

SICUREZZA DEI SISTEMI ICT

Appunti di Federico Landini

Professore: Fabrizio Baiardi, fabrizio.baiardi@unipi.it

"Libro": Security engineerin, ross anderson

If it's smart, it's vulnerable

Sistemi informatici e proprietà di sicurezza

Un sistema informatico è, ad ogni livello, formato da un insieme di moduli connessi ognuno dei quali offre un certo numero di operazioni, che permettono di leggere e manipolare informazioni e hanno un impatto sul mondo esterno per gestire il tutto ci sono **regole** (politica di sicurezza) che definisce chi può invocare una operazione e che ha il diritto di leggere e scrivere queste operazioni.

3 principali proprietà:

- confidenzialità: diritto di leggere
- integrità: diritto di aggiornare
- disponibilità: avere il diritto e si vuole farlo in un tempo finito.

Proprietà di sicurezza derivate

- Tracciabilità: chi ha invocato una operazione
- Accountability: addebitare l'uso delle risorse
- Auditability: verificare l'efficacia dei meccanismi di enforcement di una politica (come viene realizzata)
- Forensics: provare che certe azioni hanno avuto luogo
- Privacy/GDPR: chi/come/se può usare informazioni personali

Politica di sicurezza

Un insieme di regole definite dal proprietario del sistema o del processo per decidere, gli utenti che possono invocare una operazione e quando possono farlo, esistono diverse categorie di politiche da scelte indipendenti.

- Prima scelta = (come si descrive)
 - default allow = la politica definisce le operazioni vietate
 - default deny = la politica definisce le operazionimesse
- Seconda scelta = (vincoli sui proprietari del sistema)
 - Discretionary access control = decide il proprietario, tipica politica del mondo commerciale.
 - Mandatory access control = esistono vincoli globali a tutto il sistema che nemmeno il proprietario può violare, tipica del mondo militare.

Le 6 idee più studiate della sicurezza (max ranum)

- Default Allow
- Enumerating Badness
- Penetrate and Patch
- Hacking is Cool
- Educating Users
- Action is Better Than Inaction

Prima scelta

Default Allow = Enumerationg Badness



Seconda scelta: Soggetti & oggetti

I soggetti invocano le operazioni definite dagli oggetti e se un oggetto invoca le operazioni di altri oggetti allora diventa esso stesso un oggetto.

Dipende dal livello di implementazione per essere oggetto/soggetto.

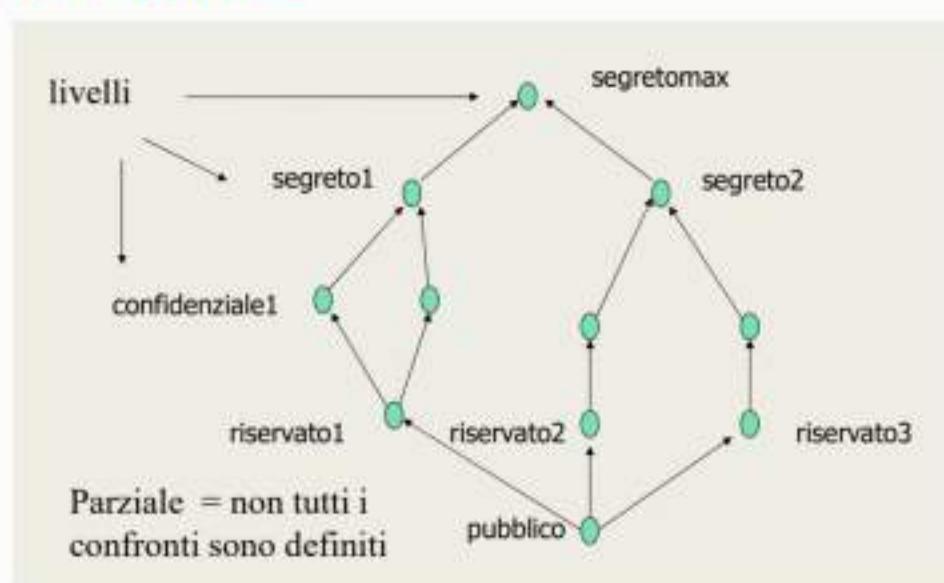
Discretionary Access Control

Per ogni oggetto esiste un proprietario (dal sistema informativo o processo aziendale che lo usa), il proprietario decide i diritti dei soggetti, non ci sono vincoli sul proprietario, per il mondo industriale.

Mandatory Access Control

- Tutti gli utenti (soggetti) sono divisi in classi
- Tutte le risorse (oggetti) divide dalle stesse classi (per semplicità)
- Le classi sono ordinate parzialmente
- Livello di un soggetto = quanto ci fidiamo
- Livello di un oggetto = quanto è importante

Ordinamento Parziale



Politica MAC

- Soggetto = utente o programma dell'utente
- Oggetto = risorsa = File
- Operazioni = read/write/append
- Diritti Utente:
 - Leggere tutti i file che hanno classe minore o uguale alla sua
 - scrivere i record dei file che hanno classe uguale alla sua
 - appendere un record ad un file che ha classe maggiore della sua.
 - permesso owner per restringere il dominio

No write up, privilegia l'integrità

Chinese Wall

Gli oggetti del sistema sono partizionati in sottoinsiemi, l'utente che opera su un oggetto di quello insieme, non opera su altri insiemi.

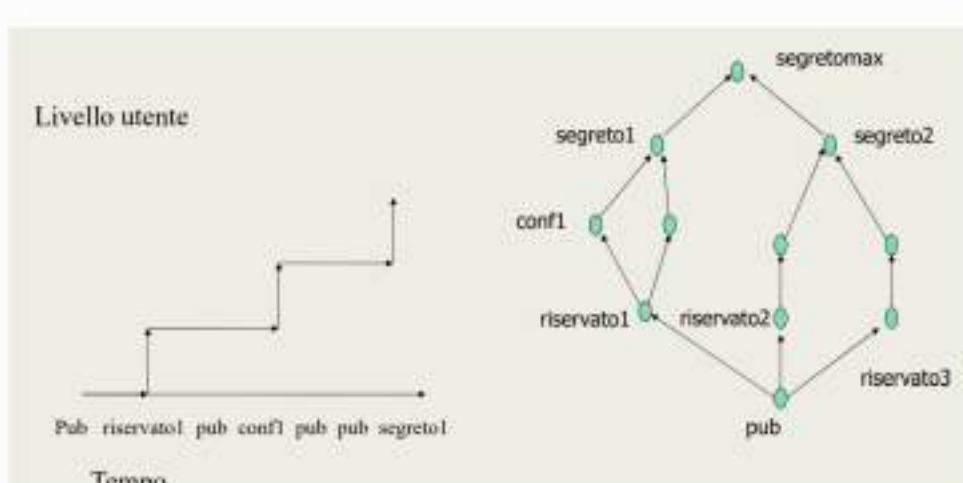
Gestisce i conflitti di interesse.

E' dipendente dal tempo = politica dinamica.
compatibile con MAC perché aggiunge vincoli.

Watermark

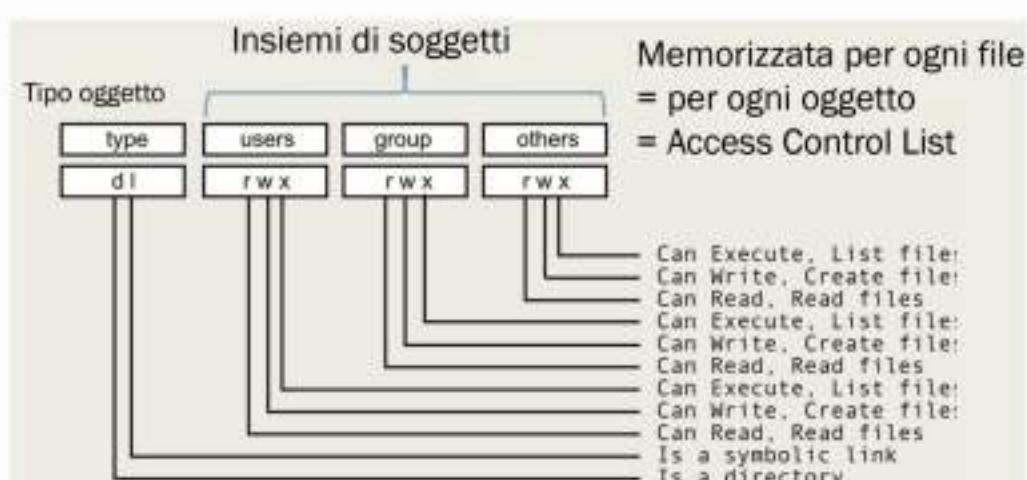
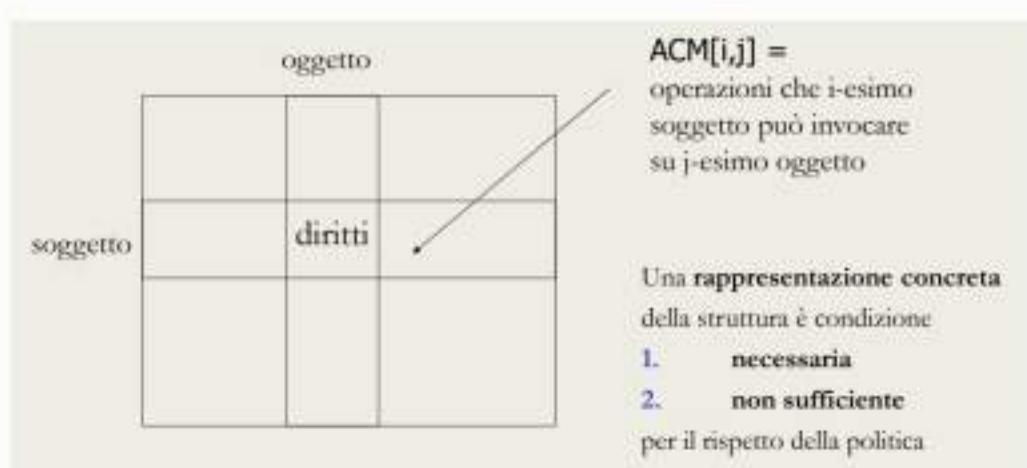
Adotta la nozione di livello ma varia di un range di "fiducia", tipo,
Utente 'x' legge oggetti di livello avanzato, non diminuisce se dopo
legge oggetti di livello inferiore

Dipende dal tempo



Politica Complessiva

Può combinare politiche diverse, deciso il confronto tra livelli si fa nascere dei vincoli per l'integrità.



Matrice di controllo degli accessi

Trusted computing base (TCB)

Esiste una parte della SI che deve garantire il rispetto della politica di sicurezza del SI.

Il TCB comprende le parti SI da realizzare.

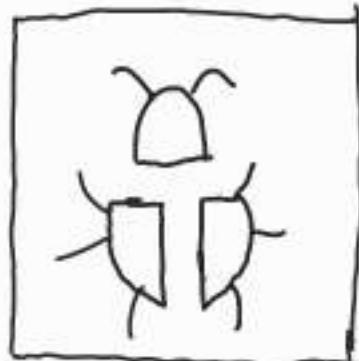
Se c'è un **errore** allora la politica non **è corretta** ed è diversa da errore funzionale.

Minore è il numero di componenti TCB, migliore è la situazione dal punto di vista della sicurezza. Ed è più facile dimostrare la correttezza del TCB.

Vulnerabilità

Un difetto (bug, errore ecc.)

- hardware
- software
- persone
- regole della politica



Violare la sicurezza = Un soggetto riesce ad eseguire una operazione per cui non ha diritti.

Tutte le vulnerabilità sono difetti, non tutti i difetti sono vulnerabili.

Vulnerabilità più famose: valori in ingresso o comportamenti utenti.

Quando l'attacante raggiunge l'obiettivo subentra al proprietario (owner) del sistema e allora può:

- Raccogliere informazioni c
- Modificare informazioni i
- Impedire ad altri di accedere alle informazioni a

Approcci alla Sicurezza

2 approcci:

- Sicurezza incondizionale
 1. Si assume che qualsiasi siano le vulnerabilità esista qualcuno interessato e in grado di violarlo.
 2. Le vulnerabilità vanno eliminate
- Sicurezza condizionale = "analisi e gestione del rischio"
 1. Analisi delle minacce
 2. Eliminare solo le vulnerabilità per le minacce reali. (intrusione)

Analisi e gestione del rischio

1. Analisi delle risorse da proteggere
2. Analisi delle minacce = sicurezza incondizionale
3. Analisi delle vulnerabilità
4. Analisi degli attacchi
5. Analisi degli impatti
6. individuazione rischio, tra accettare e contromisure

1. Analisi delle risorse da proteggere

Si individua un insieme di oggetti (risorse, fisiche, logiche ecc.) e alcune proprietà della sicurezza, si definisce una politica di sicurezza su questi oggetti (che legge, scrive, esegue ecc.)

Queste saranno gli obiettivi di sicurezza che dobbiamo garantire.

2. Analisi delle minacce

Capire chi vuole attaccare il sistema.

(rubare, modificare e utilizzo)

Possibili minacce = attaccanti; posso essere intelligenti o eventi naturali.

Safety vs Security

Safety = capacità di resistere ad eventi di origine non umana e casuale.

Security = capacità di resistere ad attacchi di origine intelligente che fanno parte di un piano malizioso per raggiungere un obiettivo.

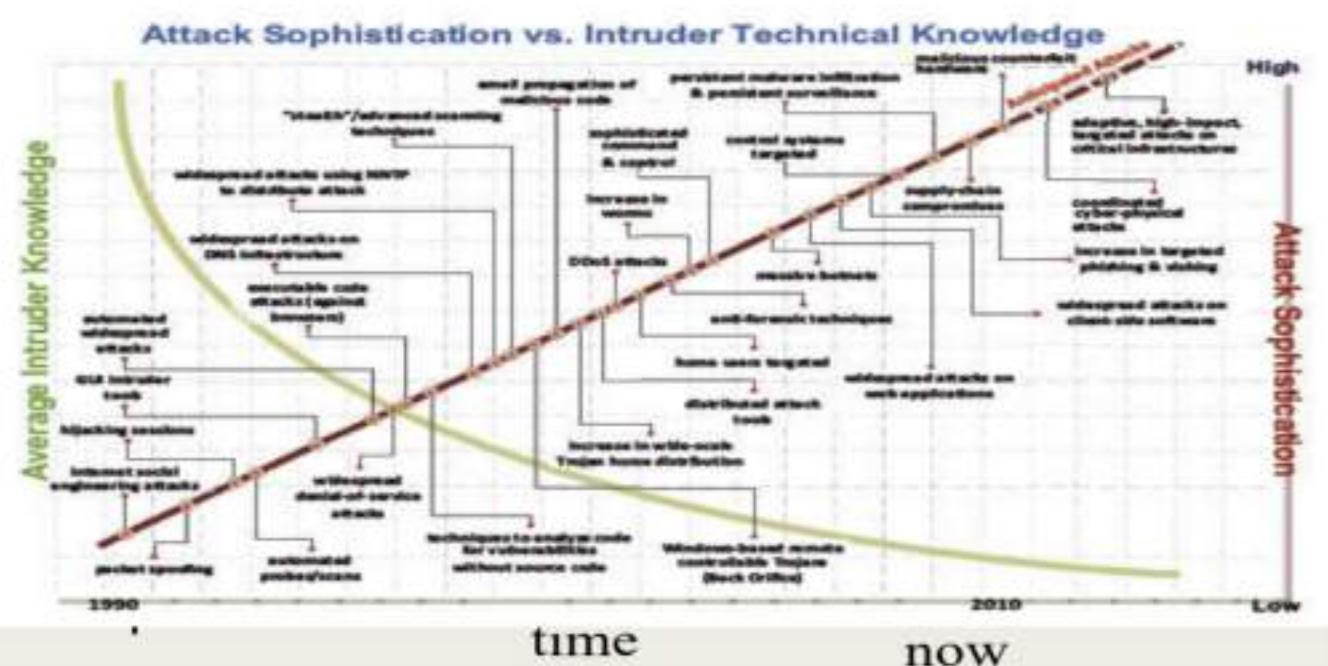
3. Analisi delle vulnerabilità

Le vulnerabilità del sistema che permette all'attaccante di ottenere in passi le risorse di sistema.

Si scoprono in modo automatico o manuale oppure da locale o remoto.

4. Analisi degli attacchi

- diritti richiesti/ottenuti
 - competenze/abilità richieste
 - rumore generato
 - automatizzato/..bile/non auto..



- locale/remoto

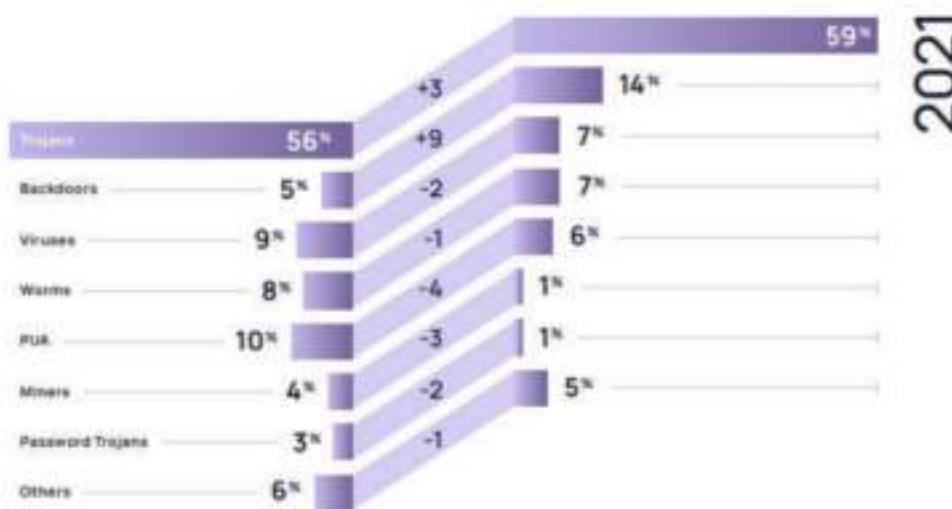
Attacco locale/remoto

Ha bersagliato un sottosistema (un nodo di una rete) purché si disponga di un account sul sottosistema e si dice Locale, senza Account si chiama Remoto.

(automatizzabile & remoto) più pericolo perché fatto da chiunque e da qualsiasi elemento in rete (remoto)

Frequency of different types of malware in 2020 vs. 2021

2020



2021

PUA o PUP = potentially unwanted application / program

Malware

Qualsiasi software fatto per creare danni.

Esempi:

- Spyware: raccoglie informazioni senza il consenso e viene installato tramite il click di un link (phishing) oppure fa parte del payload di un worm
- Trojan horses e backdoors: programmi che, sotto false spoglie di software "pacifici", si introducono nel sistema e svolgono funzioni dannose, controllando la macchina.

Esempi:

- Ransomware: rendono non utilizzabile i dati memorizzati su un sistema.
i dati vengono cifrati e avviene un riscatto.
- Password Stealing: poco usati per rubare password.
- Stepping Stones: Utilizzano per lanciare altri attacchi evitando di essere scoperto.
- Back Door RAT: Tool di amministrazione remota, permette di gestire un sistema tramite protocolli, nati per facilitare il lavoro di chi deve lavorare su macchine remote, controllando il sistema infetto.

Ransomware Effetti e Trend

The rising cost of cyber insurance
Change in US premiums (quarter-on-quarter %)

Axa ha annunciato che non
rimborserà danni per ransomware



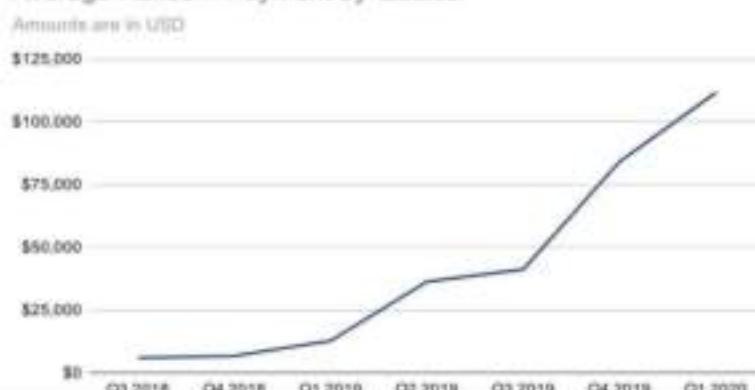
RANSOMWARE DEMANDS OVER THE YEARS

30 TIMES
GROWTH

The average ransomware demand has increased 30 times since 2012, according to a new report from the FBI. The FBI's latest Internet Crime Report shows that the average ransomware demand increased from \$1,000 in 2012 to \$30,000 in 2020.



Average Ransom Payment by Quarter



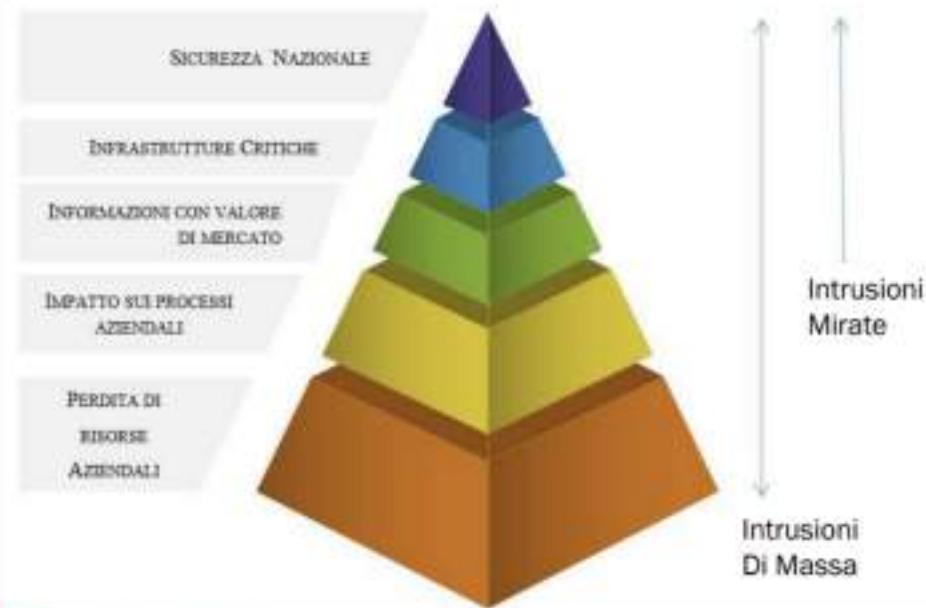
Average ransom demands compared to average ransom payments in 2020 and 2021, according to Unit 42 incident response data



- Worm: composto da 2 parti:
 - Attack vector: codice per attaccare altri nodi di una rete, attacco remoto e crea una copia worm sul nodo attaccato.
 - Payload: codice eseguito su tutti i nodi attaccati. ransoware/spyware/miner sono possibili payload.
- Virus: "worm passivo" non è in grado di diffondersi da solo, richiede un meccanismo di trasporto per passare da un nodo a un'altro o dalla posta elettronica, chiavi usb o usato per sistemi air gapped.

Intrusione mirata/non mirata

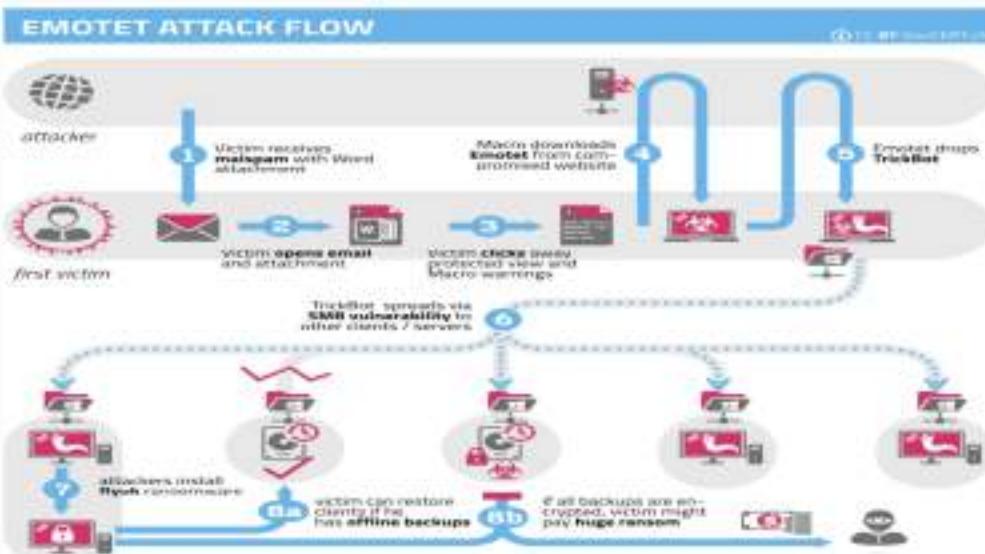
Mirata, stabilisce un obiettivo e lo persegue fino a raggiungerlo da considerare che dovrà (attaccante) minimizzare il rumore delle sue azioni (effetti sul target di sistema) e anche adattivo, cioè, cambia comportamento in base al target.
non mirata, è un worm su internet da un modo iniziale.



Social Engineering

Si studia come influenzare il comportamento individuale di un utente per capire informazioni su di lui.

Spears phishing = attacco mirato verso un utente dopo aver recuperato informazioni per l'attacco.



5. Analisi Degli Impatti

Si calcolano il danno di un attaccante ed è un insieme di costi:

- Ripristino di uno stato corretto
- Ricostruzione informazioni perse
- Perdita di produzione
- Informare i possessori dei dati (privacy)
- Danno di immagine.

Di solito i danni si dividono in: (aziende)

- 39% business disruption
- 35% perdita di informazioni
- 21% danno reputazionale

6. Contromisure

Modifiche ai componenti utilizzati.

Si aggiungono componenti per:

- Eliminare le vulnerabilità
- Filtrare le connessioni (firewall)
- Individuare il malware (endPoint protection)
- Scoprire nuove intrusioni e fermarle

Coordinamento di vari strumenti, si opera in modo:

- Statico: si cambia il sistema
- Dinami: si scambia il sistema quando è in corso un'intrusione
- Contromisura dinamiche (monitoraggio di un sistema)

Rischi

contratto tra rischio = danno medio di intrusione e costo.

contromisure = il rischio limita il costo delle contromisure

Il rischio dipende da 2 fattori:

- successo di intrusione
- danno di una intrusione

se il rischio è alto si riduce:

- Trasferendo a costo fisso = assicurazione
- modificando il sistema
- riducendo danno di intrusione es: database
- abbandonando il sistema

Visioni sulla sicurezza

- Sicurezza = confidenzialità (<-> crittografia) per leggere occorre la chiave di cifratura per conservare correttamente l'informazione.
- Per risolvere alcuni problemi sulla sicurezza occorre lavorare sulla tripla (posizione della matrice ACM)
<Utente,Risorsa,Diritti utente su risorsa>
- Ogni decisione richiede la risoluzione di 3 problemi:
 1. Identificazione utente -> risolta dalla autenticazione
 2. identificazione risorsa
 3. esame dei diritti

Autenticazione

3 classi di metodi:

- "una che consoci" = password/pin/nome cane...
- "una cosa che hai" = telefono/carta/token
- "una cosa che sei" = caratteristica biometrica tipo impronta digitale, faccia, occhio ecc.

un'autenticazione forte = uso simultaneo di 2 metodi di classi.

diverse = sms quando si accede a home banking.

BIOMETRICS COMPARISON CHART

Biometric	Verify	ID	Accuracy	Reliability	Error Rate	Errors	False Pos.	False Neg.
Fingerprint	✓	✓	99.9%	█████	1 in 500+	dryness, dirt, age	Ext. Diff.	Ext. Diff.
Facial Recognition	✓	X	99.9%	████	no data	lighting, age, glasses, hair	Difficult	Easy
Hand Geometry	✓	X	99.9%	████	1 in 500	hand injury, age	Very Diff.	Medium
Speaker Recognition	✓	X	99.9%	▶	1 in 50	noise, weather, cold	Medium	Easy
iris Scan	✓	✓	99.99%	██████	1 in 131,000	poor lighting	Very Diff.	Very Diff.
Retinal Scan	✓	✓	99.99%	█████	1 in 10,000,000	glasses	Ext. Diff.	Ext. Diff.
Signature Recognition	✓	X	99.9%	▶	1 in 50	changing signatures	Medium	Easy
Keystroke Recognition	✓	X	99.9%	▶	no data	hand injury, tiredness	Difficult	Easy
DNA	✓	✓	99.99%	█████	no data	none	Ext. Diff.	Ext. Diff.

LEGGERE ARTICOLO "NSA - LE MIGLIORI COMPETENZE SU ENCRYPTION" (slide 2 lezione 3)

FINE

INTRODUZIONE E TERMOLOGIA

VULNERABILITA', ATTACCHI E INTRUSIONI

Esempi:

- Vulnerabilità
- Attacco
- Possibili Contromisure

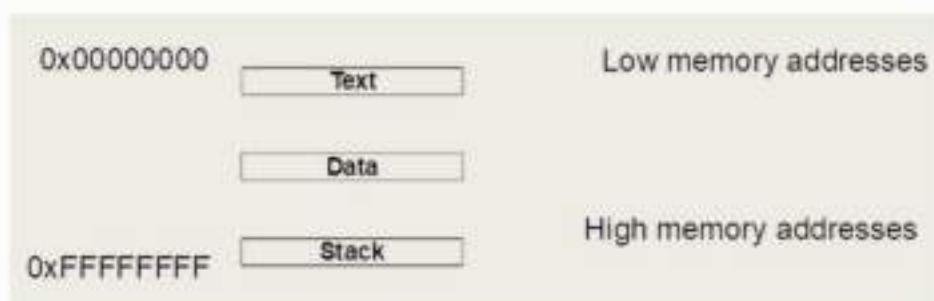
Attacco molto popolare: Stack Overflow

Buffer Overflow

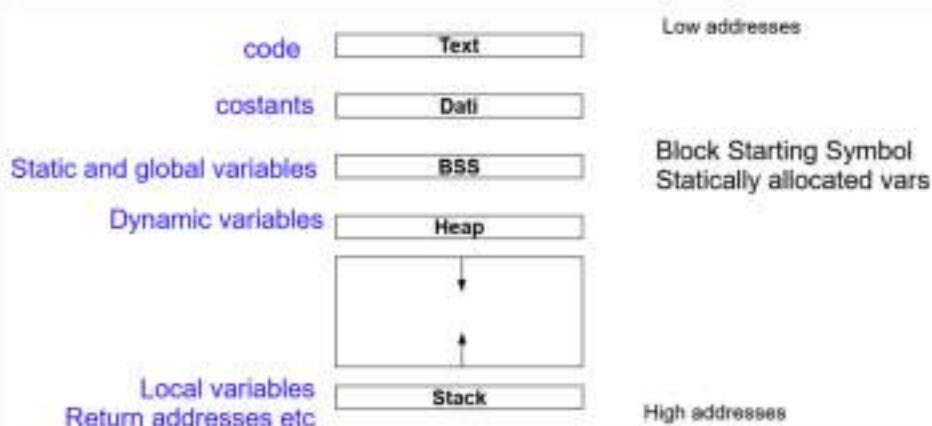
Vulnerabilità più comune nel C, assente in linguaggi di alto livello. Il danno consiste di trasmettere ad una funzione una qualità di informazioni superiore a quella prevista.

Se privo di controlli è possibile trasmettere del codice eseguibile ([Code Injection](#)) ed eseguirlo, ove può garantire il pieno possesso di tutte le funzionalità del sistema ma è un attacco molto complesso ma automatizzabile.

Memoria di un processo

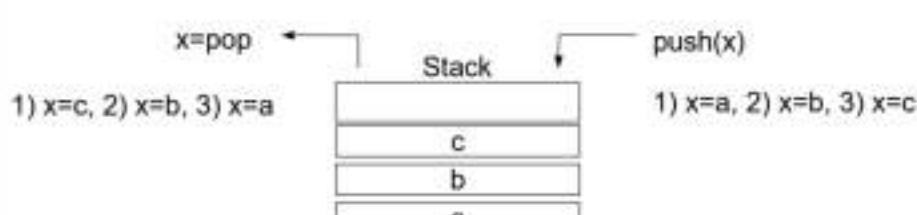


- Il **text segment** ha dimensione fissa, memorizza il codice del programma ed è solo lettura. Ogni scrittura provoca un segmentation error.
- Il **data segment** memorizza variabili statiche e dinamiche.
- Lo **stack segment** memorizza i dati per gestire call e return di funzioni.



Pila/Stack

Si utilizza la politica LIFO (last in first out), la memoria dello stack è parzionata logicamente in record (stack frame, activation frame) uno per ogni call.

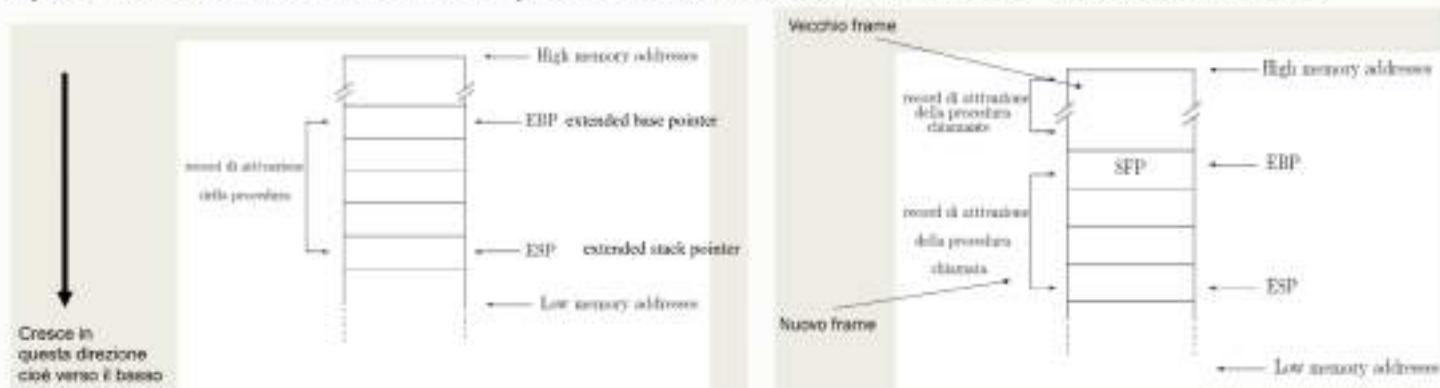


L'indirizzo di memoria della istruzione da eseguire è memorizzato nel registro **EIP** (extended instruction pointer) register, il registro **EBP** (externed base pointer) punta all'inizio dello stack frame mentre il resto **ESP** (extended stack pointer) punta alla fine dello stack frame.

Quando una posizione viene invocata (call) il supporto a tempo d'esecuzione esegue un push sullo stack di

- Indirizzo di ritorno = EIP + 4
- Indirizzo corrente = EBP

quindi coia ESP in EBP per inizializzare il nuovo "stack frame"



Analizziamo quest'esempio di codice in C per capire meglio l'allocazione dello stack frame:

```
void test_function (int a, int b)
{
    char flag;
    char buffer[10];
}

int main()
{
    test_function (1,2);
    exit(0);
}
```

Indirizzo di ritorno (ret) = EIP + 4 byte



'SFP= Saved Frame Pointer, valore utilizzato per rispristinare lo stato originale di EBP (prima della chiamata di test_function());'

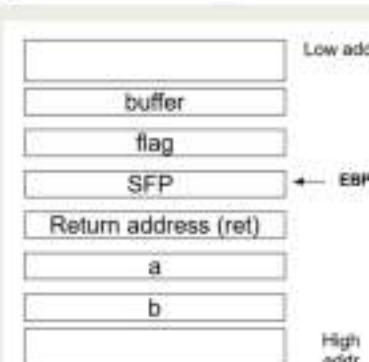
Stack Frame

Le variabili locali in "test_function" sono indirizzate con EBP ed un displacement negativo mentre i parametri sono indirizzati con un displacement positivo.

Così è indipendente da valore dello stack pointer che può cambiare.

Lo stack memorizza sia le variabili locali che i parametri di una funzione quando la funzione termina tutto il frame viene rimosso dallo stack.

```
void test_function (int a, int b)
{
    char flag;
    char buffer[10];
}
```



OverFlow: Example

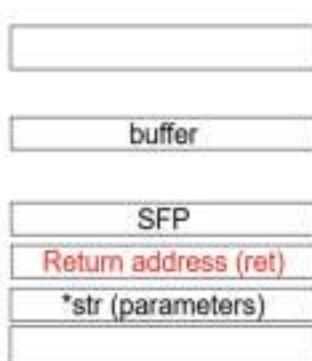
```
void overflow_function (char *str) {  
    char buffer[20];  
  
    strcpy(buffer, str); // Questa funzione copia str in buffer  
}  
  
int main() {  
    char big_string[128];  
    int i;  
  
    for(i=0; i < 128; i++)  
    {  
        big_string[i] = 'A';  
    }  
    overflow_function(big_string);  
    exit(0);  
}
```

Questo produce un overflow

Segmentation Fault

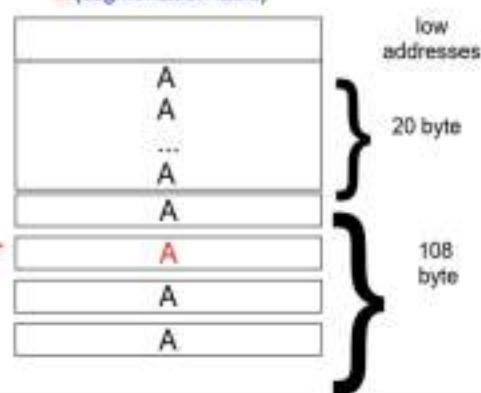
Il codice precedente può provocare un segmentation fault.

- 1) La prima invocazione inizializza correttamente lo stack frame



- 2) Quando l'invocazione termina lo stack è stato riscritto con

A (segmentation fault!)



Buffer Overrun/Overflow

Avviene quando una qualche variabile è più grande di quanto dichiarato o specificato e sovrascrive altre variabili.

4 versioni:

- Stack based buffer overrun
- Heap based buffer overrun
- V-Table and function pointer overrun
- Exception handler overrun

Stack Overflow

1. Copiando una stringa nello stack, noi aggiorniamo e distruggiamo l'indirizzo di ritorno o altri valori nello stack.
2. I valori vengono copiati codificando un programma.
3. Il nuovo valore copiato nel return address punta al codice copiato.
4. Risultato: una shell di amministrazione
5. Si nota che la funzione non ritorna
6. Il risultato è possibile se tutte le volte viene eseguita come root perchP fa parte de SO (sistema operativo)
qui un attacco locale può essere automatizzato.

Vulnerabilità: Punti di vista diversi:

1. Manca il controllo sulla dimensioni delle variabili
 2. cattivo sistema di tipi
 3. operazione di memoria errata
 4. errata dimensione dello stack
- ecc.

Esempio di vulnerabilità strutturali

1. Per controllare se un nodo B è vivo un nodo A può inviare un messaggio di ECHO, a cui B risponde con lo stesso messaggio (messaggio specifica quale nodo invia e quale deve ricevere e rispondere)
 2. E' possibile specificare un indirizzo parziale per ECHO per realizzare comunicazione uno a molti (da un nodo ad un insieme di nodi) e verificare con un solo messaggio se più nodi sono vivi
 3. In IPv4 non esiste un controllo sui campi di un pacchetto IP = nodo X può inviare qualcosa dicendo di essere nodo A
- R è una rete di 1000 nodi, con indirizzo parziale X, un indirizzo parziale comune a tutti i nodi di R
 - Il nodo A manda un messaggio di ECHO ad indirizzo X dicendo di essere il nodo B
 - Tutti i nodi di R rispondono a B
 - B per un certo tempo non può comunicare perché le sue linee di comunicazione sono intasate



Distributed Denial of Service

- L'effetto si può amplificare a piacere aumentando il numero di nodi che hanno il ruolo di A (zombies)
- Questo è un tipico attacco da cui è estremamente difficile difendersi perché B non ha un modo di scoprire che l'attacco è in preparazione

Contromisure

Per DDOS unica soluzione possibile è autenticazione delle comunicazioni, in breve IPV6 dove ogni pacchetto in rete può essere autenticato identità mittente

- Meccanismo di tipi forte (15-30% di aumento overhead di esecuzione)
- ASLR = generazione random degli indirizzi dei vari segmenti = impedisce di conoscere staticamente i vari indirizzi noti
- Canary = inserzione di un valore random prima dell'indirizzo di ritorno. Il valore viene generato ad ogni invocazione di funzione. Prima di ritornare si verifica che il valore non sia stato cambiato
- Stack non eseguibile = costo ottimo ma non si può usare in Linux. Inoltre esiste Return-Oriented-Programming
- Shadow stack
- ...

Classificazione delle vulnerabilità

Ogni tassonomia ha il proprio obiettivo e prima di utilizzarla dobbiamo capire se i suoi obiettivi sono coerenti con la nostra classificazione o il problema che vogliamo risolvere

Dove avvengono?

- **Procedurale** nelle azioni che sono eseguite = azioni mal definite
- **Organizzative** nelle persone che eseguono le azioni (azioni ben definite ma eseguite male)
- SO, compilatori ecc... **strumenti** HW e SW utilizzati (azioni ben definite ed eseguite ma con cattivi strumenti)

esempi:

- Procedurale
 - Una password comunicata per iscritto in una busta non sigillata
- Organizzative
 - Amministratori diversi e contemporanei per una stessa macchina
 - Compiti assegnati a persone non formate
- Strumenti
 - Una password trasmessa in chiaro per email
 - Mancanza di controlli sugli indici di un vettore

Classificazione delle vulnerabilità negli strumenti

- **Specifiche**
 - Uso di una libreria che comprende più funzioni del necessario.
 - Se qualcuno invoca una funzione di quelle "inutili" possiamo avere problemi di sicurezza.
 - Hardening del codice = rimozione dal codice di tutte quelle parti.
- **Implementazione**
 - Nessun controllo su: input utente, parametri di funzioni/ metodi o salti in una struttura dati.
 - queste vulnerabilità sono molto dipendenti dai tipi di linguaggio e dalla presenza di controlli a tempo.
- **Strutturale**
 - Dovuta alla composizione di moduli che sono corretti isolatamente o errati quando cooperano.
 - problemi visti con lo stack tcp/ip
 - controlli e correttezza da altri moduli.

Patch/Patching

Sono file eseguibili e non codice sorgente ove apporta modifiche o sostituisce gli stessi file binari.

però può essere bisogno di una regressione della versione software dell'applicativo (software regression), però bisogna decidere se il vecchio comportamento è un bug o una feature

ed è lento & costoso...

Occorre un test di regressione che verifica il funzionamento dopo l'implementazione di una nuova funzionalità oppure la risoluzione

di eventuali bug.

Possono accadere funzionamenti inattesi...

per garantire la qualità del software la nuova implementazione non pregiudichi le funzionalità esistenti.

Da notare che per tempo e ambiente bisogna minimizzare il numero di patch.

Alla ricerca delle vulnerabilità

Distingueremo:

- Vulnerabilità note: vulnerabilità pubbliche in moduli noti (tanti).
- Vulnerabilità non pubbliche: vulnerabilità scoperte ma non pubblicate, es: database a pagamento (centinaio).
- Zero Day: vulnerabilità scoperte e non pubblicate e usate da organizzazioni e stati negli attacchi delle loro intrusioni (decine).

Ogni sistema come una composizione di moduli standard, affetti da vulnerabilità note, e specializzati, non lo sono.

l'ordine efficacie di ricerca delle vulnerabilità:

1. vulnerabilità in moduli standard
2. vulnerabilità in moduli specializzati
3. vulnerabilità strutturali nate dalla composizione di moduli specializzati e moduli standard.

Inventario hw e sw

Un passo importante per trovare vulnerabilità in un sistema, è sapere quali sono i moduli che utilizza.

Costruire l'inventario dei moduli non è banale e spesso rappresenta il problema più complesso, perché esistono strumenti informatici in cui unico scopo è costruire un inventario e da considerare che le tutte le "best practices" di sicurezza richiedono inventario.

Vulnerability Scanner

E' uno strumento informatico che permette di:

- Costruire inventario
- Conoscere le vulnerabilità note dei moduli dei vari nodi e restituisce:
 - Moduli di ogni nodo che interagiscono con gli altri moduli
 - Le vulnerabilità di questi moduli

Lo strumento sfrutta il fatto che ogni modulo opera su una particolare porta del nodo, invia pacchetti sulla porta e dall'analisi delle risposte deduce quale modulo stiamo lavorando.

Riconosco il modulo cerco le vulnerabilità note nell'apposito DB.

Fingerprinting

Lo stack TCP/IP usa le porte per la comunicazioni tra i nodi.

Le porte note sono le porte TCP e UDP in 0-1023 e sono assegnati a specifici indirizzi.

Un vulnerability scanning attivo invia sulle porte note dei pacchetti che vioano i protocolli.

Così impara prima le risposte a richieste non standard e si identificano i moduli che implementano un servizio su una porta

Fingerprinting passivo

Opera senza interferire con il sistema da analizzare, raccoglie ed analizza pacchetti di rete in maniera "trasparente" e in base di alcune caratteristiche dei pacchetti si deduce i vari moduli che li hanno prodotti.

vulnerabilità

es: apache web

lo scanner accede ad un database che associa ad ogni modulo le vulnerabilità note o anche eventuali patch, però lo scanner non è in grado di determinare se una patch è stata applicata e questo genera dei **falsi positivi** = vulnerabilità segnalata ma che non esiste.

falsi negativi = vulnerabilità non segnalata (non appartiene al DB usato) ma che esiste.

Breach & Simulation tool = strumenti che provano ad eseguire un attacco per verificare la presenza di vulnerabilità = molto pericolosi, usati per controllo industriale.

		■ Matrice nata in medicina per analizzare fenomeni complessi che non possono essere risolti utilizzando un metodo booleano	
Sintomo scelto per diagnosi:			
		F	T
Caratteristica di Interesse		Vero Negativo	Falso Positivo
Vulnerabilità Malattia	F		
	T	Falso Negativo	Vero Positivo
ACC = $\frac{TP+TN}{TP+TN+FP+FN} = 1 - ERR$		PR = $\frac{TP}{TP+FP}$	Recall = $\frac{TP}{TP+FN}$
Accuratezza	Errore	Precisione	Sensibilità
			Specificità

CVE/NVD/CPE

- Il programma CVE è identificare, definire e catalogare le vulnerabilità della sicurezza divulgate pubblicamente. Esiste un record CVE per ogni vulnerabilità nel catalogo. Le vulnerabilità vengono poi assegnate e pubblicate da organizzazioni.
- L'NVD è il repository del governo degli USA di dati di gestione delle vulnerabilità basati su standard. questi dati permettono l'automazione della gestione delle vulnerabilità, della misurazione della sicurezza e della conformità, viene incluso un database di riferimento a elenchi di controllo di sicurezza,

difetti sw, configutrazioni errate, nomi di prodotti e metriche di impatto.

- CPE è uno schema di denominazione strutturato per sistemi informatici, sw e pacchetti. Basato sulla sintassi generica per Uniform Resource Identifiers (URI), include anche un formato di nome, un metodo per controllare i nomi e un formato di descrizione per associare testo e test a un nome



Common Vulnerability Scoring System (CVSS)

Punteggi delle vulnerabilità, fornisce un modo per acquisire le caratteristiche principale di una vulnerabilità e produrre un punteggio numerico riflettente la sua gravità. e questo punteggio (basso, medio, alto e critico) serve per aiutare le organizzazioni valutate correttamente e dare priorità ai loro processi di gestione delle vulnerabilità.

Severity	Base Score
None	0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

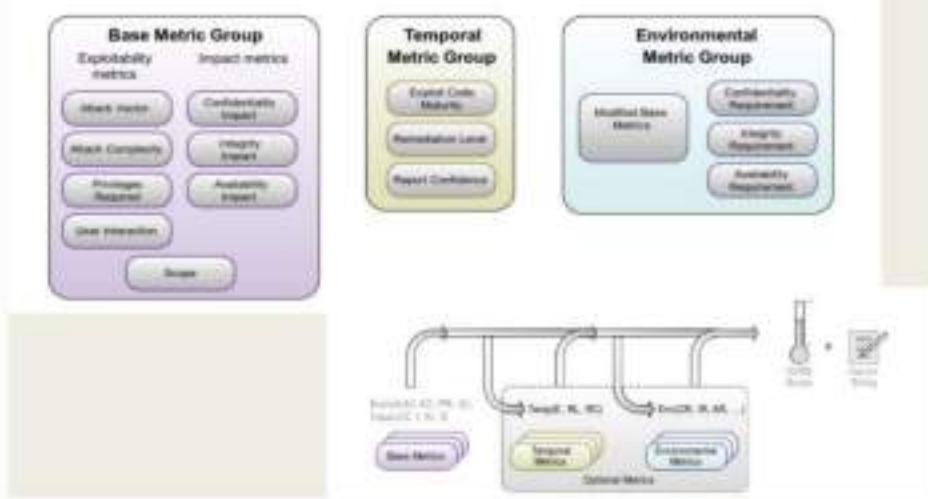
Idea strutturalmente sbagliata perché non si può valutare una vulnerabilità in maniera indipendente dal sistema che utilizza il componente affetto.

CVSS è composto da 3 gruppi di metriche:

- **Base**: indica le caratteristiche di una vulnerabilità che possono essere costanti nel tempo e negli ambienti degli utenti.
- **Temporale**: indica le caratteristiche di una vulnerabilità che cambia nel tempo ma non tra gli ambienti degli utenti.
- **Ambientale**: indica le caratteristiche di una vulnerabilità che sono rilevanti e uniche per l'ambiente di un particolare utente.

Lo scopo è definire e comunicare le caratteristiche fondamentali di una vulnerabilità così rende in maniera chiara e intuitiva la rappresentazione di una vulnerabilità per gli utenti.

Gli utenti possono invocare gruppi temporali e ambientali per fornire informazioni contestuali che riflettano in modo più accurato il rischio per il loro ambiente unico.



Access Vector

Metric Value	Description
Network (N)	The vulnerable component is bound to the network stack and the set of possible attackers extends beyond the other options listed below, up to and including the entire Internet. Such a vulnerability is often termed “remotely exploitable” and can be thought of as an attack being exploitable at the protocol level (one or more network hops away (e.g., across one or more routers)). An example of a network attack is an attacker causing a denial of service (DoS) by sending a specially crafted TCP packet across a wide area network (e.g., CVE-2004-0230).
Adjacent (A)	The vulnerable component is bound to the network stack, but the attack is limited at the protocol level to a logically adjacent topology. This can mean an attack must be launched from the same shared physical (e.g., Bluetooth or IEEE 802.11) or logical (e.g., local IP subnet) network, or from within a secure or otherwise limited administrative domain (e.g., MPLS, secure VPN to an administrative network zone). One example of an Adjacent attack would be an ARP (IPv4) or neighbor discovery (IPv6) flood leading to a denial of service on the local LAN segment (e.g., CVE-2013-6014).
Local (L)	The vulnerable component is not bound to the network stack and the attacker’s path is via machine/execute capabilities. Either: <ul style="list-style-type: none"> the attacker exploits the vulnerability by accessing the target system locally (e.g., keyboard, console), or remotely (e.g., SSH); or the attacker relies on User interaction by another person to perform actions required to exploit the vulnerability (e.g., using social engineering techniques to trick a legitimate user into opening a malicious document).
Physical (P)	The attack requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief (e.g., “evil maid attack”) or persistent. An example of such an attack is a cold boot attack in which an attacker gains access to disk encryption keys after physically accessing the target system. Other examples include peripheral attacks via FireWire/USB Direct Memory Access (DMA).

Attack Complexity

Metric Value	Description
Low (L)	Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success when attacking the vulnerable component.
High (H)	A successful attack depends on conditions beyond the attacker’s control. That is, a successful attack cannot be accomplished at will, but requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component before a successful attack can be expected. ² For example, a successful attack may depend on an attacker overcoming any of the following conditions: <ul style="list-style-type: none"> The attacker must gather knowledge about the environment in which the vulnerable target/component exists. For example, a requirement to collect details on target configuration settings, sequence numbers, or shared secrets. The attacker must prepare the target environment to improve exploit reliability. For example, repeated exploitation to win a race condition, or overcoming advanced exploit mitigation techniques. The attacker must inject themselves into the logical network path between the target and the resource requested by the victim in order to read and/or modify network communications (e.g., a man in the middle attack).

Privilegi richiesti / user interaction

Metric Value	Description
None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files of the the vulnerable system to carry out an attack.
Low (L)	The attacker requires privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges has the ability to access only non-sensitive resources.
High (H)	The attacker requires privileges that provide significant (e.g., administrative) control over the vulnerable component allowing access to component-wide settings and files.

Metric Value	Description
None (N) Required (R)	The vulnerable system can be exploited without interaction from any user. Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited. For example, a successful exploit may only be possible during the installation of an application by a system administrator.

Scope

Metric Value	Description
Unchanged (U)	An exploited vulnerability can only affect resources managed by the same security authority. In this case, the vulnerable component and the impacted component are either the same, or both are managed by the same security authority.
Changed (C)	An exploited vulnerability can affect resources beyond the security scope managed by the security authority of the vulnerable component. In this case, the vulnerable component and the impacted component are different and managed by different security authorities.

Exploit code maturity

Metric Value	Description
Not Defined (N)	Assigning this value indicates there is insufficient information to choose one of the other values, and has no impact on the overall Temporal Score, i.e., it has the same effect on scoring as assigning High.
High (H)	Functional autonomous code exists, or no exploit is required (manual trigger) and details are widely available. Exploit code works in every situation, or is actively being delivered via an autonomous agent (such as a worm or virus). Network-connected systems are likely to encounter scanning or exploitation attempts. Exploit development has reached the level of reliable, widely available, easy-to-use automated tools.
Functional (F)	Functional exploit code is available. The code works in most situations where the vulnerability exists.
Proof-of-Concept (P)	Proof-of-concept exploit code is available, or an attack demonstration is not practical for most systems. The code or technique is not functional in all situations and may require substantial modification by a skilled attacker.
Unproven (U)	No exploit code is available, or an exploit is theoretical.

Calculator:



Numero di vulnerabilità e qualità del software

Il numero delle vulnerabilità note NON può essere usato per stabilire quanto sia la qualità del prodotto stesso.

Il numero delle vulnerabilità è proporzionale al:

- Numero di vulnerabilità presenti
- Numero di ricercatori di vulnerabilità che analizzano il prodotto

Il numero di ricercatori e il valore della vulnerabilità scoperta che dipende dalla popolarità del sw usato.

Numero di vulnerabilità e intrusioni

- Scansioni di vulnerabilità esterne: eseguite all'esterno per controllare il firewall di una rete e altre tipologie di difese.
- Scansioni di vulnerabilità interne: testando ogni dispositivo su una rete, queste scansioni aiutano a identificare le vulnerabilità che lasciano un'azienda.
- Scansioni ambientali: queste scansioni si basano sull'ambiente in cui opera una tecnologia aziendale tipo cloud, IoT, mobile, siti web ecc.
- Scansioni intrusive e test di penetrazione: le scansioni a "forza bruta" tentano di sfruttare una vulnerabilità quando viene rilevata. Questo tipo di scansione può essere integrato da persone che tentano di ottenere un accesso autorizzato.

Molto fragile e importante fare queste scansioni per non far rilevare le vulnerabilità di un sistema da poter causare errori di ogni genere.

Scoperta di vulnerabilità in singolo modulo

Strumenti automatici:

- Analisi statica: strumenti in questa classe *static Application Security Testing (SAST) Tools* dipendono:
 - dal linguaggio di programmazione utilizzato
 - dal contesto web/sistema
- Analisi dinamica: noto come fuzzing, basato su generazione e trasmissione di valori errati come input del programma.

SAST:Punti di forza e di debolezza

- **Punti di forza**
 - Scalabilità, analizza grandi quantità di codice a basso costo
 - Identifica un sottoinsieme di vulnerabilità e stack overflow o SQL injection
 - Aiuta gli sviluppatori per individuare facilmente le istruzioni
- **Punti di debolezza**
 - difficile da implementare in particolare per i problemi di autenticazione, gestione errata dei diritti e uso non sicuro di crittografia.
 - Può creare falsi negativi
 - Non scopre problemi sulla sicurezza dovuti alla configurazione
 - spesso non analizzano librerie

Fuzzing: The Next Big Thing in Cybersecurity?

Fuzzing

è un metodo che invia ad una applicazione degli input malformati, cioè un blocco dell'applicazione evidenzia una debolezza del codice che può nascondere una vulnerabilità non ancora nota.

i tool di fuzzing sono il risultato della composizione di tre moduli:

- generatore di dati malformati
- distributore dei dati verso il modulo target
- componenti di monitoraggio per gestire eventuali errori e scoprire il percorso del dato in input nel modulo.

Application fuzzing: testa funzioni di tipo "pulsante" o campi di immisione di programmi grafici.

Protocol fuzzing: verifica il comportamento del programma nei casi di errore.

File format fuzzing: genera file errati da elaborare. può generare problemi in caso di file di formato errato. si testano anche funzioni, esempio la compressione dei file.

Tipologie di fuzzer

- **American Fuzz Lop:** Può compilare il codice sorgente e testarlo, se il testo non è disponibile si esegue una emulazione con QEMU (Quick Emulator)
- **Fuzzino:** Una libreria che fa parte dell'infrastruttura di compilazione LLVM.
- **ClusterFuzz:** Ambiente di test nato in google per il browser chrome
- **Sulley:** Raccolta di strumenti per programmazione python, adatto per test semplici come la generazione di dati casuali.
- **Peach:** Soluzione automatizzata per il fuzz testing di hw e sw.
- **Powerfuzzer:** Offre diversi scenari di attacco, tipo iniezioni SQL.

Time-machine torniamo a scanner (vulnerability scanner)

Eseguono sia attaccante per scoprire una vulnerabilità dei nodi che dal difensore per lo stesso motivo ma per sistemerli.

Ogni scanner permette di decidere la frequenza di invio messaggio a un certo nodo ed il numero di nodi analizzati, meno è la frequenza e numero dei nodi analizzati, minore è la probabilità che lo scanner venga rilevato dal difensore.

Stealth mode scanning = i parametri vengono settati in modo da minimizzare la probabilità di essere scoperti.

Il Client scanning = viene eseguito da:

- server quando il client si connette
- l'internet provider quando un client si connette alla rete
- service provider prima di fornire il servizio

Ambigua perché si acquisisce dati dei client e problemi di privacy.

Classi di fuzzing/fuzzer

Si classifica in diversi modi:

- Input si possono generare ex novo o mutando quelli esistenti.
 - basato sulla generazione
 - basato sulla mutazione
- Può conoscere o meno la struttura degli input in modo da modificare punti specifici o violando in modo o più punti la struttura.
 - Grammar based (informazioni sull'input e come cambiarlo)
 - Not grammar based
- Può conoscere diversi livelli la struttura del programma ed utilizzarla per garantire di aver testato tutti i cammini del programma.
 - White box "(conosco il codice sorgente)"
 - Black box "(non conosco il codice)"
 - Gray box "(conosco poco codice)"
- Possiamo applicare il fuzzing a codice o parser (text fuzzing)

Tainting analysis

Analisi premilinare del codice sorgente.

Calcola l'insieme di variabili di programma che potrebbero ricevere una variabile input e quindi potrebbe essere un bersaglio di un overflow.

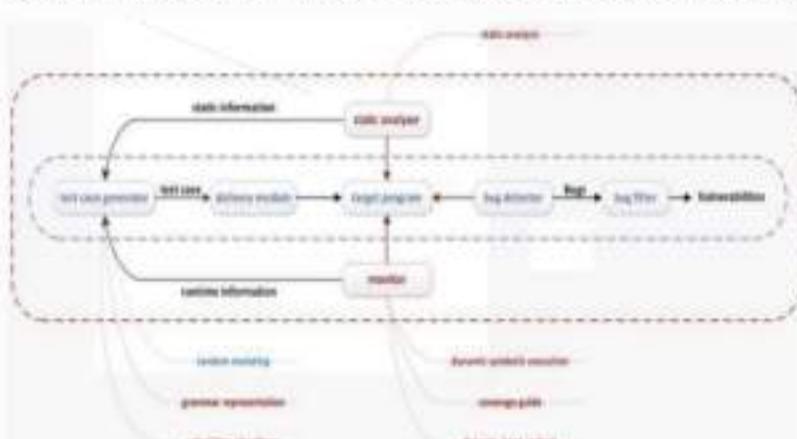
E' utile per scoprire quali istruzioni ricevono un input e se restituisce un set più grande di quello effettivo è un falso positivo.

```
if x (y=input) else (y=z);  
w=y;
```

La tainting analysis segnala che 'w' può essere stata tainted con un valore di input.

Può essere generalizzata con un analisi che colore le variabili.

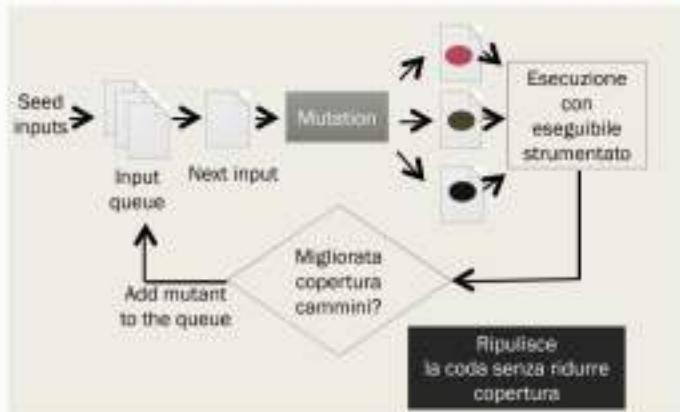
Ogni input possiede un colore diverso, e si vede come le variabili quali dipendenze dei colori possiede.



Fasi del fuzzing:



American Fuzzy Lop (AFL)



Uno dei più utilizzati, il binario viene strumentato a tempo di compilazione, il regular mode (strumenta il linguaggio macchina), Estensione (strumentazione codice mediante compilatore LLVM) e utilizza 64k contatori che rappresentano tutti gli archi in una applicazione, una hashtable ricorda il # di esecuzioni di ogni arco (si possono avere collisioni tra edge ma molto efficiente). L'applicazione viene eseguita in un processo separato da quello AFL.

Bright: Fuzz Testing for Application Security

Bright is the world's first AI-Powered Application Security Fuzz-testing tool.

Bright offers the combination of the world's leading DAST solution and a self-evolving, adaptive-learning fuzz solution. Bright applies evolution strategies and reinforcement learning to extensively analyze the response of the application and the context of a given attack surface breaking the assumed scope of the target. Bright reports vulnerabilities that are invisible to other, unintelligent fuzz testing tools.

Bright combines different technologies to raise efficiency and performance as the most comprehensive, reliable, and accurate solution. Bright comes with zero false-positives.

Quali bug/vulnerabilità scopre?

- **Memory leaks**
Incorrect management of memory allocations which is indicating available memory.
- **Injections**
Uncontrolled input supplied to a program.
- **Sensitive Data Exposure**
When an application inadvertently exposes personal data.
- **Insufficient Deserialization**
Maliciously crafted input injected into the control of a host application.
- **Buffer Overflows**
Overwriting the buffer stored in long data, which can lead to overwriting adjacent memory locations.
- **Use After Free**
An pointers to a valid object that can lead to data corruption, segmentation faults, or operates problematic faults.
- **Data Races**
Occurring when same memory location is accessed in multi-thread process, which can lead to security vulnerabilities.
- **Soft-assert Crashes**
Failing to initialize properly and causing possible fatal system errors.
- **Fenced Page**
Causing no response to inputs.
- **Uncaught Exceptions**
Not properly handling exceptions which can lead to disrupting a request handling.
- **Undefined Behavior**
Executing code not specified by language specifications (C/C++), which can lead to security vulnerabilities.
- **And many more**

Efficacia:

Technique	Effort	Code coverage	Defects Found
black box + mutation	10 min	50%	25%
black box + generation	30 min	80%	50%
white box + mutation	2 hours	80%	50%
white box + generation	2.5 hours	99%	100%

Risultati:

Time to first failure (TFFF): To measure the relative maturity of a given protocol, the time to first failure (TFFF), or the time to the first instance when that a protocol crashes, is used. It

The average time for first failure of all tests in 2016 was about 1 hour (1.8 hours). This represents a very slight increase over 2015 (1.7 hours), which might be attributable to a more diverse range of protocols in 2016.

5 most mature protocols (average time to first failure, in hours and days)

Protocol name	2016 time to first failure	2015 time to first failure	2015 time variance (avg)
TLS Client-Hello [P]	0.0 hours	0.0 hours	10.3 hours
SSH2 Client [P]	0.0 hours	1.0 hours	13.3 hours
SSH1 Server [Core] [P]	0.0 hours	0.2 hours	11.8 hours
SSL/TLS Client [P]	0.2 hours	1 hour	11.8 hours
ADMTraffic [Core] [P]	0.7 hours	1 hour	11.8 hours

5 least mature protocols (average time to first failure, in minutes and seconds)

Protocol name	2016 time to first failure	2015 time to first failure	2015 time variance (avg)
HTTP-1.1-0.9999999999999999 [S]	0.0 minutes	13.2 minutes	0.7 hours
ICMPv6-ICMPv4 [S]	0.0 minutes	0.0 minutes	0.0 minutes
NDNPTR-Traffic	0.0 minutes	0.0 minutes	0.0 minutes
Protocol Management	0.0 minutes	0.0 minutes	0.0 minutes
METTY-0.0000	0.0 minutes	0.1 minutes	1.2 hours
IANPA-2016	0.0 minutes	0.0 minutes	0.0 minutes



Standard fuzzing

ISO 26262	ISO/IEC 12207
Road vehicles - Functional Safety	Systems and Software Engineering - Software Life Cycle Processes
UN/ECE WP.29	ISO 27001
United Nations World Forum for Harmonization of Vehicle Regulations	Information Technology - Security Techniques - Information Security Management Systems
SAE/IEC 62443-4-1	ISO 22381
Secure Product Development Lifecycle Requirements	Security and Resilience — Business Continuity Management Systems
ISO/SAE DS 21434	IT-Sicherheit (Germany)
Road Vehicles — Cybersecurity Engineering	Based on ISO 27001
UL2900-1 and UL2900-2-1	NIST SP 800-55
Healthcare and Wellness Systems - Software Cybersecurity for Network-Connectable Products	Web Services — standard for software testing (USA) and others.
ISO/IEC/IEEE 29119	
Software and Systems Engineering - Software Testing	

Vulnerabilità strutturali

Non esistono metodi consolidati e la complessità nasce dalla necessità di considerare uno scenario con un insieme di moduli. Estensioni di fuzzing monitorando anche i moduli cooperanti perché un dato errato può trasmettere verso gli altri, il fallimento di ricevere un valore indica una mancanza di controlli su valori ricevuti ma anche la mancanza di controlli di chi invia i valori. Il controllo su autenticazione di canali in ingresso.

La comunicazione asimmetrica può permettere di aumentare il numero dei mittenti e quindi il numero di messaggi in input.

La delega dei controlli ad altri moduli può permettere di inviare dati manipolati nel momento in cui manca l'autenticazione sul canale in ingresso.

la valutazione di messaggi:

- Ripetuti
- Mancanti
- Inviati in ordine diverso da quello atteso.

I problemi non coperti da fuzzing sono collegati alla protezione delle informazioni scambiate, Uno scambio non protetto (in chiaro, non cifrato) può permettere accesso ad informazioni molto fragili. Tipo di problemi ne modello gerarchico.

Per tutte queste analisi è utile ricostruire i flussi di informazioni tra i vari moduli per valutare ripercussioni.

esempio: iniettando del codice malevolo in un browser posso manipolare le informazioni in un database server.

ATTACCHI INFORMATICI

Sono una particolare azione in una **intrusione** ed è composta di più azioni:

- Raccolta info
- Persistenza
- Attacchi per privilege escalation

Una sequenza di attacchi è una sottosequenza delle azioni in una intrusione.

In ogni attacco elementare possiede delle proprietà come:

- Diritti necessari
- Diritti acquisiti
- Probabilità di successo
- Rumore generato = cambiamenti nel sistema attaccato

altre proprietà descrivono in dettaglio le azioni per eseguire un attacco.

Classificazione di attacchi

- Buffer/stack/heap overflow
- Lettura illegale delle informazioni scambiate (sniffing)
- Ripetizione di alcuni messaggi legali (replay attack)
- Invocazione di operazioni in un ordine imprevisto dal progettista (interface attack)
- Intercettare e manipolare informazione scambiata tra due partner che comunicano (man-in-the-middle)
- Diversione di flussi di informazioni
- Time-to-use Time-to-check (Race condition)
- XSS (cross site scripting)
- sql injection
- Covert channel (Bell-Lapadula policy)
- Impersonare illegalmente (Masquerading)
 - un utente
 - una macchina (IP spoofing, DNS spoofing, Cache poisoning)
 - una connessione (connection stealing/embedding)

Esempio di replay attack

Utente e banca stabiliscono un canale sicuro e protetto.

Utente invia alla banca un messaggio con cui chiede di trasferire X euro dal suo conto a quello Y, Y snifferà il messaggio, lo ricorda ma prima che l'utente e la banca distruggano il canale, Y invia più volte il messaggio.

Il punto fondamentale è inserire informazioni che permettano di capire se una comunicazione è già stata ricevuta, si nota che l'attacco funziona anche se il canale è sicuro e protetto basta che Y possa riconoscere messaggio di interesse e questo è possibile se il protocollo è pubblico.

Esempio di man-in-the-middle



A e C pensano di parlare tra di loro ma tutte le comunicazioni possono essere intercettate mentre entrambi lo hanno creato con B.

Questo è possibile quando la comunicazione per creare il canale non è autenticata e la comunicazione tra A e C è effettivamente criptata ma si tratta di 2 canali: tra A e B - tra B e C.

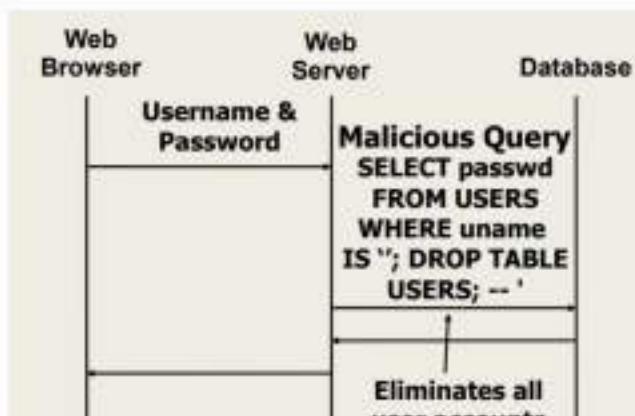
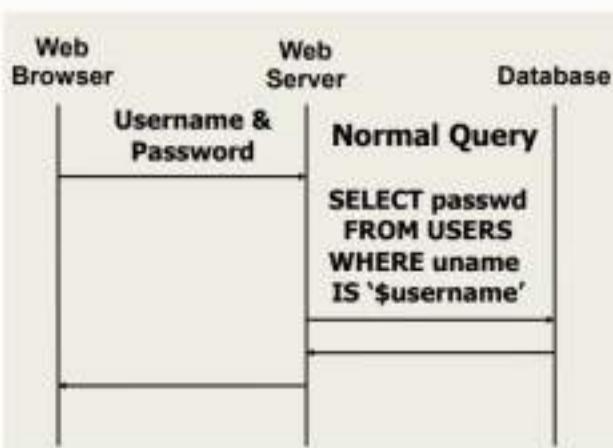
Semplicemente quando un algoritmo di routing o connessioni forzano tutti i messaggi scambiati tra A e C a passare da B.

Esempio di XSS

Un sito web permette agli utenti di memorizzare informazioni che poi viene scaricata da altri utenti.

Un utente malizioso può scaricare del codice che attacca gli altri utenti e se il sito non controlla quanto memorizzato può essere attaccato un gran numero di utenti e se si tratta di un attacco non mirato anche se talvolta viene usato per raggiungere uno specifico utente, è stato usato per la prima volta contro il sito della BBC che non controllava i commenti alle notizie che pubblicava.

SQL injection



Altro esempio:

```
View pizza order history:<br><form method="post" action="..."><br>Month<br><select><option name="month" value="1">Jan</option><br>...<option name="month" value="12">Dec</option></select><br><input type=submit name=submit value=View></form>
```

Normal SQL Query SELECT pizza, toppings, quantity, order_day FROM orders WHERE userid=4123 AND order_month=10
Attack For order_month parameter, attacker could input 0 OR 1=1 <option name="month" value="0 OR 1=1"> Dec</option> ...
Malicious WHERE userid=4123 AND order_month=0 OR 1=1 Query

La condizione WHERE sempre vera permette all'attaccante di accedere a dati privati

Difesa da SQL Injection

Whitelisting

Why? Blacklisting chars doesn't work:

Forget to filter out some characters

Could prevent valid input (e.g. username O'Brien)

Allow well-defined set of safe values:

[A-Za-z0-9]*

[0-9][0-9]

Valid input set defined through reg. expressions

Can be implemented in a web application firewall

Escaping

For valid string inputs like username o'connor, use escape characters:

Ex: escape(o'connor) = o'connor (only works for string inputs)

Prepared Statements & Bind Variables

public interface PreparedStatement extends Statement

Si utilizzano statement precompilati con parametri mancanti, si prelevano valori dalla query e si completa lo statement.

Si guadagna di efficienza e protezione da injection nel senso che se viene trasmesso un parametro che è parte di query viene ricercato quel valore.

```
PreparedStatement ps =  
    db.prepareStatement(  
        "SELECT pizza, toppings,  
         quantity, order_day  
      FROM orders  
     WHERE userid=? AND order_month=?");  
  
ps.setInt(1, session.getCurrentUserId());  
ps.setInt(2, Integer.parseInt(  
    request.getParameter("month")));  
ResultSet res = ps.executeQuery();  
  
query parsed w/o parameters  
bind variables are typed e.g. int, string, etc..."
```

Attacchi alla crittografia

- Attacco brute force
- Known-plaintext attack
- Cripto analisi differenziale
- Analisi potenza consumata
- Cripto analisi lineare
- Analisi dei tempi di esecuzione
- Meet in the middle attack (cifratura a blocchi che usa più chiavi)
- Man in the Middle
- Chosen chyper text
- Chosen plain text
- Cypher text only

Side-channel attacks

E' un attacco basato sulla misura di grandezze fisiche piuttosto che sulle debolezze che permettono ad accedere alle informazioni.

Esempi:

- Misurare le emissioni elettromagnetiche
- Misurare il consumo di energia
- Misurare il tempo di esecuzione per capire lo stato interno del sistema
- Misurare tempi di esecuzione per capire uso di cache e di meccanismi predittivi

Gerarchia di macchine virtuali ed attacchi

Qualsiasi sistema informatico è una gerarchia di macchine virtuali (assembler, os, rete ecc...).

Ogni macchina virtuale definisce un insieme di meccanismi (definisce un linguaggio di programmazione) e questi meccanismi astraggono e nascondono quelli delle macchine sottostanti e una qualunque delle macchine virtuali può essere standard con tutto quello che ciò implica sulle vulnerabilità.

Una vulnerabilità in un sistema operativo permette di attaccare una qualunque applicazione.

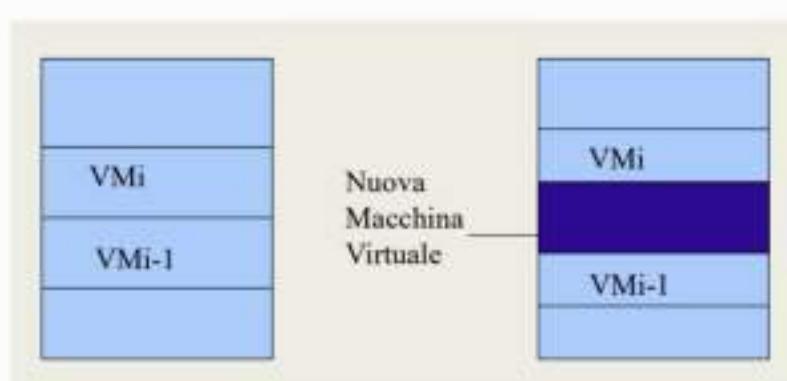
La sicurezza è un problema "olistico"

Going down

Si è osservato che attaccare i livelli bassi di un sistema il controllo sui livelli bassi della gerarchia permette di controllare tutti quelli superiori.

Un attacco interessante è quello che inserisce una nuova macchina virtuale nella gerarchia.

- Difficile da scoprire
- Con altissimi impatti dal punto di vista della sicurezza.



Blue Pill Attack

La nuova macchina può:

- Restituire informazioni false sullo stato delle macchine sottostanti.
- Inviare falsi comandi alle macchine virtuali sottostanti
- macchine in the middle

Esempio di Stuxnet, inviava comandi alle turbine per sabotarle mentre agli operatori diceva che non esistevano problemi.

ANALISI DELLE INTRUSIONI

Siamo certi di poter migliorare un sistema solo se conosciamo tutte le possibili intrusioni.

è stato dimostrato che:

se conosciamo solo alcune intrusioni e modifichiamo il sistema in modo di bloccare solo il sottoinsieme noto allora la probabilità di successo di una intrusione può aumentare.

teorema di Bayes

Teorema di Bayes

Teorema di logistica basato sul numero di ponti per attraversare un fiume.

Aumentando il numero di ponti su un fiume si aumenta il numero di percorsi, questo aumento del numero dei percorsi può portare al paradossale effetto di aumentare il tempo medio di percorrenza tra il punto di partenza e quello di arrivo.

Letto alla rovescia, cioè diminuire il numero dei ponti, permette di dimostrare il teorema sulle intrusioni.

Costruzione di una intrusione

Per la scoperta delle intrusioni è quello di adversary emulation cioè come un attaccante riesce a sequenzializzare correttamente le sue azioni per raggiungere un certo obiettivo.

Punto di partenza per modellare attaccante:

- Diritti iniziali = quelli legali
- Diritti finali = quelli che vuole ottenere

Scoperti gli attacchi necessari possiamo scoprire anche quali sono le azioni da eseguire per acquisire le informazioni necessarie per eseguire gli attacchi.

Ricordiamo che i difensori hanno tutte le informazioni necessarie su moduli del sistema e vulnerabilità mentre ogni attaccante ha un sottoinsieme di queste informazioni, quindi in generale l'attaccante NON può costruire attack graph perché gli mancano informazioni.

Gli insider sono gli attacanti più pericolosi perché sanno tutte le informazioni.

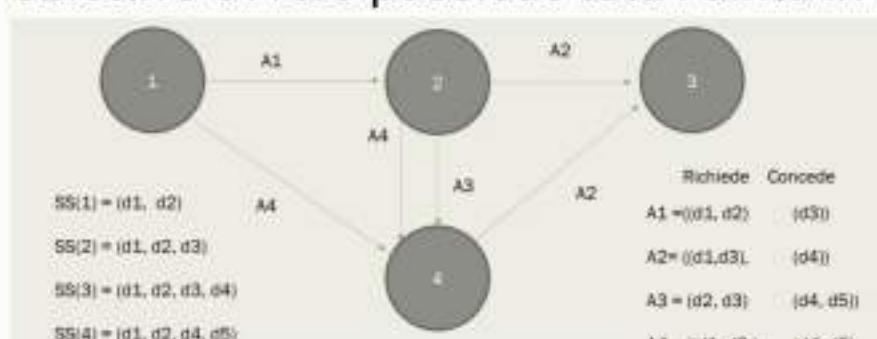
Attack Graph (intrusion graph)

Un grafo orientato ad un sistema ed un attaccante in cui:

- ad ogni nodo N è associato ad un insieme di diritti = security status ($SS(N)$) = diritti che l'attaccante possiede per gli attacchi eseguiti.
- Ogni arco A è etichettato con un attacco $att(A)$
- I diritti necessari per eseguire un attacco $att(A)$ con A che parte da N sono contenuti in $SS(N)$

Da notare che non ha cicli (attaccante è razionale e non spreca lavoro).

Il nodo iniziale I è quello in cui $SS(I)$ è dato dai diritti legali dell'attaccante, in ogni nodo finale, l'attaccante ha raggiunto un obiettivo ovvero possiede tutti i diritti in un suo obiettivo.



Acquisizione dei diritti è monotona quindi in un nodo si possono eseguire tutti gli attacchi possibili nel cammino fino a quel nodo purché:

- Non siano già stati eseguiti
- Concedano almeno un diritto che l'attaccante ancora non possiede

Acquisizione monotona è dovuta al fatto che si modellano solo i possibili attacchi e non anche le possibili azioni di difesa.

Complessivamente acquisizione monotona porta a grafi molto ricchi e di difficile costruzione se sono complessi.

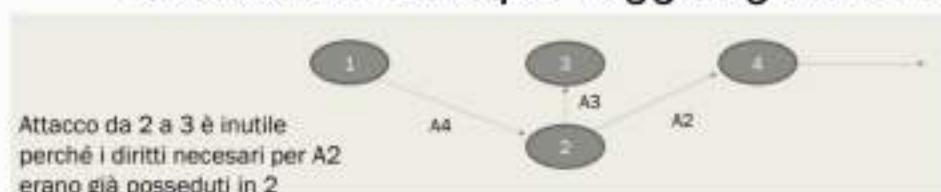
Da notare che più dettagliata la modellazione dei diritti, maggiore è la complessità del grafo e della sua costruzione.

Descriviamo attack graph

La premessa è che nel grafo avremo anche attacchi inutili perché si ipotizza che l'attaccante non aveva le informazioni che gli permettevano di capire quale attacco non era inutile.

sicché:

- Non contiene cicli
- Non contiene attacchi che concedono diritti che l'attaccante già possiede
- Può contenere attacchi inutili ovvero che concedono diritti che non servono per raggiungere un obiettivo.



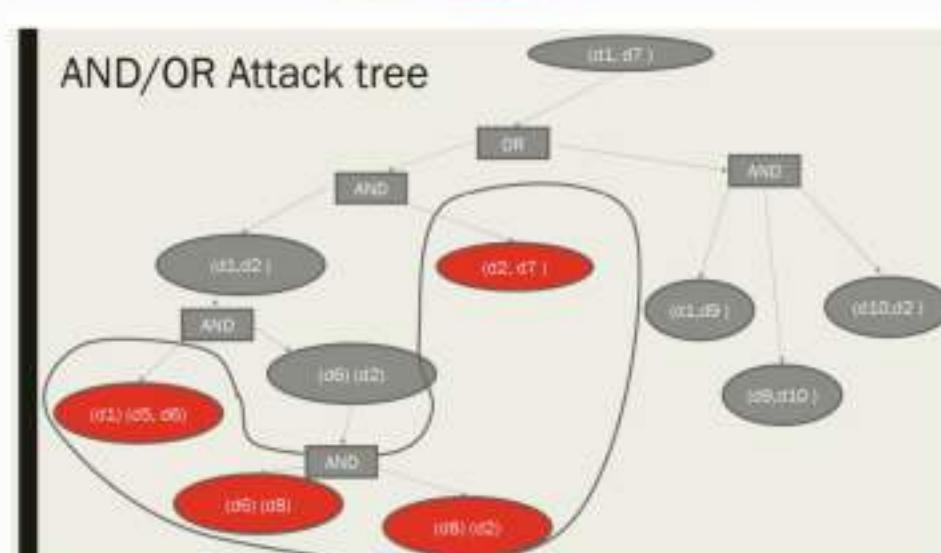
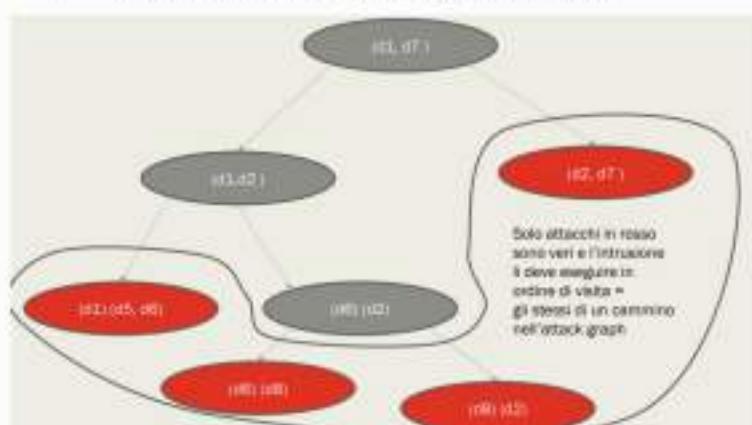
Attack Tree (intrusion tree)

Più semplificata di attack graph, una intrusione viene rappresentata come un albero, i nodi indicano gli attacchi ed i sottoalberi come un certo attacco può essere realizzato.

I nodi che corrispondono ad attacchi esistenti non sono decomposti in sottoalberi, gli altri vengono invece composti.

Il cammino sul grafo diventa la frontiera = le foglie dell'albero
possiamo avere anche una versione AND/OR dell'albero:

- AND = tutti gli attacchi nei nodi figli devono essere eseguiti.
- OR = basta un attacco.



Fermare una intrusione/una minaccia

Occore fermare un attacco in ogni intrusione che la minaccia può eseguire, fermando un attacco la catena della intrusione considerata viene interrotta.

Se fermiamo l'attacco che fa apparire più intrusioni noi fermiamo tutte le intrusioni in cui l'attacco appare.

abbiamo un problema di ottimizzazione

scegliere il minimo numero di attacchi da fermare tali che permettano di fermare tutte le intrusioni oppure

scegliere un insieme di attacchi da fermare tale che permettano di fermare tutte le intrusioni che abbiammo il costo minimo.

Ranking di vulnerabilità

Il problema di prima indica come possiamo ottenere uno scoring delle vulnerabilità.

Lo scoring di una vulnerabilità dipende dal numero di intrusioni che posso fermare eliminando la vulnerabilità stessa, inoltre, aumenta se la vulnerabilità appare in un insieme di quelle da eliminare per fermare una minaccia, inoltre, dato un insieme di vulnerabilità da eliminare possiamo stabilire uno scheduling che permette di ottimizzare la riduzione del rischio.

ad ogni passo eliminiamo la vulnerabilità che appare nel maggior numero di intrusioni non ancora fermate.

Attacchi/intrusioni automatizzabili

Attacchi e intrusioni si possono automatizzare.

Attualmente stiamo assistendo all'automazione di alcuni step dell'intrusione per ransomware ad esempio:

- Accesso iniziale nel sistema
- Esplorazione e raccolta di informazioni
- Lancio di ransomware

Persistenza

Attaccanti interessati a rubare informazioni o ad attivare malware cercano di essere persistenti nel sistema ovvero ad avere account ecc.

Moduli che attaccante installa sul sistema target interagiscono con attaccante tramite quella che si chiama infrastruttura di comando e controllo C2 infrastructure, un caso particolare di botnet, viene creata prima dell'intrusione e può essere usata anche per eseguire intrusione.

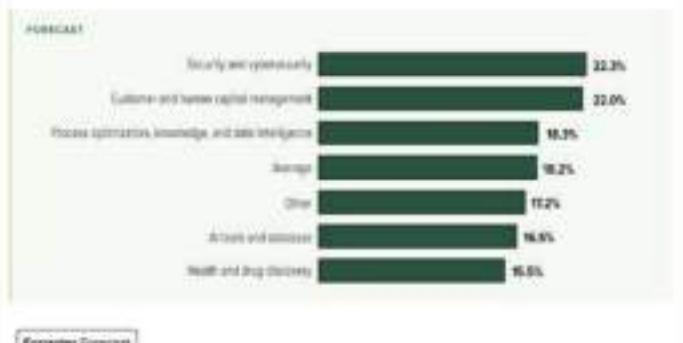
Moduli istallati interagiscono con nodi diversi della C2 infrastructure proprio per non generare rumore che possa portare alla loro scoperta fast techniques che può essere scoperta

monitorando query DNS.

Carriere nella cybersec

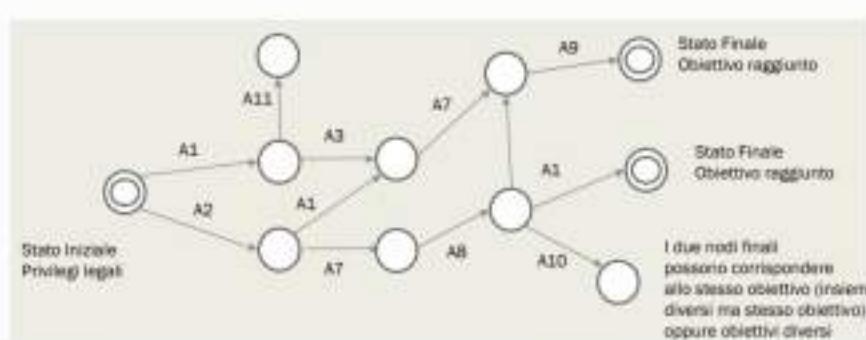


AI software spend CAGR 2020 to 2025

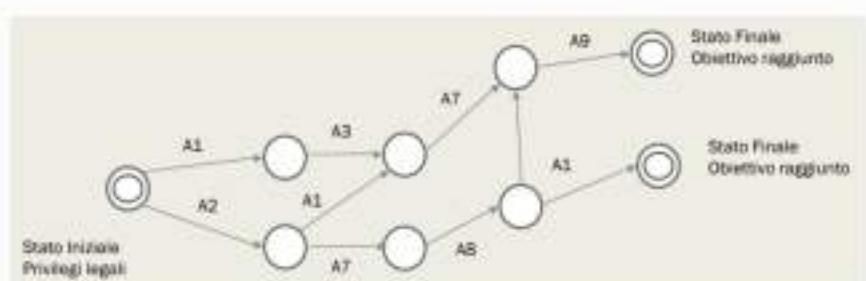


Attack graph: i tre passi

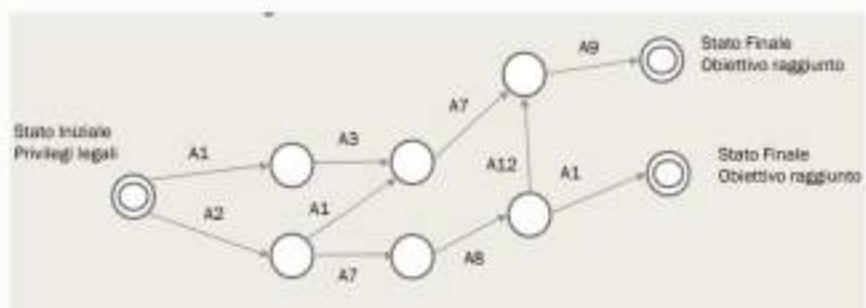
1. Costruzione del grafo (ogni attacco individua una coppia (modulo, vulnerabilità))



2. Eliminazione cammini inutili



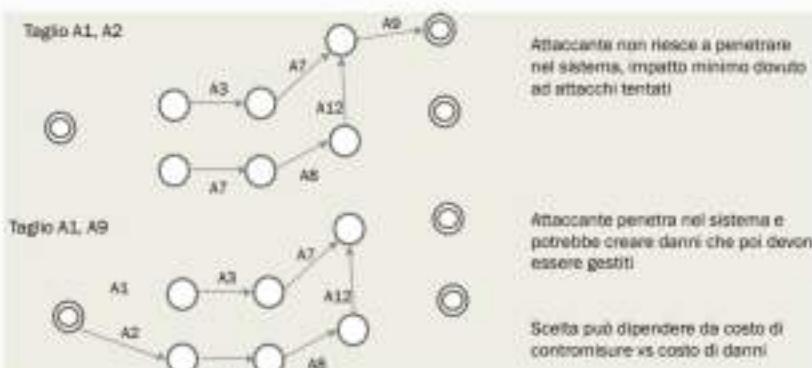
3. Calcolo del taglio



Tagli possibili = vulnerabilità da eliminare = (A1, A2), (A1, A9), (A7, A8), (A7, A12, A3), (A3, A1, A7).

Gli ultimi due tagli non sono minimi ma potrebbero essere interessanti per costo minore di patch.

Calcolo del taglio non è un semplice problema di flusso



DESCRIZIONE DI UNA MINACCIA

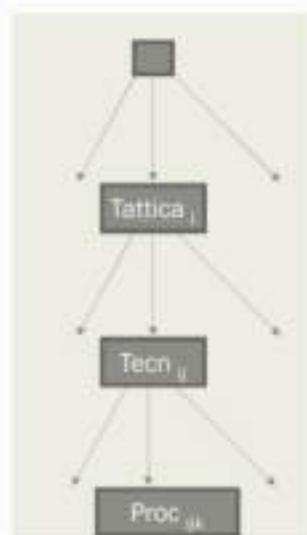
Mitre att&ck matrix

MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.

With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world — by bringing communities together to develop more effective cybersecurity. ATT&CK is open and available to any person or organization for use at no charge.



Ci fornisce varie tattiche, tecniche e procedure (TTPs) che un attaccante può usare nelle sue intrusioni.



Più versioni per coprire:

- Pre-attack
- Enterprise
- Cloud
- ICS
- Mobile

C'è una gerarchia di:

- Tattiche: obiettivi a breve termine durante una intrusione
 - Tecniche: come viene taggiunto obiettivo a breve termine
 - Procedure: dettagliata implementazione di una tecnica
- permette una emulazione molto dettagliata di un avversario e viene anche usata per standardizzare report su attacchi o per threat intelligence.

Tattiche TTPs

Le **tattiche rappresentano il "perché"**, cioè l'obiettivo tattico che gli attaccanti si prefiggono durante una delle fasi di un attacco, mentre le **tecniche rappresentano il "come"**.

Le procedure specificano in dettaglio l'implementazione delle tecniche invece la matrice descrive anche le tecniche di mitigazione che si possono applicare ed è evidente che mancano informazioni su come tattiche sono ordinate in una strategia complessiva. La matrice è focalizzata su descrizione per rilevazione e comunicazione e non sulla scoperta delle possibili intrusioni.

Tattiche

1. **Reconnaissance**: raccogliere informazioni per pianificare il futuro avversario. esempi: scanning, ricerche web, accesso ai siti della vittima.
2. **Resource Development**: stabilire le risorse per supportare le operazioni, esempi: Creazione di una infrastruttura, attacchi di account
3. **Initial Access**: come entrare nella tua rete, esempi: attacchi ad applicazione esposte, phishing, aggiunta di dispositivi hardware
4. **Execution**: eseguire codice dannoso, esempi: Utilizzo di un enterprise già esistente, deployment di un container.
5. **Persistence**: cercare di mantenere il loro punto d'appoggio, esempi: Manipolazione di account, Aggiunta componenti al boot del sistema, Attacco al sistema di autenticazione.
6. **Privilege Escalation**: tentativo di ottenere autorizzazioni di livello superiore, esempi: Abuse Elevation Control Mechanism, Exploitation for privileges, Process injection and Escape to host.
7. **Defense Evasion**: cercare di evitare di essere scoperti, esempi: Abuse elevation control mechanism, Deploy container, Indicator removal(manipolazione di un indicatore di intrusione) and Exploitation for defence evasion(attaccare modulo per sicurezza/rilevazione).
8. **Credential Access**: furto di nomi e password di account, esempi: Man-in-the-middle, Forced Authentication and Input capture.
9. **Discovery**: cercando di capire il tuo ambiente, esempi: Account discovery, Application discovery, Software discovery.
10. **Lateral Movement**: muoversi attraverso l'ambiente, esempi: Exploitation of remote services, Lateral tool transfer(installazione di strumenti sul nodo) and Taint shared content(attacco a memoria condivisa).
11. **Collection**: raccolta di dati di interesse per l'obiettivo avversario, esempi: Adversary in the middle(tutte le versioni di man-in-the-middle), Audio capture and Video capture.
12. **Command and Control**: comunicazione C2 con sistemi compromessi, esempi: Protocol tunneling and dynamic resolution.
13. **Exfiltration**: furto di dati, esempi: Exfiltration over web services, Exfiltration over network medium, Exfiltration over physical medium and Transfer to cloud account.
14. **Impact**: manipolare, interrompere o distruggere sistemi e dati, esempi: Data Destruction, Data encryption, Data Manipulation and Inhibit system recovery.

Account discovery

1001	Account Discovery	Adversaries may attempt to get a listing of accounts on a system or network or environment. This information can help adversaries determine which accounts could be used to log in followed on further.
1011	Local Account	Adversaries may attempt to get a listing of local system accounts. This information can help adversaries determine which local accounts exist on a system to aid in follow-on techniques.
1021	Domain Account	Adversaries may attempt to get a listing of domain accounts. This information can help adversaries determine which domain accounts exist to aid in follow-on techniques.
1031	Email Account	Adversaries may attempt to get a listing of email addresses and accounts. Adversaries may try to dump Exchange address lists such as global address lists (GAL).
1041	Cloud Account	Adversaries may attempt to get a listing of cloud accounts. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or cloud application.

ID	Name	Description
10001	remove@123	removes 123 users using the following command following exploitation of a exploit with ID 100001, released as an intermediate payload. It should release and user named remove@123 through creation. ¹¹
10002	agent@123	Agent@123 collects account information from the system database. ¹¹
10003	aff11	aff11 uses the command <code>use /local/accounts/aff11.aff</code> and <code>list</code> message to find accounts on the system. ¹¹
10004	aff12	aff12 has used a localized tool called <code>aff12.aff</code> to find encrypted group-accounts, power-accounts and administrators. ¹¹
10005	aff122	aff122 implemented administrative access using the command <code>id 1</code> (executing <code>id -u</code> as root). ¹¹
10010	Administrator	Administrator gathered 4 accounts and success memory information through process monitoring. ¹¹
10011	Admin	Admin uses memory administrative accounts on unrestricted host. ¹¹
10012	Administrator	Administrator can determine the accounts for each user using <code>whoami</code> command tool. ¹¹
10013	Administrator	Administrator can gain local identity access with local administrator rights. ¹¹
10014	Administrator	Administrator has user and usage by different threads. ¹¹
10015	System	System uses the <code>id -u</code> command. ¹¹

Privilege Escalation



MITRE ATT&CK for Cloud, 2021

Le intrusioni in sistemi cloud di solito non comprendono malware perché il provider è in grado di rilevarlo.
Nuovi comportamenti possibili come furto di credenziali.



creazione di nuove macchine virtuali.

Efiltrazione può essere implementata come trasferimento di dati ad un altro account.

MITRE ATT&CK for containers, 2021



Numero ridotto di tattiche, fondamentalmente stiamo attaccando una applicazione e quindi abbiamo una matrice specializzata per la specifica applicazione.

Command & Control Infrastructure

La matrice trascura azioni di attaccante per creare una propria infrastruttura di C&C che può servire sia per lanciare i primi passi di intrusione che per interagire con il sistema attaccando quando intrusione o persistenza hanno successo.

Le azioni per creare C&C NON riguardano il sistema target e quindi non è possibile tenerne conto per rilevare e sconfiggere intrusione. La creazione di C&C un problema cyber igiene ovvero la creazione è semplificata se esiste un grande numero di sistemi a bassa sicurezza perché ognuno di loro è un buon candidato per creare la botnet su cui si va a costruire C&C.

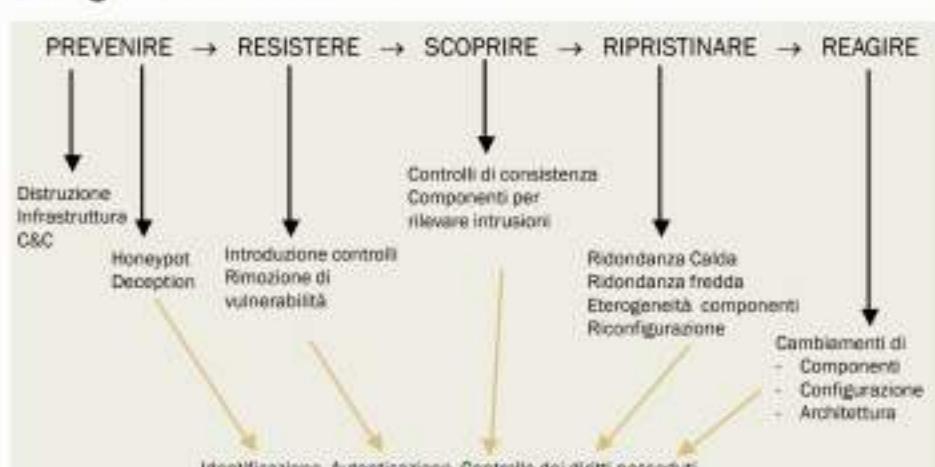
Inoltre i metodi usati per attaccare quando si crea C&C interessa minimizzare rumore e probabilità di detection. quindi i metodi usati possono differire da quelli usati quando poi si lancia intrusione vera e propria.

CONTROMISURE

- **Proattive:** applicate prima di una intrusione per impedirla ad esempio il patching din vulnerabilità.
- **Dinamiche:** Applicate durante una intrusione per fermarla ad esempio, isolamento/spegnimento di un nodo.
- **Reattive:** Applicate dopo una intrusione per fermare quelle successive, esempio, cambiare una password o patchare un sistema.

Un problema per Dinamiche e Reattive è la scoperta delle intrusioni e dei loro impatti.

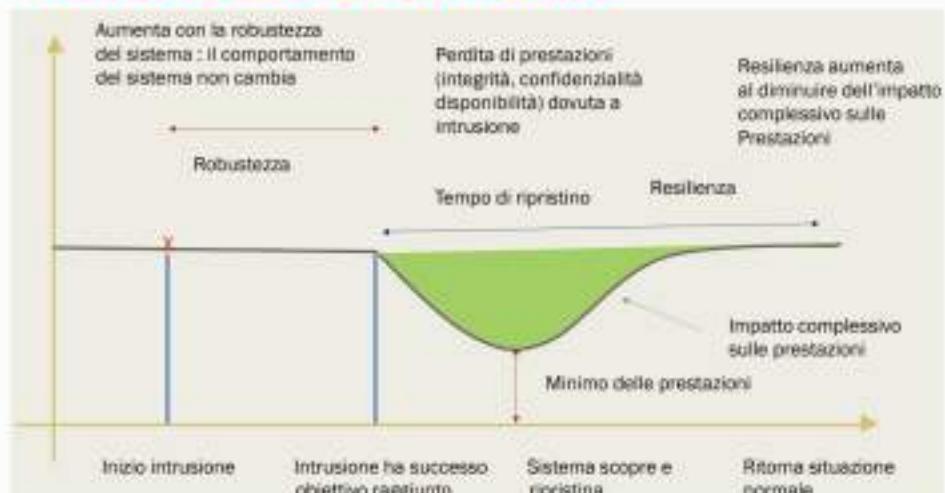
Origine militare...



NIST INCIDENT HANDLING LIFECYCLE



Robustezza e resilienza



La base sono la ridondanza ed eterogeneità.

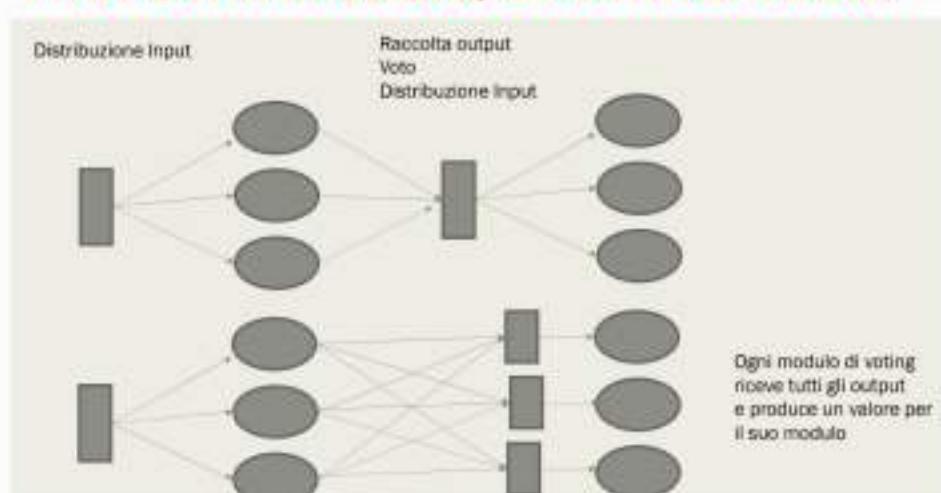
- Ridondanza:

- sistema sovradimensionato in modo da compensare eventuali perdite.
- Ridondanza fredda = esistono instanze inattive di alcuni moduli che vengono attivate solo quando quelle normalmente in uso sono state attaccate o guaste.
- Ridondanza calda = esistono più instance attive in modo da poter sopportare attacchi.
- Triple modular Redundancy = ridondanza calda dove tre copie di uno stesso modulo ricevono lo stesso input eseguono la stessa computazione e poi votano su risultato.
- TMR aumenta safety ma non sempre security

- Eterogeneità : uso di componenti diversi per evitare punti di caduta catastrofica per una singola vulnerabilità. Tipico esempio è l'uso di sistemi operativi diversi in una stessa rete.

NASA richiede che se un produttore lavora su moduli ridondati allora un altro produttore deve lavorare su voler evitare bias o stesso errore ripetuto.

TRM centralizzato/decentralizzato



Sistema Minimale

Un sottoinsieme del sistema formato da componenti particolarmente robusti e spesso eterogenei rispetto a quelli del sistema principale.

La robustezza diminuisce la probabilità che essi siano attaccati con successo.

Il sistema minimale è quello da cui parte ripristino del normale funzionamento del sistema e la perdita del sistema minimale impedisce di ripristinare un comportamento normale e può risultare in un impatto catastrofico per il proprietario del sistema (black swan).

Consideriamo il sistema di ATM:

- Perdita del singolo ATM ha un impatto grave ma prevedibile
- Perdita del database con tutte le chiavi per collegamento a vari ATM ha un impatto non calcolabile a priori e che può impedire di ripristinare uno stato consistente nel sistema complessivo.

Meccanismi comuni alle contromisure

E' basata su un insieme di meccanismi condivisi.

La condivisione può aumentare la cost effectiveness delle contromisure, i meccanismi condivisi devono però essere molto robusti perché una loro vulnerabilità può influenzare molte contromisure.

I meccanismi condivisi sono implementati su un security kernel che fa parte del TCB e che gestisce:

- Le identità degli utenti
- L'autenticazione degli utenti
- I diritti degli utenti

Autenticazione e autorizzazione

In qualsiasi sistema informativo la tripla *<soggetto, oggetto, operazione>* che associa ad un oggetto un diritto svolge un ruolo fondamentale.

Possiamo associare 2 tipi di controlli a questa tupla:

- Controlli sull'identità del soggetto.
- Controlli sul possesso del diritto *<Oggetto, operazione>* da parte del soggetto.

Controlli sulla identità del soggetto = identificazione del soggetto.

Autorizzazione è la gestione dei diritti dei vari soggetto, cioè, cosa può fare il soggetto che è stato autenticato.

I controlli su possesso dei diritti e la gestione dei diritti dei vari utenti è sostanzialmente compito del SO, anche l'autenticazione è compito del SO ma attualmente esistono componenti specializzati per autenticazione che poi distribuiscono token che certificano in qualche modo che sia avvenuta.

Tipi di autenticazione:

- Debole statica: include le password e altre tecniche che sono suscettibili di compromissione con attacchi che riproducono le sequenze di autenticazione (replay/sniffing attacks).
- Debole non statica: che contempla l'uso della crittografia o di altre tecniche, per creare password one-time valide per un'unica sessione. Questa soluzione può essere compromessa con il furto della sessione web (session hijacking).
- Forte, che adotta tecniche matematiche o crittografiche per prevenire i principali problemi dell'autenticazione debole.

Meccanismi di autenticazione:

Qualcosa che solo l'utente conosce, frase, numero ecc.

- Hashing di informazione sul server (per una o più volte)
- Uso di salt (database diversi) per impedire attacchi che precalcolano hash.

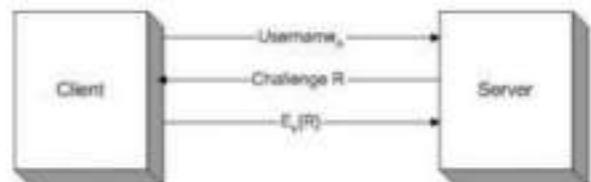
Qualcosa che solo l'utente possiede, come una chiave di cifratura, una scheda magnetica, dispositivo di autenticazione, che genera una OTP o la risposta del server.

- Generare il prossimo numero di una sequenza pseudocasuale (sincronizzazione con il server di autenticazione).
- Applicare una funzione condivisa ad un valore ricevuto dal server e restituire risultato (challenge response con possibilità di più challenge)

Qualcosa che sono l'utente è, normalmente dipende da una qualche caratteristica fisica.

- Problemi con falsi negativi/falsi positivi
- Attacchi a trasmissione di informazioni biometriche
- Difficile modificare la <>password><

Challenge - response

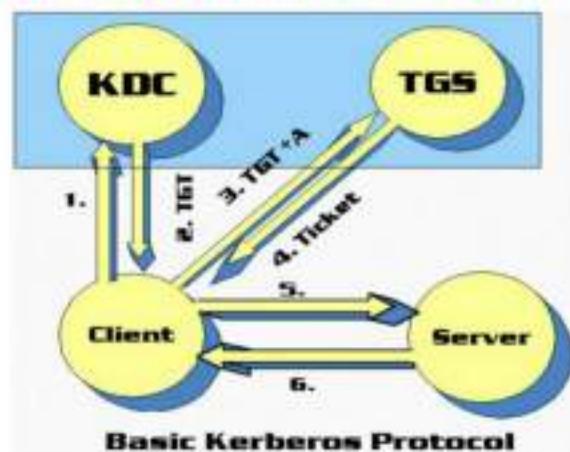


Il server può inviare più challenge

Autenticazione biometrica



Kerberos: Autenticazione



Autenticazione "three-sided" a chiave segreta condivisa(DES).

- client = l'utente che accede a un servizio
- server = è la risorsa che vuole accertarsi che i client siano autorizzati.
- repository centralizzato delle chiavi, Key Distribution Center (KDC).
- servizio di generazione Ticket Garanting Service (TGS) (Authentication Server).

La chiave segreta condivisa indica che 2 parti condividono la chiave con cui possono verificare le reciproche identità.

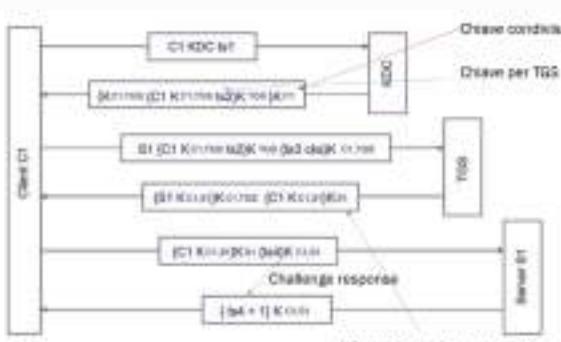
Principio di base: **Se conosci la chiave nel repository allora sei autenticato.**

I ticket sono delle convalide con cui i client provano la loro identità ai server per accedere ai servizi che questi offrono e sono validi per un tempo limitato.

KDC e TGS gestiscono dei database in cui sono memorizzate le master key (chiavi private generate a partire dalle password) dei client e dei server registrati.

Le master Key all'interno del database sono criptate con la chiave privata rispettivamente del KDC e del TGDS. Le password di tali componenti rappresentano l'ultimo livello di difesa perché chi conosce queste password può compromettere l'intera rete.

Un KDC fisicamente sicuro è fondamentale per prestazioni.



Problemi:

Ogni programma per usare kerberos deve essere modificato e ciò è tanto più complesso in base a programma che deve essere "kerberizzato".

Questo processo non è sempre possibile poiché si deve avere accesso ai sorgenti delle applicazioni da trasformare.

Tutte le password sono cifrate con una singola master key. Se il server è compromesso si deve cambiare la password a tutti i client e server registrati.

Il tempo limitato per la validità dei ticket è un problema per

applicazioni con un run time elevato, e se il server cade la rete che usa Kerberos è inutilizzabile.

Rilevazione di intrusioni difficile per utente, i problemi in ambiente time-sharing. In Unix kerberos mantiene i ticket in /tmp che è una directory pubblica cui chiunque può accedere per rubare i ticket. Vita limitata di ticket (8 ore) crea problemi per le applicazioni con tempi di esecuzione elevati. Per questo servono politiche di gestione e rinnovo delle convalide (ticket).

CONTROLLO (GESTIONE) DEI DIRITTI DEI SOGGETTI

Si indica la gestione della matrice di controllo degli accessi soffetti, oggetti e diritti.

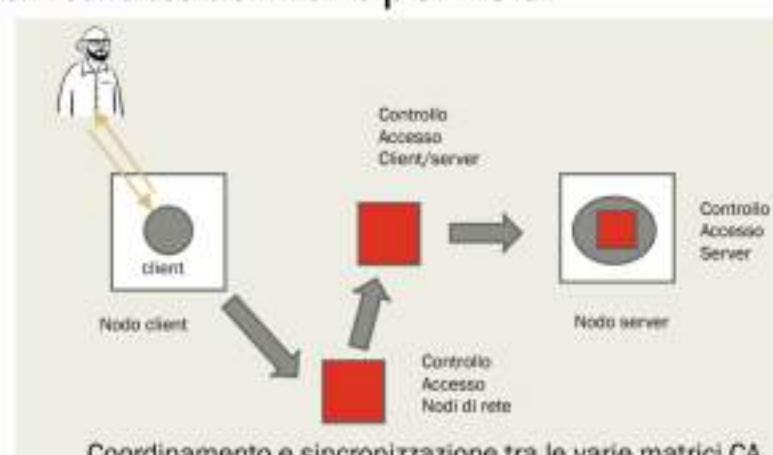
Tipica di DAC:

	01	02	03	04		Ok
S1						
S2		opi, opj, opk				
Sw						

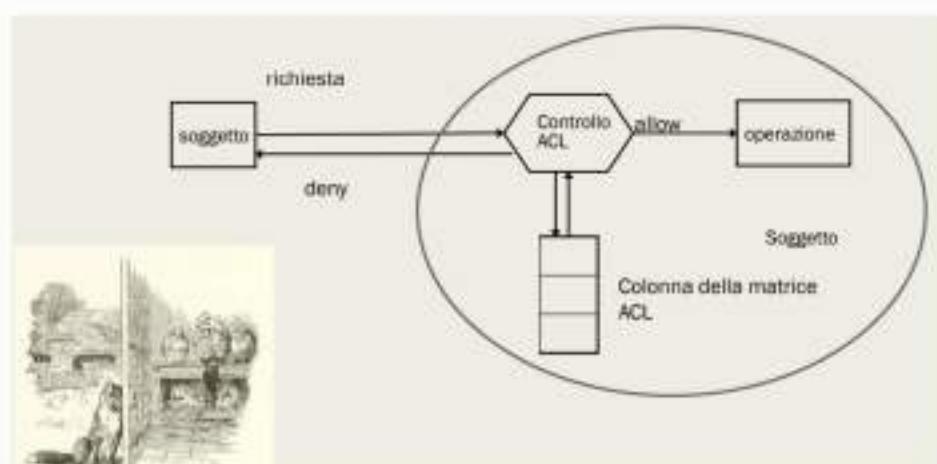
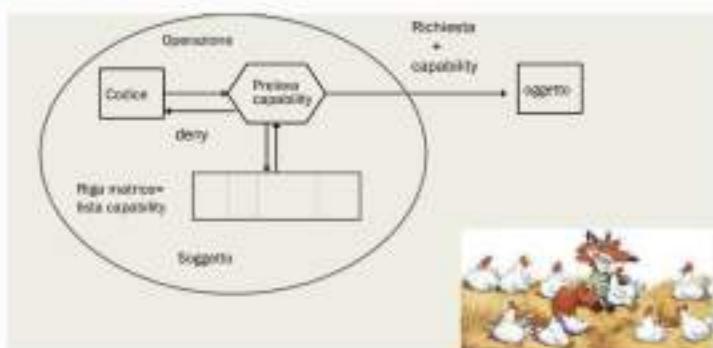
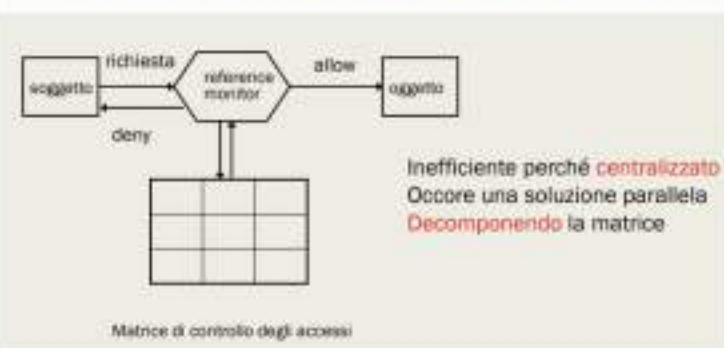
Una matrice a livello del sistema operativo per protezione di zone di memoria, file, dispositivi I/O, porte ecc.

Possiamo avere matrici anche a livello applicativo per la specifica applicazione ad esempio per la gestione di database, web server ecc.

In generale la matrice in un SO determina quali utenti (client) possono interagire con una applicazione (server). Questo può controllare mediante il meccanismo delle porte o dei canali di comunicazione. Per quelli che possono interagire, la matrice per il controllo degli accessi del server determina quali operazioni un particolare client può richiedere. Se un client non può e non potrà interagire con un server conviene impedire l'interazione al livello più basso e non permette di interagire e poi scartare la richiesta al server, questo si chiama Overhead, cioè, scartare la richiesta nel server e facilità attacchi di tipo DOS.



Controllo degli accessi in astratto



Controllo degli accessi: capability

Capability = puntatore esteso con operazioni possibili.

La soluzione si memorizza le informazioni nel soggetto che poi presenta all'oggetto la capability, il biglietto che autorizza operazione.

La soluzione più nota è quella per la protezione dei segmenti di memoria. Nella tabella per la traduzione degli indirizzi da logici a fisici troviamo anche un gruppo di bit che ci dicono le operazioni possibili su quel segmento.

Nel caso più generale ogni puntatore ad un oggetto diventa un puntatore protetto perché soggetto è interessato a manipolarlo. Una possibile soluzione è encryption e decifrazione solo quando sono esatti.

Implementazione centralizzata

Progettare e costruire architetture a capability es: Plessey PP250 dove tutti i puntatori sono capability.

Le capability sono protette dall'hw e/o dal SO tramite registri specializzati. Una funzione viene denominata da un indice in un segmento o un una tabella protetta dal SO.

Implementazione distribuita

Non può essere protetto da hw/SO perché devono essere trasferiti attraverso le reti e passare attraverso lo spazio utente, e devono essere protetti mediante crittografia.

Da notare che chi fornisce il servizio o gestisce un oggetto gestisce un **SEGRETO** (numero random, chiave di cifratura) e usa una funzione **f()**, one-way non invertibile.

capability =

protected fields	check digits (signature)
------------------	--------------------------

La capability è costruita usando:

check digits = f (SECRET, protected fields)

Quando la capability viene presentata al server con una invocazione il server controlla che:

f (SECRET, protected fields) = check digits

Se la condizione è violata, l'invocazione non viene servita

Proprietà: (Implementazione distribuita)

- **Protezione**
 - dal tampering
 - dal furto se viene autenticato chi la presenta
- **Controllo della propagazione**
 - la capability può essere create solo dal gestore
- **Delega**
 - un soggetto può trasmettere ad altri la capability ma abbiamo problema con protezione.
- **Revoca**
 - molto complessa se è stata usata delega (si "rincorre" la capability)

Controllo degli accessi: ACL

Lista di controllo degli accessi prevede di memorizzare la matrice nell'oggetto da proteggere, l'implementazione dell'oggetto prevede anche delle informazioni per controllare le richieste di operazione che arrivano all'oggetto stesso.

Caso tipico è quello dei router: un router è un componente hw/sw per istradamento dei messaggi con lenee di ingresso e di uscita, Ogni linea è collegata ad un nood di rete p ad un altro router.

Ogni linea riceve un messaggio (pacchetto) che ha un mittente ed un destinatario specificato come un indirizzo IP.

Ad ogni linea è associato una lista di regole che dicono come trattare un pacchetto che vuole entrare/uscire da quella linea.

Una tabella nel router instrada ogni pacchetto "entrato" su una o più linee in uscita in base alla destionazione.

La lista (ACL) di ogni linea del router è costruita componendo 2 casi:

La lista (ACL) di ogni linea del router è costruita componendo due casi
1. IP Range₁ → route i pacchetti da questi nodi possono passare

2. IP Range₂ → drop i pacchetti da questi nodi sono scartati

Lista = ordine è importante
si specificano dei range perché non tutti gli indirizzi IP sono noti

ACL of input 1

131.114.*.* → route traffico da 131.114.*.* viene instradato
131.4.5.6 → route traffico da 131.4.5.6 viene instradato
131.4.*.* → drop tutto altro traffico da 131.4.*.* eliminato

ACL of output 1

131.114.*.* → drop
131.4.*.* → drop
* → route

Due sottoreti non possono comunicare con sottorete connessa ad out 1

Domanda: questa soluzione è default allow o default deny??

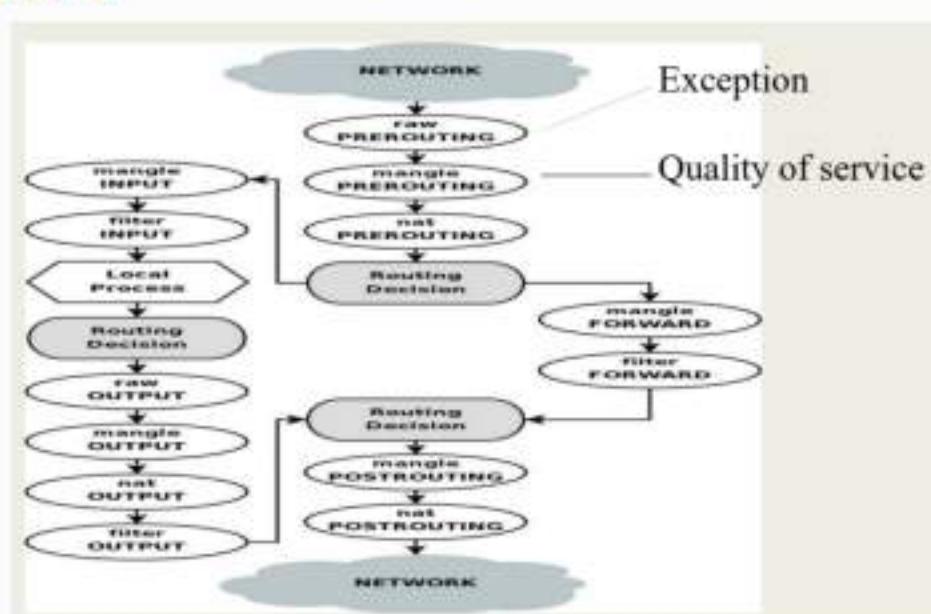
Routing in linux: iptables

- Input chain = regole per i pacchetti indirizzati al nodo
- Output chain = regole per i pacchetti che il nodo produce
- Forward chain = regole per i pacchetti che attraversano il nodo
- Default allow -> transformabile in default deny creando una lista dei pacchetti da instradare e piii aggiungere "drop all" alla fine.
- Azioni possibili: Drop, Route, Goto/return(call/ritorno ad una catena), Queue(trasmissione a codice utente), Log, Reject, Dnat/Snat/masquerade.

Nat table

- Prerouting chain = any input packet
- postrouting chain = any output packet
- NAT may change the address in a packet
- Applied before INPUT and after OUTPUT/FORWARD

Archittetura



Esempi:

- iptables -A INPUT -p UDP drop -> Aggiunta una regola che cancella tutti i pacchetti UDP.
- iptables -A INPUT -p TCP -dport 156 drop -> scarta ogni pacchetto TCP indirizzato alla porta 156.
- iptables -A INPUT -p tcp -dport 80 -j ACCEPT -> Permetti traffico web
- iptables -A INPUT .p tcp -dport 443 -j ACCEPT -> Permetti traffico HTTPS
- iptables -N newcontrol -> Crea una nuova catena dove poter inserire nuovi.

In base alle regole possiamo ad esempio vietare traffico HTTP e permette solo HTTPS

Role based access control (SO o applicazione)

Invece di ragionare sui soggetti si ragiona sui ruoli, cioè, profili professionali diversi.

La matrice specifica i diritti per i singoli ruoli, cioè, molte meno righe più semplice gestione.

è necessario una funzione (tabella) che mappi ogni utente in un ruolo e può essere acquisito dinamicamente e devono essere meccanismi per la transizione:

- | | | |
|--------------|----|-------------------|
| - Medico | -> | Medico di guardia |
| - Professore | -> | Rettore |

Molto popolato per database dove viene utilizzato per garantire un buon livello di privacy.

- Aministrativo -> Ospedale può accedere a tutte le info su costi e modalità di pagamento per ammalato ma non ad informazioni mediche.
- Sanitario -> Ospedale può accedere a tutte le informazioni mediche ma non ad informazioni sui costi.

SE - Linux

Una versione di Linux definita da NSA per permettere all'utente di definire una qualunque politica di sicurezza (DAC/MAC), la politica che il SO applica è un parametro di configurazione e Linux definisce i diritti utente sulle risorse.

Selinux definisce:

- I diritti di ogni programma sulle risorse
- I programmi che ogni utente può eseguire
- I diritti sono definiti in termini di tipo, ruoli e livelli
- Type1 può eseguire questa soluzione su type2
- Questo ruolo può eseguire il programma su questi tipi
- Esiste un controllo sul livello (ortogonale)

```
allow user_t bin_t : file {read execute write setattr}
```

Source type(s): il tipo del processo che vuole accedere all'oggetto (freccia su user)

Target type(s): il tipo dell'oggetto a cui si vuole accedere (freccia su bin)

Object class(es): La classe dell'oggetto.
(freccia su file)

Permission(s): il tipo di accesso alla classe di oggetti.
(freccia su execute)

questa è la dichiarazione dei diritti, le informazioni sul contesto di un soggetto sono contenute in un server e all'esterno viene trasmesso solo un puntatore al contesto.

La descrizione di una politica è complessa anche nel caso di politiche semplici.

ad esempio, per specificare la politica di linux, cioè per costruire un SELinux che implementi la politica di Linux richiede:

- 29 tipi
- 121 operazioni
- 27.000 regole

Il vero problema è lo scarso supporto per una descrizione ad alto livello da cui poi generare le varie regole e NON esistono strumenti per verificare la coerenza di una politica.

CONTROMISURE

Programmazione robusta

Adottare uno stile di programmazione per minimizzare vulnerabilità o gli impatti di una vulnerabilità.

Un possibile insieme di principi da utilizzare:

1. Validare gli input del programma aka input is evil
2. Impedire buffer overflow
3. Implementazione robusta limitando informazioni trasmesse all'esterno tipo:
 1. Puntatori ogici e non fisici
 2. Controllo di informazioni trasmesse e ricevute.
4. Controllare i valori trasmessi ad altre funzioni (egress filtering)
5. Controllare i risultati ricevuti

Errori da evitare:

Top 25 most Dangerous Software Weaknesses:

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-782	Out-of-bounds Write	63.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	18.81	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-114	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-284	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-302	Deserialization of Untrusted Data	6.71	+8

Selezionate in base alle vulnerabilità nel NVD. Lo score è il prodotto della frequenza del problema nelle vulnerabilità per lo score assegnato da CVSS (= frequenza x rischio generato).

Rank	ID	Name	Score	2020 Rank Change
[14]	CWE-287	Improper Authentication	6.58	0
[15]	CWE-475	MNULL Pointer Dereference	6.54	-2
[16]	CWE-738	Use of Hard-coded Credentials	6.27	+4
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84	-12
[18]	CWE-852	Missing Authorization	5.47	+7
[19]	CWE-278	Incorrect Default Permissions	5.09	+22
[20]	CWE-208	Exposure of Sensitive Information to an Unauthorized Actor	4.74	-13
[21]	CWE-322	Insufficiently Protected Credentials	4.21	-3
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2	-6
[23]	CWE-411	Improper Restriction of XML External Entity Reference	4.03	-4
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78	+3
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58	+6

Firewall: una estensione di ACL su linee

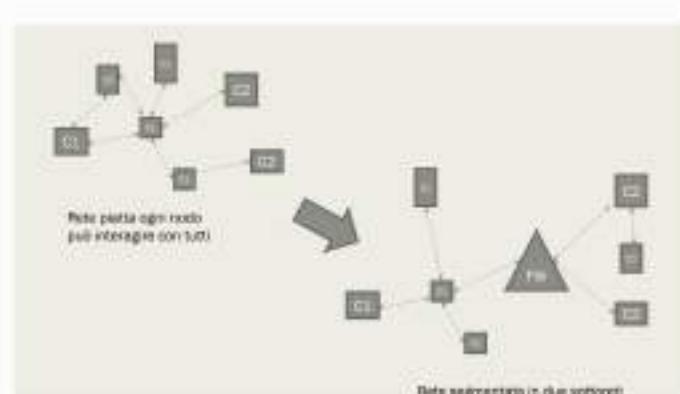
Componente che separa due reti di diversa criticità e concentra tutte le linee di ingresso/uscita tra le due reti e filtra le comunicazioni per implementare una certa politica di sicurezza. Il firewall è un meccanismo per implementare una certa politica di sicurezza e possiamo descrivere un firewall in base a due caratteristiche ortogonali:

- I protocolli che conosce e che filtra
- La sua implementazione

Il primo attributo determina il tipo di controlli che il firewall riesce ad implementare, il secondo attributo determina la robustezza del firewall.

Un firewall rappresenta un punto di centralizzazione delle comunicazioni e quindi un potenziale collo di bottiglia per le prestazioni.

Un firewall non può controllare le linee che non lo attraversano.



FIREWALL I CONTROLLI

In firewall per protocolli dello stack TCP/IP i controlli dipendono dal livello dello stack a cui operano.

Un firewall che lavora a livello 3 = IP packet inspection opera solo sui bit nell'header del pacchetto e può controllare:

- Mittente
- Destinatario
- Porte
- Protocollo usato
- Origine di connessioni di livello 4

In pratica è molto simile a quanto abbiamo visto in precedenza per il router.

Un firewall che lavora a livello 4 (TCP) è un circuit level gateway.

Opera su connessioni TCP è in grado di controllare che la connessione esista ed il contesto della comunicazione può anche controllare contenuto stateful inspection.

I firewall di livello più alto sfruttano la conoscenza del protocollo applicativo e possono anche scoprire attacchi che sfruttano vulnerabilità di questo protocollo.

Circuit Level Gateway

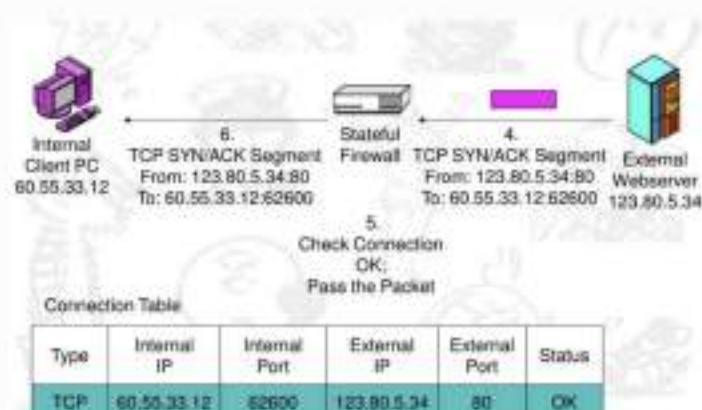
Un firewall a livello di circuito convalida che un pacchetto sia a richiesta di connessione o un pacchetto di dati appartenente a una connessione o circuito virtuale tra due livelli di trasporto pari. Oltre a consentire o meno i pacchetti, il firewall a livello di circuito determina anche se la connessione tra le due estremità è valida secondo delle regole configurabili su possibili iterazioni.

Per convalidare una sessione, un firewall a livello di circuito esamina ogni configurazione di connessione per garantire che la connessione per garantire che la connessione segue un handshake TCP legittimo e non inoltrerà i pacchetti di dati fino al completamento dell'handshaket.

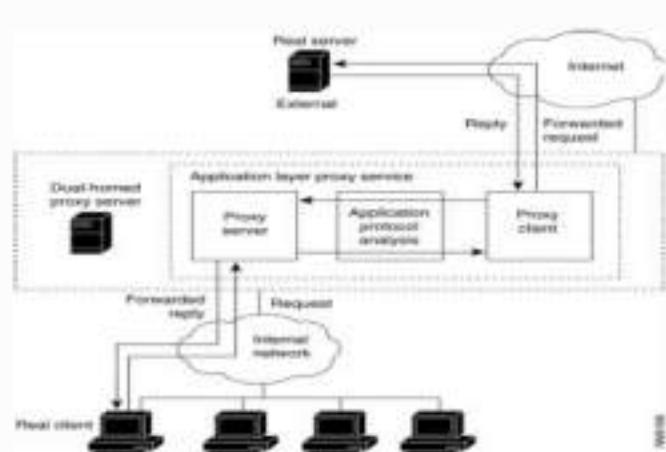
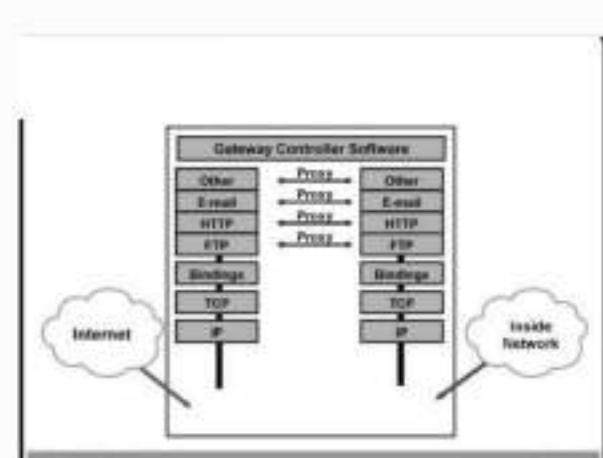
Qualsiasi informazione passata a un computer remoto attraverso un firewall a livello di circuito potrebbe sembrare provenuta da un gateway. Questo è utile per nascondere le informazioni sul gateway protetto.

Un firewall a livello di circuito mantiene una tabella del circuito virtuale (includendo lo stato completo della sessione e le informazioni sulla sequenza) di connessioni valide e consente pacchetti di rete contenenti dati quando le informazioni sui pacchetti di rete corrispondono a una voce nella tabella. Una volta terminata la connessione, i firewall rimuove la voce della tabella corrispondente.

Il filtraggio a livello di circuito presenta un vantaggio rispetto al filtraggio di pacchetti perché può compensare le carenze del protocollo UDP che non convalida mai l'indirizzo di origine. Ciò rende lo spoofing IP molto più difficile (importante per DNS, perforazione UDP).

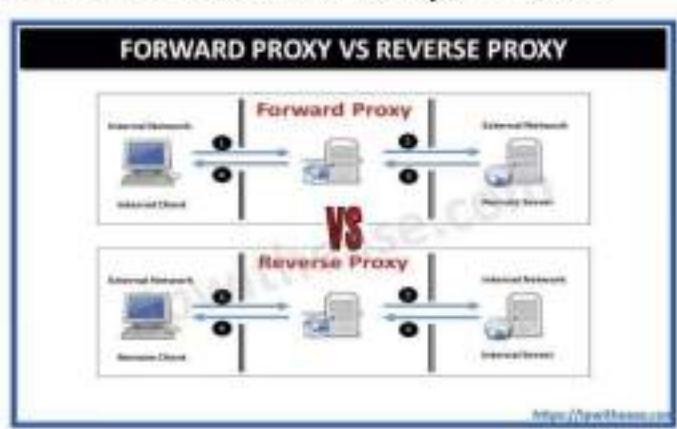


Application gateway



Proxy vs reverse proxy

Un proxy protegge i client da attacchi esterni, invece, un reverse proxy protegge i server interni da attacchi da client esterni.
La struttura può essere simile ma i controlli sono diversi.
Un reverse proxy può avere anche compiti di load balancer nel caso di più istanze del server disponibili.



Stateful packet filters, or, stateful firewall

Si tratta della tecnologia più comune.

Funziona a livello di rete e fornisce funzionalità di filtraggio dinamico dei pacchetti. L'ispezione con stato tiene traccia di ogni connessione che attraversa qualsiasi interfaccia del firewall e ne conferma la validità.

Mantiene una tabella di stato che tiene traccia di tutte le sessioni e controlla tutti i pacchetti che passano attraverso il firewall. Se i pacchetti hanno le proprietà reviste dalla tabella di stato, il firewall consente loro di passare. La tabella degli stati cambia dinamicamente in base al flusso di traffico.

I firewall con stato operano ai livelli 3,4 e 5. Da un livello di trasporto, il firewall esamina le informazioni nelle intestazioni dei pacchetti di livello 3 e dei segmenti di livello 4. Ad esempio, il firewall esamina l'intestazione TCP per SYN,RST,ACK,FIN e altri codici di controllo per determinare lo stato della connessione.

Quando si accede a un servizio esterno, il firewall "ricorda" alcune richeiste nella tabella di stato. Quando il sistema esterno risponde, il firewall confronta i pacchetti ricevuti con lo stato salvato per consentire o negare l'accesso alla rete.

Next Generation Firewall

Un tipico NGFW combina l'ispezione dei pacchetti con l'ispezione dello stato e include anche alcuni tipi di ispezione approfondita dei pacchetti, DPI, nonché altri sistemi di sicurezza della rete, come IDS/IPS, filtro antimalware e antivirus.

Mentre l'ispezione dei pacchetti nei firewall tradizionali esamina esclusivamente l'intestazione del protocollo del pacchetto, DPI esamina i dati effettivi che il pacchetto sta trasportando.

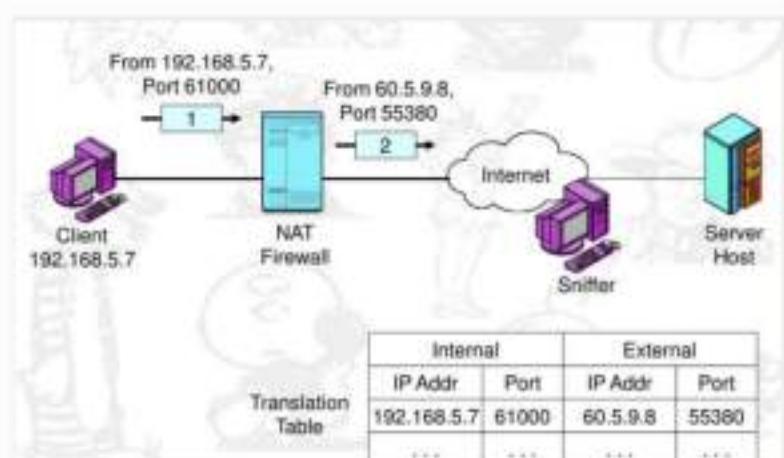
Un firewall DPI tiene traccia dell'avanzamento di una sessione di navigazione web e può notare se un payload di pacchetto, quando assemblato con altri pacchetti in una risposta del server HTTP,

costituisce una risposta formattata HTML legittima. Per trarre il massimo vantaggio, le organizzazioni devono integrare gli NGFW con altri sistemi di sicurezza, di solito un processo complesso. Più costoso di altri tipi di firewall.



NAT Firewall

La ragione per cui non arriverà mai IPV6 dato che non c'è più carenza di indirizzi ip.



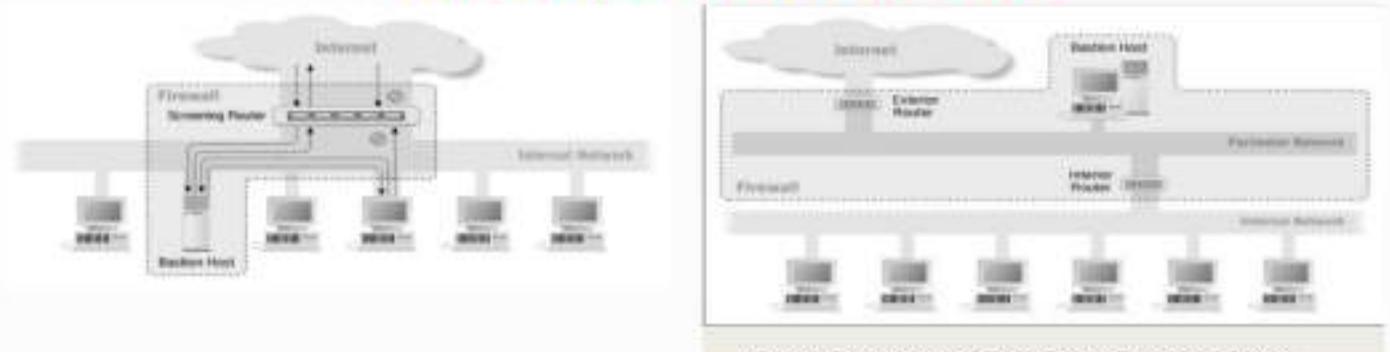
Intrusion Prevention e Egress Filtering

Firewall può interrompere una connessione (un circuito) nel momento in cui analisi del contenuto rivela che c'è una intrusione in atto.

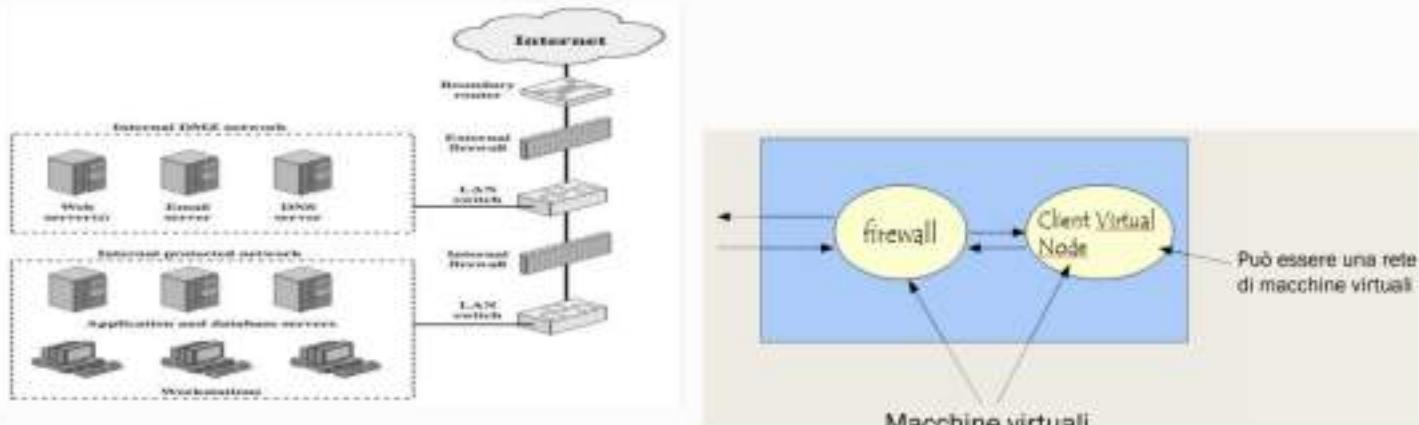
Egress filtering = filtraggio delle comunicazioni in uscita e non solo di quelle in ingresso.

Un meccanismo fondamentale per rilevare intrusioni non ancora scoperte e che hanno connesso alcuni nodi protetti ad una botnet. E' anche utile per rilevare errori di programmazione come già visto. Egress filtering aumenta ingiene dell'ecosistema informatico e permette di prevenire attacchi di tipo DDOS (zombie) o invio di spam.

FIREWALL ARCHITETTURE



Aumenta ridondanza ed essendo ridondanza sui controlli aumenta la robustezza complessiva



Honeypot/honeynet

Un sistema/rete la cui utilità sta solo nell'essere attaccato al posto di un altro.

Abbiamo due ragioni per installare un honeypot/honeynet:

- Scoprire come un sistema viene attaccato (threat intelligence, scoperta di vulnerabilità e di modi di attacco).
- Rallentare e scoprire un attaccante deviandolo verso un sistema che non provoca un danno.

Honeypot = Un sistema progettato appositamente per essere attaccato e che comprende meccanismi per ricordare il comportamento dell'attaccante e fermarlo. Obiettivo è di rallentare attaccante e scoprire modus operandi dell'attaccante stesso.

Ci sono modi alternativi di classificazione:

- In base al livello di iterazione
 - Bassa: Un semplice port listener è considerato un'iterazione estremamente bassa, perché, dopo la connessione, l'attaccante non può fare nient'altro.
 - Medio: un servizio emulato che analizza le comunicazioni e restituisce risposte simulate per comportarsi come un servizio reale.
 - Alta: richeide l'uso di servizi reali, ma ingannevoli, host completamente operativi o reti complete per ingannare.
- In base al livello di implementazione
 - Virtuale
 - Fisico
- In base all'obiettivo
 - Produzione
 - Ricerca

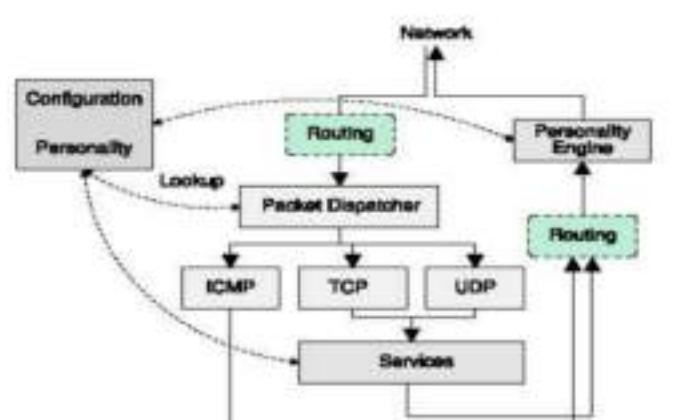
Honeynet = una rete vera collegata ad un sistema reale per essere attaccata al posto di quella vera. Il software usato è quello vero. Le sue caratteristiche sono: simulazione di reti di grosse dimensioni, elevata confirabilità, supporto di router multipli per la gestione di più reti, integrazione tra reti fisiche e reti virtuali, rilevamento di attività non autorizzate all'interno di una rete, attraverso il monitoraggio degli indirizzi IP non autorizzati di uno specifico internallo "dark space", e ogni connessione verso il dark space viene considerato come un possibile attacco o comunque come il risultato di una scansione, esamina qualsiasi attività relativa alle porte TCP e UDP nonché al traffico ICMP e emula servizi, utilizzando script in Perl, shell o altro in grado di interagire con attaccante.

Archittetura:

- Un database delle configurazioni
- Un packet dispatcher
- Un gestore di protocolli
- Un personality engine
- Un componenti di routing opzionale

Passi:

1. Packet dispatcher processa i pacchetti in arrivo e controlla indirizzo IP e l'integrità.
2. Accetta solo TCP, UDP e ICMP. Tutti gli altri pacchetti vengono scartati.
3. Interroga il file di configurazione alla ricerca del modello associato a IP di destinazione.
4. E' possibile interagire con applicazioni esterne capaci di gestire il flusso dati.
5. E' supportato un meccanismo di reindirizzamento delle connessioni grazie al quale è possibile girare una richiesta di servizio ad un sistema reale.
6. Il personality engine elabora un pacchetto di risposta e lo modifica per renderlo compatibile con lo stack utilizzato dal sistema operativo simulato.



RILEVAZIONE DI INTRUSIONI

Intrusion detection system

Un modulo/nodo del sistema che esamina informazioni e stato dei vari moduli per scoprire un attacco di una intrusione.

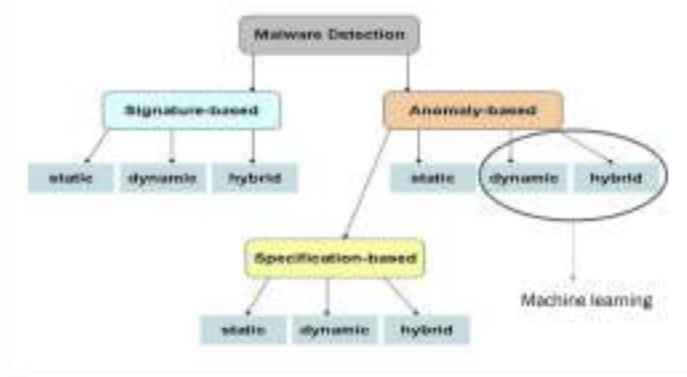
Non scopre l'intrusione che richiede una correlazione tra i vari attacchi, cioè i passi di una intrusione.

due casi fondamentali:

- Local intrusion detection system = esamina le informazioni su un singolo nodo del sistema per scoprire attacchi contro il nodo.
- Network intrusion detection system = esamina le informazioni scambiate/ricevute dai nodi di una sottorete per scoprire attacchi contro uno dei nodi della sottorete.

Un network IDS è fondamentalmente uno sniffer che cattura pacchetti e poi li esamina eventualmente ricostruendo i protocolli di livello superiore.

Cattura trasparente ai nodi che iteragiscono pone immediatamente il problema della banda di elaborazione di IDS e di eventuali pacchetti persi.



Anomali based

Il comportamento del sistema da proteggere viene osservato per un intervallo di tempo (apprendimento del comportamento normale). Richiede un tempo variabile da settimane a mesi (NON è out-of-the-box).

Apprendimento riguarda:

- Invocazioni ai servizi
- Logging di utenti
- Richieste di utenti
- Volumi di richieste e di traffico

Dopo l'apprendimento, qualsiasi comportamento troppo "distante" da quelli che sono stati osservati e appresi viene segnalato come anomalia.

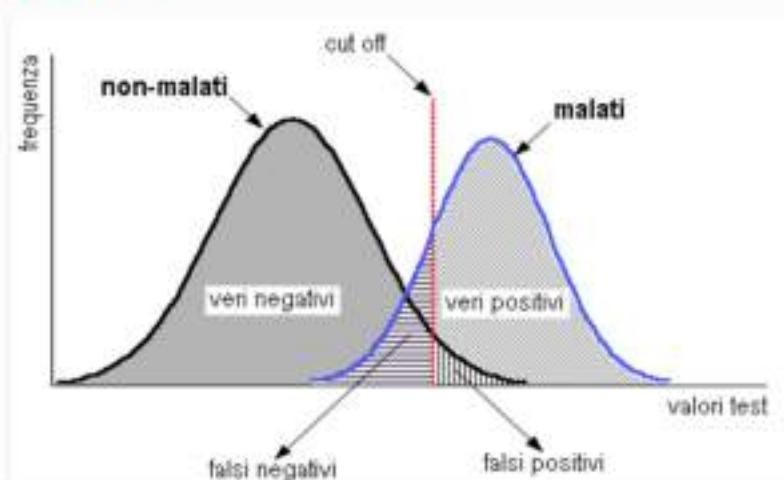
L'elemento critico è la quantità di informazioni sul sistema acquisite nella fase di apprendimento che continua nel tempo perché il comportamento cambia.

Si divide in:

- **Dinamico**: Le informazioni sul comportamento di un programma vengono raccolte dalle esecuzioni e quindi confrontate con il comportamento.
- **Statico**: Le informazioni sul comportamento atteso sono prodotte da un'analisi statica della struttura di un programma. Questo è l'unico apprendimento che non richiede osservazione e quindi tempo.
- **Ibrido**: Il comportamento atteso del programma viene raccolto per colmare il divario dovuto a un'analisi statica e l'output viene confrontato con il comportamento.

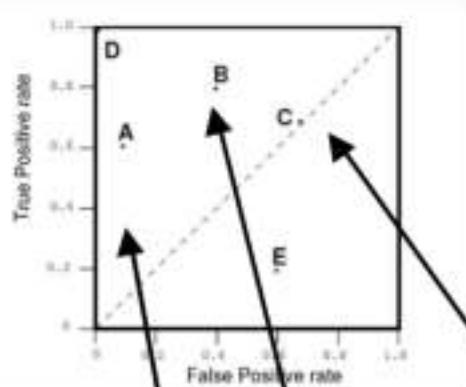


Falsi positivi



In molti casi si usano regole che confrontano una misura con una soglia e stabiliscono esistenza di intrusione in base a confronto. La soglia determina i valori che vengono considerati malattia.

Curva Receiver Operative Characteristics



Ad ogni regola per rilevare un'intrusione invia almeno x Mb/sec e apre almeno x connessione in un sec, possiamo accoppiare un punto in questo spazio secondo la probabilità di falso e vero positivo, per ogni valore di x .

Al variare di x , abbiamo una curva nello spazio ROC.

Una regola bassa e sinistra = conservativo basso numero di falsi positivi ma anche una bassa capacità di rilevamento.

Una regola alta e giusta = buona capacità di rilevamento e scapito

di molti falsi positivi e pochi veri positive.
Una regola sotto la bisettrice = peggio che casuale (= la bisettrice) può essere migliorato negandolo.

Specification based

I comportamenti normali non vengono appresi dall'esecuzione, ma sono specificati dalla politica di sicurezza.

- **Dinamico:** Le informazioni sul comportamento del programma vengono raccolte e confrontate con le specifiche del programma.
- **Statico:** Un programma viene analizzato staticamente per calcolare le specifiche e il comportamento viene confrontato con le specifiche.
- **Ibrido:** Il compilatore crea una specifica e le osservazioni vengono raccolte per essere confrontate con il comportamento del programma integrando quei casi che non possono essere risolti staticamente.

Signature based

Idea principale: alcuni comportamenti e dati (costanti) che caratterizzano e identificano completamente un malware = la firma del malware.

Tutte le firme vengono raccolte in un database che guida il rilevamento. Ciò pone 2 problemi:

- La scoperta di una firma
- L'aggiornamento della banca di dati

Un malware può essere rilevato solo se la firma è nota, cioè, non è possibile rilevare un exploit 0-Day.

Per scoprire nuovi attacchi sfruttiamo il rilevamento delle anomalie o informazioni di threat intelligence sulle minacce ricavate da un honeypot.

Si possono usare strategie alternative per definire la firma, questo approccio tradizionale viene esteso per matching parziali o scoprendo comportamenti anomali che sono risolti mediante upload del codice sospetto nel sito (cloud) del produttore dello strumento.

Meccanismo fondamentale è default allow, cioè, tutto ciò che differisce da una firma è consentito.

- Dinamico: le informazioni sul comportamento del programma vengono raccolte (anche da un'esecuzione protetta in un altro ambiente o in sandbox) e confrontate con la firma.
- Statico: Il codice del programma viene analizzato e confrontato con la firma, il meccanismo base di antivirus più semplici.
- Ibrido: Unisce i 2 approcci: un'analisi statica seleziona i

programmi sospetti e il comportamento di questi programmi viene monitorando durante una esecuzione in ambiente protetto.

WORM

Prodotti diversi hanno signature diverse

- Signature Dragen Scanner
T B T B 10 0 W IDS296/web-misc http-whisker-splicing-attack-space /20
 - Defensivew Signature
J H 6 T 0 80 |IDS296/web-misc_http-whisker-splicing-attack-space| "20"
 - Paketmon Signature
IDS296/web-misc_http-whisker-splicing-attack-space tcp * 80 "(20)"
 - Shoki Signature
tcp and (dst port 80) and (ip[2:2] > ((ip[1] & 0x0f) + (ip[12:1] & 0x0f)) + 5) and
(ip[13:16] > 0x5555) & SEARCH IDS296 web-misc http-whisker-splicing-attack-space 0x2F ALL
NULL

Signature locale e di rete

Una signature locale è formata da una sequenza di:

- Informazioni in un log
 - istruzioni in memoria
 - invocazioni al sistema operativo

Una **signature di rete** è una sequenza di informazioni che si possono trovare in uno o più pacchetti IP o in una connessione TCP.

In questo caso network IDS fa una ipotesi su come i pacchetti verranno ricevuti ed elaborati dal destinatario.

Network evasion (una delle colonne di MITRE att&ck matrix)
inserire pacchetti fake per evadere i controlli di IDS tipo: Bad checksum, time-to-live ecc.

Statico vs Dinamico

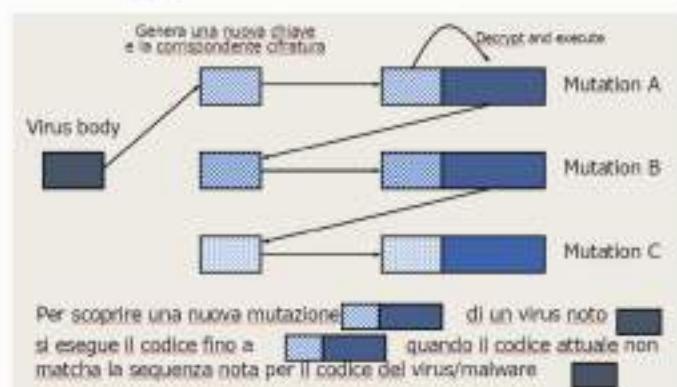
Scanner antivirus semplici tipo: Cerca firme (frammenti di codice virus noto), Euristica per il riconoscimento del codice associato al virus malware, es: virus polimorfici utilizzano spesso cicli di decrittazione o controllo dell'integrità per rilevare le modifiche ai file. Ricorda le dimensioni dei file, checksum, gli HMAC con chiave dei contenuti.

Soluzioni dinamiche con decrettazione ed emulazione generica, tipo: Carica il codice su un sistema remoto, Il sistema emula l'esecuzione della CPU per alcune centinaia di istruzioni, riconosce il corpo del virus noto dopo la decrittazione e non funziona molto bene contro virus con corpi mutanti e virus che non si trovano vicino all'inizio dell'eseguibile infetto.

Polimorfismo di malware e virus

- Malware/virus crittografati: decryptor costante seguito dal malware/corpo del virus crittografato.
- Malware/virus polimorfici: ogni copia crea una nuova crittografia casuale dello stesso corpo del virus codice decryptor costante e rilevabile.

Nota storica: Il virus "crypto" ha decrittografato il proprio corpo mediante una ricerca con chiave di forza bruta per evitare il codice esplicito del decryptor.



Zmist

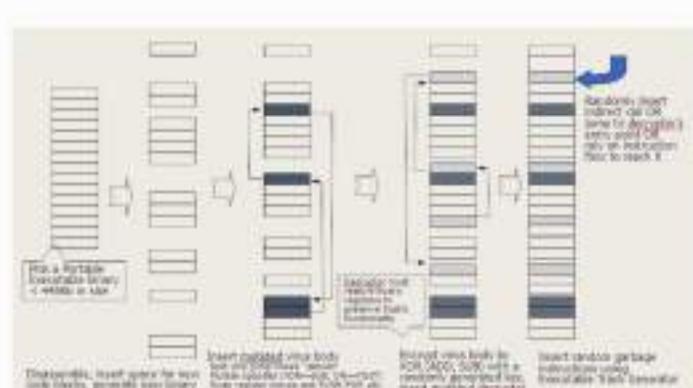
Progettato nel 2001 dallo scrittore di virus russo Z0mbie.

Tecnica: integrazione del codice del virus si fonde nel flusso di istruzioni del suo host.

Le isole "isole" di codice sono integrate in posizioni casuali nel sistema host e collegati da salti.

Quando/se il codice del virus viene eseguito, infetta ogni eseguibile disponibile.

Un punto di ingresso del virus inserito in modo casuale può non essere raggiunto in una particolare esecuzione.



Sandbox

Un ambiente sicuro per l'apertura dei file sospetti, l'esecuzione di programmi non attendibili o il download degli URL, senza compromettere i dispositivi in cui si trovano.

Separa e confinata da ambiente normale, spesso eseguita su cloud. Permette di esaminare in modo sicuro un file o un codice potenzialmente dannoso prima di utilizzarlo, tenendolo isolato dalla rete da difendere.

Nella sandbox il software può fare qualsiasi cosa ma non può accedere a risorse del sistema.

Stiamo sostanzialmente parlando di una evoluzione del concetto di MV.

Usata anche come strumento di difesa e non solo di analisi.
Meccanismo potenzialmente pericoloso perché se software esce da sandbox sistema è completamente indifeso e il malware sofisticato è in grado di capire che viene eseguito in sandbox e si comporta in modo non pericoloso, se esce il sistema è indifeso.

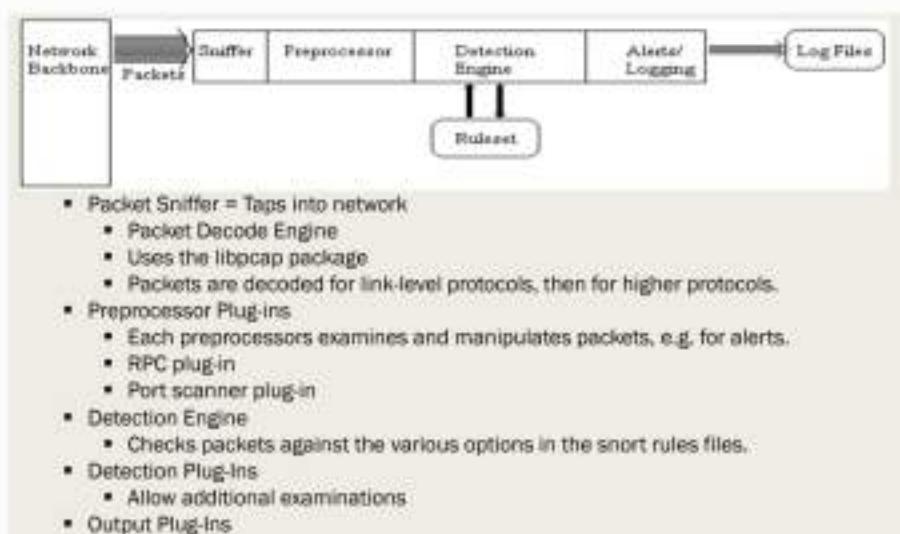
Rule based

Un modo di generalizzare idea di signature è quello di utilizzare regole che matchano i pacchetti.

In questo modo produciamo regole più astratte che possono gestire anche modifiche nel contenuto dei pacchetti.

Un IDS di questo tipo è Snort:

- Un buon sniffer (legge i pacchetti di rete)
- Un motore di rilevamento, basato su regole
- I pacchetti che non corrispondono a nessuna regola vengono scartati (solo dall'analisi in generale) o vengono registrati
- I pacchetti di corrispondenza delle regole possono anche attivare un avviso.



Regole Snort

- Sono divise in due sezioni logiche,
 - l'intestazione della regola
 - opzioni della regola.
- L'intestazione (header) della regola contiene
 - azione della regola,
 - il protocollo,
 - gli indirizzi IP di origine e destinazione
 - Le maschere di rete
 - Le informazioni sulle porte di origine e destinazione.
- La sezione dell'opzione della regola contiene
 - messaggi di avviso
 - informazioni su quali parti del pacchetto devono essere

```
alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 any (flags: SF; msg: "SYN-FIN scan")
```

ispezionate per determinate se l'azione della regola deve essere intrapresa.

- alert: genera un alert e logga pacchetto
- log: logga pacchetto
- pass: ignora pacchetto
- activate: alert e poi attiva una regola dinamica (ricorda stato)
- dynamic: idle fino a quando non viene attivata poi logga pacchetti
- drop: blocca e logga il pacchetto (un filtro e non solo uno sniffero)
- reject: blocca e logga il pacchetto e poi invia
 - TCP reset se TCP
 - ICMP port unreachable se UDP
- sdrop: blocca il pacchetto senza loggarlo

Azioni Snort

Ordine delle regole

Un pacchetto dovrebbe essere controllato nell'ordine:

Drop > Pass > Alert > Log

Questo schema è più sicuro poiché nessun pacchetto passa senza essere verificato rispetto a tutte le regole di drop. Tuttavia, la maggior parte dei pacchetti è traffico normale e non mostra alcune attività di intruso.

Testare tutti i pacchetti rispetto a tutte le regole di avviso richiede molta potenza di elaborazione.

Un ordine più efficiente, ma è la più pericolosa è:

Pass > Drop > Alert > Log

VPN

Una overlay network che emula una connessione sicura sopra una rete pubblica e quindi non sicura.

Connette delle sottoreti, cioè, reti locali ed è soggetta a DDOS perché può scoprire garbage solo decifrando.

Assumendo che ogni rete locale sia protetta da un firewall la vpn cifra ed autentica tutto il traffico tra ogni coppia di firewall
//Immagine.

VPN - IPSEC

Una estensione di IPv4 che cifra e autentica flussi di informazioni, da usare fino a quando IPv4 verrà sostituito da IPv6 (se mai succederà).

Esistono altre soluzioni che offrono sicurezza ad altri livelli della pila OSI (PGP, HTTPS,SSL, ecc).

IPSEC ha 2 possibili comportamenti o protocolli:

- Authentication Mode = authentication header
- Encapsulated Security Payload = encryption delle informazioni.

Entrambi i protocolli possono essere usati in uno di due modi:

- Transport Mode = si inseriscono nuovi campi nel pacchetto originale.

- Tunnel Mode = il pacchetto IP è protetto e diventa il contenuto informativo di un nuovo pacchetto.

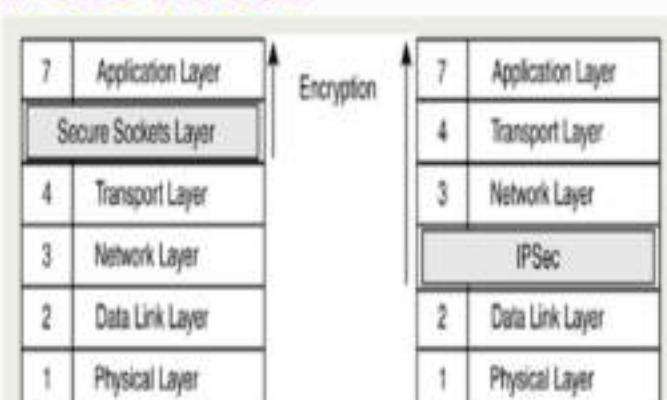
Ogni connessione è protetta mediante crittografia simmetrica perché più efficiente e perché gli elementi constitutivi della crittografia simmetrica si adattano bene ad hardware attuale (spostamento di bit, XOR, ricerche di tabelle). mentre i vari nodi che interagiscono possono essere dotati di coppia di chiavi pubblica e private.

Un protocollo per scambio iniziale protetto da crittografia asimmetrica per stabilire connessione sicura tra i nodi perché si basa generalmente su calcoli matematici di grandi dimensioni (RSA) o addizione/moltiplicazione di punti (ECC) più lente di crittografia simmetrica.

Le chiavi simmetriche vengono aggiornate periodicamente

- Per limitare gli impatti di un attacco che scopre una chiave.
- Vari algoritmi hanno vulnerabilità che possono essere sfruttate quando vengono trasmesse particolari configurazioni di bit e la probabilità di successo di questo attacco aumenta con il volume dei dati trasmessi.

IPSEC VS SSL



Complessivamente definisce 4 nuovi protocolli

- **AH**(Authentication Header) autentica i partner e protegge integrità
- **ESP**(Encapsulating Security Payload) garantisce la confidenzialità proteggendo il contenuto informativo della comunicazione.
- **IKE**(Internet Key Exchange) due partner concordano sulla chiave da usare e per il tempo da usare la chiave.
- **ISAKMP**(Internet Security Association and Key Management Protocol) usata per stabilire ed aggiornare "**Security Association (SA)**" e i suoi attributi.

Una security (SA) descrive una connessione diretta con i servizi associati al traffico che la connessione trasmette.

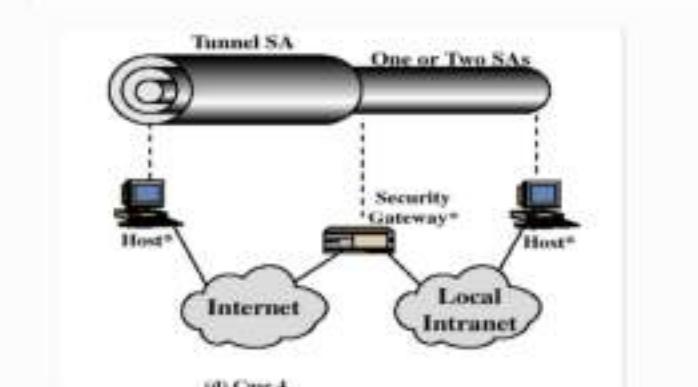
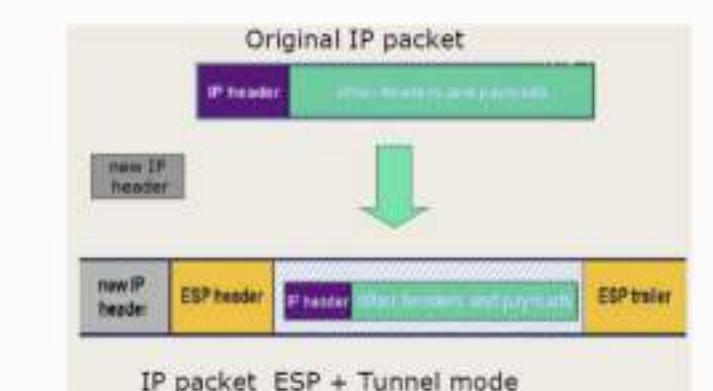
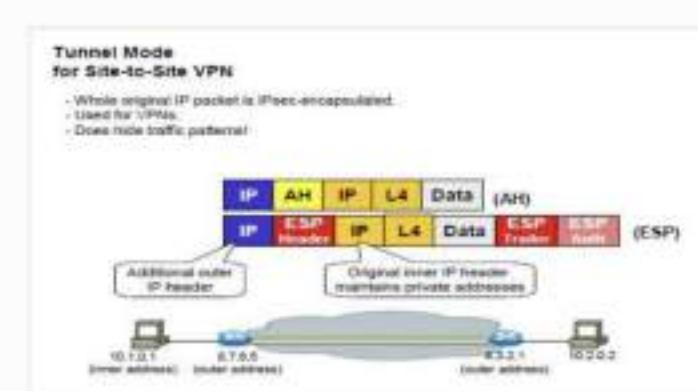
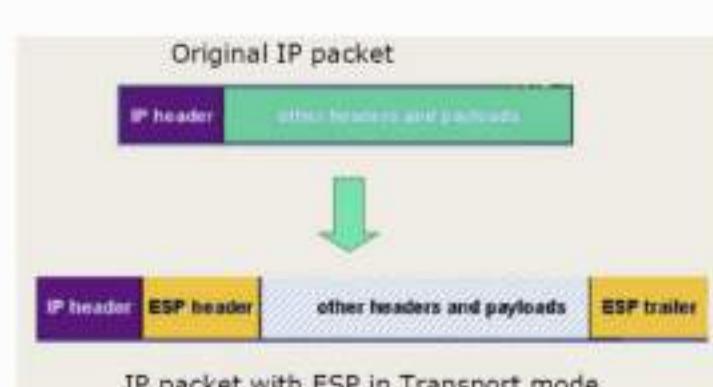
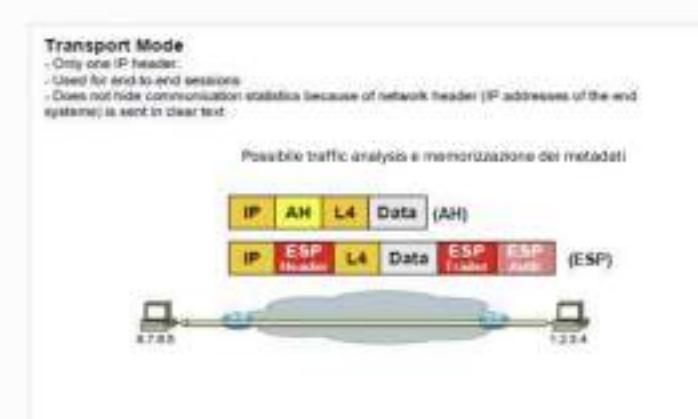
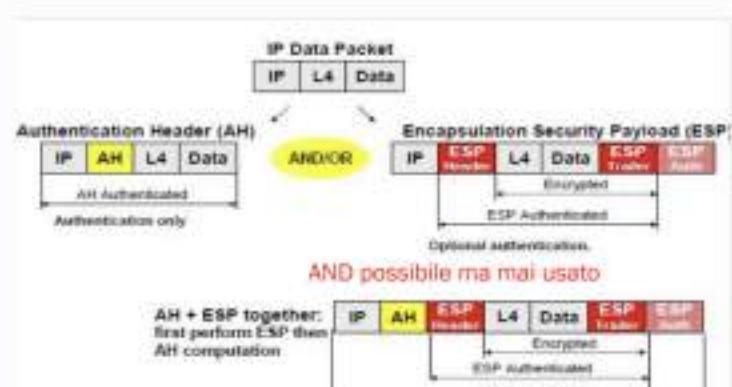
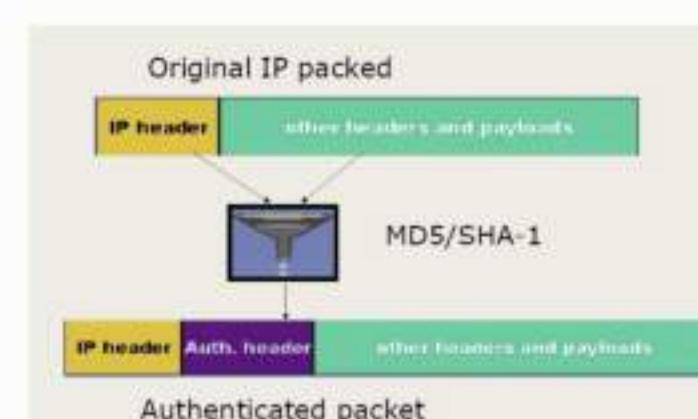
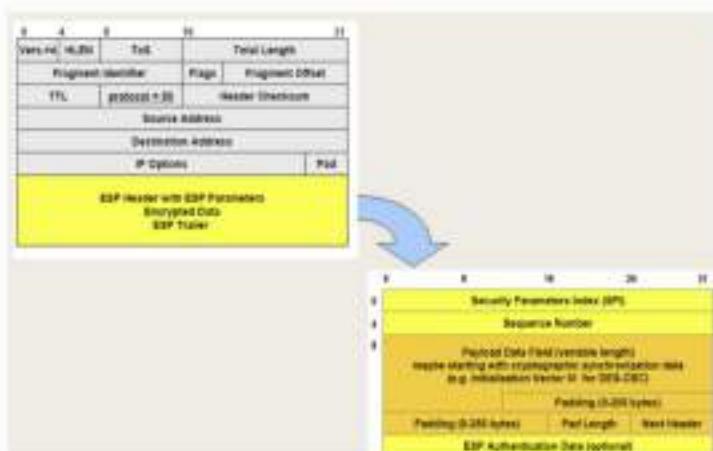
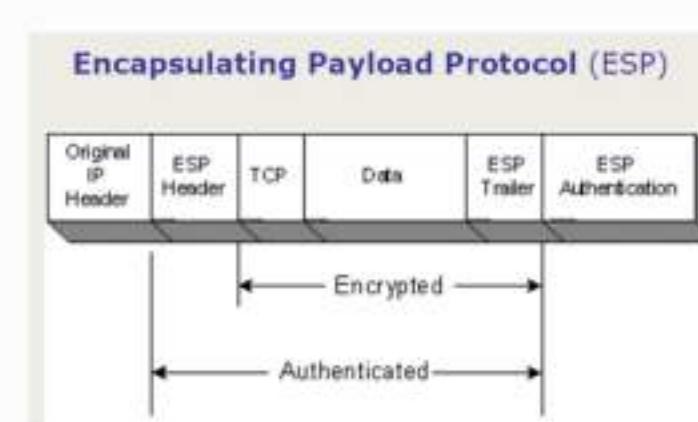
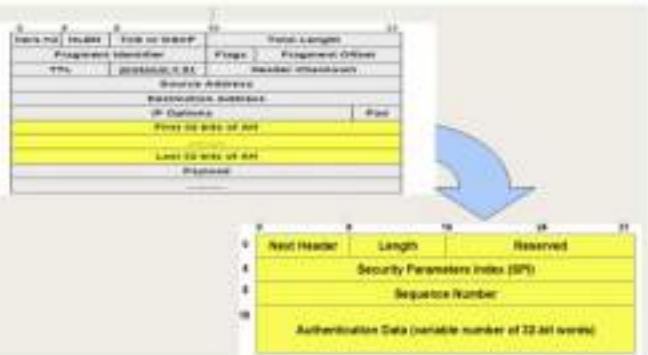
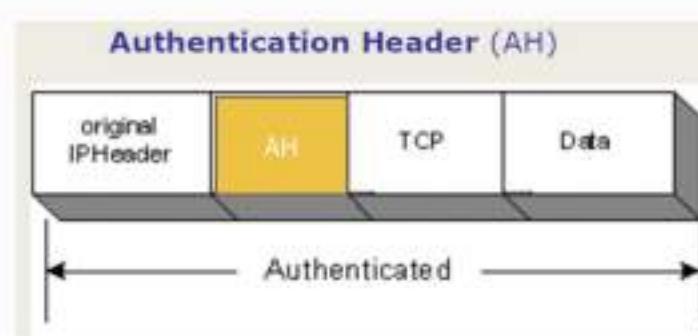
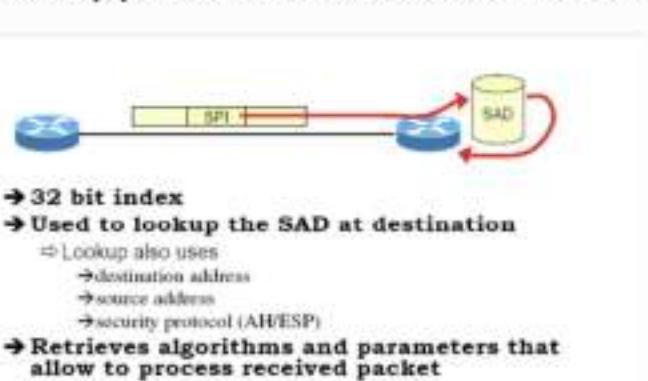
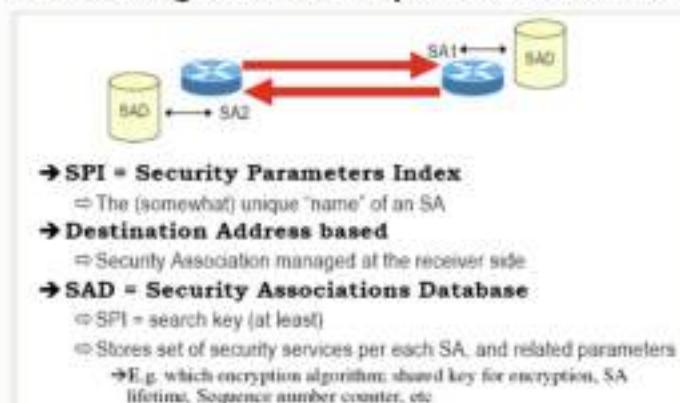
Per assicurare una connessione bidirezionale sono richieste due SA, una per ogni direzione.

La SA comprende anche tutte le informazioni per rendere sicura la comunicazione.

I servizi di sicurezza di una SA sono implementati per rendere sicura la comunicazione.

I servizi di sicurezza di una SA sono implementati mediante AH o

ESP. In generale i protocolli non sono applicati simultaneamente.



END POINT DETECTION AND RESPONSE

EDR o endpoint threat detection and (ETDR) è una contromisura per degli endpoint integrata che combina il monitoraggio continuo in tempo reale e la raccolta dei dati degli endpoint con funzionalità di analisi e risposta automatizzate basate su regole.

Il termine descrive i sistemi di sicurezza emergenti che rivelano e indagano su attività sospette su host ed endpoint, impiegando un'alta automazione per consentire di identificare e rispondere rapidamente alle minacce.

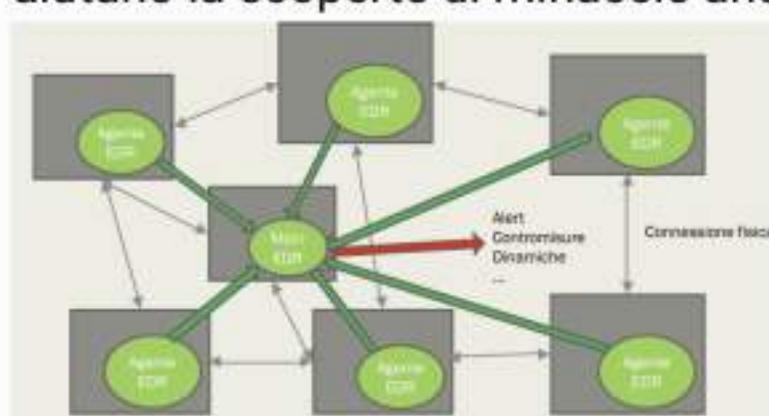
Le funzioni primarie di un sistema di sicurezza EDR sono:

- Monitora e raccogli i dati sulle attività dagli endpoint che potrebbero indicare una minaccia.
- Analizza questi dati per identificare i modelli di minaccia.
- Rispondi automaticamente alle minacce identificate per rimuoverle o contenerle e avvisa il personale di sicurezza.
- Usa strumenti forensi e di analisi per ricercare le minacce identificate e ricercare attività sospette

Un centro raccolta integrato per raccogliere, analizzare e correlare informazioni e dati su endpoint e coordinare allarmi e risposte alle minacce. tre aspetti fondamentali.

- **Endpoint data collection agents.** agenti software che monitorano sistemi, raccolgono informazioni su attacchi e anomalie e le trasmettono ad un db centrale.
- **Un motore analitico che opera in tempo reale ed analizza i dati raccolti.** Questo engine scopre pattern nei dati che rivelano un attacco o una minaccia nota e attiva una risposta immediata quale sconnettere un utente o chiudere una connessione o inviare un alert. Basato su regole contestuali e IOC ma elemento fondamentale è adversary emulation per ricostruire gli attacchi in una intrusione.
- **Analisi e forensics.** EDR può comprendere una analisi forense in tempo reale per individuare minacce che non ricadono nei pattern noti o strumenti per condurre una autopsia di un attacco già terminato. Anche qui adversary emulation aiuta a scegliere i nodi da investigare.

Gli strumenti di forensics permettono di investigare attacchi e breach passati per scoprire il funzionamento di exploit e come sono stati sconfitti eventuali strumenti per la sicurezza. Inoltre aiutano la scoperta di minacce ancora nel sistema.



EDR e Antivirus/Antimalware

Le soluzioni EDR più moderne possono essere considerate come un insieme di tutti i programmi antivirus tradizionali, ma ancora più efficaci. Quindi un antivirus fa parte di una soluzione EDR ma esistono anche altre funzioni.

Le funzioni di base di antivirus sono scansione, rilevamento e rimozione dei virus, mentre l'EDR svolge molte altre attività. Oltre all'antivirus, l'EDR può includere diverse funzioni, tra cui il monitoraggio ed operazioni consentite e vietate, tutte progettate per fornire una protezione più completa contro minacce note ed emergenti.

Poiché il perimetro della rete digitale si è esteso ed è ovunque, l'antivirus tradizionale non è più in grado di proteggere tutti i dispositivi utilizzati per accedere alle risorse aziendali. I sistemi Endpoint Detection and Response sono più adatti per proteggere dagli attacchi informatici avanzati e la loro risposta automatizzata riduce lavoro di difensori nel tentativo di proteggere le organizzazioni degli attacchi.

EDR open source

- OSSEC
 - Scansione degli endpoint e analisi dei dati di log provenienti da più endpoint.
 - Rilevamento di malware e rootkit con scansione a livello di processo e di file.
 - Risposta attiva utilizzando i criteri del firewall.
 - L'inventario del sistema recupera dati come gli ascoltatori, informazioni sull'hw, sw installato, tasso di utilizzo e servizi di rete
- TheHiveProject
 - Il dashboard dinamico offre la protezione con password per gli archivi RAR o ZIP, l'importazione di archivi zip possono contenere dati sospetti o malware
 - Opzioni di filtraggio avanzate per creare avvisi personalizzati e fornire filtri e facili esportazioni.
 - Forensics e risposta agli incidenti significa una panoramica di IP, URL, indirizzi, nomi di dominio, hashtag e file.
 - Analisi incrociata dei rapporti sugli incidenti, utilizzando servizi web come VirusTotal.

Reagire ad un attacco

2 tipi possibili di reazione:

- Reazioni sul sistema attaccato = aggiungere contromisure.
- Reazione sul sistema da cui siamo attaccati.

il secondo tipo di reazione è sempre da evitare poichè attaccante usa una infrastruttura di command & control costruita con intrusioni precedenti.

Nodi di command & control sono sacrificabili per attaccare.

Unica reazione utile è quella di smantellare infrastruttura ma questo richiede:

- Scoperta dei nodi di infrastruttura
- Eliminazione sincronizzata dei vari nodi di infrastruttura.

Stiamo parlando di offensive security tecnica che ha molti problemi

- Legali: stiamo attaccando nodi di un soggetto terzo che non possiamo informare.
- Tecnici: come attaccare per avere massima efficacia e sincronizzare attacchi.

Attribuzione

Per reagire in modo offensivo contro una intrusione occorre innanzitutto risolvere attribuzione cioè capire chi ha realizzato intrusione.

Questo spesso è possibile solo quando si sta monitorando l'infrastruttura d'attacco utilizzata.

D'altra parte se si rivelano le informazioni ottenute dal monitoraggio si ammette anche che si sta monitorando e quindi si informano attaccanti che possono essere monitorati...

Un altro modo di attribuire l'attacco è basato sul confront delle TTPs utilizzate e sulla similarità del codice degli exploit utilizzate (signature di codice)

Attribuzione di operazione softcell

Altre informazioni per attribuzione possono provenire dai metadata del malware utilizzato per intrusione.



Attribuzione determina chi paga

Assicurazioni non coprono act of war.

Se un attacco è sponsorizzato o eseguito da uno stato assicurazioni lo classificano come atto di guerra che non è coperto da assicurazioni per furti, distruzioni ecc.

Who Pays for an Act of Cyberwar?

Cyberinsurance doesn't cover acts of war. But does an cyberattack meet the definition of "warlike" actions remains blurry.



SICUREZZA DNS Domain Name System

Un database gerarchico distribuito, che associa nomi degli host a indirizzi IP.

Permette ad un utente di trovare un sistema senza conoscere il suo indirizzo IP: silicon.cs.umn.edu = 128.101.35.181

Organizza l'intera rete di domini.

I domini sono organizzati logicamente come un albero invertito. esempio, pisa.di.unipi.it specifica completa della macchina di nome pisa + dominio di + unipi, dominio dell'università di Pisa - it - sottodomino del dominio root.

Suddivisione in zone, responsabilità delegate.

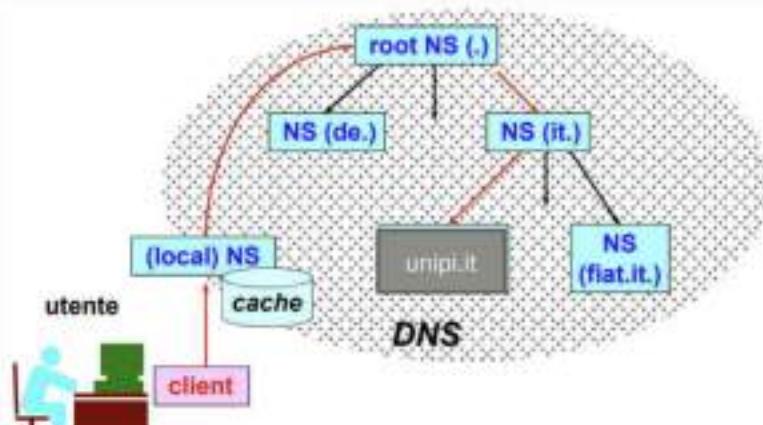
L'Internet Corporation for Assigned Names (ICANN) sovraintende agli assegnamenti dei nomi di dominio.

La risoluzione dei nomi è affidata ad un sistema distribuito.

I membri del sistema sono detti **Name Server** ed ogni applicazione deve contattare un name server per risolvere un nome.

Ogni autorità deve avere almeno un name server che gestisce la risoluzione dei nomi per quel dominio.

Un'autorità è totalmente responsabile del funzionamento dei suoi name server.



Name server e Risoluzione

Mantiene un database delle informazioni sugli host per la sua zona.

Contatta l'NS autoritativo della zona per prendere informazioni sull'host (come l'IP), Le informazioni devono essere aggiornate quando quelle dell'host cambiano nella zona e gli aggiornamenti dinamici cambiano i dati del DNS senza la necessità di ricostruire ogni altra parte dell'albero DNS.

La comunicazione con il name server è gestita dal resolver che fa parte dell'applicazione ed è invocato prima di contattare il livello di trasporto.

Se un utente su A su un dominio lancia il comando HTTP su b

- A richiede al name server su S l'indirizzo di B
- S risponde con l'indirizzo e A invia un messaggio HTTP di B

Il resolver permette all'utente di specificare nomi parziali e prova ad estenderli per ottenere nomi completamente qualificati.

RISOLUZIONE

Non esiste per gli host un metodo standard per individuare un name server sulla rete locale.

In generale il nome del name server è memorizzato in un file di configurazione oppure prelevato da un server DHCP.

Il name server utilizza porta 53 (sia UDP che TCP) e il name server può utilizzare due diverse modalità per rispondere ad una interrogazione.

- modalità ricorsiva: se non conosce la risposta, interroga altri name server e poi passa la risposta al client (usata dai server locali).
- Modalità iterativa: se non conosce la risposta passa al client l'indirizzo di un altro name server (usata dai name server di livello superiore).
- Il client sceglie la modalità di risposta settando un flag nel messaggio di richiesta.

Formato del messaggio di richiesta

Query Domain Name	
Query Type	Query Class

IDENTIFICATION: consente di far corrispondere le risposte alle richieste.

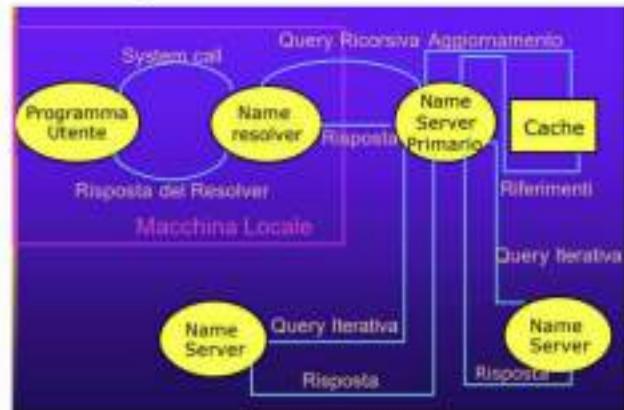
Parameter: contiene vari flag per specificare il tipo di operazione richiesta.

- Richiesta/risposta
- Corrispondenza indirizzo/nome o viceversa
- Messaggio troncato
- Risposta proveniente dalla autorità o indiretta
- Modalità ricorsiva disponibile
- Modalità ricorsiva richiesta
- Gli altri campi specificano quanti record sono contenuti nelle quattro sezioni del messaggio

Formato del messaggio, ogni messaggio può contenere più risposte

Identification	Parameter
Number of questions	Number of answers
Number of Authorities	Number of Additional
Question Sections	
Answer Sections	
Authority Sections	
Additional Information Sections	

Complessivamente:



Tipi

Il name server può essere usato per gestire oggetti di tipo diverso

A	Indirizzo di un host	Indirizzo IP
PTR	Puntatore	Nome di un dominio
CNAME	Nome canonico	Nome principale di un dominio
MX	Mail exchanger	Nome dell'host che agisce come mail exchanger per il dominio
HINFO	CPU e SO	Nome della CPU e del SO
MINFO	Mailbox info	Informazioni su una mailbox
NS	Name Server	Nome del name server che ha l'autorità sul dominio
SOA	Start of authority	Specifica la zona dell'autorità

Cache

Ogni name server mantiene in una cache tutte le corrispondenze tra nomi e indirizzi di cui viene a conoscenza.

Ad esempio risolvendo pisa.di.unipi.it name server locale mette in cache:

- name server di [it](#),
- name server di [unipi.it](#)
- name server di [di.unipi.it](#)

a questo punto una richiesta per www.di.unipi.it viene risolta direttamente dal name server locale.

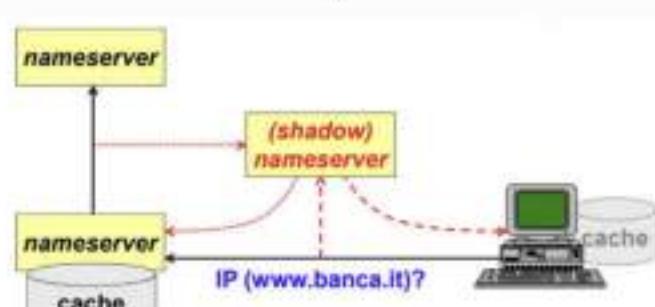
Ogni corrispondenza viene mantenuta nella cache per un periodo fissato, il tempo di validità è fissato dal name server che ha l'autorità su quel nome nel messaggio inviato agli altri name server. Per default il timeout è 2 giorni.

Attacchi al DNS

Il fatto che le comunicazioni tra nodi siano autenticate permette numerosi attacchi al DNS in modo da offrire false informazioni.

Meccanismo base è la ridirezione del traffico ad un nodo malizioso che poi può rubare informazioni contando sulla fiducia tra i due coinvolti.

Un possibile attacco è quello dello shadow server basato sullo sniffing delle richieste e sulla produzione di risposte false.



Un altro attacco è il cache poisoning che:

- cerca di attirare la vittima a fare una query sul proprio NS.
- fornisce risposta anche a query non effettuate per forzare/ sovrascrivere la cache della vittima.



Una seconda versione del cache poisoning prevede di

- fare una query
- fornisce subito una risposta falsa per sovrascrivere la cache



Autenticazione basata su nomi

Alcune applicazioni su internet, fanno uso di un meccanismo estremamente insicuro: Autenticazione basata sui nomi.

Ad esempio, "rlogin" di unix o "rsh" che usano il concetto di equivalenza remota per permettere l'accesso remoto ad un computer.

In questi reti, l'amministrazione di sistema o gli utenti, possono dichiarare (rendere nota) l'equivalenza remota di 2 account su 2 differenti macchine (ad esempio per mezzo dei file /etc/hosts.equiv oppure .rhosts).

Questa equivalenza associa due utenti di 2 differenti host semplicemente sulla base dei loro nomi e accesso al computer remoto è allora garantito se l'utente remot è dichiarato equivalente all'utente locale e se la richiesta dell'hostname coincide con quella contenuta nella definizione equivalente. Se viene usata l'autenticazione basata sul nome, è possibile accedere ad una macchina remota semplicemente truffando il nome di un host.

DNSSEC

Con l'RFC 2065 e dopo con la sua successiva 2535, l'estensione per la sicurezza del DNS fu ammessa e standardizzata in modo organizzato.

Il primo passo è provvedere all'autenticazione dei dati per gli RR scambiati in rete e con una autenticazione si ottenne anche integrità dei dati e l'autenticazione dei dati sorgenti.

L'autenticazione è ottenuta mediante firma digitale crittografica

delle informazioni e verifica delle informazioni è possibile verificando la firma.

DNSSEC specifica un nuovo RR chiamato KEY, la chiave pubblica di un sistema:

- Occorre avere una chiave pubblica autentica
- Il SIG RR è la firma digitale di una risposta/richiesta.
- Ogni insieme di RR mandato come risposta a una DNS query sarà accompagnato da una firma digitale generata mediante la chiave privata del mittente.
- Il ricevente può verificare l'autenticità e l'integrità dei messaggi verificando la firma.
- Il ricevente può verificare l'autenticità e l'integrità dei messaggi verificando la firma.

ESTENSIONI

- SIG - memorizza le firme digitali (asymmetric keys)
- KEY - memorizza le chiavi pubbliche
- NXT - autentica la non-esistenza di nomi o tipi di RR in un dominio.
- DNSSEC agisce su insiemi di RR e non su singoli RR
- DNSSEC intende provvedere a:
 - Autenticare l'origine dei dati e all'integrità di essi.
 - Fornire la distribuzione delle chiavi
 - Autenticare le transizioni e le richeiste
- KEY RR specificano:
 - il tipo di chiave (zona, host, utente)
 - Il protocollo (DNSSEC, IPSEC, TLS, ecc.)
 - L'algoritmo (RSA/MD5, DSA, ecc.)
- SIG RR specificano:
 - Il tipo degli RR (SOA,A,NS,MX,ecc.)
 - l'algoritmo(RSA/MD5,DSA,ecc.)
 - i tempi di inizio e scadenza (della validità)
 - la firma
- NXT RR specificano:
 - Il prossimo nome nella zona
 - Tutti i tipi di RR coperti dal nome corrente
 - La chiave privata non è visibile in linea ed è usata per firmare gli insiemi di RR del file di zona.
 - La chiave pubblica è pubblicata nel KEY RR
 - La chiave pubblica della zona è firmata usando la chiave privata del padre di zona.
 - La firma del padre di zona sulla chiave pubblica della zona è aggiunta al fine di zona.

Notare che in tutto **DNSSEC crittografia** è usata sempre e solo per integrità per confidenzialità delle informazioni.

File di zona

Problemi DNSSEC

nessuna firma delle query DNS.

Nessuna sicurezza nel diagolo tra DNS client e DNS (local) sever.

- Usare IPsec.

Crittografia sui server DNS:

- Sovraccarico computazionale
 - Sovraccarico gestionale (on-line secure crypto host)
 - Maggior dimensione dei record
 - Scarsa sperimentazione con prestazioni non note.

MOVING TARGET DEFENSE

Integrazione di più tecniche per impedire all'avversario di raccogliere informazioni sul sistema.

Basato sulla diversità di sistema, modifica dinamicalmente la configurazione del sw o del sistema per aggiungere incertezza, imprevedibilità e diversità.

Cambia continuamente la superficie di attacco del, Aumenta il costo per gli attaccanti.

Di conseguenza, il sistema è imprevedibile per gli aggressori, difficile da sfruttare ed è più resistente agli attacchi.

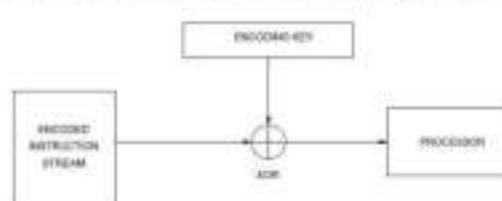
Address space payout randomization = cambia indirizzo base
dove viene caricato un programma in memoria

Name	Description	Company Name	Version	Size	Image Base	ASLR
ADVAPI32.dll	Advanced Windows 32 Base API	Microsoft Corporation	5.00.6000.16388	0x7620000	0x7620000	ASLR
CLSCatQ.DLL	COM+ Configuration Catalog	Microsoft Corporation	2001.12.6030.16	0x7700000	0x7700000	ASLR
COMCTL32.dll	User Experience Controls Library	Microsoft Corporation	6.10.6000.16388	0x7420000	0x7420000	ASLR
COMMCTRL.dll	Common Dialog DLL	Microsoft Corporation	6.00.6000.16388	0x7612000	0x7612000	ASLR
GDI32.dll	GDI Client DLL	Microsoft Corporation	6.00.6000.16388	0x778F000	0x778F000	ASLR
IMM32.DLL	IME-User Windows IMM32 API.DLL	Microsoft Corporation	6.00.6000.16388	0x7690000	0x7690000	ASLR
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	6.00.6000.16388	0x7740000	0x7740000	ASLR
locale.nls				0x20000		n/a
locale.nls				0x70000		n/a
LPK.dll	Language Pack	Microsoft Corporation	6.00.6000.16388	0x7720000	0x7720000	ASLR
MSCFT.dll	MSCFT Server DLL	Microsoft Corporation	6.00.6000.16388	0x7670000	0x7670000	ASLR
ncvapi.dll	Windows NT CRT DLL	Microsoft Corporation	7.00.6000.16388	0x7611000	0x7611000	ASLR
oleaut32.dll	OLE API	Microsoft Corporation	6.00.6000.16388	0x6400000	0x6400000	ASLR
ole32.dll	NT Layer DLL	Microsoft Corporation	6.00.6000.16388	0x7770000	0x7770000	ASLR
oleui32.dll	Microsoft OLE UI - Windows	Microsoft Corporation	6.00.6000.16388	0x7580000	0x7580000	ASLR

Instruction set randomization = encryption del codice in memoria
in modo da impedire injection attack.

Istruzioni vengono decodificate solo quando vengono eseguite o meglio caricate nel cache e variando la chiave di cifratura cambia in modo imprevedibile.

Compiler-based Randomization: Il compilatore genera varianti dello stesso codice per impedire di conoscere a priori quale versione viene eseguita.



Diversificazione del sw al momento dell'installazione.
Il sw installato tramite un installatore specializzato viene taggato ed associato ad una chiave random e al momento dell'esecuzione si verifica se la chiave associata è quella specificata dall'installer. Se è diversa l'esecuzione è vietata.

Altre soluzioni modificano i comandi mediante stringhe casuali e poi vengono eseguiti solo se matchano le stringhe casuali e non i comandi veri.

Nel seguito vediamo alcune tecniche di moving target defense utilizzate al Argonne National Lab, centro di ricerca USA su armi atomiche ecc.

Multple OS Rotational Environment

Come funziona la rotazione:

1. Il demone stabilisce una connessione SSH con la macchina Live.
2. Il computer Live viene spostato sull'IP di riserva per il rilevamento delle intrusioni.
3. Il demone stabilisce una connessione SSH con un computer selezionato nell'elenco dei server disponibili (selezionato in coda o in modo casuale).

(selezionato come coda o in modo casuale)

1. La macchina selezionata viene spostata sul server attivo
2. Il rilevamento delle intrusioni viene eseguito sulla macchina che è stata tolta dall'IP live.
3. Se il server non è stato compromesso, viene aggiunto all'elenco dei server disponibili.
4. Se il server è stato compromesso, viene aggiunto all'elenco dei server non disponibili e non verrà inserito nella rotazione.

nella rotazione

Dynamic Application Rotation Environment

- Una macchina virtuale (VM) viene selezionata in un determinato momento per gestire tutto il traffico di rete. traffico di rete ed è nota come VM attiva.

- A un intervallo di tempo predefinito (da 15 a 30 secondi), la macchina virtuale attiva viene commutazione della macchina virtuale attiva.

- Quando una macchina virtuale diventa inattiva, viene verificata l'integrità del file system e rimossa dal sistema. di attacchi e viene rimossa dalla rotazione se viene rilevata una compromissione dell'integrità. compromissione dell'integrità.

- La procedura rispecchia la MTD MORE, che ha impostato la finestra di rotazione inferiore a 60 secondi. inferiore a 60 secondi. L'idea di entrambe le strategie è quella di ruotare a un intervallo

intervallo adeguato al fine di

1. Prevenire un'impronta digitale accurata e l'identificazione dei punti di ingresso.

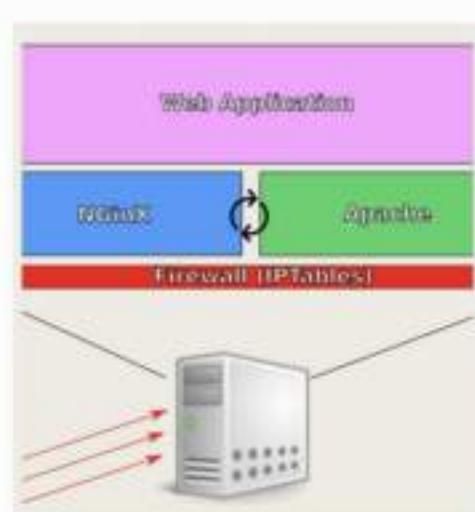
punti di ingresso.

1. contrastare gli attacchi persistenti riducendo l'esposizione del software vulnerabile.

software vulnerabile.

1. Ridurre la fattibilità di qualsiasi guadagno per l'attaccante consumando

tempo e risorse supplementari.

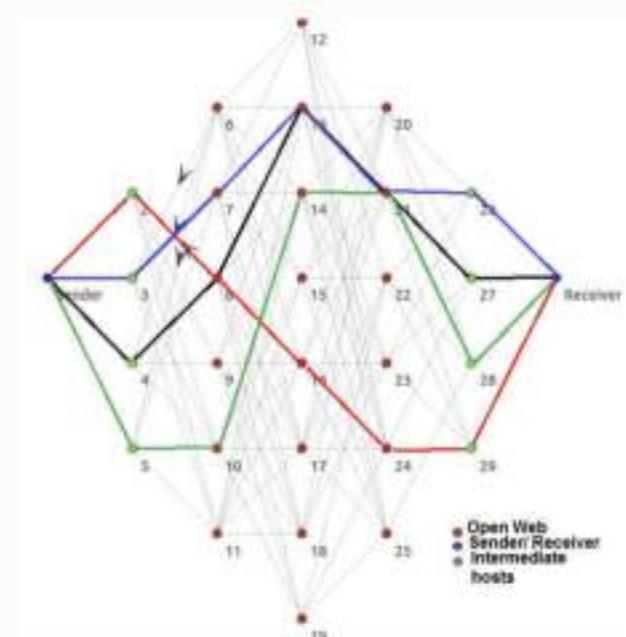


Stream Splitting

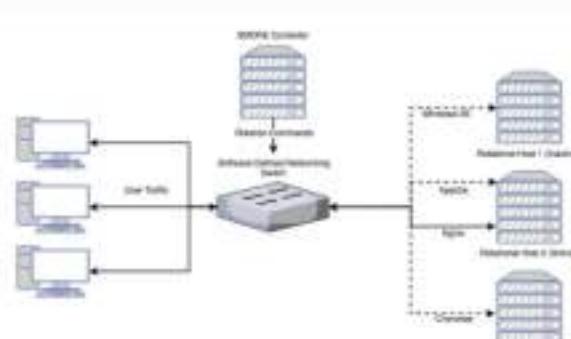
Il TCP stream splitting (SS) divide un flusso di rete in più flussi.

di rete in più flussi, rendendo difficile per un attaccante attaccare il sistema sistema, eliminando il vantaggio di configurazioni fisse del sistema e dell'architettura di rete.

architettura di rete.



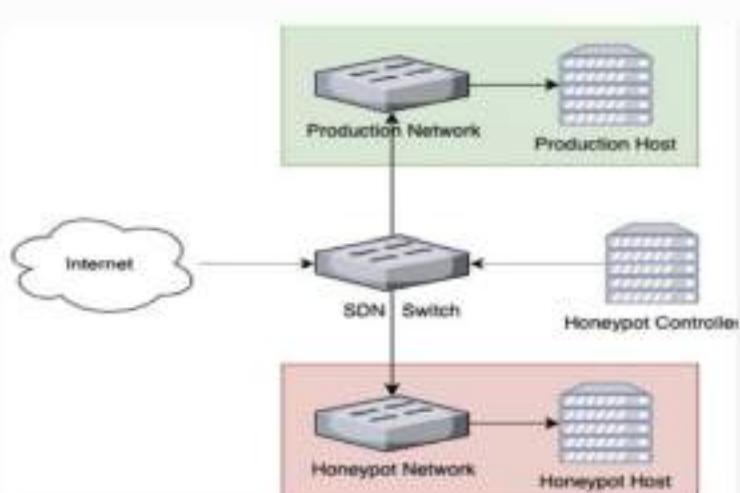
SW defined networking Multiple Operating System Rotational Environment



SMORE-MTD difende dagli attacchi zero-day utilizzando il software-defined networking per manipolare i percorsi di rete che servono le

richieste degli utenti.
per manipolare i percorsi di rete che servono le richieste degli utenti.

- Selezionando in modo casuale il server e il servizio che risponderà alla richiesta di un determinato utente. richiesta dell'utente, SMORE-MTD rende più difficile per un aggressore eseguire una ricognizione e identificare i servizi da attaccare.
- SMORE-MTD aumenta anche la resilienza alle vulnerabilità non garantendo che un un exploit dell'attaccante venga indirizzato al software vulnerabile.
- Se l'exploit viene indirizzato in un punto diverso da quello previsto dall'attaccante, quest'ultimo dovrà ripetere l'operazione. Se l'exploit viene indirizzato in un luogo diverso da quello previsto dall'aggressore, quest'ultimo dovrà ripetere l'attacco nella speranza di raggiungere il software vulnerabile; Questi attacchi ripetuti hanno maggiori probabilità di essere notati.
- SMORE-MTD semplifica l'amministrazione di un MTD ruotando i server e i servizi a livello di rete. a rotazione dei server e dei servizi a livello di rete, eliminando la necessità di installare e mantenere il software di configurazione su ogni host a rotazione. di configurazione su ogni host a rotazione. Questo riduce la complessità e aumenta la quantità di software disponibile per la rotazione.



esempio attribuzioneFile

VERIFICA DEL PROGETTO DEL SISTEMA

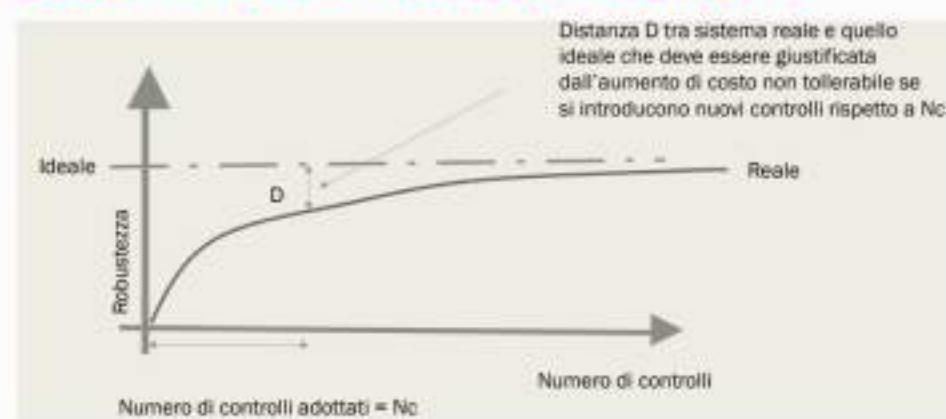
Ogni progetto deve stabilire un compromesso tra la fedeltà a questi principi e le prestazioni.

Possiamo pensare ad un sistema ideale che soddisfa completamente a questi principi, ogni sistema reale ha una certa distanza dal sistema ideale = controlli che mancano o che sono implementati solo parzialmente.

Regola fondamentale: la distanza si giustifica solo con le prestazioni ovvero.

possono mancare solo controlli che peggiorano delle prestazioni di interesse, un controllo che non peggiora le prestazioni deve essere presente

Sistema reale vs Sistema ideale



Saltzer & Schroeder

1. Economy Of Mechanism
2. Fail-safe Defaults = Default Denies
3. Complete Mediation
4. Least Privilege
5. Least Common Mechanism
6. Separation of Privilege
7. Open Design
8. Psychological Acceptability
9. Work factor
10. Compromise recording

Stabiliscono la configurazione ed il funzionamento ideale dei meccanismi di sicurezza molto difficili da soddisfare in toto per la riduzione delle prestazioni ed il costo ma ogni violazione è una potenziale vulnerabilità.

- Principle of Economy Of Mechanism
Il meccanismo di protezione deve avere un design semplice e ridotto=regola KISS
- Principle of Fail-safe Defaults
Il meccanismo di protezione dovrebbe negare l'accesso per impostazione predefinita e concedere l'accesso solo quando esiste un'autorizzazione esplicita (l'impostazione predefinita nega l'accesso)
- Principle of Complete Mediation
Il meccanismo di protezione deve controllare ogni accesso a ogni oggetto. Molto costoso e proprio per questo è uno dei più violati.
- Principle of Open Design
Il meccanismo di protezione non deve dipendere dal fatto che gli aggressori ignorino il suo progetto per avere successo. Può basarsi sull'ignoranza di informazioni specifiche come password o chiavi di cifratura.
- Principle of Separation of Privilege
Il meccanismo di protezione dovrebbe consentire l'accesso in base a più di un'informazione. = due chiavi per una cassaforte

- Principle of Least Privilege
Il meccanismo di protezione deve costringere ogni processo a operare con i privilegi minimi necessari per svolgere il proprio compito..
La distruzione di root e di tutti i modelli basati su ring.
- Principle of Least Common Mechanism
Il meccanismo di protezione deve essere condiviso il meno possibile tra gli utenti.
- Principio of Psychological Acceptability
Il meccanismo di protezione dovrebbe essere facile da usare (almeno quanto non usarlo).

Fail safe default

Un progetto conservativo deve basarsi su argomentazioni che spieghino perché gli oggetti debbano essere accessibili, piuttosto che sui motivi per cui non dovrebbero. In un sistema di grandi dimensioni alcuni oggetti saranno considerati in modo inadeguato, quindi una di mancanza di permessi è più sicuro, Un errore di progettazione o di implementazione in un meccanismo che fornisce un'autorizzazione tende a fallire rifiutando l'autorizzazione, una situazione sicura, in quanto viene sarà rapidamente individuato. D'altra parte, un errore di progettazione o di implementazione in un meccanismo che esclude esplicitamente l'accesso che esclude esplicitamente l'accesso tende a fallire permettendo l'accesso, un fallimento che può passare inosservato nell'uso normale. Questo principio si applica sia all'aspetto esteriore del meccanismo di protezione sia all'aspetto esteriore che all'implementazione sottostante. implementazione

Mediazione completa

Il controllo per ogni accesso è molto costoso e può essere ridotto solo con un adeguato supporto hw.
Recentemente sono state proposte delle evoluzioni della architettura a capability (puntatori protetto), le tagged architetture es: processori ARM.

La memoria virtuale viene partizionata in zone ed ogni zona viene dedicata ad un certo tipo di dato, nessuna zona di memoria può essere utilizzata da tipi di dato diversi (riuso ma all'interno di un certo tipo).

Alcuni bit di ogni puntatore di memoria (tag) sono dedicati ad informazioni sul tipo di dato puntato, ad ogni accesso si controlla che il tipo del puntatore sia coerente con quello dell'area di memoria puntata.

Open design

- Kerchoff: "An attacker who learns the key learns nothing that helps them break any message encrypted with a different key. That's the essence of Kerkoff's principle: that systems should be designed that way."
- Shannon: One ought design under the assumption that the enemy will immediately gain full familiarity with them.

Questo non significa che dobbiamo pubblicizzare informazioni sul sistema perché comunque attaccante deve lavorare per raccogliere.

Per valutare importanza di segretezza dobbiamo valutare importanza di azioni di raccolta informazioni durante una intrusione. all'aumentare di questo tempo aumenta l'importanza di non rilevare informazioni.

E' utile rivelare informazioni sul sistema solo quando porta ad una peer review i cui risultati vengono condivisi, altrimenti può essere dannosa.

Peer review = chi analizza il sistema deve avere le competenze necessarie per analisi.

Non legato ad open source.

Least privilege in una rete

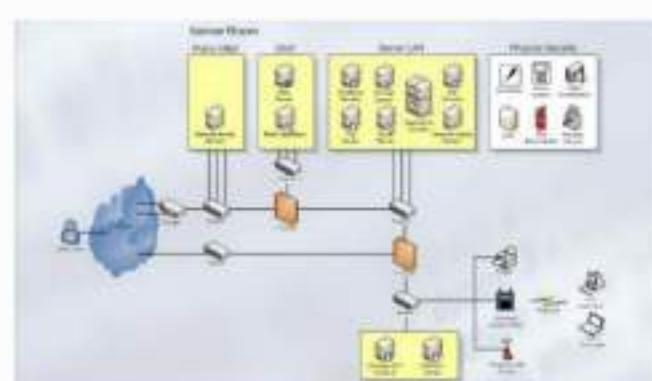
Defence - in - depth : segmentare la rete in sottoreti ognuna protetta da un firewall in modo da impedire attacchi.

In una rete piatta:

- Ogni nodo può interagire con un qualsiasi altro nodo della rete
- Dopo essere entrato nella rete attaccante può attaccare un qualsiasi altro nodo.

In una rete segmentata attaccante si muove più lentamente.

Pivoting = attaccare un nodo per raggiungere un altro nodo.



Zero trust network

Considerazione iniziale: non ha senso vpn se client è stato attaccato con successo.

Esito della richiesta di un dispositivo di accesso ad un servizio della rete dipende da informazioni disponibili sul dispositivo e da utente che lo usa e non dalla posizione fisica del dispositivo.

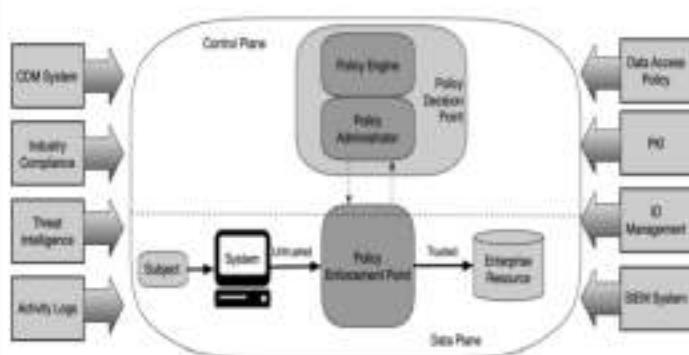
Possibile negare accesso ad un dispositivo che non è stato aggiornato e quindi con molte vulnerabilità anche se è fisicamente nella sede principale e vicino al server.

Richieste vengono elaborate da un **trust o policy engine** che riceve informazioni su utente/dispositivo/servizio, stato della rete e decide se accettare/rifiutare in base al rischio.

Inventario e EDR sono possibili sorgenti di informazioni come

anche risultati di adversary emulation e se engine accetta la richeista chiede al server di creare un canale criptato con richiedente.

La distruzione del canale criptato da parte del server impedisce ulteriori richeiste.



Zero Trust NIST

La fiducia zero presuppone che non venga concessa alcuna fiducia implicita agli asset o agli account utente basati esclusivamente su

- la loro ubicazione fisica o di rete (ad esempio, reti locali o Internet)
- proprietà degli asset (aziendali o personali=BYOD).

L'autenticazione e l'autorizzazione (sia per l'oggetto che per il dispositivo) sono funzioni discrete funzioni discrete eseguite prima di stabilire una sessione di accesso alle risorse aziendali.

Zero Trust

- è una risposta alle tendenze delle reti aziendali che includono gli utenti remoti, dispositivi personali e risorse basate su cloud che non si trovano all'interno di una rete aziendale. di rete di proprietà dell'azienda.
- si concentra sulla protezione delle risorse (risorse, servizi, flussi di lavoro, account di rete, ecc. di rete, ecc.), non i segmenti di rete = l'ubicazione della rete non è più la componente principale della sicurezza. la posizione della rete non è più il componente principale della sicurezza della risorsa.

Zero Trust: I fondamenti

Tutte le fonti di dati e i servizi informatici sono considerati risorse. Tutte le comunicazioni sono protette, indipendentemente dalla posizione della rete, che da sola non implica fiducia. L'accesso alle singole risorse aziendali è concesso su base individuale. La fiducia nel richiedente viene valutata prima di concedere l'accesso con i privilegi minimi per completare il compito. L'accesso alle risorse è determinato da criteri dinamici (= identità del cliente osservabile, applicazione/servizio e risorsa richiedente). servizio e la risorsa richiedente) e può includere attributi comportamentali e ambientali. L'azienda monitora e misura l'integrità e la sicurezza di tutti gli asset posseduti e associati.

associati. Nessun asset è intrinsecamente affidabile.

Tutte le autenticazioni e le autorizzazioni delle risorse sono

dinamiche e rigorosamente applicate prima di consentire l'accesso. accesso. Si tratta di un ciclo costante di ottenimento dell'accesso, scansione e valutazione delle minacce, adattamento, e di rivalutare continuamente la fiducia in una comunicazione continua. Un'azienda che implementa una ZTA dovrebbe disporre di Identity, Credential e Access Management (ICAM) e di asset management. (ICAM) e di sistemi di gestione degli asset. Ciò include l'uso di L'azienda deve disporre di sistemi di gestione delle identità, delle credenziali e degli accessi (ICAM) e di gestione delle risorse. L'azienda raccoglie il maggior numero possibile di informazioni sullo stato attuale degli asset, infrastruttura di rete e delle comunicazioni e le utilizza per migliorare la propria posizione di sicurezza.

Zero trust: Non esiste il perimetro

La rete aziendale non è una zona di fiducia implicita. Le risorse devono sempre comportarsi come se un aggressore fosse presente sulla rete aziendale. un aggressore sulla rete aziendale e le comunicazioni devono essere effettuate nel modo più sicuro possibile. più sicuro disponibile. Ciò implica l'autenticazione di tutte le connessioni e la crittografia di tutto il traffico. I dispositivi in rete non possono essere di proprietà dell'azienda o configurabili. I visitatori e/o i servizi I visitatori e/o i servizi a contratto possono includere risorse non di proprietà dell'azienda che necessitano di accesso alla rete. Nessuna risorsa è intrinsecamente affidabile. Ogni risorsa deve essere sottoposta a una valutazione della sua posizione di sicurezza tramite una PEP prima che venga concessa una richiesta di accesso a una risorsa di proprietà aziendale. per tutta la durata della sessione. I dispositivi di proprietà dell'azienda possono essere dotati di artefatti che consentono l'autenticazione e fornire un livello di fiducia più elevato rispetto ai dispositivi non di proprietà dell'azienda. Il soggetto sono insufficienti per l'autenticazione di un dispositivo a una risorsa aziendale. Non tutte le risorse aziendali si trovano su un'infrastruttura di proprietà dell'azienda. Le risorse includono soggetti aziendali remoti soggetti aziendali remoti e servizi cloud. I soggetti e le risorse aziendali remote non possono fidarsi completamente della loro connessione alla rete locale. I soggetti remoti soggetti remoti devono presumere che la loro rete sia ostile. Le risorse devono presumere che tutto il traffico sia monitorato e potenzialmente modificato. Tutte le richieste di connessione devono essere autenticate e autorizzate autorizzate e tutte le comunicazioni devono essere effettuate nel modo più sicuro possibile (Le risorse e i flussi di lavoro che si

muovono tra l'infrastruttura aziendale e quella non aziendale devono avere una politica e una postura di sicurezza coerente. una politica e una postura di sicurezza coerente. Le risorse e i carichi di lavoro devono mantenere la loro postura di sicurezza quando si spostano verso o da un'infrastruttura di proprietà dell'azienda. Ciò include i dispositivi degli utenti remoti e i carichi di lavoro utenti remoti e carichi di lavoro che migrano da data center on-premises a cloud non aziendali.

Least common mechanism

Afferma che i meccanismi utilizzati per accedere alle risorse non devono essere condivisi.

La condivisione delle risorse fornisce un canale lungo il quale è possibile trasmettere informazioni, pertanto tale condivisione deve essere ridotta al minimo.

L'uso di macchine virtuali imporrà automaticamente questo privilegio. Altrimenti, il sistema operativo fornirà un certo supporto (spazio di memoria virtuale) ma non un supporto completo (perché il file system apparirà come condiviso tra diversi processi).

Esempio 1

Un sito Web fornisce servizi di commercio elettronico a un'azienda. Per privare l'azienda delle entrate derivanti da quel sito Web, gli aggressori inondano il sito di messaggi e bloccano i servizi di commercio elettronico. I clienti legittimi non riescono ad accedere al sito web e, di conseguenza, si rivolgono altrove.

La condivisione di Internet con i siti degli aggressori ha causato il successo dell'attacco.

La contromisura appropriata limita l'accesso dell'attaccante al segmento di Internet collegato al sito web. Internet collegato al sito web. Le tecniche per farlo includono firewall proxy per indirizzare le connessioni sospette

Saltzer & Schroeder

Dopo aver introdotto i primi 8 S&S dicono

Analysis of traditional physical security systems have suggested two further design principles which, unfortunately, apply only imperfectly to computer systems

- **Principio del fattore di lavoro:** Confrontare il costo dell'aggiramento del meccanismo con le risorse di un potenziale attaccante. Possiamo confrontare la **robustezza** di due sistemi misurando la **quantità di lavoro** necessaria all'attaccante per una intrusione. Per **misurare** la quantità di lavoro dobbiamo poter **emulare** l'attaccante = threat intelligence + TTPs
- **Principio della registrazione di compromesso Principio della registrazione di compromesso:** Meccanismi che registrano in modo affidabile una compromissione delle informazioni

possono sostituire quelli più elaborati che impediscono completamente la perdita

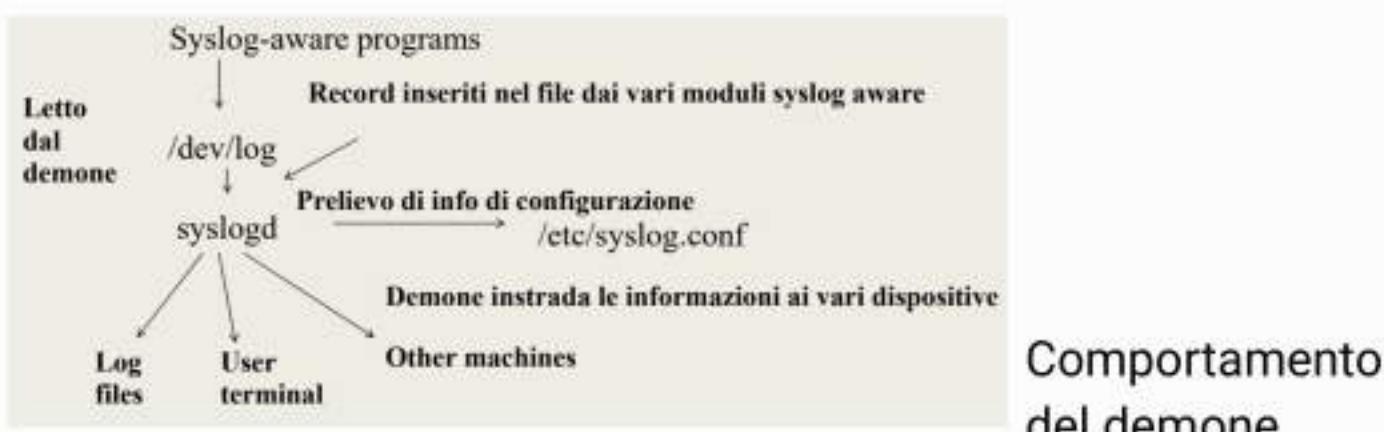
Compromise recording

Uso di log di sistema per ricordare azioni eseguite dai vari soggetti, il lag permette di controllare abuso di privilegi che però il soggetto deve possedere:

- Shutdown di impianti di distribuzione (luce, acqua e gas)
- Cancellare o accedere ad informazioni (need to know)

Tutti i sistemi operativi offrono meccanismi per logi, linux/unix offrono syslog per creare e gestire file di log e ogni informazione da loggare è caratterizzata da una severità ed è fondamentale che il file di log NON sia memorizzato sullo stesso nodo di rete di cui stiamo ricordando gli eventi.

Il miglior device per un file di log è una blockchain.



dipende dalla facility che ha prodotto (inf??) e dal livello di severità del vento.

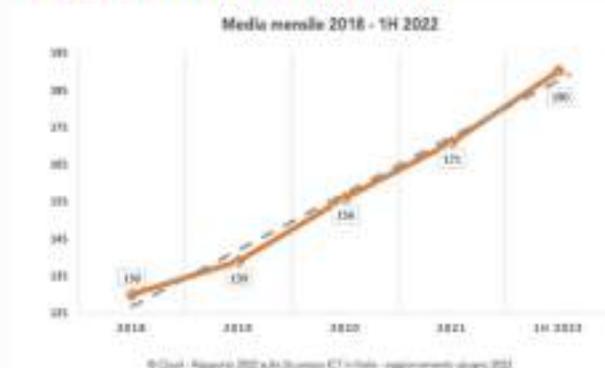
Facility Number	Facility Description	Facility Number	Facility Description
0	kernel messages	12	NTP subsystem
1	user-level messages	13	log audit
2	mail system	14	log alert
3	system daemons	15	clock daemon
4	**security/authorization messages	16	local use 0 (local0)
5	messages generated internally by Syslog	17	local use 1 (local1)
6	line printer subsystem	18	local use 2 (local2)
7	network news subsystem	19	local use 3 (local3)
8	UUCP subsystem	20	local use 4 (local4)
9	clock daemon	21	local use 5 (local5)
10	security/authorization messages	22	local use 6 (local6)
11	FTP daemon	23	local use 7 (local7)

Sorgenti di informazioni da inserire nei log.

SEVERITY LEVEL	EXPLANATION
0 EMERGENCY	A "panic" condition - notify all tech staff on call? (Earthquake? Tornado?) - affects multiple apps/servers/sites.
1 ALERT	Should be corrected immediately - notify staff who can fix the problem - an example is loss of backup ISP connection.
2 CRITICAL	Should be corrected immediately, but indicates failure in a primary system - fix CRITICAL problems before ALERT - an example is loss of primary ISP connection.
3 ERROR	Non-urgent failures - these should be relayed to developers or admins; each item must be resolved within a given time.
4 WARNING	Warning messages - not an error, but indicated that an error will occur if action is not taken, e.g. file system 85% full - each item must be resolved within a given time.
5 NOTICE	Events that are unusual but not error conditions - might be summarized in an email to developers or admins to spot potential problems - no immediate action required.
6 INFORMATIONAL	Normal operational messages - may be harvested for reporting, measuring throughput, etc. - no action required.
7 DEBUG	Info is useful to developers for debugging the app; not useful during operations.

compito a casa - challengeFile

Rapporto CLUSIT



Distribuzione degli attaccanti per tipologia (2018 – 1H 2022)

ATTACCONTI PER TIPOLOGIA	2018	2019	2020	2021	1H 21	1H 22	TH 2021 vs 1H 2022	Trend 2022
Cybercrime	1.229	1.381	1.518	1.763	925	876	-3.4%	⬇️
Espionage-Sabotage	203	303	284	217	95	154	+62.1%	⬆️
Information Warfare	38	25	44	49	25	57	+118.2%	⬆️
Hacktivism	64	48	46	20	7	36	+414.3%	⬆️
Espionage-Sabotage + Inf. Warfare	361	238	328	286	121	211	+74.0%	⬆️
Totali	1.554	1.667	1.874	2.049	1.053	1.143	+8.4%	⬇️

Work factor

Emulando un attacco possiamo scoprire:

- Il lavoro che deve fare per una intrusione.
 - Tempo necessario per intrusione.
 - Probabilità di successo.

Principale problema è legato a prove ripetute. Ci sono eventi stocastici in una intrusione il cui effetto può essere valutato solo con prove ripetute. E' molto difficile ripetere un penetration test perché disturba il sistema reale e richiede tempo ed investimento non banale. La disponibilità di una piattaforma di attacco automatico permette di ridurre il costo dei test ripetuti ma non il rumore. Necessità di utilizzare una versione "digitale" del sistema per poter lavorare su questa versione in modo automatico senza dirbare il sistema. La disponibilità di una versione digitale consente ragionamenti di tipo what-if e di security design costruendo la versione digitale prima di quella reale.

Compromise recording - ransomware

Uso backup contro rasomware.

Prevedo ma non prevengo attacco, non resisto e uso back per ripristinare nel sistema le informazioni criptate.

Nessuna robustezza ma buona resilienza il tempo perché il sistema ritorni a prestazioni normali è quelli per accesso a backup e ripristino.

Robustezza serve comunque

1. Potrei essere attaccato di nuovo se non scopro ed elimino le vulnerabilità che hanno permesso di attaccarmi.
 2. Potrebbe essere stato attaccato il backup.



Per risolvere il problema di attacco a backup **non conviene** usare disco protetto mediante SO (perché??) ed abbiamo due alternative:

- Soluzione 1 (ENISA)

Apply the 3-2-1 rule of backup. For all data:

- 3 copies,
- 2 different storage media,
- 1 copy offsite.

Mantenere i dati personali criptati secondo le disposizioni del GDPR e utilizzando controlli adeguati basati sul rischio.

- Soluzione 2 (Cloud provider)

Usare memorie immutabili o WORM = write once read many.

Una memoria worm è un disco ottico o normale configurato mediante chiavi fisiche in modo che ogni blocco fisico possa essere scritto solo una volta.

S&S e approccio sistematico

In generale una violazione dei principi indica una weakness e non usa vulnerabilità, la differenza tra weakness e vulnerabilità è la mancanza del vettore di attacco.

Le debolezze e le vulnerabilità sono entrambi stati che indicano rischi per la sicurezza. rischi per la sicurezza. Mentre la debolezza si riferisce a un errore o a un bug dell'applicazione, essa può diventare una vulnerabilità nei casi in cui può essere sfruttata per eseguire un'azione dannosa.

La differenza tra una debolezza e una vulnerabilità è la disponibilità di un payload specifico che permette di un payload specifico che ne consenta lo sfruttamento.

Una volta che l'exploit è disponibile, la debolezza è considerata una vulnerabilità e come tale comporta un rischio maggiore per la sicurezza dell'applicazione

Possiamo applicare i principi di S&S ad un **singolo modulo**, **sottosistema** per valutare la robustezza del componente analizzato.

Quando li applichiamo ad un **intero sistema** dobbiamo controlli compensativi/delegati = controlli eseguiti in un componente per rimedare alla mancanza di controlli in altri componenti.

Implementare un controllo in un modulo può permettere di migliorare le prestazioni perché è più semplice avere un supporto hw/firmware. Un firewall può limitare accesso ad alcuni servizi ed a questo punto i servizi possono utilizzare controlli "più semplici" per il filtraggio delle richieste ricevute. Il filtraggio nel firewall può usare acceleratori hw basati su schede di rete ottimizzate con PLA. Un altro esempio sono architetture di firewall che integrano router e nodi e ovviamente nel caso di controlli compensativi/delegati il blocco di un componente può portare a quello di quelli che hanno delegato i controlli.

Integrazione di strumenti per ransomware

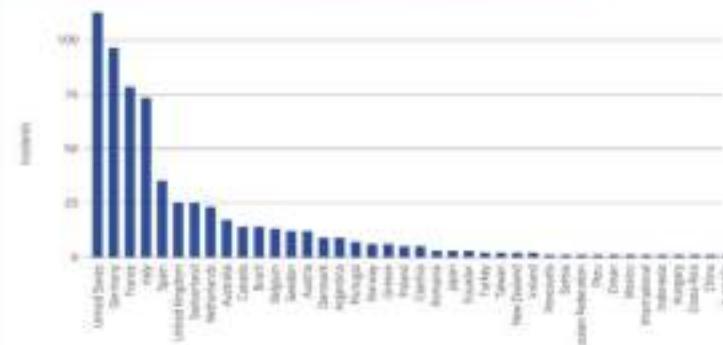
La soluzione di minacce complesse e potenti come ransomware richiede più contromisure.

Strategia complessiva ENISA contro ransomware.

- Have a good, verified and updated backup of all your business-critical files and personal data
 - Apply the 3-2-1 rule of backup.
 - Encrypt personal data according to DPR and use appropriate risk-based controls
 - Run EDR in your endpoint devices that can detect most ransomware.
 - Maintain your security awareness, security policy, and privacy protection policy up to date and working on your systems and assets to accomplish desired hygiene by network segmentation, up to date patches, backups, and appropriate identity, credential, and access management (ICAM) preferably with MFA.
 - Conduct regular risk assessment and consider taking out insurance based on its results.
 - Restrict administrative privileges: use caution when handing out administrative privileges. Always employ Principle of Least Privilege when granting any access.

Ma pochi seguono i buoni consigli

Figure 2. Number of consumer incidents in each country based on ECIS accident analysis



Tecniche per accesso iniziale

MITRE ATTACK techniques for initial access

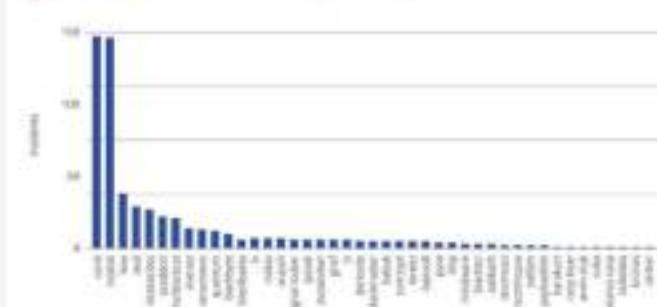
- | |
|--|
| T1133 External Remote Services: Exploited remote desktop |
| T1133 External Remote Services: RDP Brute Force ¹⁰ |
| T1133 External Remote Services: Exploited terminal services |
| T1189 Drive-by Compromise |
| T1189 Drive-by Compromise: Exploit kits |
| T1203 Exploitation for Client Execution: Exploiting Software Vulnerability |
| T1988 Exploitation for Privilege Escalation ¹¹ (not shown in chart) |
| T1988 Valid Accounts |
| T1566.001 Phishing: Spear phishing Attachment |
| T1566.002 Phishing: Spear phishing Link |

Out of the studied 823 incidents, it was not reported how the threat actors get initial access. In 984 of them, which is an overwhelming 95.3%, it is understandable that targets don't want to share how they were (or will) be vulnerable for security reasons but at the same time the lack of information does not help victims to realize what they can do to prevent such attacks.

From the rest of the known 29 incidents for which initial access was leaked, there is small amount of data upon which to draw conclusions; the distribution is as follows:

Initial Access according to MITRE	No. of incidents
T1033 External Remote Services	12
T1566 Phishing	8
T1995 Supply Chain Compromise	4
T1876 Valid Accounts	6
T1666 Exploitation for Privilege Escalation	1

Attacchi per singolo gruppo



NIST Cybersecurity Framework

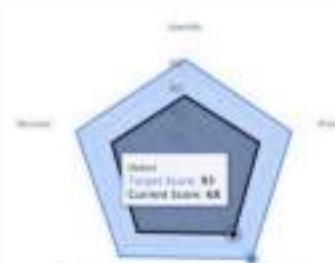
Un framework che sta diventando uno standard de facto da più
60% delle aziende USA.

Un approccio molto più focalizzato sul business che sulla tecnologia è quello stato attualmente adottato da molti altri paesi.

tecnologia per valutare stato attuale e miglioramenti richiesti. Può essere adottato da qualunque azienda indipendentemente dalle sue dimensioni per migliorare il suo livello di sicurezza, si parla di passare da un profilo ad un altro profilo.

Basato su 5 funzioni.

- Identify
 - Protect
 - Detect



- Respond
- Recovery

Funzioni comprendono categorie, sottocategorie e riferimenti, Le categorie sono fondamentalmente risultati di azioni da eseguire es. dati protetti.

Le sottocategorie specificano dal punto di vista tecnologico le azioni eseguite es. dati protetti mediante encryption.

Riferimenti informativi sono

puntatori a standard, best practices ecc. che specificano un metodo per ottenere certi risultati.



- Identify

L'identificazione si concentra sull'azienda e sul suo rapporto con il rischio di cybersecurity, tenendo conto delle risorse a disposizione.

Azioni da compiere:

- Definire tutti gli asset e gli ambienti (inventario)
- Definire lo stato attuale e quello obiettivo dei controlli (inventario dei controlli).
- Elaborare un piano per rimediare a queste lacune.
- Definire le priorità di approccio alla mitigazione in un contesto aziendale.
- Dare priorità alle esigenze di tutti gli stakeholder e dei leader aziendali coinvolti.
- Definire le modalità di comunicare sui temi della cybersecurity con tutti gli stakeholder coinvolti stakeholder.

Prevede parte su come parlare di cybersecurity ai manager ed gli azionisti.

- Protect

L'obiettivo della protezione è quello di sviluppare e implementare adeguate misure di salvaguardia per assicurare la erogazione di servizi infrastrutturali critici. La funzione Protect supporta la capacità di limitare o contenere l'impatto di un potenziale evento di cybersecurity..

Azioni da compiere:

- Controllo degli accessi: convalida delle identità e degli accessi ai diversi sistemi, strutture ecc.
- Sensibilizzazione e formazione: dare ai dipendenti e ad altri persone la possibilità di far parte del vostro piano di cybersecurity con istruzione e formazione.
- Sicurezza dei dati: gestire i dati secondo gli standard aziendali per ridurre i rischi di cybersecurity e proteggere in modo proattivo la disponibilità, l'integrità e la riservatezza.
- Processi e procedure di protezione delle informazioni: Mettere in atto le politiche, i processi e le procedure

necessarie per gestire la protezione e le procedure.

- Manutenzione: Riparare continuamente i componenti del sistema informativo e mirigarli.
- Tecnologia di protezione: Implementare le soluzioni di sicurezza necessarie per la protezione in linea con le politiche aziendali (controlli mancanti).

- Detect

Questa funzione consente di scoprire tempestivamente gli eventi di cybersecurity. Esempi di risultati le categorie di risultati di questa funzione includono: Anomalie ed eventi; Monitoraggio continuo della sicurezza; processi di rilevamento.

Monitoraggio continuo e processi di rilevamento.

Azioni da fare:

- Anomalie ed eventi: il programma rileva le attività insolite il prima possibile. L'impatto degli eventi è compreso da tutti i membri del team e non solo.
- Sicurezza e monitoraggio continuo: Monitoraggio del sistema informativo e degli ambienti e gli ambienti a intervalli specifici per identificare gli eventi di cybersecurity nell'organizzazione.
- Processi di rilevamento: le procedure e i processi di rilevamento vengono messi in atto e testati per garantire una consapevolezza tempestiva e ampia degli eventi di cybersecurity.

- Respond

Questa funzione sviluppa e implementa attività appropriate per intraprendere azioni relative a un incidente di cybersecurity rivelato. Supporta la capacità di contenere l'impatto di un potenziale incidente di cybersecurity.

Azioni da intraprendere:

- Pianificare della risposta: I processi e le procedure di risposta vengono eseguiti e mantenuti, per garantire una risposta tempestiva agli eventi di cybersecurity rilevati.
- Analisi: l'analisi viene condotta per garantire una risposta adeguata e supportare le attività di ripristino.
- Mitigazione: vengono eseguite attività per prevenire l'espansione di un evento, mitigare gli effetti e sradicare l'incidente.
- Comunicazioni: Le attività di risposta sono coordinate con le parti interessate interne ed esterne, come appropriato, per includere il supporto esterno delle forze dell'ordine.
- Miglioramenti: Le attività di risposta dell'organizzazione vengono migliorate incorporando le lezioni apprese dalle attività di rilevamento/risposta attuali e precedenti.

- Recover

Questa funzione è definita come la necessità di "sviluppare e implementare le attività appropriate per mantenere i piani di resilienza e ripristinare qualsiasi capacità o servizio".

Attività appropriate per mantenere i piani di resilienza e per ripristinare qualsiasi capacità o servizio che sono stati compromessi a causa di un evento di sicurezza della cybersecurity. Supporta il ripristino tempestivo delle normali operazioni per ridurre l'impatto di un evento di cybersecurity.

- Pianificazione del ripristino: Le procedure di ripristino vengono testate, eseguite e mantenute in modo che il programma possa mitigare gli effetti di un evento il prima possibile.
- Miglioramento: La pianificazione e i processi di ripristino vengono migliorati quando si verificano gli eventi e vengono identificate le aree di miglioramento e messe a punto le soluzioni.
- Comunicazione: Coordinarsi internamente ed esternamente per una maggiore organizzazione, pianificazione ed esecuzione e approfondite.

CYBER RISK COME VALUTARLO

Impatti di una intrusione di A = Imp_A

Probabilità di successo dell'intrusione di A = A tenta intrusione ed ha successo = $P(\text{Succ}_A | \text{Int}_A)$

- Questa probabilità dipende dalla robustezza di infrastruttura.
- Fattori puramente tecnologici e del sistema complessivo

Probabilità dell'intrusione = $P(\text{Int}_A)$ probabilità che A tenti intrusione.

Probabilità che A provochi impatto di A = $P(\text{Imp}_A) = P(\text{Succ}_A | \text{Int}_A) \times P(\text{Int}_A)$

Rischio dovuto ad A = $F(\text{Imp}_A, P(\text{Imp}_A))$

$P(\text{Int}_A)$ e $P(\text{Succ}_A | \text{Int}_A)$ non sono indipendenti perché la probabilità che A tenti intrusione dipende anche dalla probabilità di successo dell'intrusione stessa.

Se per A alcuni sistemi sono equivalenti = gli permettono di raggiungere lo stesso obiettivo allora attaccherà quello che massimizza $P(\text{Succ}_A | \text{Int}_A)$

- Terrorista
- Ransomware
- ecc.

Occorre considerare i vari attaccanti ed il rischio complessivo = somma dei rischi assumendo che $P(\text{Int}_{Aj})$ sia indipendente da $P(\text{Int}_{Ak})$

Valutazione dei parametri

$P(\text{Succ}_A | \text{Int}_A)$ può essere stimata mediante adversary emulation di A rispetto all'infrastruttura considerata, possono essere necessarie adversary emulation multiple che possono essere automatizzate usando una versione digitale dell'infrastruttura. Adversary emulation di A ci permette di tener conto di strategia di A e delle varie TTPs che utilizza nelle sue intrusioni.

$P(\text{Int}_A)$ è molto più difficile da valutare quindi non si da un valore puntuale ma un intervallo, abbiamo due framework che si possono usare:

- DREAD risk assessment model.
- OWASP likelihood assessment method

Entrambi considerano attributi i cui valori vengono poi trasformati in numeri e sommati, il valore complessivo ci dà una stima della probabilità che A tenti una intrusione. La stima è ottenuta lavorando su intervalli.

Ad esempio 5 intervalli ognuno con valore 0-10

- 40-50 Critico
- 30-40 Alto
- 20-30 Medio
- 0-20 Basso

- DREAD

Sviluppato da microsoft per valutare il rischio complessivo non probabilità di essere attaccati.

Considera 5 elementi per ordinare le varie intrusioni possibili:

- Damage - how bad would an attack be?
- Reproducibility - how easy is it to reproduce the attack?
- Exploitability - how much work is it to launch the attack?
- Affected users - how many people will be impacted?
- Discoverability - how easy is it to discover the threat?

Ed è evidente che

Damage ed effected users stimano l'impatto di una intrusion.

Reproducibility, Exploitability and Discoverability stimano quanto è probabile che intrusione avvenga.

Nel modello originale ogni elemento veniva valutato con un valore 0-10 e la somma era una stima del rischio complessivo.

- OWASP

Modello più sofisticato del precedente che valuta minaccia e sistema in modo separato.

Fattori dell'agente di minaccia:

- **Skill Level** - Quanto è tecnicamente preparato questo gruppo di agenti di minaccia?
- **Motive** - Quanto è motivato questo gruppo di agenti delle minacce a trovare e sfruttare questa vulnerabilità?
- **Opportunity** - Quali sono le risorse e le opportunità necessarie a questo gruppo di agenti di minaccia per trovare e sfruttare questa vulnerabilità?
- **Size** - Quanto è grande questo gruppo di agenti di minaccia?

Fattori legati al Sistema

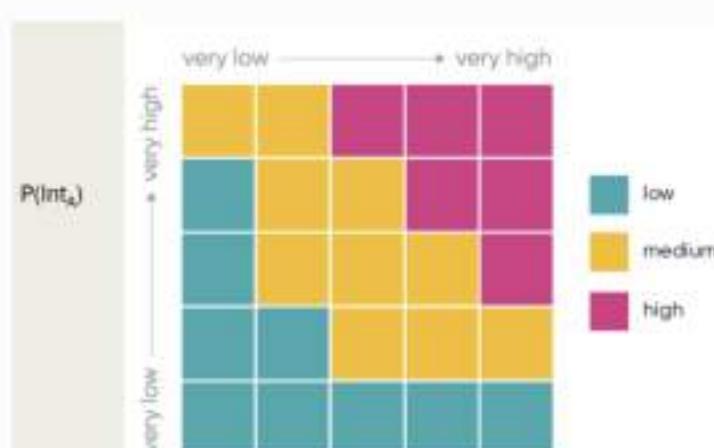
- **Ease of discovery** - Quanto è facile per questo gruppo di agenti di minaccia scoprire questa vulnerabilità?
- **Ease of exploit** - Quanto è facile per questo gruppi di agenti di minaccia sfruttare effettivamente questa vulnerabilità?
- **Awareness** - Quanto è nota questa vulnerabilità a questo gruppo di agenti di minaccia?
- **Intrusion Detection** - Quanto è probabile che un exploit venga rivelato?

Applicare al nostro problema se consideriamo tutto il sistema e non la singola, ovviamente dobbiamo adottare il punto di vista dell'attaccante ovvero come attaccante vede il nostro sistema.

Calcolo rischio

Possiamo trasformare anche $P(\text{Succ}_A \mid \text{Int}_A)$ in uno di alcuni intervalli e quindi arrivare ad una valutazione semiquantitativa della probabilità di una intrusione con successo mediante una matrice come la seguente.

Ripetendo con impatto possiamo arrivare ad una valutazione semiquantitativa del rischio.



SYSTEM DESIGN STRATEGIES

"come progettare"

Libro consigliato: Buildinf Secure & Reliable System (animale: Iuecertola).

Progettare con il privilegio minimo

Il principio del minimo privilegio afferma che gli utenti devono avere la quantità minima di accesso per svolgere un compito, indipendentemente dal fatto che l'accesso provenga da persone o da sistemi, queste limitazioni sono più efficaci quando vengono aggiunte all'inizio del ciclo di vita dello sviluppo, durante la fase di progettazione di nuove funzionalità = è quella che viene comunemente chiamata security by design, richiesta ad esempio da GDPR.

I privilegi non necessari portano a una superficie crescente per possibili errori, bug o compromessi e creano rischi per la sicurezza e l'affidabilità che sono costosi da contenere o minimizzare in un sistema funzionante.

La considerazione precedente si riferisce alla superficie d'attacco: in breve è meglio ridurre la superficie d'attacco durante il progetto perché le modifiche al sistema ed al modo di usarlo in un tempo successivo hanno un costo troppo elevato = costo di meccanismi di sicurezza dipende da quando vengono adottati.

In un mondo reale, gli utenti e gli amministratori hanno buone intenzioni ed eseguono i loro compiti in modo perfetto, senza errori. Purtroppo, nella realtà le persone commettono errori, i loro account possono essere compromessi o possono essere malintenzionati. Per questi motivi è importante progettare i sistemi con il *minimo privilegio*: i sistemi devono limitare l'accesso degli utenti ai dati e ai servizi necessari per svolgere il compito in questione. Per ridurre il rischio introdotto dagli attori umani sono necessari controlli aggiuntivi o lavori di ingegneria che aumentano i costi a causa dei tempi di progettazione, delle modifiche ai processi, del lavoro operativo o del costo opportunità.

Bisogna:

- Limitare questi costi dando priorità a ciò che si desidera proteggere. Non tutti i dati o le azioni sono uguali e la composizione dell'accesso può variare notevolmente a seconda della natura del sistema.
- Non proteggono tutti gli accessi allo stesso livello.
- Applicare i controlli più importanti ed evitare la mentalità del "tutto o niente".

E' necessario:

- Per classificare gli accessi in base all'impatto, al rischio per la sicurezza e/o alla criticità.
- Gestire in modo diverso l'accesso a diversi tipi di dati

(pubblici, aziendali, utenti, crittografici).

- Per trattare le API amministrative che possono cancellare i dati in modo diverso dalle API di lettura specifiche del servizio.

Esempio di priorità ed impatti diversi

Consideriamo un ospedale il cui sistema informativo deve gestire quattro classi di dati:

1. Informazioni sulle medicine disponibili
2. Informazioni sulle spese per ogni paziente
3. Informazioni sulle medicine prescritte ad un paziente
4. Cartella clinica con esami di un paziente

Come possiamo procedere in questo caso?

Quali sono i problemi posti da ogni classe?

Caso generale

	Description	Read access	Write access	Infrastructure access ^a
Public	Open to anyone in the company	Low risk	Low risk	High risk
Sensitive	Limited to groups with business purpose	Medium/high risk	Medium risk	High risk
Highly sensitive	No permanent access	High risk	High risk	High risk

^a Administrative ability to bypass normal access controls. For example, the ability to reduce logging levels, change encryption requirements, gain direct SSH access to a machine, restart and reconfigure service options, or otherwise affect the availability of the service(s).

Come sempre, gli accessi amministrativi sono quelli più pericolosi. Oltre ad evitare la figura di amministratore onnipotente può essere utile applicare il decimo principio di S&S (compromise recording).

API

L'API di un sistema distribuito include tutti i modi in cui qualcuno può interrogare o modificare il suo stato interno.

L'API amministrativa è più importante di quella rivolta all'utente per l'affidabilità e la sicurezza del sistema.

Gli errori di digitazione e di utilizzo delle API amministrative possono causare interruzioni catastrofiche o esporre enormi quantità di dati. Di conseguenza, le **API amministrative sono anche alcune delle superfici di attacco più interessanti per gli agenti delle minacce** e sono accessibili solo agli utenti interni e gli strumenti, quindi possono essere più veloci e facili da modificare rispetto alle API rivolte agli utenti. Tuttavia, la loro modifica comporta comunque un costo, quindi bisogna valutare attentamente la loro progettazione. Le API amministrative includono:

- API di configurazione/teardown, come quelle utilizzate per costruire, installare e aggiornare il sw o fare il provisioning dei container o della MV in cui vengono eseguiti.
- API di manutenzione e di emergenza, come l'accesso amministrativo per cancellare i dati o lo stato dell'utente danneggiato o per riavviare processi che si comportano male.

API Posix

POSIX (Portable Operating System Interface) è un insieme di interfacce standard per SO basato sul sistema Unix, le specifiche POSIX più recenti - IEEE Std 1003.1-2017 definiscono un'interfaccia e un ambiente standard che possono essere utilizzati da un OS per fornire accesso alle applicazioni conformi a POSIX. Lo standard definisce anche un interprete di comandi (shell) e programmi di utilità comuni. POSIX supporta la portabilità delle applicazioni a livello di codice sorgente, in modo che le applicazioni possano essere eseguite su qualsiasi SO conforme a POSIX.

Top Unix commands: 50 must-know commands with examples

Una linea di Unix comando to manage and monitor files, manage directories, monitor your network and users, gather data, and automate system tasks and processes.

View more

Download PDF

Vai alla lista di contenuti di questo documento con il link dell'elenco. Tuttavia, non ti trovi qui perché questo documento può essere già stato trovato o può essere stato scaricato.

Questa API di grandi dimensioni è popolare perché è flessibile e familiare a molte persone. Come API per la gestione delle macchine di produzione, viene utilizzata soprattutto per attività ben definite, come l'installazione di un pacchetto sw, la modifica di un file di configurazione o il riavvio di un demone.

Gli utenti spesso eseguono la configurazione e la manutenzione tradizionale dell'host tramite una sessione interattiva di OpenSSH o con strumenti di scripting contro l'API POSIX, entrambi gli approcci espongono l'intera API POSIX al chiamante e può essere difficile limitare e controllare le azioni di un utente durante la sessione interattiva, ciò è particolarmente vero se l'utente tenta di eludere i controlli in modo malizioso o se la stazione di lavoro collegata è compromessa.

Esistono vari meccanismi per limitare i permessi concessi all'utente tramite l'API POSIX, ma questa necessità è **un difetto fondamentale dell'esposizione di un'API molto grande**. E' meglio invece ridurre e scomporre questa grande API amministrativa in pezzi più piccoli.

Si può quindi seguire il principio del minimo privilegio per conciare l'autorizzazione solo alle azioni specifiche richieste da un particolare chiamante.

Il problema della macchina compromessa si può risolvere mediante Zero Trust inoltre si può comunicare al server il pezzo (sottoinsieme) dell'interfaccia che un utente può utilizzare.

Breakglass & Auditing

Una API minima può provocare il problema che un certo operatore non riesce a risolvere uno specifico problema, dovuto ad una circostanza eccezionale o ad un errore di progetto si può introdurre un breakglass mechanism (ispirato a scure antiincendio protetta da un vetro), questi meccanismi permettono, in generale ad amministratori, di bypassare completamente il sistema di

controllo degli accessi e di eseguire alcune operazioni (uccidere o di rimediare ad un errore (si è negato accesso ad un utente che me aveva diritto).

Il breaking glass mechanism permette di risolvere problemi ma per evitare abusi il suo utilizzo deve essere:

- Regolamentato
- Verificato (audit)

Una ulteriore ragione a favore di un audit delle varie funzioni per amministrazione.

Breakglass guidelines

La possibilità di utilizzare un meccanismo di breakglass dovrebbe essere altamente limitata. In generale, dovrebbe essere disponibile solo al team responsabile dello SLA operativo del sistema.

Il meccanismo di breakglass per la rete zero trust dovrebbe essere disponibile solo da luoghi specifici. Questi luoghi sono le panic room, luoghi specifici con controlli di accesso fisici aggiuntivi per giustificare la maggiore fiducia risposta nella loro connettività.

Il meccanismo di ripiego per una rete a fiducia zero si traduce in una strategia che non si fida della posizione della rete ma si fida di alcune posizioni con controlli di accesso fisici aggiuntivi. Tutti gli indirizzi di un meccanismo di rottura devono essere attentamente monitorati e riscontrati (audit), il meccanismo del breakgalss deve essere testato regolarmente dal team responsabile dei servizi di produzione, per assicurarsi che funzioni quando serve, e quando viene utilizzato con successo e gli utenti hanno riacquistato l'accesso, i vostri SRE e il team delle politiche di sicurezza possono diagnosticare e risolvere ulteriormente il problema di fondo.

Testing

Un sistema che usa least privilege rispetto ad uno che invece concede tutti i diritti a tutti, questo non è sorprendente poiché nel secondo caso il vero problema è quello di scoprire se **sono stati concessi troppi diritti** ma questo in generale viene scoperto dopo una intrusione che ha avuto successo. Invece quando si applica least privilege i problemi sorgono prima, in generale quando si nega a qualcuno un accesso a cui ha diritto. Questo si affronta mediante testing e debugging, ovviamente diversi da quelli per le funzionalità.

Si distingue 2 problemi:

- Testing of least privilege, per garantire che l'accesso sia concesso solo alle risorse appropriate.
- testing with least privilege, Per garantire che l'infrastruttura per i test abbia solo l'accesso necessario.

Test di = essere in grado di verificare che profili utente ben definiti (es: analista di dati, assistenza clienti, SRE) abbiano privilegi sufficienti per svolgere il loro ruolo, ma non di più.

Il secondo punto sottolinea che occorre utilizzare un ambiente dedicato per il testing (qualcosa di simile era già stato evidenziato per patches) sconsigliato usare account ad hoc.

Autenticazione ed Autorizzazione

Supponendo di aver definito alcune API ognuna delle quali abilità l'esecuzione di alcune operazioni, ogni richiesta di usare una API deve essere autenticata ed autorizzata.

- Autenticata = controllo sull'identità del richiedente abbiamo già visto meccanismi diversi di autenticazione (password, certificati ecc.)
- Autorizzata = verifica sul processo da parte del soggetto autenticato per utilizzare API.

Oltre ad identità del soggetto authorizer può considerare (come ci insegna ZT):

- Operazione
- Dispositivi Utilizzato
- Parametri della richiesta
- Metadati quali località della richiesta, stato del dispositivo...

l'interfaccia per trasmettere informazioni per A&A deve essere la più omogenea possibile nel sistema in mood da semplificare accesso alle API.

Possibile uso di MultiPartyAuthorization = coinvolgimento di più persone per conceder un diritto.

MPA and 3FA

In questi casi richiesta concessa da una piattaforma (nodo) deve essere confermata da un'altra.

In generale abbiamo un sottoinsieme di nodi della rete configurati in modo da essere più robusti degli altri, questi nodi hanno il ruolo di confermare le decisioni prese dai nodi meno robusti, caso interessante è quello in cui i nodi più robusti sono gli smartphone degli utenti, una richiesta di accesso ad un servizio viene trasmessa allo smartphone dell'utente per confermarla.

Questo è diverso dall'uso di smartphone per 2FA, sostanzialmente si chiede all'utente di confermare la richiesta di accesso e quindi siamo nel campo di autorizzazione arrivando a 3FA dove la terza A sta per autorizzazione.

Utile nel caso di malware su workstation che tenta accesso ma non contro insider attackers.

Accesso temporaneo

vi sono casi in cui non si può applicare privilegio minimo perché non si può avere un controllo fine sulle operazioni = concedere il diritto di invocare poche operazioni.

In questo caso si può ricorrere a MPA o 3FA per controllare richiesta ed evitare accessi di malware eseguito senza che utente ne sia cosciente, un ulteriore modo di compensare è quello di ridurre il tempo per cui si concede accesso ad una certa API, semplice da implementare nel caso accesso avvenga mediante un server oppure si possa interporre un proxy da richiedente e server che fornisce il servizio, interrompendo il servizio dopo un certo intervallo e forzando il richiedente e ripetere la richiesta si riduscono gli impatti di un uso malevolo di API, come già abbiamo visto parlando di firewall un proxy o un reverse proxy può anche implementare ulteriori controlli per scoprire accessi illegali.

Cosa serve

Una conoscenza completa del vostro sistema per classificare ogni parte in base al suo rischio, sulla base di questa classificazione, è necessario suddividere il sistema e l'accesso ai dati al livello più fine possibile. Le API funzionali di piccole dimensioni sono una necessità per ottenere il minimo privilegio.

Un sistema di autenticazione per convalidare le credenziali quando gli utenti tentano di accedere al sistema.

Un sistema di autorizzazione che applica una politica di sicurezza ben definita che può essere facilmente collegata ai vostri sistemi finemente partizionati.

Una serie di controlli avanzati per l'autorizzazione che possono fornire approvazioni temporanee, MF e MP.

Come minimo, il sistema deve possedere le seguenti capacità:

- Per verificare tutti gli accessi e generare segnali per identificare le minacce ed eseguire analisi forensi storiche.
- Per progettare, definire, testare ed eseguire il debug della vostra politica di sicurezza e per fornire il supporto agli utenti finali per tale politica.
- Per fornire un meccanismo di rottura quando il sistema non si comporta come ci si aspettava

Design for robustness

Diminuisce la probabilità di vulnerabilità della sicurezza o di fallimenti della resilienza. Qualsiasi modifica al sistema potrebbe introdurre accidentalmente una nuova vulnerabilità o compromettere la resilienza. Quanto meno comprensibile è il sistema, tanto più probabile è che ciò avvenga.

Facilita una risposta efficace agli incidenti, Durante un incidente, è fondamentale che i soccorritori possano valutare con rapidità e precisione i danni, contenere l'incidente e identificare e risolvere le

cause principali. Un sistema complesso e difficile comprensione ostacola notevolmente questo processo.

Aumenta la fiducia nelle asserzioni sulla sicurezza del sistema, le asserzioni sulla sicurezza sono espresse in termini di *invarianti*: proprietà che devono valere per tutti i possibili comportamenti del sistema. Ad esempio, quando riceve input malformati o dannosi, il suo comportamento non deve violare alcuna proprietà di sicurezza. In un sistema di difficile compressione, questo può essere impossibile da verificare perché solo un ragionamento astratto può stabilire gli invarianti e non i test.

Invariante

Una proprietà che è sempre vera, indipendentemente dal comportamento dell'ambiente del sistema ed è *responsabile* di garantire che una proprietà desiderata sia effettivamente un invariante, anche se l'ambiente del sistema si comporta in modo arbitrariamente imprevisto o dannoso.

L'ambiente comprende tutto ciò su cui il sistema non ha un controllo diretto, dagli utenti malintenzionati che colpiscono il frontend del servizio con richieste maligne ai guasti hw che causano crash casuali (ingegneria del caos).

Uno degli obiettivi principali dell'analisi di un sistema è quello di determinare se determinate le proprietà desiderate sono effettivamente invarianti, se alcuni comportamenti violano una proprietà di sicurezza, cioè se la proprietà dichiarata non è in realtà un invariante, allora il sistema ha una debolezza o vulnerabilità di sicurezza (dipende dal vettore di attacco).

Ad esempio, se un input dannoso provoca un comportamento inatteso perché un gestore di richieste manca di controllo di accesso o perché tali controlli sono stati implementati in modo errati, il sistema presenta una vulnerabilità di sicurezza che potrebbe consentire a un aggressore di accedere a dati privati.

Esempio:

Solo gli utenti autenticati e debitamente autorizzati possono accedere all'archivio dati persistente, tutte le operazioni sui dati sensibili nell'archivio dati persistente vengono registrate in un registro di audit in conformità con la politica di auditing, tutti i valori ricevuti dall'esterno del perimetro di fiducia sono adeguatamente convalidati o codificati prima di essere passati alle API che sono soggette a vulnerabilità di iniezione (es: API per query SQL o API per la costruzione di markup HTML).

Il numero di interrogazioni ricevute dal backend è proporzionale al numero di interrogazioni ricevute dal frontend, se il backend non risponde a una query dopo un periodo di tempo predeterminato, il frontend si **degrada con grazia**, rispondendo con una risposta approssimativa.

Quando il carico di un componente è superiore a quello che può gestire, per ridurre il rischio di guasti a cascata, il componente in questione presenterà [errori di sovraccarico](#) invece di bloccarsi. Il sistema può ricevere RPC solo da un insieme di sistemi designati e più inviare RPC solo a un insieme di sistemi designati.

Analisi degli invarianti

Uno spettro definisce il compromesso tra l'impatto derivante dalle violazioni di un invariante e lo sforzo per assicurare che esso sia effettivamente valido.

Da un lato, eseguiamo alcuni test e leggiamo parti del codice per cercare bug che potrebbero violare l'invariante. Questo non porta a un'elevata confidenza, lo scarso impegno spiega perché classi di vulnerabilità comuni e ben comprese, come SQL injection e buffer overflow, occupano le prime posizioni nella classifica delle "top vulnerability".

All'altro capo dello spettro ci sono le analisi basate su ragionamenti formali e provabili: il sistema e le sue proprietà sono modellati in una logica formale per costruire una prova (con l'aiuto di un assistente di prova automatico) che la proprietà è valida. Questo è difficile e comporta molto lavoro, esempio, uno dei più grandi progetti di verifica del sw ha costruito una prova di correttezza e sicurezza completa dell'implementazione di un microkernel a livello di codice macchina, con un impegno di circa 20 anni.

Sevvene la verifica formale stia diventando praticamente applicabile in alcuni casi, come i microkernel o le librerie di crittografia, in genere non è fattibile per i grandi progetti sw.

Invarianti e comprensibilità

Il metodo proposto usa la comprensibilità del sistema come metodo per ridurre la complessità delle analisi per stabilire la validità degli invarianti.

Si propone l'uso di [modelli mentali](#) che si focalizzano (astraggono) su [alcuni aspetti del sistema](#) e ne trascurano (volontariamente) altri metodi non pertinenti o non influenti, il problema posto dall'uso di modelli mentali è che essi vengono di solito definiti a partire dalle situazioni più frequenti di funzionamento e di interazione tra i moduli, invece la sicurezza è legata ad eventi rari (attacchi) e che quindi non possono basarsi su esperienze passate, il problema si può gestire costruendo [il sistema in mood che non violi il modello mentale](#) cioè in modo che scopra situazioni anomale e reagisca segnalandole.

Negli esempi di invarianti un modulo scopriva di essere sovraccarico ed informava gli altri. Questo riduce i comportamenti anomali che il sovraccarico può provocare.

Quindi comprensibilità = capacità di del modello umano prevedere il comportamento anche in situazioni anomale.

Comprensibilità e complessità

La comprensibilità aumenta se la complessità viene gestita decomponendo un sistema in modo orizzontale e verticale (moduli e gerarchia di MV), questo semplifica molti ma non tutti i problemi (un errore in un livello di gerarchia può permettere attacchi nei livelli superiori).

Problemi similari possono sorgere nella decomposizione orizzontale in moduli. In questo caso si può ridurre complessità centralizzando la soluzione di alcuni problemi ad esempio introducendo un solo server di autenticazione oppure decentralizzando ma mantenendo una forte sincronizzazione tra i moduli (come molti di un DB).

Un ulteriore aiuto può venire all'autonomia del singolo modulo che interagisce e coopera ma è in grado di scoprire anomalie nei moduli con cui interagisce e di reagire (uso di asincronia e di timeout).

Autonomia = funzionamento corretto senza fare assunzioni sul comportamento degli altri moduli. Minori sono le assunzioni che un modulo fa sul comportamento degli altri è maggiore è la sua autonomia e più facile è prevedere il suo comportamento.

Ulteriori elementi per ridurre complessità e quindi favorire comprensibilità e modelli umani:

- Preferire interfacce <> e che riducano i grandi di libertà dell'interpretazione.
 - Una interfaccia che accetti http offre molti grandi di libertà (ed aumenta dipendenza da altri moduli riducendo autonomia).
 - Una interfaccia che permetta la definizione di tipi di dato semplifica i controlli.
- Un interfaccia che permetta di gestire risorse di tipo diverso si semplifica se si può definire un modello dei dati comuni che permetta la semplificazione del modello umano definendo:
 - Invarianti per ogni tipo
 - Soluzioni standard per esaminare, annotare, classificare i vari dati
 - Operazioni consistenti per i vari tipi a partire dai tipi base
 - definire tipi composti in modo coerente a partire dai tipi base.
- Riconoscere operazioni idempotenti che possono essere ripetute senza errori, l'idempotenza è una proprietà importante per comunicazioni inaffidabili.

Comprensibilità ed identità

Identità ben riconoscibili per autenticazione e controllo degli accessi, sono importanti non solo per gli utenti ma anche per i vari servizi (logging, audit).

La robustezza del sistema di controllo degli accessi dipende da quella degli identificatori per identità, indirizzi IP non sono identificatori affidabili (dinamici, riusati) possono essere spoofati facilmente messaggi. Le cose non cambiano molto se aggiungo porte, Proprietà di identificatori importanti sia per sicurezza e safety (affidabilità).

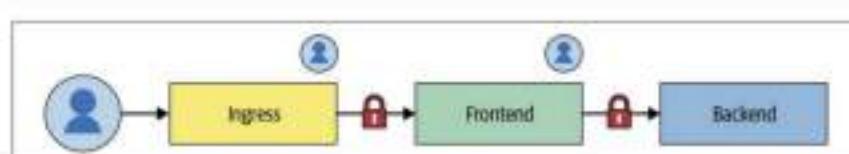
- Comprensibili ad un essere umane.
- Robusti contro spoofing (senza controlli vs password vs certificati).
- Non riutilizzabili: se un dipendente se ne va e riusciamo il suo identificatore possono nascere problemi di attribuzione.

Identità in Google

Utenti, Amministratori, macchine (identificatore è il nome DNS, identità indica il ruolo che ha quella macchina (testing, carichi utente ecc..) e workload (richiesto da un utente, assegnato ad una macchina in un certo insieme e gestito da un sistema di orchestrazione simile a Kubernetes)).

Controlli degli accessi risultate

Framework coordina i vari workload a partire da una richiesta utente, i dati vengono passati tra workload e il framework decide se i controlli degli accessi sono risolti in base ad identità dell'utente richiedente o del workload.



Trusted Computing Base

L'insieme di componenti (hw, sw, umani ecc.) il cui corretto funzionamento è sufficiente a garantire l'applicazione della politica di sicurezza o, più concretamente, il cui fallimento potrebbe causare una violazione della politica di sicurezza.

In quanto tale, il TCB deve rispettare la politica di sicurezza anche se qualsiasi entità esterna al TCB si comporta in modo arbitrario e possibilmente dannoso.

Naturalmente, l'area esterna al TCB comprende l'ambiente esterno del sistema (ad esempio, attori malintenzionati da qualche parte su internet), ma anche parti del proprio sistema che non fanno parte del TCB.

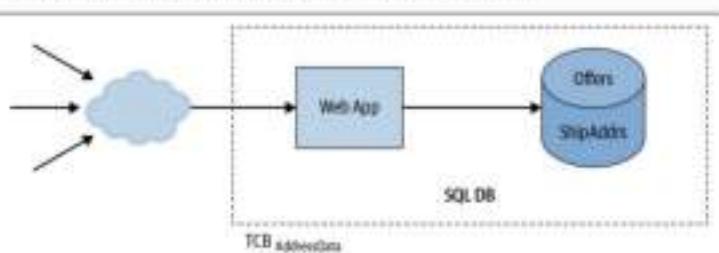
L'interfaccia tra il TCB e "tutto il resto" viene definita confine di sicurezza.

"tutto il resto - altre parti del sistema, l'ambiente esterno, i client del sistema che interagiscono con esso attraverso una rete e così via - interagisce con il TCB comunicando attraverso questo confine. Il TCB deve trattare con sospetto tutto ciò che attraversa il confine della sicurezza, sia i dati stessi che altri aspetti, come l'ordinamento dei messaggi.

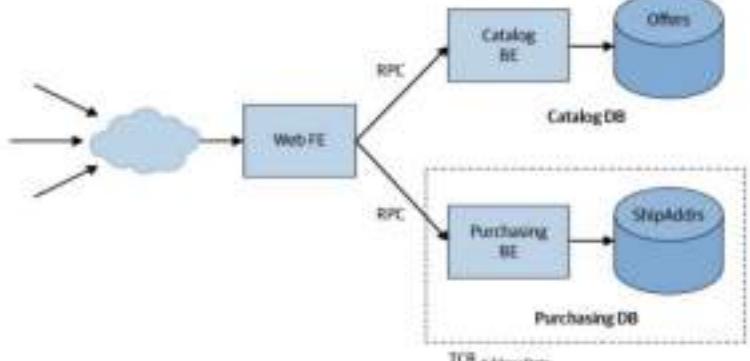
Per garantire che un sistema applichi una politica di sicurezza desiderata, è necessario comprendere e ragionare sull'intero TCB relativo a tale politica di sicurezza. Per definizione, un guasto o un bug in qualsiasi parte del TCB può comportare una violazione della politica di sicurezza.

Ragionare su un TCB diventa più difficile man mano che il TCB si allarga per includere più codice e complessità. Per questo motivo, è importante mantenere il TCB il più piccolo possibile ed escludere tutti i componenti che non sono effettivamente coinvolti nel rispetto della politica di sicurezza.

Oltre a compromettere la comprensibilità, l'inclusione di questi componenti non correlati nel TCB aggiunge rischi: un bug o un guasto in uno di questi componenti potrebbe causare una violazione della sicurezza.



Il TCB per la politica di sicurezza dei dati degli indirizzi è molto più piccolo: consiste solo nel backend di acquisto e nel suo DB e nelle loro indipendenze.



Un utente malintenzionato non può più sfruttare una vulnerabilità nel backend del catalogo per ottenere i dati di pagamento, in quanto il backend del catalogo non può accedere a quei dati in primo luogo, di conseguenza, questo design limita l'impatto delle vulnerabilità in uno dei principali componenti del sistema.

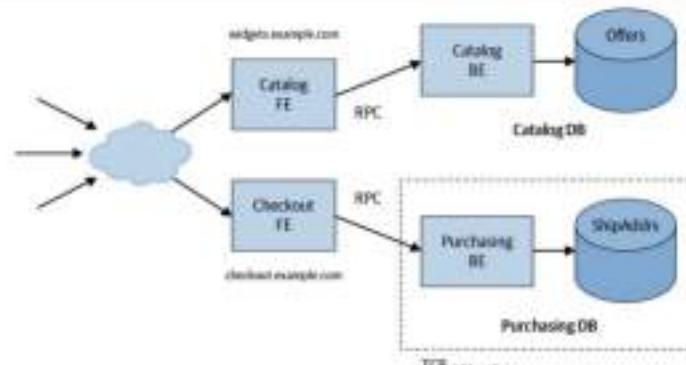
Il frontend web potrebbe servire l'intera interfaccia utente da un'unica origine web. Pertanto, una vulnerabilità nella visualizzazione del catalogo potrebbe consentire all'aggressore di accedere alle informazioni del profilo dell'utente e persino di "acquistare" articoli a nome di un altro utente.

Pertanto, in un modello di minaccia per la sicurezza web, il TCB include nuovamente tutto il fronted.

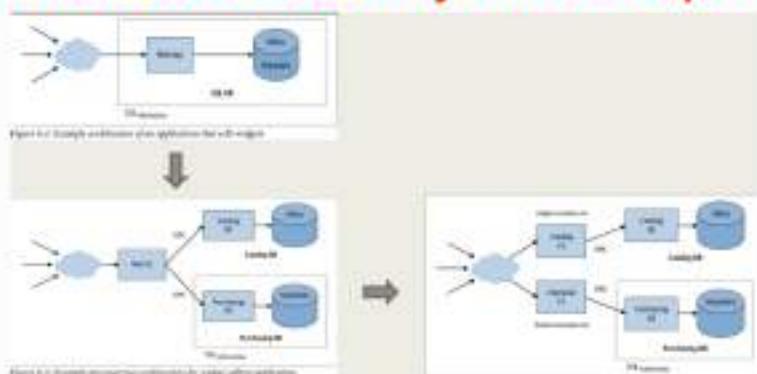
Per risolvere il problema, scomponiamo ulteriormente e utilizziamo due frontend web:

- Che implementa la ricerca e l'esplorazione del catalogo a un livello di un altro frontend responsabile dei profili di acquisto e del checkout che serve un indirizzo distinto.

Le vulnerabilità nel catalogo non possono compromettere le funzionalità di pagamento, che è segreta nella propria origine web.



Understandability or least privilege?



L'aumento di robustezza vale ovviamente solo a livello applicazione ovvero per le vulnerabilità web nel nostro caso. Se i due BE o i due (FE+BE) sono eseguiti sullo stesso SO una vulnerabilità del SO può permettere di attaccare tutti i vari moduli. Un migliore isolamento si può avere eseguendo i due Be o i due (FE+BE) su MV diverse. In questo caso è necessaria una vulnerabilità dell'hypervisor per attaccarli entrambi. Massimo aumento eseguendo i due BE o i due (FE+BE) su MV diverse, AWS ha introdotto con la NITRO architecture unità hw/sw per migliorare isolamento.

Understandability and type

Definito un tipo di dato conviene definire una funzione associata per il controllo e la validazione del tipo stesso (unica a poter validare il tipo).

A questo punto il controllo del codice si può decomporre in:

- Verifica della correttezza della funzione associata
- Verifica che ogni modulo che utilizza il tipo invochi la funzione associata.
- Altri controlli sul codice dei vari moduli.

La funzione di controllo ha funzioni simili a quelle del TCB relative però al solo tipo, purtroppo in molti linguaggi il meccanismo di encapsilazione o di definizione dei tipi non rappresenta un security boundary perché mediante parametri o reflection o conversione di tipi si può violare il tipo stesso. Quindi in generale occorre fare ipotesi che il codice che definisce il contesto di quello esaminato non sia malizioso. Ipotesi può essere valida in una organizzazione ma ci sono problemi a partire da supply chain attacks.

Design for change

Il grande numero di vulnerabilità scoperte, di nuove tecnologie d'attacco ecc. suggerisce di progettare sistemi in modo da semplificare modifiche.

Ovviamente ci sono molti aspetti di sw engineering che non possiamo trattare e quindi ci concentremo su problemi di sicurezza.

Come devono avvenire cambiamenti:

- Incrementali
- Documentali
- Testati
- Isolati
- Qualificati
- Staged = verifica i benefici di n-esimo cambiamento precedente prima di n+1-esimo.

Change = new vulnerability

Zero-Day : possono non essere quelle più critiche è più importante gestire le vulnerabilità più utilizzate o quelle raggiungibili che le zero-day, l'urgenza di cambiamento dipende da se e come attaccanti possono sfruttare le nuove vulnerabilità. Questo problema è stato discusso parlando di emulazione degli attaccanti, una vulnerabilità non raggiungibile può essere patchata con calma. Il problema è quello delle patch = sostituire codice affetto da vulnerabilità con quello patchato. Può essere utile un meccanismo a due vie.

- Via di distribuzione normale del sw aggiornato.
- Via di distribuzione rapida per eliminare vulnerabilità critiche.

Design for resilience

Progetta ogni livello di sistema in modo che sia resiliente in modo indipendente = difesa approfondita con ogni stato.

Assegnare la priorità a ciascuna funzionalità a calcolarne il costo, per capire quali funzionalità sono sufficientemente critiche da tentare di sostenere e quali sono meno importanti e possono essere limitate o disabilitate.

Compartimentare il sistema lungo confini chiaramente definiti per promuovere l'**indipendenza delle parti funzionali isolate**. In questo modo, è anche più facile costruire comportamenti di difesa complementari.

Utilizzare la ridondanza del compartimento contro i guasti localizzati. Per i guasti globali, alcuni compartimenti forniscono diverse proprietà di affidabilità e sicurezza.

Riduci i tempi di reazione **automatizzando il maggior numero possibile di misure di resilienza**. Scopri nuove modalità di errore che potrebbero trarre vantaggi dai miglioramenti dell'automazione

esistente. Mantenere l'efficacia del sistema convalidando le sue proprietà di resilienza, sia la sua risposta automatica sia qualsiasi altro attributo di resilienza del sistema.

Design for resilience - Risk assessment

Sia gli attaccanti che i difensori valutano i punti deboli di un bersaglio. Gli aggressori eseguono ricognizioni contro i loro bersagli, trovano punti deboli e quindi modellano gli attaccanti. I difensori dovrebbero limitare le informazioni esposte alle ricognizione degli attaccanti ma poichè i difensori non possono impedire completamente questa ricognizione, devono rilevarla e usarla come segnale. Un difensore ha una conoscenza interna del sistema e la valutazione può essere più dettagliata della ricognizione dell'attaccante, questo è un punto critico:

La difesa è più efficace se il difensore comprende i metodi dell'attaccante e più il difensore comprende i metodi dell'attaccante più il vantaggio è amplificato. (matrice MITRE ATT&CK)

Il difensore dovrebbe evitare i punti ciechi perché considera alcuni vettori di attacco improbabili o irrilevanti.

Design for resilience - Degradation

Se non puoi impedire tutti gli schieramenti degli avversari, puoi limitare il raggio di esplosione degli attacchi.

I livelli di sistema sono complementari, con ogni livello che anticipa i punti devoli o i probabili fallimenti del precedente. Man mano che le attivazioni della difesa si spostano attraversano gli strati, i segnali di un compromesso diventano più forti, per concentrare gli sforzi su probabili attacchi.

Quando si progetta in modo approfondito per la difesa, si presume che i componenti del sistema o addirittura interi sistemi possano guastarsi. Ciò accade per i motivi quali danni fisici, un malfunzionamento dell'hw o della rete, una configurazione errata o un bug del sw o una compromissione della sicurezza.

Quando un componente si guasta, anche il pool globale di risorse simili si riduce.

Quando il carico di un sistema supera la sua capacità, la sua risposta inizia inevitabilmente a peggiorare e ciò può portare a un sistema completamente guasto senza disponibilità.

A meno che questo scenario non sia pianificato in anticipo, è difficile sapere dove il sistema potrebbe rompersi, ma molto probabilmente sarà dove il sistema è più debole e non dove è più sicuro.

Per controllare il degrado, selezionare le proprietà da disabilitare quando si verificano circostanze terribili, facendo tutto il possibile per proteggere la sicurezza del sistema. Se si progettano

deliberatamente più opzioni di risposta per circostanze come queste, il sistema può avere punti di interruzione controllati, piuttosto che un collasso caotico.

Design for resilience - Blast radius

E' consigliabile suddividere in compartimenti la maggior parte degli aspetti dei sistemi: server, applicazioni, storage e così via. Controllo del raggio dell'esplosione = suddivisione in compartimenti dell'impatto di un evento, poiché i compartimenti su una nave garantiscono la resilienza contro l'affondamento dell'intera nave.

Quando si utilizza una configurazione a rete singola, un utente malintenzionato che compremette le credenziali di un utente può potenzialmente accedere a tutti i dispositivi della rete.

Quando si utilizza la compartmentazione, tuttavia, una violazione della sicurezza o un sovraccarico di traffico in un compartimento non compromette tutti i compartimenti.

Quando si progetta per la resilienza è necessario creare barriere compartmentali che limitare sia gli aggressori che i guasti accidentali.

Queste barriere consentono di personalizzare e automatizzare meglio le risposte. Questi limiti possono creare domini di errore che forniscono ridondanza dei componenti e isolamenti degli errori.

Design for resilience – compartmentalization

Isolare gli elementi del sistema e abilitare e controllare le interazioni essenziali per lo scopo previsto.

Domanda, è composta da parti distinte con interazioni limitate?

Compartizionare:

- Spezzare il nostro mondo interconnesso; si tratta di costruire sistemi come elementi che lavorano in modo isolato e definiscono e controllano le loro interazioni.
- Assicuriamo che l'architettura dei sistemi faciliti la sicurezza.

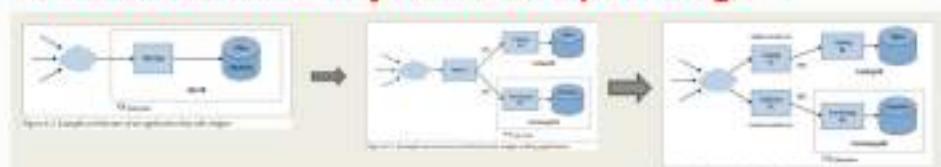
Troppo spesso la sicurezza è vista come un'arma aggiuntiva all'esterno dei sistemi esistenti per gestirne le vulnerabilità. Ciò può comportare un'insicurezza che potrebbe essere prevenuta o risolta a livello di architettura. La sicurezza più solida è integrata nell'architettura stessa dei sistemi.

La compartmentazione ci insegna a costruire sistemi definiti, discreti e limitati.

Non dovremmo vedere il mondo solo in termini di "abbiamo bisogno di una serratura su quella porta"; a volte, dobbiamo fare "non abbiamo nemmeno bisogno di una porta qui". E se hai bisogno di qualcosa, forse è una tana del topo, o una fessura per la carta, o uno spioncino. Il mondo della sicurezza informatica ha

bisogno di meno buttafori in continua crescita.

Understandability or least privilege?



Le trasformazioni viste migliorano il blast radius se consideriamo unicamente le vulnerabilità web.

Se eseguiamo su due macchine fisiche diverse miglioriamo:

- Rispetto anche a vulnerabilità del SO
- Rispetto a guasti accidentali (failure domains)
- Rilevazione di anomalie (ad esempio dovute a DOS)

Un altro modo di creare compartimenti è di usare il Chinese Wall security model = un soggetto che accede ad un certo oggetto non può accedere agli altri. Utile per confinare attacchi durante una operazione limitando oggetti che quella operazione può accedere = costi e medicine in un sistema per la gestione di dati ospedalieri.

Design for resilience - Graceful Degradation

E' meglio fare le scelte essenziali e difficili in anticipo piuttosto che sotto come in la pressione durante un incidente.

Liberare risorse e ridurre le operazioni non riuscite disabilitando le funzionalità utilizzate di rado, funzioni meno critiche o capacità di servizio ad alto costo.

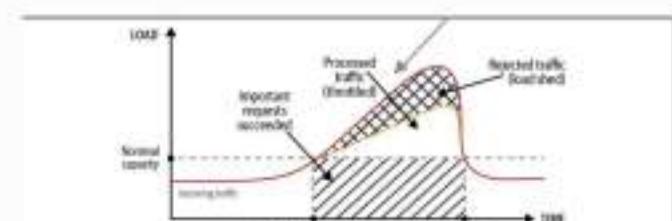
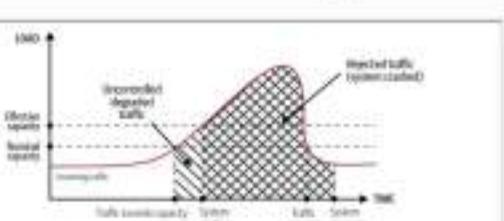
Applicare le risorse liberate per preservare caratteristiche e funzioni importanti.

Puntare affinché le misure di risposta abbiano effetto in modo rapido e automatico. Questo è più facile con server sotto controllo, poiché aggiornare arbitrariamente i parametri operativi. I client degli utenti sono più difficili da controllare perché i dispositivi client potrebbero posticipare o non essere in grado di ricevere gli aggiornamenti.

Comprendere quali sistemi sono critici per la missione dell'azienda e la loro relativa importanza e interdipendenze. Potresti preservare le caratteristiche minime dei sistemi in proporzione al loro valore relativo.

Un miglioramento della capacità del sistema di assorbire il carico o il guasto fornisce un supporto fondamentale a tutti gli altri meccanismi di resilienza e offre ai soccorritori più tempo per rispondere.

Se i singoli sistemi hanno una chiara strategia di degrado, è più semplice dare priorità al degrado a ambito più ampio, su più sistemi o aree di prodotto.



2° immagine, rifiuto di servire a bassa priorità con conseguente blocco dei clienti utenti che generano a bassa priorità.

Failing safe vs failing secure

La gestione dei guasti deve essere bilanciata tra l'ottimizzazione dell'affidabilità con guasti aperti (sicuri) e l'ottimizzazione della sicurezza con guasti chiusi (sicuri).

Per massimizzare l'affidabilità, un sistema deve resistere ai guasti e servire il più possibile in presenza di incertezza. Anche se l'integrità non è intatta, finché la sua configurazione è valida, un sistema ottimizzato per la disponibilità servirà ciò che può. Se le ACL non sono state caricate, l'ACL predefinita è "allow all".

Per massimizzare la sicurezza, un sistema dovrebbe bloccarsi completamente in casi di incertezza. Se il sistema non è in grado di verificare l'integrità, indipendentemente dal fatto che un disco guasto abbia portato via le configurazioni o che un aggressore le abbia modificate, il sistema non può funzionare e deve proteggersi il più possibile. Se le ACL non sono state caricate, l'ACL predefinita presunta è "deny all".

Le operazioni critiche per la sicurezza non devono fallire, perché un aggressore può degradare la sicurezza di un sistema usando solo attacchi DoS (si pensi a un firewall).

La degradazione controllata delle operazioni critiche per la sicurezza è possibile. Un componente a basso costo potrebbe sostituire i controlli di sicurezza che non funzionano e se applica controlli più forti, la rottura non è efficace. In questo modo si aumenta effettivamente la resilienza.



Location separation

Limitiamo l'impatto di un attaccante utilizzando una dimensione aggiuntiva: la posizione in cui il servizio è in esecuzione, per evitare che un avversario che ha compromesso fisicamente un singolo data center sia in grado di leggere i dati in tutti gli altri data center.

Per ridurre il rischio di insider, limitare i permessi di accesso degli utenti amministrativi più potenti a regioni specifiche, il modo più ovvio per ottenere una separazione delle sedi è quello di eseguire gli stessi servizi in sedi diverse (come i data center o le regioni cloud, che in genere corrispondono a sedi fisiche diverse). Si possono usare i normali meccanismi di controllo degli accessi per

proteggere le istanze dello servizio in luoghi diversi, così come si proteggono servizi diversi gli uni dagli altri.

La separazione della posizione aiuta a resistere a un attacco che si sposta da una posizione all'altra. I compartimenti crittografici basati sulla posizione consentono di limitare l'accesso alle applicazioni e ai loro dati memorizzati a luoghi specifici, contenendo il raggio d'azione degli attacchi locali.

La locazione fisica è un confine naturale di compartmentazione, poiché molti eventi avversi sono collegati a luoghi fisici. Ad esempio, i disastri naturali sono circoscritti a una regione, così come altri incidenti localizzati come interruzioni di fibra, interruzioni di corrente o incendi.

Gli attacchi malevoli che richiedono la presenza fisica dell'aggressore sono inoltre limitati ai luoghi che l'aggressore può effettivamente raggiungere, e tutti, tranne i più abili (quelli sponsorizzati dallo stato), probabilmente non hanno la capacità di attaccare più luoghi tutti insieme.

Allineamento dell'architettura fisica e logica

Quando si compartimenta un'architettura in domini di guasto e sicurezza, è importante allineare i confini fisici e logici. È possibile segmentare la rete sia in base ai rischi a livello di rete (reti esposte al traffico dannoso rispetto a quelle interne affidabili) sia in base ai rischi di attacchi fisici. È possibile:

- Applicare la segregazione di rete tra ambienti aziendali e di produzione in edifici separati.
- Suddividere ulteriormente la rete aziendale per separare le aree ad alto traffico di visitatori, come le aree per conferenze e riunioni.

Un attacco fisico, come il furto o il backdooring di un server, può consentire a un aggressore di accedere alle chiavi di crittografia o alle credenziali, che poi si traducono in ulteriori penetrazioni.

Questo può essere evitato distribuendo in modo logicamente compartimentato segreti, chiavi e credenziali ai server fisici per ridurre al minimo il rischio di una compressione fisica. Ad esempio, se si gestiscono server web in diversi datacenter fisici, è possibile distribuire un certificato separato per ogni server o condividere un certificato solo tra i server di una sede, invece di condividere un unico certificato per tutti i server.

Isolation of trust

Un servizio potrebbe anche rifiutare provenienti da luoghi con cui non si aspetta di comunicare, a tal fine, è possibile limitare le comunicazioni per impostare predefinita e consentire solo le comunicazioni previste attraverso i confini delle sedi.

Per limitare di controllo dei lavori di Google certifica ed esegue i

lavori in produzione.

- Quando il sistema certifica un lavoro da eseguire in un determinato comparto, annota il certificato del lavoro con i metadati di localizzazione del comparto.
- Ogni sede ha una propria copia del sistema di controllo dei lavori che ne certifica l'esecuzione e le macchine di quella sede accettano solo lavori provenienti da quel sistema.
- In questo modo si impedisce a un attaccante di perforare il confine del compartimento e di colpire altre posizioni.

Limitations of location-based trust

Nell'infrastruttura di rete di Google la posizione non implica alcuna fiducia. Invece, secondo la rete zero trust, una posizione di lavoro viene considerata affidabile sulla base di un certificato rilasciato alla singola macchina e di asserzioni sulla sua configurazione (come ad esempio un software aggiornato).

L'inserimento di una macchina non attendibile in una porta di rete del piano dell'ufficio la assegnerà a una VLAN guest non attendibile. Solo le postazioni di lavoro autorizzate (autenticate tramite il protocollo 802.1x) vengono assegnate alla VLAN appropriata.

Google non si basa nemmeno sull'ubicazione fisica per stabilire l'affidabilità dei server nei data center dopo una valutazione Red Team dell'ambiente di un data center.

- La squadra Rossa ha posizionato un dispositivo wireless in cima a un rack e lo ha rapidamente collegato a una porta aperta, per consentire un'ulteriore penetrazione della rete interna del data center dall'esterno dell'edificio.
- Poi un attento tecnico del datacenter ha chiuso ordinatamente il cablaggio dell'access point, apparentemente offeso dal lavoro di installazione disordinato e ritenendo che il dispositivo fosse legittimo.

Ciò dimostra la complessità dell'attribuzione della fiducia in base alla posizione fisica, anche all'interno di un'area fisicamente sicura.

Isolation of confidentiality

Una volta che abbiamo un sistema per isolare la fiducia, dobbiamo isolare le nostre chiavi di crittografia per garantire che i dati protetti da una radice di crittografia in una sede siano compromessi dall'esfiltrazione delle chiavi di crittografia in un'altra sede. Ad esempio, se una filiale di un'azienda viene compromessa, gli aggressori non devono essere in grado di leggere i dati delle altre filiali.

Per isolare la crittografia e l'impacchettamento delle chiavi in un luogo, dobbiamo assicurarci che le chiavi principali di un luogo siano disponibili solo per il luogo corretto = il sistema di

distribuzione che colloca le chiavi principali solo nei luoghi corretti. Un sistema di accesso alle chiavi dovrebbe sfruttare l'isolamento della fiducia per garantire che queste chiavi non possano essere consultate da intitâ esterne alla sede appropriata.

In base a questi principi, una determinata posizione consente l'uso di ACL sulle chiavi locali per impedire agli aggressori remoti di decifrare i dati. La decodifica viene impedita anche se gli aggressori hanno accesso ai dati crittografati (tramite compromissione interna o esfiltrazione).

Failure domains

Per affrontare i guasti completi dei componenti del sistema, la progettazione del sistema deve incorporare ridondanze e domini distinti in cui un guasto in un dominio non influisce sui componenti dell'altro dominio di guasto.

I domini di guasto realizzano l'isolamento funzionale suddividendo un sistema in più copie equivalenti ma completamente indipendenti.

Un dominio di guasto appare come un singolo sistema ai suoi client. Se necessario, ogni partizione può sostituire l'intero sistema durante un'interruzione. Poiché una partizione dispone solo di una frazione delle risorse del sistema, può supportare solo una frazione delle capacità del sistema.

La gestione dei domini di guasto e il mantenimento del loro isolamento richiedono un impegno costante, ma questi domini aumentano la resilienza del sistema come altri controlli di tipo blast radius non possono fare.

Domini di fallimento:

- Aiutare a proteggere i sistemi dall'impatto globale, perché un singolo evento non colpisce in genere tutti i domini di guasto contemporaneamente.
- Limitare l'impatto dei guasti ed evitare il collasso completo. E' particolarmente importante ridurre i guasti dei componenti critici, poiché qualsiasi sistema che dipenda da componenti critici guasti è anche a rischio di guasto completo.

Data isolation

Esiste la possibilità di avere dati errati all'origine dei dati o all'interno dei singoli domini di guasto.

Pertanto, ogni istanza del dominio di guasto necessita di una propria copia dei dati per essere funzionalmente indipendente dagli altri domini di guasto.

Un duplice approccio per ottenere l'isolamento dei dati:

- Si limita il modo in cui gli aggiornamenti dei dati possono entrare in un dominio di errore. Un sistema accetta nuovi dati solo dopo aver superato tutti i controlli di convalida per le

modifiche tipiche e sicure. Alcune eccezioni vengono giustificate e un meccanismo di breakglass può consentire l'ingresso di nuovi dati nel dominio di errore.

- In secondo luogo, la possibilità di scrivere su disco gli ultimi dati validi conosciuti rende i sistemi resistenti alla perdita dei dati di accesso. Alcuni sistemi conservano i vecchi dati per un periodo di tempo limitato, nel caso in cui i dati più recenti siano danneggiati per qualsiasi motivo. Questo è un altro esempio di difesa in profondità, che contribuisce a fornire una resilienza a lungo termine.

Practical considerations

Anche solo due domini di guasto forniscono funzionalità di regressione A/B e limitano il raggio d'azione delle modifiche al sistema a un solo dominio. Per ottenere questo risultato, un dominio funge da canarino con un criterio che non consente gli aggiornamenti a entrambi i domini di guasto contemporaneamente, geograficamente separati garantiscono l'isolamento in caso di disastri naturali.

L'utilizzo di versioni sw diverse in domini di guasto diversi riduce le possibilità che un bug distrugga tutti i server o danneggi tutti i dati. Un dominio di guasto può subire un guasto completo se anche uno solo dei suoi componenti critici si guasta possiamo utilizzare componenti alternativi per ridurre al minimo la probabilità di guasto di tutti i domini.

I componenti ad alta capacità sono quelli che servono gli utenti e assorbono i picchi di richieste degli utenti o il consumo di risorse dovuto alle nuove funzionalità. Assorbono anche il traffico Dos, fino a quando non entra in azione la mitigazione DoS o il degrado aggravato. E' necessario impegnarsi per ridurre al minimo la probabilità di un loro guasto che abbia un impatto su tutti gli utenti. Riduciamo il rischio di un guasto dei componenti ad alta capacità implementando un sistema ad alta disponibilità, cioè, copie dei componenti che offrono una probabilità di interruzione più bassa. Per ottenere questa minore probabilità, le copie hanno meno dipendenze e un tasso di modifiche limitato. Ad esempio, questo componente può utilizzare:

- I dati sono memorizzati nella cache locale piuttosto che in un DB remoto (meno dipendenze).
- Codice e configurazioni più vecchie per evitare bug recenti nelle versioni più recenti (basse tasso di modifica).

Controlling Redundancies

I sistemi ridondanti sono configurati in modo di avere più di una singola opzione per ciascuna delle loro dipendenze, la gestione della scelta tra queste opzioni non è sempre semplice e gli aggressori possono potenzialmente sfruttare le differenze tra i sistemi ridondanti.

Un progetto resiliente raggiunge la sicurezza e l'affidabilità senza sacrificare l'una per l'altra. Semmai, quando le alternative a bassa dipendenza hanno una sicurezza più forte, questo può servire da disincentivo per gli aggressori che intendono logorare il vostro sistema. La fornitura di un insieme di backend, si solito attraverso tecnologie di bilanciamento del carico, aggiunge resilienza in caso di guasto di un backend.

Di solito il sistema considera i backend ridondanti come "intercambiabili", a patto che tutti i backend forniscano gli stessi comportamenti.

Un sistema che necessita di un'affidabilità diversa dovrebbe affidarsi a un insieme di backend intercambiabili che forniscono i comportamenti di affidabilità desiderati.

Il sistema stesso deve determinare quale serie di comportamenti utilizzare e quando utilizzarli.

In questo modo si ottiene il pieno controllo dell'affidabilità del sistema, soprattutto durante il ripristino.

Considerazioni

E' evidente che il gruppo è fortemente influenzato dalla gestione di un'architettura cloud, frontend e backend come in qualsiasi applicazioni eseguite su ckoud che sono fondamentalmente applicazioni web.

I guasti hw sono quasi trascurati a favore di quelli sw (malfunzionamenti e crash) questi non possono essere mascherati mentre è relativamente semplice mascherare guasti hw vista la naturale ridondanza nel cloud.

Una caratteristica di applicazioni per cloud sono le copie dei server gestite con load balacing (con numero di copie che può aumentare in base al carico).

Esistenza di location diverse del cloud per robustezza rispetto ad eventi fisici ed incidenti umani.

Decomposizione parallela assunta quasi implicitamente (ottimo...) nessuno vincolo sulla quantità di risorse disponibili e su vincolo in tempo reale (in ICS non è nemmeno un problema di costo ma di approccio al progetto).

Convalida

Sia dal punto di vista dell'affidabilità che da quello della sicurezza, non c'è alternativa all'esercizio effettivo del sistema e alla convalida per essere sicuri.

- Che i sistemi si comportino come previsto sia in circostanze normali che inaspettate.
- Che le nuove funzionalità o le correzioni di bug non erodano gradualmente i meccanismi di resilienza stratificati.

La convalida si concentra sull'osservazione del sistema in circostanze "realistiche" ma "controllate".

A differenza dell'ingegneria del caos, che è di natura esplorativa, la validazione conferma le proprietà e i comportamenti specifici del sistema di interesse.

Una strategia generale di manutenzione della convalida comprende:

- Scoprire nuovi fallimenti
- Implementazione di validatori per ogni fallimento
- Esecuzione ripetuta di tutti i validatori
- Eliminare gradualmente i validatori quando le caratteristiche o i comportamenti rilevati non esistono più.

Alcune fonti per scoprire i fallimenti rilevanti:

- Segnalazioni regolari di bug da parte di utenti e dipendenti
- Fuzzing e approcci simili al fuzzing solo hw
- Approcci a iniezione di guasti hw(simili alla scimmia del caos)
- Giudizio analitico degli esperti in materia che gestiscono i vostri sistemi

Fine appunti

Grazie per la lettura e se notate imperfezioni/erori cercatemi su github @Landisit8 o telegram @Landisit8

Grazie per la vostra lettura.