Sicurezza dei Sistemi e delle Reti

File di Nali

Indice

Capitolo 1

Politiche di Sicurezza

1.1 Definizione Politica di sicurezza

La *gestione della sicurezza* è un *processo formale* per rispondere alle domande:

- quali sono i beni da proteggere
- quali sono le possibili minacce
- come si possono contrastare le minacce

Questo processo ha natura iterativa, ed è contenuto nella ISO 31000; in questa norma viene descritto un *modello per la gestione della sicurezza delle informazioni* che comprende le seguenti fasi:

- **Plan:**
  - stabilire politiche, processi e procedure di sicurezza
  - eseguire la valutazione del rischio
  - sviluppare un piano di trattamento del rischio
- **Do:**
  - implementare il piano di trattamento del rischio
- **Check:**
  - monitorare e mantenere il piano di trattamento del rischio
- **Act:**
  - mantenere e migliorare la gestione dei rischi
  - risposta ad incidenti, analisi di vulnerabilità e riprendere il ciclo iterativamente

1.1.1 Definizioni

- **Rischio:** esprime la possibilità che un attacco causi danni ad una organizzazione
- **Risorsa:** tutto ciò che necessita di essere protetto
  - *hw*
  - *sw*
  - *reputazione*

  La valutazione di una risorsa viene fatta in base:
  - ai costi da sostenere per sostituire la risorsa nel caso non sia più disponibile
  - perdita di incassi in caso di attacco
- **Vulnerabilità:** punti deboli che possono essere sfruttati per causare danni al sistema; possono essere classificate come:
  - critico
  - moderato
  - basso

Definizione per domanda esame: Una politica di sicurezza dei sistemi e delle reti è l'insieme strutturato di regole, requisiti e comportamenti che un'organizzazione definisce affinché i propri sistemi informatici e le proprie infrastrutture di rete applichino in modo coerente misure di protezione contro accessi non autorizzati, abusi, perdite di dati e altri rischi. Essa rappresenta ciò che il sistema deve far rispettare, automatizzando e facendo valere controlli come l'autenticazione, l'autorizzazione, il monitoraggio delle attività e la gestione delle vulnerabilità, allo scopo di garantire la riservatezza, l'integrità e la disponibilità delle informazioni, secondo gli obiettivi stabiliti dall'organizzazione stessa.

1.2 Domanda MAC, DAC e RBAC

definire l'utilizzo delle politiche di sicurezza basate su MAC, DAC e RBAC

Gli approcci DAC,MAC e RBAC fanno parte delle politiche di controllo degli accessi e forniscono approcci diversi a seconda del contesto di applicazione

1.2.1 DAC (Discretionary Access Control):

Il controllo dell'accesso viene fatto sull' **identità del soggetto richiedente** e delle **regole di accesso**. Definito *discrezionale* poichè un'entità potrebbe avere i privilegi di accessi che le permettono, a sua volta, di concedere l'accesso ad un'altra entità.
Si può rappresentare mediante una matrice, dove:

- le colonne rappresentano i soggetti
- le righe gli oggetti
- ogni cella specifica i diritti di accesso di quel soggetto a quel determinato oggetto

1.2.2 MAC (Mandatory Access Control):

La politica di controllo degli accesssi MAC, o Mandatory Access Controll si basa sul confronto tra **etichette di sicurezza** che indicano quanto sono sensibili le risorse e **autorizzazioni di sicurezza** che indicano quali entità del sistema sono idonee ad accedere a quali risorse.
Questa politica è definita *mandatoria* poichè un'entità che possiede l'accesso a una risorsa non può estendere il permesso di acesso a un'altra entità, può farlo solo l'amministratore di sistema.
I sistemi MAC si dividono in:

- **Multilevel security systems:** consiste in una struttura verticale di sistemi di sicurezza, agli utent viene assegnato un livello e possono accedere solo a risorse con un livello uguale o inferiore
- **Multilateral security systems:** l'accesso viene assegnato in base a segmenti che formano gruppi costituiti da livelli di sicurezza e parole in codice
  - si ottiene una struttura orizzontale, che contiene livelli di sicurezza verticali aggiuntivi

Vantaggi:

- molto sicuro, a prova di manomissione
- gli utenti non possono fare modifiche
- controllo automatizzato
- i dati non possono essere modificati senza apposita autorizzazione

Svantaggi:

- richiede una pianificazione dettagliata e un lavoro amministrativo
- controllo e aggiornamento dei dati di accesso
- manutenzione per aggiunta di nuovi dati o utenti e relative modifiche (elevato carico di lavoro per l'amministratore)

1.2.3 RBAC (Role Based Access Control):

Introduce il concetto di **ruolo**, ovvero una funzione che può essere assiciata a uno o più utenti (gli utenti possono avere più ruoli) e una **sessione**, ovvero una mappatura tra utente e un sottoinsieme di ruoli a cui è assegnato.
I ruoli di un utente possono fornire o meno accesso a determinate risorse.
Quattro tipi di entità:

- **Utente:** una persona che ha accesso al sistema, ogni individuo ha un ID associato
- **Ruolo:** funzione lavorativa all'interno dell'organizzazione
- **Autorizzazione:** approvazione di una modalità di accesso ad uno o più oggetti
- **Sessione:** mappatura tra utente e un sottoinsieme dei ruoli a cui è assegnato

1.3 Fare cenni sull'utilizzo di tali politiche nei Sistemi Operativi moderni

1.3.1 Unix security model

In Linux ci sono tre entità da considerare:

- **Soggetto:** può essere un utente o un processo
- **Oggetto:** file, cartelle, ...
- **Operazioni consentite:** lettura, scrittura, esecuzione

In Unix, ogni utente ha associato un id univoco, detto **UID**; può appartenere a gruppi di utenti, anch'essi identificati da un id univoco detto **GID**. Tutti gli utenti appartenenti ad un gruppo possono condividere tra loro oggetti.
Ad ogni file è assegnato un unico utente proprietario e un unico gruppo proprietario. L'autorizzazione viene concessa mediante una ACL che identifica le operazioni che i soggetti possono fare.

Processi in Linux

Ogni processo è isolato dagli altri e non possono accedere alla memoria altrui.
Ogni processo viene eseguito con le autorizzazione dell'UID dell'utente che lo sta eseguendo.
Nel momento della creazione, ad ogni processo sono assegnati tre ID (inizialmente tutti uguali all'UID):

- **Effective UID:** determina le autorizzazioni per il processo

- **Real UID:** determina l'utente che ha avviato il processo
- **Saved UID:** EUID prima di eventuali modifiche

L'utente *root* può cambiare EUID/RUID/SUID a valori arbitrari; utenti non privilegiati possono cambiare EUID solo a RUID o SUID

Unix file access control

Le modifiche agli ID sono apportate mediante i comandi *setUID* e *setGID*; questa modifica permette ai programmi non privilegiati di accedere a risorse generalmente non accessibili.
Le directory possono aver impostato uno ***sticky bit***: specifica che solo il proprietario di un file nella cartella può apportare una modifica a quel file
Il ***superuser*** è esente dalle consuete restrizioni di controllo degli accessi, ha accesso a tutto il sistema.

1.3.2 Windows security architecture

L'architettura di sicurezza di Windows è basata su più entità:

- ***Security Reference Model (SRM):*** componente che in modalità kernel esegue controlli delle autorizzazioni e manipola i privilegi degli utenti
- ***Local Security Authority (LSA):*** risiede in un processo utente, è responsabile dell'applicazione della politica di sicurezza locale, tra cui:
  - criteri per le password, come complessità e tempi di scadenza
  - politica di controllo → specifica quali operazioni su quali oggetti vadano controllate
  - impostazioni dei privilegi
- ***Security Account Manager (SAM):*** è un database che archivia i dati degli account e le informazioni di sicurezza su entità locali e gruppi
- ***Active Directory (AD):*** implementa il protocollo LDAP (*Lightweight Directory Access Protocol*)

Windows security model

Windows ha un complesso sistema di controllo dell'accesso; ogni oggetto ha ACL per permettere autorizzazioni granulari ad utenti e/o gruppi di utenti.

Security descriptor

Ogni oggetto ha un *security descriptor* che contiene:

- ***security identifier* (SID)** per il possessore e il gruppo primario dell'oggetto (SID è associato univocamente ad ogni utente)

- ***discretionary ACL* (DACL):** diritti di accesso per gli utenti e i gruppi
- ***system ACL* (SACL):** tipi di accesso che generano log

Ad ogni processo viene inoltre associato un insieme di **token**, che prende il nome di ***security context***. Nel momento in cui un processo vuole accedere ad un oggetto, presenta il suo insieme di token e il sistema controlla se il security context abbia o meno accesso a tale risorsa in base al *security descriptor* dell'oggetto.

Capitolo 2

Set-UID

Capitolo 3

Malware

Una definizione informale per malware potrebbe essere quella di *programma malevolo*, solitamente inserito di nascosto in un sistema, che ha lo scopo di compromettere la **riservatezza**, l'**integrità** o la **disponibilità** del sistema stesso.
I malware sono classificati in base a:

- **Propagazione:** software, rete, social engeneering
- **Azioni sui dati colpiti:** corruzione, furto, crittografia
- **Attack kit:** strumenti già pronti per attaccare
- **Attori e/o motivazioni dell'attacco**

3.1 Differenza tra due tipologie di malware (virus e worm) + esempio

Sono stati chiesti questi due ma per sicurezza aggiungo anche gli altri

3.1.1 Trojan

È un programma che ha un effetto evidente e atteso dall'utente, che ha però anche un effetto **nascosto** che viola le politiche di sicurezza e che viene condotto senza l'autorizzazione dell'utente

3.1.2 Virus

È un codice che può replicarsi modificando altri file o programmi per inserire codice in grado di replicarsi a sua volta; questa **proprietà di replicazione** è ciò distingue i virus dagli altri tipi di malware. Non svolge nessuna azione evidente, ma cerca di rimanere nell'ombra.

La replica richiede un certo tipo di assistenza da parte dell'utente, come ad esempio cliccare su un allegato.
Un virus è composto da tre parti:

- **Meccanismo di infezione**
- **Trigger:** evento che determina quando il payload viene attivato
- **Payload:** cosa fa il virus (oltre a diffondersi)

I virus attraversano quattro fasi:

1. **Dormiente:** il virus è inattivo in attesa di essere attivato
2. **Scatenante:** il virus viene attivato
3. **Propagazione:** il virus inserisce una copia di sé stesso in certe parti del sistema; ogni programma infetto conterrà ora un altro virus che entrerà a sua volta in fase di propagazione
4. **Esecutiva:** la funzione viene eseguita

Vettori di infezione

I principali vettori di infezione sono:

- **Boot sector** di dispositivi esterni; il codice è inserito nel boot sector e viene eseguito in fase di avvio
- **Eseguibili**
- **File macro:** il virus si attacca ai documenti per propagarsi

Esempio noto in letteratura:

Un esempio di virus può essere considerato il compression virus, che va a comprimere lo spazio occupato da un programma, per inserire un codice malevolo: così facendo la dimensione di un file è la stessa, andando a baypassare i controlli di un antivirus.

3.1.3 Worm

I worm sono programmi *stand alone* (a differenza dei virus che devono essere attivati da un qualche evento) in grado di replicarsi.
Le fasi di esecuzione sono:

- **Probing:** cerca informazioni sulla macchina
- **Expoloitation:** sfrutta le informazioni raccolte per trovare vulnerabilità
- **Replicazione**
- **Attacco** (payload)

Esempio noto in litteratura:

Esempio di worm della famiglia Nimda, che prese di mira i sistemi operativi Microsoft Windows.

3.1.4 Drive-by-download

Sfruttano **vulnerabilità del browser** per installare codice malevolo ad insaputa dell'utente nel momento in cui visita la pagina web dell'attaccante.

3.1.5 Clickjacking

L'attaccante intercetta un *click* dell'utente per costringerlo a fare delle cose contro la sua volontà.

3.1.6 Zombie e botnet

Lo *zombie* è una singola macchina, mentre la *botnet* è un insieme di macchine zombie controllate da una singola entità; vengono usate per fare DDoS, phishing, spamming, ...

3.1.7 Rootkit

È un insieme di programmi installati su un sistema per mantenere l'accesso ad un sistema, ad esempio, con privilegi di amministratore, nascondendo le prove della sua presenza e aggirando i meccanismi di controllo.
Permettono di fare attacchi anche con scarse conoscenze tecniche.

3.1.8 Scareware

Software che hanno lo scopo di diffondere shock, ansia e/o la percezione di una minaccia; sono un attacco di *social engeneering*.

3.1.9 Ransomware

Software che tiene in ostaggio il sistema per richiedere un riscatto all'utente, spesso tramite cifratura.

3.1.10 Vulnerabilità zero-day

Si intende un'exploit non ancora nota e che non ha quindi una contromisura.

3.1.11 Spear phishing

Viene **studiato nel dettaglio il bersaglio**, in modo tale da fare del phishing più mirato ed efficace.

3.1.12 Spyware

Malware che raccoglie piccole informazioni alla volta sugli utenti a loro insaputa, come ad esempio un *keylogger*.

3.1.13 APT - Advanced Persistent Threats

- ***Advanced:*** è un'applicazione con un ampia varietà di tecnologie di intrusione e malware
- ***Persistent:*** attacchi per un periodo di tempo prolungato verso il target
- ***Target:*** target selezionati in modo accurato

Le fasi principali di un attacco tramite APT sono:

- **Ricognizione:** si sceglie una vittima e la si studia
- **Weaponization:** si mette un trojan che permette accesso remoto in un payload consegnabile (email, USB, web)
- **Sfruttamento:** il codice malevolo viene attivato per portare a termine il suo scopo

3.2 Differenza tra virus metamorfico e polimorfico

3.2.1 Classificazione dei virus

È possibile classificare i virus in base alle **tecniche usate per superare i controlli di sistema:**

- **Cifratura del virus:** crea una chiave per *"crittografarsi"*; quando viene chiamato un programma infetto, con tale chiave viene decifrato il virus. Per evitare pattern di bit, durante la propagazione la chiave viene cambiata
- **Stealth virus:** si nasconde dal rilevamento da parte dell'antivirus, tramite mutazione o compressione del codice
- **Polymorphic virus:** durante la replica crea copie che svolgono la stessa funzione ma che hanno pattern di bit diversi
- **Metamorphic virus:** si riscrive completamente ad ogni iterazione per aumentare la difficoltà di rilevamento
- **Compression virus:** comprimono il file eseguibile in modo che la versione infetta abbia la stessa dimensione di quella originale

3.3 Discutere se le seguenti tecniche sono meccanismi di rilevamento utili rispettivamente per i virus polimorfici e metamorfici:

3.3.1 Static pattern matching

Utile per **virus polimorfici** poichè cambiano solo la parte cifrata del loro codice a ogni infezione, ma mantengono invariata la struttura del decryptor, che può essere riconosciuto con tecniche di pattern matching statico, perché rimane simile o identico tra le varianti.
Al contrario dei virus metamorfici che modificano completamente il proprio codice a ogni infezione, incluso il decryptor o qualsiasi parte riconoscibile.

3.3.2 Pattern matching during emulation

Utile per **virus metamorfici** poichè nonostante essi cambino completamente aspetto a ogni infezione, durante l'emulazione (cioè durante l'esecuzione simulata del codice in un ambiente controllato), il virus esegue le stesse azioni fondamentali, indipendentemente dalla sua forma.
La tecnica di pattern matching during emulation permette quindi di osservare il comportamento reale del codice eseguito (come accessi a file di sistema, chiamate di sistema sospette, o autoriproduzione).
Al contrario, emulare un virus polimorfico è inefficiente, perché richiede tempo e risorse per ottenere un'informazione che è già ricavabile staticamente.

3.3.3 Suspicious behaviour detection

Utile per **virus polimorfici** poichè il loro comportamento rimane molto prevedibile:

- Decriptano il corpo
- Si copiano in altri file
- Cercano persistenza

Per quanto riguarda i virus metamorfici possono cercare di mascherare il loro comportamento, soprattutto quelli più avanzati possono simulare comportamenti legittimi.
Questo li rende più difficili da intercettare solo con behaviour detection

Capitolo 4

Autenticazione

4.1 Discutere le problematiche di autenticazione su Web e in dettaglio uno schema challenge-response

L'autenticazione in generale, e in particolare quella web, consiste principalmente in un client che fa una richiesta a un server e un server che fornisce una risposta. Per fare in modo però che il server non inivii informazioni a client malevoli, che possono attaccare tramite attacchi di spoofing, fingendosi un client legittimo (o anche fingendosi un server legittimo), oppure con replay attack, ciò inivio di pacchetti già inviati da un client legittimo(tramite sniffing sulla rete), vengono introdotte misure di sicurezza più avanzate.
Client e server condividono informazioni segrete, che possono andare da una password a una chiave di crittografia(secret): un client che vuole accedere a un servizio web su un server, deve dimostrare di essere chi dichiara di essere. Il server presenta quindi una stringa (**challenge**) e il client, tramite il secret, può fornire la prova di identificazione richiesta e riesce ad accedere (**response**).
Questo schema fornisce segretezza, tramite uso di password o chiavi, e anche freschezza, nella misura in cui la challenge viene modificata a ogni richiesta, così che non si possa sfruttare una risposta già fornita con un replay attack.

Capitolo 5

Attacchi TCP

5.1 ARP spoofing attack, quali sono le possibili conseguenze ed eventuali contromisure

Il protocollo ARP (Address Resolution Protocol) si occupa di connettere un indirizzo fisico (MAC address) con un indirizzo logico(IP address). Per farlo ogni nodo all'interno della rete invia dei messaggi in broadcast per segnalare la sua presenza, così che gli altri nodi possano andare a popolare e/o modificare una tabella interna chiamata *ARP cache table*
Viene fatta un'assunzione di trust nella LAN, dato che le richieste non vengono tracciate, gli annunci non sono autenticati, le macchine si fidano l'una dell'altra. In questo modo un possibile attaccante può utilizzare il meccanismo automatico di aggiornamento della ARP cache table (che si aggiorna anche se non ha inviato alcuna richiesta) per fingersi un altro dispositivo.
Conseguenze: L'attaccante si impone tra due dispositivi permettendogli di intercettare il traffico (sniffing), modificarlo (per esempio per iniettare un malware) o bloccarlo (DoS).
Contromisure: Alcune possibili contromisure possono essere:

- ARP statici: si configurano manualmente le associazione degli indirizzi IP e MAC, ma non è pratico in reti dinamiche o di grandi dimensioni
- Crittografia del traffico: anche se l'attaccante intercetta il traffico, non può leggerne il contenuto.
- Strumenti di monitoraggio

5.2 Attacco TCP SYN flood e le sue contromisure + SYN-Cookie

L'attacco TCP SYN Flood è una forma di attacco DoS (Denial of Service) che mira a sovraccaricare un server esaurendone le risorse disponibili per le con-

nessioni, impedendo così agli utenti legittimi di accedere ai servizi. L'idea è di sfruttare le vulnerabilità legate alla negoziazione di una comunicazione tra client e server durante il three-way handshake, l'attaccante **continua a iniviare richieste SYN al server** senza poi rispondere ai SYN/ACK ricevuti in risposta, il server dovrà quindi attendere andando a saturare il TCB (Trasmission Control Block), impedendo le connessioni legittime.
Contromisure: ampliare la memoria del TCB in modo che possa ricevere più richieste, ridurre i timer prima che una richiesta SYN venga cancellata dalla memoria o utilizzo dei SYN-Cookie.
SYN-Cookie: Tenica per contrastare l'attacco SYN Flood, l'utilizzo di cookie permette a una sessione di restare attiva anche se la coda SYN è stata saturata da un attacco. Quando viene attivata una sessione con un three-way handshake, il server risponde a al pacchetto SYN del client con un SYN-ACK più il valore del cookie. Quando il client risponde con un ACK, il server decodifica il cookie per verificare la validità della richiesta e, solo allora, alloca le risorse necessarie per stabilire la connessione.
Questo approccio consente al server di gestire un numero elevato di richieste SYN senza esaurire le risorse, poiché non mantiene stato per le connessioni incomplete.

5.3 TCP hijacking attack + codice

```
► Internet Protocol Version 4, Src: 10.0.2.69, Dst: 10.0.2.68
▼ Transmission Control Protocol, Src Port: 23, Dst Port: 45634 ...
    Source Port: 23
    Destination Port: 45634
    [TCP Segment Len: 24]                ← Data length
    Sequence number: 2737422009          ← Sequence #
    [Next sequence number: 2737422033]   ← Next sequence #
    Acknowledgment number: 718532383
    Header Length: 32 bytes
    Flags: 0x018 (PSH, ACK)
```

5.3.1 TCP hijacking attack ed esempio rispetto al codice

Il TCP hijacking è un attacco in cui un attaccante prende il controllo di una sessione TCP già avviata tra due host, senza che uno dei due se ne accorga. L'attaccante invia pacchetti spoofati, fingendosi uno degli endpoint, sfruttando il fatto che TCP si basa su numeri di sequenza e acknowledgment per mantenere la connessione affidabile. L'obiettivo dell'attacco è quello di iniettare dati o comandi nella sessione o interromperla/alterarla.
Nel nostro scenario ipotetico, l'attaccante intercetta un pacchetto proveniente dal server verso il client in una connessione Telnet. Anche se non conosce l'intero stato della sessione, dal pacchetto osservato può dedurre quale sarà il prossimo numero di sequenza atteso dal server. A quel punto, può costruire un pacchetto contraffatto, con IP e porta del client e valori di sequenza e acknowledgment

coerenti, inserendo nel payload un comando arbitrario da far eseguire alla shell remota. Poiché Telnet è un protocollo testuale non cifrato, il comando viene trasmesso come semplice testo ASCII e, se ben formato, sarà interpretato come input valido dal server.

5.3.2 Codice

Questo pacchetto è parte di una connessione Telnet attiva dal server verso il client. L'attaccante ha intercettato questo pacchetto e ora conosce i seguenti dati cruciali:

- IP e porte della connessione
- Numero di sequenza attuale del server (2737422009)
- Numero di acknowledgment del server, che indica quale byte si aspetta dal client (718532383).
- La dimensione del payload (24 byte) che è la differenza tra sequence number e next sequence number.

Qual è il pacchetto da da spedire per portare a termine l'attacco?

L'attaccante vuole fingere di essere il client che invia dati al server. Per fare ciò, costruisce un pacchetto spoofato con questi campi:

- **Src IP:** 10.0.2.68 (IP del client)
- **Dst IP:** 10.0.2.69 (IP del server)
- **Src Port:** 45634 (porta del client)
- **Dst Port:** 23 (porta standard per Telnet)
- **Sequence number:** 718532383 (numero di sequenza che il server si aspetta dal client)
- **Acknowledgment number:** 2737422033
- **Flags:** PSH,ACK
- **Payload:** ls\n (per esempio)

Note dell'autrice:

- per trovare Src e Dst IP letteralmente inverti i due indirizzi iniziali
- Src Port è quella che prima stava sotto la dicitura Destination Port
- il Sequence number è quello che prima si chimava acknowledgment number
- l'acknowledgment number è il nezt sequence number

- le flags le copiamo
- Importante excursus sul payload: è necessario se vogliamo far eseguire al server un comando, va bene qualsiasi comando Telnet io ho mesos questo a caso

Nel caso si voglia far eseguire un comando al server, come si può procedere?

La risposta a sto point è chiaramente il payload!!
Per far sì che il server esegua un comando (es. in una sessione Telnet), è necessario inserire nel pacchetto un payload contenente il comando, terminato da un carattere di newline (\n), come avverrebbe nella digitazione manuale da parte dell'utente, ricordandosi che il comando deve avere senso nel contesto della shell remota.

5.4 Approcci alla scansione + FTP bounce scan

La scansione all'interno di una rete viene eseguita per recuperare informazioni su determinati host o server, l'obiettivo principale è ottenere informazioni sulle porta utilizzate (TCP/UDP), ciò quali porte sono aperte e in ascolto su determinati nodi, oltre che determinare quale sistema operativo è presente e se esistono sistemi di filtraggio o firewall in una determinata rete.
Lo scanning può essere attivo o passivo:

- **Attivo:** si immette traffico nella rete per recuperare informazioni
- **Passivo:** si fa sniffing senza intervenire attivamente

Diversi approcci:

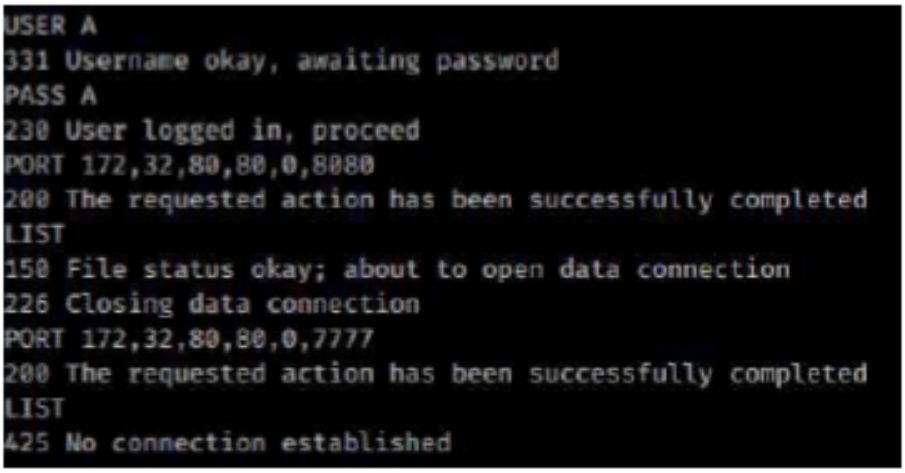
- **Verticale:** host che fa scanning di più target
- **Orizzontale:** molti host fanno scanning sullo stesso target
- **Ibrido:** mix tra i due

Infine il target può essere **singolo**, cioè una macchina, o **multiplo**, cioè anche una porzione di rete (se non tutta).

5.4.1 FTP bounce scan

L'attacco FTP bounce scan è una tecnica utilizzata per eseguire una scansione delle porte (port scanning) di un host di destinazione indirettamente tramite un server FTP, è una delle tecniche più note di port scanning attraverso terze parti.
L'attaccante invia al server FTP un comando PORT utilizzando l'indirizzo IP della vittima tramite un pacchetto spoofato. Se la porta del server è chiusa, quest'ultima risponderà con un pacchetto RST alla richiesta proveniente dal server FTP, mentre verrà eseguita una three-way handshake nel caso in cui la porta fosse aperta.

5.4.2 Commentare praticamente il risultato della seguente scansione



L'attaccante, utilizzando il comando port, riesce a capire che la porta 8080 è aperta (e quindi in ascolto), dato che viene indicato che è possibile trasferire file, mentre la porta 7777 non lo è, dato che non è stata stabilita nessuna connessione.

5.4.3 Due tecniche di scansione con TCP

TCP Connect Scan

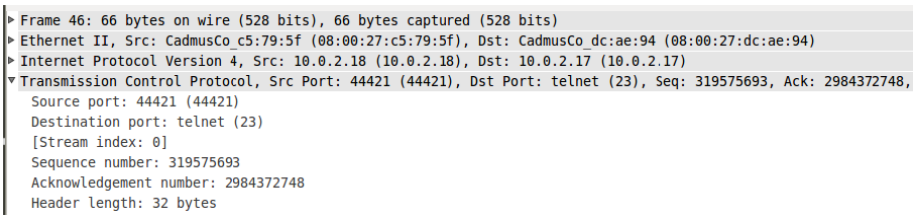
Questa tecnica tenta di stabilire una connessione TCP completa con la porta target inviando un pacchetto SYN, aspettando la risposta SYN-ACK e completando il handshake con un ACK.
Se la connessione si stabilisce, la porta è aperta. Se riceve un pacchetto RST (reset), la porta è chiusa.
E' semplice da implementare ma rumoroso.

SYN Scan (Half-open scan)

Invia un pacchetto SYN al target, ma non completa il handshake. Se riceve un SYN-ACK, invia un pacchetto RST per interrompere la connessione prima che venga completata.
Se riceve un SYN-ACK la porta è aperta, se riceve RST allora è chiusa.
Più furiva della connect scan poichè non comporta connessione, ma può essere rilevata da firewall

5.5 Reset Attack

Un TCP Reset Attack (attacco di reset TCP) è una tecnica usata per interrompere una connessione TCP esistente inviando un pacchetto TCP con il flag RST (Reset) attivo.



In una normale comunicazione TCP, il flag RST viene usato per indicare che qualcosa è andato storto e che la connessione deve essere chiusa immediatamente. Un attaccante, intercettando o osservando una connessione TCP, invia un pacchetto con flag RST a uno o entrambi i lati della connessione.

Capitolo 6

Attacchi

Capitolo 7

Scanning

Capitolo 8

Network Scanning

Capitolo 9

Firewall e NIDS