;
  - spesso, i permessi vengono assegnati ai gruppi, e gli utenti vengono inseriti nei gruppi;
  - vantaggi: Semplifica l'associazione dei permessi e la modifica dei diritti per intere categorie di utenti.

N.B: in un contesto di autenticazione/registrazione, non è mai il front end ad applicare l'hash della password; la psw viaggia pulita ma attraverso un canale cifrato, prima di arrivare al server e al backend che confronta/memorizza l'informazione. Nel caso del confronto, viene verificato che l'hash memorizzato e l'hash mandato dall'utente siano uguali (non direttamente la psw quindi). In questo modo, se un attaccante dovesse mettere mano alla tabella degli hash, non potrebbe comunque risalire alla psw (no hash inverso).

Protocolli e Software di Sicurezza

La nostra analisi dei protocolli e dei software principali di sicurezza informatica parte, nei livelli TCP/IP, in quello di rete. Questo perché nel livello fisico e data link la sicurezza è trattata in modo concettualmente diverso; verso la fine del capitolo tuttavia, ne vedremo qualche accenno.

Il concetto principale da portare via in questa introduzione è che, un grado di sicurezza innestato ad un certo livello TCP/IP, rende a catena più sicuri i livelli soprastanti. Nella seguente figura, per esempio, l'header di prova più sicuro è il primo, quello che innesta un protocollo aggiuntivo di sicurezza, IPSec in questo caso, nel livello di Rete.



Livello di rete | Livello di trasporto | Livello di applicazione

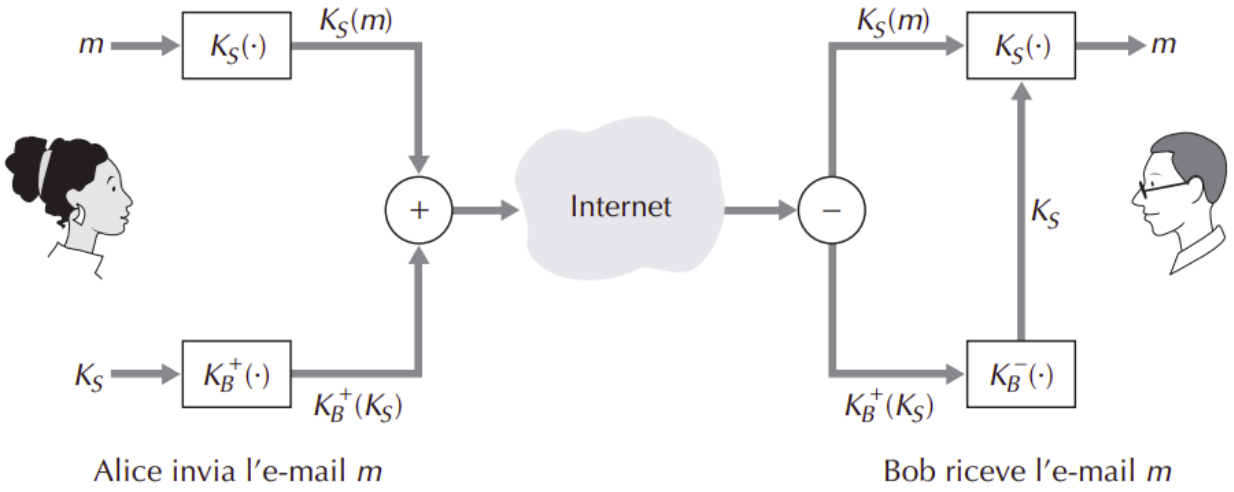
SICUREZZA DELLE MAIL

La posta elettronica, pur essendo uno strumento onnipresente, non è intrinsecamente sicura. Per renderla tale, abbiamo bisogno di garantire alcune caratteristiche fondamentali:

- Riservatezza (Confidentiality): solo il destinatario previsto può leggere il messaggio.
- Autenticazione del mittente: verificare che il messaggio provenga effettivamente da chi dichiara di averlo inviato.
- Integrità dei Dati: assicurarsi che il messaggio non sia stato alterato durante il trasporto.
- Non Ripudiabilità: il mittente non può negare di aver inviato il messaggio.

Per ottenere queste proprietà, le soluzioni di sicurezza delle email combinano i principi della crittografia simmetrica e asimmetrica, oltre alle funzioni hash. Meccanismi base di sicurezza per le email:

1. per la riservatezza (messaggio segreto):
    - mittente (Alice) cifra il messaggio vero e proprio con una chiave simmetrica (es. AES). Questa chiave è temporanea, generata per quel singolo messaggio;
    - per far sì che solo il destinatario (Bob) possa leggere il messaggio, Alice cifra questa chiave simmetrica con la chiave pubblica di Bob;
    - Alice invia il messaggio cifrato e la chiave simmetrica cifrata;
    - Bob riceve, usa la sua chiave privata per decifrare la chiave simmetrica, e poi usa quest'ultima per decifrare il messaggio. In questo modo, solo Bob può leggere il contenuto;



2. per l'autenticazione, l'integrità e la non ripudiabilità (messaggio firmato):
   - mittente (Alice) calcola l'hash del messaggio originale (un'impronta digitale unica);
   - Alice poi "firma" questo hash cifrandolo con la propria chiave privata. Questo hash cifrato è la sua firma digitale;
   - Alice invia il messaggio originale (in chiaro) insieme alla sua firma digitale;
   - destinatario (Bob) riceve il messaggio e la firma. Calcola a sua volta l'hash del messaggio originale e, separatamente, decifra la firma di Alice usando la chiave pubblica di Alice;
   - se l'hash calcolato da Bob coincide con l'hash ottenuto decifrando la firma, allora Bob sa che il messaggio non è stato alterato (integrità) e che proviene effettivamente da Alice (autenticazione e non ripudiabilità);
3. combinazione (Messaggio segreto e firmato):
   - per avere tutte le caratteristiche di sicurezza, si combinano i due schemi. Il mittente prima firma digitalmente il messaggio (garantendo integrità e autenticazione). Poi, cifra l'intero blocco (messaggio + firma) usando una chiave simmetrica, che a sua volta cifra con la chiave pubblica del destinatario (garantendo riservatezza);
   - Il destinatario decifra il tutto con la sua chiave privata, estrae il messaggio e la firma, e poi verifica la firma con la chiave pubblica del mittente.

Questo approccio è modulare ma può diventare complesso nella gestione delle chiavi e dei passaggi. Per questo, sono stati sviluppati standard e applicazioni che automatizzano il processo. Esistono sostanzialmente due soluzioni principali per applicare questi principi alla posta elettronica:

1. Pretty Good Privacy (PGP):
   - è stata una delle prime e più popolari applicazioni software per la crittografia delle email, sviluppata da Phil Zimmermann. Offre servizi di sicurezza completi,

garantendo la riservatezza dei messaggi, l'autenticazione del mittente, l'integrità dei dati e la non ripudiabilità. Per ottimizzare, PGP include la compressione prima della cifratura e funzioni di compatibilità email per gestire vari tipi di allegati e messaggi di grandi dimensioni tramite segmentazione. Utilizza una combinazione di algoritmi: RSA per la crittografia a chiave pubblica, algoritmi simmetrici come IDEA, Triple DES o AES per il messaggio, e SHA-1 per l'hashing. Il suo modello di fiducia distintivo è il "Web of Trust": una rete decentralizzata dove gli utenti attestano la validità delle chiavi pubbliche altrui firmandole, a differenza del modello centralizzato delle Autorità di Certificazione. Le chiavi pubbliche sono gestite in un "Public Key Ring" locale;

2. S/MIME (Secure/Multipurpose Internet Mail Extensions):
    - è uno standard ampiamente adottato per la sicurezza delle email, ed è comunemente integrato nei client di posta elettronica commerciali come Microsoft Outlook o Apple Mail. A differenza di PGP, la differenza cruciale di S/MIME risiede nel suo modello di fiducia: si basa sui certificati X.509. Questo significa che la validità delle chiavi pubbliche e l'identità dei mittenti sono attestate da una Gerarchia di Autorità di Certificazione (CA), che agiscono come enti centralizzati e fidati. S/MIME fornisce servizi di sicurezza essenzialmente identici a quelli di PGP: garantisce la riservatezza del messaggio, l'integrità dei dati, l'autenticazione del mittente e la non ripudiabilità.

SICUREZZA DEL LIVELLO DI TRASPORTO:

La sicurezza a questo livello è garantita principalmente da SSL (Secure Sockets Layer) e dal suo successore più moderno, TLS (Transport Layer Security). Questi protocolli sono diventati lo standard di fatto per stabilire canali di comunicazione sicuri tra un client (ad esempio, il tuo browser) e un server (ad esempio, un sito web) su Internet. La loro applicazione più visibile è l'HTTPS, che altro non è che il protocollo HTTP che viaggia su un canale protetto da SSL/TLS. Caratteristiche di Sicurezza Offerte da SSL/TLS:

1. Riservatezza (Confidentiality):
    - assicura che i dati scambiati tra client e server non possano essere letti da intercettatori esterni;
    - questo si ottiene usando la crittografia simmetrica (con algoritmi come AES, RC4) per cifrare il flusso di dati effettivo;
    - la chiave simmetrica, necessaria per la cifratura/decifratura, viene scambiata in modo sicuro usando la crittografia asimmetrica (con algoritmi come RSA o Diffie-Hellman) durante la fase iniziale della connessione.

2. Autenticazione:
    - autenticazione del Server (obbligatoria): il client verifica l'identità del server per assicurarsi di non connettersi a un impostore. Questo avviene tramite i certificati X.509 del server, firmati da un'Autorità di Certificazione (CA) fidata;
    - autenticazione del Client (opzionale): in alcuni scenari (es. accesso a risorse aziendali protette), anche il server può richiedere al client di autenticarsi con un proprio certificato digitale.
3. Integrità dei Dati:
    - garantisce che i dati trasmessi non siano stati alterati o manomessi durante il trasporto;
    - si ottiene calcolando un Message Authentication Code (MAC) o un hash crittografico (es. SHA) del messaggio, che viene poi trasmesso insieme al messaggio stesso. Il ricevente ricalcola il MAC/hash e lo confronta con quello ricevuto.

Il cuore di SSL/TLS è l'Handshake Protocol, una serie di passaggi che client e server eseguono all'inizio di ogni sessione per stabilire una connessione sicura e generare una chiave simmetrica condivisa (la "session key") che verrà usata per tutta la comunicazione successiva:

1. ClientHello: il client avvia la connessione, inviando le versioni di SSL/TLS che supporta, le suite di cifratura che preferisce e un numero casuale (nonce);
2. ServerHello:i il server risponde, scegliendo la versione e la suite di cifratura che supporteranno entrambi, invia un proprio numero casuale e il suo certificato digitale (X.509);
3. Scambio di chiavi: il client verifica la validità del certificato del server (firmato da una CA fidata). Se valido, il client genera un "pre-master secret" (un'altra componente casuale) e lo cifra usando la chiave pubblica del server (ottenuta dal certificato). Invia questo segreto cifrato al server. Solo il server, con la sua chiave privata, può decifrarlo.
4. Generazione della chiave di sessione: entrambi (client e server) ora possiedono tutti gli ingredienti (i due numeri casuali, il pre-master secret) per derivare indipendentemente la chiave simmetrica di sessione. Questa chiave non viene mai trasmessa direttamente;
5. ChangeCipherSpec & Finished: client e server si notificano a vicenda che passeranno ora alla comunicazione cifrata e si scambiano messaggi di "Finished" (cifrati con la nuova chiave di sessione) che fungono da verifica finale dell'handshake.

HTTPS e la Difesa dal Man-in-the-Middle (MITM):

HTTPS è semplicemente HTTP che utilizza SSL/TLS come strato di protezione. Opera sulla porta 443 (anziché la 80 di HTTP). SSL/TLS è la difesa chiave contro gli attacchi Man-in-the-Middle (MITM). In un MITM, un attaccante si interpone tra il client e il server, intercettando e potenzialmente alterando la comunicazione. Il ruolo del certificato del server (firmato da una CA) è cruciale qui: quando il tuo browser verifica che il certificato del sito è valido e firmato da una CA fidata (le cui chiavi pubbliche sono pre-installate nel tuo sistema), si assicura che sta parlando con il server legittimo e non con un impostore. Se l'attaccante tentasse di presentare un falso certificato, il browser lo rileverebbe perché non sarebbe firmato da una CA fidata o perché i dettagli non corrisponderebbero. Anche la revoca dei certificati è fondamentale per la sicurezza di SSL/TLS, permettendo di invalidare certificati compromessi prima della loro scadenza naturale.

SICUREZZA DELLE WIRELESS LAN:

Le nostre reti Wi-Fi, pur essendo comodissime, portano con sé una sfida di sicurezza intrinseca: sono "nell'aria". A differenza delle connessioni cablate, dove un cavo fisico limita l'accesso, le onde radio si diffondono, rendendo potenzialmente accessibile a chiunque nel raggio d'azione il traffico che viaggia sulla rete. È come parlare in una piazza affollata anziché in una stanza chiusa. Per questo, la protezione delle reti wireless è diventata fondamentale. Durante la storia sono stati sviluppati diversi protocolli; vediamoli in ordine:

1. WEP (Wired Equivalent Privacy):
    a. agli albori delle reti Wi-Fi, lo standard che doveva garantire la sicurezza era il WEP. L'idea era semplice: usare una chiave segreta condivisa e l'algoritmo RC4 per cifrare i dati, in modo da rendere la rete "equivalente" a una cablata in termini di privacy. Sembrava una buona idea sulla carta. Purtroppo, il WEP si è rivelato un fallimento clamoroso. Aveva gravi difetti di progettazione: usava un valore, l'Initialization Vector (IV), troppo corto e spesso riutilizzato, il che permetteva agli attaccanti di scoprire facilmente la chiave di cifratura. Inoltre, il suo meccanismo di controllo dell'integrità era debole e facilmente manipolabile. In pratica, il WEP è stato rotto anni fa e non offre alcuna protezione reale. Se la tua rete usa ancora WEP, è come se non avesse alcuna password. Spesso usata per ‘attacchi miele’, ossia che attira vittime come api con il miele attraverso reti wifi aperte e libere, per poi attaccare brutalmente chi ci si collega;
2. WPA (Wi-Fi Protected Access):
    a. quando le vulnerabilità di WEP divennero evidenti e pericolose, fu chiaro che serviva una soluzione immediata. Nacque così il WPA, pensato come un ponte tra

il WEP e uno standard più robusto che sarebbe arrivato in futuro. L'obiettivo era migliorare la sicurezza senza richiedere alle persone di comprare hardware nuovo di zecca. WPA ha introdotto il TKIP (Temporal Key Integrity Protocol), un protocollo che gestiva le chiavi molto meglio, cambiandole dinamicamente e introducendo un controllo di integrità (MIC - Message Integrity Check) più serio rispetto al vecchio WEP;

3. WPA2 (Wi-Fi Protected Access II):
   a. la vera soluzione robusta è arrivata con il WPA2, che rappresenta la piena implementazione dello standard IEEE 802.11i. Questo è il protocollo che dovremmo usare oggi per proteggere le nostre reti Wi-Fi. Il cuore crittografico di WPA2 è l'algoritmo AES (Advanced Encryption Standard), utilizzato in una modalità particolarmente sicura chiamata CCMP. Questo garantisce sia una cifratura molto forte per la riservatezza dei dati, sia un'integrità robusta per assicurare che nessuno abbia manomesso i tuoi dati.

Intrusion Detection System

Immagina un Firewall come un vero e proprio "posto di blocco" o un doganiere per il traffico di rete. La sua funzione principale è controllare e filtrare tutto ciò che entra ed esce da una rete, specialmente tra due aree che hanno livelli di sicurezza diversi, come la tua rete interna e l'Internet pubblico. L'obiettivo è quindi impedire accessi non autorizzati dall'esterno verso la tua rete privata, controllare cosa esce (per evitare fughe di dati) e nascondere o "schermare" parti della tua rete interna agli sguardi indiscreti. È anche uno strumento prezioso per monitorare il traffico, grazie ai suoi log dettagliati. Si assume tuttavia che la minaccia principale venga dall'esterno, ma gli attacchi possono anche nascere dall'interno. Non possono proteggere da bug sconosciuti nei protocolli o da configurazioni errate, e impostarli bene è un equilibrio delicato tra libertà e sicurezza. A volte, possono anche rallentare un po' la rete. Nella configurazione di un firewall, ci sono due approcci filosofici opposti:

- Default Deny (tutto ciò che non è espressamente permesso è proibito): questa è la filosofia più sicura. Il firewall blocca per impostazione predefinita ogni connessione. Sei tu, l'amministratore, a dover specificare esplicitamente quali servizi o tipi di traffico sono consentiti. È un approccio proattivo che richiede un'analisi attenta di ciò che ti serve, ma rende molto difficile per un attaccante aggirare le regole;
- Default Permit (tutto ciò che non è espressamente proibito è permesso): l'approccio opposto. Il firewall permette tutto il traffico a meno che tu non lo blocchi esplicitamente. Questo può essere più comodo all'inizio, ma ti costringe a reagire

ogni volta che viene scoperta una nuova vulnerabilità, chiudendo "buchi" man mano che appaiono. È un approccio reattivo e meno sicuro.

I firewall si distinguono principalmente per il "livello di intelligenza" con cui analizzano il traffico:

1. Packet Filter (Filtro a Pacchetto):
   - sono i più semplici e "stupidi". Prendono decisioni solo basandosi sulle intestazioni dei pacchetti di rete, guardando informazioni come l'indirizzo IP di origine e destinazione, la porta (es. porta 80 per il web, 25 per l'email) e il tipo di protocollo (TCP, UDP, ICMP). Sono veloci e trasparenti, ma non "capiscono" il contesto di una connessione e sono vulnerabili a tecniche come l'IP spoofing. Le regole complesse sono difficili da gestire;
2. Stateful Inspection (Ispezione dello Stato):
   - rappresentano un notevole passo avanti. Questi firewall non si limitano a guardare le intestazioni, ma tengono traccia dello "stato" delle connessioni attive. Ad esempio, sanno se un pacchetto TCP fa parte di una sessione già stabilita (come dopo un handshake TCP completo). Questa "memoria" del contesto rende la filtrazione molto più intelligente e sicura, permettendo di bloccare pacchetti che sembrano legittimi ma non fanno parte di una conversazione in corso.
3. Application-level Gateway (Proxy Firewall):
   - sono i più sofisticati e, potenzialmente, i più sicuri. Invece di far passare il traffico direttamente, agiscono come un intermediario (un "proxy"). Quando il client vuole connettersi a un server esterno, si connette prima al proxy firewall. Il proxy analizza la richiesta a livello di applicazione (quindi "capisce" protocolli come HTTP, FTP, SMTP), ispezionando anche il contenuto del dato, non solo le intestazioni. Se la richiesta è valida, il proxy crea una nuova connessione con il server esterno e inoltra la richiesta. Il suo vantaggio principale è che non c'è mai una connessione diretta tra client interno e server esterno, aumentando enormemente la sicurezza e mascherando la struttura della rete interna. Lo svantaggio è che possono introdurre un certo overhead di performance e richiedono un proxy specifico per ogni servizio che vuoi filtrare a questo livello.

Architetture e buone pratiche:

I firewall possono essere implementati in diverse architetture, dalla semplice configurazione su un router a quelle più complesse che includono un DMZ (Demilitarized Zone). La DMZ è una rete "cuscinetto" tra la tua rete interna e Internet, dove vengono posizionati i server pubblici (come i server web o email) in modo che, se venissero compromessi, non

darebbero accesso diretto alla tua rete privata. Per configurare un firewall al meglio, è fondamentale conoscere bene i tuoi asset, implementare sempre la filosofia "default deny", usare i proxy firewall per i servizi critici, e soprattutto, monitorare costantemente i log per rilevare attività sospette.

IDS (Intrusion Detection System):

Se i firewall sono i "guardiani" che decidono chi entra ed esce, gli Intrusion Detection System (IDS) sono i "vigili" che sorvegliano ciò che accade all'interno della rete o di un sistema, alla ricerca di attività sospette che potrebbero indicare una violazione della sicurezza. Il loro scopo è rilevare un'intrusione in corso o un'attività malevola, segnalarla e, in alcuni casi, provare a prevenirla o scoraggiarla. È fondamentale capire che un IDS non è un firewall. Un firewall blocca il traffico in base a regole predefinite; un IDS, invece, analizza il traffico e gli eventi, cercando anomalie o firme di attacchi, e poi allerta l'amministratore. Abbiamo bisogno degli IDS perché molti attacchi riescono a eludere i firewall, o magari provengono dall'interno della nostra stessa rete, da minacce "insider" che un firewall di perimetro non può intercettare. Inoltre, un IDS può aiutarci a scovare attacchi totalmente nuovi, per i quali non esistono ancora regole di blocco. Esistono due metodi fondamentali con cui un IDS cerca le intrusioni:

1. IDS basati su firma (Signature-based):
    - sono come degli "identikit" degli attacchi. Questi IDS hanno un database di firme o modelli predefiniti di attacchi conosciuti (es. una sequenza specifica di pacchetti, un codice malevolo). Quando il traffico o un evento corrisponde a una di queste firme, scatta l'allarme. Sono molto precisi nel rilevare attacchi noti, ma non possono rilevare attacchi nuovi, i cosiddetti "zero-day", per i quali non esiste ancora una firma. Richiedono quindi aggiornamenti costanti del database delle firme.
2. IDS Basati su Anomalia (Anomaly-based):
    - questi IDS non cercano attacchi noti, ma imparano a riconoscere il comportamento "normale" di un sistema o di una rete. Creano una specie di "baseline" del traffico o delle attività tipiche (es. quali programmi vengono eseguiti, a quali ore, con quali volumi di dati). Tutto ciò che devia significativamente da questa "normalità" viene segnalato come un'anomalia, potenzialmente un attacco. Il vantaggio principale è che possono rilevare attacchi sconosciuti o nuove varianti, dato che cercano deviazioni dal comportamento standard, ma tendono a generare più falsi positivi (allerta per attività normali ma inusuali), richiedono un periodo di "addestramento" per imparare la normalità e possono essere ingannati se l'attaccante riesce a mantenere le proprie azioni entro i confini della normalità percepita.

Un'evoluzione degli IDS sono gli Intrusion Prevention System (IPS). La differenza chiave è nel loro comportamento. L'IDS si limita a rilevare e allertare, mentre l'IPS, una volta rilevata un'intrusione, può agire attivamente per bloccarla (es. rilasciare pacchetti, bloccare un IP sorgente, riconfigurare dinamicamente un firewall). Un IPS è quindi in linea di principio più proattivo, ma un'azione automatica basata su un falso positivo può causare interruzioni di servizio legittime.

Infine, un concetto interessante legato al rilevamento è l'Honeypot. Non è un IDS nel senso stretto, ma una risorsa di rete volutamente vulnerabile o ingannevole, progettata per attirare gli attaccanti. L'obiettivo non è servire scopi produttivi, ma agire come "esca" per:

- raccogliere informazioni sulle tecniche e gli strumenti che gli attaccanti utilizzano;
- imparare sui nuovi attacchi (zero-day) prima che vengano usati contro sistemi reali;
- deviare gli attaccanti dai sistemi critici. Un Honeypot è altamente monitorato per osservare ogni mossa dell'intruso.