SICUREZZA

Davide Belcastro

June 19, 2023

1 Introduzione

Appunti relativi al Corso di Sicurezza,tenuto nell'anno accademico 2022/2023 dal docente Casalicchio,terzo anno Informatica presso L'Università di Roma La Sapienza.
Gli appunti sono scritti in italiano tranne il primo capitolo che è scritto in inglese.
NON SONO VENDIBILI.

2 The mean of cybersecurity

The cybersecurity is the set of measures and controls that ensures the Confidentiality, Integrity and Availability of the computer's resource like: hardware,software and firmware. The cybersecurity, also, ensures that the information are elaborate,saved, transmitted.

2.1 CIA

- Confidentiality
  - Data: ensures that unauthorized persons do not see any other private data
  - Privacy: ensures that people only see their own information
- Integrity
  - Data: ensures that information is modified only by authorized persons
  - System: ensures that the system works without external manipulation
- Availability: ensures that the system works and that the system is availability when an user wants it

There are other two important concepts: Authenticity and Accountability.
Authenticity: make sure the user is who they claim to be
Accountability: a person's action must be uniquely traced back to that person

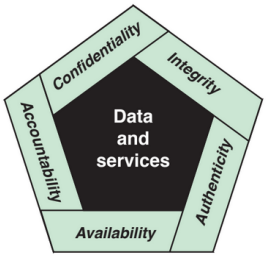


Figure 1: CIA

2.2 Three levels of impact

In case of loss of one or more elements there are three levels of impact.

- Low: the loss could have an negative effect on the organization.
  The organization will continue to work but the efficenty is reduced.
- Moderate: the loss could have an negative effect on the organization.
  The organization will continue to work but the efficenty is reduced and there is an moderate money loss,the persons are not in danger of life.
- High: the loss could have an disastrous effect on the organization.
  The organization stop to work, there is an high money loss,the persons are in danger of life.

3 Computer Security Challenges

There are more challenges in Cybersecurity. For example

- complex systems are needed to guarantee all things
- the securuty's alghoritm must be counterintuitive
- the computer security are needed of an continue check
- an hacker must find one way to enter in the system,the defenders must find all way to not allow at the hacker to enter
- the computer security must be integral part of the project design
- more user see an safety system like little user friendly

4 Computer Security Model

Asset: resource that the user want to protect.
The asset could be classificate in different thing

- Hardware: computer system
- Software: operating system
- Data: file and database
- Network:link to comunicate,WAN,LAN,...

Some terms:

- Adversary: individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.
- Attack: an attivity that want stop,destroy, the system's resources
- Countermeasure: a device or techniques that has as its objective to stop the attack
- Risk: measure of the extent of the threat
- Security Policy: a set of criteria for the provision of security services
- Asset: resource that the user want to protect
- Threat: any circumstance that may have a negative impact
- Vulnerability: weakness of a computer system

In our case, we must protect the Network.

4.1 Network vulnerability

- Corrupted (loss of integrity)
- Leaky (loss of confidentiality)
- Unavailable or very slow (loss of availability)

Una minaccia rappresenta un potenziale danno ad una risorsa, un attacco è una minaccia che viene eseguita e se ha successo porta ad una violazione indesiderata della sicurezza.
L'agente che compie l'attacco è chiamato "attaccante" o "agente che minaccia".

Two type of attack

- Passive: attempt to gather information/resource without to modify the resource or its own operation
- Active: attempt to modify the resource or its own operation

Other classification

- Internal: "insider", the insider is a authorized person that performs unauthorized actions
- External: "outsider", unauthorized person that performs unathorized actions

5 Conseguenze delle minacce

- Divulgazione non autorizzata: un'entità ottiene accesso a dati per il quale non è autorizzata
  Attacchi
  - Esposizione: i dati vengono rivelati direttamente ad un entità non autorizzata (intensionale:insider rilascia dati sensibili, o per errore umano/hw o SO)
  - Intercettazione: un'entità accede direttamente ai dati che viaggiano in rete(attacco comune in LAN,qualsiasi dispositivo connesso in LAN può ricevere copia dei pacchetti destinati ad un altro dispostivo)
  - Inferenza: un attaccante accede indirettamente ai dati(analisi del traffico in rete,informazioni sul db dopo query ripetute)
  - Instrusione: un attaccante ottiene accesso ai dati sensibili bypassando sistemi di sicurezza
- Inganno: un'entità autorizzata riceva dati falsi credendo siano veri
  Attacchi
  - Macheramento: un 'entita non autorizzata ottiene l'accesso ai dati sensibili spacciandosi per un'altra entità autorizzata
  - Falsificazione: dei dati che ingannano l'entità (autorizzata) che li legge
  - Ripudio: entità rinnega di aver inviato/ricevuto dati
- Interruzione: interruzione del corretto funzionamento di un sistema
  Attacchi
  - Interdizione: interrompe il funzionamento del sistema disabilitando uno dei suoi componenti
  - Corruzione: modifica il funzionamento del sistema
  - Ostruzione: ostacola il funzionamento,sovraccaricandolo o ostacolando la comunicazione
- Usurpazione: controllo del sistema da parte di entità non autorizzate
  Attacchi
  - Appropriazione indebita: entità assume il controllo fisico in modo non autorizzato di una risorsa
  - Uso improprio: una componente del sistema svolge funzionalità dannose per la sicurezza del sistema stesso

Minacce alle risorse

- Hardware
  - Disponibilità: attrezzatura disabilitata o rubata
  - Confidenzialità: attrezzatura rubata
  - Integrità: Null
- Software
  - Disponibilità: cancellati dei programmi
  - Confidenzialità: copia del programma(non autorizzata)
  - Integrità: programma modificato che svolge operazioni indesiderate
- Dati
  - Disponibilità: cancellati file
  - Confidenzialità: lettura non autorizzata dei dati
  - Integrità: file modificati o creati nuovi
- Network
  - Disponibilità: messaggi distrutti o cancellati
  - Confidenzialità: letti messaggi
  - Integrità: modificati messaggi

6 Security requirements

Distinguiamo in Requisiti tecnici e funzionali.
Requisiti tecnici

- access control: limitare l'accesso ad utenti autorizzati
- Identification and Authenticaion: identificare gli utenti ai sistemi informativi
- Systems and Communication protection: proteggere le comunicazioni dell'organizzazione
- Systems and Information integrity: identificare e correggere subito le falle del sistema

Requisiti funzionali

- Consapevolezza e formazione: assicurarsi che gli utenti di un azienda siano a conoscenza dei rischi
- Responsabilizzazione: assicurarsi che le azioni di un utente possano essere ricondotte in modo univoco a quell'utente
- Valutazione sicurezza: effettuare controlli di sicurezza
- Pianificazione emergenza: stabilire piani d'emergenza per recuper post-catastrofe
- Manutenzione: eseguire manutenzione sul sistema
- Protezione fisica: limitare accesso fisico ai sistemi informativi
- Sicurezza del personale: assicurarsi che le persone che si trovano all'interno di un organizzazione siano fidati
- Valutazione dei rischi: valutare periodicamente il rischio per le operazioni
- Acquisizione dei sistemi e servizi: assegnare risorse sufficienti ai servizi per proteggerli

7 Superfici di attacco

Una superficie d'attacco consiste nell'insieme delle vulnerabilità raggiungibili e sfruttabili in un sistema.
Esempi

- Porte aperte verso l'esterno di un server web
- servizi disponibili tramite firewall
- codice che elabora dati in entrata,email,..
- interfacce,sql,moduli web
- un dipendente con accesso a dati sensibili vulnerabile ad attacchi di ingegneria sociale

Le superfici di attacco possono essere categorizzate nel modo seguente

- rete: vulnerabilità su una rete aziendale,WAN,Internet, protocolli di rete, DoS
- software: vulnerabilità nel codice dell'applicazione
- umano: vulnerabilità generate dall'ingegneria sociale

Un'analisi delle superfici d'attacco è una tecnica utile per valutare la scala e la gravità delle minacce di un sistema.

8 Alberi d'attacco

Struttura dati gerarchica e ramificata che rappresenta un insieme di tecniche per sfruttare le vulnerabilità di sicurezza.
Radice: Obiettivo dell'attacco
Nodi intermedi: sotto-obiettivi
Foglie: diversi modi per iniziare l'attacco

9 Sicurezza delle reti

Si divide in

- Sicurezza delle comunicazioni
  - TLS
  - SSH
  - HTTPS
- Sicurezza dei dispositivi
  - Firewall
  - Intrusion detection: genera allarme
  - Intrusion prevention: rileva e "blocca"

10 Strategia di sicurezza informatica

- Politica di sicurezza: cosa dovrebbe fare lo schema di sicurezza?
- Attuazione: in che modo lo dovrebbe fare?
- Garanzia: funziona davvero?

11 Autenticazione degli utenti

Identificazione: l'utente fornisce un identità presunta.
Autenticazione: modo per stabilire la validità dell'identità presunta.
Distinguiamo tra Autenticazione degli utenti e dei messaggi.
Autenticazione dei messaggi: procedura per stabilire che la fonte sia autentica e che il contenuto ricevuto non sia stato alterato.
Autenticazione degli utenti: stabilire l'identià presunta dell'utente.

Un'identità autenticata ha accesso ha :transazioni/funzionalità/file/processi/dispositivi.

11.1 Requisiti

Distinguiaimo i requisiti in

- Basic
  - Identificazione utenti
  - Autenticare l'identità deglu utenti
- Derivati
  - Usarre autenticazione a più fattori
  - Impiegare meccanismi di autenticazione resistenti ad attacchi
  - Imporre vincoli su password
  - Proibire la stessa password per un determinato numero di volte
  - Permettere l'uso di una password temporanea per il primo accesso
  - Memorizzare le password criptate

11.2 Modello generale di autenticazione utenti

Step1: l'utente deve essere registrato presso il sistema
Per registrarsi : l'utente si rivolge ad un'autorità di registrazione(RA) per diventare un sottoscrittore presso un fornitore del servizio di credenziali(CSP). La RA è fidata e garantisce l'identità del richiedente al CSP.
Il CSP lascia le credenziali al richiedente : una credenziale è una struttura dati che collega un utente ad un token, un token può essere una password,una chiave,...
Può essere generata dal CSP, dal sottoscrittore(utente registrato) o da terze parti.
Token e credenziali si possono usare per eventi di autenticazione successivi.
Una volta che l'utente è registrato il processo di autenticazione si effettua tra l'utente e uno o più sistemi che eseguono l'autenticazione.
La parte che deve essere autenticata si chiama "claimant", mentre chi deve autenticare si chiama "verifier".
Una volta che il claimant dimostra al verifier delle credenziali corrette, il verifier può controllare se effettivamente il claimant è chi dichiara di essere.
Il verificatore trasmette informazioni circa l'identità del claimant, momento della registrazione,e altre informazioni importanti verificate nel processo di registrazione, al relying party(RP) di cui si fida.
L'RP usa queste informazioni per prendere decisioni di controllo accesso/autorizzazione.
Nella realtà il sistema è molto più complesso, però queste sono le funzioni necessarie per un sistema di autenticazione sicuro.

11.3 Mezzi di autenticazione

- Qualcosa che l'individuo conosce: Password,PIN, risposte a domande prestabilite
- Qualcosa che l'individuo possiede: Smartcard,chiave fisica,..
- Qualcosa che l'individuo è(biometria statica): Riconoscimento facciale,retina,impronta digitale

- Qualcosa che l'indivudo fa(biometria dinamica): Voce,Scrittura,...

Ognuno di questi ha i suoi pro e i suoi contro, le password possono essere indovinate o l'utente può dimentcarle, le smart card possono perdersi o rubarle,i sistemi biometrici possono dare falsi positivi e/o falsi negativi.
Altri fattori: il costo.
L'autenticazione multifattore si riferisce all'uso di uno o più mezzi usati in un sistema di autenticazione

11.4 Valutazione dei rischi per l'autenticazione degli utenti

Tre concetti importanti e distinti

- Livello di garanzia: descrive il grado di certezza da parte di un'organizzazione che un utente abbia presentato una credenziale che si riferisce alla propria identità.
  - Livello 1: Scarsa o nessuna fiducia: Utente che si registra presso un form di discussione e fornisce un id e password
  - Livello 2: Discreta fiducia, usato in molte applicazioni(usare un protocollo di autenticazione sicuro)
  - Livello 3: Alta fiducia, usato nelle applicazioni in cui i clienti devono accedere a dati di alto valore ma non massimo(usare almeno due tecniche di autenticazione)
  - Livello 4: Fiducia molto alta, usato quando i clienti devono accedere a dati di un valore molto alto,dove gli accessi improri portano ad un pericolo elevato. Molti fattori di autenticazione e registrazione in presenza
- Impatto potenziale Ci sono 3 livelli di impatto potenziale in caso di violazione della sicurezza.
  - Livello 1: Un errore di autenticazione possa avere un limitato effetto,funzionalità ridotta ma danno molto lieve
  - Livello 2: Grave effetto negativo,perdita finanziaria significativa e funzionalità molto ridotta
  - Livello 3: Effetto catastrofico,gravi perdite finanziare, persone in pericolo di vita,stop delle funzionalità
- Area di rischio Una perdita finanziaria nel caso in cui ci fosse un accesso non autorizzato al sistema cosa comporta?
  Anche qui ci sono 3 livelli.
  - Livello 1: Nel peggiore dei casi, una sola perdita finanziaria non recuperabile
  - Livello 2: Nel peggiore dei casi, una grave perdita finanziaria non recuperabile
  - Livello 3: Nel peggiore dei casi, una catastrofica perdita finanziaria non recuperabile

11.5 Autenticazione basata su Password

Sistema molto usato formato da ID e password.
L'ID serve per vedere se un utente è registrato nel sistema e per controllare i suoi permessi(a cosa può accedere).
VULNERABILIÀ DELLA PASSWORD
Una tecnica usata nei sistemi per memorizzare le password degli utenti è quella di memorizzare password criptate.
Principali tipi di attacchi e contromisure

- Attacco offline basato su dizionario: un hacker può ottenere il file delle password e confrontare le password memorizzate sul file con password di uso comune,problema..le password sono criptate,infatti gli hacker confrontano la stringa risultante dalla funzione che cripta e che prende in input la password di uso comune con le password memorizzate nel file.
  Contromisure: rigidi controlli di accessi a questi file,misure di rilevamento delle intrusioni.

- Attacco ad un account specifico: L'hacker cerca di indovinare la password di un utente specifico,tramite ingegneria sociale.
  Contromisure: bloccare l'account dopo n tentativi
- Attacco con password comuni: Se l'hacker ha una lista di ID, prova su ognuno di loro una lista di password comuni.
  Contromisure: imporre vincoli su scelta delle password
- Indovinare la password di un singolo utente: Meccanismo più avanzato dell'attacco ad un account specifico.
  Contromisure: Imporre password difficili e non banali
- Appropriazione di una work-station: L'attaccante aspetta che un utente lasci incustodita la work-station. Contromisure: Effettuare il logout automatico dopo un peridio di inattività.
- Sfruttare gli errori dell'utente: L'utente scrive password su fogli di carta,... e l'attaccante la scopre.Contromisure: educazione degli utenti
- Sfruttare l'uso di password mupltiple: Un utente potrebbe inserire la stessa password su più sistemi a lui associati
- Monitoraggio elettronico: Se la password viene comunicata attraverso una rete per accedere ad un sistema remoto,diviene vulnerabile ad intercettazioni.

Nonostante le diverse vulnerabilità, la password rimane il meccanismo più usato per diverse ragioni
-Oggetti fisici tipo card sono scomodi da portare in giro e possono essere persi
-Tecniche si sistemi biometrici utilizzano HW e SO particolare che deve essere accettato sia dal lato client che server

11.5.1 Uso di password hashed

Un sistema usato soprattutto in UNIX consiste nel memorizare le password trasformate con una funzione hash più l'aggiunta di un salt.
L'utente sceglie una password, questa password viene unita con un salt(pseudo casuale oggi,legato all'istante di registrazione, nelle vecchie implementazioni),la nuova password viene memorizzata in hash nel file delle password insieme al valore in chiaro del salt.
Quando un utente inserisce la sua password nel sistema, viene unita al salt(preso nella rispettiva riga dell'ID) e viene applicata la funzione di hash che confronta il risultato con la password memorizzata nel file.
A cosa serve il salt?
Impedisce che le password duplicate siano visibili sul file delle password
Aumenta difficoltà della password
Diventa quasi impossibile scoprire se un utente ha inserito la stessa password su due o più sistemi

11.5.2 Password Cracking

L'approccio più usato per scoprire una password (password cracking o guessing) è quello di sviluppare un grande file di possibili password e di provare ognuna di queste rispetto ai file delle passwor, ogni password deve essere aggiunta ad ogni valore di salt disponibile e poi confrontata con gli hash memorizzati,se non trova corrispondenze prova variazioni sul file delle possibili password.
Alternativa: l'attaccante genera una lista di possibili password e, per ogni password, genera l'hash associato ad ogni possibili valore di salt,il risultato è una lista enorme di stringhe che sono le possibili password che prende il nome di rainbow table.Quest'approccio si contrasta usando un valore di salt molto grande insieme ad una funzione di hash a lunghezza elevata.

APPROCCI MODERNI:
Hardware più sofisticato e algoritmi di intelligenza artificiale hanno aumentato la capacità del password guessing, L'AI ha bisogno di password realmente in uso che ad oggi sono disponibili,dopo che, grazie ad alcuni attacchi, sono riusciti ad ottenere 32 milioni di password in chiaro da un servizio di giochi online.

11.6 Controllo degli accessi al file delle password

Spesso gli hash sono memorizzati in un file separato dagli ID utenti, chiamato "shadow".
Molti sistemi UNIX psono suscettibili a intrusioni impreviste.

11.7 Strategia della selezione della password

4 tecniche

- Educazione degli utenti
- Password generate dal computer
- Controllo reattivo delle password: il sistema esegue periodicamente il password craker per trovare password indovinabili e cancella le password che ha indovinato,informando l'utente.Ha degli svantaggi: consuma risorse,le password rimangono vulnerabili finche il sistema che verifica non le trova.
- Politica delle password complesse: Il sistema pone dei vincoli all'utente: 2 meccanismi, il primo usa applicazioni di regole(almeno x caratteri,...) il secondo è password checker,costruisce un grande dizionario di password scadenti e controlla che la password dell'utente non sia li dentro.Inefficiente in termini di spazio e di tempo.
  Ulteriore metodo: Bloom filter: meccanismo efficiente per rifiutare delle parole in una lista,implementato in Linux.

11.8 Autenticazione basata su token

Sono chiamati token gli oggetti che un utente possiede allo scopo di effettuare l'autenticazione.

- Memory Card: immagazzinano dati ma non elaborano,memorizzano un codice di sicurezza che può essere letto da un lettore di carte,usate per l'accesso fisico.
  Alcune memory card hanno una scheda di memoria, quese memory card possono essere usate con un PIN, l'utente quindi deve inserire sia la card che il pin associato.
  Svantaggi: richiedono un lettore speciale,perdita del token, scomodo usarla su computer.
- Smart Token : token più intelligenti: hanno un microprocessore incorporato,interfaccia utente per interazione uomo/token,interfaccia elettronica:contact(inserita in un lettore di smart card), contactless(richiede solo stretta vicinanza con il lettore)
  Protocolli di autenticazione dell'utente con il token
  - Statico: quando l'utente vuole accedere ad un servizio, inserisce il token e la password,se la password è quella memorizzata nel token allora può accedere al servizio
  - Dinamico: quando l'utente inserisce il token,quest'ultimo genera una password temporanea (OTP) che l'utente deve inserire da una qualche parte.
  - sfida-risposta: l protocollo funziona in questo modo:
    -L'utente inserisce il proprio smart token e richiede l'accesso al sistema protetto.
    -Il sistema di autenticazione invia al token una "sfida" casuale, solitamente un numero o una stringa di testo.
    - Lo smart token utilizza la sfida per generare una "risposta" univoca, solitamente una password dinamica o un codice di autenticazione unico.
    - Lo smart token invia la risposta al sistema di autenticazione.
    -Il sistema di autenticazione verifica la validità della risposta confrontandola con quella attesa.
    -Se la risposta è valida, l'utente viene autenticato e può accedere al sistema protetto.

    Questo meccanismo di autenticazione rende più difficile per l'attaccante falsificare la password dinamica e confermare la propria identità perché la sfida è casuale e la risposta generata dallo smart token è univoca per ogni richiesta di autenticazione. Inoltre, il sistema di autenticazione può verificare la validità della risposta utilizzando algoritmi crittografici sicuri e difficili da manipolare.

Per l'autenticazione dell'utente, la categoria più importanti degli smart token è la smart card.
Una smart card ha un intero microprocessore: Processore,Memoria e I/O port
3 tipi di memoria: ROM,EEPROM (può essere programmata e letta elettricamente, ma, a differenza delle ROM tradizionali, può anche essere cancellata e riscritta. Questo processo di cancellazione e riscrittura può essere effettuato singolarmente per ciascuna cella di memoria, il che la rende particolarmente adatta per l'archiviazione di dati che vengono frequentemente aggiornati o modificati.), RAM

Uso di smart card come carte d'identità elettroniche: contiene dati personali,numero di documento,numero di accesso alla carta(CAN),zona leggibile dalla macchina(MRZ).Queste ultime due vengoni usate come password. Tre funzioni elettroniche:
ePass contiene l'immagine del viso e due impronte digitali,si accede con le due password.
eID: cognome,nome,laurea,dataNascita, si accede con le due password e con l'eID PIN.
eSig : creazione della firma elettronica, sempre con passoword(CAN) e eID PIN. I cittadini effettuano la firma elettronica con il PIN e Sign.

11.9 Autenticazione Biometrica

- Riconoscimento volto
- impronta digitale
- geometria della mano
- retina
- iride
- firma
- voce

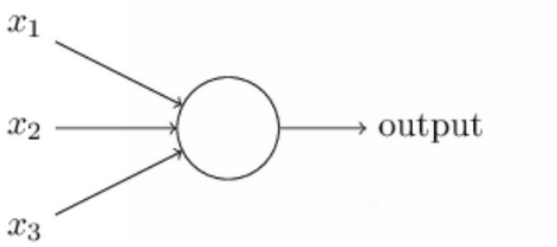
Si differiscono in base al costo e all'accuratezza.
Il più preciso e accurato è l'iride, ma anche il piu cosroso, mentre il meno costoso e il meno accurato è la voce, una via di mezzo è la retina abbasstanza accurata e non troppo costosa.La mano è molto costosa e poco affidabile.
Nel database, oltre a utente e password devono essere memorizzate le caratteristiche biometriche rappresentate come un insieme di numeri(una stringa numerica rappresenta una caratteristica biometrica unica), questo insieme di numeri viene chiamato template dell'utente.
L'utente ora ha :id,pass,valore bioemtrico.
A seconda dell'applicazione per l'autenticazione si usa la verifica o l'identificazione.
La verifica è analoga al sistema della verifica delle password, si utilizza un sensore biometrico che estrae la stringa numerica e la confronta con quella associata all'utente con ID E PASSWORD precedentemente inseriti.
Per l'identificazione,l'utente usa solo sensore biometrico, senza inserire altre informazioni, il sistema confronta solo se quell'informazione biometrica è nel database oppure no.Se è presente accede,altrimenti no.

11.9.1 Accuratezza biometrica

Quando il sistema deve confrontare il modello memorizzato con il modello inserito, viene usato un algoritmo che restiuisice un numero che indica quanto i due modelli sono simili.
C'è il problema dei falsi positivi o dei veri negativi.

11.10 Password Guessing: sistemi moderni con Neural Network

In un sistema UNIX like, le password sono memorizzate nel file /etc/passwd,ogni riga ha questo formato
username:stub:userID:GroupId:InfoUser:PathAssoulutoDellaHome:PathAssolutoDellaShell
Lo stub collega la real password che sta dentro /etc/shadow, ogni riga dentro shadow è cosi
user:HasPas:LastPassChange:...



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Figure 2: perceptron

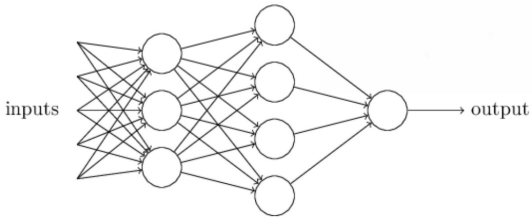


Figure 3: rete neurale

L HashPas ha 3 campi idHash(quale funzione di hash è stata usata),salt,passwordHash
Come vengono indovinate le password?

- brute-force: genero randomicamente segumenze di stringhe,che poi hasho,l'hash risultate lo confronto con la password che voglio indovinare presente sul file delle password.
- attacco a dizionario: in rete sono presenti password gia usate da altri utenti e si provano queste password
- deep-learning: utilizzo di reti neurali

11.10.1 Cosa è una rete neurale?

In primis, un neurone umano, ha n input e un output.
Si calcola la funzione di attivazione in questo modo: ogni input i ha un "peso" w-i e la rete neuale da in output,nel caso più semplice, 0 se la somma di tutti gli input i moltiplicati per il rispettivo w è minore o uguale ad un valore soglia(threshold),1 altrimenti.
I pesi inizialmente sono inseriti casualmente,man mano che la rete viene addestrata, i pesi si aggiornano e diventano sempre più precisi.
Tutto questo è un perceptron, l'equivalente di un neurone umano(prende n input e da un output), tanti neuroni uniti a più livelli formano una RETE NEURALE a x livelli.

11.10.2 Come impara la rete?

L'apprendimento avviene in due fasi

- feed forward propagation: gli do l'input e calcola l'output,calcolo tutte le funzioni di attivazione di ogni neurone,ogni livello ha bisogno dell'output del livello precedente. Alla fine genero l'output

finale che viene confrontato con l'output reale(quello che mi aspetto), aggiorno i pesi tenendo conto della differenza che c'è tra i due output.

- backward propagation: aggiorno i pesi dall'ultimo layer fino all'inizio andando all'indietro.

L'aggiornamento dei pesi viene fatto tramite un algoritmo chiamato: "Calcolo del gradiente"

11.10.3 Password guessing with NN

Questi processi utilizzano milioni di password realmente usate da utenti come dataset di training
Lo scopo è quello di costruire una rete neuale che generi passoword "umane",cioè possibili password generate da persone umane.
Esempio Passgan: ci sono due reti neurali, una rete distigue se un input è generato da una rete o è reale,l'altra rete genera l'input.
Una fa da descriminatore e deve decidere se l'input è una password è reale o generata,l'altra password in base al feedback che gli dice il descriminatore adatta i pesi e si allena.
Quando il descriminatore non ha più possibilità di capire se la password è umana oppure è generata da una rete,allora l'allenamento è finito e la rete è pronta a sapere generare password umane.
Il descriminatore si costuisce partendo dal file delle password reali create da umani.
Si può fare di meglio?
Si...
Si usa l'architettura Normalizing Flows: il training è più stabile.
Data una password generata da un utente, si va a creare un piano di punti in cui, si vanno a creare password simili (che si rappresentano come punti nel piano)a quella password che potrebbero essere state inserite da utenti,più sono vicine (nel piano)alla password dell'utente e più sono reali.
Performance molto più elevate in termini di tempo di allenamento.

12 Controllo degli accessi

Il controllo degli accessi è un elemento centrale della sicurezza informatica.
I principali obiettivi della sicurezza informatica sono impedire ad utenti non autorizzati di accedere alle risorse.

12.1 Requisiti

-Di base

- Limitare l'accesso al sistema informativo ai soli utenti autorizzati
- Limitare agli utenti autorizzati di svolgere le sole funzioni per cui sono autorizzati

-Derivati

- Adottare il principio del minimo privilegio
- Limitare tentativi di accesso non riusciti
- Terminare automaticamente una sessione utente dopo il verificarsi di alcune condizioni
- Altri...

12.2 Contesto del controllo degli accessi

- Autenticazione: Verifica che le credenziali di un utente siano valide(stabilisce se l'utente è autorizzato ad accedere al sistema).
- Controllo: Determina se lo specifico accesso richiesto dall'utente è consentito.
- Autorizzazione: Concessione di un diritto/permesso ad un'entità del sistema.

La funzione di controllo opera mediante un database che specifica, per ogni utente, a quali risorse può accedere e in che modo.
Nella realtà, le funzioni di controllo, può essere ripartita tra più componenti che operano insieme.

12.3 Politiche di controllo degli accessi

Una politica di controllo degli accessi, che può essere integrata in un database di autorizzazioni, definisce quali tipi di accesso sono consentiti e da parte di chi.
Le politiche di controllo sono generalmente suddivise in

- Controllo degli accessi discrezionale(DAC): controlla l'accesso sulla base dell'identità del soggetto richiedente,definita discrezionale perchè un entità potrebbe avere i privilegi di concedere,ad un'altra entità, l'accesso ad una determinata risorsa.
- Controllo degli accessi obbligatorio(MAC): Un' entità non può, solo a suo giudizio, concedere ad un'altra entità l'accesso ad una risorsa,consente l'accesso confrontando l'etichette di sicurezza (che indicano quanto siano sensibili le risorse) con le autorizzazioni di sicurezza(che indicano se le entità sono autorizzate ad accedere a determiante risorse)
- Controllo degli accessi basato su ruoli(RBAC): controlla l'accesso in base ai ruoli che gli utenti ricoprono all'interno del sistema.
- Controllo degli accessi basato sugli attributi(ABAC): controla l'accesso sulla base degli attributi dell'utente,della risorsa a cui può accedere.

12.4 Elementi principali del controllo degli accessi

- Soggetto(o processo): entità capace di accedere agli oggetti,qualsiasi utente che ottiene l'accesso ad un oggetto,l'ottiene attraverso un processo che rappresenta quell'utente,che assume i suoi permessi di accesso.
  Tre classi di soggetto
  - Proprietario: può essere il creatore del file.
  - Gruppo: insieme di utenti con determinati privilegi su una risorsa,i privilegi sono assegnati al gruppo e valgono per tutti gli utenti all'interno del gruppo
  - Altri(globale): utenti che non ricadono negli altri due tipi di utenti,tipicamente si concedono i minimi diritti di accesso.
- Oggetto: entità che contiene informazioni come: record, pagine, file, directory,...
- Permesso di accesso: descrive le modtalità in cui un utente può accedere all'oggetto:Lettura,Scrittura,Esecuzione,Cancellazione,Creazione,Ricerca.

12.5 Controllo degli accessi discrezionale

Approccio usato per gestire questi tipi di access: Matrice degli accessi in cui sono presenti soggetti e oggetti.
Ogni cella indica il diritto di accesso da parte di un soggetto per un determinato oggetto.
La matrice può essere implementata in diversi modi

- Access matrix: generalmente sparsa
- Access contro lists: decomposizione per colonne
- Capability Lists: decomposizione per righe

L'access matrix ha le righe che sono i soggetti(user1,user2,..) e le colonne che sono gli oggetti(file1,file2,...)
I vantaggi dell'Access Contro List:si possono trovare facilmente tutti i permessi di un oggetto(ci sono n blocchi in cui l'i-esimo blocco punta all'i-esimo +1 blocco,ogni blocco rappresenta i permessi del oggetto i),risulta svantaggioso trovare tutti i permessi di un soggetto,risulta vantaggiosa perchè può

contenere una voce di default per garantire permessi di accesso ad utenti non elencati.
Capability,l'opposto.
Quest'ultime producono capability ticket,ogni ticket specifica per un determinato utente quali operazioni è autorizzato ad eseguire sugli oggetti.
Ogni soggetto può avere diversi ticket e può essere autorizzato a prestarli o cederli,questi ticket pongono un problema di sicurezza rispetto alle contro lists,l'integrità del ticket deve essere protetta e non falsificabile(il SO dovrebbe mantenere tutti i ticket in una zona di memoria inaccessibile agli utenti).

12.6 Modello per il controllo degli accessi

Lo stato di protezione di un sistema è l'insieme delle informazioni, in un determinato istante, che specificano i permessi di accesso di ogni soggetto relativi ad ogni oggetto.
Introduciamo un modello DAC(discrezionale)
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.
Il modello affronta tutti e 3 i requisiti.
Per rappresentare lo stato di protezione includiamo nella matrice di controllo degli accessi:
Processi: i permessi di accesso consistono nella possibilità di cancellare un processo, fermarlo(bloccarlo) e riattivarlo.
Dispositivi: i permessi di accesso consistono nella possibilità di leggere/scrivere sul dispositivo, bloccare l'utilizzo del dispositivo.
Locazioni o regioni di Memoria: i permessi consistono nella possibilità di scrivere in alcune regioni della memoria dove di norma questo non è possibile
Soggetti: i permessi di accesso rispetto ad un soggetto consistono nel concedere o cancellare i suoi permessi relativi ad altri oggetti
Un utente A effettua una richiesta per l'oggetto X,si interroga la matrice di accesso per determinare se la richiesta è dentro la cella M[A,X].
In caso affermativo l'accesso è permesso,in caso contrario l'accesso è negato, colui che interroga la matrice è il "controllore" che si basa sul contenuto attuale della matrice.
Il modello include un insieme di regole come "proprietario" e "controllo" e concetto di flag di copia, che controllano le modifiche alla matrice d'accesso.
Se in una determinata cella è presente il flag di copia, l'utente può concedere quel permesso , con o senza flag di copia, ad un altro utente.
Altre regole

- Se X è proprietario di un oggetto, allora può concedere permessi di accesso su quell'oggetto con o senza flag di copia ad altri utenti.
- Un soggetto X può rimuovere qualsiasi permesso di accesso di oggetti, su un oggetto dove X ne è proprietario o "controllo"
- Un soggetto X può leggere la porzione di matrice di un oggetto che controlla o ne è proprietario
- Un soggetto X può creare un oggetto di cui diventa proprietario
- Il proprietario di un oggetto può cancellare l'oggetto e di conseguenza eliminare tutte le voci di quell'oggetto nella matrice.
- Un soggetto X puo creare un sogetto Y che possiede,Y ha controllo su se stesso
- Se X è proprietario di Y(soggetto) allora X può cancellare i permessi di Y

12.7 Domini di protezione

Un dominio di protezione è un insieme di oggetti combinati a dei permessi di accesso relativi a questi oggetti. In termini di matrici di accesso,un dominio di protezione è una riga di quella matrice, ogni utente ha un dominio di protezione, ogni utente può generare nuovi utenti che hanno come domini di protezione un sottinsieme del suo dominio di protezione.
L'associazione tra un processo e un dominio può essere statica o dinamica.

Esempi di domini di protezione sono (in Unix) distinzione tra utente e kernel, un programma utente viene eseguito in modalità utente che non può accedere in alcune parti della memoria,le routine di sistema sono fatte in modalità kernel che esegue istruzioni privilegiate.

12.8 Controllo degli accessi ai file in UNIX

Ad ogni utente viene assegnato un ID, un utente è membro di uno(il suo) o più gruppi, ogni gruppo ha un ID. Ogni file in UNIX ha 9 bit di permessi: 3 user,3 group, 3 other.Ogni terna è rwx(read,write,execution).
Un particolare ID utente è indivuduato come superutente, ha accesso a tutto il sistema.
Stickybit: se applicato a una directory, specifica che solo il proprietario di qualsiasi file nella directory può rinominare, spostare o modificare un file.
Il bit "setUID" (Set User ID) viene utilizzato per indicare che l'esecuzione del file deve avvenire con i privilegi dell'utente proprietario del file, invece che con i privilegi dell'utente che lo esegue. In altre parole, quando un file ha il bit "setUID" impostato, chiunque esegue il file acquisisce temporaneamente i privilegi dell'utente proprietario del file. Questo è spesso utilizzato per eseguire programmi con privilegi di amministratore, come il comando "sudo".
Il bit "setGID" (Set Group ID) invece viene utilizzato per indicare che l'esecuzione del file deve avvenire con i privilegi del gruppo proprietario del file, invece che con i privilegi del gruppo dell'utente che lo esegue. In altre parole, quando un file ha il bit "setGID" impostato, chiunque esegue il file acquisisce temporaneamente i privilegi del gruppo proprietario del file. Questo può essere utile in situazioni in cui un gruppo di utenti deve accedere ad una risorsa

12.9 Controllo degli accessi basato su ruoli

Sistemi d'accesso RBAC si basano su ruoliche gli utenti assumono in un sistema piuttosto che sull'identità dell'utente.
Un ruolo diventa una funzione amministrativa all'interno dell'organizzazione,gli utenti vengono assegnati a diversi ruoli sia staticamente che dinamicamente.
A ogni ruolo sono associati permessi d'accesso che cambiano abbastanza raramente.
Viene utilizzata una matrice: RuolixOggetti e una matrice: UtentixRuoli.
Tipicamente vi sono molti più utenti che ruoli,la matrice UtentixRuoli può avere celle contrassegnate(indica che all'utente i è stato assegnato quel ruolo) oppure in bianco(indica che l'utente non ha quel ruolo),un utente può avere più ruoli,più utenti possono avere lo stesso ruolo.
Questo modello si adotta molto bene alla teoria del minimo privilegio,ogni ruolo ha il meno possibile di privilegi(quelli che effettivamente gli servono).

12.9.1 Estensioni RBAC

RBAC-0: è il modello base.
RBAC-1: permette di aggiungere le gerarchie di ruoli(ogni ruolo può ereditare permessi da un altro ruolo.
RBAC-2: estende RBAC-0 e permette di aggiungere vincoli.
RBAC-3: include RBAC-0,1,2.

- RBAC-0: contiene 4 tipi di entità nel sistema.
  Utente: individuo che ha accesso al sistema informatico, ogni utente ha un ID.
  Ruolo: funzione lavorativa all'interno dell'organizzazione che gestisce il sistema informatico,ogni ruolo ha una descrizione dell'autorità.
  Permesso: approvazione di una determinata modalità d'accesso a uno o più oggetti.
  Sessione: mappatura fra utente e sottoinsieme precisato dell'insieme dei ruoli assegnati all'utente. Una sessione è utilizzata per definire una relazione temporanea tra utente e uno o più ruoli a cui l'utente è stato assegnato,l'utente stabilisce una sessione con i suoli ruoli strettamente necessari per eseguire un determinato compito(esempio del minimo privilegio).

- RBAC-1:permettono di avere una struttura gerarchica.Sfruttano il concetto di ereditarietà,permettendo ad un ruolo di includere implicitamente i permessi di accesso ad un ruolo subordinato(tipicamente

in una struttura ad albero si trovano nella parte bassa).Un ruolo può ereditare più ruoli,più ruoli possono ereditare dallo stesso ruolo.

- RBAC-2: i vincoli permettono di adattare RBAC alle specifiche politiche amministrative e di sicurezza di un organizzazione. Un vincolo è una condizione relativa ai ruoli o relazione definita tra i ruoli.
  Tipi di vincoli
  - ruoli mutuamente esclusivi: un utente può essere assegnato soltanto ad un ruolo nell'insieme.Può essere statico o dinamico(cioè relativo ad una sola sessione).Proprietà: utente può avere solo un ruolo (sia durante la sessione che staticamente) , qualsiasi permesso può essere assegnato soltanto ad un ruolo dell'insieme.In questo modo non ci sono permessi sovrapposti.Due utente appartenenti a ruoli diversi,avranno permessi diversi. cardinalità: consiste nell'impostare un numero massimo rispetto ai ruoli,esempio: numero massimo di utenti che possono essere assegnati ad un ruolo,oppure vincolo sul numero massimo di ruoli che può avere un utente,o il numero massimo di ruoli che un utente può attivare durante una sessione,numero massimo di ruoli a cui può essere concesso un determinato permesso.
  - ruolo prerequisito: vincoli di condizione,un utente potrebbe avere un ruolo solo se è stato assegnato già ad un altro ruolo.
- RBAC-3: unione di RBAC-1 e RBAC-2

12.10 Controllo degli accessi basato su attributi

Gli attributi sono caratteristiche che definiscono aspetti del soggetto,oggetto, delle condizioni d'ambiente e/0 delle operazioni richieste che sono predefinite e preassegnate da un'autorità.
Gli attributi contengono informazioni che indicano: la classe di informazione fornita dall'attributo,nome, valore.

- Attributi del soggetto: Ogni soggetto ha degli attributi associati che definiscono l'identità e le caratteristiche del soggetto,possono includere: nome,ruolo,organizzazione,etc..
- Attrbuti dell'oggetto: esempio: titolo, data,autore,...
- Attributi dell'ambiente: attributi legati al contesto operativo in cui avviene l'accesso alle informazioni,esempio :la data,l'ora attuale,attività di virus non sono legati ad un particolare oggetto o risorsa ma possono essere rilevanti per il controllo degli accessi.

ABAC: si distingue perchè controlla l'accesso valutando, per ogni richiesta, le regole in base agli attributi(del soggetto e oggetto), le operazioni, e il relativo ambiente. Si basa sulla valutazione degli attributi del soggetto, dell'oggetto, e di una relazione formale.
I sistemi ABAC possono applicare concetti: DAC,RBAC,MAC. Permettono un numero maggiore di input disceti nel processo di controllo degli accessi.
Permettono di combinare un numero illimitato di attributi per soddisfare qualsiasi regola di controllo dell'accesso.Soddisfano una vasta gamma di requisiti.

12.10.1 Architettura ABAC

L'accesso di un soggetto ad un oggetto avviene cosi: -soggetto richiede l'accesso all'oggetto, la richiesta è indirizzata ad un meccanismo di controllo accessi
-Il meccanismo di controllo accessi ha un insieme di regole basata su una politica di controllo accessi preconfigurata.Sulla base di queste regole il meccanismo valuta gli attributi del soggetto,dell'oggetto, attuale condizioni d'ambiente per determinare se concedere l'accesso o no.
-Permette di accedere se l'utente è autorizzato, nega altrimenti.
Si possono stablire regole per la politica andando a formare qualsiasi regola combinando attributi di soggetto,oggetto,condizioni d'ambiente.Molto efficace e flessibile.
Tuttavia il costo in termini di progettazione e prestazione è molto elevato.
E' migliore rispetto a DAC che usa liste di controllo accessi.
Requisiti di fiducia: ABAC ha bisogno di molte relazioni complesse di fiducia,(in DAC:la fiducia spetta

al proprietario dell'oggetto che è lui a decidere per il suo oggetto gli accessi)in ABAC il proprietario dell'oggetto non ha controllo e la fiducia si basa su :Autorità degli attributi del soggetto,e sviluppatori di politiche.

12.10.2 Politiche ABAC

Una politica è un insieme di regole e relazioni che definiscono, sulla base dei privilegi(o permessi,autorizzazioni,diritti) dei soggetti, il comportamento legittimo all'interno dell'organizzazione e come le risorse debbano essere protetti nelle varie condizioni d'ambiente.
I privilegi rappresentano il comportamento legittimo di un soggetto(definiti da un autorità) e inclusi in una politica.
L'accesso è gestito come una funzione booleana che prende in input gli attributi di un soggetto,oggetto,ambiente) e restituisce true se il soggetto ha accesso all'oggetto,false altriementi.
La funzione si basa delle policy formate da un insieme di regole che si riferiscono a vari soggetti/oggetti.
Esempio: se dobbiamo gestire l'accesso al noleggio di un film in cui i film possono essere:visibili a tutti,vietati ai minorenni,vietati ai minori di 13, con RBAC dovremmo inserire i 3 ruoli,con ABAC basta imporre una funzione "can-access" che prende in input l'utente,il film e l'ambiente,e si valuta l'attributo dell'utente "eta" e l'attributo del film "genere" e si fa un semplice if, se può vederlo gli do l'accesso,altrimenti no.
Vantaggio: evito di inserire i ruoli

12.11 Gestione dell'identità,delle credenziali e degli accessi

Chiamato anche ICAM
E' un approccio completo per gestire e implementare identità digitali,credenziali e controllo degli accessi.
Progettato per:
-creare identità digitali di persone e di ciò che definiscono entità non-persona.
-associare queste identità a credenziali (una struttura dati che lega un identità)
-utilizzare le credenziali per accedere alle risorse in modo autorizzato.

12.11.1 Gestione dell'identità

La gestione dell'identità riguarda l'assegnazione degli attributi a un'identità digitale e l'associazione di quesa identità ad un individuo.Obiettivo: definire un'identità digitale affidabile.
L'approccio più diffuso, ancora ad oggi, per il controllo degli accessi consiste nel creare una rappresentazione digitale di un'identità specifica per l'applicazione.
La decisione sul controllo degli accessi saranno basate sia dall'identità di un utente ma anche sui suoi attributi.
La creazione di un'identità digitale inizia con la raccolta dei dati dell'identità attraverso un processo di iscrizione, un identità digitale è un insieme di attributi che ,se aggregati, rappresentano univocamente un utente all'interno del sistema.
Per stabilire il livello di fiducia nei confronti dell'individuo rappresentato,l'agenzia può effettuare controlli sul suo passato.
Ciclo di vita della gestione delle identità:
-Meccanismi,politiche per proteggere le informazioni relative all'identità personale.
-Controllo dell'accesso sui dati dell'identità
-Tecniche per condividere dati di identità
-Revoca di un identità.

12.11.2 Gestione delle credenziali

Una credenziale è una struttura dati che lega un identità ad un token posseduto e controllato da qualcuno.
Esempio di credenziali: smart card,chiavi crittografiche private, certificati digitali. Gestione delle credenziali cosa comprende:

- Un individuo autorizzato certifica la necessità di un altro individuo di ottenere una credenziale
- L'indivudo si iscrive per ottenere le credenziali dimostrando la propria identità
- Viene generata una credenziale
- Credenziale rilasciata all'indivudo
- Credenziale mantenuta per tutto il suo ciclo di vita

12.11.3 Gestione degli accessi

Ha l'obiettivo di garantire un'adeguata verifica dell'identità nel momento in cui un individuo tenta di accedere,utilizza le credenziali fornite.
Necessari tre elementi di supporto:
Gestione delle risorse: regole su una risorsa che necessita di controllo dell'accesso(esempi di regole:requisiti di credenziali,attributi degli utenti,condizioni ambientali sono richiesti per accedere ad una determinata risorsa)
Gestione dei privilegi: Questi attributi rappresentano le caratteristiche di un individuo utilizzate per accedere alle risorse.I privilegi sono attributi associati ad identità digitali.
Gestione delle politiche: determina ciò che è consentito e ciò che non è consentito durante una transazione di accesso, una politica specifica quali azioni,questo utente può eseguire su quell'oggetto.

12.11.4 Federazione delle identità

Tratta due aspetti:
Come fidarsi delle identità di individui esterni che necessitano di accedere ai nostri sistemi?
Come garantire l'identità dei nostri individui che necessitano di collaborare con organizzazione esterne?
Federazione delle identità è un termine usato per descrivere la tecnologia,le politiche,gli standard che permettono ad una organizzazione di fidarsi di identità digitali,attributi,etc.. create da un altra organizzazione.

12.11.5 Trust framework

Le applicazioni hanno bisogno di potersi fidare delle identità digitali, gli attributi di ogni individuo che è necessario conoscere e verificare per poter effettuare una transazione dipendono dal contesto.
Gli utenti definiscono accordi con un fornitore di servizi d'identità per ottenere identità digitali e credenziali.Ci deve essere fiducia da entrambi i lati quando si effettuano transazioni(utente e parte che si fida dell'utente).
La parte che si fida richiede che l'utente sia autenticato con un certo livello di sucrezza e che gli attributi assegnati all'utente dal fornitore d'identità siano accurati e che il fornitore sia autoritativo.Il fornitore deve essere certo di avere informazioni accurate sull'utente, la parte che si fida può usare le informazioni dell'utente secondo la legge.
L'utente deve avere la certezza di poter affidare le sue informazioni al fornitore e alla parte che si fida,ogni parte vuole sapere quanto sono affidabili le informazioni sull'altra parte e se tutte queste procedure siano davvero attuate.

12.11.6 Open identity framework

Metodo aperto e standardizzato per lo scambio affidabile di identità e attributi.

- OpenID: Uno standard aperto che consente agli utenti di essere autenticato da determinati siti cooperanti utilizzando un servizio di terze parti
- OIDF(OPEN ID FOUNDATION): OpenID Foundation è internazionale organizzazione senza scopo di lucro di individui e aziende impegnate a consentire, promuovere e proteggere OpenID tecnologie
- Information Card Foundation è a comunità senza scopo di lucro di aziende e individui che lavorano insieme per evolversi l'ecosistema delle schede informative

- Open Identity Trust Framework è un specifica standardizzata e aperta di a quadro di fiducia per l'identità e scambio di attributi, sviluppato congiuntamente da OIDF e ICF
- OIX Lo è Open Identity Exchange Corporation indipendente, neutrale, internazionale fornitore di quadri di fiducia di certificazione conforme al modello OITF
- ASN Attribute Exchange Network è online Gateway su scala Internet per l'identità fornitori di servizi e parti facenti affidamento a accedere in modo efficiente all'utente asserito, autorizzato e verificato online attributi di identità in volumi elevati a costi accessibili

13 Sicurezza dei database

In questo capitolo vengono esaminate le tipiche problematiche relative alla gestione delle basi di dati.
Perche dobbiamo gestire la sicurezza?
-La crescente complessita dei DBMS ha introdotto nuove vulnerabilità
-SQL ha delle vulnerabilità
-I gestori del database di un'azienda hanno il ruolo di garantire la disponibilià,potrebbero non avere le competenze per dedicarsi alla sicurezza del database
-La maggior parte delle aziende utilizza piattaforme eterogenee di basi di dati, che crea ulteriore complessità per i gestori delle basi di dati.

13.1 SQLi

Attacchi SQL injection.
Una delle minacce più diffuse e pericolose per la sicurezza della rete.
Utilizzato per sfruttare la debolezza delle pagine web attuali(dinamiche).
Un attacco SQLi viene utilizzato per inviare al database istruzioni malevoli, per estrarre grandi quantità di dati.
SQLi può essere anche sfruttato per modificare e/o cancellare informazioni sul database.

13.1.1 Tipico attacco SQLi

L'attacco è praticabile se l'input dell'utente non viene correttamente filtrato dai caratteri di escape contenuti nelle istruzioni SQL. Tecniche di iniezione:
Un attacco consiste nel troncare volutamente una stringa di testo concatenandole un nuovo comando,l'attaccante terminerà la stringa che inserisce con un commento in modo da ignorare quello che viene dopo(che è scritto sul lato server dell'applicazione).
Commento "–".
";" separa un istruzione dalla successiva.

13.2 Tipi di attacco SQLi

- Input utente: Se l'applicazione ha dei form/submit con cui riceve input dall'esterno, che poi vengono inviati all'applicazione dall'esterno con GET/POST, l'hacker può iniettare comandi malevoli per ottenere ciò che vuole.
- Variabili del server: Le applicazioni web utilizzano le variabili del server per uso personale, se queste variabili vengono salvate in un database senza alcun tipo di controllo si possono creare vulnerabilità SQLi,in quanto si possono modificare le variabili e quando verrano eseguite le query per inserire nel database queste variabili, verrà eseguita anche l'istruzione contenuta all'interno delle variabili.
- Iniezioni di secondo ordine: Si verifica quando si utilizzano meccanismi di sicurezza poco forti,l'attaccante può sfruttare dei dati gia presenti nel database per attacco SQLi, in questo caso l'input che modifica la query rendendola pericola proviene anche dall'interno.
- Cookie: Un hacker può fare in modo che quando viene eseguita una query ,per il contenuto del cookie per l'utente che sta entrando nell'applicazione,la struttura della query potrebbe essere modificata.

- Input fisico dell'utente: Input può essere fisico come codici a barre,in opportuni contesti.

I tipi di attacco possono essere classificati in tre categorie principali: in banda, inferenziale, e fuori banda.
Un attacco "in banda" sfrutta lo stesso canale di comunicazione per iniettare codice ed ottenere risultati tramite pagine web.
Includono i seguenti tipi

- Tautologia: Si iniettano istruzioni sempre vere per ottenere informazioni su intere tabelle.
- Commento di fine riga: annullo il codice legittimo dell'applicazione
- Query piggybacked: L'hacker aggiunge ulteriori query oltre quella prevista.

Attacco inferenziale:
Non c'è un trasferimento di dati,ma l'attaccante ottiene informazioni analizzando le risposte del server

- Query illecite o logicamente scorrette: Si inviano istruzioni errate per analizzare l'errore che invia il database in modo da scoprire le vulenrabilità.
- Blind SQL injection: L'attaccante pone istruzioni di tipo vero/falso, se la query è vera, la pagina continuerà a funzionare correttamente,se risulta essere falsa,la pagina,anche se non mostra un chiaro messaggio d'errore, difficilmente continuerà a funzionare come prima.Con una serie di istruzioni l'attaccante è in grado di ottenere informazioni sulla struttura e le vulernabilità.

In un attacco "fuori di banda" viene sfruttuato un altro canale di comunicazione,esempio le email,difficlmente i server del database effettuano rigidi controlli sulle comunicazioni in uscita attraverso altri canali,esempio: risultati della query inviati al richiedente tramite email.

13.3 Contromisure

Bisogna unire più tecniche per difendersi.
Raggruppiamo le tecniche in 3 categorie

- Programmazione difensiva.
  - Pratiche manuali di Programmazione difensiva: Controllare/pulire l'input che si riceve dall'esterno.
  - Inserimento parametrizzato della query: Questo approccio cerca di bloccare attacchi SQLi in quanto, qualsiasi input dall'esterno venga inserito, non andrà a modificare l'istruzione SQL.
  - SQL DOM: sono delle API che abilitano il controllo automatico dei caratteri di escape,lo sviluppatore può usarle per effettuare dei rigidi controlli sulle iniezioni esterne degli utenti.
- Detection:
  3 metodi
  - Signature based(basate su firma): Individuano specifiche sequenze di attacco,potrebbero non andare bene per codice automodificante
  - Anomaly based: Viene addestrato distinguendo un modello normale del funzionamento del sistema da uno anomalo
  - Analisi del codice: Uso di suite che generano una serie di attacchi SQLi per rilevare vulnerabilità
- Prevenzione a tempo di esecuzione: Controllano la query per vedere se è coerente con il modello previsto. Sono presenti tool automatici

13.4 Controllo degli accessi al database

I DBMS utilizzano 3 politiche diverse per gestire l'accesso

- Amministrazione Centralizzata: Solo un numero limitato di utenti può dare/revocare i permessi per l'accesso al db
- Amministrazione basata su proprietà: Solo il proprietario della tabella può concedere/revocare permessi per quella tabella
- Amministrazione decentralizzata: Il proprietario può autorizzare utenti a concedere/revocare permessi ad altri utenti su quella tabella.

I permessi possono essere anche in modo che in una stessa tabella non tutti gli utenti possano vedere tutte le colonne ma solo un sottinsieme.
GRANT OPTION e REVOKE OPTION: Autorizzazioni a cascata, se levo un permesso ad un utente con l'opzione "option", di conseguenza levo i permessi a tutti gli utenti a cui l'utente a cui l'ho revocato l'ha concessi.
Mentre GRANT OPTION: quelli che ottengono il permesso,possono concederli anche loro.

13.4.1 RBAC

RBAC è una buona soluzione per il controllo degli accessi all'interno di un database.
Classifichiamo gli utenti che vogliono accedere al database in 3 diverse categorie.

- Proprietario dell'applicazione:Proprieario di tutti gli oggetti del database(tabelle,colonne,etc..)
- Utente finale diverso dal proprietario: Utente che utilizza gli oggetti del database
- Amministratore: ha la responsabilità di amministrare parzialmente o totalmente il db

Per ogni attività che un utente può svolgere all'interno dell'applicazione si definiscono uno o più ruoli che specificano quali sono i permessi necessari.Il proprietario assegna i ruoli agli utenti.
Gli amministratori si occupano dei ruoli più delicati.
Un sistema per il controllo degli accessi basato su ruoli deve avere le seguenti funzionalità:
-Creazione ed eliminazione dei ruoli
-Definizione dei permessi all'interno dei ruoli
Assegnare/rimuovere ruoli agli utenti.
SQL Server che usa RBAC ha 3 tipi diversi di ruoli. Ruoli a livello di server,Ruoli a livello di database, ruoli definiti dall'utente.
I primi due sono ruoli fissi,possono soltanto essere assegnati ad utenti ma non modificati.

- Ruoli fissi del server: indipendenti da qualsiasi database,usati per semplificare l'amministrazione del server,evitano di concedere il controllo totale del server ad un singolo utente
- Ruoli fissi del database: valgono all'interno del singolo database
- Ruoli definiti dagli utenti: è possibili assegnare permessi a porzioni del database,con opportuni permessi un utente può definire ruoli.
  - Ruolo standard: un utente autorizzato può assegnare quel ruolo ad altri utenti
  - Ruolo applicazione: Il ruolo viene abilitato quando l'applicazione esegue una porzione specifica di codice

13.5 Inferenza

L'inferenza è il processo che consiste nell'effettuare interrogazioni legittime al database,e dedurre informazioni attraverso le risposte ricevute.
Il percorso di trasferimento dei dati attraverso il quale si ottengono le informazioni riservate è definito CANALE DI INFERENZA.
Il meccanismo è il seguente: si utlizzano le dipendenze funzionali che legano gli attributi all'interno di una tabella e si "ricostruisce" la tabella creando 2 o più viste.

13.5.1 Contromisure

- Rilevamento in fase di progettazione: Dividere la tabella in più tabelle.
- Rilevamento in fase di esecuzione query: Si analizza a run time l'esecuzione della query se viene rilevata un'inferenza,la query viene bloccata

13.6 Cifratura nei database

La cifratura rappresenta l'ultimo meccanismo di difesa all'interno di un database.
2 svantaggi

- Gestione delle chiavi: Bisognerebbe inviare a tutti gli utenti autorizzati la chiave per decifrare i dati
- Rigidità: Effettuare la ricerca in tempo "breve" diventa complesso quando i dati sono cifrati

La cifratura può essere applicata a livello di tabelle,di attributi,di record, etc..
Se un azienda si affida ad un gestore di DBMS e cifra tutte le tabelle,ma decide di non fornire la chiave di decifratura al server,quando un utente deve accedere ad un dato, dovrà scaricare l'intera tabella in locale,decifrare e poi ottenere il dato decifrato.
Se il database è cifrato in modo che ogni singolo elemento sia cifrato in modo individuale,quando l'utente fa una query, sistema cifra la sua istruzione e va a prendere dal database,ma se l'utentr vuole più di un dato che rispetta un vincolo tipi "tutti gli utenti con eta minore di 20", diventa complicato in quanto i dati cifrati non rispettano l'ordine nello stesso modo dei dati non cifrati.
C'è un secondo approccio: cifrare ogni riga come un unico blocco,ogni blocco cifrato corrisponde ad una riga nel database ed ha una sola decifratura che è la riga nel database decifrata.Ogni riga nella tabella cifrata ha: campo per l'intera riga cifrata, e n campi (con n = numero di colonne della tabella non cifrata) dove ci sono degli indici. Se un attributo nella tabella non cifrata può avere un valore da 1 a 100,nella tabella cifrata,la rispettiva colonna avrà 1,se il valore può essere tra 101 e 200,la tabella avrà 2 e cosi via...Se il campo è testual e non numerico si definiscono indici sulla base della prima lettera(1 se inizia con A o B,2 se inizia con C e D,etc..).Questi parametri per mappare i dati con gli indici sono memorizzati nel client e non nel server.
Questa procedura migliora l'estrazione dei dati perchè, ora, se vogliamo prendere tutti gli utenti con eta minore di 20,andiamo ad analizzare gli indici!! che non sono cifrati e mantengono l'ordinamento che ci aspettiamo noi.
Questo meccanismo migliora il problema che si verificava nel primo approccio in cui ogni oggetto è criptato individualmente e non possiamo applicare operatori di confronto in quanto i dati criptati non hanno lo stesso ordinamento dei dati non criptati.
Da ricordare che la decriptazione avviene grazie al client che ha tutte le chiavi e i metadati per mappare gli indici con i dati.Il server non memorizza niente.

14 Sicurezza nei datacenter

Un datacenter è una struttura aziendale che ospita un numero elevato di server.
Anche i datacenter necessitano di sicurezza,ospitano enormi quantità di dati.
Sono contenuti in un'area fisica ben definita.
Pericoli: DoS,SQLi,Malware,pericoli relativi all'infrastruttura fisica.
Modello di sicurezza nei datacenter a più livelli

- Sicurezza del sito: Protezione della struttura con sistemi di controllo,barriere di sicurezza,hardware ridondante,etc..
- Sicurezza fisica: autenticazione a più fattori,uso di specifiche procedure di controllo
- Sicurezza della rete: Un datacenter adotta tutte le possibili tecniche di sicurezza che possono essere applicate,in quanto ospita tanti dati accedibili da reti esterne.
- Sicurezza dei dati: Specifiche tecniche che servono a proteggere i dati memorizzati(crittografia)

14.1 TIA-942

Requisiti minimi che devono rispettare tutte le infrastrutture dei datacenter

- Ridondanza nel sistema
- Controllo ambientale
- Architettura di rete

Da vedere bene

15 Malware

Il malware rappresenta una delle minacce più importanti nei sistemi informatici.
Il malware è un programma che viene inserito in un sistema(di nascosto) con l'intento di compromettere la confidenzialità,l'integrità e la disponibilità dei dati.

15.1 Classificazione

Un malware si può dividere in due categorie generali,in primo luogo su come si diffonde o si propaga per raggiungere i target destinati, in secondo luogo, su come si comporta una volta raggiunto l'obiettivo.
Una seconda classificazione: esistono malware che necessitano di un programma host(come virus),altri sono indipendenti.
ALtra classificazione: alcuni malware si replicano,mentre altri non si replicano.

15.2 Mezzi di propagazione

Infezione di eseguibili da parte di un virus che successivmente vengono diffusi ad altri sistemi
Sfruttamento delle vulnerabilità di un software sia localmente che in rete per consentire ad un malware di replicarsi
Utilizzo di ingegneria sociale che convicono gli utenti a bypassare i meccanismi di sicurezza per installare trojan o rispondere al phishing.

15.3 Azioni

Le azioni che compie un malware appena raggiunto l'obiettivo possono comprendere: corruzione di un file,compromissione di un servizio,furto di informazioni,comportamento furtivo con cui il malware nasconde la propria presenza quando ci sono tentativi di rilevazione antivirus.
Un attacco ibrido utilizza più metodi di infezione per massimizzare la velocità di contagio,alcuni si aggiornano in modo da modificare il meccanismo di propagazione una volta distribuiti.
Ad oggi infatti i malware utilizzano tecniche combinate,in passato no.

15.4 Kit di Attacco

All'inizio, lo sviluppo di malware richiedeva una competenza notevole,ad oggi non è cosi grazie alla presenza di tool-kit che aiutano la creazione di malware,questi tool-kit prendono il nome di crimeware.
E' possibile selezionare il meccanismo di propagazione,le vulnerabilità che si vogliono sfruttare e tanto ancora,non servono competenze per combinare queste opzioni e creare un malware.
Un malware progettato con questi tool-kit tende a essere meno sofisticato di uno progettato da zero.
Esempi di tool-kit:Zeus e Angler

15.5 Sorgenti d'attacco

Alcuni attaccano solo per dimostrare abilità tecnica, altri per ricattare le vittime in cambio di soldi,altri ancora provengono da gruppi organizzati molto pericolosi.
Ciò ha creato una vendita di kit d'attacco enorme.

15.6 Advanced Persistent Threat

Gli APT sono degli attacchi informatici che utilizzano un'ampia varietà di tecnologie di intrusione applicate in modo persistente a target specifici per un periodo prolungato(anche anni).
Un tipo di attacco informatico altamente sofisticato e mirato, che può essere condotto da un gruppo di hacker, un'organizzazione o persino da un governo.
Gli APT sono attacchi che si concentrano sul lungo termine e utilizzano una vasta gamma di tecniche, come la spear-phishing, l'infiltrazione di malware, l'utilizzo di exploit di zero-day e l'ingegneria sociale per ottenere l'accesso ai sistemi target e rubare informazioni sensibili. Gli APT possono rimanere attivi per mesi o addirittura anni senza essere rilevati, utilizzando tecniche di evasione per evitare la rilevazione da parte degli strumenti di sicurezza informatica.
ADVANCED: utilizzo di un'ampia varietà di tecnologie che, combinate e selezionate per il target specifico, insieme sono molto sofisticati,mentre nel singolo, le componeneti non sono molto sofisticati.
PERSISTENT: l'attacco verso il target selezionato è prolungato.
THREAT: minacce ai target selezionati come conseguenza della volontà di attaccanti organizzati unito al coinvolgimento attivo delle persone aumenta il livello di minaccia rispetto ad attacchi automatizzati.

15.6.1 Attacchi APT

Lo scopo: varia dal furto di proprietà intellettuale, ai dati relativi alla sicurezza e all'infrastruttura fisica.
Tecniche: ingegneria sociale,e-mail di phishing, download da siti web compromessi
Intento: infettare il bersaglio con malware con molteplici meccanismi di propagazione,una volta preso il bersaglio,si utlizzano tecniche sofisticate e combinate per rimanere nascosto nel target per molto tempo.

15.7 Virus

Un virus è un tipo di malware,nello specifico con il termine virus si intende un frammento di codice(tipicamente codice macchina) che si lega ad un file eseguibile e lo infetta,in realtà il termine virus viene ad oggi ancora usato per intendere un malware in generale,ma non è cosi.
Il virus quando infetta un file, include una copia di se stesso in quel file,in questo modo il virus si propoga più velocemente visto che si creano n copie di se stesso,è specifico per il sistema operativo o l'hardware scelto e sfrutta i loro punti deboli.
Un virus quindi ha una sequenza di istruzioni che gli permettono di creare copie perfette di se stesso.
Quando un computer infetto entra in contatto con un file non infetto, una nuova copia del virus viene trasferita subito su quel file,in questo modo il virus si può trasferire da computer a computer qualora il file sia inviato per email o usb.
Un virus che si inserisce in un file è autorizzato a fare tutto ciò che il file è autorizzato a fare e viene eseguito segretamente quando il programma/file infetto viene eseguito ed ha tutti i privilegi che ha l'utente che lancia il programma.
L'aggiunta di rigidi controlli negli ultimi anni ha portato alla creazione di macro virus che sfruttano i contenuti attivi(macro)supportati da alcuni documenti come excel,adobe,microsft word,questi documenti sono facilmente modificabili e non sono protetti dagli stessi controlli di accesso dei programmi.

15.7.1 Componenti di un virus

- Meccanismo di infezione: mezzo con cui il virus si diffonde permettendogli di replicarsi, chiamato anche VETTORE DI INFEZIONE
- Trigger: evento che determina l'inizio dell'attacco (chiamato anche BOMBA LOGICA)
- Payload: ciò che il virus fà,oltre a diffondersi.

15.7.2 Fasi del virus

- Fase dormiente: virus inattivo,al verificarsi di una condizione il virus si attiva(non tutti i virus hanno questa fase)

- Fase di propagazione: il virus inserisce una copia di se stesso in altri programmi,la copia potrebbe non essere del tutto identica in modo da sfuggire al meccanismo di rilevamento,ogni progrmma infetto ha un clone del virus che a sua volta entra nella fase di propagazione.
- Fase di attivazione: Il virus viene attivato per svolgere la funzione per cui è stato progettato,anche qui c'è un trigger che lo attiva(ingresso di un programma,creazione di n copie di se stesso,etc..)
- Fase di esecuzione: viene eseguita la funzione che può essere dannosa o innocua.

I virus sono specifici per SO e HW e sfruttano le loro vulnerabilità,i macrovirus invece colpiscono specifici tipi di documenti.

15.7.3 Macro virus

Macrovirus o codice di scripting,sono diventati negli ultimi anni la tipologia di virus più diffuso.
Si attacca a documenti e utilizza le macro(o codice di scripting) di questi documenti per propagarsi.
Sono molto dannosi e pericolosi perchè:
-Indipendenti dalla piattaforma,qualsiasi sistema operativo o hw che utilizzano questi documenti possono essere infettate
-Infettano documenti,in un computer sono molti i documenti presenti
-Si diffondono con facilità in quanto i documenti sono molto condivisi tra gli utenti(come ad esempio la posta elettronica)
-Dal momento che infettano documenti e non programmi di sistema, ci sono minori controlli degli accessi in quanto si suppone che siano gli utenti a modificarli.
-Sono più facili da creare rispetto ai viurs normali.
Negli ultimi anni sono aumentati i sistemi di protezione sulle macro in quanto tutte le macro dei documenti devono essere firmate e l'utente viene avvisato se un suo documento ha delle macro non firmate o con firme non attendibili,di per se gli utenti creano delle macro per svolgere operazioni ripetute.

15.7.4 Struttura di un macro virus

Un macrovisu di excel è diverso di uno di adobe,dipendono dal documento che attaccano.
Possono attivarsi da sole ad esempio all'apertura del documento o in altre circostanze,e possono eseguire diverse operazioni non solo sul documento che hanno infettato ma anche leggere/scrivere file e chiamare altre applicazioni.

15.8 Classificazione dei virus

Possiamo classificare il virus in base al target che vogliono attaccare

- Boot sector infector: infetta un master boot e si diffonde quando un sistema viene avviato
- File infector: infetta i file che il sistema operativo considera eseguibili
- Macro virus: infetta file con macro o codice di scripting
- Virus multipartito: infetta il file in diversi modi,infetta più tipi di file

Una seconda classificazione si può fare in base a come si nascondono all'interno del sistema

- Virus criptato: utilizzano la crittografia per occultare il prorpio contenuto, una porzione di codice di virus crea una chiave e cripta la restante parte,per decriptarlo utilizza sempre la chiave.Quando il virus si replica viene usata una nuova chiave diverse(essendo generata casualmente).Essendo che i virus sono cifrati con chiavi diverse non è possibile identificare pattern all'interno dei file.
- Virus furtivo: una forma di virus progettata proprio per sfuggire alle tecniche di rilevamento anti-virus,può usare mutazione del codice.
- Virus polimorfo: un virus che quando si replica crea copie di se stesso equivalenti ma con pattern diversi, per fare ciò può inserire operazioni inutili o cambiare l'ordine di operazioni indipendenti,altri approcci:crittografia dove anche la parte di codice che genera la chiave(mutation engin) viene aggiornata ad ogni utilizzo

- Virus metamorfico: come un virus polimorfo ma questo virus cambia completamente,cambiano aspetto e comportamento

15.9 Worm

Un worm è un programma che cerca attivamente altre macchine da infettare e poi ogni macchina infettata serve da rampa di lancio per infettare altre macchine.
I programmi worm sfruttano vulnerabilità nei programmi client e server per ottenere l'accessoad ogni nuovo sistema,possono utilizzare connessioni di rete per diffondersi tra i sistemi ma anche tramite USB.
Per replicarsi utilizza: email,condivisione di file(usb),esecuzione remota(esegue una copia di se stesso),accesso remoto ai file(per copiarsi da un file ad un altro accede in remoto ai file),login remoto(accede come utente ad un sistema e poi utilizza comandi per replicarsi).
Le fasi sono le stesse dei virus ma nella fase di propagazione svolge le seguenti attività:

- cerca i possibili host da infettare esaminando le tabelle degli host, le rubriche e gli elenchi di amici,..
- sfrutta i meccanismi di accesso individuati per trasferire una copia di se stesso al sistema remoto

15.9.1 Definizione di worm dettagliata

Un "worm" è un tipo di malware che si diffonde autonomamente attraverso le reti informatiche, senza la necessità di un'azione umana per la sua propagazione. A differenza dei virus, i worm non hanno bisogno di essere allegati a un file eseguibile o di infettare altri file per diffondersi.
I worm possono diffondersi attraverso diverse modalità, come la scansione di reti alla ricerca di vulnerabilità, l'utilizzo di exploit per sfruttare le vulnerabilità del sistema o l'invio di copie di se stessi a indirizzi email presenti sul sistema infetto.
Una volta che il worm ha infettato un sistema, può causare diversi problemi, come la saturazione della rete, il rallentamento del sistema, la cancellazione di file, il furto di informazioni sensibili e la compromissione della sicurezza del sistema.

15.10 Ricerca del target

La prima fase di propagazione consiste nel cercare altri sistemi da infettare ,questo processo si chiama SCANNING, le copie che infettano le macchine continuano questo processo finchè non si è creata un ampia rete di macchine infette

15.10.1 Strategie di scanning

- Random: ogni host compromesso analizza indirizzi ip casuali che genera un elevato volume di traffico nella rete che può causare problemi ancora prima dell'attacco
- Hit-list:l'hacker crea una lista di possibili macchine vulnerabili, processo lento e prolungato per evitare di farsi scoprire, una volta che la lista è completata si incominciano ad attaccare, ogni macchina ha una porzione di lista di macchine da attaccare
- topologica: utilizza informazioni contenute nelle macchine vittime per trovare altri host da infettare
- sottorete locale: se infetto un host "dentro" un firewall mi risulta più facile infettare altri host dentro il firewall in quanto non devo bypassare il firewall.

15.10.2 Worris Worm

Questo worm fu progettato per diffondersi su ambiente UNIX e fece uso di svariate tecniche di propagazione.
Quando una copia entrava su un host il suo primo compito era quello di scoprire altre macchine note a quell'host da poter attaccare,analizzava le tabelle degli host ritenuti per l'host attaccato "affidabili", i file di inoltro della posta elettronica,etc..

Per ciascun host scoperto provava una serie di tecniche per ottenere l'accesso:
-accede con login e password attraverso password cracking
-sfruttava bug nel protocollo finger di unix che riporta il luogo in cui si trova un utente remoto
-sfruttava un bug nel processo che inviava e riceveva posta elettronica Se il worm riusciva ad entrare inviava dei comandi di bootstrap con la shell, li eseguiva e poi si disconnetteva,questi comandi richiamavano il programma genitore e scaricavano la restante parte del worm.Ora il worm ha infettato un altra macchina e il processo si ripete.

15.10.3 Tecnologie del worm

- Multipiattaforma: i worm più recenti non rimangono circoscritti alle macchine windows ma anche UNIX.
- Multi-exploit: i worm più recenti penetrano sfruttando molti modi
- Diffusione ultra veloce: si sfruttano varie tecniche per rendere veloce la propagazione
- Polimorfismo: ogni copia del worm originale presenta delle modifiche del codice(cambia solo l'aspetto)
- Metamorfismo: ogni copia del worm cambia la funzione(la copia non è equivalente a quello originale)
- Mezzi di trasporto: ideali per diffondere un'ampia di attacchi dannosi,sono molto veloci ad espandersi
- Zero day exploit: I worm devono sfruttare vulnerabilità non note per essere ancora più efficaci

15.10.4 Mobile code

I mobile code sono programmi che possono essere trasmessi ad un insieme eterogeneo di piattaforme.
Viene trasmesso da un sistema remoto ad uno locale e poi eseguito in locale.
I worm sono apparsi per la prima volta su telefoni cellulari nel 2004,questi tipi di worm comunicano tramite bluetooth o tramite MMS, il malware dei telefoni possono disabilitare completamente il telefono,cancellare i dati o forzare il dispositivo ad inviare messaggi costosi.

15.11 Vulnerabilità drive by-downloads

Sfruttare bug delle applicazioni utente,quando un utente visualizza una pagina web gestita dall'attaccante, automaticamnte vengono scaricati malware senza che l'utente se ne possa accorgere o ne abbia dato il consenso.Questo funziona quando un utente visita il sito compromesso e sia vulnerabile agli exploit utilizzati.
Gli attacchi WATERINGHOLE sono una variante e utilizzano tecniche più mirate,l'attaccante ricerca le vittime designate per identificare i siti web che potrebbero andare a visitare.
l codice di attacco può anche essere scritto in modo da infettare solo i sistemi appartenenti all'organizzazione target e non intraprendere alcuna azione per conto di altri visitatori del sito.

15.12 Malvertising

E' una tecnica in cui si posizionano i malware su siti web senza effettivamente comprometterli.
L'attaccante compra annunci pubblicitari che con molta probabilità andranno inseriti nei siti web designati,questi annunci saranno compromessi, in questo modo quando l'utente entra su un sito web viene infettato grazie all'annuncio pubblicitario.(anche qui codice dinamico).
ALtri malware colpiscono pdf e infettano un utente solo se apre il pdf, questi documenti possono condividersi o tramite phising o posta indesiderata.

15.13 Clickjacking

Vulnerabilità utilizzata da un attaccante per acquisire i click di un utente (infettato).
L'attaccante forza l'utente a compiere determinate azioni senza rendersene conto, addirittura può modificare pagine web e spostare alcune icone per indurre l'utente a fare clic su un pulsante corrotto.
In pratica l'attaccante dirotta i clic destinati ad una pagina e li rindirizza ad un'altra pagina(molto probabilmente corrotta).
Allo stesso modo anche le sequenze di caratteri digitati su tastiera possono essere dirottati,un utente può essere indotto a credere di stare ad inserire la password su un sito sicuro quando in realtà la sta inserendo in un sito corrotto.

15.14 Propagazione con Social Engineering

Tecnica di progazione che consiste nell'ingannare gli utenti per favorire la compromissione dei loro sistemi.

15.14.1 Spam

Email non richieste dall'utente in cui cercano di invogliare l'utente a compiere determinate azioni.

15.14.2 Trojan horse

Un Trojan horse è un programma che contiene codice nascosto,che se invocato svolge alcune operazioni indesiderate.
Alcuni esempi: ci sono dei programmi che dichiarano di essere software antivirus, e una volta avviati in realtà parte un programma "nascosto".
Possono continuare a eseguire il programma "finto" quello che l'utente pensa di fare,in più eseguono codice nascosto.
Possono continuare ad eseguire solo la funzione originale ma modificata per svolgere attività dannose.
Possono svolgere solo la funzione nascosta senza eseguire quella "finta".
TROJAN HORSE PER TELEFONI CELLULARI
Ci sono trojan horse che copiano il template della pagina google o di siti bancari,molto usato sia su andorid che apple.
XcodeGost: utilizzato in app legittime, ma Xcode è un sistema che installa del malware appena si scarica l'applicazione.

15.15 Corruzione del sistema

Una volta che il malware è presente nel sistema,cosa farà?
Potrebbe distruggere i dati,rubare alcune informazioni,l'azione malevola potrebbe non verificarsi subito ma solo all'attivazione della bomba logica(al verificarsi di certe condizioni)

15.15.1 Chernobyl

Infetta i file eseguibili alla loro apertura,quando si verifica una certa condizione sovrascrive il primo megabyte del disco rigido con degli zeri.
Cercava anche di riscrivere il codice BIOS,se l'attacco aveva successo il computer era inutilizzabile

15.15.2 Klez

Si diffonde via email e fa si che i programmi anti-virus vengano disattivati e i file sul disco rigido vengono svuotati.

15.15.3 Ransoware

Questi malware criptano i dati del'utene e richiedono di pagare un riscatto per essere decriptati.
WannaCry criptava dati e chiedeva pagamenti in bitcoin.

15.16 Bomba logica

La bomba logica è una componente chiave dei malware,si tratta di un codice incorporato nel malware per esplodere quando si verificano determinate condizioni.

15.17 Zombie,bot

I bot assumono segretamente il controllo di un computer collegato ad internet e,attraverso il bot,lancia attacchi che sono difficili da ricondurre al creatore del bot.Un bot viene installato su centinaia o migliaia di computer,più bot possono lavorae in modo coordinato,un insieme di bot che lavora insieme si chiama BOTNET.

15.17.1 Uso dei bot

- Attacchi DDos: perdita del servizio
- Spamming: si possono inviare migliaia di email spam
- Sniffing del traffico: utilizzano sniffer per intereccetare dati che viaggiano in chiaro che viaggiano su una macchina compromessa
- Keylogging: cattura le sequenze di caratteri che vengono digitati sulla tastiera per recuperare dati sensibili
- Diffusione di nuovo malware: le botnet sono utilizzate per diffondere nuovi bot
- Installazione di componenti aggiuntivi pubblicitari e di browser helper object(BHO): creo un sito web e metto annunci pubblicitari e mi faccio pagare in base a quanti click fanno,uso bot che cliccano da soli
- Attacchi a reti: usati per attaccare le reti
- Manipolazione di sondaggi/giochi online: il bot simula l'utente e può simulare sondaggi

Un worm si propaga e si attiva da solo, il bot è controllato da un server chiamato CC, non deve essere di continuo ma può essere attivato periodicamente quando il bot rileva di avere accesso alla rete.
La funzione di controllo inzialmente usava meccanismi IRC,tutti i bot si collegavano ad un canale specifico su questo server e consideravano i messaggi ricevuti come comandi.
Oggi comunicano tramite HTTP,utilizzano ip generati in modo automatico con cui il malware entra in contatto.
La CC di una botnet può essere organizzata in diversi modi, ma in generale, l'infrastruttura è composta da almeno due elementi: il server di comando e controllo e i bot. Il server di comando e controllo è un server remoto gestito dagli operatori della botnet, che utilizzano questo server per inviare comandi ai bot. I bot, invece, sono i computer compromessi che sono stati infettati dal malware della botnet e che ascoltano i comandi inviati dal server di comando e controllo.
Quando un computer viene infettato da un malware di una botnet, diventa un "bot" controllato dal server di comando e controllo. Il malware installato sul computer contatta il server di comando e controllo per registrarsi e aspetta di ricevere comandi. Gli operatori della botnet possono quindi inviare comandi al server di comando e controllo, che a sua volta li inoltra ai bot. I bot eseguono i comandi, che possono includere la raccolta di informazioni sul computer infettato, l'invio di spam, l'attacco di altri computer o la partecipazione a una rete di botnet distribuita per eseguire attacchi DDoS.
Per mascherare la loro attività, gli operatori delle botnet spesso utilizzano tecniche di evasione, come l'uso di server di comando e controllo distribuiti o la crittografia del traffico di rete. Tuttavia, le forze dell'ordine e i fornitori di servizi di sicurezza informatica stanno sempre cercando di identificare e disattivare i server di comando e controllo delle botnet per prevenire la loro attività dannosa.

15.17.2 Keylogger e spyware

Keylogger intercetta quello che un utente scrive su tastiera,mentre spyware monitora l'attivtà di un utente su un computer come cronolgoia,contenuto di attività di navigazione.

15.17.3 Phishing

Pagine identiche a pagine vere di siti bancari o altro che provano ad ingannare l'utente.
Vengono invate tramite email con botnet.
Una variante ancora più pericolosa SPEAR-PHISING.
Arrivano email molto specifiche per il destinatario,in cui sono presenti dei suoi dati,in questo modo l'utente è ancora più propenso a credere.

15.18 Backdoor

Un backdoor detto anche trapdoor,è un punto d'accesso che permette di acquisire l'accesso al sistema evitando le consuete procedeure di accesso di sicurezza.
Sono usate anche in modo legittimo dagli utenti per testare i programmi evitando ogni volta le procedure di accesso.
La backdoor è un codice che riconosce una sequenza speciale di input o che viene attivato da un certo utente o da un'improbabile sequenza di eventi,se succedono una di queste cose si bypassano tutti i sistemi di sicurezza.
Diventa una minaccia quando se ne fa un uso improprio.

15.19 Rootkit

E' un insieme di programmi installati su un sistema per mantenere un accesso segreto ad esso con privilegi di amministratore,nascodento le prove della propria presenza,può atturare delle modifiche senza farsi vedere,ha tutti i privilegi possibili.
Classificazione:

- Persistente: si attiva ogni volta che il sistema entra in funzione,il codice deve essere memorizzato in modo persistente in un archivio,facile da rilevare
- Residente in memoria: non dispone di codice persistente quindi non sopravvive ad un riavvio,può risultare difficile da rilevare visto che si trova solo in memoria
- Modalità utente: intercetta le chiamate alle API e altera i risultati restituiti
- Modalità kernel: il rootkit può nascondere la presenza del malware rimuovendolo dalla lista dei processi attivi del kernel
- Basato su macchina virtuale: il codice è invisibile al sistema colpito in quanto si trova su una macchina virtuale
- Modalità esterna: il malware accede direttamente all'hardware,si tira fuori dalla normale modalità di funzionamento del sistema colpito.

15.20 Rootkit in modalità kernel

Alcuni rootkit apportano modifiche al kernel per nascondere la loro presenza,uno stesso programma antivirus potrebbe essere soggetto a modiche di basso livello che fa il malware dall' "interno". Cosa succede se un rootkit agisce nel kernel?gli utenti interagiscono con il kernel tramite syscall,c'è una tabella con gli indici univoci che identificano le porzioni di codici di ogni syscall.

- Modificare la tabella: le syscall vengono completamente deviate
- Modifca i target della tabella: l'attaccante modifica il codice della syscall
- Reindirizza la tabella delle chiamate a sistem: l'attaccante reindirizza riferimenti all'intera tabella ad una nuova tabella che si trova in un altra locazione di memoria del kernel.

15.21 Contromisure

- Policy
- Sensibilizzazione
- Mitigazione delle vulnerabilità
- Mitigazione delle minacce

Per una corretta prevenzione dovremmo assicurarsi che tutti i sistemi siao aggiornati, impostare rigidi controlli di accesso, formazione degli utenti per non farsi ingannare dall'ingegneria sociale.
Se le misure di prevenzione falliscono è possibile ricorrere a meccanismi tecnici quali

- Rilevamento: rilevare il malware
- Identificazione: identificare quale malware ho rilevato
- Rimozione: rimuovere il malware e tutte le modifiche che ha fatto

Se il rilevamento ha avuto successo, ma gli altri due step no: allora bisogna eliminare ogni file infetto e ricercare una versioen di backup pulita.
Requisiti per effettuare contromisure efficaci

- Generalità: sapere gestire un'ampia varietà d'attacchi
- Tempestività: l'approccio dovrebbe rispondere in modo tempestivo per evitare troppi danni
- Resilienza: dovrebbe funzionare anche contro le tecniche dei malware per nascondersi
- Costi minimi di DoS: l'approccio non dovrebbe influenzare il l'esecuzione normale del sistema
- Trasparenza: non dovrebbero richiedere modifiche ai sistemi operativi o sistemi o hardware presenti
- Copertura globale e locale: dovrebbe funzionare per attacchi sia interni che esterni alla rete in cui è connesso

15.22 Generazioni di software antivirus

- 1 gen: scanner semplici, identifica malware noti, basato sulla firma del malware(stessa struttura in tutte le copie o caratteri jolly)
- 2 gen: scanner euristici,cerca frammenti di codici di norma associati a malware..,non deve per forza essre un malware noto,un approccio potrebbe essere il controllo dell'integrità
- 3 gen: trap di attività,identificano un malware in base alle sue azioni e non alla sua struttura,questi programmi risiedono in memoria
- 4 gen: protezione completa,sono package formati da varie tecniche antivirus usate in combinazione,incorporano funzionalità di controllo dell'accesso che limitano la capacità di penetrare per il malware.

15.23 Sandbox analysis

E' un metodo per rilevare malware, si esegue codice potenzialmente dannoso in una sandbox o su macchina virtuale,in modo che il suo comportmento può essere controllato,l'esecuzione in sandbox permette il rilevamento di malware.
Consente l'esecuzione del codice in un ambiente controllato dove il suo comportamento può essere attentamente monitorato senza minacciare il sicurezza di un sistema reale.
Esecuzione di software potenzialmente dannoso in tali ambienti consente il rilevamento di complessi crittografati, polimorfici o malware metamorfico.
Il problema di progettazione più difficile con l'analisi sandbox è quello di determinare per quanto

tempo eseguire ciascuna interpretazione. Malware più sofisticati non si attivano subito ma al contrario utilizzano tecniche di evasione quali sospesione prolungata(il codice del malware rimane idle per molto tempo) in modo che la sandbox non è in grado di riconoscere l'attacco(se interrompe la simulazione di un programma troppo presto(prima che cominci l'attacco effettivo)). Più a lungo dura la simulazione, più è probabile trovare malware. Ma il sandobox ha disponibilità limitata di tempo e di risorse dato che deve analizzare tanti potenziali malware.
E' una sfida continua in cui gli hacker verificano se stanno su una sandbox o ambienti virtuali,in modo da modificare il comportamento del malware in questo caso. Implementano bombe logiche per innescare l'attacco in modo da ingannare la sandbox,allo stesso modo gli analisti stanno migliorando le sandbox.

15.24 Analisi dinamica host-based dei malware

"Host-based behavior blocking software" è integrata con il sistema operativo di un host e controlla l'esecuzione dei programmi in tempo reale,cercano di anticipare e bloccare azioni malevoli prima che possano danneggiare il target.Analizzano in tempo reale cosa vuole fare il programma che è in esecuzione in modo da bloccarlo(eventualmente).

15.25 Approcci di scansione perimetrale

Si possono utilizzare software anti-virus sui firewall e sugli IDS di un'organizzazione,in modo da bloccare il flusso di traffico anomalo,tale approccio si limita ad osservare il contenuto del malware,in quanto non può analizzare il comportamento(essendo pacchetti in transito nella rete).
2 tipi di monitoraggi

- Monitor d'ingresso: Posizionati al confine tra la rete aziendale e Internet,fanno parte del software di filtraggio dei pacchetti del router o del firewall. Esempio di tecnica: ricerca di traffico in entrata verso IP locali inutilizzati.
- Monitor d'uscita: Posizionati nel punto d'uscita delle singole LAN e far parte del software di filtraggio del router in uscita. Cobtrolla il traffico in uscita monitorando ad esempio un eccessivo numero di email inviate.

16 Denial of Service

Un'azione che impedisce o compromette l'uso autorizzato di sistemi esaurendo le risorse che servono per rendere disponibili quei sistemi(cpu,link di comunicazione).
3 categorie di risorse che possono essere attaccate

- Larghezza di banda di rete: capacità dei link di rete che collegano un server ad Internet(ISP)
- Risorse di sistema: Un attacco DoS che prende di mira le risorse di sistema in genere mira a sovraccaricare o arrestare in modo anomalo le sue software di gestione della rete.
- Risorse applicative: In genere comporta un numero di richieste valide, ognuna delle quali consuma risorse significative, limitando così la capacità del server di rispondere a richieste di altri utenti.

16.1 Classici attacchi DoS

16.1.1 Flooding ping command

Lo scopo è quello di sovraccaricare la capacità della rete di uno specifico bersaglio.
Se un attaccante ha accesso ad un sistema con una connessione di rete più grande rispetto ad un altro sistema(quello scelto come vittima), allora l'attaccante può generare traffico elevato in modo che il sistema vittima non è in grado di gestire.
La rete del sistema vittima prende solo i pacchetti che può gestire,gli altri li scarta, ma quelli che prende consumano tutte le risorse del sistema,di conseguenza gli altri pacchetti avranno basse possibilità di sopravvivere.
In questo attacco la fonte dell'attaccante è identificata e questo è uno svantaggio,inoltre essendo l'ip

reale,la vittima risponderà alle richieste andando a creare ancora più traffico nella rete,ciò comporta un aumento delle possibilità che l'attacco venga rilevato.
L'attaccante vorrebbe attaccare con ip falsificati.

16.1.2 Source Address Spoofing

I pacchetti usati dagli attaccanti usano IP sorgenti falsificati, con un accesso sufficientemente privilegiato è possibile creare pacchetti con IP falsificati tramite l'interfaccia RAW-Socket.
L'attaccante invia pacchetto con ip x (preso random) se x esiste la vittima risponde ad x,altrimenti il pacchetto viene scartato.
Difficile da rilevare.

16.1.3 SYN spoofing attack

Questo attaccao ha l'obiettivo di riempire la tabella delle richieste di connessione TCP,in una normale richiesta di connessione TCP abbiamo il client che fa una richiesta al server, il server registra in una tabella di richieste di connessione il suo ip e risponde con SYN-ACK, ora se il client ha realmente chiesto lui una connessione risponderà con ACK e la connessione viene instaurata. Se il client non ha mai chiesto la connessione invia un RST e la connessione viene rifiutata. In entrmabi i casi gli ip escono dalla tabelle di richieste di connessione.
Se l'attaccante genera un numero enorme di IP random(alcuni saranno veri altri inesistenti) e fa richieste TCP ad un server, quelli veri risponderanno con un RST in quanto non hanno mai chiesto loro la connessione.
Quelli falsi(non esistono) non risponderanno mai e il server, visto che usa TCP che è un protocollo sicuro dovrà rinviare la richiesta per n volte fino a quando non dichiara chiusa la connessione.
Il tempo che intercorre tra il server che dichiara chiusa la connessione è talmente elevato che la tabella di richieste di connessione TCP si va a riempire in quanto vengono generate moltissime richieste con IP falsificati. Di conseguenza gran parte delle richieste vere di utenti legittimi verranno scartate poichè la tabella è piena. Il server è tagliato fuori da Internet.
L'obiettivo dell'attaccante è generare più ip inesistenti possibili in modo che rimangano per molto tempo all'interno della tabella.

16.1.4 Flooding attacks

Assumono diverse forme a seconda del protocollo di rete usato per implementare l'attacco, in tutti i casi lo scopo è di sovraccaricare la rete.
Qualsiasi tipo di pacchetto va bene: ICMP,UDP,TCP,SYN..

16.2 Distributed Denial of Service:DDoS

Uso di più sistemi per generare attacchi.
L'attaccante utilizza una falla nel sistema operativo per ottenere l'accesso e installare il suo programma su di esso(zombie).
Può formare una botnet(tanti zombie sotto il controllo dell'attaccante).
Si usa una struttura ad albero, l'attaccante invia comandi ad alcuni zombie che inoltrano al loro rispettivo sottoalbero e cosi via..

16.2.1 Http-Based-Attacks

Due approcci

- HTTP FLOOD: attacco che bombarda il server web con richieste http, un attacco DDos che proviene da tanti bot diversi,le richieste sono operazioni lente
  Variante: Http flood ricorsivo, i bot partono dal link http e in ricorsione fanno richieste a tutti i link forniti dal sito web,operazione chiamata SPIDERING.

- SLOWLORIS: si cerca di monopolizzare tutti i thread che un server web usa per gestire in parallelo le richieste.Le richieste http stabiliscono che deve essere utilizzata una riga vuota per indicare la fine della richiesta,terminata la richiesta il server può rispondere,ma se le richieste

non hanno la riga vuota?Quest attacco si basa su questo concetto,invio richieste senza riga vuota a tutti i thread cosi la connessione non si interrompe mai e ogni thread rimane attivo. Tutte le altre richieste legittime verrano scartate in quanto non ci sono thread disponibili per analizzarle. Differisce da altri attacchi in quanto non sono richieste cattive,semplicemente non terminano mai. Difficile da rilevare

16.2.2 Reflection attacks

L'attaccante invia richieste ad un servizio noto,l'ip con il quale fa richieste è l'ip della vittima. Il server risponde alla vittima. Se il numero di richieste è troppo elevato la vittimi si sovraccarica,c'è il rischio che anche il servizio noto si sovraccarichi,infatti si cerca di avere un servizio con capacità maggiori del servizio vittima. Difficile da rintracciare in quanto ci sono server intermediari che inviano richieste non fatte da loro.
2 varianti

- Simple reflection attack: l'attaccante invia all'intermediario un pacchetto di dimensione x e vuole che lui risponde all'attaccante con pacchetto di dimensioni molto maggiore rispetto ad x,i servizi udp fanno questo. L'obiettivo è sovraccaricare il bersaglio senza allertare l'intermediario. Se l'attaccante distribuisce in modo ciclico le richieste ad n intermediari diventa ancora più difficile capire da dove provengono.
- Amplification attack: genero più pacchetti di risposta per ogni pacchetto inviato. Invio il pacchetto in broadcast,tutti gli host su quella rete possono potenzialmente rispondere generando una marea di risposte.

16.2.3 DNS Amplification attacks

Sfrutta il comportamento del DNS per convertire una piccola richiesta in una risposta più grande.

16.3 DoS attack Defense

Si possono avere diversi accorgimenti ma non si può evitare del tutto.
A volte i siti si intasano in modo casuale e non per un attaccante.
Si usano termine come slashdotted,flashcrowd,flashevent. Non ci si può fare niente.
4 linee di difesa contro DoS o DDoS

- Prevention: bloccare indirizzi contraffatti,utilizzare TCP modificate(eliminare una entry di connessione incompleta quando la tabella va in overflow),bloccare IP in broadcast,gestire richieste con cose tipo "io non sono un robot"
- Detection: filtrare i pacchetti che potrebbero far parte dell'attacco
- Traceback: rintracciare l'attaccante per prevenire attacchi futuri
- Reaction: limitare gli effetti di un attacco

16.3.1 Responding

Per rispondere con successo a un attacco DoS, è necessario un buon piano di risposta agli incidenti. Questo deve includere i dettagli su come contattare il personale tecnico per il tuo ISP. Questo contatto deve essere possibile utilizzando mezzi non in rete, poiché quando sei sotto attacco la tua connessione di rete potrebbe non essere utilizzabile.
Identificare il tipo di attacco,filtrare e tracciare i pacchetti,avere un piano di ripresa del sistema, impedire un futuro attacco.

17 Intrusion detection

Classi di intrusioni

- Cyber criminals: Sono individui di un gruppo criminale organizzato con l'obiettivo di ottenere una ricompensa finanziara.
  Cosa fanno: furto d'identità,furto di credenziali,furto di dati.
  Si incontrano in forum clandestini per scambiarsi suggerimenti
- Activists: Individui che lavorano insider o membri che lavorano in gruppo motivati da ragioni politiche.
  Cosa fanno: DoS,distruzione dei dati,..
- Sponsored organizations: Gruppi di hacker sponsorizzati dallo stato per effettuare attività di spionaggio

Diversi livelli di abilità

- Apprendista: Gli hacker con competenze minime che usano toolkit di attacco.E' facile difendersi in quanto sono attacchi gia noti.
- Jouneyman: hacker con competenze sufficienti a scoprire nuove vulnerabilità o ad estendere l'attacco ad un target specifico. Difendersi è difficile.
- Master: Hacker in grado di scrivere nuovi toolkit potenti. Molto difficile difendersi

Gli attacchi possono essere benigni o maligni.
Esempi di intrusioni

- Indovinare password
- Copiare un database
- Visualizzare dati sensibili senza autorizzazioni
- Accedere ad un modem per ottenere l'accesso alla rete interna, in maniera non autorizzata

Le tecniche di intrusion detection possono essere efficaci contro attacchi noti,meno su attacchi fatti da Master che usano mirate su determinate target in cui scoprono delle vulnerabilità nuove(attacchi zero-day)

17.1 Intrusion behavior(comportamento delle intrusioni)

Le tecniche e i comportamenti cambiano di continuo,ma le fasi sono sempre le stesse

- Target Acquisition and Information Gathering(raccolta informazioni): L'attaccante identifica e caratterizza i sistemi bersaglio utilizzando informazioni disponibili al pubblico.
- Initial Access: Si ottiene accesso al sistema target sfruttando vulnerabilità (password deboli,malware)
- Privilege Escalation: Azioni intraprese per incrementare i privilegi disponibili all'aggressore per raggiungere gli obiettivi desiderati.
- Information Gathering: L'attaccante accede con i permessi per raccogliere informazioni
- Maintaining Access: Azioni che servono per mantenere l'accesso come:aggiunta di credenziali.
- Covering Tracks(copertura delle tracce): Uso di roolkit per nascondere sistemi installati o disabilita registri per rimuovere le tracce

Vediamo un esempio reale di Intrusion behavior

- Target Acquisition: Esplorare sito web per ottenere delle info, inviare email per vedere la risposta e ottenere informazioni sulla base della risposta
- Initial Access: Brute force sulla password del sistema,sfruttare vulnerabilità, inviare email di spear-phising per ingannare l'utente a inserire la password

- Privilege Escalation: Sfruttare vulnerabilità per ottenere privilegi elevati, rubare password amministratore
- Information Gathering: Eseguire scansione file alla ricerca di informazioni desiderate.
- Maintaining Access: Modificare/disabilitare programmi antivirus, installare strumenti di amministrazione remota
- Covering Tracks: Uso di rootlik per nascondere file installati.

Aclune definizioni

- Security Intrusion: Azione non autorizzata per bypassare meccanismi di sicurezza di un sistema
- Intrusion Detection: Meccanismi per identificare possibili intrusioni nel sistema

17.2 Intrusion Detection System(IDS)

Un IDS comprende tre componenti logiche:

- Sensori: responsabili della raccolta dati,in input prendono qualsiasi parte del sistema che potrebbe contenere prove di intrusione. Raccolgono queste informazioni e le inviano all'analizzatore.
- Analizzatore: Gli analizzatori ricevono in input da uno o più sensori ed è responsabile di verificare se si è verificata un intrusione. Può fornire indicazioni sulle azioni da intraprendere a seguito di un intrusione.
- User interface: Consente ad un utente di controllare il comportamento di un sistema.

Un IDS può usare un singolo sensore e analizzatore,come un HIDS su un host, o un NIDS su un dispositivo firewall,IDS più sofisticati hanno più sensori che inviano ad un analizzatore centralizzato.
Classificazione degli IDS

- Host-based IDS(HIDS): Monitorano le caratteristiche di un singolo host e gli eventi che si verificano.
- IDS basati su rete(NIDS) monitorano il traffico di rete.
- IDS ibridi: combina informazioni provenienti da diversi sensori sia host che network e li invia ad un analizzatore centrale.

17.3 Motivazioni IDS

Se un intrusione viene rilevata con sufficiente rapidità,l'intruso può essere identificato ed espulso dal sistema. Anche se non è sufficientemente tempestivo, meglio tardi che mai.

17.4 Postulato di Anderson

Si può distinguere tra un attaccante esterno e un utente legittimo construendo un modello di operazioni svolte da utenti legittimi.
Ma individuare attacchi provenienti dall'interno(attaccanti che entrano nel sistema spacciandosi per altri) che agiscono in modo non autorizzato è più difficile.

17.5 Requisiti di un IDS

Eseguire continuamente una minima supervisione umana.
Essere in grado di riprendersi dopo un crash.
Resistere alle sovversioni:capire se il programma è stato modificato da un attaccante controllando se stesso.
Overhead minimo.
Monitorare un gran numero di host.
Se alcuni componenti dell'ids smettono di funzionare deve essere comunque in grado di funzionare.
2 approcci

- Anomaly detection: Viene analizzato il comportamento in tempo reale dell'applicazione per vedere se rispetta un modello di comportamento legittimo(costruito raccogliendo comportamenti di utenti legittimi)
- Signature detection: Utilizza un insieme di regole/procedure d'attacco note che vengono confrontate con il comportamento attuale.

17.5.1 Anomaly detection

3 tipi di approcci

- Statistical: Analisi del comportamento usando modelli.
- Knowledge based: classifica il comportamento in base ad un insieme di regole che modellano il comportamento legittimo.
- ML: determinano automaticamente un modello di classificazione a partire dai dati di addestramento con tecniche di data mining.

17.5.2 Signature Detection

2 tipi di approcci

- Signature approches: Confrontano un'ampia raccolta di modelli noti di dati dannosi con dati memorizzati sul sistema o in rete. Se c'è una differenzia sostanziale si rileva l'attacco,la differenza deve essere abbastanza grande da non rilevare falsi allarmi. Utilizzato negli anti virus.
- Rule-based heuristic: Prevede l'uso di regole per identificare penetrazioni note,le regole definiscono anche comportamenti sospetti.

17.6 Host-Based Intrusion Detection(HIDS)

Aggiungono un software specializzato nei sistemi vulnerabili e monitora l'attivitè in vari modi per rilevare comportamenti sospetti.
Il suo scopo principale è rilevare intrusioni, registrare eventi sospetti e inviare avvisi. Il vantaggio principale di un HIDS è che può rilevare intrusioni esterne e interne.
Possono utilizzare sia anomaly detection che signature detection per rilevare comportamenti non autorizzati.

17.6.1 Data Sources and Sensors

Componente fondamentale del rilevamento intrusioni è il sensore che raccoglie i dati.
Dati comuni

- File log: La maggior parte dei sistemi operativi moderni include un software di che raccoglie informazioni sulle attività degli utenti.Gli svantaggi sono che le registrazioni potrebbero non contenere le informazioni necessarie o non le contengono in una forma conveniente, e che gli intrusi possono tentare di manipolare i registri per nascondere le proprie azioni.
- accesso registro di sistema
- checksum integrità dei file: Un approccio comune per rilevare l'attività di un intruso in un sistema è quello di scansionare periodicamente i file critici per verificare se ci sono state modifiche rispetto alla di base desiderata, confrontando le checksum crittografiche correnti di questi file con un record di valori buoni conosciuti. Gli svantaggi includono la difficoltà di monitorare i file che cambiano.
- tracce di chiamate a sistema: Una registrazione della sequenza di chiamate di sistema da parte dei processi su un sistema,è ampiamente riconosciuta come la fonte di dati preferita per gli HIDS. Lavora bene su Linux un pò meno su Windows.

17.6.2 Anomaly detection

Il maggior lavoro si basa analizzando le chimate a sistema,Le chiamate di sistema sono il mezzo con cui i programmi accedono alle funzioni centrali del kernel, fornendo un'ampia gamma di interazioni con le funzioni di basso livello del sistema operativo.
Per questo motivo forniscono informazioni dettagliate sull'attività del processo che possono essere utilizzate per classificarla come normale o anomala.

17.6.3 HIDS basato su firma o euristica

L'alternativa degli HIDS basati su firme o euristiche è ampiamente utilizzata, in particolare nei prodotti antivirus (A/V), più correttamente considerati come antimalware. Questi sono molto utilizzati sui sistemi Windows e sono anche incorporati nei proxy di posta e di applicazioni web nei firewall e negli IDS di rete. Utilizzano un database di firme di file, che sono schemi di dati trovati in software noti o regole euristiche che caratterizzano un comportamento dannoso noto. Questi prodotti sono abbastanza efficienti nel rilevare le minacce informatiche note, ma non sono in grado di rilevare le minacce zero-day che non corrispondono alle firme o alle regole euristiche conosciute.

17.6.4 Distributed HIDS

Coordinamento di più HIDS.
Un IDS distribuito potrebbe dover gestire diversi formati di dati dei sensori. In un ambiente eterogeneo, i diversi sistemi possono utilizzare sensori e approcci diversi per raccogliere dati e approcci diversi alla raccolta dei dati per il rilevamento delle intrusioni.
È possibile utilizzare un'architettura centralizzata o decentralizzata. Con un'architettura centralizzata centralizzata, c'è un unico punto centrale di raccolta e analisi di tutti i dati dei sensori. Questo facilita il compito di correlare le segnalazioni in arrivo, ma crea un potenziale collo di bottiglia e un singolo punto di guasto. Con un'architettura decentralizzat, c'è più di un centro di analisi, ma questi devono coordinare le loro attività e scambiare informazioni.
3 componenti principali:
-Host agent module: Raccoglie dati dell'host e li invia al gestore centrale.
-LAN monitor agent module: Funziona come l'host agent ma analizza il traffico LAN e l'invia al gestore centrale.
Central manager module: Riceve tutti i dati e li combina per rilevare intrusioni, lo schema è stato progettato per essere indipendente da ogni sistema operativo

17.7 NIDS

Monitora il traffico in punti selezionati della rete.
Esamina in tempo reale i pacchetti.
Può esaminare la rete a livello applicativo/trasporto/rete.
Analisi presso il sensore.
I NIDS sono tipicamente inclusi nell'infrastruttura di sicurezza perimetrale di un'organizzazione, incorporati o associati al firewall.
Con il crescente utilizzo della crittografia,tuttavia, i NIDS hanno perso l'accesso a contenuti significativi, Pertanto,possono solo formarsi parte della soluzione in quanto non riescono ad analizzare il contenuto effettivo per trovare anomalie.
Una tipica struttura NIDS comprende una serie di sensori per il monitoraggio del traffico di pacchetti, uno o più server per le funzioni di gestione del NIDS e una o più console di gestione per l'interfaccia umana.
L'analisi dei modelli di traffico per rilevare le intrusioni può essere effettuata dal sensore, dal server di gestione o da una combinazione dei due.

17.7.1 Tipi di sensori in rete

- In linea: un sensore in linea viene inserito in un segmento di rete in modo che il traffico che sta monitorando debba passare attraverso il sensore, vantaggio:possono bloccare l'attacco se viene rilevato.

- Passivo: Controlla una copia del traffico di rete,il pacchetto effettivo non passa attraverso il sensore. E' più efficiente in quanto non aggiunge un dispostivo nel link di comunicazione e l'invio/ricezione dei pacchetti è più veloce.

17.8 Distribuzione dei sensori

In una rete aziendale molto grande, sono necessari diversi sensori e il loro posizionamento diventa fondamentale, i sensori possono essere situati all'interno o all'esterno del firewall,ci sono pro e contro. Inseriti all'interno del firewall hanno la possibilità di filtrare i pacchetti che riescono ad entrare nel firewall anche se sono malevoli. Il carico di lavoro è minore rispetto a se fosse esterno al firewall ma non ha la possibilità di vedere effettivamente tutti gli attacchi che vengono fatti.
Un sensore esterno invece ha la possibilità di controllare tutto il traffico in entrata ,carico di lavoro elevato,non ha la possibilità di vedere i pacchetti che navigano all'interno della rete aziendale.

17.8.1 Tecniche di intrusion detection

Come per il rilevamento delle intrusioni basato sull'host, il rilevamento delle intrusioni basato sulla rete fa rilevamento delle firme e rilevamento delle anomalie. A differenza del caso degli HIDS, sono disponibili diversi prodotti NIDS commerciali per il rilevamento delle anomalie.
Vediamo alcuni attacchi adatti alla signature detection

- Attacchi a livello applicativo: Protocolli che analizzano: FTP,HTTP,IMAP,POP.Il NIDS è alla ricerca di modelli di attacco che sono stati identificati come mirati a questi protocolli
- Attacchi a livello trasporto: Analizza TCP,UDP, Esempi di attacchi sono frammentazione insolita dei pacchetti, scansioni di porte vulnerabili e attacchi specifici al TCP, come i SYN flood.
- Attacchi a livello rete: I NIDS analizzano in genere IPv4, IPv6, ICMP e IGMP a questo livello. Esempi di attacchi sono gli indirizzi IP e valori illegali dell'intestazione IP.
- Servizi applicativi inattesi: Il NIDS cerca di determinare se l'attività su una connessione di trasporto è coerente con il protocollo di applicazione previsto. Un esempio è un host che esegue un servizio applicativo non autorizzato.
- Violazioni delle politiche: Esempi sono l'uso di siti Web inappropriati e l'uso di protocolli applicativi vietati.

Attacchi adatti all'anomaly detection

- DoS
- Scanning:strumenti software per cercare di individuare porte di rete aperte e servizi esposti su un sistema target. Questi strumenti eseguono una scansione della rete cercando di identificare le porte aperte su cui i servizi sono in ascolto, e quindi tentano di connettersi ad essi per identificare eventuali vulnerabilità del sistema.
- Worm:I worm possono essere individuati anche perché possono far comunicare tra loro host che di solito non lo fanno, possono anche far sì che gli host utilizzino porte che di solito non usano.

17.9 Stateful Protocol Analysis(SPA)

Sottinsieme del rilevamento delle anomalie che confronta il traffico di rete osservato rispetto al protocollo del traffico di rete benigno universale predeterminato.
Questo lo distingue dalle tecniche addestrate con protocolli di traffico specifici dell'organizzazione. Tiene traccia degi stati dei protocolli di rete,trasporto,applicazione per accertarsi che procedano come previsto.
Svantaggio:elevato utilizzo di risorse che richiede.

17.9.1 Logging alerts(registrazione degli avvisi)

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 affinare i parametri e gli algoritmi di rilevamento delle intrusioni. L'amministratore della sicurezza può utilizzare queste informazioni per progettare tecniche di prevenzione. Le informazioni tipiche registrate da un sensore NIDS includono le seguenti informazioni:
-Timestamp
-Id sessione
-Tipo di evento
-Protocollo(violato)
-Indirizzi ip sorgente e destionazione
-Porte TCP/UDP
-Informazioni sullo stato
-Numero di byte trasmessi

17.10 Honeypots

Sistemi esca progettati per: attirare un potenziale attaccante lontano dai sistemi critici,raccogliere informazioni sull'attivtà dell'aggressore,incoraggiare l'aggressore di rimanere nel sistema a lungo per permettere agli amministratori di rispondere.
I sistemi sono pieni di informazioni inventate dove un utente legittimo non ne avrebbe accesso,tutte le comunicazioni in entrata con molta probabilità sono attacchi, una comunicazione in uscita suggerisce che il sistema è stato compromesso.

17.10.1 Classificazioni Honeypot

- A bassa interazione: Consiste in un pacchetto software che emula particolari servizi IT per fornire un'interazione iniziale realistica ma non esegue una versione completa di tali servizi. E' sufficiente per essere utilizzato come componente di ids distribuito per avvisare un attacco imminente.Identifica un obiettivo meno realistico.

- Ad alta interazione: Un sistema reale con un sistema operativo completo,Può occupare un attaccante per un periodo prolungato,richiede un numero significativo maggiore di risorse,se compromesso potrebbe utilizzato per avviare attacchi ad altri sistemi.Identifica un obiettivo più realistico.

La ricerca più recente si è concentrata sulla costruzione di intere reti honeypot che emulano un'azienda, possibilmente con traffico e dati reali o simulati.
Una volta che gli hacker si trovano all'interno della rete, gli amministratori possono osservare il loro comportamento in dettaglio e definire le difese.
Possono essere distribuiti in diversi luoghi a seconda del tipo di informazioni che l'organizzazione è interessata a raccogliere e il livello di rischio che l'azienda deve affrontare.
Honeypot all'esterno del firewall è utile per tracciare tentativi di connessione a indirizzi IP, non aumenta il pericolo per la rete interna. Anzi riduce il carico di attacchi che può avere un firewall,svantaggio: non è in grado di intrappolare aggressoi interni. Honeypot interni al firewall l'opposto invece.
La rete di servizi disponibili all'esterno, come il Web e la posta, spesso chiamata DMZ (demilitarized zone), è un altro candidato per l'individuazione di una honeypot.
L'amministratore della sicurezza deve assicurarsi che gli altri sistemi nella DMZ siano al sicuro da qualsiasi attività generata dall'honeypot. Uno svantaggio di questa posizione è che una tipica DMZ non è completamente accessibile e il firewall di solito blocca il traffico verso la DMZ. Il firewall di solito blocca il traffico verso la DMZ che tenta di accedere a servizi non necessari.
Un honeypot completamente interno presenta diversi vantaggi. Il vantaggio più importante è che vantaggio più importante è quello di poter intercettare gli attacchi interni. Una honeypot in questa posizione può "correggere" un firewall mal configurato che inoltra il traffico non consentito da Internet alla rete interna. Ci sono diversi svantaggi. Il più grave è che l'honeypot può essere compromesso in modo da poter attaccare altri sistemi interni. Un'altra difficoltà per questa posizione dell'honeypot è

che, come per honeypots su DMZ, il firewall deve adattare il suo filtraggio per consentire il traffico verso l'honeypot, complicando così la configurazione del firewall e potenzialmente compromettendo la rete interna.

17.11 Example IDS:SNORT

Snort è un IDS open source, altamente configurabile e portatile, basato su host o su rete. IDS. Snort è definito un IDS leggero, con le seguenti caratteristiche:
- Facile da implementare sulla maggior parte dei nodi (host, server, router) di una rete.
- Funzionamento efficiente che utilizza una piccola quantità di memoria e di tempo del processore.
- Facilmente configurabile dagli amministratori di sistema che devono implementare una specifica soluzione di sicurezza in poco tempo.
Snort è in grado di eseguire l'acquisizione di pacchetti in tempo reale, l'analisi dei protocolli e la ricerca di contenuti.
Snort è stato progettato principalmente per analizzare i protocolli di rete TCP, UDP e ICMP,anche se può essere esteso con plugin per altri protocolli. Snort è in grado di rilevare una varietà attacchi e sonde, sulla base di una serie di regole configurate dall'amministratore del sistema.

17.11.1 Architettura SNORT

quattro componenti logici:
-decodificatore di pacchetti che elabora ogni pacchetto
-detection engine(tecniche di rilevamento): esegue il lavoro effettivo di rilevamento, analizza ogni pacchetto in base ad un insieme di regole definite dall'utente per la SNORT corrente,l'utente definisce delle caratteristiche e questo processo analizza ogni pacchetto per vedere se è conforme a quelle caratteristiche,se non è conforme lo scarta.
-logger: Il logger memorizza il pacchetto rilevato in formato leggibile dall'uomo(per ogni pacchetto che corrisponde ad una regola)
- Alerter: Per ogni pacchetto rilevato è possible inviare un avviso su file,database,socket,...

17.11.2 Regole

Snort utilizza un linguaggio di definizione delle regole semplice e flessibile che genera le regole utilizzate dal motore di rilevamento(detection engine).
Sebbene le regole siano semplici e facili da scrivere, sono abbastanza potenti da rilevare un'ampia varietà di traffico ostile o sospetto. Ogni regola è composta da un'intestazione fissa e da zero o più opzioni.
L'intestazione contiene i seguenti elementi:
Azione: L'azione della regola indica a Snort cosa fare quando trova un pacchetto che corrisponde ai criteri della regola. che corrisponde ai criteri della regola.
Protocollo: Snort procede nell'analisi se il protocollo corrisponde a questo campo(versione attuale riconosce TCP,UDP,ICMP).
Indrizzo IP sorgente: viene specificato un indirizzo ip specifico da cui deve provenire il pacchetto o un gruppo di IP o un gruppo di IP da cui non deve provenire,etc..
- Porta di origine: Questo campo indica la porta di origine del protocollo specificato (ad esempio, una porta TCP).
-Direzione: unidirezionale/bidirezionale.
Indirizzo IP di destinazione: Indica la destinazione del pacchetto.
- Porta di destinazione: Indica la porta di destinazione.

18 Buffer Overflow

Attacchi di tipo Overflow,uno dei comuni attacchi che deriva da una programmazione inaccurata delle applicazioni.
Questo capitolo si focalizza su come si verifica un buffer overflow e quali metodi possono essere utilizzati per individuarli e prevenirli.
Buffer overflow è una condizione che si verifica dopo un interazione in base alla quale è possibile inserire

nel buffer più dati in input rispetto alla capacità del buffer,sovrascrivendo altre informazioni. Gli aggressori sfruttano tale condizione per mandare in crash il programma o per inserire codice predisposto che consente di ottenere il controllo del sistema.
Un buffer overflow può verificarsi come conseguenza di un errore di programmazione quando un processo tenta di memorizzare dati oltre i limiti di un buffer di dimensioni fisse e sovrascrive le celle di memoria adiacenti. Se queste celle adiacenti contengono dati sensibili,si rischia di mandare in crash il programma o addirittura che l'attaccante prende il controllo del programma in quanto potrebbe andare il controllo al codice che inserisce l'attaccante.
Esempio: la funzione gets di C che prende in input dei caratteri,non fa un controllo su quanti dati l'utente inserisce,se l'utente inizializza due variabili con char size=8,in memoria vengono allocate adiacenti,se alla prima variabile l'utente inserisce più di 8 caratteri,gets non li blocca..i restanti caratteri andranno a sovrascrivere le celle adiacenti. Questo è un caso di buffer overflow in cui l'attaccante ha sovrascritto le celle di memoria adiacenti.
Per sfruttare qualsiasi tipo di buffer overflow,l'attaccante ha bisogno di

- Identificare una vulnerabilità di buffer overflow in qualche programma che può essere sfruttata utilizzando dati esterni sotto il controllo degli attaccanti
- Capire come quel buffer sarà memorizzato in memoria e quindi il modo in cui corrompere le locazioni di memoria adiacenti, in modo da alterare il flusso di esecuzione del programma.

Per scoprire le vulnerabilità si può tracciare l'esecuzione del programma mentre processa input sovradimensionati o tramite strumenti come fuzzing che identifica automaticamente programmi potenzialmente vulnerabili,quello che l'attaccante fa una volta corrotta la memoria dipende da quali valori vengono sovrascritti.
La responsabilità di evitare questi attacchi è a livello di linguaggio di programmazione,alcuni linguaggi come Python,Java non permettono di salvare più dati di quanti ne possa contenere una variabile,in questo modo si evitano attacchi di questo tipo,ma questa "sicurezza" che danno questi linguaggi ha un costo in termini di consumo di risorse e tempo di compilazione.
Linguaggi come C e i suoi derivati invece permettono di accedere a basso livello alla memoria,con un vantaggio in termini di tempo di esecuzione e consumo risorse,ma con il rischio di subire attacchi di questo tipo.

18.1 Stack buffer overflow

Questo evento si verifica quando il buffer che viene attaccato si trova nello stack di solito come una variabile locale,questi attacchi si chiamano STACK SMASHING.
Quando una funzione chiama un altra funzione,ha bisogno di salvarsi da qualche parte l'indirizzo di ritorno di quella funzione,di locazioni di memoria dove salvare i parametri da passare alla funzione chiamata,e i valori dei registri che vuole continuare ad usare quando la funzione chiamata termina. Tutti questi valori sono salvati in una struttura chiamata STACK FRAME. Ogni chiamata a funzione ha bisogno di una locazione di memoria dove salvare i suoi dati,anche questi dati locali sono memorizzati nello stack frame della funzione.
Esempio: una funzione P chiama una funzione Q:
La funzione P: salva indirizzo di ritorno,puntatore al vecchio frame e i parametri da passare a Q,tutto questo nello stack.
La funzione Q inserisce il puntatore al suo frmae,alloca spazio per le sue variabili locali ,esegue il suo codice,impost il puntatore dello stack al frame ,e esegue una return che torna al programma P. Ora la funzione P estrae i parametri di Q e continua l'esecuzione dalla riga successiva.
Come abbiamo visto,lo stack frame è una memoria temporanea nello stack che si crea quando una funzione viene chiamata, nello stack si crea un puntatore all'indirizzo di ritorno,sotto uno allo stack frame della funzione e sotto tutti i suoi dati locali, etc..
La possibilità di sovrascrivere puntatore a frame e indirizzo di ritorno costituisce il cuore di un attacco buffer overflow.
Quando un programma viene lanciato,il sistema operativo crea un processo in memoria principale,ogni processo ha una propria immagine in cui è presente: codice del programma,dati globali,stack,heap, gli stackframe sono posizionati uno dopo l'altro all'interno dello stack che cresce verso il basso,mentre l'heap verso l'alto.

Se l'attaccante inserisce un dato in input (letto tramite gets) troppo grande,il valore va a sovrascrivere il puntatoreal frame salvato e l'indirizzo di ritorno della funzione, causando un Segmentation Fault, in questo caso si è mandato in crash il programma.
Ma se l'attaccante volesse trasferire il controllo del programma ad una funzione scelta da lui?
Il modo più semplice per farlo è che l'input che causa l'overflow contenga l'indirizzo di destinazione proprio nel punto in cui sovrascriverà l'indirizzo di ritorno salvato nello stack frame.
Invece di tornare alla funzione chiamante salterà all'indirizzo fornito. Ad esempio se l'attaccante inserisce l'indirizzo di inizio della funzione stessa,la funzione verrà eseguita due volte,poi la seconda volta siccome il puntatore allo stack frame non è più valido andrà in crashma comunque la funzione è stata eseguita due volte come volev l'attaccante.

18.2 Altre vulnerabilità di stack overflow

Fin'ora abbiamo visto esempi di attacchi quando l'input veniva letto.
Esistono altri tipi di attacchi buffer overflow che si generano tutte le volte che i dati vengono copiati o mischiatiin un buffer dove almeno un pò di dati vengono letti dall'esterno del programma.
Se il programma non controlla che il buffer sia abbastanza capace o che i dati copiati termino correttamente,allora si può verificare un buffer overflow.
Può succedere che il programma leggi in modo sicuro dal'esterno,ma poi in un secondo momento il programma può trasferire in un'altra funzione il dato in maniera non sicura,causando buffer overflow.
Altra funzione,oltre gets,non sicura: sprintf.
fgets è sicura.
Se io leggo in modo sicuro i byte di una stringa inseriti in input,esempio alloco max 16 char per una stringa in input,l'utente me ne inserisce 30,io ne leggo con fgets solo 16,se poi con sprintf concateno la stringa in input con la parola "inserita:", e salvo tutto in "tmp" se supero la size di tmp vado in buffer overflow,sprintf non è sicura.
Può succedere che sovrascrive il puntatore al frame e non l'indirizzo di ritorno.
Per evitare questi problemi bisogna controllare tutti i possibili casi in cui lavoro con input presi dall'esterno.
Lista di funzioni non sicure

- gets
- sprintf
- strcat
- strcpy
- vsprintf

E' importante sapere che quando si verifca buffer overflow,indirizzo di ritorno della funzione o puntatore allo stack frame, vengono cancellati del tutto,quindi non c'è più possibilità di ripristinare il servizio,va in crash.
Poco importa all'attaccante che ha come obiettivo quello di sostituire il codice del programma con una shell,ma anche se l'attaccante non dovesse farlo,il programma in crash comporta fatti più o meno gravi,lato client un solo messaggio d'errore,lato server abbastanza grave.

18.3 Shellcode

L'obiettivo è eseguire una shell su attacco di tipo buffer overflow.
Si utilizza il meccanismo di buffer overflow,inserendo nel buffer un programma che va ad eseguire una shell.
Deve essere ad hoc per il sistema operativo che sta usando,e dipende anche dal processore.
Lo shellcode ha tre passaggi:
-scrittura codice C
-passaggio da C in assembly
-da assebly a codice esadecimale.
La funzione c execve sostituisce il codice del programma con quello di una shell, in windows invece c'è

system.
Lo shellcode deve essere eseguito in modo del tutto indipendente da dove è collocato in memoria,possono essere usati solo indirizzi relativi.
Non devono essere presenti valori NULL all'interno dello shellcode,può comparire solo alla fine di tutto il codice.
All'interno del codice non può starci NULL perchè il NULL è inteso come fine stringa,quindi deve stare solo alla fine(quelli che devono mettere NULL me li devo calcolare a runtime.

18.4 Esempio di attacco stack overflow

Il programma target ha il buffer di size 64 e il programma ha setuid root,cioè il programma viene eseguito con i permessi root.
L'attaccante deve analizzare il programma per determinare la posizione del buffer e quanti dati sono necessari per mandarlo in buffer overflow cioè per raggiungere il puntatore al frame precedente e l'indirizzo di ritorno del suo stack frame.
Per fare questo l'attaccante esegue il programma con un debugger,in questo modo è possibile vedere gli indirizzi dove si trovano nello stack questi valori e capire quanti byte sono necessari per sovrascrivere quello che serve all'attaccante.
L'attaccante deve scrivere nel programma destinazione anche i comandi che vuole far eseguire sulla shell,in quanto lo stdinput è lo stesso del programma che va a sostituire,i privilegi che prende la shell sono gli stessi privilegi con cui viene eseguto il programma.
In questo esempio l'attaccante sfrutta il fatto che il programma viene eseguito in modalità root e può leggere il file shadow per prendersi le password criptate per fare un pò di password craking.
Per riempire il buffer e posizionare il programma alla fine di esso,si usano una serie di NOP.
Lo shellcode può essere usato per diverse funzioni: eliminare regole del firewall,esecuzione di una shell remota a cui connettersi,change root,..
Un ulteriore obiettivo sono i programmi che forniscono un servizio di rete,utily di sistema,programmi che gestiscono formati di documenti(GIF o JPEG), l'input non proviene da terminale o connessione di rete ma dal file che viene decodificato e visualizzato.

18.5 Difese contro i buffer overflow

Due approcci per difendersi

- A tempo di compilazione: obiettivo di rendere un programma più sicuro,attraverso una serie di azioni:scrittura del programma e ricompilazione
- A tempo di esecuzione: cercano di fare detection su attacchi overflow,se viene identficato un attacco overflow,l'esecuzione del programma viene abortita

18.5.1 Tempo di compilazione

Hanno l'obiettivo di prevenire o rilevare i buffer overflow, irrobustendo i programmi quando questi vengono compilati.Ci sono diversi approcci.
-Scelta del linguaggio di programmazione:
Ci sono linguaggi di programmazione ad alto livello:python,java.
Ci sono controlli che fa il compilatore per evitare ciò.
Questi linguaggi sono meno performanti(troppi controlli), inoltre sono linguaggi ad alto livello che non permettono di accedere in alcune risorse più sensibili.
-Tecniche sicure di scrittura del codice:
Se usiamo C, i programmatori devono usare delle buone pratiche di sicurezza,controlli su dimensioni di buffer,andrebbe revisionato tutto il codice per sostituire codice non sicuro in codice sicuro.
Il programmatore non deve solo pensare ciò che ci si aspetta,ma essere consapevole di come le cose potrebbero andare storte,ogni volta che scrivo su un buffer devo controllare che ci sia spazio sufficiente.
-Estensioni del linguaggio e uso di librerie sicure:
Andare a controllare i range delle strutture dati una volta che i programmi sono stati compilati. Semplice su strutture dati statiche,più difficile su strutture dinamiche,la dimensione della memoria viene calcolata a tempo di esecuzione e non tempo di compilazione.Perdita di efficienza,tutti i programmi e

le librerie devono essere ricompilate con il compilatore modificato.
Ulteriore approccio: introdurre una libreria dinamica che sovrascrive il comportamento delle librerie standard,quando il programma viene eseguito,la libreria dinamica riscrive in modo sicuro le funzioni che non sono sicure.
Necessario modificare il sistema operativo.
-Protezione dello stack:
Inserire codici di entrata e di uscita della funzione che servono pe controllare eventuali cambiamenti,se vengono rilevati il programma viene interrotto.
Diversi approcci:
Il primo: stakguard:
Il codice di entrata aggiunto all'ingresso della funzione scrive un valore canary,sotto l'indirizzo allo stack frame,prima di allocare spazio per le variabili locali,il codice d'uscita controlla che il canary non sia stato modificato prima di eseguire le normali funzioni di uscita da una funzione.Qualsiasi attacco di buffer overflow dovrà cambiare il canary prima di cambiare lo stack frame,e l'attacco verrà rilevato e bloccato.
Il canary deve essere imprevedibile e diverso su vari sistemi(un numero random creato ogni volta che viene creato il processo del programma in esecuzione).
Problemi: tutti i programmi che utilizzano questa tecnica devono essere ricompilati, Per ogni funzione vengono allocati due ulteriori locazioni di memoria(entrata e uscita) che crea problemi con il debugger.
Secondo:Stackshield and return Address Defender(RAD):
Include entry and exit code,non lo memorizza all'interno dello stack,i programmi di debug rimangono compatibili,le memorizza in zone di memoria sicure(non accedute). Abbiamo il codice fuori dallo stack,che può essere usato per controllare se ci sono state delle modifiche allo stack.Anche qui, i programmi devono essere ricompilati per usare questa estensione(di norma sono estensioni di GCC).

18.5.2 Tempo di esecuzione

Come gia discusso,le maggiori tecniche di controllo a tempo di compilazione richiedono che i programmi devono essere ricompilati.
Per prevenire esecuzione di codice all'interno dello stack,marchiamo delle pagine di memoria in modo "non eseguibili",serve supporto hardware(dobbiamo etichettare le pagine di memoria virtuale come non eseguibili).
Evitiamo codice eseguibile nello stack,partendo dal fatto che codice eseguibile dovrebbe stare negli spazi degli indirizzi dei processi, Ci sono dei problemi: in alcuni casi i codici devono essere eseguiti all'interno dello stack.Seppur questa tecnica protegge da buffer overflow in quanto rende lo stack non eseguibile,ha delle limitazioni quando ci sono alcune funzioni/librerie che si eseguono nello stack(le stesse chiamate ricorsive di una funzione).
Ulteriore approccio: Randomizzare l'assegnazione di spazi di memoria,abbiamo sempre ipotizzato che siano memorizzati in modo adiacente.
Ad ogni esecuzione del programma,gli spazi di memoria possono essere allocati in modo casuale,compreso l'indirizzo dove si troverà lo stack in modo da rendere quasi impossibile per un attaccante prevedere dove si trova il buffer di destionazione,se lo faccio per ogni esecuzione del processo,l'attaccante non può calcolare di preciso la posizione dello stack e del puntatore allo stack frame.
Lo posso fare anche andando a posizionare le librerie standard in modo casuale nello stack o nella memoria virtuale(ci sono attacchi anche a librerie/heap/..),non solo attacci allo stack.
Altro approccio: inserire pagine di guardia all'interno della memoria(presupponendo che un processo abbia a disposizione molta più memoriavirtuale di quella di cui ne ha effettivamente bisogno.
Tra gli intervalli di indirizzi usati per ciascuno dei componenti dello spazio di indirizzamento, vengono posti degli spazi vuoti(pagine di guardia), qualsiasi tentativo di accesso provoca l'interruzione del processo,questo può prevenire attacchi di buffer overflow,si possono mettere pagine di guardia tra gli stack frame o tra le diverse allocazioni sull'heap pre prevenire attacchi di stack e heap overflow,a discapito di un overhead di tempo di esecuzone,poichè ci sono molte pagine della memoria che devono essre mappate.

19 Sicurezza dei Software

Molte vulenrabilità di sicurezza informatica derivano da pratiche di programmazione scorretta.
Gli errori vengono raggruppati in tre categorie:

- Interazione non sicura tra le componenti
  - Neutralizzazione impropria di caratteri speciali in un comando SQL(SQLi)
  - Neutralizzazione impropria di input durante la generazione di pagine Web(Cross-site Scripting)
- Gestione non oculata delle risorse
  - Copia del buffer senza controllare dimensione(buffer overflow)
  - Uso di funzioni potenzialmente pericolose
- Difese deboli
  - Autorizzazione mancante
  - Mancata cifratura per dati sensibili
  - Assegnazione dei privilegi in modo errato

La consapevolezza di questi problemi è fondamentae nella scrittura del codice del programma.
Le "falle" maggiori nei software sono

- input non validati
- cross site scripting
- buffer overflow
- gestione scorretta degli errori
- inejction flaw(difetti di iniezione)

Svariati approcci per ridurre il numero di vulnerabilità nel software

- Eliminare le vulnerabilità prima che si manifestino attraverso l'utilizzo di metodi avanzati per la realizzazione del software
- Scoprire le vulnerabilità attraverso testing
- Ridurre l'impatto delle vulnerabilità attraverso architetture più resilienti

La sicurezza è strettamente legata alla qualità e all'affidabilità del software,queste due si occupano dell'errore accidentale di un programma,esistono dei testing che mirano ad eliminare quanti più bug possibili, i testing generano input probabili e errori comuni.
La sicurezza del software invece si occupa di gestire input che si differenziano completamente dall'input che il programma si aspetta,per questo potrebbero non essere gestiti dai metodi di testing legati all'affidabilità e qualità.
Tutti gli aspetti legati allo scrivere codice in sicurezza vengono messi insieme in un unica definizione.
La PROGRAMMAZIONE DIFENSIVA O SICURA: è il processo di implementazione del software che garantisce il suo funzionamento anche quando è sotto attacco,la regola di base è di non dare nulla per scontato e di gestire ogni possibile stato di errore.
Un errore dei programmatori è quello di concentrarsi sui passi necessari per garantire il buon funzionamento e il normale flusso di esecuzione del programma,piuttosto che concentrarsi su ogni tipo di input che il programma può ricevere,fanno assunzioni sul tipo di input che il programma riceverà,la programmazione difensiva non vuole questo!.
La programmazione difensiva esige quindi un cambio di mentalità rispetto alle pratiche di programmazione tradizionali.

19.1 Gestione dell'input del programma

L'errata gestione dell'input è uno degli errori più comuni nell'ambito della sicurezza del software. Occorre identificare tutte le sorgenti di dati di input e tutti i presupposti sulla dimensione e il tipo di valore che assumono,le due principali aree critiche relative all'input sono dimensione e significato/interpretazione.
Nel leggere/copiare input da sorgenti,i programmatori fanno assunzioni sulla dimensione dell'input,non verifica se l'input superi una certa dimensione,se viene superata si verifica buffer overflow.
I testing dei programmi potrebbero non rilevare tale vulnerabilità perchè si concentrano su input classici che potrebber inserire un nomrale utente.Dobbiamo sapere che ogni input potrebber essere pericoloso,possiamo usare buffer dinamici ma anche qui dobbiamo stare attenti che non superano la memoria disponibile,queste verifiche devono essere fatti ogni qualvolta che usiamo un valore sconosciuto a noi.

19.1.1 Interazione dell'input del programma

Altro problema: che significato diamo all'input del programma?
Occorre prestare attenzione nell'identificare l'insieme di caratteri in uno e dunque wuali caratteri si stanno leggendo,determinate anche il significato,dovremmo verificare che i valori ineriti rappresentano effetivamente il tipo di dati previsto,altrimenti potrebbero sorgere delle vulnerabilità.

19.1.2 Injection Attacks

Il termine injection attacks si riferisce ad un'ampia varietà di falle di programma relative alla gestione non corretta dei dati di input,se il valore inserito dall'utente viene direttamente passato come parametro al server che può interrogare un database o una shell o altro,l'attaccante invece di inserire un normale parametro può inserire comand,questo è un tipico attacco di command injection. Per evitare ciò il programmatore deve verificare che i dati d'input siano conformi a tali assunzioni prima del loro utilizzo,si possono confrontrare con un pattern che descrive la loro natura e scartare quelli che non sono di quel pattern.
SQL injection è un altro tipo di attacco simile a command injection,con la differenza che in sqli si inserisce codice sql,in command injection si inseriscono metacaratteri di shell.
Terza variante di injcetion: code injection: l'input è un codice che viene eseguito dal sistema attaccato ad esempio in buffer overflow abbiamo del codice inserito,in quei casi il codice iniettato è linguaggio macchina rivolto ad uno specifico sistema informatico.
Dobbiamo anche considerare quelle variabili che vengono prese da altri file e non direttamente dall'esterno,queste variabili potrebbero essere vulnerabili perchè gli attaccanti potrennero bypassare alcune procedure e inserire manualmente alcune variabili che noi crediamodi ricevere da una normale esecuzione dell'applicazione,se questi input non sono opportunamente controllati rischiamo di subire attacchi.
Se io uso include path ."funzione.php"; per prendere il percorso assoluto della funzione php partendo dalla directory dove mi trovo,se la variabile path viene inserita in una richiesta http un attaccante può inserire un url che poi il programma esegue!. Questo attacco prende il nome di PHP code Injection, Infatti si può inserire in php anche url in modo da far provenire il codice da un altro punto della rete.Se l'attaccante inserisce path=codice malevolo che sta in rete Viene eseguito.
La variabile path deve essere opportunamente filtrata.
E' buona norma non usare variabili quando inserisco include o require, o se le inserisco bisogna controllare cosa contengono.
Ricapitolando: è necessario identificare tutte le sorgenti di input e validare qualunque asunzione fatta sull'input prima di utilizzare e di comprendere il significato e l'interpretazione dei valori forniti.

19.2 Attacchi cross-site scripting

Un'altra classe di vulnerabilità riguarda l'input fornito da un programma da un utente che poi viene trasmesso ad un altro utente.
Attacchi XSS(cross site scripting) perche si riconstrano principalmente in pagine web con script.
Questa vulnerabilità richiede di inserire codice script nel contenuto HTML di una pagina web visualizzata dal browser di un utente(il codice può essere javascript).
La variante più comune sono gli attacchi xss refelction,se un sito prevede che utenti inseriscano dati

tramite form,se questi non vengono opportunatamente filtrati,l'attaccante può inserisce link a codici javascript o codici javascript che,successivamente,quando altri utenti visitano il sito possono essere vittime di attacchi xss, in quanto viene eseguito il codice javascript malevolo.
XSS si basa sul fatto che tutti i contenuti del sito siano ugualmente fidati e dunque è lecito interagire con altri contenuti dello stesso sito,gli attaccanti sfruttano questo per ottenere privilegi elevati e far eseguire il codice javascript inserito da loro.
Per prevenire questo attacco occorre controllare tutti gli input forniti dall'utente e rimuovere codice pericoloso.
Gli attacchi XSS rilevano un errore nel gestire in modo corretto sia l'input che l'output del programma,in questo cado il target non è il programma ma sono tutti gli utenti successivi che visitano il programma.
Altri attacchi XSS: request forgery, http response splitting, anche qui il problema è l'utilizzo poco prudente di input non fidati e non verificati.

19.3 Validazione della sintassi dell'input

Dobbiamo controllare che i dati abbiano il formato che ci aspettiamo, o controlliamo che facciano match con il formato che ci aspettiamo oppure lo confrontiamo con dei valori pericolosi noti,nel primo caso si intende "allowlisting" nel secondo "denylisting", il problema è che si scoprono continuamente metodi per aggirare questi controlli,se si cerca di bloccare dati di input pericolosi noti è probabile che l'attaccante ne scopra altri,mentre se accettiamo solo dati buoni è più probabile che non veniamo attaccati.
Il confronto allowlisting si fa con espressioni regolari,i dati che non superano questo confronto possono essere scartati,inviando un messaggio d'errore,oppure possiamo correggere noi il dati eliminando i valori non buoni,questa procedura prende il nome di escape dei metacaratteri per rendere l'input sicuro.
La codifica UTF-8 ha una codifica a 16 bit che permette di rappresentare molti caratteri che la codifica a 8 bit di ASCII non permette.
Questo è necessario per la crescente esigenza di supportare gli utenti in tutto il mondo e per interagire con loro utilizzando le proprie lingue.
Ogni carattere dovrebbe avere una codifica univoca,tuttavia è possibile che dei comuni caratteri ASCII abbiamo codifiche multiple.
Un programma non case sensitive che non tiene conto neanche degli accenti potrebbere considerare a == A,à == A , A' == A, etc.., questi problemi mettono in evidenza i problemi legati alle codifiche multiple e al controllo dei valori dei dati pericolosi.
A fronte di possibilità di codifich multiple,i dati d'input devono essere trasformati in una rappresenazione unica e minima.Questo processo si chiama CANONIZZAZIONE e consiste nel sostituire le codifiche alternative equivalenti con un univo valore comune,in questo modo i dati in input possono essere confrontati con una singola rappresentazione di valori di input accettabili.
Ci sono librerie appositi anti-XSS per evitare controlli espliciti per ciascun campo.
Se i dati hanno valore numerico emergono ulteriori problemi.
I valori numerici vengono rappresentati con 8,16,32,64 bit e possono essere signed o unsigned,i problemi si verificano quando un valore di una certa dimensione viene convertito in un altro,per esempio è possibile leggere una dimensione di un buffer come un intero unsigned e successivamente confrontarlo con la dimensione massima accettabile dal buffer,alcuni linguaggi di programmazione è possibile che il valoe unsigned della dimensione del buffer venga letto come signed,se l'attaccante inserisce un valore molto grande che viene trattato come un numero negativo(con il bit superiore impostato),trattandosi di un numero negativo è evidente che quando confronto con la dimensione del buffer(positiva e piccola) il confronto viene soddisfatto,ovvero sto richiedendo memoria al buffer meno di quanta ne ha,ma quando vado ad usare questa memoria, il valore è unsigned e potrebbe essere più grande della memoria del buffer causando overflow,risulta essenziale prestare attenzione al controllo delle assunzioni fatte sui valori dei dati e che siano coerenti su ogni utilizzo di essi.

19.4 Input fuzzing

Si tratta di una tecnica di testing del software che per l'input da esplorare è molto ampio. Il fine è quello di determinare se il programma o la funzione gestiscono correttamente tutti questi tipi di input. Il vantaggio del fuzzing è dato dalla sua semplicità e dal non dipendere daassunzioni sull'input previsto

per ogni programma. Tali test contribuiscono a stabilire l'affidabilità e le falle nella sicurezza dei programmi.Si può generare in maniera casuale sulla base di un certo template, servono per analizzare possibili scenari di bug.
Lo scopo di utilizzare template è quello di aumentare la probabilità di rilevare bug,svantaggio: fanno assunzioni di input,non trova bug generati da altri input su cui non ho fatto assunzioni.
Per ottenere una copertura totale è buona norma usare le due tecniche combinate.
Limiti del metodo fuzzing: identifica soltanto dei tipi di errori semplici nella gestione dell'input,difficile che individui un bug generato univocamente da un ristretto numero di valori di input,ciononostante fuzzing rileva bug molto importanti,per questo deve essere usato come componente aggiuntiva,questi tool usati da molti sviluppatori per trovare bug nel sistema vengono usati anche dagli attaccanti per trovare bug nel software comunemente distribuito,per questo gli stessi sviluppatori devono utilizzare fuzzing e sfruttarlo al massimo per evitare che un attaccante scopri bug prima di lui.

19.4.1 Astrazione di un programma

Tre aspetti da considerare

- Gestione dell'input
- Processamento dei dati:algoritmi,interazione con SO
- Gestione dell'output

19.5 Scrivere codice di programma sicuro

L'algoritmo potrebbe non implementare in maniera corretta tutti i casi o le varianti del problema,ciò consentirebbe ad alcuni input apparentemente legittimi di causare comportamenti pericolosi all'interno del programma(quanto meno comportamento non previsto).
Importante anche non lasciare del codice aggiuntivo che è stato utilizzato per facilitare debug.
E' importante specificare tutte le assunzioni,un 'alternativa consiste nel ricorrere a metodi formali nello sviluppo e nell'analisi del software che garantisce che il software sia corretto per costruzione.

19.6 Accertarsi che il linguaggio macchina corrisponda all'algortimo

Altro problema: accertarsi che l'algortimo scritto in un linguaggio di programmazione sia lo stesso del linguaggio macchina,problema ampiamente ignorato da molti programmatori. Potrebbero starci dei programmatori di compilatori malevoli che fanno in modo che quando il programma viene compilato,venga aggiunto del codice aggiuntivo visibile solo a linguaggio macchina,controllare che l'algortimo sia lo stesso in progetti enormi è quasi impossibile,l'unica area in cui è richiesto un simile controllo è nello sviluppo di sistemi informatici affidabili con un livello molto alto,richiede la convalida della corrispondenza tra progetto,codice sorgente e codice oggetto.

19.7 Interpretazione corretta dei valori dei dati

Altro problema:corretta interpretazione dei dati,,esistono linguaggi come il C che permettono ai dati di inserire non solo numeri ma anche puntatori a locazioni di memoria,liste indicizzate,etc..,questo comporta possibili errori e un attaccante potrebbe approfittarne,la miglior difesa è usare un linguaggio di programmazioen fortemente tipizzato(non il C),anche linguaggi non tipizzato potrebbero usare librerie standard scritte in C che potrebbero avere alcuni bug,l'unico rimendio è di rimanere aggiornati su segnalazioni riguardo librerie che hanno dei bug riscontrati ed evitarle. Se si usa il C occorre fare attenzione.

19.8 Utilizzo corretto della memoria

Utilizzare in modo corretto la memoria allocata dinamicamente,bisogna allocarla e deallocarla continuamente quando non serve,altrimenti si riempie nel brevo tempo.Questo fenomeno prende il nome di MEMORY LEAK e spesso il programma va in crash una volta che l'heap è esaurito,ottimo modo per un attaccante di implementare un DoS. E' difficile capire quando la memoria non è più necessaria,può

capitare che si verifichi memory leak,ci sono librerie che fanno questi controlli,alcuni linguaggi invece, a differenza del C che non fa controlli di alloco/dealloco, come il Java o C++, gestiscono l'allocazione e il rilascio in modo automatico,anche se questo richiede un overhead molto alto,questi programmi risultano molto affidabili , il ricorso a tali linguaggi è fortemente consigliato per evitare problemi di gestione della memoria.

19.9 Race condition

Pevenire la race condition su memoria condivisa
Senza un'opportuna sincronizzazione tra processi/thread che accedono alla memoria condivisa si rischiano problemi di race condition,alcuni valori potrebbero essere compromessi.
Race condition: si verifica quando più processi competono per ottenere l'accesso incontrollato a determinate risorse,non è semplice,si può rischiare il deadlock,non esiste un metodo semplice di recupero dal deadlock se non interrompendo uno o più processi.Un attaccante potrebbe provocare un deadlock per ottenre un DoS,occorre fare attenzione e separare tutte le aree che richiedono accesso a zone di memoria condivise e individuare le primitive migliori da utilizzare.

19.10 Interagire con il sistema operativo e altri programmi

Ogni programma viene eseguito sotto il controllo del sistema operativo,ad eccezione dei sistemi embedded i programmi non vengono eseguiti in modo isolato,ma al contrario è il sistema operativo che li controlla,il processo incorpora alcune informazioni del sistema operativo come le variabili d'ambiente che possono essere utilizzate per personalizzare il funzionamento del programma,questi dati dovrebbero essere considerati esterni e dovrebbero essere validati.
Nei sistemi è prevista la presenza di utenti multipli, ognuno con le sue risorse ed i permessi,i programmi che accedono alle risorse potrebbero andare in errore se non ci sono i giusti permessi,d'altra parte può essere pericoloso dare troppo permessi in quanto un eventuale bug può compromettere le risorse e/o il sistema.

19.10.1 Variabili d'ambiente

Sono un insieme di valori stringa che ogni processo eredita dal genitore e possono influenzare il modo in cui un processo si comporta,il sistema operativo le inserisce nella memoria del processo alla sua creazione,corrispondono ad una copia delle variabili del genitore. Un programma può modificare le sue variabili d'ambiente in un qualsiasi momento e queste verrano copiate ai loro figli.Alcune variabili note: PATH che specifica l'insieme delle directory dove cercare un comando,IFS che specifica il delimitatore nella shell. Le variabili d'ambiente possono essere sfruttate per effettuare attacchi contro un programma.Anche le variabili d'ambiente sono degli input esterni che devono essere validati,l'obiettivo è compromettere un programma nel quale vengono concessi i privilegi di superutente o amministratore per forzarlo ad eseguire il codice scelto dall'attaccante,se un programma viene eseguito con i privilegi del proprietario del file e non dell'utente che lo esegue,se un attaccante modifica le variabili d'ambiente e poi lancia un programma che interagisce con le variabili d'ambiente si possono generare attacchi.L'attaccante può anche modificare IFS,questi tipi di attacco sono molto difficili da prevenire se eseguiti su uno script shell,alcuni sistemi UNIX bloccano l'impostazione di alcune variabili d'ambiente critiche per programmi che vengono eseguiti come root,ma non previene attacchi che vengono eseguiti con privilegi di altri utenti(proprietari del file). E' riconosciuto che scrivere script shell sicuri è molto difficile e il loro utilizzo è fortemente sconsigliato,bisognerebbe resettare tutte le variabili critiche.
Se è necessario usare uno script shell, è utile ricorrere ad un programma wrapper compilato per richiamare lo script,questo genera un insieme di variabili d'ambiente sufficientemente sicuro prima di chiamare lo script desiderato.Se usato correttamente offre un grado di sicurezza abbastanza alto,fa molti controlli prima di eseguire lo script.
Quando compilo un programma,avviene il cosidetto linking,tutte le librerie standard che includono vengono importate nel programma,questo se il linking è statico,causa un notevole spreco di spazio visto che la funzione è quella e rimane sempre quella,esiste il linking dinamico,un processo che usa linking dinamico non incorpora le funzioni ma ha una tabella con nomi e puntatori alle funzioni che usa,una singola copia di ciascuna libreria che fa riferimento ad una entry della tabella viene copiata e viene usata da tutti i processi che ne fanno richiesta(questo è il vantaggio,nel linking statico,ogni

processo incorpora tutte le funzioni che richiede), nel linking dinamico invece vengono incorporate una volta sola e una tabella di ogni processo punta alle librerie che deve usare,per diverse ragioni molti processi necessitano di versioni differenti della stessa libreria con lo stesso nome,si introduce un possibile attacco,l'attaccante inserisce una versione personalizzata di una libreria comune. Per cercare all'interno della directory le librerie in modo dinamico si usa la variabile d'ambiente LD-LIBRARY-PATH, se un attaccante modifica la variabile facendo in modo che il suo programma risulti per primo, quando il programma target viene eseguito e chiama la funzione il codice dell'attaccante viene eseguito con i privilegi di tale programma.
Anche qui i sistemi possono bloccare l'utilizzo di questa variabile quando il programma viene eseguito con privilegi diversi.Alternativamente si può utilizzare un eseguibile collegato staticamente,ma con overhead maggiore di memoria
Molti programmi utilizzano variabili personalizzati per permettere agli utenti di modificare il comportamento del programma semplicemente cambiando alcuni parametri, anche qui dobbiamo controllare come vengono modificati questi parametri,se si accorpano i valori di tali variabili con altre informazioni all'interno del buffer si genera buffer overflow,tutto questo ci fa capire che bisogna prestare attenzione al modo in cui un programma interagisce con il sistema in cui viene eseguito.

19.11 Uso dei minimi privilegi

L'attaccante molte volte diviene in grado di eseguire codice con privilegi e diritti di accesso del programma compromeso.
Se i privilegi che ottiene sono più elevati,ne consegue una PRIVILEGE ESCALATION.
Ne consegue che i programmi dovrebbero essere eseguiti con il minor numero di privilegi necessari per portare a termine la loro funzione. Questo approccio prende il nome di privilegio minimo.
Se un programma ha necessità di privilegi privilegiati rispetto a quelli che ha un utente che lo lancia,occorre prestare attenzione a definire appropiati privilegi di utente e di gruppo necessari.Un programma di questo tipo diventa un programma target per un attaccante che voglia acquisire privilegi aggiuntivi,una scelta:concedere privilegi aggiunti di utente o solo di gruppo?.
Dove possibile,la seconda risulta la preferibile.
Altro aspetto:garanzia che un programma privilegiato possa modificare solo i file e le directory strettamente necessarie.
Occorre assegnare con attenzione le corrette proprietà di file e gruppi ai file e directory gestiti dai programmi privilegiati.
Il problema sorge quando un programma privilegiato viene eseguito come root,questo diventa un programma target per un attaccante,il principio del minimo privilegio suggerisce di concedere tale accesso il più raramente possibile e per il minor lasso di tempo possibile.
Sappiamo ora che una buona progettazione di programmi difensivi richiede che i programmi grandi e complessi vengano suddivisi in moduli più piccoli e che a ciascuno di essi vengano concessi i privilegi strettamente necessari e solamente per il lasso di tempo per il quale ne ha bisogno.Essendo più piccoli risultano più semplici da testare.
Un'ulteriore tecnica consiste nell'eseguire i programmi potenzialmente vulnerabili in una qualche sandobx che fornisce un maggior isolamento e controllo del programma in esecuzione,i sistemi UNIX hanno chroot jail che se configurata(difficile cofigurarla) anche se il programma viene compromesso può accedere solo ai file nella porzione chroot jail del file system,se configurata male potrebbe creare problemi,comunque non è una soluzione completa, meglio la virtualizzazioen applicativa.

19.12 Chiamate di sistema e funzioni di libreria standard

Ad eccezione dei sistemi embedded, nessun programma informatico contiene tutto il codice necessario per essere eseguito,molte volte si tratta di fare delle chiamate al sistema operativo e/o alle funzioni di libreria standard,quando vengono chiamate,i programmatori fanno delle assunzioni sul loro effettivo funzionamento,in molti casi è giusto,ma in altri no.
Quando un programma è in esecuzione, il sistema operativo lo controlla insieme ad altri n processi, e deve gestire l'accesso alle risorse in modo equo,di conseguenza le risorse devono essere bufferizzate,può succedere che il sistema operativo per sue ottimizzazioni modifichi l'ordine delle richieste,etc...Questo può portare ad un conflitto con gli obiettivi del programma,a meno che il programmatore non l'abbia tenuto conto.

19.13 Prevenire la race condition sulle risorse di sistema condivise

Il sistema operativo fornisce diversi metodi per serializzare gli accessi alle risorse condivise.
Prima tecnica:lock sui file,ha dei problemi,se un processo decide di ignorare il lock lo può benissimo fare e il sistema non lo impone,tutti i programmi devono cooperare per fare funzionare il lock,secondo problema: due processi potrebbero creare contemporaneamente il lock su una stessa risorsa e avere race condition.
Seconda tecnica: operazione atomica

19.14 Utilizzo sicuro dei file temporanei

Molti programmi devono memorizzare una copia temporanea dei dati mentre li elaborano. A questo servono i file temporanei.
Il sistema operativo mette a disposizione delle posizioni ben note per la collocazione dei file temporanei.Problema: se un attaccante cerca di indovinare il nome del file(magari perche l'utente lo chiama con l'id del processo che è univoco) temporaneo che verrà utilizzato da un programma privilegiato per poi creare il file nel lasso di tmepo in cui il programma verifica che non esiste e poi lo crea,abbiamo anche qui una RACE CONDITION.
Per creare e utilizzare i file temporanei in modo sicuro è preferibile far uso di un nome casuale,e la creazione dovrebbe avvenire tramite un operazione atomica,la funzione di C mkstemp è ideale per questo scopo,mentre tmpfile e tempam risultano non sicure se usate senza attenzione inoltre solo chi crea il file dovrebbe potervi accedere e solo il proprietario dovrebbe avere il permesso di rimuoverlo.

19.15 Interazione con altri programmi

I programmi possono anche usufruire di funzioni di altri programmi,anche qui,se non si presta la dovuta attenzione è possibile che si creino vulnerabilità di sicurezza,un programma nuovo non può assumere che un programma vecchio che lui stesso sa usando in una parte sia stato implementato tenendo conto di tutte le problematiche della sicurezza.Altro problema:integrità e confidenzialità dei dati,se queste informazioni passano tra due o piu programmi sullo stesso sistema informatico allora bisogna usare pipe o file temporanei,se si trovano su sistemi diversi allora bisogna accertarsi che vengano usati protocolli sicuri e che facciano uso di crittografia forte.Quando un programma invoca un programma figlio dovrebbe accertarsi che termini correttamente.

19.16 Gestione dell'output del programma

Anche l'output può essere classificato,come l'input in binario o testuale. I binari possono codificare strutture complesse, quelli testuali invece possono rappresentare un certo insieme di caratteri.
In ogni caso è fondamentale che l'output sia effettivamente conforme alla struttura e all'interpretazione prevista.Se l'output contiene un risultato inaspettato questo potrebbe essere dannosso per gli utenti,esempio attacchi XSS,il programma deve sempre controllare l'output che sia conforme a quello che ci si aspetta.
Inoltre,alle fonti non fidate non deve essere permesso mostrare l'ouput agli utenti,come per il filtraggio dell'input,l'obiettivo è consentire esclusivamente ciò che è sicuro piuttosto che cercare di rimuovere ciò che è dannoso, visto che l'interpretazione di cio che è pericoloso può cambiare nel tempo. Altro problema è che diversi insiemi di caratteri ammettono diverse codifiche dei metacaratteri che possono variare l'interpretazione di ciò che viene considerato output valido,se il programma non conosce la codifica utilizzata potrebbe formulare assunzioni diverse.Nelle falle di sicurezza derivanti dall'output del programma,l'obiettivo della compromissione non era il programma che generava l'output ma il programma utilizzato per visualizzarlo.La codifica deve essere conforme alle aspettative di visualizzazione.

20 Symmetric Encryption

Viene chiamata anche: Crittografia convenzionale,Crittografia a chiave segreta o chiave singola.
Componenti

- Plaintext: Messaggio (originale,da inviare) inserito nell'algortimo
- Encryption: Algoritmo di crittografia che esegue varie sostituzioni e trasformazioni.

- Secret key: Chiave segreta che determina l'algoritmo di criptazione
- Ciphertext: Il messaggio criptato prodotto in uscita,dipende dalla chiave e dall'algoritmo
- Decryption algorithm: L'algoritmo di cifratura eseguito al contrario

20.1 Cryptography

I sistemi criptografici sono generalmente classificati in tre dimensioni indipendenti

- Tipo di operazioni utilizzate per trasformare il testo in chiaro in testo cifrato.
  Tutti gli algoritmi di cifratura si basano sue due principi generali: sostituzione: ciascun elemento del testo in chiaro viene mappato in un altro elemento e la trasposizione in cui gli elementi del testo in chiaro vengono riorganizzati,la maggior parte dei sistemi prevede che ci siano più fasi di sostituzioni e trasposizioni
- Numero di chiavi utilizzate,se mittente e destinatario utilizzano la stessa chiave, il sistema viene definito simmetrico,chiave singola,chiave segreta,convenzionale. Se il mittente e destinatario utilizzano due chiavi diverse il sisgtema viene definito asimmetrico,a due chiavi,crittografia a chiave pubblica.
- Modo in cui viene elaborato il testo in chiaro:
  - Cifrario a blocchi: elabora input un blocco alla volta producendo un blocco di uscita per ogni blocco d'ingresso
  - Cifrario a flusso: elabora gli elementi in ingresso in maniera continua producendo un elemento alla vola man mano che procede

20.2 Criptoanalisi

Processo che ha come obiettivo quello di scoprire il testo in chiaro o la chiave.
Non c'è una procedura standard,dipende dalla natura dello schema di crittografia e dalle informazioni di cui si dispone.
Il problema più difficile è quando si dispone solo del testo cifrato,in generale si suppone che l'attaccante conosca l'algoritmo di cifratura chie viene usato.
Un possibile attacco in queste condizioni è l'approccio a "brute force" che consiste nel trovare tutte le chiavi possibili,se lo spazio delle chiavi è molto grande questo diventa impossibile. In questo caso l'attaccante deve effettuare dei testi statistici, deve avere un idea generale del tipo di testo in chiaro.
L'attacco di solo testo cifrato è il più semplice da difendere perchè l'avversario ha la minor quantità di informazioni possibili, in generale però non è cosi,l'attaccante può avere più messaggi criptati,può avere informazioni su alcune parti in chiaro come ad sempio il format di un file che conosce ha sempre lo stesso pattern,può trovare una coppia cifratura,decifratura.Da queste informazioni diventa più facile dedurre la chiave.

20.3 Tipo di attacco noto alla criptoanalisi

- Ciphertext only: l'attaccante conosce solo l'algortimo di cifratura e il testo cifrato
- Known plaintext: l'attaccante conosoce: algoritmo di cifratura,testo cifrato da decifrare,uno o più coppie testo cifrato,decodifica
- Chosen plaintext: l'attaccante conosce: algoritmo di crittografia,testo cifrato insieme al suo decifrato.
- Chosen ciphertext: l'attaccante conosce l'algoritmo di crittografia,testo cifrato da decodificare , il presunto testo cifrato insieme al suo testo in chiaro scelto dall'hacker.
- Chosen text: l'attaccante conosce l'algortimo di cifratura,testo cifrato da decodificare,messaggio in chiaro scelto dall'hacker insieme al testo cifrato,presunto testo in chiaro insieme al suo testo cifrato.

Le copie cifrato/decifrato servono per facilitare la ricerca della chiave.
Uno schema di crittografia è computazionalmente sicuro se il testo cifrato generato dallo schema soddisfa uno o entrambi dei seguenti criteri:
- Il costo della rottura del cifrario è superiore al valore delle informazioni criptate. - Il tempo necessario per decifrare il cifrario supera la durata utile delle informazioni.
Sfortunatamente, è molto difficile stimare l'impegno necessario per analizzare con successo un testo cifrato. Tuttavia, supponendo che non vi siano debolezze matematiche nell'algoritmo, allora è indicato un approccio a forza bruta, e in questo caso possiamo fare delle stime ragionevoli su costi e tempi. Un approccio a forza bruta prevede di provare tutte le chiavi possibili fino a quando non si ottiene una traduzione intelligente del testo cifrato in quello in chiaro. In media, la metà di tutte le chiavi possibili per ottenere un successo.

20.4 Feistel Cipher Structure

Algoritmo di crittografia simmestrica a blocchi.
In ingresso all'algoritmo di crittografia ci sta un blocco di testo in chiaro di lunghezza 2w bit e una chiave K, il blocco di testo viene diviso in due metà(L0 e R09 e le due metà passano attraverso n cicli chiamati ROUND che poi si combinano per produrre il blocco di testo cifrato.
Ogni round i ha come input Li-1 e Ri-1 derivati dal round precedente e una sottochiave Ki derivata da K, ogni Ki è diversa in generale,vengono generate attraverso un algortimo di generazioni di sottochiavi.Tutti i round hanno la stessa struttura.
Durante ogni round viene applicata una funzione F alla metà destra dei dati (R), e l'output di questa funzione viene combinato con la metà sinistra dei dati (L) utilizzando l'operatore XOR.
La funzione di round (F) prende in input la metà destra dei dati (R) e una sottochiave del round (Ki), che è derivata dalla chiave principale dell'algoritmo. La funzione di round può variare a seconda dell'implementazione specifica dell'algoritmo Feistel.
Dopo aver applicato la funzione di round, l'output viene combinato con un'operazione di XOR con la metà sinistra dei dati (L). L'operazione XOR combina i bit dell'output con i bit della metà sinistra dei dati, bit a bit. Questo passaggio introduce la confusione e diffusione nei dati.
Successivamente, viene eseguita una permutazione delle due metà dei dati (R e L) scambiandole di posizione. In altre parole, la metà destra (R) diventa la nuova metà sinistra, mentre la metà sinistra (L) diventa la nuova metà destra. Questo scambio di posizione è importante per garantire che l'algoritmo Feistel sia reversibile e che la decifrazione sia possibile.
In sintesi, durante ogni round dell'algoritmo Feistel, si applica una funzione di round alla metà destra dei dati, si esegue un'operazione XOR tra l'output della funzione di round e la metà sinistra dei dati, e infine si scambiano le due metà dei dati. Questi passaggi vengono ripetuti per ogni round dell'algoritmo per ottenere la cifratura finale dei dati.
In generale, un cifrario a blocchi simmetrico è composto da una sequenza di round, in cui ogni round esegue sostituzioni e permutazioni condizionate dal valore della chiave segreta. L'esatta realizzazione di un cifrario a blocchi simmetrico dipende dalla scelta dei seguenti parametri e caratteristiche di progettazione:
- Dimensione del blocco: Blocchi più grandi significano una maggiore sicurezza (a parità di altre condizioni),ma velocità di cifratura/decifratura ridotta. Una dimensione di blocco di 128 bit è un compromesso ragionevole ed è quasi universale tra i recenti progetti di cifrari a blocchi recenti.
- Dimensione della chiave: Una dimensione maggiore della chiave significa maggiore sicurezza, ma può ridurre la velocità di crittografia/decrittografia. La lunghezza della chiave più comune negli algoritmi moderni è di 128 bit.
- Numero di round: L'essenza di un cifrario a blocchi simmetrico è che un singolo round offre una sicurezza inadeguata, ma che più round offrono una sicurezza crescente. sicurezza. La dimensione tipica è di 16 round.
- Algoritmo di generazione delle sottochiavi:La maggiore complessità di questo algoritmo dovrebbe portare a una maggiore difficoltà di crittoanalisi.
- Funzione di round: Anche in questo caso, una maggiore complessità significa generalmente una maggiore resistenza alla crittoanalisi.

Gli algoritmi di crittografia simmetrica più utilizzati sono i cifrari a blocchi. Un cifrario a blocchi elabora il testo in chiaro in blocchi di dimensioni fisse e produce un blocco di testo cifrato di dimen-

sioni uguali per ogni blocco di testo in chiaro.

Tre dei più importanti cifrari a blocchi simmetrici.

20.5 Data Encryption Standard(DES)

- Processo di crittografia
- Il testo in chiaro ha una lunghezza di 64 bit e la chiave di 56 bit (cifrario a blocchi). - La struttura del DES è una variazione minore della rete di Feistel; 16 round di elaborazione - Dalla chiave originale di 56 bit, vengono generate 16 sottochiavi, una delle quali viene utilizzata per ciascun round.
- Processo di decodifica
- Essenzialmente uguale al processo di cifratura. - Si utilizza il testo cifrato come input dell'algoritmo DES, ma si utilizzano le sottochiavi Ki in ordine inverso. Vale a dire, utilizzare K16 alla prima iterazione, K 15 alla seconda iterazione e così via.

20.6 3DES

Utilizza tre chiavi e tre esecuioni dell'algoritmo DES.
Il 3DES (Triple Data Encryption Standard) utilizza tre chiavi e tre esecuzioni dell'algoritmo DES. La sequenza di funzionamento segue un ordine di cifratura-decifratura-cifratura (EDE):
Quando si scoprì che la chiave a 56 bit del DES non era abbastanza lunga per garantire sicurezza contro brute force, venne introdotto il 3DES.
I blocchi hanno sempre dimensione 64,la chiave 168 cioè 56x3.
Viene eseguito il primo des con la chiave K1 dei primi 56 bit e il testo in chiaro,si esegue DES e l'output va al secondo step DES con la seconda chiave K2 di 56 bit.Il terzo step DES prende in input l'output del secondo + gli ultimi 56 bit della chiave e fornisce il messaggio cifrato. C = E(K3, D(K2, E(K1, p)))
dove:
C = testo cifrato
P = testo in chiaro
E[K, X] = cifratura di X utilizzando la chiave K
D[K, Y] = decifratura di Y utilizzando la chiave K
La decifratura è semplicemente l'operazione inversa con le chiavi invertite
P = D(K1, E(K2, D(K3, C)))
Non c'è alcuna rilevanza crittografica nell'uso della decifratura per la seconda fase della cifratura 3DES. Il suo unico vantaggio è quello di consentire agli utenti di 3DES di decifrare dati cifrati da utenti del vecchio DES singolo:
C = E(K1, D(K1, E(K1, P))) = E[K, P]
Con tre chiavi distinte, 3DES ha una lunghezza effettiva della chiave di 168 bit. Lo standard FIPS 46-3 consente anche l'uso di due chiavi, con K1 = K3; ciò fornisce una lunghezza della chiave di 112 bit. Lo standard FIPS 46-3 include le seguenti linee guida per 3DES:
3DES è l'algoritmo di cifratura simmetrica approvato FIPS di scelta.
Il DES originale, che utilizza una singola chiave di 56 bit, è consentito solo per sistemi legacy. I nuovi acquisti dovrebbero supportare 3DES.
Si incoraggiano le organizzazioni governative con sistemi DES legacy a fare la transizione a 3DES.
Si prevede che 3DES e l'Advanced Encryption Standard (AES) coesistano come algoritmi approvati FIPS, consentendo una transizione graduale ad AES.
È facile capire che 3DES è un algoritmo formidabile. Poiché l'algoritmo crittografico sottostante è DEA (Data Encryption Algorithm), 3DES può vantare la stessa resistenza alla crittoanalisi dell'algoritmo DEA. Inoltre, con una lunghezza della chiave di 168 bit, gli attacchi di forza bruta sono praticamente impossibili.

20.7 AES:Advanced Encryption Standard

Alla fine, AES è destinato a sostituire 3DES, ma questo processo richiederà alcuni anni. Il NIST prevede che 3DES rimarrà un algoritmo approvato (per l'uso del governo degli Stati Uniti) nel futuro

prevedibile.
AES utilizza una lunghezza di blocco di 128 bit e una lunghezza di chiave che può essere di 128, 192 o 256 bit. Nella descrizione di questa sezione, assumiamo una lunghezza di chiave di 128 bit, che probabilmente è quella più comunemente implementata.
L'ingresso agli algoritmi di crittografia e decriptazione è un singolo blocco di 128 bit.
AES lavora con blocchi di dati che vengono rappresentati come matrici quadrate di byte. Un blocco di dati standard per AES è di 128 bit, quindi la matrice sarà di dimensione 4x4 (4 righe e 4 colonne).

Durante il processo di crittografia o decrittografia, il blocco di dati viene copiato nella "matrice di stato". Questa matrice di stato viene modificata durante ogni fase dell'algoritmo.

Alla fine della fase finale di crittografia o decrittografia, il blocco di dati viene copiato in una "matrice di output". Quindi, il risultato finale sarà rappresentato come una matrice.

La chiave utilizzata nell'algoritmo AES è di 128 bit, quindi anch'essa viene rappresentata come una matrice quadra di byte. La chiave viene poi espansa in una "matrice di parole". Ogni parola è di 4 byte e l'espansione completa della chiave è di 44 parole per una chiave a 128 bit.

Nelle matrici (sia per l'input che per la chiave), i byte vengono ordinati per colonna. Ciò significa che i byte sono disposti uno sotto l'altro all'interno della colonna.

Per quanto riguarda i commenti specifici:

AES non utilizza una struttura di cifratura a blocchi chiamata "struttura Feistel". La struttura Feistel coinvolge l'uso di metà del blocco di dati per modificare l'altra metà, scambiando le due metà. Invece, AES elabora l'intero blocco di dati contemporaneamente durante ogni round, utilizzando sostituzioni e permutazioni.

La chiave di input viene espansa in un array di parole. Durante ogni round dell'algoritmo AES, quattro parole distinte (di 128 bit ciascuna) fungono da "chiavi di round".

L'algoritmo AES utilizza quattro diverse fasi: una fase di permutazione (ShiftRows), seguita da tre fasi di sostituzione (SubBytes, MixColumns, AddRoundKey). Queste fasi vengono ripetute per un numero specifico di round, a seconda della lunghezza della chiave utilizzata (10 round per una chiave a 128 bit).

- Substitute Bytes: utilizza una tabella S-box per sostituire byte per byte del blocco
- Shift rows: Una semplice permutazione che viene eseguita riga per riga
- Mix columns: Una sostituzione che altera ogni byte in una colonna
- Add round key: un semplice XOR bit a bit del blocco corrente con una parte della chiave espansa.

4. La struttura è abbastanza semplice. Sia per crittografia che per la decrittografia, l'algoritmo inizia con una fase di Add round key seguita da nove round ciascuno comprende tutte e 4 le fasi,seguite da un decimo round di tre fasi(SubBytes,ShiftRows,addRoundKey).
5. Solo la fase Add Round Key utilizza la chiave, per questo motivo la cifratura inizia e finisce sempre con una fase Add round key.
Qualsiasi altra fase, applicata all'inizio o alla fine, è reversibile senza conoscere la chiave e quindi non aggiungerebbe alcuna sicurezza.
6. La fase Add Round Key da sola non sarebbe affidabile. Gli altri tre fasi insieme rimescolano i bit, ma da sole non fornirebbero alcuna sicurezza perché non usano la chiave.
Possiamo vedere la cifratura come operazione alternata zioni di crittografia XOR (Add Round Key) di un blocco, seguite da scram- bling del blocco (le altre tre fasi).
7. Ogni fase è facilmente reversibile, per substitute byte, shift row, mx columns nell'algoritmo di decriptazione viene usata una funzione inversa. Nella fase add round key l'inverso si ottiene effettuando una xor nello stesso round.
8. Come nella maggior parte dei cifrari a blocchi, l'algoritmo di decriptazione fa uso di una chiave espansa in ordine inverso,l'algoritmo di decriptazione non è lo stesso di quello di criptazione,conseguenza della struttura di AES.
Durante la decriptazione AES, vengono eseguite le seguenti fasi, in ordine inverso rispetto all'algoritmo di criptazione.
9. Una volta stabilito che tutti e quattro gli stadi sono reversibili, è facile verificarlo che la decrittazione recupera il testo in chiaro.
10. Il round finale di crittografia e decrittografia è composto solo da tre fasi. Ancora una volta, questa è una conseguenza della particolare struttura di AES ed è necessario per rendere reversibile la cifratura.

20.8 Dettagli dell'algoritmo

Vediamo nel dettaglio l'algoritmo AES.
Trasformazione dei byte sostitutivi:chiamata anche SubBytes è una semplice ricerca di tabella.
AES definisce una matrice 16x16 valori di byte,chiamata S-box che contiene una permutazione di tutti i possibili 256 valori.
Mapping: i 4 bit più a sinistra vengono utilizzati come valore di riga,i 4 bit più a destra come valore di colonna. Questi valori fungono da indice nella S-box per selezionare un unico valore di uscita del bit.
Esempio: se ho 95 da mappare allora prendo riga 9 colonna 5 e vedo dentro la S-box che valore ho.(2A)
InvSubBytes: è la trasformazione inversa fa uso della S-box inversa, in cui leggendo un valore ricavo il suo output.
Nell'esempio di prima vado a leggere la riga 2 e colonna A: leggerò 95 nella matricr S-box inversa.
La S-box è progettata per resistere agli attacchi crittoanalitici noti.
Nello specifico, gli sviluppatori di AES hanno cercato un design con una bassa correlazione tra bit di ingresso e bit di uscita e la proprietà che l'uscita non può essere descritta come a semplice funzione matematica dell'input.
Trasformazione shift row: la prima riga di stato non cambia.Nella seconda riga viene eseguito uno spostamento circolare a sinistra di 1 byte. Per la terza riga, viene eseguito uno spostamento circolare a sinistra di 2 byte. Per la terza riga, viene eseguito uno spostamento circolare a sinistra di 3 byte.
La trasformazione shift row inverse(invShiftRows): fa la stessa cosa ma in direzione opposta.
Mix Column Transformation: opera per ciascuna colonna individualmente. Ogni byte di una colonna viene mappato in un nuovo valore che è una funzione di tutti e 4 i byte nella colonna,la mappatura fa uso di equazioni su campi finiti, è progettato per fornire una buona miscelazione tra i byte di ciascuna colonna.
La trasformazione della colonna mix combinata con la trasformazione della riga di spostamento garantisce che dopo pochi round, tutti i bit di output dipendono da tutti i bit d'input.
Add Round Key Transformation: i 128 bit di stato vengono sottoposti ad uno xor bit a bit con la chiave di quel round(anch'essa una matrice).
L'inverso è identico perche l'operazione XOR è la stessa del suo inverso. La complessità di questa fase unita alle altre tre fasi garantiscono all' AES sicurezza.

20.9 AES key expansion algorithm

Quando si parla dell'espansione della chiave, si intende il processo mediante il quale la chiave di input viene trasformata in una sequenza di "parole di round" che vengono utilizzate durante l'esecuzione degli specifici round dell'algoritmo AES.

Inizialmente, la chiave di input viene rappresentata come una matrice quadra di byte. Ad esempio, nel caso di una chiave a 128 bit, la matrice sarà di dimensione 4x4, composta da 16 byte.

Successivamente, questa matrice di chiave viene espansa in una matrice di parole. Ogni parola è composta da 4 byte (32 bit). Durante l'espansione, vengono generati ulteriori byte e parole di round sulla base dei byte originali della chiave. Prende come input una 4-word 16 byte e produce un array di 44-word 156 byte. Questo è sufficiente per fornire una chiave rotonda di 4 word per la fase iniziale di Add Round Key e altre per i restanti round dell'algoritmo.
La chiave viene copiata nelle prime quattro parole della chiave espansa. Il resto della chiave espansa viene compilata in quattro parole alla volta. Ogni parola aggiunta w[i] dipende dalla parola immediatamente precedente, w[i - 1], e dalla parola quattro posizioni indietro, w[i - 4].
L'espansione della chiave in AES è il processo mediante il quale una chiave di input viene trasformata in una serie di sottochiavi, o chiavi di round, che vengono utilizzate durante i vari round dell'algoritmo AES.

Durante l'espansione della chiave, vengono generate una serie di sottochiavi a partire dalla chiave di input. Il numero di sottochiavi generate dipende dalla lunghezza della chiave e dal numero totale di round dell'algoritmo AES.

Per una chiave a 128 bit, ad esempio, vengono generate 11 sottochiavi (4 parole di 32 bit ciascuna) in totale, una per ogni round, compreso il round iniziale. La chiave di input viene divisa in parole da 4 byte ciascuna, che costituiscono la chiave di round iniziale.

Successivamente, attraverso un processo di espansione della chiave, le sottochiavi vengono generate sequenzialmente utilizzando operazioni come la sostituzione dei byte, la rotazione delle parole e

l'aggiunta di un byte di round costante. Queste operazioni vengono applicate alle parole precedenti per generare le nuove parole di round.

20.10 Mix columns and Add round key

- Mix columns: Opera su ciascuna colonna individualmente,Mappatura di ogni byte su un nuovo valore che è una funzione di tutti e quattro i byte in la colonna,Uso di equazioni su campi finiti,Fornire una buona combinazione di byte in colonna.
- Add Round key:Semplicemente XOR State con bit di chiave espansa,Protezione dalla complessità dell'espansione della chiave round e da altre fasi di AES.

20.11 Stream Ciphers:cifrature a flusso

Un cifrario a blocchi elabora l'input un blocco di elementi alla volta, producendo un blocco di uscita per ogni blocco di ingresso. Un cifrario a flusso elabora gli elementi di input continuamente, producendo un elemento alla volta, man mano che procede. Sebbene i cifrari a blocchi sono molto più comuni, ci sono alcune applicazioni in cui una cifratura a flusso è più appropriata.
Esaminiamo il cifrario a flusso più popolare RC4.

20.11.1 Stream cipher structure

Un cifrario a flusso opera crittografando il testo in chiaro uno o otto bit alla volta, utilizzando un flusso di numeri casuali generati da una chiave. Questo flusso di numeri casuali viene chiamato "keystream". Il keystream viene generato da un generatore di numeri pseudocasuali utilizzando la chiave di crittografia come input.

Per crittografare il testo in chiaro, viene eseguita un'operazione chiamata XOR (OR esclusivo) bit a bit tra il keystream e il testo in chiaro. Ad esempio, se il keystream successivo generato è "01101100" e il prossimo byte di testo in chiaro è "11001100", allora il byte di testo cifrato risultante sarà "10100000". Questo processo viene ripetuto per ogni byte del testo in chiaro.

Nella decriptazione, viene utilizzata la stessa sequenza random (keystream) generata in fase di crittografia. Il testo cifrato viene combinato con il keystream utilizzando l'operazione XOR per ottenere il testo in chiaro originale.

Il cifrario a flusso ha il vantaggio di essere veloce e richiedere meno codice rispetto ai cifrari a blocchi. Tuttavia, la sicurezza dipende dalla sicurezza della chiave di crittografia e del generatore di numeri pseudocasuali. Se la chiave o il generatore sono deboli o compromessi, la sicurezza del cifrario a flusso può essere compromessa.

D'altra parte, un cifrario a blocchi crittografa i dati in blocchi di dimensioni fisse, come 128 bit o 256 bit alla volta. Ogni blocco di testo in chiaro viene crittografato utilizzando una chiave deterministica. La sequenza di testo cifrato dipende dalla chiave e dai blocchi precedenti, quindi anche due blocchi identici di testo in chiaro avranno codifiche cifrate diverse.

I cifrari a blocchi sono noti per la loro robustezza e resistenza agli attacchi crittanalitici, ma richiedono più tempo per crittografare grandi quantità di dati rispetto ai cifrari a flusso.
Il cifrario a flusso può essere efficiente per la trasmissione di dati in tempo reale, come lo streaming video, poiché può cifrare i dati un carattere alla volta.

20.11.2 RC4 Algorithm

RC4 è un cifrario a flusso con dimensione variabile della chiave, con operazioni orientate ai byte. L'algoritmo si basa sull'uso di una permutazione casuale. Sono richieste da otto a sedici operazioni di macchina per ogni byte di output, e ci si aspetta che il cifrario funzioni molto velocemente in software. RC4 è utilizzato negli standard SSL/TLS (Secure Sockets Layer/Transport Layer Security) che sono stati definiti per la comunicazione tra browser Web e server. È anche utilizzato nel protocollo WEP (Wired Equivalent Privacy) e nel protocollo più recente WPA (WiFi Protected Access), che fanno parte dello standard LAN wireless IEEE 802.11. RC4 è stato tenuto segreto da RSA Security. Nel settembre 1994, l'algoritmo RC4 è stato pubblicato in modo anonimo su Internet nella lista di remailer anonimi dei Cypherpunks.

L'algoritmo RC4 è notevolmente semplice e abbastanza facile da spiegare. Viene utilizzata una chiave di lunghezza variabile da 1 a 256 byte per inizializzare un vettore di stato S di 256 byte, con elementi S[0], S[1], ..., S[255]. Ogni elemento del vettore di stato, S[0] fino a S[255], rappresenta un numero a 8 bit da 0 a 255. In ogni momento, S contiene una permutazione di tutti i numeri a 8 bit da 0 a 255. Per la cifratura e la decifratura, un byte k viene generato da S selezionando una delle 255 voci in modo sistematico. Man mano che ogni valore di k viene generato, le voci in S vengono nuovamente permutate.

- Inizializzazione RC4: Per iniziare, le voci di S sono impostate uguali ai valori from da 0 a 255 in ordine crescente; cioè S[0] = 0, S[1] = 1, ... , S[255] = 255.Viene creato anche un vettore temporaneo chiamato T. Se la lunghezza della chiave K è di 256 byte, allora la chiave K viene copiata in T. Altrimenti, se la lunghezza della chiave K è diversa da 256 byte, i primi elementi di T vengono copiati dalla chiave K fino a raggiungere una lunghezza pari a quella della chiave (keylen). Successivamente, se necessario, la chiave K viene ripetuta più volte per riempire completamente il vettore T.
  In pratica, questo passaggio inizializza il vettore di stato S utilizzando i valori da 0 a 255 in ordine crescente e crea un vettore temporaneo T che rappresenta la chiave di cifratura. Se la chiave è più corta di 256 byte, viene ripetuta per riempire il vettore T.
  Quindi, utilizzando il vettore temporaneo T come guida, si eseguono gli scambi tra gli elementi di S. Questa operazione di scambio viene eseguita per generare una permutazione casuale degli elementi di S, creando un ordine diverso da quello iniziale. Ciò assicura che il vettore di stato S abbia una disposizione casuale dei numeri da 0 a 255.

  In sostanza, il passaggio descritto implica l'applicazione di scambi tra gli elementi di S sulla base delle istruzioni fornite dal vettore temporaneo T. Alla fine del processo, S conterrà ancora tutti i numeri da 0 a 255, ma disposti in una permutazione casuale.

- Stream Generation: Cicla attraverso gli elementi di S, selezionando un byte alla volta. Fai l'operazione di XOR tra il byte selezionato di S e il corrispondente byte del testo (o del testo cifrato) per ottenere un byte del keystream,dopo 255 ripete da 0.Per decriptare uguale solo invece del testo in chiaro si usa il testo cifrato.
  Quando si crittografa, si applica l'operazione di XOR tra il keystream generato e il testo in chiaro. Il risultato è il testo cifrato, che rappresenta il blocco di dati crittografato.

  Durante la decriptazione, viene utilizzato lo stesso keystream generato in fase di crittografia. Si applica l'operazione di XOR tra il keystream e il testo cifrato per ottenere il testo in chiaro originale.

20.11.3 Modalità di funzionamento del cifrario a blocchi

Un cifrario a blocchi simmetrico elabora un blocco di dati alla volta. Nel caso di DES e 3DES, la lunghezza del blocco è di 64 bit. Per quantità più lunghe di testo in chiaro, è necessario sary per suddividere il testo in chiaro in blocchi a 64 bit (imbottendo l'ultimo blocco se necessario).
Per applicare un cifrario a blocchi in una varietà di applicazioni, sono disponibili cinque modalità di funzionamento.Le cinque modalità sono intese per coprire virtualmente tutte le possibili applicazioni di cifratura per le quali un cifrario a blocchi potrebbe essere usato. Queste modalità sono destinate all'uso con qualsiasi cifrario a blocchi simmetrico, inclusi triplo DES e AES.Vediamo le più importanti.

- Electronic Codebook Mode(ECB): Modalità più semplice, il testo in chiaro viene gestito b bit alla volta e ogni blocco è crittografato utilizzando la stessa chiave. Viene utilizzato "Codebook" perché esiste un testo cifrato univoco per ogni blocco di b bit di testo in chiaro. Non sicuro per messaggi lunghi poiché il testo in chiaro ripetuto è visto nel testo cifrato ripetuto. Ad esempio, se è noto che il messaggio inizia sempre con un determinato campo predefinito,Se il messaggio ha elementi ripetitivi, con un periodo di ripetizione multiplo di b bit, allora,questi elementi possono essere individuati dall'analista. Per superare le carenze di sicurezza è necessaria una tecnica in cui la stessa,blocco di testo in chiaro, se ripetuto, produce diversi blocchi di testo cifrato.

- Cipher Block Chaining (CBC) mode: nell'algoritmo di crittografia l'input è lo XOR tra il blocco di testo in chiaro corrente e il blocco cifrato precedente,viene usata la stessa chiave per ogni blocco.L'input alla funzione di crittografia zione per ogni blocco di testo in chiaro non ha alcuna

relazione fissa con il blocco di testo in chiaro. Pertanto, i pattern ripetuti di b-bit non vengono esposti. Per decriptare un blocco cifrato, si fa lo XOR tra il blocco con il blocco cifrato precedente per produrre il blocco di testo in chiaro.

Se ogni blocco cifrato dipende dalla chiave e dal blocco precedente,il primo blocco come lo creo?
Initialization Vector: è un vettore di inizializzazione che va in XOR con il primo blocco di testo in chiaro insieme alla chiave,questo vettore viene usato anche nella decriptazione per prendere il primo blocco di testo in chiaro.
L'IV deve essere noto sia al mittente che al destinatario. Per la massima sicurezza, il vettore dovrebbe essere protetto così come la chiave.
Ciò potrebbe essere fatto inviando l'IV utilizzando la crittografia ECB.
Uno dei motivi per proteggere l'IV è il seguente:
Se un avversario è in grado di ingannare il ricevitore facendogli usare un valore diverso per IV, allora l'avversario è in grado di invertire i bit selezionati nel primo blocco di testo in chiaro.

20.12 Error Propagation in CBC

Con la modalità ECB, se c'è un errore in un blocco del testo cifrato trasmesso, solo il corrispettivo blocco in chiaro è compromesso.
Tuttavia, in modalità CBC, questo errore si propaga.Per esempio, un errore nel Ck trasmesso ovviamente corrompe Pk e Pk+1.
Sono interessati eventuali blocchi oltre Pk+1?No.
Nell'algoritmo di decriptazione un blocco P(plaintext) viene calcolato a partire da C(criptotext) e da C-1.Ogni P dipende da C e C-1,e ogni errore in C compromette P e P+1,quindi un errore in C non compromette altri blocchi.
Per criptare,supponiamo ci sia un errore in P, un errore in P influisce su C. Ogni C lo calcolo a partire da P e da C-1,quindi C influisce su C+1.Questo effetto continua indefinitamente, in modo che tutti i blocchi di testo cifrato (eccetto quelli prima di C) sono interessati. se un blocco di plaintext Pk è corrotto, il blocco di ciphertext corrispondente Ck sarà anch'esso corrotto. Inoltre, poiché il blocco Ck è utilizzato nel calcolo del blocco di ciphertext successivo Ck+1, l'errore si propagherà ai blocchi successivi di ciphertext. Pertanto, Ck+1 sarà corrotto e così via per i blocchi successivi.

Durante la decodifica, quando si arriva al blocco di plaintext Pk+i, si utilizzano sia il blocco di ciphertext corrispondente Ck+i che il blocco di ciphertext precedente Ck+i-1. Entrambi i blocchi di ciphertext saranno corrotti a causa dell'errore propagato.

Tuttavia, l'algoritmo di decodifica CBC continuerà comunque a decodificare il blocco di plaintext Pk+i utilizzando Ck+i corrotto e Ck+i-1 corrotto. Ciò significa che il blocco di plaintext Pk+i sarà decriptato correttamente, a eccezione del primo blocco Pk che è stato decriptato utilizzando Ck corrotto e Ck-1 non corrotto.

In altre parole, l'errore nell'ultimo blocco di ciphertext influirà solo sul blocco di plaintext corrispondente Pk, mentre gli altri blocchi di plaintext saranno ripristinati correttamente durante la decodifica utilizzando i blocchi di ciphertext corrispondenti corrotti.

20.13 Cipher Feedback Mode

È possibile convertire qualsiasi cifrario a blocchi in un cifrario a flusso utilizzando la modalità di cifratura feedback (CFB). Un cifrario a flusso elimina la necessità di imbottire un messaggio in modo che sia un numero integrale di blocchi. Inoltre, può operare in tempo reale. Pertanto, se viene trasmesso un flusso di caratteri, ogni carattere può essere crittografato e trasmesso immediatamente con un cifrario a flusso orientato ai caratteri.
Una proprietà desiderabile di un cifrario a flusso è che il testo cifrato abbia la stessa lunghezza del testo in chiaro. Pertanto, se vengono trasmessi caratteri a 8 bit, ogni carattere deve essere crittografato con 8 bit. Se si utilizzano più di 8 bit, si spreca capacità di trasmissione.
Per la decodifica, si utilizza lo stesso schema, tranne che per il fatto che l'unità di testo cifrato ricevuta viene XORata con l'uscita della funzione di crittografia per produrre l'unità di testo in chiaro. Si noti che è la funzione di crittografia a essere utilizzata, non quella di decrittografia.

20.14 Counter Mode:CTR mode

Viene utilizzato un contatore pari alla dimensione del blocco di testo in chiaro. L'unico requisito indicato nel documento SP 800-38A è che il valore del contatore deve essere diverso per ogni blocco di testo in chiaro. deve essere diverso per ogni blocco di testo in chiaro che viene crittografato. In genere, il contatore viene inizializzato a un certo valore e poi incrementato di 1 per ogni blocco successivo. (modulo 2b, dove b è la dimensione del blocco). Per la crittografia, il contatore viene criptato e e poi XORato con il blocco di testo in chiaro per produrre il blocco di testo cifrato;Per la decifrazione, viene utilizzata la stessa sequenza di valori dei contatori, e ogni contatore crittografato viene XORato con un blocco di testo cifrato per recuperare il blocco di testo in chiaro corrispondente. corrispondente blocco di testo in chiaro.
Lista dei vantaggi di CTRmode

- Efficienza Hardware: CTR può essere eseguito in parallelo su diversi blocchi plaintext o ciphertext,in questo modo il throughput è limitato solo da quanto si riesce a parallelizzare,con il concatenamento invece ogni blocco necessita di quello precedente.
- Efficenza software: anche il software può essere usato in modo efficente per lo stesso discorso hardware,si può eseguire in parallelo.
- Preprocessing: La preelaborazione consiste quindi nella preparazione del keystream per i successivi blocchi di dati. Una volta generato il keystream iniziale utilizzando il contatore, è possibile pre-calcolare i successivi keystreams incrementando semplicemente il contatore anziché avviare nuovamente il generatore di numeri pseudo-casuali. Questo approccio migliora l'efficienza computazionale, poiché evita il sovraccarico di chiamate ripetute al generatore di numeri pseudo-casuali per ciascun blocco di dati.

  In breve, la preelaborazione in CTR mode si riferisce alla generazione iniziale del keystream utilizzando un contatore e un generatore di numeri pseudo-casuali, nonché alla preparazione del keystream per i successivi blocchi di dati tramite l'incremento del contatore. Questo permette di ottenere il flusso di chiavi necessario per la cifratura e la decrittografia in modo efficiente.
- Random access: E' possibile elaborare l'i-esimo blocco di testo in chiaro o cifrato con accesso casuale. Con le modalità di concatenamento, il blocco Ci non può essere com- messo fino a quando non vengono calcolati gli i - 1 blocchi precedenti. Interessante per le applicazioni in cui è memorizzato un testo cifrato e si desidera decifrare un solo blocco.
- Sicurezza dimostrabile: Si può dimostrare che CTR è sicuro almeno quanto le altre modalità
- Semplicità: A differenza delle modalità ECB e CBC, la modalità CTR richiede solo l'implementazione dell'algoritmo di crittografia

20.15 Key distribution

Affinchè la crittografia simmetrica funzioni, le due parti devono condividere la stessa chiave e quella chiave deve essere protetta dall'accesso da parte di altri. Inoltre è opportuno apportare frequenti modifiche alle chiavi per limitare la quantità di dati compromessi nel caso in cui un utente malintenzionato apprende la chiave. La forza di qualsiasi sistema crittografico dipende dalla tecnica di distribuzione delle chiavi per evitare accessi non autorizzati, può essere ottenuta in diversi modi.
Per due parti A e B:
-Una chiave potrebbe essere selezionata da A e consegnata fisicamente a B
- Una terza parte potrebbe selezionare una chiave e consegnarla fisicamente ad A e B
-Se A e B hanno utilizzato in precedenza una chiave,una delle parti potrebbe trasmettere la nuova chiave criptata all'altro,utilizzando la vecchia chiave per criptare
-Se A e B hanno una connessione sicura(criptata) verso una terza parte C,C può inviare usando questi link di comunicazione la chiave ad A e B.
Le opzioni 1 e 2 richiedono la consegna manuale di una chiave. Per la crittografia di collegamento(link encryption), questo è un requisito ragionevole, poiché ogni dispositivo di crittografia di collegamento dovrà solo scambiare dati con il suo partner all'altro capo del collegamento. Tuttavia, per la crittografia end-to-end, la consegna manuale è scomoda. In un sistema distribuito, ogni host o terminale

potrebbe dover scambiare dati con molti altri host e terminali nel tempo. Pertanto, ogni dispositivo ha bisogno di un numero di chiavi, fornite dinamicamente. Il problema è particolarmente difficile in un sistema distribuito su vasta area.

L'opzione 3 è una possibilità sia per la crittografia di collegamento che per la crittografia end-to-end, ma se un attaccante riesce a ottenere accesso a una chiave, allora tutte le chiavi successive sono rivelate. Anche se vengono effettuati frequenti cambiamenti alle chiavi di crittografia di collegamento, questi dovrebbero essere fatti manualmente. Per fornire chiavi per la crittografia end-to-end, l'opzione 4 è preferibile.

La figura 20.10 illustra un'implementazione che soddisfa l'opzione 4 per la crittografia end-to-end. Nella figura, la crittografia di collegamento viene ignorata. Questa può essere aggiunta o meno a seconda delle necessità. Per questo schema, vengono identificati due tipi di chiavi:
Chiave di sessione(session key): Viene utilizzata per crittografare i dati durante una specifica sessione di comunicazione tra due sistemi finali, come host o terminali. Quando i due sistemi desiderano comunicare, stabiliscono una connessione logica o un circuito virtuale. Durante la durata di questa connessione logica, tutti i dati utente(quelli che invia un utente) vengono crittografati con una chiave di sessione generata una sola volta. Alla conclusione della sessione o connessione, la chiave di sessione viene distrutta. Questo approccio di utilizzare una chiave di sessione unica per ogni sessione di comunicazione contribuisce alla sicurezza complessiva del sistema, poiché anche se un attaccante intercetta la chiave di sessione, questa avrà validità solo per quella specifica sessione e non potrà essere utilizzata per decifrare comunicazioni future.
Chiave permanente: E' una chiave utilizzata tra entità per il fine di distribuire le chiavi di sessione ai sistemi finali. La chiave permanente viene utilizzata per stabilire una relazione di fiducia tra le entità coinvolte nel processo di comunicazione. Questa chiave permanente viene scambiata in modo sicuro tra le entità inizialmente o attraverso un processo di autenticazione sicuro. Una volta stabilita una connessione sicura utilizzando la chiave permanente, questa può essere utilizzata per distribuire le chiavi di sessione per le future sessioni di comunicazione tra le entità.
Entità coinvolte: Key Distribution Center (KDC) e il Security service module (SSM).
Il Key Distribution Center (KDC) è un'entità responsabile della distribuzione delle chiavi e della gestione dell'autenticazione all'interno del sistema. Il suo compito principale è determinare quali sistemi sono autorizzati a comunicare tra loro. Quando due entità desiderano comunicare, il KDC fornisce una chiave di sessione monouso per quella connessione specifica. Questa chiave di sessione viene utilizzata per crittografare i dati scambiati tra le due entità durante la comunicazione.

Il modulo di servizio di sicurezza (SSM) può consistere in funzionalità a un livello di protocollo specifico. Il suo ruolo è quello di eseguire la crittografia end-to-end e ottenere le chiavi di sessione a nome degli utenti. In pratica, l'SSM gestisce l'interazione con il KDC per ottenere la chiave di sessione appropriata per la connessione desiderata. Utilizza questa chiave di sessione per crittografare i dati dell'utente prima di inviarli e decrittografarli quando vengono ricevuti.

Quindi, il KDC determina quali sistemi sono autorizzati a comunicare e fornisce le chiavi di sessione, mentre l'SSM gestisce la crittografia end-to-end e ottiene le chiavi di sessione dai servizi di sicurezza a nome degli utenti. Questo sistema permette una comunicazione sicura tra le entità e protegge i dati durante la trasmissione.
L'host invia un pacchetto richiedendo una connessione. Questo potrebbe essere il primo passo per avviare una comunicazione con un altro host.

Il servizio di sicurezza (Security Service) riceve il pacchetto e lo memorizza nel buffer. Successivamente, richiede al Key Distribution Center (KDC) una chiave di sessione per la connessione. La comunicazione tra l'SSM e il KDC viene crittografata utilizzando una chiave master condivisa solo tra l'SSM e il KDC.

Il KDC distribuisce la chiave di sessione a entrambi gli host coinvolti nella comunicazione. Questa chiave di sessione sarà utilizzata per crittografare i dati scambiati tra gli host durante la connessione.

Il pacchetto precedentemente memorizzato nel buffer del servizio di sicurezza viene trasmesso. Ora che entrambi gli host hanno ricevuto la chiave di sessione dal KDC, il servizio di sicurezza può utilizzare questa chiave per crittografare il pacchetto prima di inviarlo.

Questo approccio di distribuzione delle chiavi automatica fornisce la flessibilità e le caratteristiche dinamiche necessarie per consentire a più utenti terminali di accedere a più host e per gli host di scambiare dati tra di loro.
Quindi, il processo descrive una sequenza di azioni in cui viene richiesta una connessione, viene ot-

tenuta una chiave di sessione dal KDC e successivamente viene trasmesso un pacchetto utilizzando la chiave di sessione per garantire la sicurezza della comunicazione.
Nel processo descritto, non viene specificato esplicitamente dove si trovi la chiave permanente. Tuttavia, possiamo assumere che la chiave permanente sia già stata stabilita e condivisa in precedenza tra il KDC e gli host coinvolti nella comunicazione.

La chiave permanente può essere memorizzata in un luogo sicuro all'interno del KDC o può essere condivisa tramite un processo di scambio di chiavi sicuro tra le entità coinvolte. Questo scambio può avvenire al momento dell'inizializzazione del sistema o può essere gestito da un'autorità di certificazione (CA) che emette certificati digitali contenenti le chiavi pubbliche associate alle entità coinvolte.

Una volta stabilita la chiave permanente, viene utilizzata per stabilire una relazione di fiducia tra le entità e per facilitare la distribuzione delle chiavi di sessione per le comunicazioni future. La chiave permanente può essere utilizzata come base per generare le chiavi di sessione o come parte del processo di autenticazione per accertare l'identità delle entità coinvolte.

21 Autenticazione di messaggi

Protegge dagli attacchi attivi.
Verifica che il messaggio ricevuto sia autentico.

- Contenuti non alterati
- Devono venire da una sorgente autentica
- Devono arrivare puntuali e nell'ordine con cui sono stati inviati

Può usare una crittografia convenzionale.

- Solo chi invia e riceve condividono la chiave

Nella modalità di crittografia ECB, se un utente malintenzionato riordina i blocchi di testo cifrato, allora ogni blocco si potrà ancora decifrare con successo. Tuttavia, il riordino potrebbe cambiare il significato dei dati complessivi.
Sebbene possano essere utilizzati numeri di sequenza a un certo livello (ad esempio, per ogni pacchetto IP), di solito non viene associato un numero di sequenza separato a ogni blocco di testo in chiaro. Pertanto, il riordinamento dei blocchi rappresenta una minaccia per la corretta interpretazione dei dati quando vengono decifrati.

21.1 Autenticazione del messaggio senza Confidentiality

L'autenticazione del messaggio si riferisce alla verifica dell'integrità e all'origine del messaggio. Ciò significa che il destinatario può verificare che il messaggio ricevuto sia stato effettivamente inviato dall'entità attesa e che il messaggio non sia stato modificato durante la trasmissione.

La crittografia del messaggio da sola non fornisce un metodo sicuro di autenticazione. Mentre la crittografia protegge il contenuto del messaggio rendendolo illeggibile a terzi, non garantisce che il mittente sia autentico o che il messaggio non sia stato alterato.

Per combinare l'autenticazione e la riservatezza in un singolo algoritmo, è possibile crittografare il messaggio insieme al suo "tag" di autenticazione. Il tag di autenticazione è una firma crittografica o un codice di autenticazione che viene calcolato utilizzando una funzione crittografica sicura, come ad esempio l'HMAC (Hash-based Message Authentication Code). Crittografando sia il messaggio che il tag di autenticazione, è possibile fornire sia la riservatezza che l'autenticazione del messaggio.

Di solito, l'autenticazione del messaggio viene fornita come una funzione separata rispetto alla crittografia del messaggio. Ciò significa che i due aspetti possono essere trattati indipendentemente e combinati se necessario.

Ci sono situazioni in cui l'autenticazione del messaggio senza riservatezza può essere preferibile. Ad esempio, quando lo stesso messaggio viene trasmesso a più destinazioni, l'autenticazione può essere utile per garantire che tutte le destinazioni ricevano lo stesso messaggio originale. Inoltre, in uno scambio di messaggi in cui una delle parti ha un carico di lavoro elevato e non può permettersi di decifrare tutti i messaggi in entrata, l'autenticazione del messaggio può essere utilizzata per verificare la validità dei

messaggi senza doverli decifrare completamente. Infine, l'autenticazione di un programma informatico in testo normale può essere un servizio attraente in cui si desidera verificare l'integrità del programma senza svelarne il contenuto riservato.

In sintesi, l'autenticazione del messaggio senza riservatezza si concentra sulla verifica dell'integrità e dell'origine del messaggio, mentre la riservatezza si riferisce all'assicurazione che il messaggio non sia leggibile da terzi non autorizzati. La combinazione di autenticazione e riservatezza può essere realizzata crittografando il messaggio insieme al suo tag di autenticazione.

21.2 MAC:Message Authentication Code

Una tecnica di autenticazione coinvolge l'uso di una chiave segreta per generare un piccolo blocco di dati, noto come codice di autenticazione del messaggio (MAC, Message Authentication Code), che viene aggiunto al messaggio. Questa tecnica assume che due parti comunicanti, diciamo A e B, condividano una chiave segreta comune K-ab. Quando A ha un messaggio da inviare a B, calcola il codice di autenticazione del messaggio come una funzione complessa del messaggio e della chiave: MACm = F(K-ab, m).

Il messaggio più il codice vengono trasmessi al destinatario previsto. Il destinatario esegue lo stesso calcolo sul messaggio ricevuto, utilizzando la stessa chiave segreta, per generare un nuovo codice di autenticazione del messaggio. Il codice ricevuto viene confrontato con il codice calcolato. In questo modo, il destinatario può verificare l'autenticità del messaggio confrontando i due codici.
Se assumiamo che solo il destinatario e il mittente conoscano l'identità della chiave segreta e se il codice ricevuto corrisponde al codice calcolato, allora:

Il destinatario può essere certo che il messaggio non sia stato alterato. Se un attaccante modifica il messaggio ma non il codice, il calcolo del codice da parte del destinatario sarà diverso dal codice ricevuto. Poiché si assume che l'attaccante non conosca la chiave segreta,un attaccante cerca di modificare il contenuto del messaggio senza alterare il codice di autenticazione, il destinatario rileverà l'incoerenza tra il messaggio ricevuto e il codice calcolato. Poiché l'attaccante non conosce la chiave segreta utilizzata per generare il codice, non può modificare il codice in modo tale da farlo corrispondere alle modifiche apportate al messaggio. In altre parole, il destinatario può rilevare l'alterazione del messaggio confrontando il codice ricevuto con quello calcolato in base alla chiave segreta condivisa.

Il destinatario può essere certo che il messaggio provenga dal presunto mittente. Poiché nessun'altra persona conosce la chiave segreta, nessun'altra persona potrebbe preparare un messaggio con un codice corretto.

Se il messaggio include un numero di sequenza (come avviene con X.25, HDLC e TCP), il destinatario può essere certo della corretta sequenza, perché un attaccante non può modificare con successo il numero di sequenza.

È possibile utilizzare diversi algoritmi per generare il codice. La specifica NIST, FIPS PUB 113, raccomanda l'uso di DES. DES viene utilizzato per generare una versione crittografata del messaggio e gli ultimi bit del testo crittografato vengono utilizzati come codice. Un codice di 16 o 32 bit è tipico.

Il processo appena descritto è simile alla crittografia. Una differenza è che l'algoritmo di autenticazione non ha bisogno di essere reversibile, come invece deve essere per la decrittazione. Risulta che, a causa delle proprietà matematiche della funzione di autenticazione, essa è meno vulnerabile all'essere violata rispetto alla crittografia.

21.3 One-Way Hash Function

Un'alternativa al codice di autenticazione del messaggio(MAC) è la funzione di hash unidirezionale (one-way hash function). Come il codice di autenticazione del messaggio, una funzione di hash accetta un messaggio di dimensione variabile M in ingresso e produce un valore di hash (message digest) di dimensione fissa H(M) in uscita. Tipicamente, il messaggio viene riempito fino a diventare un multiplo intero di una certa lunghezza fissa (ad esempio, 1024 bit) e il riempimento include il valore della lunghezza del messaggio originale in bit. Il campo di lunghezza è una misura di sicurezza per aumentare la difficoltà per un attaccante di produrre un messaggio alternativo con lo stesso valore di hash. A differenza del codice di autenticazione del messaggio, una funzione di hash non richiede un input di chiave segreta.
Come funziona?
una funzione di hash prende solo il messaggio in ingresso e produce un valore di hash (digest). La

funzione di hash è una funzione unidirezionale, il che significa che non è possibile ottenere il messaggio originale a partire dal valore di hash. La funzione di hash è utilizzata per verificare l'integrità dei dati. Se anche un singolo bit del messaggio cambia, il valore di hash risultante sarà completamente diverso. La funzione di hash non richiede una chiave segreta e può essere utilizzata per generare un valore di hash per qualsiasi tipo di dati.
Il destinatario può ottenere il messaggio originale utilizzando il valore di hash generato dal mittente e la stessa funzione di hash. Quando il messaggio e il valore di hash vengono ricevuti, il destinatario calcola nuovamente il valore di hash del messaggio utilizzando la stessa funzione di hash. Se il valore di hash calcolato corrisponde al valore di hash ricevuto, allora il destinatario può essere certo che il messaggio non è stato alterato durante la trasmissione.

Tuttavia, è importante notare che la funzione di hash è unidirezionale, il che significa che non è possibile ricostruire il messaggio originale direttamente dal valore di hash. La funzione di hash prende in ingresso il messaggio e produce il valore di hash, ma non può essere invertita per ottenere il messaggio originale.

Pertanto, se il destinatario ha bisogno del messaggio originale, il mittente deve inviarlo insieme al valore di hash. In tal caso, il destinatario può calcolare nuovamente il valore di hash del messaggio ricevuto e confrontarlo con il valore di hash fornito per verificare l'integrità del messaggio.

Illustriamo tre modi in cui il messaggio può essere autenticato utilizzando una funzione di hash.
Crittografia simmetrica del valore di hash: Il valore di hash del messaggio viene crittografato utilizzando una chiave di crittografia condivisa tra il mittente e il destinatario. Solo il mittente e il destinatario conoscono questa chiave. Quando il destinatario riceve il messaggio crittografato, può decifrarlo utilizzando la stessa chiave di crittografia e verificare se il valore di hash calcolato sul messaggio decifrato corrisponde al valore di hash crittografato ricevuto. Se corrispondono, il destinatario può autenticare il messaggio.

Crittografia a chiave pubblica del valore di hash: il mittente calcola il valore di hash del messaggio utilizzando una funzione di hash crittografica. Quindi, il mittente crittografa il messaggio stesso utilizzando la chiave privata per creare una firma digitale, non il valore di hash. Il mittente invia sia il messaggio in chiaro che la firma digitale (il messaggio crittografato con la chiave privata) al destinatario. Il destinatario riceve il messaggio in chiaro e la firma digitale e utilizza la chiave pubblica del mittente per decriptare la firma digitale e ottenere il valore di hash originale. Successivamente, il destinatario calcola nuovamente il valore di hash del messaggio ricevuto utilizzando la stessa funzione di hash crittografica. Per verificare l'integrità del messaggio e l'autenticità del mittente, il destinatario confronta il valore di hash appena calcolato con il valore di hash originale ottenuto dalla decrittazione della firma digitale. Se i due valori di hash corrispondono, il destinatario può essere ragionevolmente certo che il messaggio non sia stato alterato durante la trasmissione e che sia stato inviato dal mittente autentico.

Keyed hash MAC (Message Authentication Code), mittente e destinatario condividono una chiave segreta comune. Questo tipo di autenticazione del messaggio utilizza una funzione di hash crittografica insieme alla chiave segreta per calcolare un codice di autenticazione del messaggio, chiamato MAC.

Ecco come funziona il processo:

Mittente: Il mittente calcola il valore di hash del messaggio concatenando la chiave segreta con il messaggio stesso utilizzando una funzione di hash crittografica. Il risultato è il MAC.

Mittente: Il mittente invia sia il messaggio originale che il MAC calcolato al destinatario.

Destinatario: Il destinatario riceve il messaggio e il MAC.

Destinatario: Utilizzando la stessa chiave segreta condivisa con il mittente, il destinatario calcola il valore di hash del messaggio ricevuto concatenando la chiave segreta con il messaggio utilizzando la stessa funzione di hash crittografica.

Destinatario: Il destinatario confronta il MAC calcolato localmente con il MAC inviato dal mittente. Se i due MAC corrispondono, il destinatario può autenticare il messaggio.

La chiave segreta condivisa tra il mittente e il destinatario garantisce che solo coloro che conoscono la chiave possano generare un MAC corretto. Poiché la chiave segreta non viene inviata insieme al messaggio, un attaccante non può modificarlo senza conoscere la chiave segreta.
La funzione di hash viene principalmente utilizzata per garantire l'integrità dei dati, mentre per la trasmissione sicura di messaggi con autenticazione e riservatezza, viene spesso utilizzato un MAC o una combinazione di crittografia e autenticazione.

21.4 Hash Function Requirements

Le funzioni di hash hanno determinati requisiti per poter essere utili per l'autenticazione del messaggio. Questi requisiti sono i seguenti:

H può essere applicato a un blocco di dati di qualsiasi dimensione. H produce un output di lunghezza fissa. H(x) è relativamente facile da calcolare per qualsiasi x, rendendo pratiche sia le implementazioni hardware che software. Per ogni codice h dato, è computazionalmente impossibile trovare x tale che H(x) = h. Una funzione di hash con questa proprietà viene definita come una funzione "one-way" o resistente alla pre-immagine. Per ogni blocco x dato, è computazionalmente impossibile trovare y diverso x tale che H(y) = H(x). Una funzione di hash con questa proprietà viene definita come resistente alla seconda pre-immagine. Questo è talvolta indicato come resistente alle collisioni deboli. È computazionalmente impossibile trovare una coppia (x, y) tale che H(x) = H(y). Una funzione di hash con questa proprietà viene definita come resistente alle collisioni. Questo è talvolta indicato come resistente alle collisioni forti. Le prime tre proprietà sono requisiti per l'applicazione pratica di una funzione di hash per l'autenticazione del messaggio.

La quarta proprietà è la proprietà "one-way": è facile generare un codice dato un messaggio, ma virtualmente impossibile generare un messaggio dato un codice. Questa proprietà è importante se la tecnica di autenticazione coinvolge l'uso di un valore segreto. Il valore segreto stesso non viene inviato; tuttavia, se la funzione di hash non è "one-way", un attaccante può facilmente scoprire il valore segreto. Se l'attaccante può osservare o intercettare una trasmissione, ottiene il messaggio M e il codice di hash MDM = H(SAB —— M). L'attaccante quindi inverte la funzione di hash per ottenere SAB —— M = H-1(MDM). Poiché l'attaccante ha ora sia M che SAB —— M, è semplice recuperare SAB.

La quinta proprietà garantisce che sia impossibile trovare un messaggio alternativo con lo stesso valore di hash di un dato messaggio. Ciò impedisce la falsificazione quando viene utilizzato un codice di hash crittografato. Se questa proprietà non fosse vera, un attaccante potrebbe eseguire la seguente sequenza: primo, osservare o intercettare un messaggio più il suo codice di hash crittografato; secondo, generare un codice di hash non crittografato dal messaggio; terzo, generare un messaggio alternativo con lo stesso codice di hash. Una funzione di hash che soddisfa le prime cinque proprietà viene definita come una funzione di hash debole. Se la sesta proprietà è soddisfatta, allora viene definita come una funzione di hash forte. Una funzione di hash forte protegge contro un attacco in cui una parte genera un messaggio da firmare per conto di un'altra parte.

Oltre a fornire autenticazione, un messaggio di digest fornisce anche l'integrità dei dati.

21.5 Security of Hash function

Resistenza di una funzione di hash contro gli attacchi di crittoanalisi e di forza bruta.
La crittoanalisi di una funzione di hash coinvolge lo sfruttamento di debolezze logiche nell'algoritmo. Gli attacchi di crittoanalisi cercano di trovare vulnerabilità nell'algoritmo stesso per ottenere informazioni sul messaggio originale o generare collisioni.

D'altra parte, un attacco di forza bruta implica l'esecuzione di un numero elevato di tentativi per trovare una collisione o una corrispondenza di hash. La resistenza di una funzione di hash agli attacchi di forza bruta dipende principalmente dalla lunghezza del codice di hash prodotto dall'algoritmo.Nel contesto del brano, si menziona che se si desidera una resistenza alle collisioni, la lunghezza del codice di hash influisce sulla forza dell'algoritmo contro gli attacchi di forza bruta. Ad esempio, per un codice di hash di lunghezza "n", lo sforzo richiesto è proporzionale a 2 alla (n/2).

I ricercatori Van Oorschot e Wiener hanno presentato un progetto di una macchina di ricerca di collisioni per l'algoritmo MD5, che ha una lunghezza del codice di hash di 128 bit. Questa macchina avrebbe potuto trovare una collisione in 24 giorni. Di conseguenza, un codice di 128 bit può essere considerato inadeguato.

Un passo successivo, se si considera il codice di hash come una sequenza di 32 bit, è una lunghezza del codice di hash di 160 bit. Con una lunghezza del codice di hash di 160 bit, la stessa macchina di ricerca richiederebbe oltre quattromila anni per trovare una collisione. Tuttavia, con l'avanzamento della tecnologia, il tempo sarebbe molto più breve. Pertanto, attualmente una lunghezza di 160 bit può risultare sospetta in termini di sicurezza.
Miglioramento a una semplice funzione di hash mediante l'esecuzione di uno switch circolare di 1 bit del valore di hash dopo l'elaborazione di ogni blocco di dati. Ecco una spiegazione del procedimento:

Inizializzazione:

Impostare il valore di hash a n bit su zero. Elaborazione di ogni blocco di dati successivo come segue:

Ruotare il valore di hash corrente di 1 bit verso sinistra. Eseguire l'operazione XOR tra il blocco di dati e il valore di hash. Ripetere il passaggio 2 per ogni blocco di dati. L'approccio di uno switch circolare di 1 bit è utile per aggiungere una variazione al valore di hash dopo ogni blocco di dati elaborato, al fine di rendere il processo di hashing più robusto e meno prevedibile. Tuttavia, l'uso di RXOR (operazione di XOR con rotazione) può essere inutile se il codice di hash crittografato viene utilizzato con il testo normale, in quanto la crittografia può nascondere l'effetto dell'operazione di XOR. Pertanto, se il digest (codice di hash) viene crittografato insieme al testo normale, l'utilità dell'approccio RXOR potrebbe essere limitata.

21.6 Secure Hash Algorithm(SHA)

SHA-1, SHA-2, SHA-3 e SHA-512 sono algoritmi di hash crittografici appartenenti alla famiglia Secure Hash Algorithm (SHA), sviluppati dal National Institute of Standards and Technology (NIST) degli Stati Uniti.

Ecco una panoramica di ciascun algoritmo e le loro principali differenze:

SHA-1 (Secure Hash Algorithm 1): È stato il primo algoritmo della famiglia SHA ed è stato ampiamente utilizzato in passato. Tuttavia, è considerato debole dal punto di vista della sicurezza e non viene più raccomandato per nuove applicazioni crittografiche. Produce un hash di 160 bit.

SHA-2 (Secure Hash Algorithm 2): Questa famiglia di algoritmi di hash include SHA-224, SHA-256, SHA-384 e SHA-512. Questi algoritmi sono considerati sicuri e sono comunemente utilizzati per scopi crittografici. La principale differenza tra di loro è la lunghezza dell'hash prodotto: SHA-224 produce un hash di 224 bit, SHA-256 di 256 bit, SHA-384 di 384 bit e SHA-512 di 512 bit.

SHA-3 (Secure Hash Algorithm 3): È il successore di SHA-2 ed è stato progettato per essere più sicuro e resistente agli attacchi crittografici noti. L'algoritmo utilizza una struttura chiamata Keccak, che è stata selezionata come vincitrice del concorso per lo sviluppo di SHA-3 nel 2012. Le varianti di SHA-3 includono SHA3-224, SHA3-256, SHA3-384 e SHA3-512, che producono hash rispettivamente di 224, 256, 384 e 512 bit.

Le principali differenze tra SHA-1, SHA-2 e SHA-3 sono legate alla sicurezza e alla lunghezza dell'hash prodotto. SHA-1 è considerato debole e non viene più raccomandato per nuove applicazioni crittografiche. SHA-2 è generalmente sicuro e ampiamente utilizzato. SHA-3 è un nuovo algoritmo che offre sicurezza avanzata e ha una struttura diversa rispetto a SHA-1 e SHA-2.

È importante notare che l'aumento della lunghezza dell'hash, come passare da SHA-256 a SHA-512, aumenta la resistenza agli attacchi crittografici, ma richiede anche maggiori risorse computazionali. La scelta dell'algoritmo dipende dalle specifiche esigenze di sicurezza e prestazioni del sistema in cui verrà utilizzato.

21.6.1 SHA-512

Ecco una spiegazione generale di come funziona SHA-512:

Preparazione del messaggio: Il messaggio da hashare viene suddiviso in blocchi di 1024 bit. Se il messaggio non è un multiplo di 1024 bit, vengono applicati dei padding per raggiungere la lunghezza appropriata.

Inizializzazione degli stati: SHA-512 utilizza una serie di costanti predefinite per inizializzare gli stati interni dell'algoritmo. Questi stati sono matrici di 64 bit.

Elaborazione dei blocchi: SHA-512 elabora i blocchi del messaggio uno per uno. Per ogni blocco, vengono eseguite una serie di operazioni che coinvolgono permutazioni, rotazioni, operazioni logiche e operazioni modulari.

Aggiornamento degli stati: Man mano che ogni blocco viene elaborato, gli stati interni vengono aggiornati in base ai risultati delle operazioni eseguite. Questo processo continua fino a quando tutti i blocchi del messaggio sono stati processati.

Generazione dell'hash: Una volta che tutti i blocchi sono stati elaborati, gli stati interni vengono combinati per generare l'hash finale di 512 bit. Questo hash rappresenta in modo univoco il messaggio originale.

SHA-512 è progettato per essere resistente a diversi tipi di attacchi crittografici, come le collisioni (trovare due messaggi diversi che generano lo stesso hash) e le preimmagini (trovare un messaggio che

produce uno specifico hash). La sua lunghezza di 512 bit lo rende molto resistente alle collisioni brute force.

L'algoritmo SHA-512 è ampiamente utilizzato in applicazioni di sicurezza e crittografia, come crittografia di dati sensibili, autenticazione di password e firma digitale. È considerato uno degli algoritmi di hash crittografici più sicuri al momento della sua introduzione.

21.7 SHA-3

SHA-2 condivide la stessa struttura e le stesse operazioni matematiche dei suoi predecessori (come SHA-1 e SHA-256), il che ha sollevato alcune preoccupazioni sulla sicurezza dell'algoritmo.

A causa del tempo richiesto per sostituire completamente SHA-2 nel caso diventasse vulnerabile, il National Institute of Standards and Technology (NIST) degli Stati Uniti ha annunciato nel 2007 una competizione per produrre SHA-3, una nuova variante dell'algoritmo di hash.

Sono stati stabiliti dei requisiti per SHA-3. L'algoritmo deve supportare lunghezze di hash di 224, 256, 384 e 512 bit. Inoltre, l'algoritmo deve elaborare piccoli blocchi di dati alla volta, anziché richiedere l'intero messaggio da memorizzare in memoria prima di elaborarlo. Questo approccio consente un'elaborazione più efficiente e flessibile dei messaggi di grandi dimensioni.

In sintesi, SHA-3 è stato introdotto per affrontare alcune preoccupazioni sulla sicurezza di SHA-2 e per fornire una migliore efficienza nell'elaborazione di messaggi di grandi dimensioni. È stato selezionato attraverso una competizione indetta dal NIST e soddisfa i requisiti specificati, inclusa il supporto per diverse lunghezze di hash e l'elaborazione di blocchi di dati in modo incrementale.

22 HMAC

HMAC, acronimo di "Hash-based Message Authentication Code" (Codice di Autenticazione del Messaggio basato su Hash), è una costruzione crittografica utilizzata per garantire l'integrità e l'autenticità dei dati trasmessi su una rete. Funziona utilizzando una funzione di hash crittografica, come SHA-1 o SHA-256, in combinazione con una chiave segreta.

A differenza delle semplici funzioni di hash crittografiche, che producono solo un hash del messaggio, HMAC aggiunge un ulteriore passaggio per rendere il codice di autenticazione dipendente anche dalla chiave segreta. Ciò fornisce un meccanismo per verificare l'autenticità dei dati utilizzando una chiave condivisa tra il mittente e il destinatario.

HMAC è stato scelto come il MAC obbligatorio per l'implementazione nella sicurezza IP (Internet Protocol) per diversi motivi. Innanzitutto, HMAC è stato progettato appositamente come un codice di autenticazione del messaggio, il che significa che è stato progettato per garantire l'integrità e l'autenticità dei dati. Inoltre, HMAC utilizza una chiave segreta per aggiungere un livello di protezione aggiuntivo.

Inoltre, HMAC è ampiamente utilizzato in altri protocolli Internet, come il protocollo di sicurezza dei trasporti (TLS) e la transazione elettronica sicura (SET). Ciò significa che esistono librerie e implementazioni di HMAC ampiamente disponibili, semplificando l'implementazione e l'interoperabilità tra diversi sistemi.
SHA-1 non è stato concepito per essere utilizzato come MAC perché non si basa su una chiave segreta!!!!!

22.1 Obiettivi di progettazione di HMAC

-Utilizzare senza modifiche le funzioni hash disponibili che funzionano e il cui codice è libero.
-Consentire sostituzione fscile della funzione hash nel caso se ne trovi una migliore.
-Preservare le prestazioni originali della funzione hash senza aumentare troppo il degrado.
-Utilizzare e gestire le chiavi in modo semplice.
HMAC tratta la funzione hash come una scatola nera: codice preconfezionato pronto all'uso senza modifiche,facile da sostituire,se la sicurezza della funzione hash fosse compromessa rischierebbe anche tutto l'HMAC.
"Based on reasonable assumptions on the embedded hash function" significa che l'analisi crittografica della robustezza del meccanismo di autenticazione si basa su presupposti ragionevoli sulla funzione di hash incorporata.

22.2 Algoritmo HMAC

L'algoritmo HMAC combina due blocchi derivati dalla chiave segreta K (Si e So) con il messaggio M e applica la funzione di hash H due volte per generare il codice di autenticazione del messaggio.

Questo processo di combinazione e applicazione della funzione di hash rende HMAC resistente a diversi tipi di attacchi, come l'attacco a forza bruta o l'attacco di modificazione dei dati.

22.3 Security of HMAC

La sicurezza dipende dalla forza della funzione hash, c'è una forte relazione tra la "forza" di HMAC e quella della funzione hash,la probabilità di successo di un attacco HMAC è equivalente ad un attacco riuscito alla funzione hash in uno dei seguenti attacchi: L'attaccante è in grado di calcolare un output della funzione di compressione anche con un IV casuale, segreto e sconosciuto all'attaccante, L'attaccante trova collisioni nella funzione hash anche quando l'IV è casuale e segreto.

23 Crittografia Autenticata(AE)

L'Encryption Autenticata (AE) è un sistema crittografico che fornisce sia la confidenzialità (cifratura) che l'autenticità (integrità) dei dati. Garantisce che i dati criptati rimangano privati e non possano essere manomessi.

AE viene tipicamente implementato utilizzando una modalità di operazione di cifratura a blocchi. Un esempio di tecnica di encryption autenticata è OCB AE (Offset Codebook Mode with Authentication), che è stato approvato come standard nella rete LAN wireless IEEE 802.11 ed è incluso in MiniSec, un modulo di sicurezza open-source per l'IoT.

OCB è simile alla modalità Electronic Codebook (ECB), ma affronta una delle debolezze di ECB. In ECB, se lo stesso blocco di testo in chiaro viene cifrato con la stessa chiave, produce lo stesso blocco di testo cifrato, il che non è sicuro per messaggi di lunghezza considerevole. OCB introduce una variabile chiamata "Z" e ogni blocco di testo in chiaro viene XORato con un valore diverso di "Z".

In OCB, viene utilizzato l'algoritmo di cifratura AES. Il messaggio in chiaro viene diviso in blocchi di n-bit (n=128) e la sua lunghezza non deve superare n-bit. Il numero totale di blocchi dipende dalla lunghezza del messaggio e viene calcolato come m = ceil(len(M)/n), dove "ceil" indica l'arrotondamento per eccesso.

Viene scelto un valore "N" arbitrario di n-bit chiamato "nonce". Se vengono crittografati messaggi multipli con la stessa chiave, è necessario utilizzare un nonce diverso ogni volta in modo che ciascun nonce venga utilizzato solo una volta. Ogni diverso valore di "N" produrrà un set diverso di "Z".

Nel processo di cifratura, vengono calcolati i valori di "L" e "R" utilizzando l'algoritmo AES. Successivamente, vengono calcolati i valori di "Z" per ciascun blocco di testo in chiaro in base ai valori precedenti di "Z" e "L". Infine, viene generato un codice di autenticazione di lunghezza "tau" e un tag finale, che determina il livello di autenticazione, con un tag più lungo che garantisce una maggiore sicurezza.

24 Crittografia Asimmetrica: Asymmetric Encryption:RSA

L'Encryption RSA a chiave pubblica (Rivest, Shamir e Adleman) è un algoritmo crittografico sviluppato da Rivest, Shamir e Adleman presso il MIT nel 1977.

Nell'algoritmo RSA, il testo in chiaro e il testo cifrato vengono trattati come interi compresi tra 0 e n-1, dove n è un numero primo.

L'algoritmo fa uso dell'esponenziazione degli interi modulo un numero primo.

Per cifrare un messaggio M, si utilizza la formula C = M-alla-e mod n, dove C rappresenta il testo cifrato.

Per decifrare il messaggio cifrato C, si utilizza la formula M = C-alla-d mod n, dove M rappresenta il testo in chiaro. Questa formula è ottenuta calcolando M-alla-e-alla-d mod n.

Sia il mittente che il destinatario conoscono i valori di n ed e. Solo il destinatario conosce il valore di d.

L'algoritmo di cifratura a chiave pubblica RSA utilizza una chiave pubblica PU = e, n e una chiave privata PR = d, n.

La chiave pubblica PU viene utilizzata dal mittente per cifrare il messaggio, mentre la chiave privata PR viene utilizzata dal destinatario per decifrare il messaggio.

La sicurezza dell'algoritmo RSA si basa sulla difficoltà di fattorizzare il numero n in fattori primi, che è un problema computazionalmente complesso.
Nella crittografia RSA, è necessario che n sia un numero prodotto di due numeri primi distinti, solitamente indicati come p e q. Entrambi p e q devono essere numeri primi per garantire la sicurezza del sistema.

La scelta di numeri primi grandi e distinti come p e q è fondamentale per garantire la sicurezza dell'algoritmo RSA. Se n non fosse il prodotto di due numeri primi, sarebbe più facile per un attaccante scomporre n in fattori primi e rompere la crittografia.
n deve essere maggiore di M.
Come genero quindi le chiavi?
-seleziono p e q
-n=p*q
-calcolo grado di n=p-1*q-1
-seleziono e compreso tra 1 e grado di n
calcolo d tale d sia coprimo con grado di n
K privata = d,n
K pubblica = e,n
C = M alla e mod n
M = C alla d mod n

24.1 RSA Requisiti

E' possibil trovare valori e,d,n tale che M alla ed mod n = M per ogni M ¡ n
Deve essere facile calcolare Malla e ed C alla d per ogni valore M ¡ n
Non deve essere possibile trovare d conoscendo e ed n.
Le prime due facilmente soddisfatte,la terza soltanto per grandi valori di e ed n.

24.2 Sicurezza di RSA

I quattro possibili approcci all'attacco dell'algoritmo RSA sono i seguenti
-Brute force: provo tutte le chiavi private
-Attacchi matematici: Diversi approcci, tutti equivalenti alla fattorizzazione del prodotto di due numeri primi
-Timing attacks: Dipendono dal tempo di esecuzione dell'algoritmo di decrifrazione
-Attacchi al testo cifrato scelto: Sfrutta le proprietà dell'algoritmo RSA

24.3 Attacchi matematici a RSA

Si basa sulla fattorizzazione in primi
La fattorizzazione è un problema difficile, ma non così difficile!!! - Fino al 1994 l'attacco utilizzava l'approccio noto come setaccio quadratico - Dal 1999 è stato utilizzato un nuovo algoritmo di setaccio quadratico: Setaccio a campi numerici generalizzati (solo il 20sforzo di calcolo necessario prima) - Raccomandazioni: - Dimensione della chiave 1024 - 2048 - Selezione di p e q per rendere la fattorizzazione più complicata.
La lunghezza di p e q dovrebbe differire di poche cifre,con chave di 1024 bit sia p che q dovrebbero essere dell'ordine di grandezza di 1075/10100
Sia p-1 che q-1 devono contenere un fattore primo grande
mcd(p-1,q-1) deve essere piccolo
con e min n e d min n*1/4 allora d è facilmente dimostrabile.

24.4 Timing Attacks

Un attaccante può determinare una chiave privata tenendo traccia del tempo impiegato da un computer per decifrare i messaggi.
- Gli attacchi temporali sono applicabili non solo a RSA, ma anche ad altri sistemi di crittografia a chiave pubblica.
- Questo attacco è allarmante per due motivi:
-proviene da una direzione del tutto inaspettata
-Si tratta di un attacco al solo testo cifrato.
Il sistema di destinazione utilizza una funzione di moltiplicazione modulare che è molto veloce in quasi tutti i casi ma che in alcuni richiede molto tempo,l'attacco procede bit per bit a partire dal bit più a sinistra,se l'algoritmo de decriptazione è sempre lento quando un'iterazione è lenta con un bit settato ad 1 allora si presume che questo bit sia ad 1, se invece sono veloci si presume sia 0.
Nella pratica, le implementazioni di esponenziazione modulare non hanno variazioni di tempo così estreme, in cui il tempo di esecuzione di una singola iterazione può superare il tempo medio di esecuzione dell'intero algoritmo. Tuttavia, esiste abbastanza variazione per rendere questo attacco praticabile.

24.4.1 Contromisure

- Tempo di esponenziazione costante
  Assicurarsi che tutte le esponenziazione richiedano lo stesso tempo prima di restituire un risultato
  - Si tratta di una soluzione semplice, ma degrada le prestazioni

- Ritardo casuale
  - Una prestazione migliore potrebbe essere ottenute aggiungendo un ritardo casuale all'algoritmo di all'algoritmo di esponenziazione per confondere l'attacco di temporizzazione - Se i difensori non aggiungono rumore sufficiente, gli attaccanti potrebbero comunque avere successo raccogliendo ulteriori misurazioni aggiuntive per compensare i ritardi casuali

- Accecamento
  - Moltiplicare il testo cifrato per un numero casuale prima di eseguire l'esponenziazione - Questo processo impedisce all'attaccante di attaccante di sapere quali bit del testo cifrato che vengono elaborati all'interno del computer e quindi impedisce l'analisi bit per bit essenziale per l'attacco di temporizzazione

24.5 Scambio di chiavi:DIFFIE HELLMAN

La Diffie-Hellman Key Exchange (scambio di chiavi Diffie-Hellman) è un algoritmo di crittografia a chiave pubblica che consente a due parti di stabilire in modo sicuro una chiave segreta condivisa che può essere utilizzata successivamente per crittografare i messaggi.

È stato il primo algoritmo di crittografia a chiave pubblica pubblicato, introdotto da Diffie e Hellman nel 1976 insieme all'esposizione dei concetti di chiave pubblica. Da allora, è stato ampiamente utilizzato in numerosi prodotti commerciali e protocollari di sicurezza.

Il funzionamento del protocollo di scambio di chiavi Diffie-Hellman si basa sulla difficoltà di calcolare logaritmi discreti. L'idea di base è che due parti, chiamate Alice e Bob, condividono pubblicamente un insieme di parametri comuni: un numero primo p e un generatore g di un sottoinsieme ciclico del gruppo dei residui modulo p.

Successivamente, Alice e Bob scelgono segretamente due numeri, rispettivamente a e b, e calcolano i corrispondenti valori A e B utilizzando la formula A = ga mod p e B = gb mod p. A e B sono quindi scambiati pubblicamente tra Alice e Bob.

Infine, Alice e Bob possono calcolare la stessa chiave segreta condivisa K utilizzando i valori scambiati e i loro segreti individuali. Alice calcola K = Ba mod p e Bob calcola K = Ab mod p. Poiché l'operazione di elevamento a potenza modulare è computazionalmente difficile da invertire, solo Alice e Bob conosceranno il valore di K e potranno utilizzarlo per crittografare e decrittografare i messaggi successivi.

La sicurezza del protocollo Diffie-Hellman si basa sulla difficoltà computazionale di calcolare logaritmi discreti in un campo finito. L'attacco più conosciuto contro Diffie-Hellman è l'attacco del logaritmo

discreto, che cerca di invertire l'operazione di elevamento a potenza modulare per determinare i segreti a e b.

Tuttavia, la sicurezza del protocollo può essere migliorata utilizzando gruppi ciclici di grandi dimensioni, come i gruppi di numeri primi di lunghezza sufficientemente elevata, che rendono computazionalmente impraticabile l'attacco del logaritmo discreto.
- La sicurezza dello scambio di chiavi di Diffie-Hellman risiede nel fatto che, mentre è relativamente facile calcolare gli esponenziali modulo primo, è molto difficile calcolare i logaritmi discreti. logaritmi discreti. Per i primi grandi, quest'ultimo compito è considerato inaffrontabile.
p e g sono noti a tutti.

24.6 Man in the Middle Attack(MITM)

Nel contesto descritto, si tratta di un attacco di tipo man-in-the-middle (MITM) contro il protocollo di scambio di chiavi Diffie-Hellman. L'attaccante, chiamato Darth, si interpone tra Alice e Bob per intercettare e manipolare le comunicazioni.

L'attacco procede come segue:

Darth genera le sue chiavi private XD1 e XD2, insieme alle rispettive chiavi pubbliche YD1 e YD2. Alice trasmette a Bob la sua chiave pubblica YA. Darth intercetta YA e la sostituisce con la sua chiave pubblica YD1, inviandola a Bob. Darth calcola anche la chiave K2. Bob riceve YD1 e calcola la chiave K1. Bob trasmette ad Alice la sua chiave pubblica XA. Darth intercetta XA e lo sostituisce con la sua chiave pubblica YD2, inviandola ad Alice. Darth calcola anche la chiave K1. Alice riceve YD2 e calcola la chiave K2.

A questo punto, Bob crede di condividere una chiave segreta con Alice, ma in realtà Bob condivide una chiave segreta (K1) con Darth, mentre Alice condivide una chiave segreta (K2) con Darth. Tutte le comunicazioni future tra Bob e Alice sono compromesse.

In seguito, Darth può intercettare i messaggi cifrati inviati da Alice, decifrarli utilizzando la chiave segreta K2 e leggere il contenuto del messaggio originale (M). Darth può anche inviare messaggi cifrati a Bob utilizzando la chiave segreta K1, permettendogli di eavesdrop (ascoltare di nascosto) sulla comunicazione o addirittura modificare il messaggio prima che Bob lo riceva.

Questo attacco è possibile perché il protocollo di scambio di chiavi Diffie-Hellman non autentica i partecipanti. Darth riesce a sostituire le chiavi pubbliche di Alice e Bob con le sue chiavi pubbliche manipolate, permettendogli di stabilire segretamente le chiavi con entrambi i partecipanti.

Per superare questa vulnerabilità, il protocollo di scambio di chiavi Diffie-Hellman può essere potenziato con l'uso di firme digitali e certificati di chiavi pubbliche. Le firme digitali consentono di autenticare le chiavi pubbliche dei partecipanti, garantendo che provengano da fonti legittime. L'uso di certificati di chiavi pubbliche può fornire una struttura di trust per verificare l'autenticità delle chiavi pubbliche scambiate tra i partecipanti. Queste misure aggiuntive prevengono l'attacco MITM e garantiscono l'integrità e l'autenticità delle comunicazioni.