Sicurezza di Sistema e di Rete

Concetti, minacce, difese, crittografia e controllo accessi

Corso: Sistemi di Calcolo 2
Docente: Riccardo Lazzeretti
Riferimenti: W. Stallings, *Operating Systems: Internals and Design Principles* (cap. 14–15); *Cryptogr*

Indice

1 Cosa significa "sicuro"

1.1 Principi iniziali (dalle slide)

- La sicurezza non è (solo) un **prodotto** (es. firewall), ma un **processo** continuo da gestire.
- **Nulla è al 100% sicuro**.
- La sicurezza di un sistema è quella del suo **anello più debole** (weakest link).
- La **security by obscurity** non funziona.
- La **crittografia** è potente ma **non basta** da sola.
- **Non affidarti agli utenti**: gli utenti ignorano gli avvisi, cliccano comunque ("dancing pigs").

In sintesi (dal testo): La sicurezza è un **processo** che va gestito correttamente: nulla è completamente sicuro, ma bisogna lavorare per avere un livello il più alto possibile. Cercare sicurezza tenendo nascoste le implementazioni dei propri sistemi non è sempre la scelta giusta: rendendo pubblico il proprio codice si può sperare che qualche utente trovi falle e le segnali. La sicurezza di un sistema è pari alla sicurezza del **componente meno sicuro**. La **crittografia** è un ottimo strumento ma non basta. Il **cybercrime** è un problema reale che va affrontato.

2 Definizioni e obiettivi

2.1 Computer Security (NIST)

Protezione di un sistema informativo per preservare **Confidenzialità**, **Integrità**, **Disponibilità** di risorse (HW, SW, firmware, dati, telecomunicazioni).

2.2 Triade CIA + altri concetti

Confidenzialità prevenire accessi/divulgazioni non autorizzati.

Integrità dati e sistema modificabili solo in modo autorizzato; sistema svolge le funzioni previste.

Disponibilità accesso/uso non negato agli utenti autorizzati.

Autenticità fiducia nell'origine/validità di messaggi e soggetti.

Accountability tracciabilità univoca delle azioni per analisi forense e dispute.

Precisazioni (dal testo): La *Computer Security* è la protezione di un sistema di calcolo per preservare l'**integrità**, la **disponibilità** e la **confidenzialità** di informazioni e risorse. La triade **CIA** comprende *Confidentiality*, *Integrity* e *Availability*. Si aggiungono due concetti fondamentali: **Authenticity** (verificare se una informazione è genuina e se proviene dal mittente indicato) e **Accountability** (poter tracciare le responsabilità di una falla nella sicurezza). Le possibili minacce riguardano **hardware**, **software**, **dati** o **comunicazioni**.

3 Minacce e attacchi

3.1 Ambito e tipologie

Figure (descritte): *scope* della sicurezza e panoramica minacce/attacchi.

Attacchi passivi intercettazione/analisi traffico; difficili da rilevare, prevenzione tramite **cifratura**.

Attacchi attivi modifica/iniezione: *replay*, *masquerade*, *modification*, *denial of service*.

3.2 Pattern d'intrusione

Hackers (status/thrill); **Insider** (più difficili da rilevare: conoscenza e privilegi); **Criminal enterprise** (coordinati, obiettivi specifici); **APT** (sponsor statale, ciclo a 8 fasi: ricognizione → weaponization → delivery → exploit → install → C&C → obiettivi → occultamento).

Categorie di attacco (dal testo): Gli attacchi si dividono in **attivi** (leggono, modificano, generano e distruggono informazioni) e **passivi** (leggono informazioni). Gli attivi possono essere: *replay* (catturo informazioni e le rimando), *masquerade* (mi spaccio per qualcun altro), *modification of messages*, *denial of service* (mando informazioni non richieste per intasare una rete).

Profili (dal testo): Un *hacker* cerca di inserirsi in un sistema per sfida personale, per rivelare informazioni nascoste o per cercare vulnerabilità e segnalarle. L'*insider attack* prevede di farsi assumere in una compagnia per fare spionaggio e rubare dati. Una *criminal enterprise* è un gruppo organizzato di hacker. Un *advanced persistent threat* è una criminal enterprise sponsorizzata da una nazione per avviare un attacco verso un'altra nazione.

4 Malware

4.1 Categorie e termini

Backdoor/trapdoor; Trojan; *mobile code* (Java/ActiveX/JS/VBScript); Virus (meccanismo d'infezione, trigger, payload; boot/file/macro); Worm; Rootkit (persistenti/memory; user/kernel mode).

Attacchi blended e multi-minaccia (es. **Stuxnet**: USB, vuln Windows, analisi/reti, escalation privilegi, componenti PLC).

Definizioni (dal testo): Un *malware* è un software malevolo. Una *backdoor* è una porta di servizio, spesso installata volontariamente per avere accesso secondario. Un *trojan horse* è un programma apparentemente innocuo che nasconde software malevolo. Un *virus* infetta processi e si riproduce. Un *multiple-threat malware* infetta vari file; un *blended attack* usa vari

metodi di infezione e trasmissione per diffondersi più velocemente e gravemente. Un *rootkit* fa ottenere permessi di amministratore; può essere *persistente* se sopravvive al reboot o *memory based* altrimenti, e operare in *user mode* o *kernel mode*.

5 Buffer overflow

5.1 Concetti base

Errore di programmazione: scrittura oltre un **buffer** (stack/heap/globali) ⇒ corruzione dati/-controllo/violazioni di memoria/esecuzione arbitraria.

Figure: stack frame, EBP/ESP, chiamate di funzione, sovrascrittura del return address, *shellcode*.

5.2 Difese

In compilazione: linguaggi sicuri; *safe libs* (`strlcpy`/`strlcat`/`snprintf`); stack canaries (terminator, random, random XOR).
Di sistema: **W⊕X**/NX/DEP (dati non eseguibili), **ASLR**, *return-to-libc*. Nessuna misura è risolutiva da sola: vanno combinate.

Dal testo: Il *Buffer Overflow* sfrutta errori nella programmazione per cui diventa possibile scrivere più dati della capacità disponibile nel buffer, sovrascrivendo locazioni di memoria adiacenti. Questo può portare alla corruzione dei dati, al trasferimento del controllo e all'esecuzione di codice dell'attacker. Per sfruttarlo bisogna trovare vulnerabilità nel programma, capire come il buffer è salvato in memoria e determinare il potenziale di corruzioni. Uno *stack* contiene *stack frame* (uno per ogni funzione chiamata) con variabili locali, parametri passati e *return address*; si possono utilizzare *frame pointer*. Sfruttando l'overflow, modificando il contenuto di un array nello stack si può sovrascrivere fino al *return address* e farlo puntare a *shell code*. Un linguaggio che si assicura che letture e scritture rimangano nei limiti del buffer impone *memory safety*. Le protezioni si distinguono in: **compile-time defense** (rende il programma resistente) e **stack protection mechanism** (rivela e risolve attacchi in programmi esistenti). Tecniche: *canarini* (controllo di un valore nello stack prima del ritorno), separare sezioni di memoria scrivibili/-leggibili (*WOX*) o dividere il codice eseguibile da quello modificabile (*DEP, Data Execution Prevention*).

6 Bot, botnet e DDoS

6.1 Bot e costruzione della rete di attacco

Bot = software che prende il controllo di host terzi (zombie/drone) con canale di **remote control** e meccanismo di **spreading** (scanning: random/hit-list/topological/local subnet).

6.2 Usi dei bot

DDoS, spam, sniffing credenziali, keylogging, diffusione malware, adware/BHO, attacchi IRC, manipolazione sondaggi/giochi.

6.3 Denial of Service

Flood ICMP/UDP/SYN; spoofing indirizzi sorgente; handshake TCP e **SYN spoofing**; DDoS gerarchici.

Difese: prevenzione/preemption (anti-spoofing, rate control, SYN cookies, blocco broadcast, servizi sospetti, puzzle, server replicati), rilevazione/filtraggio, *traceback*; piano di risposta con ISP, filtri standard, monitor/IDS.

Bot e botnet (dal testo): Un *Bot* si diffonde come un virus ma senza effetti indesiderati immediati. Prende il controllo di dispositivi per creare una *botnet* (zombie o drone) fino all'attivazione per lanciare attacchi non tracciabili. Caratteristiche: capacità di eseguire codice, *facile controllo da remoto*, *facilità di propagazione*. Sfruttano vulnerabilità comuni a molti sistemi; tramite *ping* e indirizzi IP casuali tentano di entrare in reti locali e propagarsi. Con una botnet si possono fare: *DDoS*, *spamming*, *sniffing traffic*, *keylogging*, diffusione di *malware*, installazione di pubblicità per remunerazione, modificare chat e manipolare giochi.

DoS/DDoS (dal testo): Un *DoS* prevede l'esaurimento delle risorse di un sistema (banda di rete, risorse di sistema o applicazioni) per renderlo indisponibile a richieste lecite. Esempio classico: *flooding di ping* da una rete con più capacità verso una con meno capacità. Altri casi: *ICMP flood*, *UDP flood*, *TCP SYN flood* (invio di SYN, il server resta in attesa dell'ACK che non arriva, la coda si esaurisce). Una *botnet* rende l'attacco più efficace e meno tracciabile: *DDoS*. Possibili contromisure: limitare ping al secondo o inserire *captcha*. I *service provider* potrebbero bloccare pacchetti con indirizzo IP mittente non corrispondente o tenere traccia dei percorsi dei pacchetti, ma è raro che implementino tali funzionalità.

7 Strumenti crittografici

7.1 Numeri casuali

TRNG (sorgenti fisiche) vs **PRNG** (pseudocasuali): requisiti di *randomness* e *unpredictability*.

7.2 Cifratura simmetrica

DES (56 bit, storico) e **3DES**; **AES** (128-bit block; chiavi 128/192/256).

Attacchi: *crittanalisi* e *brute force*. Blocchi vs *stream ciphers*.

7.3 Cifratura a chiave pubblica

Requisiti e algoritmi: **RSA**, **Diffie–Hellman** (scambio chiavi), **DSS** (solo firma), **ECC** (sicurezza equivalente con chiavi più piccole), cenno a *omomorphic encryption*.

Autenticazione a chiave pubblica e **certificati**.

7.4 Autenticazione messaggi e hash

MAC e **hash** sicuri: requisiti (one-way, collisioni deboli/forti), approcci d'attacco (crittanalisi, brute force), famiglia **SHA** (SHA-1; SHA-256/384/512). **Busta digitale** (*digital envelope*).

Crittografia (dal testo): È importante e necessaria ma non sufficiente. Alla base ci sono numeri casuali che nei computer sono in realtà *semi-casuali*. **Cifratura simmetrica**: si genera una chiave specifica per una comunicazione; un attacco *brute force* non è realizzabile per l'enorme spazio delle chiavi. I dati possono essere divisi in *blocchi* e cifrati separatamente, oppure trattati come *stream* da cifrare continuamente. **Cifratura asimmetrica**: si generano due chiavi, una pubblica e una privata; con la pubblica si cifrano i dati e solo chi possiede la privata può decifrarli; deve essere semplice creare le coppie ma impossibile risalire alla privata dalla pubblica. Esempi: *RSA*, *exchange algorithm*, *digital signature standard*, *crittografia a curva ellittica*, *homomorphic cryptography*. **MAC** (Message Authentication Code): si calcola con una chiave e si aggiunge al

messaggio; il ricevente ricalcola il MAC per verificare che coincida. Si usano *funzioni di hash* con output a lunghezza fissa indipendente dall'input.

8 Cifratura dei dati a riposo

Approcci: appliance backend, *tape encryption* a livello libreria, cifratura in background di laptop/PC.

9 Autenticazione

9.1 Password

Hash + **salt** (evita duplicati, rende più difficile l'*offline dictionary*, impedisce correlazioni multi-sistema). Schema UNIX: `crypt(3)` (DES); varianti moderne: **MD5-based**, **bcrypt** (OpenBSD).

9.2 Token

Memory card (magstripe, memoria interna); **smart card/smart token** (con CPU, interfacce manuali/elettroniche; protocolli statici/dinamici/challenge-response).

9.3 Biometria

Statiche (volto, impronte, geometria mano, retina, iride) e dinamiche (voce, firma). *Figura:* trade-off *costo vs accuratezza*.

Autenticazione (dal testo): **Password utente**: si salva l'*hash* della password; si usa un *sale* in modo da evitare hash uguali per password uguali; il sale è salvato insieme all'hash ed è usato nei tentativi successivi. **Token**: contengono informazioni per identificarci; una **smart card** ha anche potenza di calcolo. **Biometrie**: caratteristiche uniche e stabili nel tempo (facciali, impronte digitali, forma della mano, retina, iride, firma, voce); pericolo principale è che vengano "rubate" perché non si possono resettare; esistono anche biometrie comportamentali.

10 Controllo accessi

10.1 Modelli

DAC (per identità e regole), **MAC** (etichette/clearance), **RBAC** (per ruoli, standard NIST). Matrici/ACL/capability; estensioni della matrice d'accesso.

Figure: organizzazione della funzione di controllo accessi e rappresentazione RBAC.

Controllo di accesso (dal testo): Assegna il tipo di accesso in base alle circostanze o all'utente che lo richiede. Politiche: **DAC** – discretionary access control (controllo in base all'identità e possibilità di concedere ad altre identità); **MAC** – mandatory access control (controllo in base all'identità); **RBAC** – role-based access control (controllo in base al ruolo nel sistema, un utente può avere più ruoli).

11 Antivirus

11.1 Strategie

Prevenzione (ideale ma impossibile in assoluto); **Rilevazione** → **Identificazione** → **Rimozione** o ripristino da *backup* pulito.

Generic Decryption (GD): emulatore CPU + firma + controllo dell'emulazione per individuare virus polimorfici.

Digital Immune System (IBM → Symantec): risposta rapida a virus Internet-driven (mail integrata, codice mobile).

Antivirus (dal testo): Controlla i file prima di memorizzarli e, se malevoli, li rimuove salvo diversa indicazione dell'utente. Nei sistemi più grandi si usa un *digital immune system* che identifica potenziali virus e li invia a una macchina (interna o esterna) che li analizza, crea una *signature* e la distribuisce ai sistemi locali.

12 Firewall e IPS

12.1 Capacità e limiti

Choke point per policy, monitoraggio, NAT, VPN IPsec. Limiti: bypass, minacce interne, WLAN insicure, dispositivi portatili infetti.

12.2 Tipi di firewall

Packet filter (default deny/allow, regole su IP/porte/protocollo/interfaccia; debolezze: bugs TCP/IP, spoofing, source route, tiny fragments); Stateful inspection (directory di connessioni, verifica porte alte, numeri di sequenza); Application-level proxy; Circuit-level gateway; *socket-level gateway.* Posizioni e **firewall distribuiti**.

Firewall (dal testo): Blocca pacchetti non legittimi provenienti dalla rete. Può lavorare a livello di **trasporto** (bloccare porte), a livello di **rete** (bloccare indirizzi IP) o a livello di **applicazione**. La gestione può prevedere il blocco di tutti i pacchetti eccetto quelli esplicitamente consentiti oppure il blocco dei soli pacchetti specificati. I firewall sono generalmente numerosi in una rete: uno all'ingresso dalla rete esterna, uno su ogni macchina e altri eventuali fra gli switch.

13 Intrusion detection

13.1 Principi e tipi

L'intruso si comporta in modo **diverso** dall'utente legittimo (quantificabile). IDS efficace: deterrente, rilevazione precoce, raccolta informazioni per hardening.

Host-based IDS (anomaly: soglie/profili; signature: regole/pattern). Dati di audit: *native* vs *detection-specific.* **Network IDS/IPS**, **honeypot** (singolo o rete) e **deployment**.

Intrusion detection e honeypot (dal testo): Si può monitorare il traffico di ogni host per creare statistiche e riconoscere immediatamente livelli anomali dovuti a un'intrusione. Una *honeypot* è un sistema (singolo o rete) che simula la rete interna ed è posta generalmente prima del firewall verso l'esterno per attirare attacchi e raccoglierne informazioni; può essere inserita anche all'interno della rete per verificare quali attacchi sono riusciti a superare il firewall.

14 Qualità del software vs sicurezza

La qualità mira ai *guasti accidentali*; la sicurezza affronta input **scelti dall'attaccante** (distribuzioni rare).

Defensive/secure programming: assumere nulla, validare tutto, gestire input/ambiente/config, evitare **injection**.

Security by design (dal testo): La sicurezza può essere garantita solo da **codice di buona qualità** → *security by design*.

15 Hardening del sistema operativo

Pianificare installazione/configurazione/aggiornamento/manutenzione di OS e applicazioni chiave (Linux/Windows).

Top 4 mitigazioni efficaci: (1) patch OS; (2) patch applicazioni terze parti; (3) *least privilege* (privilegi admin solo a chi ne ha bisogno); (4) *whitelisting* applicazioni approvate.