Appunti Cybersecurity

Alessandro Zappatore

24 novembre 2024

Indice

1 Cybersecurity essentials

1.1 Definizioni di sicurezza

La **sicurezza informatica** è l'insieme dei servizi, delle regole organizzative e dei comportamenti individuali che proteggono i sistemi informatici di un'azienda. Ha il compito di proteggere le risorse da accessi indesiderati, garantire la riservatezza delle informazioni, assicurare il funzionamento e la disponibilità dei servizi a fronte di eventi imprevedibili.

C.I.A = Confidentiality, Integrity, Availability

- **ASSET** (risorsa) = l'insieme di beni, dati e persone necessarie all'erogazione di un servizio IT;
- **VULNERABILITÀ** = debolezza di un asset, quel sistema che fa si che attraverso questa vulnerabilità io possa andare ad eseguire un attacco efficace;
  - eg. debolezza della password;
  - eg. introduzione di un bug durante un aggiornamento software;
  - eg. backdoor.
- **MINACCIA** = evento intenzionale o accidentale (eg. perdere il foglio con le password scritte) che può causare la perdita di una proprietà di sicurezza;
- **ATTACCO** = verificarsi di una minaccia di tipo "evento intenzionale";
- **EVENTO** (negativo) = verificarsi di una minaccia di tipo "evento accidentale";
- **incidente** = un evento di sicurezza che compromette l'Integrità, la confidenzialità o la disponibilità di un asset;
- **(data) breach** = un incidente che genera la rivelazione (disclosure) o la potenziale esposizione (exposure) dei dati;
- **(data) disclosure** = un data breach per cui è stato confermato che dei dati sono stati effettivamente rivelati a persone non autorizzate.

Rischio:

- **probabilità** = più è probabile che un evento accada maggiore attenzione dovrò fare per prevenirlo;
- **impatto** = quanto un evento possa impattare sul nostro sistema. (Anche se un evento è poco probabile che accada ma ha un impatto alto devo comunque proteggermi)

1.2 Proprietà di sicurezza

Autenticazione Il servizio di **autenticazione** si preoccupa dell'autenticità della comunicazione. Vengono definiti due tipi di autenticazione:

1. **autenticazione (della controparte)** = solo l'utente utilizzatore si deve autenticare;
2. **mutua autenticazione (delle controparti)** = sia l'utente che il servizio si devono autenticare;
3. **autenticazione dell'origine dei dati** = avere la certezza che il dato sia stato prodotto dalla persona corretta.

Non ripudio Prova formale, usabile in tribunale, che dimostra in modo innegabile l'autore dei dati. Sono certo di chi ha creato i dati, non solo che sono stati creati nello stesso modo dell'originale. (Eg. firma su un documento)

Autorizzazione (Controllo degli accessi) Il controllo degli accessi è l'abilità di limitare e controllare l'accesso ai sistemi e alle applicazioni tramite canali di comunicazione. Ogni entità interessata ad accedere ad un servizio deve prima essere autenticata.

Confidenzialità (Riservatezza) La confidenzialità è l'atto di proteggere i dati trasmessi dagli attacchi passivi.

Integrità dei dati Il principio di integrità è applicabile ad un flusso di dati e garantisce che i dati non vengano modificati o perso durante il tragitto tra mittente e destinatario.

Non replay Evitare **attacchi** di tipo **replay**, cioè la ripetizione del messaggio. Se viene inviato più volte il dato mi accorgo che sono delle copie.

1.3 Strategie di sicurezza

Security by design Il software è stato progettato fin dalle fondamenta per essere sicuro.

1.3.1 Least Privilege

Un processo o un altro tipo di entità dovrebbe ricevere i **privilegi** e le **risorse minimi** per il **periodo di tempo minimo** richiesto per completare un'attività.

1.3.2 Separation of Duties

La separazione dei compiti richiede che il completamento di una specifica attività sensibile o l'accesso a oggetti sensibili dipenda dal soddisfacimento di una pluralità di condizioni. Attuare un'azione solo se sono soddisfatte 2 o più condizioni in **parallelo**. (Eg. valigetta che richiede due persone per essere aperta)

1.3.3 Separation of Privilege

Interrompere un singolo privlegio tra più soggetti indipendenti in modo che siano necessarie più autorizzazioni per eseguire un'azione. Attuare un'azione solo se sono soddisfatte delle condizioni in **sequenza**. Ho la necessità di qualcuno che validi il mio operato.

1.3.4 Defense in Depth

Avere più livelli di protezione in cui un livello successivo fornirà protezione in caso di violazione di un livello precedente.

1.3.5 Fail Secure, Fail Safe

- **Fail Secure** = Do più importanza alla sicurezza informatica. (Eg. problema nel sistema bancario, non voglio che nessuno entri)
- **Fail Safe** = Do più importanza alla sicurezza delle persone. (Eg. problema al sistema di un ospedale, voglio accedere comunque alle informazioni dei pazienti anche rinunciando alla sicurezza informatica)

1.3.6 Complete Mediation

Ogni richiesta da parte di un soggetto di accedere a un oggetto in un sistema informatico deve essere sottoposta ad una valida ed efficace procedura di autorizzazione. Validare ogni azione compiuta dall'utente.

1.3.7 Least Common Mechanism

Questo principio afferma che un numero minimo di meccanismi di protezione dovrebbe essere comune a più utenti, poiché i percorsi di accesso condiviso possono essere fonti di scambio di informazioni non autorizzato.

1.3.8 Psychological Acceptability

Facilità d'uso e intuitività dell'interfaccia utente che controlla e interagisce con i meccanismi di controllo.

1.3.9 Weakest Link

La sicurezza globale di un sistema è data dalla sicurezza del suo anello debole.

1.3.10 Zero Trust

Si deve presupporre che la sicurezza sia sempre a rischio di minacce (esterne e interne). Sempre effettuare tutti i controlli di sicurezza.

1.3.11 Open Design Principle

Rendere di libero accesso l'implementazione di sicurezza in modo da poter ricevere dei feedback dalle altre persone che possono aiutare a scovare bug ed eventualmente fornire anche le soluzioni.

1.4 Attacchi Cyber

1.4.1 IP spoofing (source address spoofing)

L'attaccante utilizza l'indirizzo IP della vittima e si spaccia per essa.

Attacchi:

- Falsificazione di dati;
- Accesso (non autorizzato) a sistemi.

Contromisure:

- NON usare mai autenticazione basata sugli indirizzi di rete.

1.4.2 Packet sniffing

Lettura dei pacchetti destinati ad un altro nodo della rete.

Attacchi:

- permette di intercettare qualunque cosa (password, dati, $\cdots$).

Contromisure:

- reti non broadcast;
- crittografia del payload dei pacchetti.

1.5 Denial-of-Service (DoS)

Si tiene impegnato un host in modo che non possa fornire i suoi servizi.

Attacchi:

- impedisce l'uso di un sistema/servizio.

Contromisure:

- nessuna definitiva, solo palliativi quantitativi.

1.5.1 Distributed Denial-of-Service (DDos)

Software per DoS installato su molti nodi (daemon, zombie o malbot) costituendo una **Botnet**.

- DDoS a un server;
- DDoS a una rete;
- DDoS a un dispositivo di rete.

1.5.2 Shadow server

Elaboratore che si pone come fornitore di un servizio senza averne il diritto.

Metodi:

- **Sniffing della richiesta** = L'attaccante intercetta la richiesta e si spaccia per il server legittimo;
- **Mapping errato** = faccio un DNS poisoning al fine di deviare il traffico dall'indirizzo corretto a quello malevolo.

Attacchi:

- Fornitura di un servizio sbagliato;
- Cattura di dati forniti al servizio sbagliato.

Contromisure:

- Autenticazione del server.

1.5.3 Connection hijacking (MITM) (data spoofing)

Si prende il controllo di un canale di comunicazione e si inseriscono, cancellano o manipolano dei pacchetti.

Man-in-the-middle MITM:

- **Logico** = Devio il traffico in modo che arrivi al server malevolo;
- **Fisico** = intercetto la comunicazione ponendomi tra il server e il client.

Attacchi:

- Lettura, falsificazione e manipolazione di dati.

Contromisure:

- Risercatezza, autenticazione, integrità e serializzazione di ogni singolo pacchetto di rete.

1.6 Trojan/MATE/MITB

Attaccare i dispositivi più deboli per avere accesso alle informazioni più importanti (Smartphone, smart-TV, $\cdots$).

Man-At-The-End MATE

Man-In-The-Browser MITB

1.7 Malware

- **Virus** = codice che provoca dei danni e si replica, viene propagato dagli umani (involontariamente);
- **Worm** = virus che provoca dei danni perché si replica (satura le risorse), viene propagato automaticamente;
- **Trojan** = vettore di malware, strumento con cui lo introduco nella macchina;
- **Backdoor** = punto di accesso non autorizzato;
- **Rootkit** = strumenti per accesso privilegiato, nascoti ed invisibili.

1.8 Ransomware

Malware che blocca parti delle risorse per poi richiedere un riscatto.

1.9 Phishing

Attacco non tecnologico che utilizza il social engineering. Attrarre via mail o IM l'utente di un servizio di rete su un server fasullo per catturare credenziali di autenticazione o altre informazioni personali oppure convincere la vittima ad installare un plugin o estensione che è in realtà un virus o un trojan.

Varianti:

- **spear phising** = include molti dati personali per aumentare la credibilità del messaggio;
- **whalung** = mirato a persone importanti tipo CEO o CIO.

2 Crittografia

Algoritmo di crittografia Sono delle sequenze algoritmiche di basso livello (xor, and, or, $\cdots$) che mescolano i dati i modo quasi casuale.

Ogni algoritmo di crittografia ha in input una chiave (sequenza di bit), conosciuta solo dal mittente, che viene utilizzata durante la fase di criptazione.

Il destinatario dovrà applicare un algoritmo di decrittografia complementare che userà una chiave, uguale o diversa a quella del mittente, per ottenere il messaggio iniziale.

Principio di Kerchoffs Specifica del principio di open design per gli algoritmi di crittografia.

Se le chiavi:

- sono tenute segrete;
- sono gestite solo da sistemi fidati;
- sono di lunghezza adeguata (altrimenti si potrebbe provare a scoprirla con brute force, 2^n tentativi necessari).

Allora non importa che gli algoritmi di crittografia siano tenuti segreti, anzi è meglio siano pubblici in modo da essere studiati.

2.1 Crittografia simmetrica

Crittografia con **chiave comune e unica**, a **basso carico di elaborazione**. Per la confidenzialità di grandi quantità di dati. Per condividere la chiave comune il miglior modo è utilizzare la crittografia asimmetrica.

2.1.1 Algoritmi di crittografia simmetrica

EX-OR (XOR) Se l'input è casuale anche l'output sarà casuale, abbiamo il 50% di possibilità di ottenere 1 o 0.

Chiave equivalente (resistenza) Numero di bit tali per cui si può scoprire la chiave applicando brute force per 2^n tentativi.

DES Chiave a 56 bit + 8 di parità, blocco dati da 64 bit. Se applicato 3 volte viene detto **3DES** usando 2 o 3 chiavi (solitamente 2 operazioni di crittografia e 1 di decrittografia).

- 3DES con 2 chiavi = Una chiave K_1 per la prima e terza operazione e una chiave K_2 per l'operazione centrale. (Bastano 2^{56} tentativi se si hanno 2^{59}B di menoria)
- 3DES con 3 chiavi = utilizzo tre chiavi diverse per le 3 operazioni. (Servono 2^{112} tentativi)

2DES è vulnerabile a un attacco di tipo known-plaintext (l'attacco si può fare solo se è nota almeno una sola plaintext) detto meet in the middle. Per chiavi di N bit servirebbero solo 2^{N+1} tentativi.

2.1.2 Attacco meet-in-the-middle

L'attaccante deve conoscere una coppia P e C e il numero N di bit della chiave è noto.

$$M = enc(K_1, P)$$

$$C = enc(K_2, M)$$

Calcola prima 2^N valori $X_i = enc(K_i, P)$. Poi calcola 2^N valori $Y_j = dec(K_j, C)$. Cerco $K_i = Y_J$.

IDEA Chiave a 128 bit, blocco dati da 64 bit, utilizza XOR, addizione mod16 e moltiplicazione mod$2^{16}+1$. Più lento rispetto al DES ma più veloce del 3DES.

RC2, RC4 Più veloci di DES, con chiave a lunghezza variabile e blocchi da 64 bit. RC2 accetta chiavi da 8 a 1024 bit (solitamente 64).

AES Ora si usa AES, con chiave fino a 256 bit e blocchi da almeno 128 bit.

2.1.3 Applicazione algoritmi a blocchi

La funzione di base si può applicare solo ad un blocco di dati di lunghezza prestabilita.

ECB (Electronic Code Book) Per cifrare dati in **quantità superiore**. Ogni blocco viene cifrato e decifrato con lo stesso algoritmo separatamente.

Cifratura:

$$C_i = enc(K, P_i)$$

Decifratura:

$$P_i = enc^{-1}(K, C_i)$$

Vantaggi:

- Performance, possibile parallelizzare le operazioni;
- Tolleranza agli errori, un errore in trasmissione provoca errore nella decifratura di un solo blocco.

Svantaggi:

- Basso livello di sicurezza per messaggi lunghi;
- Cifra allo stesso modo blocchi identici e quindi si presta ad attacchi known-plaintext;
- Possibile scambiare due blocchi qualunque.

CBC (Cipher Block Chaining) Per cifrare dati in **quantità superiore**. Richiede IV (Initialization Vector), XOR tra blocco cifrato precedente e blocco da cifrare, poi applicazione algoritmo di cifratura.

Cifratura:

$$C_i = enc(K, P_i \oplus C_{i-1})$$

Decifratura:

$$P_i = enc^{-1}(K, C_i) \oplus C_{i-1}$$

Vantaggi:

- Maggiore sicurezza, no attachi know-plaintext, o block swapping.

Svantaggi:

- Se ho un errore in trasmissione provoca errore nella decifratura di due blocchi;
- Esecuzione più lenta, non parallelizzabile.

Padding Per cifrare dati in **quantità inferiore**. Aggiungo bit per riempire lo spazio vuoto. Alcuni tipi offrono controllo d'integrità, applicando padding a tutti i blocchi. Viene utilizzato solo per l'ultimo blocco di una sequenza più lunga.

Il padding viene utilizzato così tanto che anche se non ci fosse un blocco più piccolo si crea un blocco intero di padding per far si che il ricevente tratti nello stesso modo tutti i casi possibili.

Vantaggi:

- Semplice.

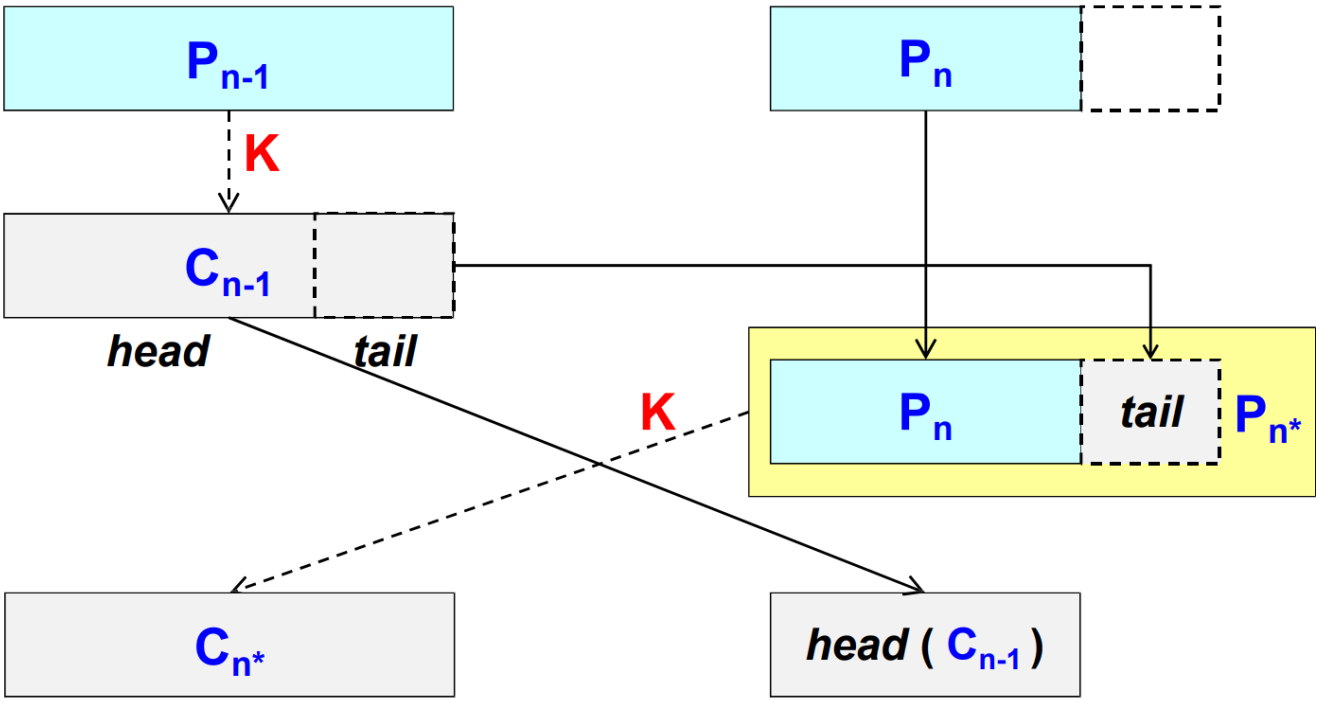
Svantaggi:

- Aggiungo dei bit, quindi trasmetto più dati del necessario;
- Quali bit utilizzare per il padding.

Metodi di padding:

- Un bit a 1 seguito da tutti 0;
- Un byte con valore 128 seguito da byte nulli;
- Ultimo byte pari alla lunghezza del padding;
- Byte nulli;
- Byte random;
- Byte con valore L (lunghezza del padding);
- Numero progressivo da 1 a L;
- Byte con valori L-1.

CTS (Cipher Text Stealing) Per cifrare dati in **quantità inferiore**. L'ultimo blocco è riempito con byte del penultimo blocco, questi due blocchi vengono scambiati durante la cifratura. Deve essere per forza l'utlimo blocco della sequenza, non funzionerebbe se fosse il primo.



Criptazione:

1. Ottengo C_{n-1} cifrando con la chiave K P_{n-1};
2. vado a vedere quanti bit mancano per ottenere l'ultimo blocco completo, allora divido C_{n-1} in due sotto blocchi $head = dimensionedelblocco - k$ e $tail = k$;
3. Estraggo la coda scorporandola dalla testa e la concateno a P_n;
4. Applico l'algoritmo di crittografia con la chiave K su P_{n*} per ottenere il blocco di cifratura ibrido C_{n*};
5. In fine trasmetto C_{n*} e solo la testa di C_{n-1}.

Decriptazione:

1. Decifra C_{n*} per ottenere P_n e la coda;
2. Arriva la testa e la combina con la coda ottenuta prima per ottenere C_{n-1};
3. Ricava P_{n-1};
4. Riordina P_{n-1} e P_n ed ottiene il plain text originale.

Vantaggi:

- Non aggiungo dati in trasmissione.

Svantaggi:

- Aumento il tempo e del carico di elaborazione.

CTR (Counter mode) Per cifrare dati in **quantità inferiore**. Accesso random al testo cifrato, usa algoritmo a blocchi per cifrare n bit alla volta. Non cifro direttamente il blocco, metto i dati in xor con un blocco a parte. I dati del blocco devono essere noti solo al mittente e destinatario, che devono essere d'accordo sul nonce (numero utilizzabile una sola volta e poi cambiato) da generare e il contatore.

Vantaggi:

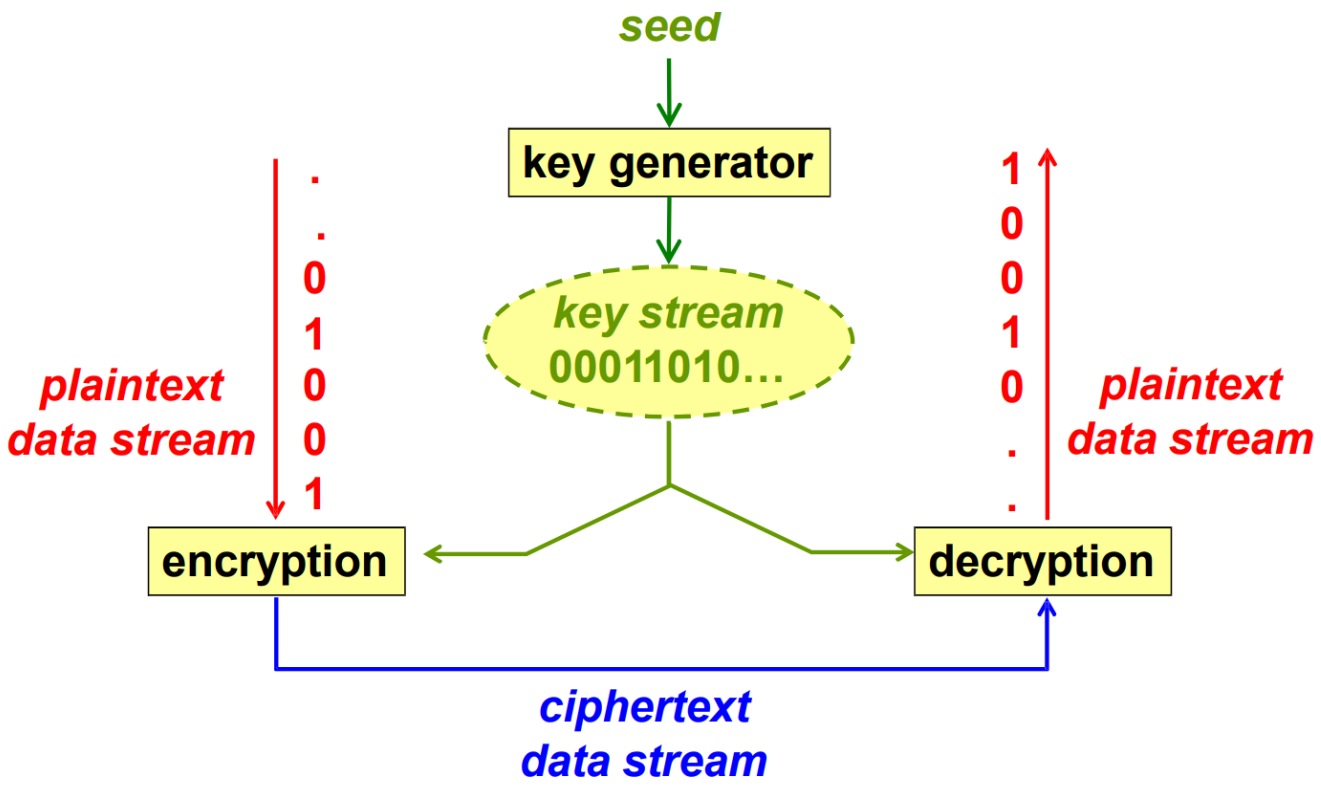
- Tolleranza agli errori, un solo blocco viene danneggiato.

Svantaggi:

- La generazione del nonce deve essere robusta.

2.1.4 Algoritmi stream

Operano su un flusso di dati senza richiederne la divisione in blocchi, tipicamente su un bit o byte. Vecchi tipi RC4, SEAL.



Salsa20 e ChaCha20 Chiavi da 128 o 256 bit (lunghezza dell'input del key generator). Operazione base: add-rotate-xor su 32 bit. Effettuano 20 volte mixing su input. Ottengo un blocco di 512 bit di keystream.

$$f(Key256, Nonce64, Counter64)$$

key è una passphrase condivisa tra mittente e destinatario.

Salsa20 effettua 20 volte il mixing sull'input, esistono anche Salsa12 e Salsa8 (più veloci ma meno sicuri). ChaCha20 utilizza nonce da 96 bit e ridotto il block counter a 32 bit.

2.1.5 Distribuzione chiavi

Per una comunicazione privata tra n persone occorrono $\frac{n(n-1)}{2}$ chiavi. Avviene tramite algoritmi per scambio chiavi.

Si possono distribuire su un canale diverso rispetto a quello utilizzato per il dato **OOB (Out-Of-Band)**. Eg. passaggio della chiave fisicamente tra mittente e destinatario.

2.2 Crittografia asimmetrica

Le chiavi sono diverse e hanno funzionalità reciproca (se cifro con la privata decifro con la pubblica e viceversa). È possibile generare un messaggio segreto per uno specifico destinatario conoscendone solo la chiave pubblica. Più lento rispetto alla crittografia simmetrica. Usata per condividere pochi dati (eg. chiave) oppure per una chiave digitale con le funzioni di hashing.

Riservatezza senza segreti condivisi Si cripta con la chiave pubblica e decripta con la privata perché posseduta solo dal ricevente.

Firma digitale Si usa la chiave privata per cifrare, perché dimostra che il mittente è l'unico che può cifrare quei dati dimostrandone il possesso effettivo.

2.2.1 Algoritmi a chiave pubblica

DSA Elevamento a potenza e logaritmo del risultato, utilizzato solo per firma digitale.

RSA Può solo cifrare dati il cui valore sia inferiore al modulo pubblico. Funzionamento:

1. modulo pubblico $N = P * Q$, con P e Q primi, grandi e segreti
2. $\phi = (P-1)(Q-1)$
3. esponente pubblico E tale che $1 < E < \phi$, E coprimo (non devono condividere divisori, eccetto 1) di ϕ
4. esponente privato: $D = E^{-1}\%\phi$
5. chiave pubblica: (N, E), chiave privata: (N, D)
6. P e Q vengono cancellati.

RSA può cifrare solo dati il cui valore sia inferiore al modulo N.
Cifratura:

$$c = p * E\%N$$

Decifratura:

$$p = c^D\%N$$

Solitamente le chiavi pubbliche hanno un E che contiene solo due bit a 1 per ottimizzare le prestazioni.
Debolezze:

- Se vengono utilizzati esponenti piccoli per la chiave pubblica;
- Se uso la stessa chiave per firma e cifratura;
- Usare l'utente come "oracolo".

Soluzioni:

- Usare il numero di Fermat, 65537;
- Aggiungere sempre del padding fresco prima di cifrare il messaggio;
- Usare chiavi RSA diverse per cifrare e firmare;
- Non accettare di cifrare o formare dati grezzi, controllare il formato del documento;
- L'utente deve sempre controllare il blocco decifrato controllando il formato e restituendo un errore generico.

2.2.2 Distribuzione chiavi per crittografia asimmetrica

Diffie-Hellman (algoritmo di key-agreement) Sfrutta la difficoltà di risoluzione del problema dell'algoritmo discreto.

1. A e B concordano due interi pubblici g (generatore) e p (primo, grande), $1 < g < p$
2. Lunghezza chiave DH = numero di bit di p;
3. A scelto a caso un intero grande $x > 0$, calcola: $X = g^x\%p$;

4. B scelto a caso un intero grande $y > 0$, calcola: $Y = g^y \% p$;
5. A e B si scambiano pubblicamente X e Y;
6. A calcola $K = Y^x \% p$;
7. B calcola $K' = X^y \% p$;
8. $K = K' = g^{xy} \% p$;
9. K viene utilizzato per altri algoritmi di crittografia.

Se l'attaccante può manipolare i dati allora è possibile un attacco **man-in-the-middle**. Risolvibile richiedendo pre-autenticazione.

La complessità per rompere Diffie-Hellman sta nella complessità di estrarre un logaritmo discreto. Si utilizzano le moltiplicazioni successive per trovare x e y, ha tempo lineare in P ma esponenziale nel numero di bit di P, come è esponenziale il tempo per fattorizzare dei numeri primi.

L'unico algoritmo polinomiale nel numero di bit in P che può risolvere RSA e Diffie-Hellman è l'algoritmo di Shore ma è necessario un computer quantistico per risolverlo.

Curve ellittiche Problema del logaritmo discreto sulla curva, più complesso, permette di avere chiavi più corte. Firma digitale: ECDSA, key agreement: ECDH, key distribution: ECIES.

2.3 Funzioni di hash e digest

2.3.1 Digest

È un riassunto a lunghezza fissa del messaggio da proteggere. Deve essere veloce da calcolare, difficile da invertire e non generare troppe collisioni (digest uguali). Un algoritmo di digest a n bit è insicuro quando vengono generati più di $2^{\frac{n}{2}}$ digest perché si ha una probabilità di collisione pari al 50%.

2.3.2 Funzioni di hash

Dividono il messaggio in blocchi e applicano la funzione base per ottenere il valore di hash. Messaggio M in N blocchi.

$$V_k = f(V_{k-1}, M_k)$$

$V_0 = IV$ e $h = V_N$

Solo l'ultimo è il digest dell'intero messaggio, perché step by step mantengo ancora i valori precedenti.

2.3.3 SHA

SHA-1 Algoritmo con blocco da 512 bit e digest da 160 bit che è stato rotto dopo un po' a causa del digest troppo piccolo;

SHA-2 Per risolvere velocemente l'attacco ad SHA-1 si è modificato solo il numero di bit 512 per il blocco e da 224 a 512 bit per il digest, ma **senza modificare l'algoritmo** in se.

SHA-3 Per SHA-3 si è indetta una competizione che ha portato alla scrittura di un **nuovo algoritmo** con blocco tra 1152 e 576 bit e digest da 224 a 512 bit.

2.3.4 KDF (Key Derivation Function)

Algoritmo che deriva una o più chiavi segrete da una password utilizzando una funzione pseudorandom.

2.3.5 MAC, MIC, MID

Come sfruttare l'hash per creare funzioni sicure.

- **MIC** (Message Integrity Code): per garantire l'integrità dei messaggi viene aggiunto il digest a partire dal messaggio;
- **MAC** (Message Authentication Code): fornisce autenticazione e integrità, perché senza integrità ha poco senso, in quanto il contenuto potrebbe essere si autenticato ma allo stesso tempo modificato da un attaccante;
- **MID** (Message IDentifier): identificatore univoco per evitare attacchi di tipo replay, aggiungendo un id al messaggio.

2.3.6 Autenticazione tramite cifratura simmetrica

Si invia una copia cifrata dei dati e una in chiaro, solo chi conosce la chiave può confrontare la copia con l'originale.

L'**integrità** è data dal fatto che non è possibile decifrare partendo da un messaggio diverso da quello inviato con la stessa chiave ottenere dei dati corretti.

L'**autenticazione** viene mantenuta dal fatto che io ho un segreto condiviso.

Vantaggi:

- verifica dell'integrità esatta e non approssimata;
- per messaggi molto piccoli il digest potrebbe causare molte collisioni favorendo questa tecnica;
- non è necessaria la funzione di hash.

Svantaggi:

- l'attaccante conoscendo il messaggio in chiaro e quello cifrato può risalire alla chiave segreta;
- operazioni complesse;
- raddoppio dei dati trasmessi;
- tempo più che raddoppiato.

2.3.7 Autenticazione tramite digest e cifratura simmetrica

Si invia un digest cifrato dei dati, solo chi conosce la chiave può confrontare il digest trasmesso con quello calcolato sui dati ricevuti in chiaro.

L'**integrità** è data dal fatto che il digest non è invertibile.

L'**autenticazione** è data dal fatto che sfrutto la chiave condivisa. Vantaggi:

- pochi dati aggiuntivi.

Svantaggi:

- due operazioni (digest + criptazione);
- rimane il problema dello scambio della chiave.

2.3.8 Autenticazione tramite keyed-digest

Si invia un digest calcolato non solo sui dati ma anche sulla chiave. Soluzione più veloce. Attaccabile scambiando l'ordine dei blocchi.

Il messaggio viene concatenato con la chiave e del messaggio + chiave faccio l'hash. Il destinatario calcola e confronta l'hash per verificare integrità e autenticazione.

Integrità e **autenticazione** sono le stesse della cifratura tramite digest + cifratura simmetrica ma in maniera più veloce.

HMAC La funzione di hash prende in input un blocco da b byte e genera un blocco da l byte, con $b \geq l$.

CBC-MAC Sfrutta un algoritmo di cifratura simmetrico a blocchi, in modalità CBC con IV nullo, prendendo come MAC la cifratura dell'ultimo blocco. È sicuro solo per messaggi a lunghezza fissa.

Vantaggi:

- una sola operazione di hash (digest);
- pochi dati aggiuntivi.

2.3.9 Garantire integrità e riservatezza

1. **authenticate-and-encrypt** (A&E): si decifra prima di verificare l'integrità, vulnerabile a attacchi DoS. La riservatezza la eseguo con una chiave K_1 per l'integrità applico, per esempio keyed-digest, con una chiave K_2.
$$enc(K_1, p) \| mac(K_2, p)$$
2. **authenticate-then-encypt** (AtE): si decifra prima di verificare l'integrità, vulnerabile a attacchi DoS, sicura solo con CBC o cifratura stream.
$$enc(K_1, p \| mac(K_2, p))$$
3. **encrypt-then-authenticate** (EtA): si può evitare di decifrare se il MAC è errato, sicura, bisogna includere nel MAC l'IV e gli algoritmi.
$$enc(K_1, p) \| mac(K_2, enc(K_1, p))$$

2.3.10 Authenticated encryption

Unica opzione che garantisce riservatezza e autenticazione. Si usa una sola chiave e un solo algoritmo, più veloce, meno errori nel combinare le funzioni. Crea **IGE** (Infinite Garble Extension), che causa errore su tutti i blocchi dopo quello manipolato in caso di attacco.

Confronto algoritmi AE:

- **GCM** (Galois/Counter Mode): il più popolare, on-line (l'algoritmo agisce sull'input al momento senza averlo tutto) single-pass AEAD, parallelizzabile
- **OCB** 2.0 (Offset Codebook Mode): il più veloce, on-line single-pass AEAD
- **EAX** (Encrypt then Authenticate then X(trans)late): on-line double-pass AEAD, lento ma piccolo
- **CCM** (CTR mode with CBC-MAC): off-line double pass, il più lento

AEAD (Authenticated Encryption with Associated Data) Schema di AE che contiene dati associati non confidenziali, la cui integrità è protetta. Usata nell'header dei pacchetti rete.

2.3.11 Autenticazione tramite digest e cifratura asimmetrica

Si invia anche un digest, cifrato con la chiave privata del mittente. Solo chi conosce la chiave pubblica può confrontare il digest trasmesso con quello calcolato sui dati ricevuti.

2.3.12 Firme RSA e funzioni hash

Una funzione di hash da usarsi in uno schema RSA deve essere resistente alle collisioni e difficile da invertire (quindi da falsificare).

2.3.13 PKCS

Sono un gruppo di standard.

PKCS #1 definisce le primitive per l'uso di RSA: conversione e rappresentazione di grandi numeri, algoritmi base di cifratura/decifratura, algoritmi base di firma e verifica. Queste primitive devono essere usate per creare uno schema crittografico sicuro.

- schemi di cifratura/decifratura: RSAES-OAEP, RSAES-PKCS
- schemi di firma/verifica (con appendice):

  Gli schemi di firma sono detti con appendice perché non si cifrano i dati ma un loro riassunto (hash), RSA tratta direttamente solo dati di dimensione minore del modulo pubblico n.

2.3.14 Autenticazione e integrità: analisi

Tramite segreto condiviso:

- utile solo per il ricevente
- non usabile come prova senza rivelare la chiave segreta
- non usabile per il non ripudio

Tramite crittografia asimmetrica:

- essendo lenta la si applica solo al digest
- usabile come prova formale
- usabile per il non ripudio
- equivale alla firma digitale

Con una sola chiave privata è possibile generare infinite firme digitali.

2.4 Prestazioni crittografiche

Le prestazioni dipendono dalla CPU e dalla sua cache. Non sono un problema sul client, ma potrebbero esserlo sul server, ovviato con acceleratori crittografici. L'algoritmo più veloce è RC4, seguito da AES-128-CBC, DES-CBC e DES-EDE3-CBC. RSA 1024 è più lento di RSA 2048.

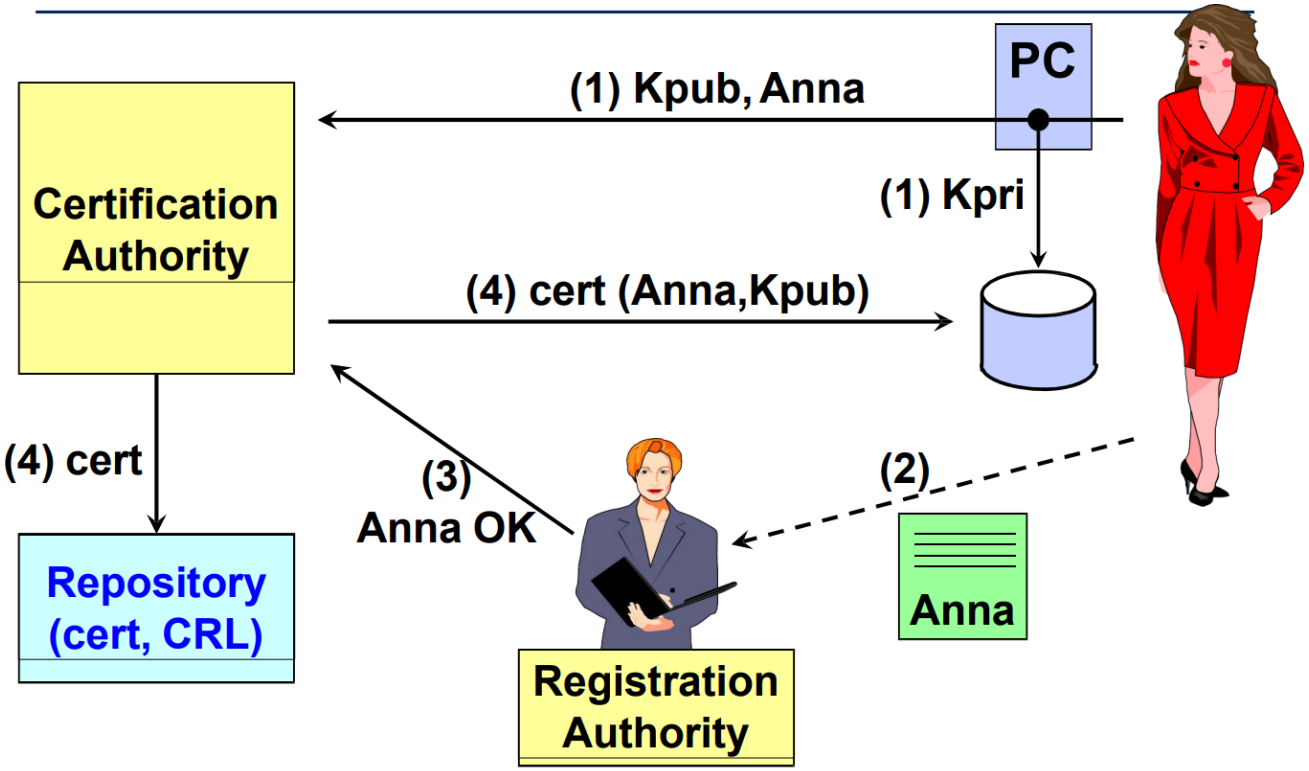
1. lunghezza chiavi crittografia simmetrica: 80, hash: 160, lunghezza chiavi crittografia asimmetrica: 1248, ECC: 160 = attaccabile in poco tempo
2. lunghezza chiavi crittografia simmetrica: 96, hash: 192, lunghezza chiavi crittografia asimmetrica: 1776, ECC: 192 = legacy
3. lunghezza chiavi crittografia simmetrica: 112, hash: 224, lunghezza chiavi crittografia asimmetrica: 2432, ECC: 256 = attaccabile a medio termine
4. lunghezza chiavi crittografia simmetrica: 128, hash: 256, lunghezza chiavi crittografia asimmetrica: 3248, ECC: 160 = sicurezza a lungo termine
5. lunghezza chiavi crittografia simmetrica: 256, hash: 512, lunghezza chiavi crittografia asimmetrica: 15424, ECC: 512 = per ora inattaccabile senza computer quantistici

3 Certificati

È una struttura dati per legare in modo sicuro una chiave pubblica ad alcuni attributi. Tipicamente lega chiave a identità, è firmato in modo elettronico dall'emettitore, detto autorità di certificazione (CA). Ha una scadenza e può essere revocato.

PKI: infrstruttura tecnica e organizzativa preposta alla creazione, distribuzione e revoca dei certificati a chiave pubblica. È composta da:

- end entity: utente finale, umano o non umano
- CA: autorità fidata da uno o più utenti che crea e assegna certificati e CRL, firmandoli digitalmente
- RA: componente opzionale che può essere utilizzata per alleggerire il carico di lavoro del CA, tipo verificare l'identità di una end entity
- repository: denota ogni metodo utilizzato per memorizzare i certificati e le CRL
- relying party: ogni entità che dipende dai dati in un certificato per prendere decisioni



3.1 Certificati X.509

Struttura: versione, numero di serie, algoritmo per la firma, richiedente, lasso di validità, soggetto, informazioni chiave pubblica, firma digitale.

X.509 si basa sull'uso di crittografia a chiave pubblica e firme digitali. Lo standard non detta l'uso d uno specifico algotitmo. Le informazioni contenute nel certificato sono firmate calcolando il valore di hash delle informazioni stesse e generando una firma digitale utilizzando il valore di hash e la chiave privata della CA. Il certificato può poi essere distribuito.

Tutti gli utenti che sono a conoscenza della chiave pubblica del CA possono verificare la chiave pubblica dell'utente certificata. Nessuno oltre al CA può modificare il certificato senza essere scoperto.

3.2 Revoca dei certificati

Un certificato può essere revocato prima della sua scadenza naturale, su richiesta dei titolare o automaticamente dall'emettitore. Quando si valida una firma si deve verificare che il documento fosse valido all'atto della firma. La varifica è a carico del ricevente, il relying party (RA). Meccanismi di revoca:

- CRL (Certificate Revocation List): elenco dei certificati revocati, firmato dalla CA o da un delegato
- OCSP (On-line Certificate Status Protocol): risposta puntuale su un singolo certificato, firmato dal server

Struttura CRL X.509: versione, algoritmo di firma, richiedente, data di aggiornamento, elenco date revoche certificati, firma digitale del CA. Le CRL sono firmate dalla CA che ha emesso i certificati e da una revokation authority delegata dalla CA.

3.2.1 X.509 versione 3

Raccoglie in un unico documento le modifiche necessarie a estendere le definizioni dei certificati e delle CRL. Esistono estensioni di due tipi: pubbliche, ossia definite nello standard e quindi note a tutti, e private, uniche per una certa comunità di utenti. Le estensioni sono aggiunte al formato del certificato.

Un'estensione può essere definita critica o non critica. Nel processo di verifica devono essere rifiutati i certificati che contengono un'estensione critica non riconosciuta; un'estensione non critica può essere ignorata se sconosciuta. Il differente trattamento è a carico di chi effettua la verifica: il Relying Party (RP). X.509v3 definisce 4 categorie di estensioni:

- key and policy information
- crtificate subject and certificate issuer attributes
- certificate path constraints
- CRL distribution points

Key and policy information Key usage: identifica lo spazio delle applicazioni pper il quale la chiave pubblica può essere usata, può essere critica o non critica, se è critica allora il certificato può essere usato solo per gli scopi la cui corrispondente opzione è definita.

Certificate subject and certificate issuer attributes Subject alternative name: consente di usare diversi formalismo per identificare il possessore del certificato, critica se il campo subject-name è vuoto.

Certificate path constraints Basic constraints: indica se il soggetto del certificato può agire da autorità di certificazione, è possibile definire la massima profondità dell'albero di certificazione, si consiglia di definirla come critica.

Name constraints: solo per CA, fissa lo spazio dei nomi certificabili da una CA, specifica whitelist/blacklist.

CRL distribution points Identifica il punto di distribuzione della CRL da usare come verifica della validità di un certificato.

Estensioni private È possibile definire estensioni private, comuni a una certa comunità di utenti. Tra queste, PKIX Authotity Information Access: indica come accedere a informazioni e servizi della CA che ha emesso il certificato.

3.2.2 OCSP

Standard IETF-PKIX per verificare in linea se un certificato è valido (non revocato), è un'alternativa alle CRL. Le risposte sono firmate dal server (non dalla CA) e il certificato del server non è verificabile con OCSP. Sono possibili risposte precalcolate, che diminuiscono carico sul server ma rendono possibili attacchi replay. Funzionamento:

1. A e B hanno dei certificati rilasciati dalla CA
2. A vuole connettersi con B e gli invia il certificato
3. B crea una richiesta OCSP che contiene il numero di serie del certificato e lo invia alla CA
4. il responder della CA controlla lo status del certificato nel suo database
5. se il certificato è valido, conferma la validità inviando una risposta a B
6. B verifica la risposta della CA utilizzando la chiave pubblica della CA
7. B si connette con A

Trusted Responder Il server OCSP firma le risposte con una coppia chiave/certificato indipendente dalla CA per la quale sta rispondendo. Il servizio è pagato dall'azienda o dagli utenti.

Delegated Responder Il server OCSP firma le risposte con una coppia chiave/certificato diversa in base alla CA per la quale sta rispondendo. Il servizio è pagato dalla CA.

3.3 Applicazioni relative all'uso e memorizzazione dei certificati

3.3.1 Timestamping

Prova della creazione dei dati prima di un certo istante di tempo. Regolata dalla TSA (Time-Stamping Authority). RFC-3161 descrive il protocollo di richiesta e il formato della prova. Funzionamento:

1. l'hash dei dati originali viene calcolato, a esso viene aggiunto il timestamp dato dal TSA e il suo risultato (hash A) viene calcolato
2. la firma del TSA viene verificata decifrandola con la chiave pubblica del TSA, producendo hash B
3. hash A e B vengono confrontati

3.3.2 PSE (Personal Security Environment)

Ogni utente dovrebbe proteggere la propria chiave privata e i certificati delle root CA fidate. Può essere sw o hw.

Smart-card crittografiche Carte a chip a memoria e con capacità crittografiche autonome. È semplice realizzare card in grado di svolgere crittografia simmetrica e complesso con quella asimmetrica. Dispongono di poca memoria.

HSM (HW Security Module) Acceleratore crittografico per server, ha memoria protetta e capacità crittografiche autonome.

API di sicurezza PKCS #11 descrive le API per creare e manipolare i token crittografici. Può essere utilizzato in hw e sw.

3.4 Formati per documenti elettronici sicuri

3.4.1 PKCS #7 e CMS

PKCS #7 è lo standard RSA per la busta sicura (posta elettronica), CMS è la sua evoluzione. Permette autenticazione. integrità, riservatezza dei dati con algoritmi simmetrici o asimmetrici. Permette più firme su uno stesso oggetto e può includere certificati per revocare la firma. È un formato ricorsivo. Gli algoritmi base sono:

- digest: MD5, SHA-1
- firma: RSA, DSA
- key management: DH, RSA, 3DES e PBKDF2
- cifratura contenuto: 3DES-CBC, RC2-CBC
- MAC: HMAC-SHA1

3.4.2 Struttura CMS

CMS racchiude i due campi content type e content in contentInfo.

Tipi di contentType

- data: codifica di una generica sequenza di byte
- signedData: dati e firme digitali parallele
- envelopedData: dati cifrati e chiave cifrata per i destinatari
- authenticatedData: dati, MAC e chiave cifrata per i destinatari
- digestedData: dati e digest
- encryptedData: dati cifrati con algoritmo simmetrico

3.4.3 PKCS #10 (Certificate Signing Request)

Descrive il formato per la richiesta di un certificato, (CSR). La richiesta contiene: DN, chiave pubblica e attributi.

3.4.4 PKCS # 12

Definisce un formato archivio per memorizzare oggetti crittografici in un solo file, consentendone il trasporto. Trasporta una chiave privata e uno o più certificati, quindi l'identità digitale di un utente.

3.4.5 Documenti firmati

Un documento firmato può:

1. essere avvolto nella firma
2. avvolgere la firma
3. avere una firma separata

É possibile avere firme multiple.

3.4.6 Electronic Signature (ES) Europea

Dati che sono logicamente associati con altri dati in formato elettronico che ne forniscono un mezzo di autenticazione. Una firma scannerizzata è una firma elettronica. È un concetto legale distinto dalle firme digitali.

AES (Advanced Electronic Signature) È una ES che soddisfa i seguenti requisiti:

- è in relazione univoca con il firmatario
- consente di identificare il firmatario
- è creata usando strumenti che il firmatario può mantenere sotto il suo controllo
- è in relazione con i dati ai quali si riferisce in modo che ogni successiva modifica dei dati possa essere individuata

3.4.7 Qualified Certificate (QC)

Certificato che garantisce l'identità di una persona. Contiene:

- l'indicazione che si tratta di un QC
- l'indicazione del certificatore e dello stato in cui è stato emesso
- indicazioni sulle limitazioni di utilizzo del certificato
- indicazioni sul limite delle transazioni commerciali effettuabili con quel certificato

Qualified Electronic Signature È una AES apposta usando QC e dispositivi di firma sicuri. Ha valore legale equivalente alla firma autografa.

3.4.8 Standard ETSI per firma elettronica

Si chiama CAdES (CMS Advanced Electronic Signature), è un formato di firma grezzo. Esistono anche XAdES e PAdES (per formati XML e PDF). ASiC (Associated Signature Containers) sono contenitori per associare documenti elettronici con firme detached.

4 Attacchi alle reti IP

IP non ha alcuna autenticazione degli indirizzi e i pacchetti non sono protetti. Sono quindi attaccabili tutti i protocolli che utilizzano IP come trasporto.

4.1 Protezione DHCP

DHCP è un protocollo non autenticato che sfrutta IP e ne eredita le debolezze, debole ad attacchi **shadow server** che opera al posto del vero DHCP server, l'attaccante può sferrare attacchi:

- **DoS:** l'host non può contattare la rete esterna, se configurato in modo errato dall'attaccante (l'attaccante fornisce un indirizzo IP non esistente);
- **MITM logici:** si fornisce come configurazione una subnet da 2 bit (eg. indirizzo da 32 bit avremo 30 bit di rete e 2 di subnet) in modo da avere 4 indirizzi possibili (00 [non assegnabile, identifica la rete], 01, 10, 11 [non assegnabile, indirizzo di broadcast]) ma solo 2 validi. Assegna uno dei due indrizzi all'host e l'altro lo tiene per se dicendo di essere il default gateway. L'attaccante può alterare tutte le comunicazioni della vittima.

Alcuni switch Cisco offrono:

- **DHCP snooping:** gli switch hanno delle porte "trusted" e accettanno traffico DHCP solo da queste porte considerate affidabili;
- **IP guard:** lo switch accetta solo pacchetti da host che hanno ottenuto indirizzi ottenuti da DHCP della rete stessa. Però è necessaria una lista di indirizzi IP considerati validi.

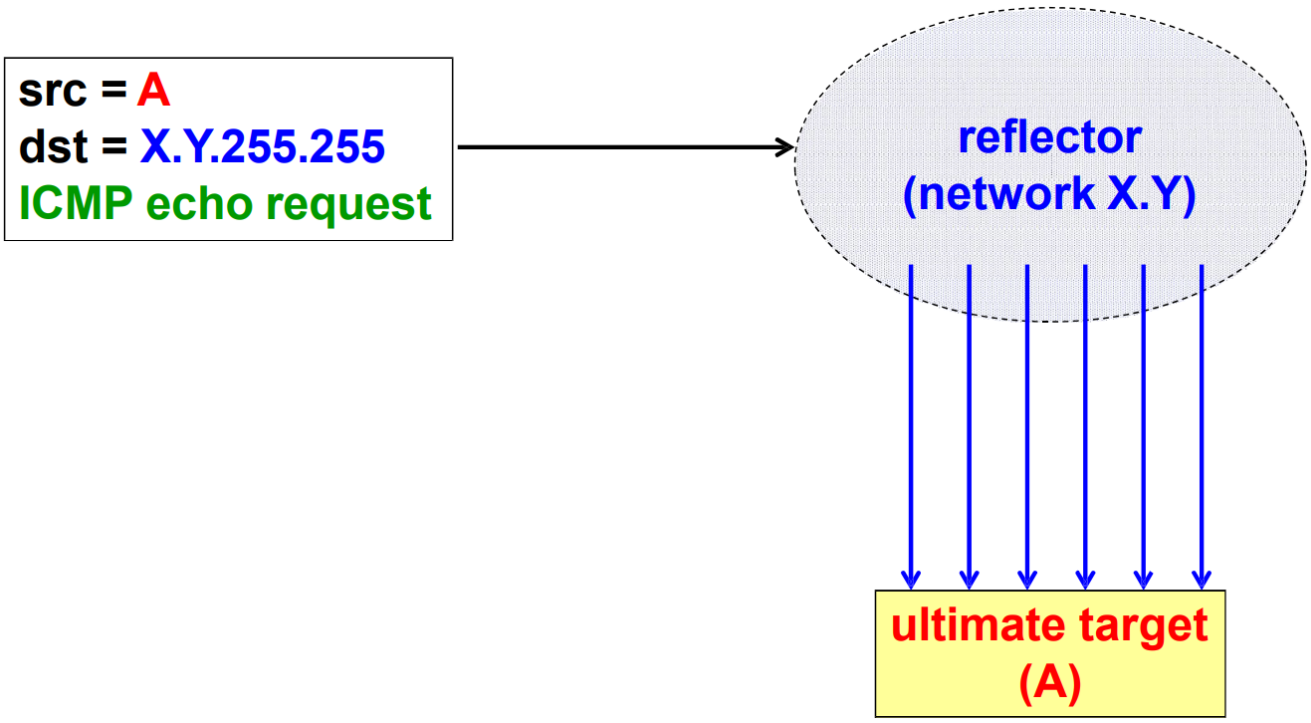
Si può anche usare **HMAC-MD5** per autenticare i messaggi con key-digest e autenticare la comunicazione, ma è scarsamente adottato perché è difficile da implementare su host dinamici.

4.2 Sicurezza ICMP

è un protocollo di servizio per reti a pacchetto che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete. Privo di autenticazione, soggetto a:

- **Smurfing:** attacco DDoS durante il quale un gran numero di pacchetti ICMP vengono inviati in broadcast su una rete. I dispositivi sulla rete rispondono all'IP sorgente, che l'attaccante ha designato come IP della vittima;

  L'attaccante crea una richiesta echo request con indirizzo di destinazione una rete broadcast detta reflector, che involontariamente



  - Per contrastare l'attacco, basta rifiutare il broadcast IP o identificare il responsabile con strumenti di network management.

- **Fraggle:** Come smurfing ma con pacchetti UDP, che è poco utilizzato quindi più probabile che non sia disattivato, su porte 7 e 19.

  - Per attacchi dall'esterno basta disattivare il broadcast, invece per quelli interni si può cercare di individuare l'attaccante tramite tecniche di network managemente.

4.3 ARP poisoning

ARP è utilizzato per scoprire l'indirizzo MAC di un nodo del quale si conosce l'indirizzo IP e si fa parte della stessa rete per comunicazioni a livello 2. Il richiedente manda un frame in broadcast, ARP request, per avere l'indirizzo MAC richiesto, risponderà solo, tramiete ARP reply, il dispositivo desiderato con il suo indirizzo. Il risultato è memorizzato nella tabella ARP.

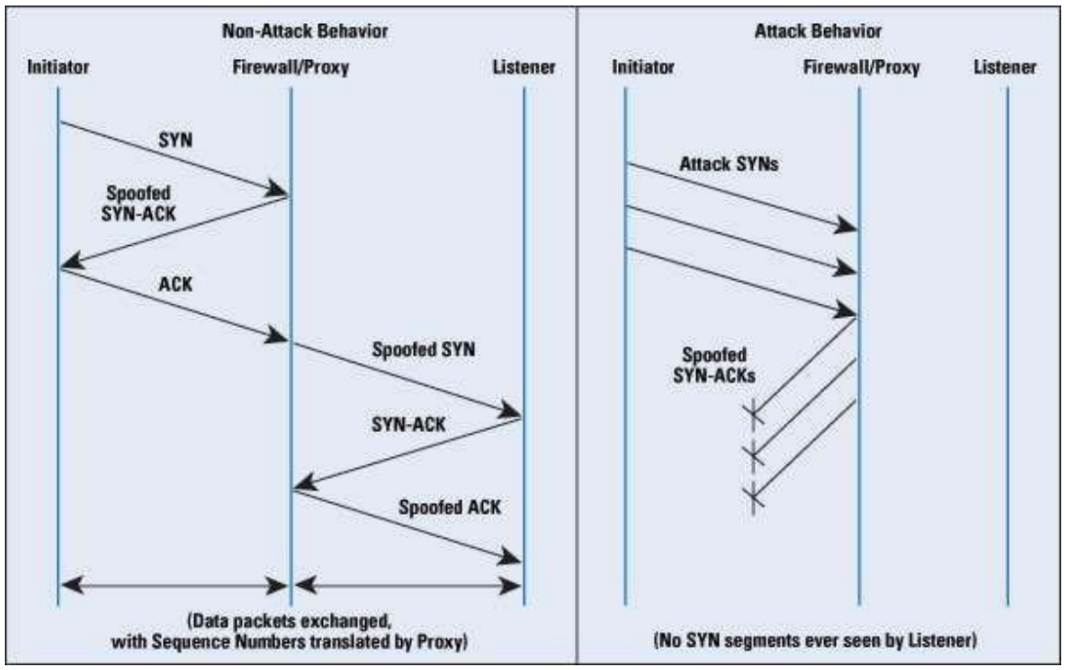
- **ARP poisoning:** si inseriscono dati falsi nella tabella per inviare dati ad altri dispositivi. Ad esempio cambiare l'indirizzo IP del default gateway per farlo collegare ad uno shadow server. Il problema è che gli host rispondono in automatico a tutte le richieste e sovrascrivono i dati della tabella, anche i dati inseriti manualmente.

4.4 TCP SYN flooding

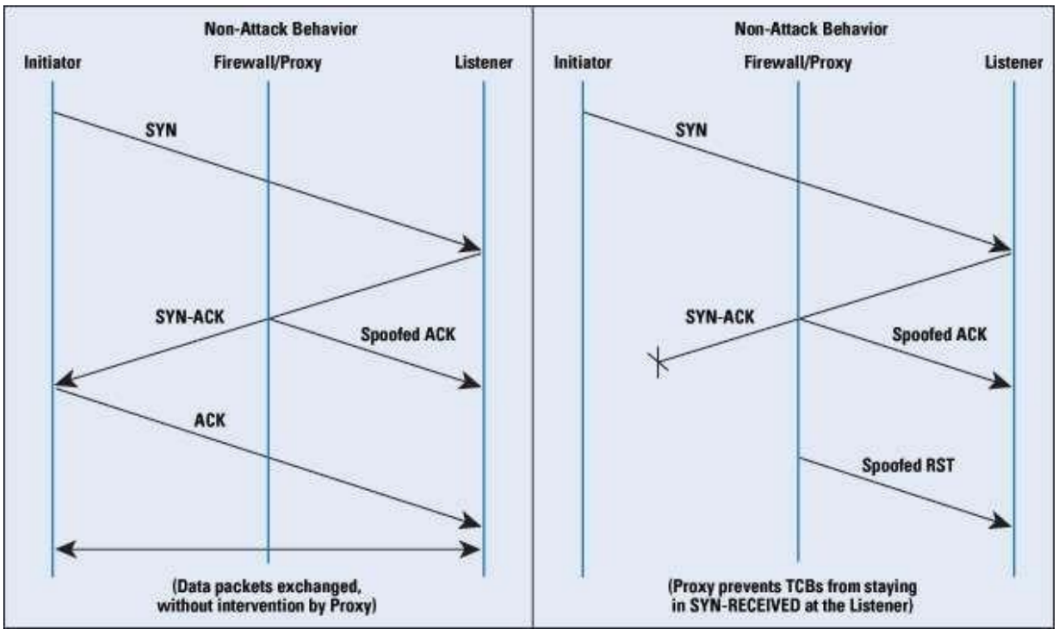
Nel **3-way handshake** quando il server riceve il SYN salva in una coda che è stato ricevuto un syn ma la connessione è pendente, e manda in dietro un SYNACK. Quando il server riceve l'ACK passa dalla coda delle connessioni pendenti ad una delle connessioni stabilite. L'attaccante può cercare di saturare la coda delle connessioni pendenti (SYN-receivend) facendo IP spoofing (IP falsi). La tabella dei SYN si riempie allora il server non può rispondere più. La tabella ha un timeout (solitamente 75 secondi) ma basta che l'attaccante continui ad inviare richieste.

Per difendersi:

- **abbassare il timeout**, rischiando di eliminare client validi ma lenti;
- **aumentare le dimensioni della tabella**, aggirabile con più richieste;
- **SYN interceptor**, usare un router come intercettatore tra client e server, questo fingerà di essere il server finale. La richiesta viene intercettata dal router e manda uno spoofed SYNACK al client con l'indirizzo reale del server. Se il client è benevolo manda l'ACK. Il router deve ripetere tutto il 3-way handshake impersonificando il client.



- **SYN monitor**, quando il server risponde con il SYNACK il router manda uno spoofed ACK al server, se il client è benevolo manda un ACK, al massimo verrà scartato come duplicato. Il router dopo aver mandato lo spoofed ACK ha fatto partire un timer, se non vede l'ACK reale manda un messaggio di spoofed reset al server che cancellerà la connessione TCP.



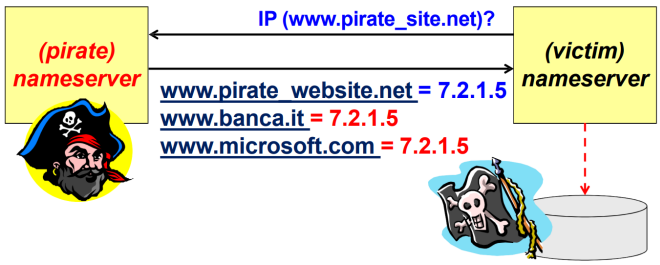
- **SYN cookie**, l'unica vera soluzione, usa il numero di sequenza del pacchetto SYN-ACK per trasmettere cookie al client e riconoscere così i client che hanno già inviato il SYN senza memorizzare nulla sul server. Il client dovrà riinviare il SYN cookie e il server lo accetterà senza memorizzare nulla nella tabella.

4.5 Sicurezza DNS

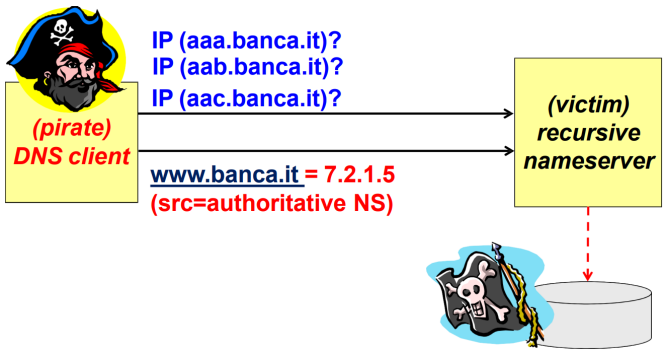
L'architettura del DNS è gerarchica. Si appoggia su UDP. Il client contatta prima il proprio name server locare, se non ha il dato inoltra la richiesta al root name server e così via fino a ricevere la risposta. Gli attacchi possibili sono:

- **DNS shadow server:** l'attaccante può fare sniffing per intercettare le DNS query e spoofing per generare risposte false. Nelle query potrebbe rispondere con indirizzi non esistenti **DoS**, oppure potrebbe dare un indirizzo IP fasullo con l'obiettivo di intercettare informazioni confidenziali.
- **DNS cache poisoning:**
  - Versione 1: si cerca di avvelenare un server generico, si cerca di convincere il client associato al NS di mandare una DNS query per un nome DNS per un server Web di un sito malevolo gestito dall'attaccante. L'attaccante ha pure instaurato uno shadow name server, quando vede la richiesta al suo server manda una DNS reply; il problema è che nella DNS query possono essere aggiunte delle associazioni non richieste dal client per altri siti. Il name server locale sovrascrive la cache e quando il client proverà ad accedere al sito vero la vittima contatterà in realtà il sito malevolo dell'attaccante.

    Se il client non richiede il sito malevolo il problema non si pone.



  - Versione 2: l'attacco parte dall'attaccante stesso che non ha bisogno di uno shadow server. L'attaccante usa un client e invia delle DNS query, manda al NS delle richieste per siti fasulli, per ognuna il DNS crea una trasaction-ID. L'attaccante manda al NS manda risposte fasulle con l'indirizzo IP che si vuole disallineare con l'indirizzo IP quello del sito malevolo ma con il nome del sito da attaccare. Per ogni richiesta manda un trasaction-ID diverso per cercare di indovinare quello giusto. Se la richiesta viene accettata il NS sovrascrive la sua cache.



- **DNF flash crowd:** attacco DoS che sommergono il NS di richieste.

4.6 DNSsec

Versione sicura di DNS.

Fornisce protezione end-to-end tramite firme digitali create dagli amministratori dei server DNS e verificate dal software di risoluzione del ricevente. Evita di doversi fidare dei NS intermedi.

Le query stesse non sono firmate, e non esiste una root CA, non fornisce siurezza nel dialogo tra DNS client e server.

DNSsec è scarsamente utilizzato perché è difficile da configurare per gestire la PKI.

4.7 Sicurezza del routing

Poca sicurezza nell'accesso sistemistico ai router per la gestione e nello scambio di tabelle di routing perché avviene con autenticazione su indirizzi IP. È possibile attivare protezione con keyed-digest però rimane il problema delle chiavi condivise.

4.8 Protezione IP spoofing

Per proteggere dagli impostori interni e esterni. Si utilizzano filtri presenti nel router, che scarta tutto il traffico dalla rete internet con indirizzo IP della rete interna oppure scarta un pacchetto della rete interna ma che ha come indirizzo IP non appartenente alla rete interna. Protezione solo parziale perché non ci difende da attacchi esterni o interni con indirizzi corretti.

5 VPN

Il livello rete è un buon compromesso su cui applicare la sicurezza. Possiamo fare sicurezza end-to-end, cioè rendere sicuro tutto il percorso tra mittente e destinatario. Se non ci fidiamo solo della rete pubblica possiamo applicare le tecniche di sicurezza attraverso le VPN.

È una tecnica hw o sw per realizzare una rete privata utilizzando canali e apparati di trasmissione non affidabili. La VPN è utilizzata nelle infrastrutture insicure e crea una connessione sicura tra gli host.

VPN tramite rete nascosta Indirizzamento non standard per non essere raggiungibili da altre reti. Quando l'indirizzo passa per la rete pubblica viene mascherato l'indirizzo IP. Aggirabile se qualcuno scopre gli indirizzi usati accedendo al NAT, vulnerabile a sniffing. Il traffico nella rete pubblica viaggia in chiaro, solo gli indirizzi sono mascherati.

VPN non sicura.

VPN mediante tunnel I router (VPN gateway) tra rete sicura e insicura, provvedono a incapsulare i pacchetti di rete all'interno di altri pacchetti in modo tale che i dispositivi della rete pubblica utilizzino gli indirizzi del pacchetto esterno. I router controllano l'accesso alle reti tramite ACL (Access Control List).

Se il pacchetto da trasmettere supera la massima dimensione consentita, deve essere frammentato.

I dati viaggiano in chiaro e il pacchetto IP può essere visto spacchettando il pacchetto.

VPN non sicura.

VPN tramite tunnel IP sicuro Prima di essere incapsulati i pacchetti di rete vengono protetti con MAC o key-digest (integrità e autenticazione), cifratura (riservatezza) e numerazione (contro replay). Il pacchetto interno è sicuro e come indirizzo IP viene utilizzato quello esterno. Anche detta **Secure VPN** (S-VPN).

Se le operazioni di crittografia sono svolte dagli end point si può considerare come sicurezza end-to-end

5.1 IPsec

Architettura IETF per fare sicurezza al livello 3, sia end-to-end che VPN di parte della rete, sia in IPv4 che in IPv6. Permette di creare VPN su reti non dfidate e di fare sicurezza end-to-end.

Definisce due formati particolari:

- **AH** (Authentication Header): per integrità, autenticazione, protezione da replay;
- **ESP** (Encapsulating Security Payload): per riservatezza.

Usa il protocollo **IKE** (Internet Key Exchange) per lo scambio delle chiavi.

5.1.1 IPsec Security Association

Connessione logica unidirezionale protetta tra due sistemi IPsec detta SA. Ogni SA viene associata ad una serie di algoritmi, paramenti e chiavi per garantire la protezione. Occorrono due SA per avere protezione completa in un canale bidirezionale. Si potrebbero adottare tenciche di protezione diverse per le due direzioni.

5.1.2 Database locali IPsec

L'associazione tra SA e sue caratteristiche vengono salvate in un database.

- **SPD** (Security Policy Database): contiene le security policy da applicare ai diversi tipi di comunicazione, configurato a priori oppure agganciato a un sistema automatico;

- **SAD** (SA Database): elenco delle SA attive e delle loro caratteristiche.

5.1.3 Funzionamento di IPsec

1. il modulo IPsec riceve pacchetto IP e controlla quale policy applicare consultando la SPD
2. ottiene le regole di sicurezza e crea/legge SA
3. ottiene algoritmi e parametri dalla SAD
4. il pacchetto IP è ora protetto da IPsec

IPsec in transport mode Usato per fare sicurezza end-to-end, ossia usato dagli host, non dai gateway. È computazionalmente leggero ma non protegge i campi variabili.

IPsec in tunnel mode Usato per fare VPN, solitamente dai gateway. Protegge i campi variabili ma ha una computazione più costosa.

5.1.4 AH

Fornisce integrità (tramite funzione di hash) e autenticazione dei dati, protezione, parziale, da replay attack (scartando pacchetti vecchi). Utilizza HMAC. Opera sopra IP.

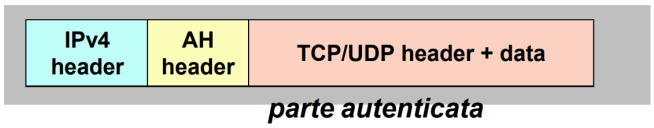


Figura 1: AH in trasport mode

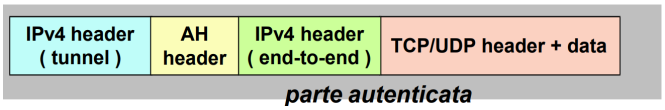
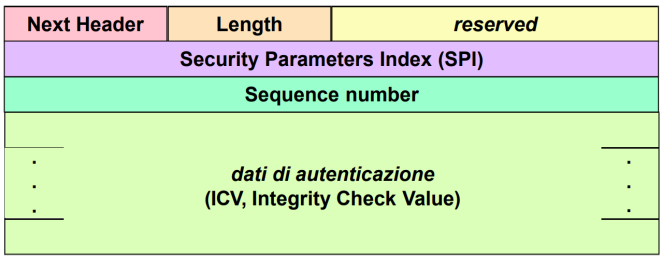


Figura 2: AH in tunnel mode



- Next header
- lenght: lunghezza complessiva, AH complessivo;
- SPI: numero rappresentativo della SA;
- Sequence numebr: protezione attacchi di tipo replay;
- dati autenticazione: MAC

Processamento di un pacchetto con AH

1. in parallelo, si estrae AH e si normalizza il pacchetto
2. si estrae ICV (dati di autenticazione) da AH
3. si estrae SPI (Security Parameters Index) da AH e lo si cerca nella SAD
4. si calcola il valore di autenticazione tramite algoritmi e parametri della SAD e pacchetto IP nornalizzato
5. si confrontano i valori ricevuti da ICV e quelli calcolati per vedere se corrispondono

Normalizzazione per AH

- azzerare campo TTL/hop limit
- se il pacchetto contiene Routing Header: fissare il campo di destinazione all'indirizzo del destinatario finale, fissare il contenuto del RH e Address Index al valore che avrà a destinazione
- azzerare tutte le operazioni che hanno bit C attivo

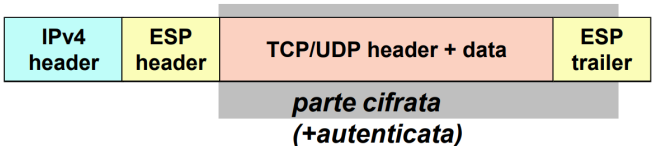
Funzionamento HMAC-SHA1-96

1. dato M normalizzarlo generando M'
2. allineare a 160 bit M' generando $M'p$
3. calcolare la base di autenticazione B=HMAC-SHA1$(K, M'p)$
4. ICV: 96 leftmost bit di B

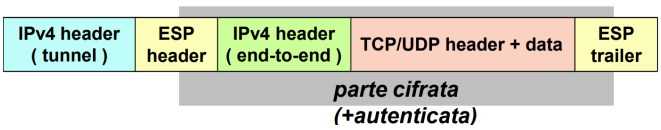
5.1.5 ESP

In origine poteva fornire solo condifenzialità. Usa come meccanismo base DES-CBC, fornisce integrità e autenticazione, riduce la dimensione del pacchetto e risparmia una SA. La sicurezza è offerta solo a ciò che segue l'header.

ESP in transport mode Usato dagli host, non dai gateway, non nasconde l'header.



ESP in tunnel mode Usato solitamente dai gateway, nasconde anche gli header.



Posso scegliere quali algoritmi applicare ai dati impostando a NULL gli algoritmi che non voglio applicare.

5.2 Protezione parziale da replay in IPsec

Comune ad AH e ESP.

1. quando si crea una SA, il mittente inizializza il sequence number a 0;
2. quando si invia un pacchetto, si incrementa il sequence number di 1;
3. quando si raggiunge il sequence number $2^{32} - 1$ si negozia una nuova SA per evitare una qualunque ripetizione del sequence number.

Nella pratica la protezione viene applicata ad una finestra mobile, sliding window tecnique, di 32 o 64 bit. Si sposta la finestra man mano che si riceve il primo pacchetto della finestra.

	Transport Mode	Tunnel Mode
AH	Autenticazione, integrità, no replay per payload+header del pacchetto originale	Autenticazione, integrità, no replay per payload+header del pacchetto originale ed header esterno
ESP	Autenticazione, integrità, no replay e riservatezza per payload del pacchetto originale	Autenticazione, integrità, no replay e riservatezza per payload+header del pacchetto originale
ESP + AH	Autenticazione, integrità, no replay per payload+header del pacchetto originale Riservatezza per payload del pacchetto originale	Autenticazione, integrità, no replay per payload+header del pacchetto originale ed header esterno Riservatezza per payload+header del pacchetto originale

5.2.1 IPsec v3

AH è opzionale, ESP è obbligatorio. Supporto per multicast da singola sorgente e AEAD. Possiede ESN (Extended Sequence Number).

5.3 Comparazione metodi di sicurezza

End-to-end security IPsec è adottato a livello host tramite transport-mode SA. Il pacchetto è completamente protetto su tutto il percorso. Vantaggi:

- È computazionalmente leggero;
- può prevenire attacchi provenienti dall'interno;
- ogni end system è incaricato di fare operazioni crittografiche per il proprio traffico.

Svantaggi:

- I dispositivi potrebbero non avere IPsec;
- Il traffico è cifrato anche nelle reti locali, ma nelle reti locali vengono utilizzati dei tool che controllano il traffico che se è cifrato non è ispezionabile.

Basic VPN Ipsec adottato a livello gateway di frontiera, gli host non fanno nulla e non sanno dell'esistenza della VPN, tunnel-mode SA. Protegge i campi variabili. Vantaggi:

- Gli host non devono dedicarsi ad operazioni crittografiche;
- Il traffico è sempre in chiaro nella LAN, possibile individuare pattern tramite EDS.
- I gateway sono dei colli di bottiglia;
- Il traffico nella LAN non è protetto.

End-to-end security con Basic VPN Utilizza entrambe le sopra citate. Gli end system creano una SA con trasport mode e i gateway utilizzano tunnel-mode. Vantaggi:

- Si può autenticare il traffico su tutto il percorso.
- Gli host devono supportare IPSec e fare operazioni crittografiche;
- I gateway potrebbero diventare dei colli di bottiglia.

Secure gateway Versione speciale della Basic VPN. Si protegge il percorso dall'end system remoto fino al gateway locale. Il primo host si connette direttamente alla WAN, il gateway che riceve il pacchetto indirizzato al secondo host è protetto da IPsec, solo tunnel-mode SA. Utilizza 2 indirizzi IP header esterno IP pubblico, IP interno per comunicare con la rete privata. L'end system deve supportare IPsec.

Secure remote access Come sopra, ma entrambi gli host utilizzano IPsec, quindi sia transport-mode che tunnel-mode SA.

5.3.1 IPsec key management

Componente fondamentale per IPsec, fornisce ai sistemi IPsec comunicanti le chiavi simmetriche necessarie per l'autenticazione e/o la cifratura dei pacchetti. Le chiavi vengono distribuite OOB o automaticamente.

ISAKMP Internet Security Association and Key Management Protocol, definisce le procedure necessarie per negoziare, stabilire, modificare e cancellare la SA. Non indica il metodo da usare per lo scambio delle chiavi. Lo scambio è realizzato solitamente dal protocollo **OAKLEY**.

IKE Internet Key Exchange, crea una SA per proteggere lo scambio ISAKMP. Con questa SA protegge la negoziazione della SA richiesta da IPsec, può essere riutilizzata più volte per negoziare altre SA IPsec. Modi di funzionamento:

- main mode: 6 messaggi, protegge l'identità delle parti
- aggressive mode: 3 messaggi, non protegge identità
- quick mode: 3 messaggi, negoziazione solo della SA IPsec

- new group mode: 2 messaggi

Metodi di autenticazione:

- digital signature: non repudiation della negoziazione IKE
- PKE: protezione dell'identità in aggressive mode
- revised PKE: meno costoso, solo 2 operazioni a chiave pubblica
- pre-shared key: l'ID della contropart può essere solo il suo indirizzo IP

5.3.2 VPN concentrator

Apparecchiature special-purpose che fungono da terminatori di tunnel IPsec per alleggerire il peso computazione degli host. Per accesso remoto di singoli client oppure per creare VPN site-to-site. Prestazioni elevate.

5.4 Prestazioni IPsec

- router: CPU potente o acceleratore crittografico, non gestito in outsourcing
- firewall: CPU potente
- VPN concentrator: massima indipendenza dalle altre misure di sicurezza

IPsec riduce il throughput di rete, perché i pacchetti hanno dimensione maggiore e si ha un maggior numero di pacchetti.

IPsec è applicabile solo a pacchetti unicast, tra parti che hanno attivato la SA tramite chiavi condivise o certificati quindi in gruppi chiusi.

6 Firewall e IDS/IPS

L'obiettivo dei firewall è quello di fornire controllo di sicurezza per il transito dei dati tra due reti. Una di queste è considerata trusted e una untrusted. E' necessario regolare il traffico in entrambe le direzioni. I firewall si possono utilizzare anche per limitare l'accesso ad internet.

6.1 Firewall

- **Ingress firewall**: collegamenti incoming, tipicamente per selezionare i servizi offerti dall'esterno;
- **Egress firewall**: collegamenti outgoing, controllo dell'attività del personale (limitiamo i servizi accessibili dalla rete).

Un firewall è composto da più prodotti, non un solo dispositivo hw.

Un firewall deve garantire un buon equilibrio tra sicurezza, funzionalità (quanto è semplice configurare, utilizzare e gestire il firewall) e costo. Nella maggior parte dei casi si opta ad avere una sicurezza elevata ma non troppo per non inficiare troppo sulla funzionalità.

Principi inderogabili (livello minimo di sicurezza):

1. il FW deve essere l'unico punto di contatto della rete interna (trusted) con quella esterna (untrusted);
2. solo il traffico autorizzato può attraversare il FW;

   Solitamente i firewall hanno delle condizioni per determinare cosa fare nei vari casi. Nel caso in cui non ci sia riscontro con le regole specifiche si una la default action.
3. il FW deve essere un sistema altamente sicuro esso stesso.

6.1.1 Politiche di autorizzazione

Potremmo avere due casi:

1. Tutte regole di allow e la default action di deny;
2. Tutte regole di deny e la default action di allow.

Whitelisting Tutto ciò che non è espressamente permesso è vietato. Fornisce maggiore sicurezza ma è più difficile da gestire perché bisogna sapere tutti i servizi da autorizzare. Dafault action = deny.

Blacklisting Tutto ciò che non è espressamente vietato è permesso. Minore sicurezza rispetto al Whitelisting, più facile da gestire. Default action = allow.

6.2 Tecnologie di FW

6.2.1 Packet filter

Storicamente disponibile sui router, effettua controlli sui singoli pacchetti IP a **livello rete**.

Del pacchetto controlla 5 campi (quintupla IP):

1. IP sorgente;
2. IP destinazione;

3. Protocollo (TCP/UDP);
4. Porta sorgente;
5. Porta destinazione.

Pro:

- basso costo (disponibile su tutti i router e OS);
- ottima scalabilità e prestazioni perché siamo ad un livello basso (indipendente dalle applicazioni).

Contro:

- controlli poco precisi, quindi più facile da ingannare (IP spoofing, pacchetti frammentati);
- arduo supportare servizi con porte allocate dinamicamente, perché dovrei cambiare ad ogni variazione le regole del FW (eg. FTP);
- configurazione complessa;
- difficile fare autenticazione degli utenti, al più può identificarli tramite l'indirizzo IP, ma è molto debole come autenticazione.

6.2.2 Stateful packet filter

Simile al packet filter ma state-aware. Riceve informazioni di stato dal livello trasporto e quello applicativo, distingue le nuove connessioni da quelle già aperte. Accetta servizi solo se la richiesta parte da un dispositivo della rete interna che richiede un servizio alla rete esterna. Si salvano gli stati e quando arriva un pacchetto dall'esterno si controlla se è l'inverso di quello in uscita e lo accetta senza passare per le politiche di autorizzazione.

Pro:

- Migliori prestazioni rispetto al packet filter.

Contro:

- Molte delle limitazioni proprie del packet filter.

6.2.3 Circuit-level gateway

FW non application-aware. Crea un circuito tra client e server a **livello trasporto**, ma non comprende i dati in trasporto, si limita a copiare tra le sue interfacce i segmenti TCP o i datagrammi UDP (se rispettano le regole di controllo accessi), deve riassemblare pacchetti IP quindi protegge da alcuni attacchi L3/L4. Controlla se la connessione TCP o UDP è avvenuta con successo, se sì permetterà tutto il traffico per quella interfaccia.

Per UDP controlla imposta un timer e controlla se riceve una risposta autorizza tutto il traffico, per TCP controlla se il 3-way handshake è avvenuto con successo.

Rompe il modello client server per una specifica connessione perché si interpone tra i due: **Pro**:

- i server sono più protetti: isola da attacchi che riguardano handshake TCP e frammentazione pacchetti IP;
- può autenticare il client (con modifiche alle applicazioni, quasi mai utilizzato).

Contro:

- Rimangono limitazioni proprie al packet filter.

Il più famoso è **SOCKS**.

6.2.4 Application-level gateway

Composto da una serie di proxy che esaminano il contenuto dei pacchetti a **livello applicativo**. Ogni proxy è dedicato ad un singolo applicativo, ogni proxy vedrà tutto del pacchetto del suo protocollo. Può opzionalmente effettuare il mascheramento o rinumerazione degli indirizzi IP interni e lavorare come se fosse un NAT, normalmente ha funzioni di autenticazione.

Pro:

- regole più granulari e semplici (basta inserire l'url di un sito per bloccarlo) rispetto a packet filter;
- server più protetti;
- può autenticare il client (eg. possiamo vedere le password contenute nel pacchetto);
- massima sicurezza tra tutte le tipologie di FW.

Contro:

- ritardo nel supporto per le nuove applicazioni (bisogna sviluppare il proxy per ogni nuova applicazione);
- consuma più risorse ed è più lento (ogni proxy è un processo in user mode);
- mancanza di trasparenza per i client;
- può esporre il SO del FW ad attacchi (perché arriviamo in cima allo stack protocollare).

Varianti:

- trasparent proxy: meno intrusivo per i client, più complesso
- strong application proxy: solo comandi/dati sono trasmessi

6.2.5 HTTP (forward) proxy

Server HTTP che fa solo da front-end e poi passa le richieste a un server esterno.

Pro:

- cache delle pagine esterne per tutti gli utenti interni;
- autenticazione e autorizzazione degli utenti interni;
- possibili vari controlli (visibilità completa sui messaggi HTTP).

6.2.6 HTTP reverse proxy

Server HTTP che fa solo da front-end e poi passa le richieste a un server interno.

Pro:

- obfuscation del server: nasconde le caratteristiche del server (eg. il client non sa quale OS gira sul server);
- acceleratore SSL: mantenere cifrato il traffico tra proxy e rete esterna e decifrarlo per inoltrarlo al server interno;
- load balancer: distribuire il carico tra più repliche del server interno;
- web accelerator: cache di contenuti statici;
- compressione;
- spoon feeding: riceve dal server tutta una pagina creata dinamicamente e la serve poco alla volta al client.

6.2.7 WAF (Web Application Firewall)

Modulo installato su proxy per filtrare il traffico applicativo, utilizza HTTP. Può decidere se un messaggio HTTP è autorizzato in base agli header, il contenuto o i comandi contenuti.

6.2.8 Local/Personal Firewall

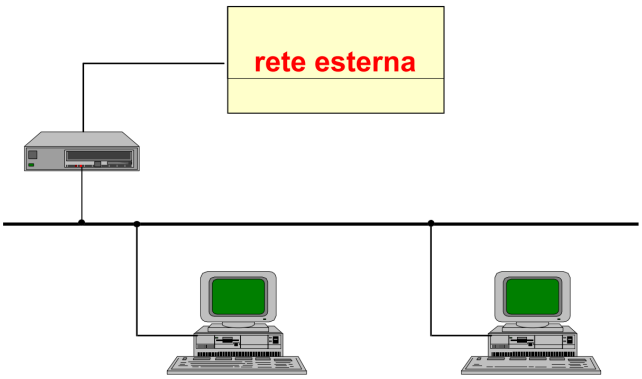
Firewall installato direttamente sul nodo da difendere. È tipicamente un packet filter.

Rispetto a un normale FW in rete può controllare i processi a cui è permesso aprire collegamenti verso altri nodi e ricevere richieste di collegamento. Utili come antivirus per impedire la diffusione dei virus.

6.3 Architetture di FW

6.3.1 Screening router (choke)

Router che filtra il traffico di rete, sia su livello IP che superiore. È economico ma insicuro.

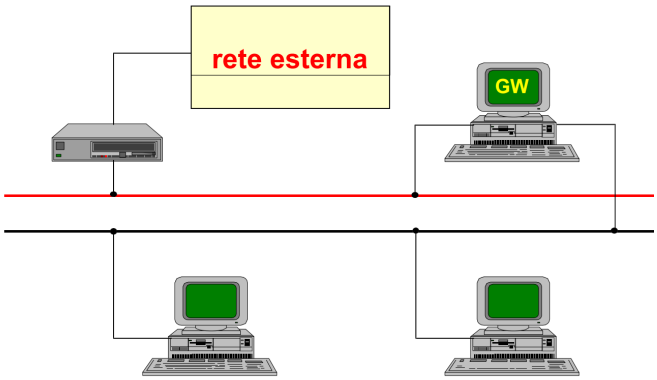


6.3.2 Application gateway (proxy)

Servizio che svolge il lavoro per conto di un applicativo, tipicamente con controllo d'accesso.

6.3.3 Dual-homed gateway

Sistema con due connessioni di rete e routing disabilitato. Facile da realizzare, è possibile mascherare la reten interna, ma richiede più hardware rispetto a screening router ed è poco flessibile e grosso sovraccarico di lavoro.



6.3.4 Screened host gateway

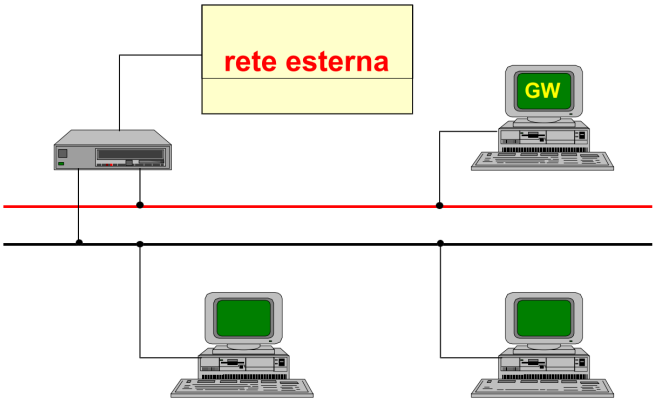
Il router blocca traffico da interno a esterno e viceversa tranne se arriva dal o al bastion. Il bastion host ospita circuit o application gateway per controllare i servizi autorizzati.

Pro:

- maggiore flessibilità.

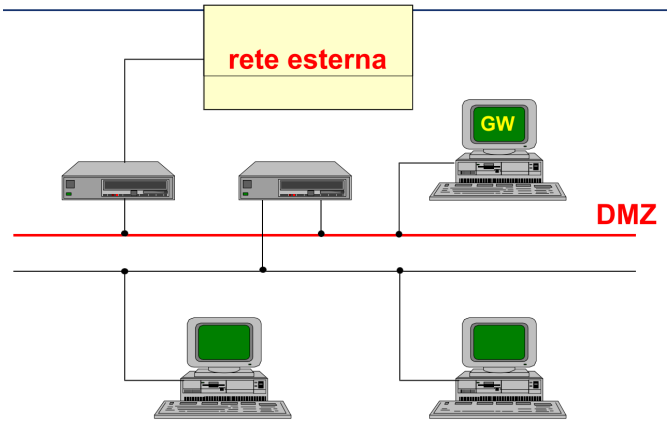
Contro:

- si possono mascherare solo gli host/protocolli che passano dal bastion;
- è più costoso e complesso da gestire;
- ricompare il problema del sigle-point-of-failure.

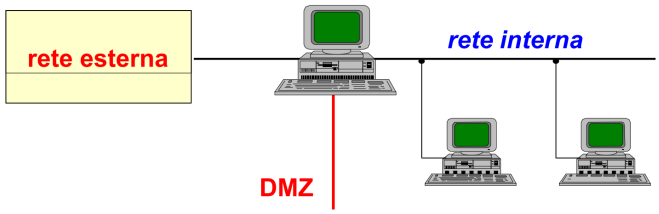


6.3.5 Screened subnet

Uno o più screening router sono utilizzati come firewall per definire tre diverse sottoreti: un router esterno separa la rete esterna dalla rete di perimetro, e un router interno separa la rete di perimetro da quella interna. La rete di perimetro è anche detta DMZ (demilitarized zone), e ospita i server che sono accessibili sia dalla rete interna che da quella esterna. Si può configurare il routing in modo che la rete interna sia sconosciuta. E' possibile collegare i server della rete interna, pubblici, alla DMZ, in modo da avere un altro livello di protezione in caso di attacchi.



Versione 2 screened subnet Per motivi di costo e di semplicità di gestione spesso si omettono i router, incorporando le loro funzioni nel gateway con 3 interfacce, rete esterna, rete interna e DMZ. Anche noto come **FW a tre gambe**. Ricompare il single-point-of-failure.



6.4 IDS e IPS

6.4.1 IDS (Intrusion Detection System)

È un sistema per identificare individui che utilizzano un computer o una rete senza autorizzazione, esteso anche all'identificazione di utenti autorizzati ma che violano i loro privilegi. Si basa sul fatto che il pattern di comportamento degli utenti non autorizzati si differenzia da quello degli utenti autorizzati.

Gli **IDS passivi** richiedono una configurazione proattiva dell'utente che inseriscono i pattern attesi, utilizzano checksum crittografici e riconoscimento di pattern, invece **IDS attivi** che riconoscono i pattern autonomamente, all'inizio fanno learning (analisi statistica del funzionamento del sistema e analizza la rete), dopo aver ottenuto i pattern locali passa in una fase di monitoring (analisi attiva di traffico dati, sequenze, azioni e confronta con i pattern) e quando riscontra una discrepanza reaction (confronto con paramertri statistici). La fase di learning può continuare durante l'utilizzo per continuare a studiare la rete ed identificare nuovi pattern. Attivi perché evolvono la loro configurazione nel tempo.

Caratteristiche topologiche:

- **HIDS** (Host-Based IDS): analisi dei log e attivazione di strumenti di monitoraggio interni al SO;
- **NIDS** (Network-Based IDS): attivazione di strumenti di monitoraggio del traffico di rete.

6.4.2 SIV E LFM

Esempi di HIDS.

- SIV: controlla i file di un nodo per rilevarne cambiamenti, ad esempio file di registro;
- LFM: controlla i file dei log e rileva pattern d'attacco.

6.4.3 NIDS

Componenti:

- sensor: controlla traffico e log, attiva i security event rilevanti e interagisce con il sistema, collocati in posizioni diverse della rete;
- director: coordina i sensor e gestisce il security database (conservare gli alert dei sensor e decide se comunicarli all'amministratore);
- IDS message system: consente la comunicazione sicura e affidabile tra i componenti dell'IDS.

6.4.4 IPS (Intrusion Prevention System)

Per velocizzare e automatizzare la risposta alle intrusioni, è IDS e FW dinamico distribuito. Non un prodotto ma una tecnologia che si appoggia su prodotti già esistenti.

6.4.5 Honey pot

Se rilevo un attacco ridireziono il traffico in una rete specifica, **Decoy DMZ**, per analizzare il tipo di attacco e studiarlo e prevenirlo sulla rete ufficiale.

7 Access control e audit

7.1 Access control

Controllo di accesso: processo di concessione o negazione di richieste specifiche per ottenere e utilizzare informazioni e i relativi servizi di elaborazione delle informazioni, e accedere a strutture specifiche. O anche processo mediante il quale l'uso delle risorse è regolato in base a una politica di sicurezza ed è consentito solo da entità autorizzate in base a tale politica.

Un meccanismo di controllo dell'accesso avviene tra un utente e le risorse di sistema. Il sistema deve prima autenticare un'entità in cerca d'accesso.

L'access control dovrebbe permettere di specificare: chi è autorizzato, a che risorse, per quanto tempo, in quali giorni, con che modalità, eseguendo quale operazione.

7.1.1 Subjects, objects, actions

- un soggetto è un'entità in grado di accedere agli oggetti;
- un oggetto è una risorsa a cui è controllato l'accesso;
- un'azione descrive il modo e operazione con cui un soggetto può accedere a un oggetto.

Subjects Generalmente, il concetto di soggetto si identifica con quello di processo. Qualsiasi utente o applicazione ottiene effettivamente l'accesso a un oggetto tramite un processo che rappresenta tale utente o applicazione. Il processo assume gli attributi dell'utente, come i diritti d'accesso. Un soggetto è in genere ritenuto responsabile delle azioni che ha avviato e lascia una traccia di controllo.

Objects Entità utilizzata per contenere, ricevere e trasmettere informazioni. Ad esempio: record, file, pagine, directory, ma anche bit, porte di comunicazione, ecc...

Il numero e tipo di oggetti da proteggere da un sistema di controllo degli accessi dipende dall'ambiente in cui opera e dal compromesso desiderato tra sicurezza e complessità.

Azioni Azioni tipo: read, write, execute, delete, create, search.

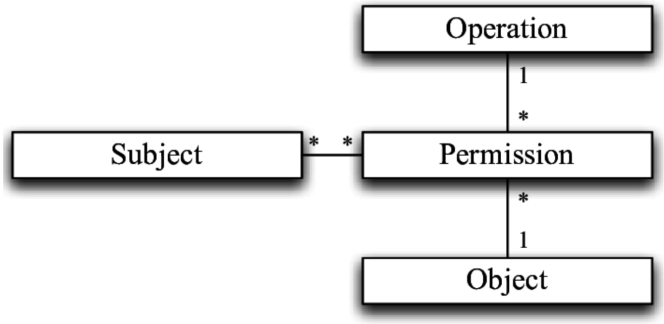
7.1.2 Strategie di access control

DAC (Discretionary Access Control) Controlla l'accesso in base all'identità del richiedente e alle regole d'accesso (autorizzazioni) che indicano ciò che i richiedenti sono o non sono autorizzati a fare. È detta **discrezionale** perché un'entità potrebbe avere diritti d'accesso che consentono all'entità, di sua spontanea volontà, di consentire a un'altra entità di accedere a qualche risorsa.

Il proprietario del sistema può accedere liberamente alle risorse, e stabilisce quali utenti e gruppi possono accedervi.

Contro:

- poco efficiente, perché bisogna specificare i permessi per tutti gli utenti separatamente.

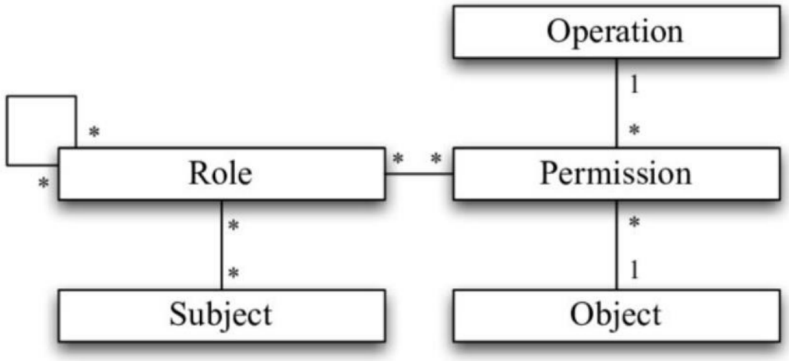


Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

MAC (Mandatory Access Control) Controlla l'accesso in base al confronto delle etichette di sicurezza, che indicano quanto sono sensibili o critiche le risorse di sistema, con le autorizzazioni di sicurezza, che indicano che le entità di sistema sono idonee ad accedere a determinate risorse. È detta **obbligatoria** perché un'entità che dispone di autorizzazione per accedere a una risorsa non può, di sua spontanea volontà, consentire a un'altra entità di accedere a quella risorsa.

RBAC (Role-Based Access Control) Controlla l'accesso in base ai ruoli che gli utenti hanno all'interno del sistema e sulle regole che stabiliscono a quali accessi sono consentiti da utenti in determinati ruoli. Nella politica l'azione viene concessa o negata ad un gruppo di soggetti associati ad un ruolo.

Più efficiente di DAC, la dimensione della access matrix è in base ai ruoli e non agli utenti.



ABAC (Attribute-Based Access Control) Controlla l'accesso in base agli attributi dell'utente, la risorsa a cui accedere e le condizioni dell'ambiente attuale.

Tecnica più complessa ma più flessibile, estrema granularità dovuta alle condizioni che mettono in relazione le azioni e gli oggetti.

7.2 Audit

Il system audit è un'attività ordinaria per valutare le prestazioni del sistema, i controlli di sicurezza, ecc...

Il monitoring è un'attività ordinaria che tiene traccia di tutte le attività eseguite sul sistema, come il rilevamento delle intrusioni e altre.

Auding e monitoring sono le capacità di osservare gli eventi, comprendere le prestazioni e mantenere l'integrità del sistema. Queste operazioni vengono spesso eseguite registrando gli eventi nel file di registro ed esaminando tali file in un secondo momento (logging).

7.2.1 Internal e external audit

Internal audit Viene eseguito dai revisiori all'interno dell'organizzazione. Scopi: identificare i rischi ogni volta che si presentano in relazione a problemi di prestazioni, sicurezza e conformità. Inoltre tengono d'occhio ciò che viene fatto per mitigare questi problemi con l'obiettivo di aiutare le organizzazioni a funzionare meglio.

External audit Svolto da revisori all'esterno dell'organizzazione. Sono enti indipendenti, spesso contabili pubblici certificati.

Internal Audit	*External Audit*
It is performed by auditors who are employees of the organization.	It is performed by the external professional auditing body.
Auditors generally have much wider authorizations.	Auditors generally have restricted authorizations.
The objective is to identify loopholes in processes for betterment of the organization.	External auditing is done by the organization to build confidence among clients and shareholders.
Audit report is not published outside, it is used only for internal purpose.	Audit report is published outside of the organization.
It can be executed anytime. Generally, it is done on regular basis.	It does not happen too frequently. Generally it occurs once in a year.

8 Tecniche e sistemi di autenticazione

8.1 Autenticazione

RFC-4949 È il processo di verificare l'affermazione che un sistema o una risorsa abbia un determinato valore. Affermazione è coppia attrivuto valore, come username + password.

whatis.com Il processo di determinare che qualcuno o qualcosa sia chi o cosa dica di essere. Coinvolge sia utenti che processi/dispositivi.

NIST IR 7298 Verificare l'identità di un utente, processo, o dispositivo, spesso come prerequisito per accedere a risorse o informazioni in un sistema informativo.

Si autorizzano attori, come esseri umani, software o hardware (authN). Autorizzazione (cosa posso fare) != autenticazione.

8.1.1 Fattori di autenticazione

Conoscenza Qualcosa che solo l'utente conosce. Rischio: memorizzazione e dimostrazione/trasmissione.

Possesso Qualcosa che l'utente possiede. Rischi: l'autenticatore, per furto, clonazione, uso non autorizzato.

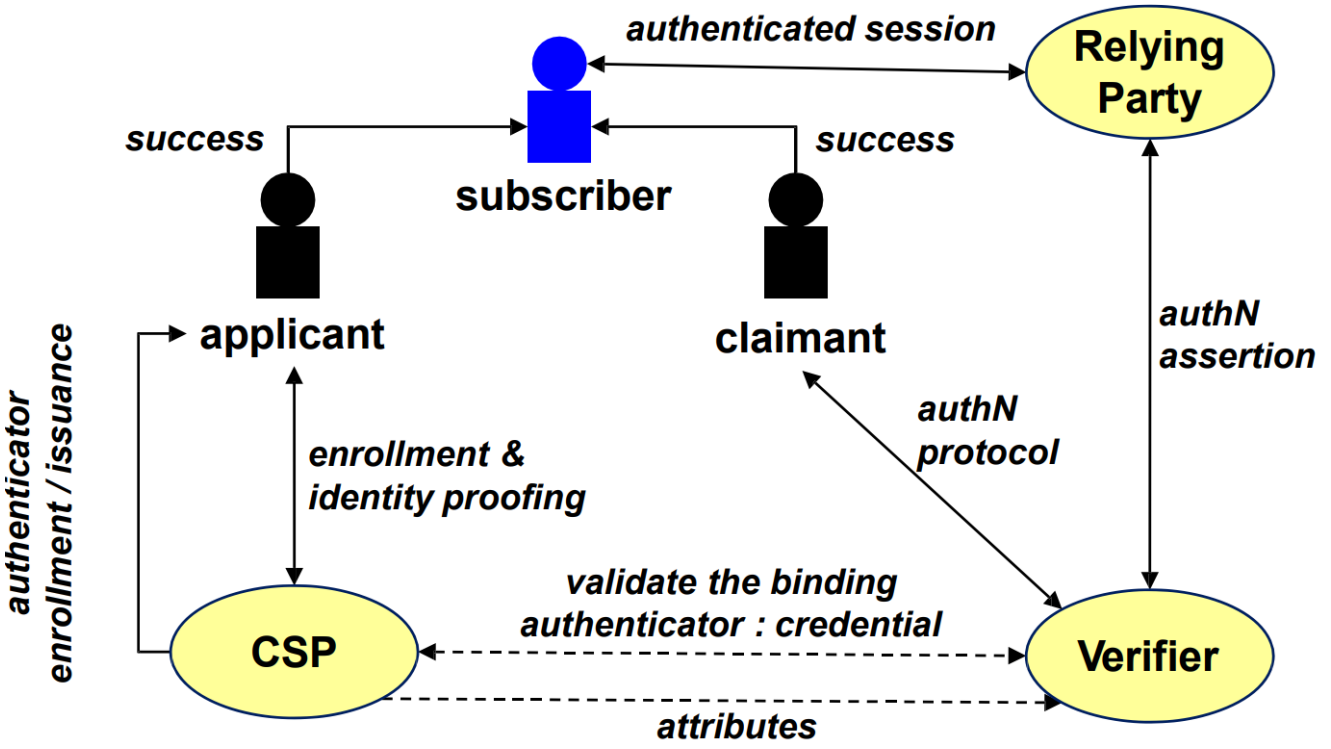
Essenza Qualcosa che l'utente è. Rischi: contraffazione e privacy, non può essere rimpiazzato quando compromesso.

8.1.2 Autenticazione digitale

Entità logiche:

- CSP (Credential Service Provider): emette o registra le credenziali utente, verifica e memorizza attributi associati
- verifier: esegue un protocollo di autenticazione per verificare il possesso di un autenticatore e credenziali
- RP (Relying Party): richiede e riceve un'asserzione di autenticazione dal verifier per certificare l'identità utente

I ruoli possono essere distinti o combinati in una sola entità.



Inizialmente l'attore è un applicant, ossia un'entità che non si è mai registrato al servizio a cui vuole accedere. In questa fase, l'applicant interagisce con il CSP, che registra l'applicant. Per registrarsi, l'applicant deve presentare le proprie credenziali. Il CSP valida le credenziali e le memorizza. A questo punto, l'applicant diventa claimant presso quel servizio (servizio detto relying party). L'asserzione di autenticazione non è fatta direttamente al RP, ma al verifier, che dialoga con il claimant tramite un protocollo di autenticazione. Quando il verifier riceve informazioni dia utenticazione dal claimant, e le utilizza per dialogare con il CSP. Il CSP, avendo memorizzato in fase di registrazione gli attributi dell'applicant, passa questi attributi al verifier che li userà per autenticare il claimant al RP. Se l'autenticazione avviene con successo, il verifier manda un'asserzione al RP e il claimant diventa subscriber.

Passi per autenticazione:

1. il verifier chiede autenticazione
2. il claimant invia il proprio identificativo utente
3. il verifier richiede una prova
4. il claimant risponde inviando un segreto che solo lui può conoscere; il segreto non è trasmesso direttamente, il claimant applica una funzione F che restituisce la prova della sua asserzione di autenticazione
5. il verifier, che ha salvato il segreto, applica al segreto un'altra funzione f e ottiene un altro risultato, che viene confrontato con la prova ricevuta per stabilire se l'autenticazione ha avuto successo

8.2 Autenticazione basata su password ripetibili

Il segreto è la password dell'utente.

Il client genera e trasmette la prova, il server la memorizza e la valida. Due casi:

- il server conserva le password in chiaro, il controllo d'accesso è verificare se ciò che è stato inserito e la password corrispondono

- il server conserva i digest delle password, il controllo d'accesso è verificare se ciò che è stato trasmesso e l'hash corrispondono

Pro: semplice per l'utente, a condizione che ne abbia una sola.

Contro: conservazione delle password, password indovinabili e validazione lato server.

Altri problemi includono: pssword sniffing, attacchi ai database delle password, riuso password, attacchi MITM e cryptography ageing.

8.2.1 Attacchi

Attacco dizionario Se l'algoritmo di hash è noto, e gli hash delle password sono noti, allora è possibile fare un attacco dizionario, comparando hash noti con quelli delle password cifrate.

Rainbow table Tecnica di trade-off per memorizzare e cercare una tabella. Utilizza una funzione di riduzione ($r : h \Rightarrow p$). Ogni riga rappresenta più password.

Per generare la tabella, scelgo un set di password iniziali dell'insieme di password P, calcolo catene di hashing (alternando funzione di riduzione e funzione di hashing) di lunghezza k per ognuna, e conservo solo l'ultima password di ogni catena. Dato un valore hash h da invertire, calcolo una catena cche inizia da h applicando la funzione di riduzione e poi quella di hash alternando. Se a un certo punto un valore corrisponde a uno dei valori finali della tabella, il valore iniziale permette di ricreare la catena completa.

Per proteggere le password da questi attacchi si usa il sale, valore diverso per ciascun utente, random e lungo.

8.2.2 Autenticazione forte

Definizione ECB (internet banking) L'autenticazione forte è basata sull'uso di due o più fattori di autenticazione. Gli elementi selezionati devono essere indipendenti. Almeno un elemento dovrebbe essere non riutilizzabile e non replicabile. La procedura di autenticazione deve proteggere la riservatezza dei dati di autenticazione.

Definizione PCI-DSS (carte di credito) Richiesta autenticazione a più fattori (diversi) per accessi al Cardholder Data Environment(CDE). Obbligatorio per accesso da reti untrusted o per gli amministratori.

Altre definizioni

- un protocollo crittografico a sfida (Handbook of Applied Cryptography)
- una tecnica resistente a un ben definito insieme di attacchi

8.3 Sistemi di autenticazione a sfida

8.3.1 Challenge-response authentication (CRA)

Una sfida viene inviata al claimant (utente che si vuole autenticare), che risponde con una soluzione calcolata usando un qualche segreto e la sfida. Il verifier confronta la risposta con la soluzione calcolata tramite un segreto correlato al claimant.

La sfida deve essere non ripetibile per evitare attacchi replay. Di solito la sfida è un nonce (numero arbitrario utilizzabile una volta sola) random. La funzione che calcola la soluzione deve essere non invertibile, altrimenti un ascoltatore sul canala può registrare il traffico e calcolare il segreto del claimant.

8.3.2 Sistemi a sfida simmetrici

Una sfida viene inviata al claimant, che risponde con il risultato di un calcolo che coinvolge il segreto condiviso con il verifier. Il verifier effettua lo stesso calcolo e confronta il suo risultato con la risposta. Il segreto è condiviso tra verifier e claimant. Spesso la funzione usata per il calcolo della risposta è una funzione di hash, perché più veloce.

Il segreto deve essere noto in chiaro al verifier, quindi è possibile fare attacchi contro la tabella degli id del verifier. SCRAM (Salted CRA Mechanism) risolve questo problema usando hash di password sul verifier.

8.3.3 Mutua autenticazione con protocolli a sfida simmetrici

Supponiamo che A e B vogliano autenticarsi mutualmente. L'identità è dichiarata esplicitamente solo da chi inizia lo scambio.

1. A si pressenta a B, esplicitando la sua identità
2. per autenticare A, B invia ad A una sfida S_b (la sfida è in chiaro)
3. A utilizza una funzione f di cifratura o hashing per rispondere alla sfida; nel farlo, utilizza un segreto condiviso con B, in questo caso una chiave simmetrica K_{ab}; nel frattempo, A invia anche la propria sfida, S_a
4. B verifica localmente, grazie alla risposta ricevuta, se l'autenticazione ha avuto successo
5. B risponde alla sfida S_a utilizzando la chiave K_{ab}

Alcuni di questi passaggi possono essere accorpati.

1. A invia la richiesta e la sfida allo stesso momento
2. B risponde inviando la sua sfida e la risposta
3. A risponde con la risposta alla sfida di B

Questo metodi di autenticazione non sono robusti perché vulnerabili ad attacchi.

1. un attaccante M, senza conoscere la chiave simmetrica, può comunque fingere di essere uno delle due entità che si vogliono autenticare
2. M si presenta come A, inviando una sfida a B
3. B risponde alla sfida cifrando la risposta con la propria chiave
4. M riceve le informazioni e scarta la parte cifrata, catturando la sfida C_b (è in chiaro, solo la risposta è cifrata)
5. M appre una seconda connesione verso B, trasmettendo la sfida ricevuta da B come sfida
6. B risponde alla sfida con un'altra sfida, cifrando la risposta con la stessa chiave
7. M riceve la risposta e la ritorna a B

In poche parole, utilizzo l'altra parte per generare la risposta a una sfida anche se non conosco la chiave di cifratura.

8.3.4 Sistemi a sfida asimmetrici

Più forte, richiede cifratura asimmetrica (il claimant deve avere coppia chiave pubblica/privata).

1. il claimant invia un certificato contenente id e chiave pubblica
2. il verifier invia una sfida calcolata utilizzando la chiave pubblica del certificato del claimant su un nonce random X
3. il claimant decifra la sfida utilizzando la propria chiave privata e invia la risposta al verifier
4. il verifier verifica che la risposta ottenuta sia equivalente al nonce random inviato

Il sistema è sicuro perché solo il claimant possiede la giusta chiave privata per decifrare il messaggio inviato dal verifier.

Sistema robusto, non richiede memorizzazione di segreti sul verifier, ma la procedura è lenta e il verifier deve verificare che il certificato sia valido.

Firma involontaria da parte del claimant: il verifier potrebbe inviare come sfida non la cifratura di nonce e chiave, ma l'hash di un certificato o documento elettronico; il claimant, fidandosi del server, usa la sua chiave privata per cifrare la sfida, creando quindi la firma digitale di quel documento.

8.4 OTP

In fase di autenticazione, la password utilizzata è sempre diversa.

Pro:

- password valida solo una volta per il protocollo di autenticazione
- immune allo sniffing

Contro:

- soggetto a attacchi MITM, il verifier deve autenticare
- difficile fornire password ai subscriber, le password sono molte e si possono esaurire
- l'inserimento della password è difficile

Per fornire le OTP si possono utilizzare password precalcolate o applicativi software.

8.4.1 Sistema S/KEY

Prima implementazione di OTP.

1. al momento della sottoscrizione, il claimant fornisce un segreto iniziale S, come una passphrase
2. applica una funzione di hash sul segreto ottenendo l'OTP P_1
3. applica nuovamente la funzione di hash su P_1 ottenendo P_2
4. il processo di hashing viene ripetuto fino a ottenere P_n
5. l'utente ora ha n OTP salvate a lato del claimant
6. il verifier memorizza solo l'ultima password della sequenza, P_n
7. quando il verifier deve autenticare il claimant, chiederà la password P_{n-1}

8. il claimant invia la password P_{n-1} e il verifier verifica, con funzione di hash, di aver ottenuto P_n

9. P_n viene scartata, e alla prossima autenticazione verrà svolta la stessa operazione ma con la password successiva, fino all'esaurimento delle password

Il sistema richiede che la passphrase (pp) sia lunga almeno 8 caratteri. La pp viene concatenata con un seed inviato dal server, il che permette di utilizzare la stessa pp su server diversi (basta cambiare seme).

Le password sono lunghe 64 bit, compromesso tra sicurezza e complessità. Per semplificare l'inserimento di queste password, si è creato un vocabolario inglese di 2048 parole dal quale si possono pescare le password.

Contro delle OTP:

- scomode da utilizzare
- scomode per accedere ai serivizi con autenticazione ripetuta
- costose se basate su autenticatori hw
- non possono essere utilizzate da processi solo da esseri umani
- è necessario un generatore di buone password random
- problemi legati alla trasmissione all'utente

Con autenticatori hw, problemi legati al DoS e al social engineering.

8.4.2 TOTP

Le OTP possono essere generate a partire dal tempo e dal segreto del claimant. Quando l'utente deve identificarsi nei confronti del verifier, il claimant non pesca da un insieme di OTP precalcolato, ma la calcola sul momento utilizzando:

- segreto condiviso con il verificatore, S_{id}
- l'istante temporale in cui la password viene calcolata

Richiede un calcolo locale al claimant, e sincronizzazione tra clock del claimant e del verifier (altrimenti avrebbero un seed diverso). Un attacco efficace è la disincronizzazione dei clock. Viene consentita una minima finestra di disincronizzazione per evitare errori.

A differenza di OTP, a lato verifier viene memorizzato il segreto condiviso, quindi è attaccabile.

Esempio (RSA SecurID):

1. il claimant invia al verifier, in chiaro: user (id), PIN (S_{id}), token-code calcolato a partire da seed e dall'istante di tempo

2. il verifier, in base a user e PIN verifica contro tre possibili token-code: quello con riferimento temporale corrente, precedente e futuro (finestra temporale d'errore)

È possibile che il claimant invii al verifier un token code speciale, il duress code, che fa scattare un allarme nel caso il claimant si trovi sotto minaccia.

Per comunicare con questo protocollo, claimant e verifier devono installare due componenti ACE (Access Control Engine).

8.4.3 Event-baesd OTP

Usa un contatore intero monotonico C come input oltre al seed: $p(ID, C) = h(C, S_{id})$ Il contatore viene incrementato dal claimant ogni volta che è necessario creare una nuova OTP.

Vantaggi rispetto a TOTP:

- possibile fare autenticazioni a raffica, con serve aspettare il lasso di tempo della finestra d'errore
- è possibile precalcolare OTP (intercettabile da un attaccante)

È possibile che ci siano disincronizzazioni.

8.4.4 OOB OTP

L'OTP è generata lato verifier e trasmessa al claimant tramite un canale diverso rispetto a quello con cui stanno comunicando.

1. alla richiesta di autenticazione, il claimant invia al verifier l'identificativo utente e un segreto
2. il verifier utilizza i dati ricevuti per generare dati di autenticazione, ossia l'OTP
3. l'OTP viene trasmessa su un canale diverso
4. il claimant riceve l'OTP e la trasmette sul canale utilizzato per comunicare con il verifier

L'OTP da claimant a verifier va trasmesso in maniera sicura, ad esempio con cifratura, per evitare MITM

8.4.5 2/MFA

Utilizzo di più di un fattore di autenticazione per proteggere il verifier e aumentare la forza dell'autenticazione. Ad esempio, PIN+OTP. È rischioso se non ci sono meccanismi di protezione su unlock multipli.

8.5 Zero-knowledge proof (ZKP)

Una parte, prover, dimostra all'altra, verifier, che una certa affermazione è vera, senza fornire altre informazioni (ad esempio, senza password).

Esempio concreto: caverna di Ali Babà

1. A deve identificarsi nei confronti di B
2. A entra nella grotta, può prendere due strade
3. B entra nella grotta, non sa che strada abbia preso A
4. B chiede ad A di uscire da una delle due strade
5. se A non conosce la parola magica, non può passare da una strada sbarrata dalla porta
6. in ogni caso, A non trasmette mai a B la propria password

Problema: A è già dalla parte giusta. Soluzione: ripetere più volte la prova.

Nel concreto, conoscena del logaritmo discreto con un certo valore.

8.5.1 ZKPP

Molti algoritmi, spesso associati a stabilire una chiave segreta condivisa, in modo da poter fornire mutua autenticazione e channel binding. Per stabilire una chiave, si utilizzano gli algoritmi PAKE.

8.6 Autenticazione biometrica

Assicurarsi di star interagendo con esseri umani, ad esempio tramite impronte digitali, ecc...

Problemi:

- FAR (False Acceptance Rate)
- FRR (False Rejection Rate)
- accettazione psicologica
- privacy
- insostituibile
- mancanza di standard

FAR e FRR visualizzabile come sovrapposizione di due curve gaussiane.

8.7 Kerberos

Realizzato per autenticazione da remoto. Caratterizzato dall'esistenza di una terza parte fidata, (TTP) che autentica un claimant nei confronti del servizio. Questa possiede il segreto condiviso con il claimant e crea una struttura dati, detta ticket, che il claimant utilizza per autorizzarsi al servizio.

Il ticket è una struttura dati che il claimant non è in grado di interpretare, generata dalla TTP e che verifica il claimant al server.

Termini chiave:

- realm: dominio di Kerberos, insieme di sistemi che utilizzano Kerberos come sistema di autenticazione
- credenziale: user.instance@realm

Flusso di autenticazione con Kerberos:

1. ogni realm ha un authentication server, che possiede tutti i segreti degli utenti del realm, il suo compito è generare un ticket speciale detto Ticket Granting Ticket (TGT)
2. il TGT permette al client di autenticarsi a un Ticket Granting Server (TGS)
3. il TGS consente al client di autenticarsi nei confronti di uno specifico server applicativo

Il ticket è cifrato con una chiave simmetrica, K_s, che deve essere la chiave del server rispetto a cui ci si sta autenticando. Contiene:

- id del server
- id del client
- indirizzo del client
- timestamp
- durata
- chiave Ks, c che permette la comunicazione cifrata tra client e server

L'autenticatore è un'altra struttura dati che viene cifrata con la chiave Ks, c e contiene solo client id e indirizzo e timestamp. Serve per autenticare il client nei confronti di chi dovrà fornire il servizio.

Richiesta di TGT:

1. il client interagisce con l'authentication server trasmettendo il suo identificativo e quello del servizio con cui deve interagire (il TGS)
2. l'AS manda un messaggio, cifrato con la chiave simmetrica del client, contenente
   - $K_{c,TGS}$ utilizzata per la comunicazione sicura tra client e TGS
   - TGT, cifrato con K_{TGS}, la chiave del TGS

Richiesta del ticket:

1. il client invia al TGS l'id del server con il quale intende comunicare, il TGT e l'autenticatore, che è cifrato con $K_{c,TGS}$
2. il TGS estrae il vero ticket, che solo lui può decifrare, e il contenuto dell'autenticatore, controllando che la validità temporale dettata dal timestamp e le corrispondenze tra id siano validi
3. il TGS invia al client un messaggio (ticket) cifrato con la chiave simmetrica condivisa tra TGS e client, contenente
   - il ticket per il servizio al quale il client vuole autenticarsi, cifrato con la chiave del servizio
   - una chiave simmetrica $K_{c,s}$ per la comunicazione sicura tra client e server

Uso del ticket:

1. il ticket ricevuto dal TGS viene inviato al servizio insieme all'autenticatore, il tutto cifrato con $K_{c,s}$
2. il servizio decifra il ticket, confronta i dati ricevuti e, fidandosi del TGS, autentica il client
3. il servizio invia al client il timestamp dell'autenticatore + 1 cifrato con $K_{c,s}$, dando la prova di essere stato in grado di decifrare il messaggio del client

8.7.1 Single sign-on

Fornire all'utente un'unica credenziale con cui autenticarsi per tutte le operazioni su qualunque sistema.

SSO fittizio Password diverse in un file unico.

SSO vero Tipo Kerberos, le applicazioni devono essere modificate per supportarlo.

8.8 FIDO

Autenticazione asimmetrica forte, prevede 2FA e autenticazione biometrica. Flusso di autenticazione:

1. in fase di autenticazione è generata una coppia di chiavi pubblica/privata tramite sistema biometrico
2. la chiave pubblica è trasmessa al server
3. per effettuare il login, l'utente utilizza autenticazione biometrica per sbloccare la chiave privata
4. la chiave privata viene utilizzata per la firma digitale del testo che viene poi trasmesso al server
5. il server verifica la firma con la chiave pubblica

Non vi è nessuna terza parte a cui fare affidamento (a differenza di Kerberos), e a lato server non vengono conservati segreti. Immune al phishing perché la risposta non può essere riutilizzata. Generando una coppia di chiavi a ogni accesso, non esiste tracciabilità tra servizi diversi usati dallo stesso utente.

9 Sicurezza delle applicazioni di rete

9.1 Sicurezza di messaggio

Le proprietà di sicurezza (autenticazione, integrità, segretezza) sono contenute nel messaggio.

Pro: garantisce il non ripudio.

Contro: richiede modifica delle applicazioni, protocollo di sicurezza apposito per comunicare con altre applicazioni.

Due applicazioni dovrebbero condividere solo il canale logico (socket).

9.2 Sicurezza di canale

I messaggi generati non posseggono già le proprietà di sicurezza, queste sono garantite dal canale di comunicazione.

Pro: non richiede modifica alle applicazioni.

Contro: non garantisce il non ripudio.

Il livello sessione è quello ideale per implementare le funzioni di sicurezza, ma non esiste nel modello TCP/IP. È stato proposto un nuovo livello di sessione sicura: si svincola lo sviluppo del protocollo dallo sviluppo dell'applicazione e garantisce interoperabilità tra applicazioni.

9.2.1 SSL

Fornisce:

- autenticazione obbligatoria del server e opzionale del client
- riservatezza dei messaggi
- integrità e autenticazione dei messaggi
- protezione da replay e filtering

SSL lavora a livello di sessione quindi è applicabile a tutti i protocolli basati su TCP.

La peer authentication viene garantita durante la fase di handshake di SSL svolto all'apertura del canale. Il server si autentica presentando la sua chiave pubblica (certificato X.509) e subendo una sfida asimmetrica implicita. L'autenticazione del client è opzionale, ma quando avviene è una sfida esplicita.

L'autenticazione e integrità dei dati avviene mediante il calcolo di un MAC (keyed-digest).

La protezione da attacchi replay e filtering avviene tramite il calcolo di un MID, che identifica il numero di sequenza dei dati.

La riservatezza è garantita tramite generazione dal client di una session key usata per la cifratura simmetrica dei dati. La chiave viene comunicata al server o concordata tramite crittografia a chiave pubblica (RSA, DH).

9.2.2 Flusso di operazione di SSL

1. il browser vuole connettersi a un indirizzo web
2. server e browser negoziano le configurazioni di sicurezza (quali chiavi e algoritmi utilizzare, ecc...)
3. peer authentication del server tramite certificato
4. opzionalmente, quella del browser
5. viene aperto il canale sicuro SSL

Per evitare di rinegoziare a ogni connessione i parametri crittografici, il server SSL può offrire un session identifier. Se il client presenta un session id valido, il server apre subito il canale SSL.

9.2.3 Architettura protocollare con SSL

Si hanno gli header del livello 3 e 4, e un SSL record protocol che fornisce alcune informazioni sul contenuto del messaggio. Dopo l'header, 4 alternative:

- dati di livello applicativo
- protocollo di handshake SSL che realizza la negoziazione dei parametri
- SSL change cipher spec protocol, da un lato permette di comunicare l'intenzione di voler iniziare un canale sicuro, dall'altro cambiare protocollo di cifratura
- SSL alert protocol, per gli avvisi

Generazione di un messaggio SSL:

1. i dati applicativi (es generati da HTTP) vengono frammentati
2. i frammenti sono compressi
3. su ogni frammento viene calcolato un MAC per integrità e identificazione
4. applicazione di padding per successiva cifratura
5. aggiunta header di SSL record protocol

Contenuti dell'header SSL:

- intero senza segno che rappresenta il tipo di dati che contiene: 20 (change cipher spec), alert (21), handshake (22), appdata (23)
- intero senza segno per la versione
- bit della lunghezza dei dati

Calcolo MAC Si utilizza una funzione di digest (come SHA-1) che richiede:

- chiave concordata all'inizio della sessione
- sequence number, intero a 64 bit mai trasmesso ma calcolato implicitamente

Calcolo MAC:

1. il MAC viene generato a partire dai dati compressi, chiave e numero di sequenza
2. la cifratura simmetrica si applica sull'insieme di dati compressi, MAC e padding
3.

È authenticate-then-encrypt.

9.2.4 Protocollo di handshake

Serve per:

- concordare la coppia di algoritmi per la confidenzialità e integrità
- scambiare numeri casuali tra client e server per la successiva generazione di chiavi
- stabilire chiave simmetrica
- negoziare session-id
- scambiare i certificati

Processo per la generazione di parametri crittografici:

1. viene generato un segreto condiviso con algoritmo a chiave pubblica (pre-master secret)
2. dal pre-master secret, client e server ricavano il master secret, che è comune a tutte le sessioni, e tiene conto del client randome e server random della prima connessione
3. per ogni singola connessione, client e server avranno chiavi per MAC, chiavi per cifrare e IV diversi per ogni connessione

9.2.5 Meccanismi effimeri per generazione delle chiavi

Una volta ottenuto un certificato X.509, è possibile generare una chiave simmetrica tramite DH effimero. Questo garantisce perfect forward secrecy, ossia la chiave privata del server viene utilizzata solo per firma, per evitare che chi la conosca possa decifrare tutte le sessioni.

9.2.6 Scambio delle credenziali

1. client hello e server hello, vengono negoziati i parametri crittografici
2. il server invia il suo certificato, un'eventuale richiesta di certificato del client e eventuale server key exchange
3. autenticazione del client con certificato (opzionale), scambio chiave simmetrica usata come parametro per future comunicazioni (client key exchange)
4. la comunicazione termina con un messaggio di change cipher spec e finished

Client hello Contiene

- versione di SSL preferita dal client
- 28 byte pseudorandom (client random)
- id della sessione (0 per nuova sessione)
- elenco delle cipher suite (algo di scambio chiavi, algo di cifratura simmetrico, algo di hash per il MAC) supportate dal client
- metodi di compressione supportati dal client

Hello del server simile, ma con suo random e algo supportati.

Certificate del server Contiene il certificato del server con l'indirizzo del server. Può essere utilizzato per firma o cifratura. Se è solo per firma sarà necessaria una fase di server key exchange.

Certificate Request (opzionale) Richiesta del certificato del client, specifica la lista delle CA che il server ritiene fidate.

Server key exchange (opzionale) Non sempre presente, contiene chiave pubblica del server, firmato dal server stesso. Serve solo nel caso in cui il server ha certificato RSA solo per firma, usa DH anonimo per pre-master secret o problemi di export. È l'unico messaggio con firma esplicita del server.

Certificate del client Trasporta il certificato per la client authentication. Il certificato deve essere stato emesso da una delle CA fidate elencate dal server nel messaggio di certificate request.

Client key exchange Con questo si va a concordare la chiave simmetrica. Potrebbe contenere pre-master secret cifrato con la chiave pubblica del server o la parte pubblica di DH.

Certificate verify Nel caso l'autenticazione del client fosse richiesta, si ha la risposta alla sfida simmetrica esplicita. Il client calcola un hash su tutti i messaggi precedenti e lo firma con la chiave privata del client. Solo in caso di client authentication.

Change cipher spec Provoca l'aggiornamento degli algoritmi da usare nella connessione e fa parte di un protocollo specifico, non dell'exchange.

Finished Primo messaggio protetto con algoritmi negoziati, autentica tutto lo scambio. Contiene un MAC calcolato su tutti i messaggi di handshake precedenti tramite l'uso del master secret.

Come il server vince la sfida implicita:

1. il server ha trasmesso il suo certificato
2. il certificato potrebbe essere stato utilizzato per cifrare il pre-master secret nel messaggio di client key exchange
3. il server dovrà quindi usare la sua chiave privata per decifrarlo (è stato cifrato con RSA)
4. il pre-master secret viene utilizzato per derivare le varie chiavi che saranno utilizzate per il messaggio di finished e quelli successivi

9.2.7 Casi

No chiave effimera, no client auth Il server invia la sua chiave pubblica insieme al certificato, quindi il client cifra il pre-master secret con la chiave del server e la trasmette nel client key exchange.

NO chiave effimera, client auth Il server richiede il certificato al client. Il client invia certificato e certificate verify, la risposta alla sfida simmetrica esplicita che contiene hash di tutti i messaggi precedentemente inviati.

Chiave effimera, no client auth Certificato del server utilizzato solo per firma, server invia server key exchange che contiene chiave RSA firmata con chiave privata del server. Il client key exchange conterrà o il pre-master secret o l'esponente DH cifrati con la chiave RSA trasmessa durante il server key exchange.

9.2.8 Scambio dati e chiusura canale

Quando il canale deve essere chiuso, si usa il protocollo di alert.

1. il client invia un messaggio del tipo LAST-1 alert close notify
2. il server risponde chiudendo la connessione

9.2.9 Ripresa di una sessione

Il client utilizza un session id != 0, ignorando autenticazione e scambio chiavi.

9.2.10 Problemi di SSL-2

- chiave troppo corta (40 bit), stessa lunghezza della chiave di cifratura
- MAC debole
- il MAC era calcolato sui byte di padding ma non sulla sua lunghezza
- handshake non autenticato, quindi un attaccante avrebbe potuto cambiare le ciphersuite per forzare l'uso di cifratura debole

Soluzioni di SSL-3:

- usa chiavi da 128 bit
- usa HMAC
- cambia ordine di MAC e padding
- handshake autenticato

Ha inoltre introdotto compressione dei dati e possibilità di rinegoziare connessione.

9.3 TLS

È SSL ma standardizzato, enfasi su algoritmi di cifratura a hashing standard, non proprietari.

9.3.1 TLS e server virtuali

Allo stesso IP sono associati diversi nomi logici. Soluzioni:

- certificato collettivo (wildcard): chiave privata condivisa tra tutti i server, trattato in modo diverso da ogni browser
- certificato con elenco di server in subjectAltName: chiave privata condivisa tra tutti i server, occorre riemettere un certificato a ogni aggiunta o cancellazione di server
- utilizzo di estensioni

Estensione ALPN Il protocollo applivativo è negoziato tramite il TLS stesso per velocizzare la creazione di una connessione. Si aggiunge header ALPN negli hello.

9.3.2 DTLS

TLS ma appoggiato a UDP. In competizione con IPsec e sicurezza applicativa.

9.3.3 Downgrade TLS

Il client invia la versione più alta supportata, il server notifica al client la versione sa usare (la più alta in comune).

Downgrade insicuro: alcuni server non mandano la risposta corretta ma chiudono la connessione, il client tenta con un numero di versione inferiore.

Attacco: l'attaccante invia una falsa risposta al server per forzare downgrade fino a raggiungere una versione vulnerabile.

Prevenzione: Fallback Signaling Cipher Suite Value, segnala al server che si sta tentando di connettersi un SSL più vecchio.

9.4 Sicurezza di HTTP

9.4.1 HTTP digest authentication

Obsoleto. Viene calcolato, lato client, un keyed digest MD5 applicato su un primo hash contenente username, dominio e password, e un secondo hash contenente metodo e URI. La risposta è MD5 sul primo hash, il secondo e un nonce.

9.4.2 HTTP e SSL

Due alternative:

- TLS then HTTP
- HTTP then TLS

Non sono scelte equivalenti, impatta su FW e IDS.

Tramite client authentication è possibile identificare l'utente che ha aperto un canale.

9.4.3 HSTS

Estensione di HTTPS, il server HTTP indica che la sua interazione con il client deve avvenire solo tramite HTTPS. Evita downgrade di protocollo, la sua scadenza può essere rinnovata a ogni accesso.

9.4.4 HPKP

Il sito HTTPS specifica il digest della propria chiave pubblica e/o una o più CA della propria catena. Il client prende nota e rifiuta di collegarsi a un sito con chiave diversa.

9.5 Sistemi di pagamento elettronico

Architettura di pagamento via web:

1. il merchant offre un servizio
2. il cardholder effettua ordine
3. ricevuto l'ordine, il merchant fa una redirect al POS virtuale
4. il POS virtuale invia una richiesta di dati della carta al cardholder (su TLS)
5. il cardholder invia i dati (su TLS)
6. il POS invia i dati della carta al payment network per validarli
7. se i dati sono validi, il POS riceve OK e invia OK al merchant

9.5.1 PCI DSS

Richiesto da tutte le carte di credito per transazioni Internet.

requisiti Per costruire e mantenere una rete protetta:

- installare e mantenere una configurazione con FW per proteggere i dati dei titolari delle carte
- non usare password di sistema predefinite o altri parametri di sicurezza impostati dai fornitori

Per proteggere i titolari delle carte:

- proteggere i dati dei titolari delle carte memorizzati
- cifrare i dati dei titolari delle cate quando trasmessi su reti pubbliche

Rispettare un programma per la gestione delle vulnerabilità:

- usare e aggiornare l'antivirus
- sviluppare e mantenere applicazioni e sistemi protetti

Implementare misure forti per il controllo degli accessi:

- limitare l'accesso ai dati dei titolari delle carte solo se effettivamente indispensabili per lo svolgimento dell'attività commerciale
- dare un id univoco a ogni utente del SI
- limitare la possibilità di accesso fisico ai dati dei titolari delle carte

Monitorare e testare le reti con regolarità:

- monitorare e tenere traccia di tutti gli accessi effettuati alle risorse di rete e ai dati dei titolari
- eseguire test periodici

Adottare una politica di sicurezza.