



Book

Chapters covered in the course (EN version)

- 1 - Overview
- 2 - Cryptographic Tools
- 3 - User Authentication
- 4 - Access Control
- 6 - Malicious Software
- 7 - Denial-of-Service Attacks
- 11 - Software Security
- 12 - Operating Systems Security
- 13 - Cloud and IoT Security
- 16 - Physical and Infrastructure Security
- 22 - Internet Security Protocols and Standards
- 24 - Wireless Network Security

L01 >

-> Internet Security:

- > Dai una definizione formale di internet security.
- > Da chi viene applicato?
- > Rispetto a cosa viene applicato?

-> Triade CIA:

- > Descrivi formalmente le componenti della CIA.
- > Sono presenti altri elementi? Se si descrivili formalmente.
- > A quale caratteristica spesso viene associata una risorsa critica?

-> Dizionari di sicurezza:

- > Descrivi tutti gli elementi che formano il dizionario di sicurezza.
- > In quanti modi possono essere classificati gli attacchi?
- > Su quale elemento è relazionato il rischio? Su cosa si basa?

-> Conseguenze degli attacchi:

-> Divulgazione:

- > A quale elemento della triade CIA è relazionata?
- > Quali sono gli attacchi relazionati a tale minaccia?

-> Inganno:

- > A quale elemento della triade CIA è relazionata?
- > Quali sono gli attacchi relazionati a tale minaccia?

-> Interruzione:

- > A quale elemento della triade CIA è relazionata?
- > Quali sono gli attacchi relazionati a tale minaccia?

-> Usurpazione:

- > A quale elemento della triade CIA è relazionata?
- > Quali sono gli attacchi relazionati a tale minaccia?

-> **Framework di sicurezza**:

- > Descrivi tutti i requisiti dei framework di sicurezza.

INTRODUZIONE >

Per computer security si intende la capacità di protezione di un sistema autonomo di informazione nel raggiungere obiettivi di Confidentiality, Integrity e Availability rispetto alle sue risorse digitali.

● TRIADE CIA:

- CONFIDENTIALITY:

Prevenzione di accesso / divulgazione non auth delle inform.
e la protezione della privacy personale.

- INTEGRITY:

Prevenzione di modifiche / distruzione delle inform e la
non repudiability e autenticità delle informazioni.

- AVAILABILITY: Garanzia di accesso/uso veloce e affidabile.

NO: Più le risorse sono critiche, più necessiteranno V.
alta disponibilità

Sono stati aggiunti anche altri principi come:

- Non Repudiation: Verifica inconfondibile delle azioni commesse
- AUTH: Verifica dell'identità
- Authorization: Verifica dei permessi
- Accountability: Tracciamento delle azioni commesse.

TERMINOLOGIA >

● ATTACK :

Attività malevola di alterare / distruggere / raccogliere ecc. le risorse del sistema (detto anche minaccia realizzata). Può essere

- PASSIVO: Attacco che non impatta le risorse (es: raccolta dati).
- ATTIVO: // == altera o influenza // (es: distruzione dati).

O anche

- INTERNO: Provocato da attori che usano in maniera **impropria** le autorizzazioni concesse.

- ESTERNO: Provocato da attori esterni al perimetro di sicurezza.

● RISCHIO:

Misura della portata della minaccia basata su:

- Probabilità che si realizzzi

- Danni che causerebbe in caso di realizzazione.

● VULNERABILITÀ: Punto debole che viene può essere sfruttato negli attacchi.

● THREAT: Minaccia al sistema e le sue operazioni.

● ASSET: Risorse o gruppi di risorse che necessitano protezione.

● AVVERSARIO:

Individuo, gruppo o autorità che compie o vuole compiere **attacchi**.

- SCRIPT KIDDIES / AI KIDDIES: Attaccanti con poca esperienza.

- CYBER CRIMINALS: Gruppi criminali mossi da motivazioni econ.

- NATION / STATE ACTORS:

- INSIDER THREATS: Attaccanti interni all'organizzazione.

● OWNER: Proprietario che vuole proteggere gli asset.

● CONTRO-MISURA:

CONSEGUENZE DELLE MINACCIE ➤

Le minacce realizzate sono classificate in:

● DIVULGAZIONE NON AUTORIZZATA:

È una minaccia alla confidentiality e si manifesta con:

- EXPOSURE: Es: fuga di dati sanitari

- INTERCEPTION: Es: Sniffing dei pacchetti

- INFERENCE: Senza effettuare intercettazione effettiva.

- INTRUSIONE: ES: Hackers

● INGANNO:

È una minaccia all'integrità dei dati e/o del sistema. Può essere:

- MASQUERADE: Finge di essere auth.
- FALSIFICATION: Falsifica info su chi me
- REPUDIATION: Nego di aver fatto qualcosa

● INTERRUZIONE:

È una minaccia all'integrità o disponibilità dei dati.

- INCAPACITATION: Impedisce il funzionamento (Disponibilità)
- CORRUPTION: Corrompe il funzionamento (Integrità)
- OBSERVATION: Interferisce nelle comm.
DDos

● USURPATION:

È una minaccia all'integrità di un sistema.

- USO IMPROPRI:
- APPROPRIAZIONE INDEBITA: Botnet

FRAMEWORK DI SICUREZZA >

Diversi framework di sicurezza richiedono dei requisiti come:

- ACCESS CONTROL:

Limitazione delle informazioni e le azioni permesse agli utenti autorizzati.

- AWARENESS & TRAINING:

Verifica continua che il personale capisca i rischi di sicurezza e le loro responsabilità.

- AUDIT & ACCOUNTABILITY:

Logging per consentire monitoring e investigazione delle azioni.

- CERTIFICATION & ASSESSMENT:

Verifica periodica di sicurezza ed eventuali correzioni.

- IDENTIFICATION & AUTHORIZATION:

Prima di condere l'accesso alle risorse.

- RISK ASSESSMENT:

- CONFIGURATION MANAGEMENT:

- INCIDENT RESPONSE :

L02 >

-> Cifratura simmetrica:

- > Che cos'è?
- > Su cosa si basa?
- > Quali sono i suoi requisiti?

-> Algoritmi:

-> Definisci per ogni algoritmo:

- > Nome
- > Se è a blocchi o stream
- > Dimensione di ogni blocco (se a blocchi)
- > Dimensione della chiave
- > Vantaggi
- > Svantaggi

-> Quali fra questi sono quelli di riferimento?

-> Tipologie di cifratura:

-> Block cypher:

- > Come funziona?
- > Come puo' essere implementato?
- > Ha particolari svantaggi?

-> Stream cypher:

- > Come funziona?
- > Qual'è la sua unità di misura?
- > Vengono effettuate delle operazioni sulla chiave?
- > Quali sono i suoi vantaggi?

-> Tipologie di attacchi:

- > Quali sono i principali attacchi relazionati?
- > Quali di questi sono particolarmente efficaci contro la symetric encryption?
- > Quali di questi sono particolarmente efficaci contro le hash function?
- > Quali di questi sono particolarmente efficaci contro le asymmetric encryption?

-> Hash function:

-> Descrivila formalmente.

- > Crea sequenze fisse o variabili?
- > Le crea facilmente o con difficoltà?
- > Le crea solo di size specifiche o qualsiasi?
- > Qual'è il suo scopo principale?
- > Quali sono le sue proprietà principali?
- > Quali sono le sue proprietà aggiuntive?

-> Vulnerabilità:

- > A quali attacchi sono vulnerabili?
- > Quali sono le formule per quantificarne la resistenza?
- > Quali sono i principali algoritmi che la implementano?

-> MAC:

- > Che cos'è?
- > Cosa garantisce?
- > Descrivine il funzionamento ad alto livello.
- > Descrivi in quali modi viene implementato.

-> Numeri casuali:

- > Quali sono i requisiti di un numero casuale? Vengono misurati?
- > I principali algoritmi si basano su numeri casuali?
- > Cos'è un numero pseudo-casuale?
- > Su cosa si basano i generatori di numeri pseudo-casuali?

SYMMETRIC ENCRYPTION

La sym. encryption prevede l'uso di una **private key** condivisa fra sender e receiver per criptare e decriptare dei dati.

FUNZIONAMENTO GENERALE

I dati (plaintext) passano attraverso una funzione crittografica $y = E[X, K]$. Il receiver decripta con la f. inversa $x = D[y, K]$

REQ. DI SYM. ENCR.

SECURE KEY DISTRIBUTION

La private Key deve essere conosciuta da S/R e condivisa attraverso un canale di com. sicuro.

COMPUTATIONAL SEC.:

Anche con risorse infinite, decifrare manualmente la chiave dovrebbe occupare tanto tempo.

KHEKOFF'S PRINCIPLE:

La sicurezza non dipende dal tenere l'algo. segreto.

RESISTANCE TO CRYPTOANALYSIS:

Resistenza a known plaintext, chosen plaintext e chosen cypher text attacks.

SEC. KEY GENERATION:

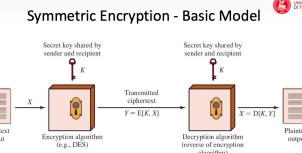
La p. key deve essere generata usando generatori di testo/num casuali o pseudo casuali

SEC. KEY STORAGE: Su sistemi protetti da muro HW o altri.

SEC. KEY ROTATION: Periodicam. per ridurre rischi crypto analysis.

ALGORITMI USATI

DES:



	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard

AES = Advanced Encryption Standard

Cifra plaintext a blocchi di 64 bit con una chiave di 56 bit. È ormai stato **dismesso** per la dimensione della chiave **troppo corta** che lo rendono vulnerabile a brute force.

- TRIPLE DES:

Prevede di cifrare tre volte un plaintext a blocchi di 64 bit usando DES, con **2/3 chiavi da 112 / 168 bit**.

+ È resistente a brute force / freq. analysis

- È lento.

- AES

Nato per risolvere gli svantaggi del T.DES, cifra un plaintext a blocchi da 128 bit con chiavi da **128 / 192 / 256 b**.

È attualmente l'algoritmo simmetrico di riferimento.

• TIPOLOGIE DI CIFRATURA:

• BLOCK CIPHER

Il plain text viene cifrato un blocco per volta.

In caso sia maggiore della dimensione del blocco si puo' gestire con:

→ ELECTRONIC CODEBOOK [ECB]

Il plaintext viene diviso in blocchi.

Ogni blocco viene **cifrato** usando la **stessa chiave**.

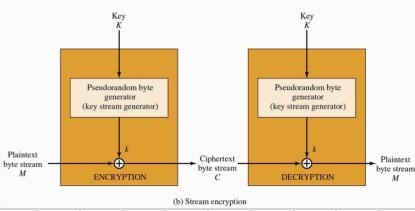
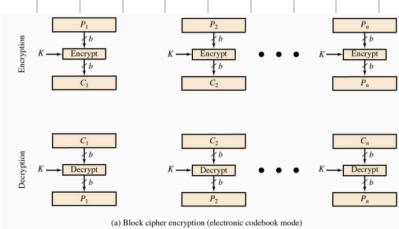
- vulnerabile a frequency based attacks.

→ MODES OF OPERATION:

Modi alternativi creati per rendere il block cipher sicuro.

+ La chiave puo' essere ri-utilizzata.

• STREAM CIPHER:



Il plaintext viene cifrato un **elemento per volta**, usando solitamente come 'unità di misura' 1 Byte.

- FUNZIONAMENTO

La chiave viene elaborata da uno **pseudo-random number generator** e viene applicata all'elemento del plain text

+ Più veloce

TIPOLOGIE DI ATTACCHI

- KNOWN PLAIN-T: Dato plain e cipher text, si analizza il proc.
- CHOSEN PLAIN-T: Dato plain, si critta per analizzare il risultato
- CHOSEN CIPHER-T: ~~cipher, si decripta~~
- S. • FREQ. ANAL : Analisi di pattern nel cipher
- S. • MAN IN THE M.: Intercettazione e/o modifica di com. fia attori
- S. • BRUTE FORCE: In cui in media, metà delle chiavi sono provate per avere successo.

HASH FUNCTION

MDS. SHA1, SHA2, SHA3
COLX ✓ COL X ✓

È una funzione che crea una **sequenza fissa** di caratteri casuali facilmente di qualsiasi size a partire da un blocco di dati con lo scopo di creare **un'impronta digitale**.

- PROPRIETÀ PRINCIPALI

- ONE-WAY: Non deve essere possibile invertire la hash F.
- DETERMINISTIC: Deve dare lo stesso risultato per lo stesso msg.
- AVALANCE EFF: Deve produrre un output molto diff. ad ogni var. msg.

- PROPRIETÀ AGGIUNTIVE

- PRE-IMG. RESISTANT: Dato l'output deve essere difficile \approx per $h(x)$
- Z° PRE-IMG. RESISTANT: Dato x codice \approx trovare $y \neq x$ tale che $h(x) = h(y)$
- STRONG. COLLISION RES: Per ogni x, y diversi deve essere diff. trovare $h(x) = h(y)$

- VULNERABILITÀ

Sono vulnerabili a **cryptanalysis** e **brute-force attacks**.

La resistenza delle hash function contro attacchi brute-force è:

- 2^n : Per pre-image, 2^n pre-image.

- $2^{\frac{n}{2}}$: Per collision resistance.

$n = \text{hash len}$

MESSAGE AUTHENTICATION CODE > MAC

È un blocco di dati che viene aggiunto al messaggio trasmesso per garantire l'integrity e l'authentication del sender.

• FUNZIONAMENTO: AD ALTO LIVELLO

Sender e receiver si accordano su un secret comune K

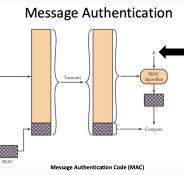
• SENDER: $\text{MAC}_s = F(M, K)$

Calcola il MAC uscendo il messaggio e il secret e lo invia v.

• RECEIVER: $\text{MAC}_r = F(M, K) == \text{MAC}_s$

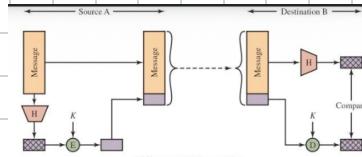
Calcola il MAC e lo confronta con quello ricevuto.

con il msg.



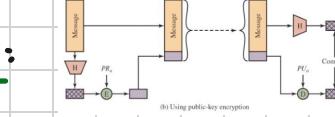
• IMPLEMENTAZIONE

• CON SYM. KEY:



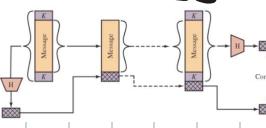
Genera il MAC sull'Hash del messaggio con una symm. key.

• DIGITAL SIGNATURE:



"firma" l'Hash del messaggio usando la private K del sender.

• KEY CONCAT



Genera l'Hash sul messaggio con append della chiave.

RANDOM NUMBERS >

La generazione di numeri random, necessari in sicurezza, si basa su:

- UNIFORM DISTRIBUTION: Ogni numero ha la stessa probabilità degli altri.

- INDEPENDENCE: Nessun numero può essere dedotto da altri.

Attualmente non esistono test per misurare l'indipendenza. Difatti,

molti algoritmi si basano su algoritmi di generazione deterministici che generano numeri pseudo casuali.

• NUM. PSEUDO CASUALE.

Numeri prodotti da un algoritmo che supera i test di uniform distribution e independence (del non essere dependenti) ma che si basa su generazioni non veramente casuali, del quale ci si fida comunque.

I veri generatori di numeri casuali si basano sull'osservazione di fenomeni naturali imprevedibili.

L03 >

-> Cifratura asimmetrica:

- > Definiscila in maniera formale.
- > Da chi è stata scoperta?
- > Quali sono gli utilizzi possibili? (specifica bene quale chiave di chi viene usata)
- > Quali sono i requisiti per la cripotassi?

-> Algoritmi di cripotassi:

- > Elenca i principali algoritmi.

-> RSA:

- > È un algoritmo block-cypher o stream cypher?
- > Come possono essere classificati i plaintext e ciphertext in RSA?
- > Come sono definite le operazioni di cripotassi e decripotassi in RSA?

-> Digital Signature:

- > Cosa verifica?
- > Come funziona? Sia dal punto di vista del sender, che dal punto di vista del receiver.

-> Authentication:

- > Definisci formalmente.
- > Descrivi i requisiti dell'autenticazione.
- > Descrivi gli attori coinvolti nel processo di AuthN
- > Cosa sono IAL e AAL?

-> Password-based AuthN:

- > Elenca tutti gli attacchi possibili

-> Gestione concettuale in UNIX:

- > Su quale elemento tecnico si basano i sistemi UNIX?
- > Che cos'è un salt? A cosa serves?
- > Descrivi i processi di sign-up e sign-in.

-> Gestione implementativa UNIX:

- > Su cosa si basava il sistema originale? Quali erano dimensioni di hash e key?
- > Su cosa si basano le moderne versioni UNIX? Quali erano dimensioni di hash e key?

-> Gestione implementativa LINUX:

- > Che algoritmi vengono usati?
- > Che cos'è il PAM

CRIPTOGRAFIA SIMMETRICA ➤

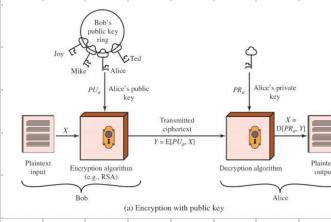
Tipologia di crittografica che si serve di una chiave pubblica e privata per trasmettere info. senza dover trasmettere un secret

Scoperta da Diffie e Hellman.

• UTILIZZI

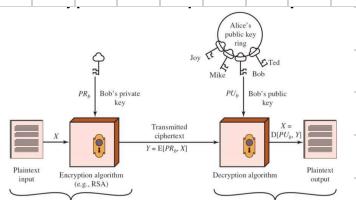
- TRASMISSIONE SICURA

Dato un plain text, lo si cifra usando la **public key** del receiver. Infine il receiver userà la sua **private key** per decifrare il messaggio.



- DIGITAL SIGNATURE / VERIFICA OWNER/INTEGRITY

Dato il plaintext, il sender lo cifra con la **sua private key**. Il receiver userà la **public key** del sender per decifrare il msg. e verificarne l'integrità.



- SIMMETRIC KEY DISTRIBUTION (Diffie-Hellman)

• REQUISITI

- Facilità di creazione di paio private/public key.
- Difficoltà di decifrare senza chiave o trovare chiave

• ALGORITMI

- Digital Signature Standard : Usata per la digital signature
- Elliptic Curve Cryptography : Alternativa ad RSA con chiavi più piccole.

• RSA:

Algoritmo **block cipher** basato su una formula matematica in cui plaintext e cyphertext sono interi $\in [0, n-1]$ per n

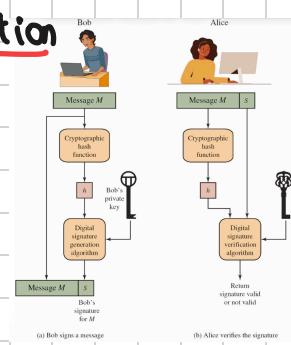
- CIFRATURA: $c_i = M^e \pmod{n}$ public = e, n

- DECIFRATURA: $M = c_i^d \pmod{n}$ private = d, n

Consente di verificare: origine, integrità e non-repudiation

• FUNZIONAMENTO:

- SENDER: Crea un hash del msg. e lo firma con la sua private key. Invia: msg + sign.
- RECEIVER: Crea un hash del msg. che verrà usato come input insieme alla public key (sender) nell'algoritmo di verifica.



AUTENTICAZIONE > AUTHN #Capitolo 3 libro

Verifica l'identità di un sistema o di un utente.

• REQUISITI DI IDENTIFICATION & AUTHN

conto

- IDENTIFICARE: sistemi, processi e device che agiscono per l'utente.
- AUTENTICARE: L'identità dei sistemi/processi/device come **prerequisito** per consentire l'accesso ai sistemi.

3. MULTI-FACTOR AUTHN:

Per network access (utenti privilegiati / non privilegiati) e LAN access (utenti privilegiati).

- | Qualcosa che so (Pin/PSW)
- | Qualcosa che ho (smartcard)
- | == che sono (FaceID, impronta)
- | == che faccio (voce, scrittura)

4. REPLAY-RESISTANCE: Resistenza a re-invio delle Authn req. in rete.

5. PREVENT ID REUSE: Per un periodo di tempo.

6. ID EXPIRATION: Dopo // // // //

7. MINIMUM PSW COMPLEXITY: Con lunghezza e char diversi da vecchia V.

8. PREVENT PSW REUSE: Per un po' di generazioni.

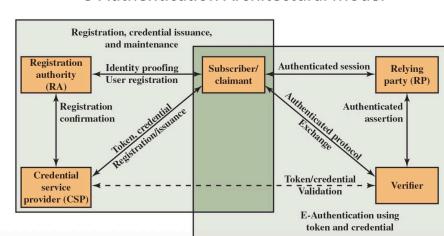
9. CONSENTI PSW TEMPORANEE: Per cambio password.

10. STORAGE & TRASMISSIONE DI PSW CRIPTATE

11. OBSCURE FEEDBACK: Di password authn.

• ATTORI:

The NIST SP 800-63-3
e-Authentication Architectural Model



- SUBSCRIBER / CLAIMANT: È il soggetto da autenticare.

- REGISTRATION AUTHORITY (RA):

Soggetto trusted al quale il subscriber si sottoscrive per la verifica dell'identità, che (se positiva) verrà trasmessa al CSP per erogare le credenziali.

- CREDENTIAL SERVICE PROVIDER (CSP):

Eroga le credenziali al subscriber dopo la conferma della RA.

Le credenziali effettuano un **binding** fra l'identità del subscriber e il token in suo possesso.

- VERIFIER:

Soggetto che verifica il possesso e controllo del token, che trasmette l'esito al Relying Party.

- RELYING PARTY (RP):

Soggetto che si affida al verifier per la verifica dell'AuthN.

ASSURANCE LEVEL >

- IAL1: Nessuna necessità di verifiche id.

- IAL2: Richiede evidenza dell'identità vera con verifiche remote o in presenza.

- IAL3: Richiede evidenza dell'identità **solo in presenza**.

E di AuthN Assurance Levels:

- AAL1: Richiede ID + PSW.

- AAL2: AuthN con multifactor

- AAL3: Verifica approfondita con multifactor.

PASSWORD-BASED AUTHN >

• ATTACCHI

1. OFFLINE ATTACKS: Il password file viene rubato

Entity in Model	SPID example
RA/CSP (combined)	Poste Italiane, Aruba, InfoCert, Namirial, etc.
Subscriber/Claimant	Italian citizen registering and using SPID
Verifier	The Identity Provider's authentication system
Relying Party (RP)	INPS, Agenzia delle Entrate, your bank, university portal

2. SPECIFIC ACCOUNT ATTACK: Brute-force delle psw su uno specifico ID.

3. POPULAR PASSWORD ATTACK: Si provano password semplici contro più utenti.

4. PSW GUESSING contro un utente:

5. WORKSTATION HIJACKING

6. USER MISTAKES: Social engineering

7. MULTIPLE PSW USE

8. ELECTRONIC MONITORING: Sniffing, eavesdropping

● GESTIONE IN UNIX: CONCETTUALE

Tutti i sistemi UNIX possiedono una password table, in cui salvano:

- SALT:

Stringa di dimensione fissa come numero casuale /pseudo casuale

Serve a:

- Nascondere la presenza di password con lo stesso hash.
- Rendere più difficili i dictionary attacks, incrementando il ~~#~~ di password possibili di 2^b per b num. bits del salt.
- Rendere impossibile sapere se un utente usa la stessa password.

- HASHED PASSWORD:

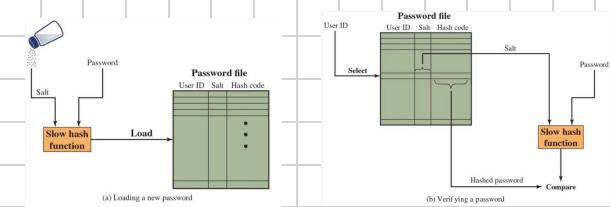
Un hash della password, combinata con il salt (pre hash) in una funzione di hash lenta per rallentare gli attacchi.

- USER ID: Uscita per indicizzare la tabella.

Il funzionamento prevede:

● FASE DI REGISTRAZIONE:

L'utente sceglie un USER ID e una password, la quale viene "hashata" con il salt e salvata nella password table.



• FASE DI LOGIN

L'utente invia ID e Password. Il sistema ricalcola l'hash basandosi sul salt per l'utente (se esiste) e lo **ricalcola** per confrontarlo con quello nella password table.

• GESTIONE UNIX: IMPLEMENTAZIONE

• SCHEMA ORIGINALE

Si usava crypt(3), una variante del DES per cifrare fino a 8 caratteri insieme ad un hash da 12 bit, trasformandolo in una hash function lenta (eseguita 25 volte).

• SCHEMI PIÙ ROBUSTI

Distribuzioni più recenti adottano MDS con salt da h8 bit consentendo lunghezze arbitrarie di password in hash 128 bit.

MDS è rallentato da un loop con 1000 iterazioni.

FreeBSD usa Bcrypt con salt da 128 bit e hash 192bit.

• IMPLEMENTAZIONE LINUX

Usa SHA-512 o yescript con rallentamenti configurabili e

• SHADOW PSW FILE: Accessibile solo da superuser

• PLUGGABLE AUTH MODULE

Modulo centralizzato di autenticazione usabile dalle applicazioni.

L04 >

-> Gestione delle Password.

- > Quali sono i principali attacchi che sfruttano le password weakness?
- > Quali sono i principi sui quali si è evoluta la password defence?
- > Come si è evoluta il password cracking?

-> Definisci i metodi principali di Token-based Auth.

- > Cos'è e come funziona la Biometric AuthN?
- > Cos'è e a cosa serve la Challenge-Response protocol?
- > Come funziona per le password?
- > Come funziona per la biometria?

-> Quali sono gli attacchi che si possono subire in AuthN?

PASSWORD BASED AUTHN >

• PASSWORD WEAKNESS

• DICTIONARY ATTACKS:

Si crea un dizionario di password. Per ogni password, si genera un hash con ogni salt possibile per tentare il match.

• RAINBOW TABLE:

Variante molto grande che contiene hash precompilati con ogni salt possibile per diminuire il tempo (counter: psw lunghe).

Sono comuni gli attacchi basati su brute-force e dictionary attack.

• MODERN APPROACHES

Ai giorni nostri difesa e password cracking sono migliorati:

- DIFESA:

Spesso si avverte una complex password policy (min. psw compl.)

- ATTACCO:

- CPU / GPU più potenti

- PSW analysis derivate dai leak.

- Algoritmi per generare password probabili

• PASSWORD SELECTION

Si attuano strategie volte a eliminare psw semplici, consentendo alcune facili da ricordare.

- USER EDUCATION:

- COMPUTER GENERATED PSW

- REACTIVE PSW CHECK: Il sistema prova a 'bucare' le password

- COMPLEX PSW POLICY: // // valuta se la password è sicura

TOKEN-BASED AUTH >

È un oggetto fisico usato dall'utente per AutN. Possono essere:

• MEMORY CARDS: ES: Hotel Card, Credit card (banda m.)

Card che contengono info. ma non possono processare niente.

• UTILIZZO

- DA SOLO: Per avere un accesso fisico

- CON PW O PSW: Per autenticazione. La combo risulta migliore della sola password (crack PW + access card).

- Token lost - Reader Speciale - Può essere scambiato

• SMART TOKENS: ES: Credit card (No Banda magnetica)

Token smart che contiene un microprocessore e che dialoga con il reader/writer con un' interfaccia elettronica contact o contact-less.

• AUTENTICAZIONE:

- STATIC: L'utente fa AuthN con il token e il token fa AuthW con il computer

- DYNAMIC PSW GENERATOR:

Si genera una password periodicamente inserita dall'utente o dal token stesso.

- CHALLENGE-RESPONSE:

Il sistema genera una challenge a cui il token risponde generando una risposta.

TOKEN-BASED AUTHN > SMART CARD

Tipologia di smart token più importante dall'aspetto di una carta di credito.

• STRUTTURA MEMORIA:

- ROM: Contiene dati che non cambiano.

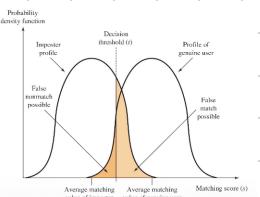
- EEROM: Contiene dati e applicazioni

- RAT : // dati temporanei.
- eID ES: Carta d'Identità Elettronica
 - Usato per autenticare i cittadini all'uso dei servizi del governo.
 - La CIE contiene nome, cognome, data nascita, foto e impronte
- AUTHN PROTOCOLS:
 - SAML 2.0
 - OpenID Connect
 - PKI & Digital Cert
- TOKENS BASED AUTHN > NEW TOKENS
- OTP:
 - Usano un secret combinato con il current-time o nonce (altrui) in una hash function / block cypher per generare un codice da usare durante AutN.
- TOTP:
 - Variante dell'OTP che usa l'orologio interno per generare il codice
 - FUNZIONAMENTO
 1. Setup di un secret condiviso con qr code.
 2. In AutN si genera il codice che vale ogni 30s.
che viene calcolato anche dal server per match.
- BIOELECTRIC AUTH >
 - Si basa su impronte, retina, voce e faccia. Solitamente sono usate impronte e faccia, anche se le valutazioni devono tenere conto di privacy e resilienza agli attacchi.
- ENROLLMENT:
 - Processo di registrazione di un utente al sistema biometrico.
 - Dopo la scansione, i dati fisici vengono convertiti in un codice numerico (Template) che sarà usato per la verifica. In aggiunta viene impostato anche un PIN.



• VERIFICA SCANSIONE BIOMETRICA

In fase di verifica, dopo aver convertito le caratteristiche fisiche nel valore numerico, questo si confronta con un threshold impostato. Se $>$ threshold \rightarrow passa



• VERIFICATION: Scansione Biometrica + PIN

• ID: Scansione Biometrica (wo PIN)

CHALLENGE - RESPONSE PROTOCOLS >

Protocolli di AuthN usati nell'autenticazione remota (es. Internet) per prevenire attacchi di manipolazione dei pacchetti:

• PSO. PROTOCOL

1. Client invia lo username U all'Host.

2. Host restituisce un rand. number r , insieme a 2 funzioni di hash/crypto $f(\cdot), h(\cdot)$

3. Il Client manda all'Host l'esecuzione di $f(r; h(P))$

4. L'Host calcola $f(r, h(P)) == f(r', h(P(U)))$ e torna l'esito

Approcci moderni usano SECURE REMOTE PASSWORD per effettuare l'authN on device.

• BIOMETRIC AUTHN:

Prevede una verifica locale della biometria tramite lo sblocco della chiave privata e verificata dalla chiave pubblica nel server senza inviare niente al server.

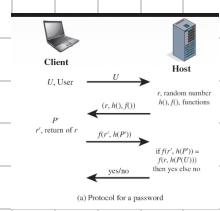
AUTHN ISSUES >

- CLIENT ATTACKS: Attacchi senza accesso al server.

- HOST //: Attacchi diretti al server.

- EAVESDROPPING: Attacchi che coinvolgono prossimità

- MFA FAKE FATIGUE: MFA authN finti per farsi dare MFA vero



L05 >

-> **Intro ad Access Control.**

- > Come si puo' definire l'access control?
- > Concettualmente cosa deve impedire?
- > Che cos'è il least privilege principle? È usato dall'access control?

-> **Access control architecture:**

- > In che punto del tempo avviene l'access control?

- > Con cosa interagisce?

- > Cosa contiene la cosa con cui interagisce?

-> Access Control Policies:

- > Definisci tutte le access control policy.

- > Quali di queste si basano sull'identità dell'utente autenticato?

-> **Access Control Actors:**

- > Quali sono gli attori che partecipano all'access control?

- > Quale struttura viene usata per regolare gli accessi alle risorse da parte degli utenti?

- > Che cos'è una ACL?

- > Che regole puo' contenere?

ACCESS CONTROL >

• DEFINIZIONI

1. NISTR 7298:

È l'accettazione o rifiuto di richieste specifiche per
 → info. o sistemi di elab. di tali informazioni
 → accessi fisici a struttura

2. RFC 4494

Processo che regola l'accesso alle risorse secondo una politica specifica che consente l'accesso solo a chi è autorizzato.

L'obiettivo è di impedire l'accesso a utenti illegittimi e impedire l'accesso in modo non autorizzato a utenti leggitti

Il NIST-800-171 regola i req. di sicurezza che si basano sul ^{least privilege p.}

• LEAST PRIVILEGE PRINCIPLE No root, Cambio ruoli → perm.

Gli utenti / servizi devono avere sempre solo i permessi necessari.

ACCESS CONTROL ARCH >

La fase di Access Control avviene dopo Auth e prevede l'interazione con un DB di ruoli per verificare l'accesso alla risorsa da parte dell'utente autenticato. Il DB contiene le Access Control Policies:

• A.C. POLICY

Definisce i tipi di accesso, le condizioni e da chi. Sono:

• DISCRETIONARY A.C. [DAC] ES: Sharepoint, G.Drive

L'A.C. avviene sulla base dell'identità e delle Autz
 ad altri dell'utente. L'utente a sua discrezione può concedere l'accesso v.

• MANDATORY A.C. [MAC]: ES: Android

L'A.C. avviene sulla base dell'identità e delle elichelte

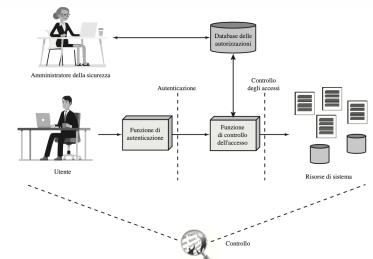


Figura 3.1 Relazione tra il controllo degli accessi e altre funzionalità di sicurezza. Fonte: basata su [SAND94].

di sicurezza (misura di criticità risorse). Non contente di concedere l'accesso al solo giudizio del singolo utente.

- ROLE-BASED A.C. [RBAC]

L'A.C. avviene sui ruoli che gli utenti hanno a sistema.

- ATTRIBUTE BASED A.C. : A.C. su attributi utente e risorsa.

ACCESS CONTROL > ATTORI

Nell'A.C. partecipano

- SOGGETTO: Un'entità / processo capace di richiedere l'accesso alle risorse per conto di utenti / applicazioni.
→ owner della risorsa → gruppo assegnato → all'altro
- OGGETTO: È una risorsa capace di leggere / scrivere info. con accesso controllato.
- PERM. DI ACCESSO: Descrive la modalità con cui un soggetto può richiedere l'accesso all'oggetto.
→ read → write → execute → create → search → del

ACCESS CONTROL > UNIX

Unix usa gli inode, strutture contenenti informazioni su un particolare file (es: access control policies), contenuti in una inode table.

Ogni directory contiene puntatori agli inodes associati alla lista di file contenuti.

- TRADITIONAL ACCESS CONTROL

Ogni utente ha associato uno user ID e ogni gruppo un group ID.

- CREAZIONE FILE

Ogni file creato viene associato ad un owner user e al suo group ID di appartenenza o al group ID del parent folder (se ha set GID abilitato).

• PROTECTION BITS

Ogni file ha associati 12 bit, dei quali

- a bit: sono i permessi read, write ed execute per user group (del file) e others.

Se applicati su una cartella:

→ read/write: Consentono di leggere,...,sui file contenuti

→ execute: Consente la navigazione nella cartella.

- Set UID: (Set User ID)

→ Se applicato a una cartella: viene ignorato

→ // // a un file: Durante la sua esecuzione verranno in aggiunta temporaneamente applicati i permessi dell'owner del file ✓.

- Set GID: (Set Group ID)

→ Se applicato a una cartella: Il file erediterà il gruppo ✓ della dir

→ // // a un file: Stesso discorso di SetUID ma sul gruppo.

- Sticky Bit: Solo l'owner può rinominare, muovere o eliminare il file.

• MODERN APPROACH

Vengono impostate delle ACL tramite il comando

setfacl -m U:username:rwx filename ← singolo user

setfacl -m m::rwx filename ← intera maschera

Le ACL usano una maschera per impostare il grado massimo che le entries possono avere

L06 >

- INTERNET SECURITY :

- SICUREZZA A LIVELLO APPLICATION [2]

- // // NETWORK [1]

- // = LINK [3]

- PUBLIC KEY:

- QUAL'È IL PROBLEMA NELLA TRASMISSIONE DELLA CHIAVE PUBBLICA?

- INTRO ALLO STANDARD USATO PER RISOLVERE IL PROBLEMA.

- STANDARD DI RISOLVIMENTO PROBLEMA

- GERARCHIA CA E INTRO MODO DI VERIFICA

- SCRUTTURA DI UN CERTIFICATO [1]

- GLI ATTORI DELLA CHAIN OF TRUST E I LORO CERTIFICATI [3]

- STEP DI VALIDAZIONE CERT [ALTO LVL + 5 STEP]

- POSSO REVOCARE UN CERTIFICATO?

- COSA FA IL PROTOCOLLO ACME?

INTERNET SECURITY ➔

Ogni layer può applicare dei protocolli di sicurezza distinti:

Protocol	Layer	Scope	Use Case	Authentication	Encryption
HTTPS/TLS	4-7	Application	Web, Email	X.509 Certs	Yes
SSH	4-7	Application	Remote access	Key/Password	Yes
IPsec	3	Network	VPN, Site-to-Site	PSK/Certs	Yes
WPA2/WPA3	2	Link	Wireless	PSK/802.1X	Yes
802.1X	2	Link	LAN Access	EAP Methods	-
MACsec	2	Link	Ethernet	802.1X	Yes

- APPLICATION: Comm. sicura fra applicazioni

- HTTPS / TLS: garantiscono la CIA tramite certificati.

- SSLe : garantisce la // // username / password.

- NETWORK: IPSEC garantisce la comm. sicura fra hosts con i certificati

- LINK:

- WPA2/WPA3: Garantisce CIA nelle connessioni wifi fra wireless station ed AP tramite criptografia

- 802.1X: Protocollo di authN a livello link.

- MACsec: Fornisce CIA per la comunicazione ethernet.

PUBLIC KEY ➔

- DISTRIBUTION & MAN-IN-THE-MIDDLE

La chiave pubblica per consentire la comm. sicura deve essere trasmessa al sender in maniera sicura. Se viene intercettata da un'elemento intermedio si rischia che questo dia la sua chiave per decifrare i messaggi, per poi mandarli al receiver.

- X.509 AUTH SERVICE

X.509 è uno Standard per fare AuthN basata su un modello a certificati.

Il funzionamento prevede che chi "trasmette" crei il proprio certificato (come coppia di chiavi PUBL/PRIV) e si "faccia firmare" il cert. da una certification authority con la sua chiave privata

Il cert e la public key saranno poi pubblicati su un

registro pubblico della CA.

che il cert sia valido nel

Chi valida riceverà il certificato e pub. Key che potrà verificare

Esempio: In HTTPS viene usato (il famoso lucchetto del browser).

X.509 >

• GERARCHIA CA

Ogni CA espone la sua chiave pubblica, che a sua volta può essere validata da altre CA formando un albero.

Ogni CA possiede dei client certificates e parent certificates. e nel processo di verifica bisogna validare tutta la gerarchia.

• STRUTTURA CERTIFICATO

Segue una struttura che si "espande" a seconda della versione (corrente: V3):

1. VERSION: la versione

2. SERIAL NUMBER: numero identificativo.

3. ALGORITHM: Specifica algoritmo e param. usciti per la firma del CA

4. ISSUER NAME: Specifica la CA parent.

5. PERIODO DI VALIDITÀ: Nella forma not before, not after.

6. SUBJECT NAME: Chi ha fatto il cert

7. PUBLIC KEY INFO: Specifica algo, param e chiave del **SUBJECT**.

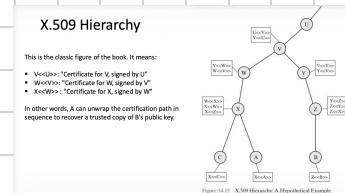
8. ISSUER ID (v2)

a SUBJECT ID (v2)

10 EXTENSIONS: (v3)

Contengono dati relativi alla v3 come:

- SUBJECT ALTERNATIVE NAMES: Contengono alias di hostname v. e IP
- KEY USAGE: Specifica l'uso della chiave (es: cifratura, sign...)
- CONSTRAINTS:



- AUTHORITY INFORMATION ACCESS: Contiene le location (es: Issuer)

• SIGNATURE: Specifica algo, params e hash della firma del **SUBJECT**.

• CHAIN OF TRUST: ACTORI

• ROOT CERTIFICATE:

Emessi dalle root C.A., entità di cui ci si fida, pertanto i loro certificati **non scadono**, sono firmati da loro stessi e sono ben protetti.

→ PRIVATE KEY: Firmano i loro cert e quelli delle intermediate CA

→ PUBLIC // : Validano i cert. delle intermediate CA.

• INTERMEDIATE CERT:

Emesso da una intermediate CA che dipende da una root CA (o da altre intermediate) con certificati che scadono

• END-ENTITY CERT: Emesso dalla entità finale e firmato da

X509 > VALIDAZIONI

La validazione avviene in maniera distribuita sui singoli dispositivi in cui nel S.O., browser e (a volte) app hanno memorizzato le chiavi pubbliche e certificati delle root CA.

Per cui la gestione dei root CA sono critici.

• STEP DI VALIDAZIONE

1. Verificare che non sia scaduto coi campi appropriati (s)

2. == che la firma della CA esiste ed è valido

3. Risalgo la catena delle CA validando le firme

Nel caso dei siti web

4. Check Hostname: Che combaci con campi come Subject. o Subject Alt. Names

5. == Revocation Status: Per verificare se il cert. è stato

(Non funziona bene)

revocato dalla CA anche se valido.

X. SOA > REVOCATION LIST

Ogni CA puo' esporre una revocation list contenente tutti i certificati revocati che pero' nel tempo è diventata troppo grande. Solitamente si usano protocolli come CRLite o OCSP per interrogare la revocation list, ma non sono tanto affidabili.

Si preferisce avere **durate carte e log pubblici**.

PROTOCOLLO ACME >

Protocollo automatico per l'emissione e rinnovo dei certificati fra server e CA.

Introduce il problema del: "Come so che xi tu il subject del certificato?"

- CHALLENGE AUTHN:

Per risolvere, ACME usa le challenge based Authn:

- TRAMITE TOKEN:

Si invia un token e si chiede di metterlo in un folder preciso.

Dopo di che, diversi server nel mondo faranno verifiche autonome in un breve lasso di tempo.

- TRAMITE DNS:

Alternativa che chiede di salvare anche un DNS record di tipo TXT come verifica rafforzata.

L⁰⁷>

TLS

- INTRO :

- COS'É IL TLS? COME INTEGRA LA CIA?

- STRUTTURA TLS & FUNZIONI [S]

- PROCESSO DI ELAB. DATI [4]

- STRUTTURA RECORD PROT [4]

- HANDSHAKE:

- LE FASI [4]

- QUANTO DURA?

- GESTIONE CHIAVI:

- QUANTI E QUALI SECRET CI SONO?

- COS'É LA FW SECRECY:

- POSSO CIFRARE USANDO SOLO LA PUB KEY OTTENUTA?

- EPHMTERAL DIFFIE-HELLMAN

- COS'É E COME FUNZIONA

- EFF: SI PUO' EFFICIENTARE TLS?

- SERVER NAME INDICATION: COS'É E PERCHÉ USARE

TLS >

È un protocollo usato prima del Transport Layer usato dalla maggior parte dei protocolli applicativi per integrare la CIA:

→ C: Tramite cifratura simmetrica

→ I: hash crittografico

→ A: cifratura asimmetrica (pk)

HTTP usa il TLS dalla versione 2.

• STRUTTURA TLS

È formato da 2 layer:

- 1 fra Handshake, Change Cypher Spec, Alert, Heart beat o HTTP
- Record Protocol

• FUNZIONI DEL LIVELLO SUP:

• RECORD PROTOCOL

Fornisce compressione, cifratura e MAC per tutti i TLS messages.

• TLS MESSAGE TYPES:

• HANDSHAKE:

Handshake a livello TLS per scambiarsi parametri

• CHANGE CYPLER SPEC: Per scambio di dati sul cifrario

• ALERT: Per notifica su errori e warning

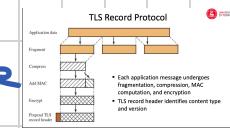
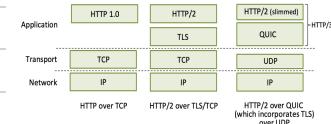
• HEART-BEAT: Per mantenere la connessione attiva.

• HTTP: O altri protocolli applicativi di comunicazione.

• PROCESSO DI ELAB. DATI

1. FRAMM.: I dati trasmessi in TLS hanno 16 KB come dim. massima. Se maggiori → frammentazione

2. COMPRESSEIONE: Tramite 1 degli algoritmi negoziati all'inizio



3. INTEGRITY: Tramite aggiunta MAC ai dati compressi.

4. CONFIDENTIALITY: I dati vengono cifrati con **cifratura simmetrica**.

5. APPEND TLS RECORD HEADER

NB: Nelle versioni moderne, solitamente si aggiunge MAC **dopo** cifratura

• STRUTTURA RECORD PROTOCOL + CONTENT

- CONTENT TYPE: Definire il msg type del livello superiore.
- TLS VERSION:
- LENGTH: Massimo **16 KB**
- PAYLOAD
- MAC: Usa un sequence number implicito per mitigare i replay attacks.

General TLS record format	
Offset	Order
0	0
6	1 Content Type
8	2 Length (size)
12	3 Legacy version (negotiated)
16	4 Protocol message(s)
20	5 Message Authentication Code (optional)
24	6 Padding (one byte only)

TLS > NAMING CONVENTION

• FNO A TLS 1.2:

TLS -<key exchange>-<authn>-with-<Encryption>-<hash-fn>

- Example: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 - ECDHE: Elliptic Curve Diffie-Hellman Ephemeral (key exchange)
 - RSA: Server authentication using RSA signature
 - AES_128_GCM: Encryption with 128-bit AES in Galois/Counter Mode (authenticated encryption)
 - SHA256: Hash function for HMAC and key derivation



• DA TLS 1.3

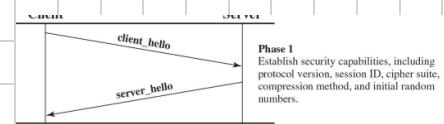
TLS-<Encryption>-<hash-fn> ▪ Example: TLS_AES_128_GCM_SHA256

Key exchange e Authn sono negoziati separatamente.

TLS > HANDSHAKE [1.2]

1. SCAMBIO & NEGOZIAZIONE PARAMETRI

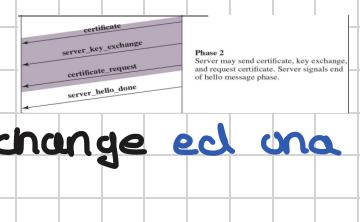
Il client manda i suoi parametri (algoritmhi supportati) al server che risponde con i suoi parametri negoziati.



NB: Si scambiano anche numeri casuali per evitare replay attacks

2. SERVIZI: SCAMBIO INFO AGGIUNTIVE + READY

Il server puo' inviare: il suo certificato, key exchange ed una



richiesta del certificato del client.

pronto

// deve inviare: messaggio di fine hello per segnalare di è V.

3. CLIENT: SCAMBIO INFO Agorawillie

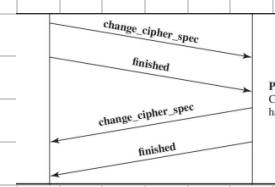


Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Il client può mandare: certificato o una verifica del certificato

// deve mandare: le sue chiavi

4. CLIENT & SERVER CHANGE CYPHER SPEC



Phase 4
Change cipher suite and finish handshake protocol.

Entrambi si scambiano le cypher spec e segnalano che sono pronti

Il tutto dura 2 RTT, con performance ridotte

TLS > GESTIONE CHIAVI

- PRE-MASTER SECRET:

Chiave iniziale negoziata all'inizio e scambiata usando

Diffie-Hellman (senza trasmissione web) o RSA (con i cert).

- MASTER SECRET

Generata a partire dal pre-master secret usando una funzione che randomizzerà uscendo i num. casuali scambiati in hand-shake

Serve per implementare la forward secrecy

- FW SECRECY:

Proprietà di sic. che prevede di non lasciare la stessa chiave

su tutte le comm. per evitare che gli attaccanti conoscano

subito tutti i messaggi se nel futuro riescono a rompere chiave

- SESSION KEYS: Chiavi di Client/Server derivate da Master Secret

Alla fine il certificato viene usato solo come verifica e non come trasmissione.

EPHEMERAL DIFFIE - HELLMAN >

Curve Ellittiche

EDH / EC DH sono protocolli di scambio chiavi sicuri basati su formule matematiche per scambiare le chiavi senza ri-trasmetterle.

FUNZIONAMENTO: Dati A, B comunicatori

1. A e B scelgono e si scambiano α, q tali che

→ siano primi → α sia prime root di q

2. I due generano x_A, x_B priate tali che $x_A < q, x_B < q$

3. $\equiv \equiv$ y_A, y_B public tali che

$$y_A = \alpha^{x_A} \pmod{q}$$

$$y_B = \alpha^{x_B} \pmod{q}$$

4. Entrambi si calcolano la chiave condivisa con:

$$k = (y_B)^{x_A} \pmod{q}$$

$$k = (y_A)^{x_B} \pmod{q}$$

Così facendo arrivano alla stessa chiave senza trasmetterla

TLS > HANDSHAKE 1.3

• 1 RTT: Si mandano i dati di DH assumendo che anche il server + RTT TCP usi la stessa modalità collaborando in 1 RTT

• 0 RTT: Si ri-usa una connessione precedente specificando nell'hello già i dati insieme a un resumption secret.
È vulnerabile a **replay attack**.

• 0 RTT con cookie

da 1.3

Si usa un cookie cifrato contenente lo status della sessione che viene generato dal server ed inviato al client. Il client se vuole riprendere manderà il ticket nelle connessioni future, che verrà decifrato e validato dal server.

SERVER NAME INDICATION >

È un dato aggiuntivo specificato dal client durante la fase di client hello (in chiaro) contenente hostname del server.

È necessario perché il server http può servire diverse applicazioni quindi deve capire quale servizio e quale cert. rilasciare. (Virtualhost)
Ci sono alternative ma non vengono usate.

L08 >

-> **Intro SSH:**

- > Che cos'è?
- > Qual'è il suo scopo principale?
- > Cosa sostituisce?
- > Su cosa si basa il suo sistema di AuthN?

-> **Struttura:**

- > Su quale protocollo di trasporto è basato?
- > Da quali layer è formato? Cosa fa ogni layer?
- > È considerato un protocollo sicuro?

-> **Configurazione:**

- > Come avviene la configurazione di una connessione SSH? Definisci tutti gli step.
- > È simile ad altri protocolli già visti?

-> **Autenticazione:**

-> AuthN della connessione:

- > In quali modi si puo' autenticare?
- > Che cos'è T.O.F.U? Cosa usa?

-> AuthN dell'utente:

- > In quali modi si puo' autenticare?
- > Ci sono dei modi obbligatori? Se sì quali?
- > Ci sono dei modi facoltativi? Se sì quali?

SSH > Porta 22

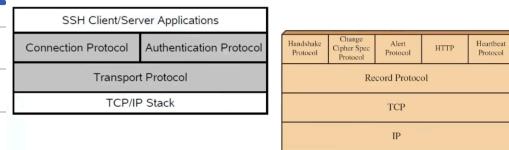
È un protocollo usato per il login e accesso a shell remota nato per sostituire protocolli insicuri come telnet.

Usa la public key come sistema di authN.

• STRUTTURA

SSH è posizionato sopra TCP e offre:

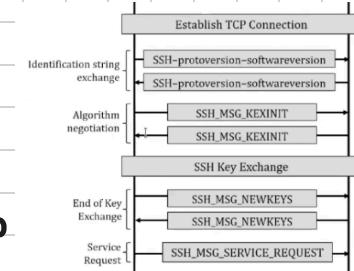
- TRANSPORT PROTOCOL: Come protocollo di trasporto sicuro.
- AUTHN PROTOCOL: Per authN l'utente
- CONNECTION PROTOCOL: Per stabilire 1+ canali di conn.



È uno dei più sicuri ma anche vulnerabile se configurato male.

• CONNESIONE

Avviene in maniera simile a TLS con conn. TCP, scambio versione SSH, negoziazione algo, scambio chiavi e scambio dati.



• AUTHN: CONNESSIONE

Per autenticare la connessione client/server si può usare

- un database locale con le chiavi pubbliche.
- ≈ certificato da CA

Solitamente si usa TOFU (si chiede se ci si fida la 1 volta e si salva la chiave).

NB: Questo crea una finestra di attacco all'inizio per la PK trasm.

• AUTHN: USER

È orchestrata dal server che guida il processo. Può essere

- [OBB.] PUBLIC KEY: Challenge based authN in cui il client trasmette [ES: Github] la privata key che il server verifica con pub K
- USERNAME + PSW [opt]

- HOST BASED [OPI]

- CONNECTION PROTOCOL

Una volta autenticati, si possono creare più canali di comm.
per fare multiplexing fra vari messaggi di app. level.

- TUNNELLING: Meccanismo che consente di creare un tunnel
dalla propria porta al server remoto.

SSH ➤ SINTASSI COMANDI

- LOCAL: -L

`ssh -L source :xxxxx remote :yyyyy user@server`

- **source :xxxxx** : Specifica l'host e la porta di origine da cui fare forwarding.
'source' si può omettere

ES: localhost: 8080

- **remote :yyyyy** : Specifica il remote e la porta a cui fare forwarding.

- localhost: 80: localhost del ssh server
- 192.168.1.0: 80: IP privato nella rete del

- **user@server** : Specifica il server ssh di destinazione.

- REMOTE : -R

`ssh -R remote :yyyyy local :xxxxx user@server`

- **remote :yyyyy** : Specifica il remote e la porta da cui fare forwarding.

- **local :xxxxx** : Specifica l'host e la porta di origine a cui fare forwarding.

'local' si può omettere

- **user@server** : Specifica il server ssh di destinazione.

L8Q >

- SSH: ESEMPIO DI TUNNELLING CON PORT FWD
 - ELENCA GLI STEP CON LE PORTE
 - QUANTE SONO LE CONNESSIONI TCP?
 - COME FUNZIONA L'INCAPSULAMENTO? VALE PER TUTTI I PROTOCOLLI?
 - SI PUÒ INVERTIRE UN TUNNEL?
- SICUREZZA A LIV. 2:
 - PERCHÉ È NECESSARIA?
 - AUTH: CON WIFI
 - QUALI SONO GLI ATTORI COINVOLTI E COME COMMUNICANO.
 - ELENCA GLI STEP DI AUTHN WIFI
 - AUTH: CON CELLULAR
 - COME FUNZIONA INT? È SIMILE A QUANDO USCO CON WIFI?
 - ELENCA GLI STEP DI AUTHN CELL.

SSH >

• ESEMPIO DI TUNNELING CON PORT FORWARDING

TCP

Ci sono 3 connessioni in questo esempio:

(1) La connessione interna su localhost

per collegarsi all' ssh client: localhost:xxxxx → localhost:80

(2) La connessione fra ssh client ed ssh server per effettuare il tunnelling: client :xxxxx → server: 22 [wupssle]

(3) La connessione fra il bastion della rete e il server finale
bastion: xxxx → server.f.: 80

In fase di comm., l'incapsulamento prevederà di prendere la parte dati di (1) per metterla in un msg. SSH.

È possibile fare anche il contrario col flag -R.

SICUREZZA A LIVELLO 2 >

È necessaria per garantire la CIA nelle comm. fra nodi soprattutto in WiFi dove il comm. link è una banda di frequenza.

• AUTHN: IN WI-FI

Aviene tramite un AUTHN SERVER (AS)

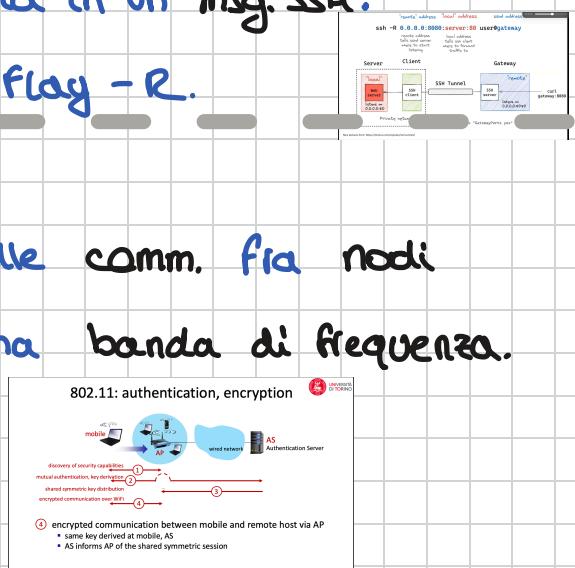
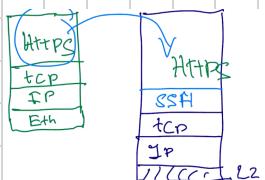
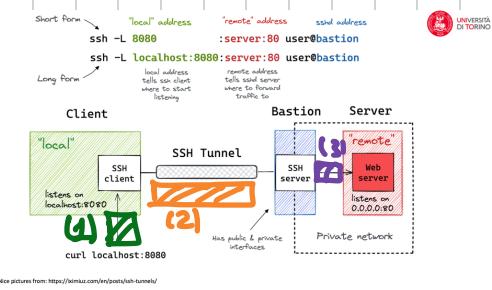
che comunicherà col nodo attraverso l'AP usando il protocollo EXTENSIVE AUTHN PROTOCOL (EAP) per la validazione della password basata su chiave simmetrica.

• STEP

1. HOST e A.P. negoziano un canale di comunicazione **cifrato** senza AuthN.

2. HOST e A.S. negoziano un altro canale **cifrato**, effettuando una mutual authN basata su:

- secret condiviso: password



- nonce: numeri pseudo casuali per prevenire replay ATC.
- hash: Per garantire l'integrity e authN del messaggio

La mutual AuthN consente la derivazione della chiave symm.

3. AS invia la sym. key all'AP.

Le chiamate successive non passeranno più da AuthN Server

• KEY DERIVATION IN WPA/3

1. AS genera un NONCE_{AS} e lo invia all'host.

2. host genera

- NONCE_{host}

- $K_{AP-HOST}$: Basata da NONCE_{host} e NONCE_{AS}

3. host invia NONCE_{host} con HMAC basato su NONCE_{AS} e il secret condiviso

4. AS riceve NONCE_{host} e deriva $K_{HOST-AP}$ post controllo HMAC.

• AUTHN: IN CELLULAR

AS è formato da M.N.E (foreign net) e H.S.S. (home net)

L'U.E. usa i key per la comunicazione con eNodeB

e i key (secret) comune con il suo H.S.S.

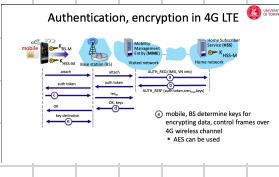
• STEP.

1. U.E. invia il suo IMSI e informazioni sulla foreign network alla propria home network (H.S.S.) usando il Foreign M.N.E (attach)

2. H.S.S. restituisce

- auth-token: Informazione cifrata dal H.S.S. con secret condiviso usata da U.E. per capire se la risposta arriva da H.S.S.

- xres: La expected response che U.E. deve inviare per autenticarsi. Viene salvata dal H.S.S. della



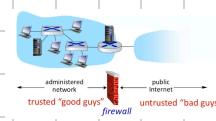
foreign network

3. U.E. riceve **solo** auth-token e genera res_m , nello stesso modo con cui è stato generato x_{res} e lo invia alla M.N.E.
4. M.N.E (foreign) confronta x_{res} e res_m e se sono **uguali** genera delle chiavi che saranno derivate da $nodeB$ e U.E.
5. U.E. e $nodeB$ derivano delle chiavi simmetriche per la comunicazione.

- FIREWALL
 - DEFINIZIONE
 - CATEGORIE
 - SEGMENTATION
- SEGMENTAZIONE RETI
 - COS'È E A COSA SERVE
 - QUALI SONO GLI ATTORI COINVOLTI
 - COS'È E COM'È DIVISO IL PROCESSO DI FILTERING

FIREWALL >

Componente di rete usato per isolare una rete di "trusted devices" da altre reti gestendo il passaggio o meno dei pacchetti in transito.



• CATEGORIE DI FIREWALL:

- STATELESS: Controlla i pacchetti senza memoria dello stato
 - Es: Port block torrent
 - ✓ Non richiede risorse particolari
 - ✗ È meno efficace.
- STATEFUL: Controlla i pacchetti con memoria dello stato generale della comm.
 - ✓ È più efficace [leg: Tiene traccia di conn. TCP]
 - ✗ Richiede più risorse [leg: memoria aggiuntiva]

- APPLICATION GATEWAY:

• STATELESS : CONFIGURAZIONE FILTRI

Si basano su:

- SOURCE IP
- DESTINATION IP
- PROTOCOL: = Upper layer IP
- SOURCE PORT
- DEST PORT
- ICMP TYPE
- TCP SYN
- TCP ACK

• STATEFUL: FILTRI

Necessita il tracciamento delle connessioni attive TCP in delle tabelle. Viene impostato un timeout sulla connessione per prevenire attacchi basati sul sovraccarico della tabella.

Es: ACL Switch aumentata che boccia ACK senza conn stabilita prima.

Access Control Lists							
ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs							
action	source address	dest address	proto	source port	dest port	flag bit	check connection
allow	222.22.16	222.22.16	TCP	> 1023	80	any	
allow	outside of	222.22.16	TCP	80	> 1023	ACK	
allow	222.22.16	outside of	UDP	> 1023	53	—	
allow	inside of	222.22.16	UDP	53	> 1023	—	
deny	all	all	all	all	all	all	

Access Control Lists							
ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs							
action	source address	dest address	proto	source port	dest port	flag bit	check connection
allow	222.22.16	222.22.16	TCP	> 1023	80	any	
allow	outside of	222.22.16	TCP	80	> 1023	ACK	✗
allow	222.22.16	outside of	UDP	> 1023	53	—	
allow	inside of	222.22.16	UDP	53	> 1023	—	
deny	all	all	all	all	all	all	

FIREWALL > SEGMENTAZIONE

Processo di divisione della rete in compartimenti diversi con diversi livelli

di sicurezza, i quali sono protetti fra di loro con firewall

• ATTORI:

- DMZ: De-Militarized Zone EG: Web Server

Zona senza particolari protezioni esposta verso l'esterno

- PROTECTED NETWORK EG: Server e database

Area di rete protetta da particolari misure di sicurezza

• PROCESSO DI PACKET FILTERING: NEFTILTER

Consente di specificare policy di filtro del pacchetto in diverse fasi del pacchetto:

- TIPI DI LAYER:

→ CONN TRACKING: S.O. traccia la config. della connessione.

→ MANGLE:

→ NAT

→ FILTER

- TIPI DI MONENTI:

→ PRE-ROUTING: Pacchetto appena ricevuto

→ FORWARD: Pacchetto da inviare ad altri host.

→ INPUT/OUTPUT: Della comm. con l'host

→ POST ROUTING:

• ESEMPIO DI FIREWALL: IPTABLES

```
root@mail-bigdata:~# iptables -nL -v -t filter
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt in     out    source          destination
 5913K 6109M ACCEPT  0   --  lo      *       0.0.0.0/0        0.0.0.0/0
20936 2788K ACCEPT  1   --  *      *       0.0.0.0/0        0.0.0.0/0
458M 1041G ACCEPT  0   --  *      *       0.0.0.0/0        0.0.0.0/0
556K 177M ACCEPT  0   --  bond0   *       192.168.128.0/23  0.0.0.0/0
103K 5375K ACCEPT  0   --  bond0   *       130.192.95.208/28 0.0.0.0/0
6831 390K ACCEPT  6   --  bond0   *       0.0.0.0/0        0.0.0.0/0
25357 1488K ACCEPT  6   --  bond0   *       0.0.0.0/0        0.0.0.0/0
1145K 62M ACCEPT  6   --  bond0   *       0.0.0.0/0        0.0.0.0/0
1831K 107M DROP   0   --  *      *       0.0.0.0/0        0.0.0.0/0
                                                 state RELATED,ESTABLISHED

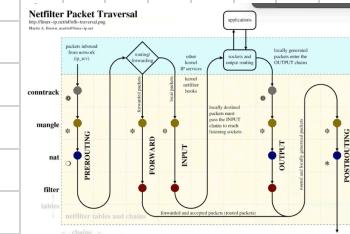
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt in     out    source          destination

Chain OUTPUT (policy ACCEPT 431M packets, 569G bytes)
 pkts bytes target  prot opt in     out    source          destination
```

- iptables.... -t

Specifica il livello su cui trovare la tabella: filter

- CHAIN:



Specifica a che operazione siamo: INPUT: pre consegna

- prot:

È il protocollo : 1: ICMP , 6: TCP

- target: Decide se accettare o rifiutare il pacchetto.

- in: È l'interfaccia di rete

- source/dest: gli IP

- state RELATED, ESTABLISHED:

Connessioni già stabilite (es: TCP)

- dpt: dest. port.

• ESEMPIO: Tabelle netfilter

```
nft add table inet filter
nft add chain inet filter input "{ type filter hook input priority 0 ; policy drop ; }"
nft add rule inet filter input ct state established,related accept
nft add rule inet filter input iif lo accept
nft add rule inet filter input tcp dport { 22, 80, 443 } accept
nft add rule inet filter input drop # redundant
nft list ruleset           # view all active rules
```

- Creates a table in the inet family (handles both IPv4 and IPv6)
- Defines an input chain with default policy DROP (deny all by default)
- Allows established connections, loopback traffic, and SSH/HTTP/HTTPS
- Final explicit drop rule (redundant due to policy, but explicit)

• ESEMPIO: IP TABLES

```
# Set default policy to DROP
iptables -P INPUT DROP

# Accept established and related connections
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Accept all traffic on loopback interface
iptables -A INPUT -i lo -j ACCEPT

# Accept SSH, HTTP, and HTTPS
iptables -A INPUT -p tcp -m multiport --dports 22,80,443 -j ACCEPT

# Drop everything else (redundant due to default policy)
iptables -A INPUT -j DROP
```

- -j: indica se accettare o meno

- -P: protocollo

- -P: Default

- -i: net. interface (lo = loopback)

LCTY

- FIREWALL
 - QUALI SONO LE AIRE TIPOLOGIE DI FIREWALL CHE AGISCONO A LIVELLO APPLICATIVO.
- DETECTORS:
 - COSA SONO
 - TIPOLOGIE PRINCIPALI
 - COME OPERANO?
 - COME SONO COLLOCATE?
 - QUALI SONO I METODI DI INSPECTION
 - COS'È IL SIEM?
 - == EDR?
- PRATICHE OP. SEC
 - QUALI SONO?

FIREWALL >

Tipologie aggiuntive / simili di firewall

- APPLICATION GATEWAY:

Svolge un ruolo simile al bastion nel tunnelling delle richieste verso servizi remoti con application gateway.

- APPLICATION-BASED FIREWALL Es: Cloudflare rileva cross-site-script.

Firewall che agisce a livello application tramite **ispezione del contenuto dei messaggi scambiati.**

DETECTORS >

Parte dell'operational security è la possibilità di rilevare e prevenire attacchi con i detector.

- NIDS: NETWORK INTRUSION DETECTION SYSTEM

Agisce tramite strutture installate che agiscono nei pacchetti scambiati.

→ I_DS: Effettua solo detection degli attacchi (con logging eventuale)

→ I_PS: \neq detection e prevention \neq (Es: Application Firewall).

Non sono usati tanto perchè ormai tutto il traffico è cifrato.

- HIDS: Host Intrusion Detection System

Forma di detection e prevention che avviene tramite l'installazione di software specializzato con permessi di lettura dei pacchetti; ↳ Backdoor

- LOGGING: Usato sia da NIDS che da HIDS, deve essere

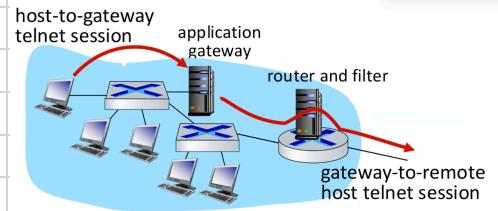
configurato correttamente per evitare che stia nella stessa macchina.

Sia per NIDS che per HIDS, si può operare tramite

- DEEP PACKET INSPECTION: Ispezione del contenuto dei pacchetti

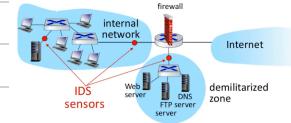
- CORRELATION INSPECTION: Sulla base dei comportamenti dei singoli nodi

Soltanamente ↑ perché il traffico è cifrato.



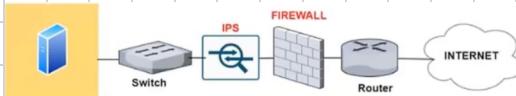
● COLLOCAMENTO

A livello di collocamento, vengono messi sia nella frontiera che anche dentro le reti (DMZ, privata).



● ESEMPI DI SETUP

Entrambi agiscono sugli switch con la copia dei pacchetti



● DETECTION METHODS

1. SIGNATURE BASED: Usa una serie di known attack fingerprints per rilevare gli attacchi tramite blacklist.

2. SPECIFICATION BASED: Consente solo a determinati pacchetti compatibili ad una whitelist il passaggio

3. ANOMALY BASED: Rileva attacchi tramite la comparazione
→ richiede tanti dati
- Poco usato con un modello di "utente corretto" def. sugli utenti con

4. BEHAVIORAL: Sulla base del comportamento del nodo.

Attaccanti e difensori giocano a gatto e topo con definizione di regole ed evasione da esse.

DETECTORS > LOGGING

Soltanamente si cerca di centralizzare la gestione dei logs tramite i SIEM (Security & Information Event Management). (Elastic search)

La raccolta deve tenere conto dello spazio richiesto per loggare tutto e requisiti legali di retenzione del logging.

EDR > ENDPOINT DETECTION & RESPONSE → o NIDS

Sistemi che includono HIDS con response automatica a chi sta monitorando (es: CrowdStrike)

PRATICHE DI OP. SECURITY >

● VULNERABILITY SCANNING : Tramite tool appropriati per il test del syst.

● HONEY POT: Macchine "lasciate scoperte" per poter essere attaccate in modo da capire se ci sono attacchi in corso.

Soltanente funziona su sistemi automatici.

ESEMPI >

● SNORT:

Sistema di IDS/IDS configurabile con delle regole nella forma:

action protocol source IP source Port → dest IP dest Port (opts)

- Example rules:

```
alert tcp any any -> 192.168.1.0/24 80 (msg:"HTTP GET detected"; content:"GET"; sid:1001;)
alert tcp any any -> any 22 (msg:"SSH connection attempt"; flags:S; sid:1002;)
alert icmp any any -> any any (msg:"ICMP ping detected"; itype:8; sid:1003;)
```

In cui

- **action**: puo' essere 'alert', 'drop', 'log', 'pass'
- **flags**: S=Syn A=Ack
- **content**: fa match sul payload.

● SURICATA:

Evoluzione di snort.

- VIRUS:

- COS'É
 - STRAT & COUNTER

- WORM

- COS'É
 - STRAT & COUNTER

- TROJAN & BACK DOOR

- COS'É
 - STRAT

- ROOT KIT

- COS'É
 - STRAT

- RANSOMWARE

- COS'É
 - STRAT

- ALTRI MALEWARE

SOFTWARE MALEVOLO

VIRUS >

È il più vecchio software malevolo. Il suo obiettivo è quello di infilarsi negli eseguibili o codice boot, per poi diffondersi.

- STRATEGIE (ES.):

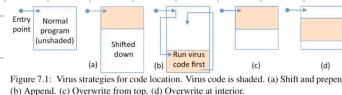


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend.

(b) Append. (c) Overwrite from top. (d) Overwrite at interior.

Il virus sposta il codice in basso (**Shift down**) e su ci mette un per poi restituire il main che andrà a decifrare e avviare il virus vero e proprio, controllo.

- COUNTER: Sui di anti-virus che verificano se i file sono cambiati (con hash) o tramite ricerca dei virus (Diz. di hash del virus).

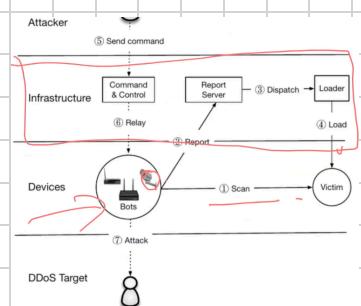
Solitamente sfruttano le azioni dell'utente.

WORM >

SW malevolo stand-alone che si propaga usando la rete

- STRATEGIA (SW. MIRAI)

1. Un nodo worm manda messaggi nella rete per cercare di entrare in dispositivi deboli.
2. Infettata la vittima, il nodo fa da report a dei nodi di report service e a un loader per il caricamento del worm e censimento nella botnet.
3. La botnet segue gli ordini dai nodi command & control dell'ATK.



- CONTROMIURE: Localizzare e buttare giù il provider nell'infrastr. (spazio in cloud)

TROJAN & BACKDOOR >

- TROJAN: ES: App fraudolenta sullo store, Softonic XD

Software autonomo che cerca di farsi passare per software benevolo.

- STRAT: Post-installazione, prova a rubare i dati.

- BACKDOOR:

Punto di accesso lasciato da un attaccante dopo aver

completato un attacco, per consentire l'accesso remoto saltando AuthN e AuthR.

- REMOTE ACCESS TROJAN (RAT): Trojan + Backdoor.

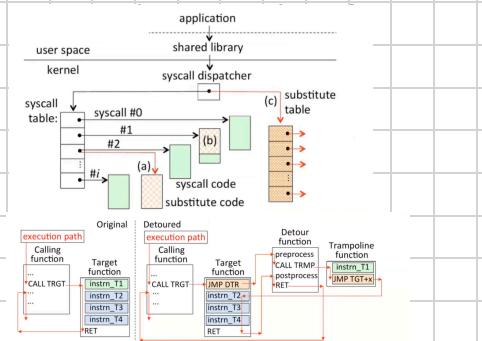
ROOTKIT >

È un malware standalone che viene usato per nascondere la sua presenza e quella dei malware con cui è incluso.

- STRAT:

In kernel space prova ad attaccare le system calls per rendersi invisibile con:

- sostituzione dell'intera tabella di sys.calls
- shift down del codice di una sys.call
- sostituzione totale



Per lo shift down / inline hooking si usa una detour function che faccia da wrapper del codice malevolo.

RANSOMWARE >

Malware autonomo che appena infetta il PC prova a cifrare il numero maggiore di files per poter chiedere un riscatto.

- STRAT: WannaCry

1. Attaccante genera (pub_m , priv_m)

2. Per ogni vittima v , genera (pub_v , $E(\text{priv}_v, \text{pub}_m)$), ovvero cifra la priv. key della vittima con la sua chiave pub.

3. Ottenuto l'accesso, per ogni file F :

3.A: Genera una chiave sim. casuale K

3.B: cifra K con pub_v

3.C: cifra il file con la chiave K , modificando il file stesso aggiungendo K cifrato e il contenuto cifrato

h. se la vittima paga: l'attaccante decifra con la sua

private key prv_m , la private key dell'utente prv_{vi}

Viene usata una cifratura simm. per cifrare velocemente.

ALTRI MALWARE >

- ADWARE: Mostre pubblicità
- CRYPTOMWARE: Focalizzato a usare risorse per minare
- KEYLOGGER: ↪
- SPYWARE: Malware pensato solo per spiare
- LOGIC BOMB: Malware triggerato solo da specifiche condizioni.