

Attacchi Denial of Services

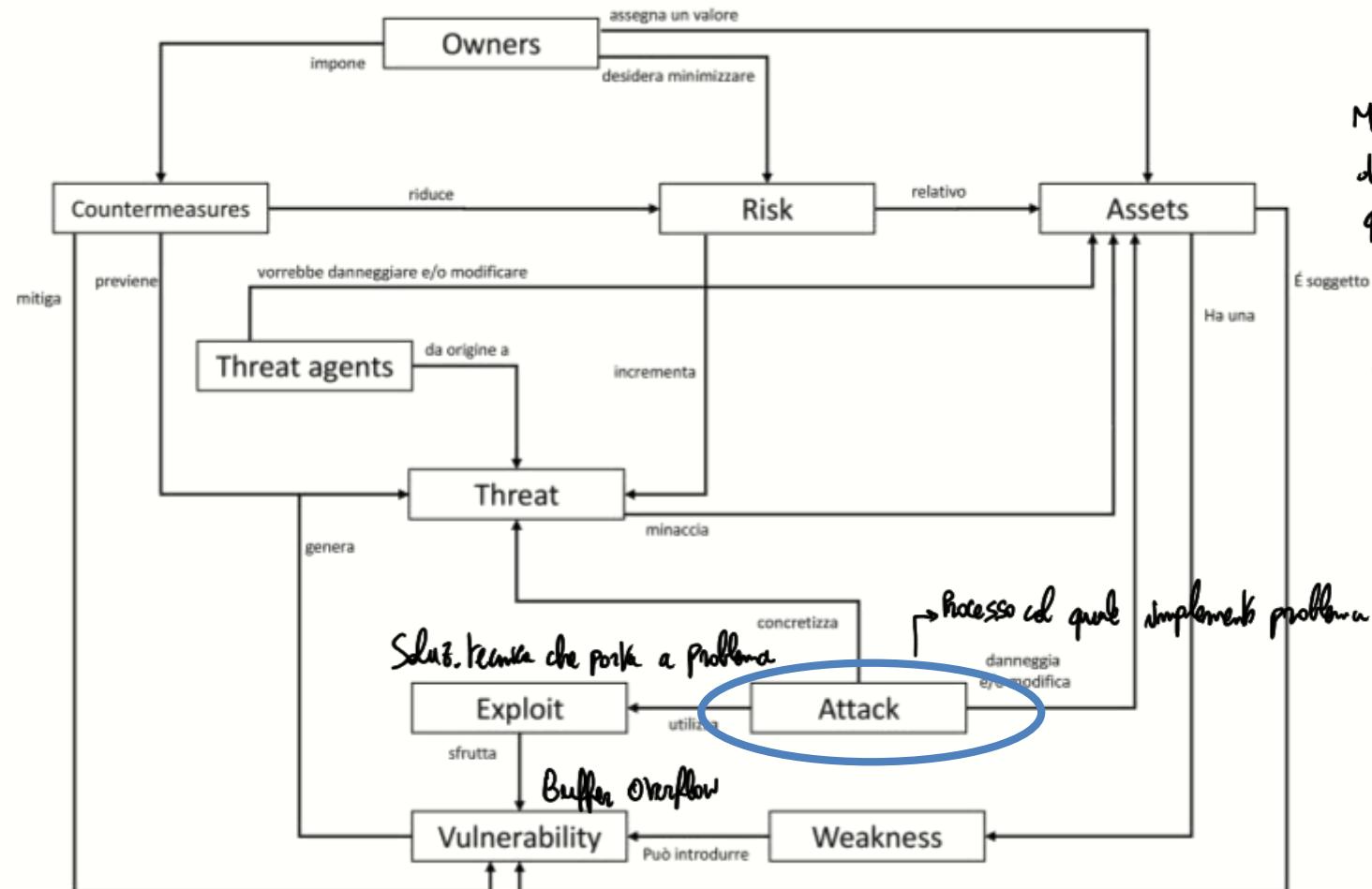
Cybersecurity

<date>
Location



Università
degli Studi
della Campania
Luigi Vanvitelli

Concetti Fondamentali



Minaccia: minaccia di effettuare
danno ad asset (prob. che
quelle cose avvengano).

Attacco: concretizza la minaccia.
Possibilità che avvenga all'attacco è
una minaccia.

Vulnerabilità: fornendo input scorretto posso produrre effetti anomali sul programma.

Exploit: pezzo di codice che fa sì che io sfrutta Vulnerabilità.

Attacco: processo che esegue exploit che sfrutta vulnerabilità.

Weakness di Seg.Fault: Debolezza strutturale.

Debolezza strutturale in un programma che non gestisce in modo
corretto la memoria è quella.

Weakness: debr. strutturali legate a come ho strutturato programma
che possono portare a vulnerabilità.

• Su C se uso malloc posso avere vulnerabilità su heap.

↳ Si attacca la vulnerabilità, la debolezza si testa.

Attacco informatico

Attacco: Un azione intrapresa contro un obiettivo con l'intenzione di compiere una azione non permessa dalla policy di sicurezza

Ad esempio, alterare, disabilitare, danneggiare o ottenere accesso e fare uso non autorizzato di un asset.

Asset (di nostro interesse) : Hardware, Software, Dati

DDoS: attacco eseguito attraverso collez. di sistemi

- **Attacchi DoS (Denial of Services):** Attacchi informatici che hanno come target sistemi e servizi e hanno come obiettivo il requisito di Disponibilità (Availability)
- **Disponibilità (Availability):** consiste nel garantire che una risorsa è disponibile agli individui autorizzati al momento appropriato

A. Albero che ogni device esegue

B. Come coordinare i vari device

C.

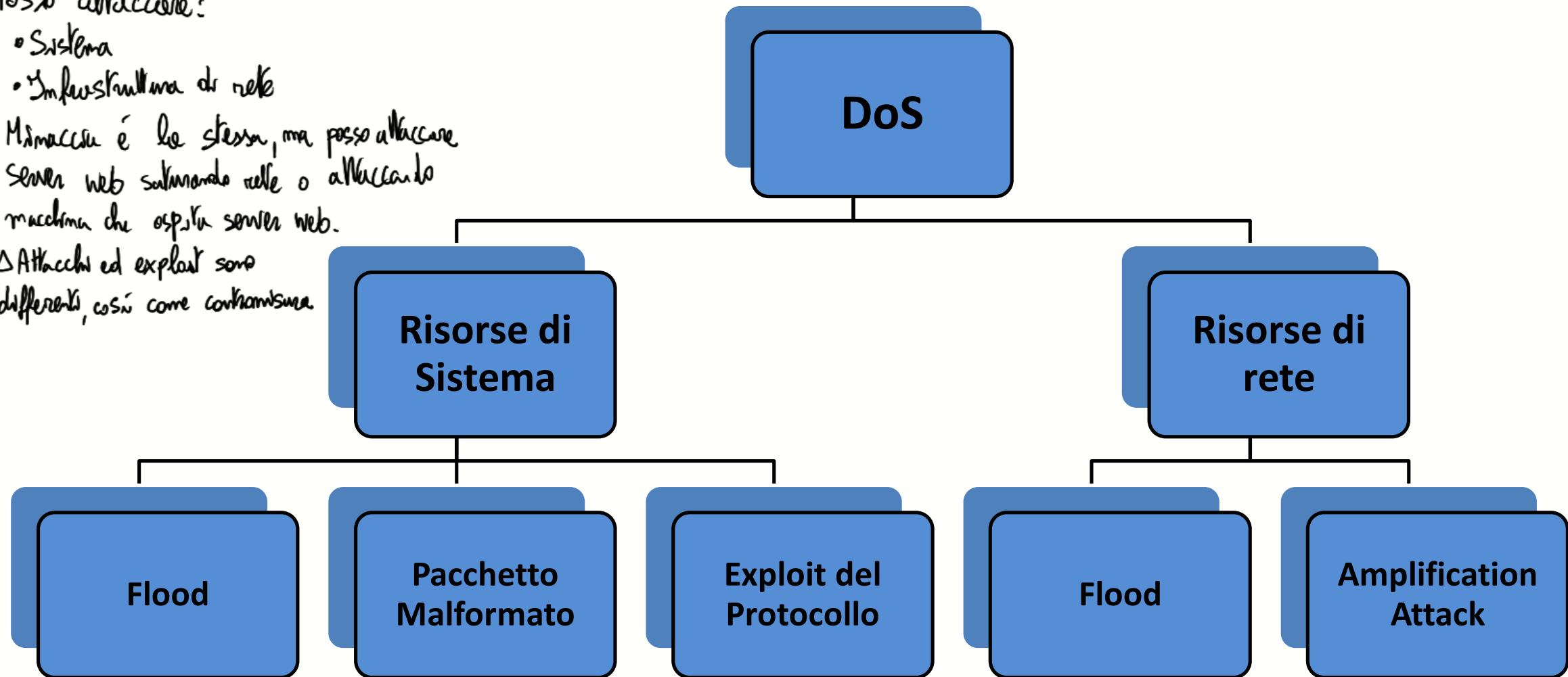
Tassonomia attacchi DoS (classificazione)

Posso attaccare:

- Sistema
- Infrastruttura di rete

Minaccia è la stessa, ma posso attaccare
server web salvandolo nelle o attaccando
macchina che ospita server web.

Δ Attacchi ed exploit sono
differenti, così come contrattacco.



Possibili strategie di difesa:

- Router identifica attacco e redirige traffico su altra sezione di rete con filtro molto più stringente. Riconoscimento tutto da rete.

Nota: Possibile attacco è anche inviare pacchetti grandiosi

1. Cosa: attacco.

2. Come:

2.1: flooding: si sommerge di richieste (segnali o rete) simili

| è un genere facilmente identificabile, ma le richieste sono banali ed è un problema distinguere da legit

2.2: dati malformati con una struttura diversa da quella che mi aspetto:

| es. molti particolari di costituzione pacchetti lessici, ma scorretti: XML con tag nel tag nel tag...

| questo nesting può saturare memoria della macchina. Attaccano in genere sistema. Rete vs. Transport.

2.3: vulnerabilità vere e proprie sui protocolli.

2.1.1: flooding: in generale

2.1.2: Amplification: attacchi fa poche richieste, ma per come è fatto il sistema genera un mare di messaggi in rete (es. http in modo circolare)

| In generale esaurita il system, ma anche rete malformata può causare problemi.

Tassonomia degli Attacchi

In rete posso attaccare banda o latenza. Sul sistema ho professoressa, memoria, disco ecc.

Che tipo di attacco? Qual è la risorsa?

Consumo di Risorse

- *Flood*: ricezione di un gran numero di richieste ed ogni richiesta è associata ad una risorsa (memoria, processi, porte, ...)
- *Pacchetto Malformato*: messaggi opportunamente malformati possono comportare utilizzo eccessivo di risorse
- *Exploit del protocollo*: sfruttare punti deboli specifici di un protocollo per provocarne occupazione eccessiva di risorse e/o un crash

Più particolari sono le risorse che subiscono più i differenti valori che vengono attaccate.

Consumo di Banda

- *Flood*: Generazione di enorme numero di messaggi che provocano l'occupazione della banda
- *Amplification Attack*: la quantità di dati generati dall'attaccante è amplificata da un passaggio intermedio e colpisce con molti più dati la/le vittima/e

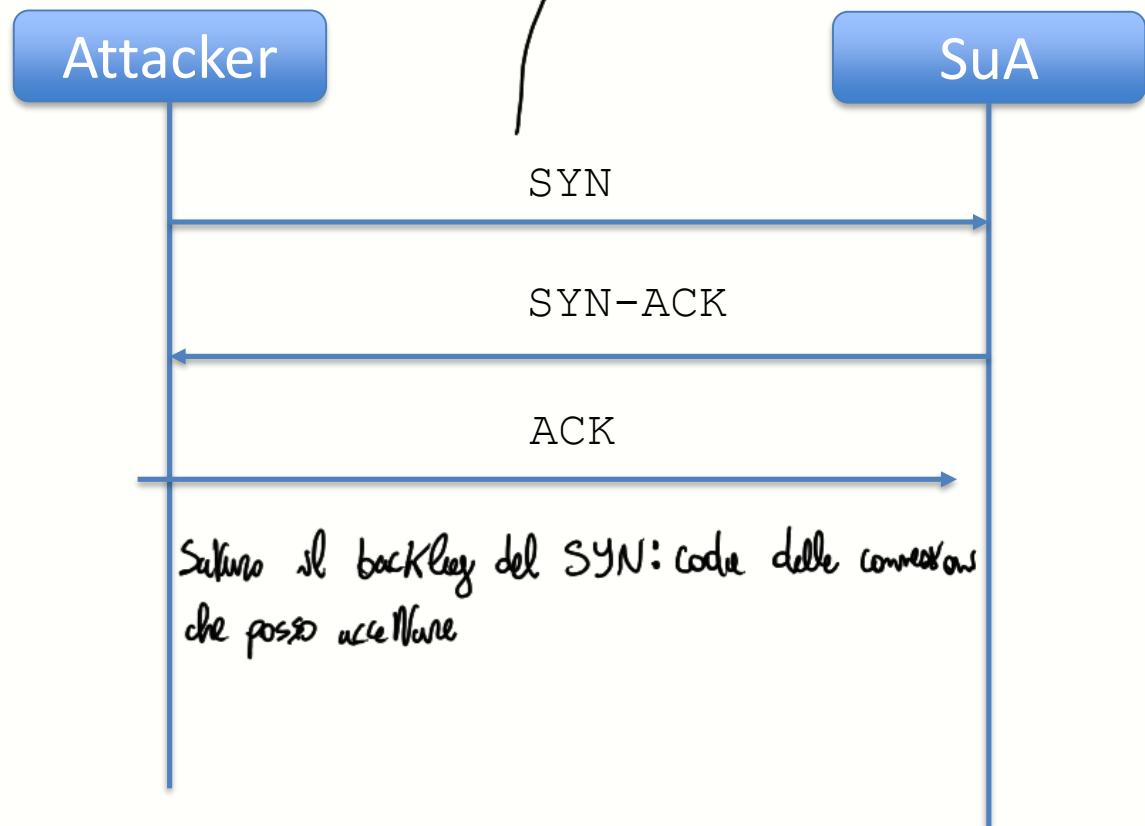
Esempi di Attacchi di Tipo Flood

Vulnerabilità: Invia e attesa del ACK
↳ Usata per creare attacco.

TCP Syn Flood

- Comportamento:
 - SYN
 - SYNACK
 - ACK
- Il Server occupa risorse per ogni richiesta SYN, in attesa del SYNACK..
- Il numero di processi e tentativi di connessione in attesa non può essere infinito...

TCP Handshake



Allocò buffer di memoria, blocca quel numero di porte ecc.

Soluzione 1: (DoS, non DDoS)

- Blacklisting dell'IP sorgente. Non risolve spoofing perché non ho autenticazione a liv. 3.
↳ Faccio synflood mirato; DoS mirato. \times

Concentriamoci sulla weakness: Three Way handshake: client manda messaggio-pacchetto. Server fa lavoro, manda messaggio-pacchetto, client fa lavoro e manda messaggio-pacchetto. Allacciando si utilizza lavoro:
lavora prima server del client. Soluzione, far lavorare prima il client.

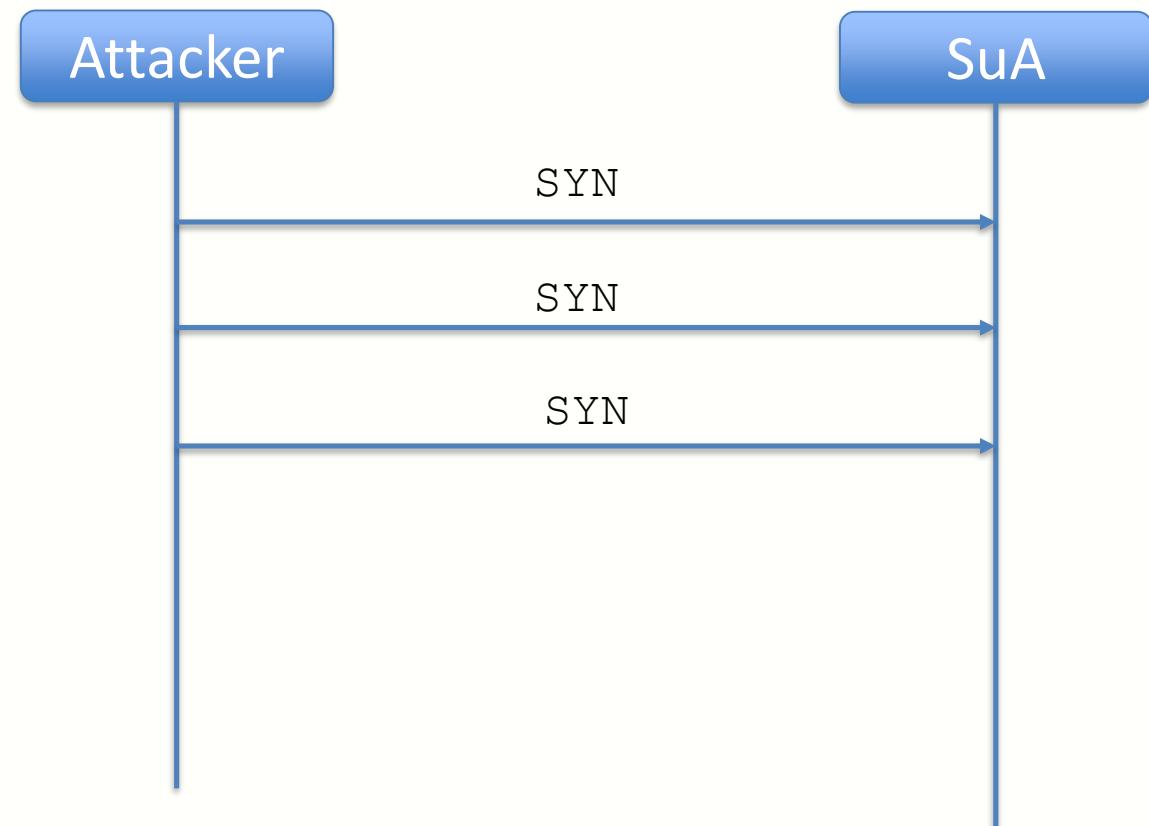
1. Attacco coda
2. Se arrivò a saltare, non crea shutdown dati ma segnala che deve finire e manda l'ack.

NOTA: Ilcaptcha funziona così: non ci vuole mandare a mandare per il server, ma per il browser ci mettiamo un po' a convincerla.

Esempi di Attacchi di Tipo Flood

TCP Syn Flood

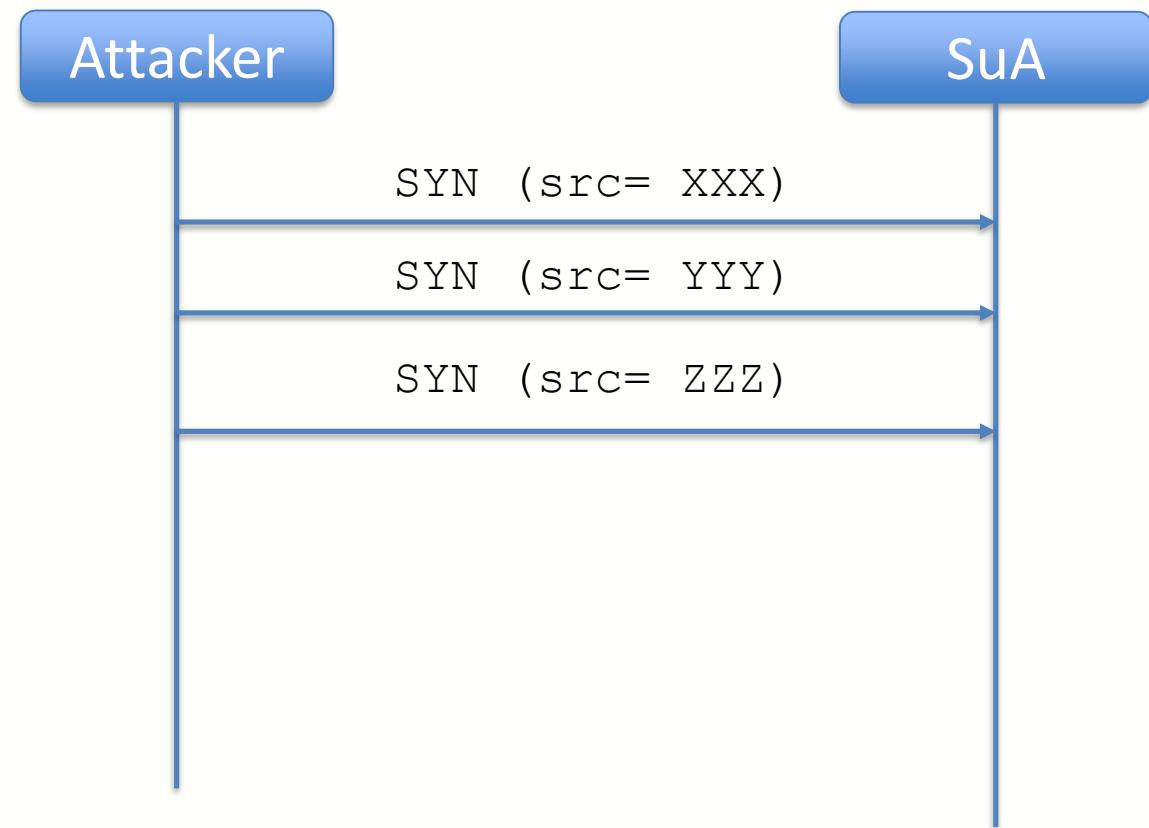
- Comportamento:
 - Spedizione di messaggi di SYN
 - Scartare senza risposta i messaggi SYNACK
- Il Server occupa risorse per ogni richiesta SYN, in attesa del SYNACK..
- Il numero di processi e tentativi di connessione in attesa non può essere infinito...



Esempi di Attacchi di Tipo SYN Flood+Spoofing

TCP Syn Flood *Misurazione syn flood oppure blacklists qualcuno*

- Comportamento:
 - Spedizione di messaggi di SYN
 - Nel pacchetto IP inserire un SRC fittizio
 - Scartare senza risposta i messaggi SYNACK
- Il Server occupa risorse per ogni richiesta SYN, in attesa del SYNACK..
- Il numero di processi e tentativi di connessione in attesa non può essere infinito...



Syn Flood: Un esempio in codice

- Esempi di attacchi SYN in Python:
 - <https://github.com/EmreOvunc/Python-SYN-Flood-Attack-Tool>
- Libreria di base: Scapy (<https://scapy.net>)
 - Packet crafting
 - Fornisce tutti i metodi base per creare message packets di rete in python
- Esempio Syn Flood: [Codice](#)
- Documento di Studio su SYN Flood e Contromisure: RFC 4987

Prendo struttura chi byte e la butto sulla rete. Ma come faccio? SO fa handshake in modo trasparente.
Devo parlare direttamente sulla scheda.

Cercati libreria Scapy: fu costruire packet ad hoc.

↳ IP_Packet è oggetto pacchetto IP. (Flag "S" è SYN) (Send (IP_P / TCP_P))

Comandi utili

- Netstat -s : Statistics sui protocolli statistica protocolli usati su macchina
- Netstat -atn : All, TCP, Not Resolution Viene mostrato solo
- Netstat -atn |grep SYN_RECV |wc -l conta i syn ricevuti

→ Tutte le configurazioni del Kernel di rete

- /proc/sys/net/ipv4/tcp_max_syn_backlog (max. max di connessioni attive)
- /proc/sys/net/ipv4/tcp_syncookies (abilita o disabilita syncookies)

/proc in UNIX è cartella associata al kernel di Linux

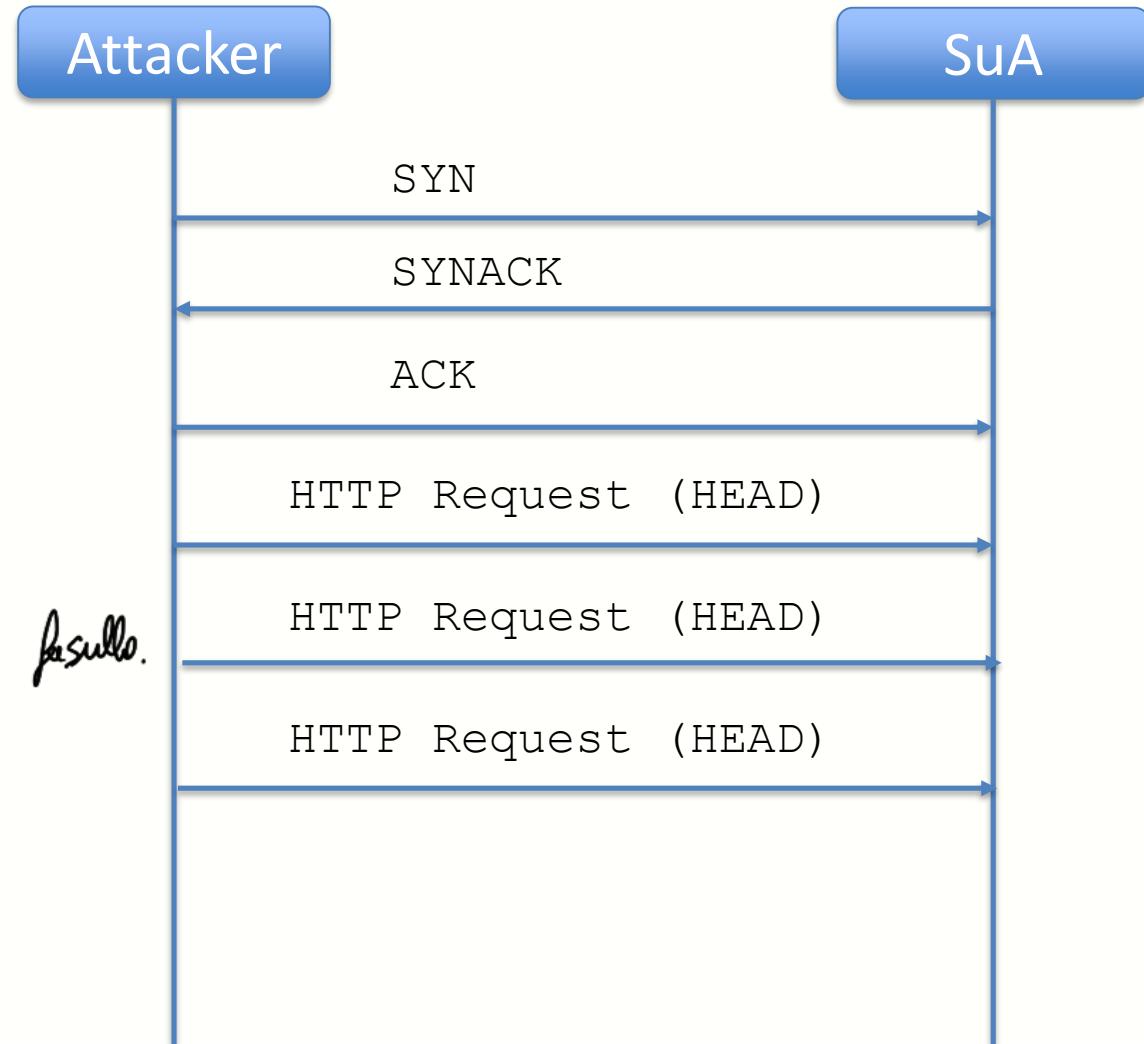
- Usa i syn cookie fino al quando arriva a oltre il 70% del backlog pieno.

Esempio di Flood a Livello Applicativo: HTTP Flood

- Risorse Consumate: tempo di CPU, Banda
- Comportamento:
 - Inoltro di richieste successive HTTP GET/HEAD ad alta velocità

Non posso fare spoofing, perché il SYN ha np address falso.
Ora la connessione mi serve.

Come mi metto nella fila se ho varie richieste di head?



Consumo bandiere oltre che risorse. Lui non occupa molto bandiera perché chiude un HEAD. Se invia una GET diventa più pericoloso.

Esempio di Flood a Livello Applicativo: HTTP Flood

- Risorse Consumate: tempo di CPU, Banda
- Comportamento:
 - Inoltro di richieste successive HTTP GET/HEAD ad alta velocità

Sembra su HTTP. No spoofing a livello IP. A nulla ha senso se attacca un server compromesso e il suo IP è corretto.
Qui blacklisting ha un senso diverso. Come minimo, macchina che attacca è quella compromessa.

```
HEAD / HTTP/1.1
Host: 192.168.1.11
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1
Accept: */*

HTTP/1.1 200 OK
Date: Mon, 14 Oct 2019 14:59:36 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 22 Sep 2019 14:39:28 GMT
ETag: "2c39-5932545469fb5"
Accept-Ranges: bytes
Content-Length: 11321
Vary: Accept-Encoding
Content-Type: text/html
```

HTTP Flood: Un esempio in codice

- Esempi di attacchi HTTP Flood in Python:
 - <https://github.com/D4Vinci/PyFlooder>
- Librerie utilizzate:
 - Argparse
 - socket
- Esempio HTTP Flood: [Codice](#)

FQDN: Formal Qualified domain name

NOTA: Configurando DNS, posso fare in modo che www.nome.it e nome.it siano associati a IP.

socket.gethostbyname controlla che i dati mappati su PC di default, poi se non lo fa, chiede per DNS.

NOTA: Se sniffando ho l'richiesta DNS \Rightarrow Riconosca. Se più di 1 richiesta,

NOTA: Uso thread per aprire più connessioni col server perché devo spammare.

- flush svuota buffer e forza una stampa sul terminal.
- Attack: apre socket su server remoto:
socket(socket.AF_INET, socket.SOCK_STREAM)
↳ creare una socket sull'Internet.
- dos.connect(ip, port) si connette con TCP automatico.

byt: classico request line HTTP.

Esempio di Abuso di pacchetti Malformati: SlowLoris

- Risorsa esaurita: le porte disponibili sul server
- Comportamento:
 - Apre n connessioni
 - In ogni connessione
 - Comincia a spedire una HTTP Request
 - Non conclude mai la richiesta



Caraicità di questo attacco: crea attacco nel quale non fanno nulla: low (low band) e slow (slow).

Faccia pochissime richieste: server ha rete vuota, mancanti buffer e processare buffer che risulta corrotto.

IDEA: Sprutta Weakness di TCP che dà alla vulnerabilità di HTTP.

Trasferimento dati TCP avviene per stream di byte: non sa quando messaggio finisce e finisce, vede solo flusso byte per un verso e viceversa. (client - server è previsto in TCP)

A livello applicativo il concetto CLIENT-SERVER si applica a livello superiore. Ma non è detto che client e server siano gli stessi.

TCP ragiona a livello di byte. Mando richiesta HTTP. Quanto è grande header HTTP? Quando c'è una linea vuota.

Ricavo tutto i dati fino a linea vuota. Header in pacchetto TCP come lo mando? Max 2^{16} bit come dimensione pacchetto. Al livello applicativo devo mettere nel pacchetto da mandare con TCP con frammentazione.

Differenze da implementazione a implementazione.

Mando richiesta ma nell'header non metto linea vuota di fine header. Poi aspetto un po' e mando un'altra parte. Potrebbe essere lecito se elaboro il pacchetto man mano che mando.

Dopo qualche attesa a mandare 1 byte di header ogni tant.

Server analizza e aspetta il prossimo messaggio. Server è solo un attesa

dei messaggi e ha thread dedicati ad ogni messaggio

Esempio di Abuso di pacchetti Malformati: SlowLoris

- Risorsa esaurita: le porte disponibili sul server
- Comportamento:
 - Apre n connessioni
 - In ogni connessione
 - Comincia a spedire una HTTP Request
 - Non conclude mai la richiesta
 - Se apro un altro thread posso gestire quella richiesta. Apache per esempio, apre thread e manda. Se lancio 1000 socket, io apro 1000 thread (dipende da come è configurato e dal sistema d^{operativo} per vedere max.m. di thread) Attacco salva o m. thread MAX gestibili o m. connection MAX apribile rim contemporanea. MEMORIA, BANDA, CPU libere. È difficile accorgersi di questo attacco.

```
GET / HTTP/1.1\r\n
Host: host\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322;
.NET CLR 2.0.50313; .NET CLR 3.0.4506.2152;
.NET CLR 3.5.30729; MSOffice 12)\r\n
Content-Length: 42\r\n
X-a: <stringa casuale>\r\n
X-a: <stringa casuale>\r\n
X-a: <stringa casuale>\r\n
X-a: <stringa casuale>\r\n
...
...
```

Ogni riga Header
fittizia in un pacchetto
TCP dedicato

SlowLoris: Un esempio in codice

- Esempi di attacchi SYN in Python:
 - <https://github.com/gkbrk/slowloris>
- Librerie utilizzate:
 - Argparse
 - socket
- Esempio SlowLoris: [Codice](#)

• ArgParser: libreria per passaggio di parametri, in modo che posso fare -help o passare un modo intelligibile ai parametri.

(many = "?", può essere un qualcosa possibilità dell'argomento).

Arrivo a riga 101 perché prima non ci serve niente.

- Sendlime: righe di slowflows

- 118: user-agent = usata per identificare browser che uso. Lui usa una lista di user-agent comuni per fare in modo che siano tutte diverse.

- 150: avvio socket, verifico se serve HTTPS, man mi interessano certificati.

- 162: mando lui prima richiesta -

Fino a 167 faccio tutto tutto correttamente, ma non vengo da mettere header. Per lui slowflows.

- 187 controlla se mettiamo le socket e se c'è le crea. Rimuoviamo socket se c'era problemi. Se tolgo socket per errore me crea un'altra.

NOTA: qui non uso Thread, perché è tutto molto più lento.

L'unico server potrei avere processo unico che gestisce le socket.

**COME PROVARE E VERIFICARE QUESTI
ATTACCHI?**

Creare un Laboratorio Virtuale di sicurezza

Ambiente in cui effettuare esercizi di Penetration testing:

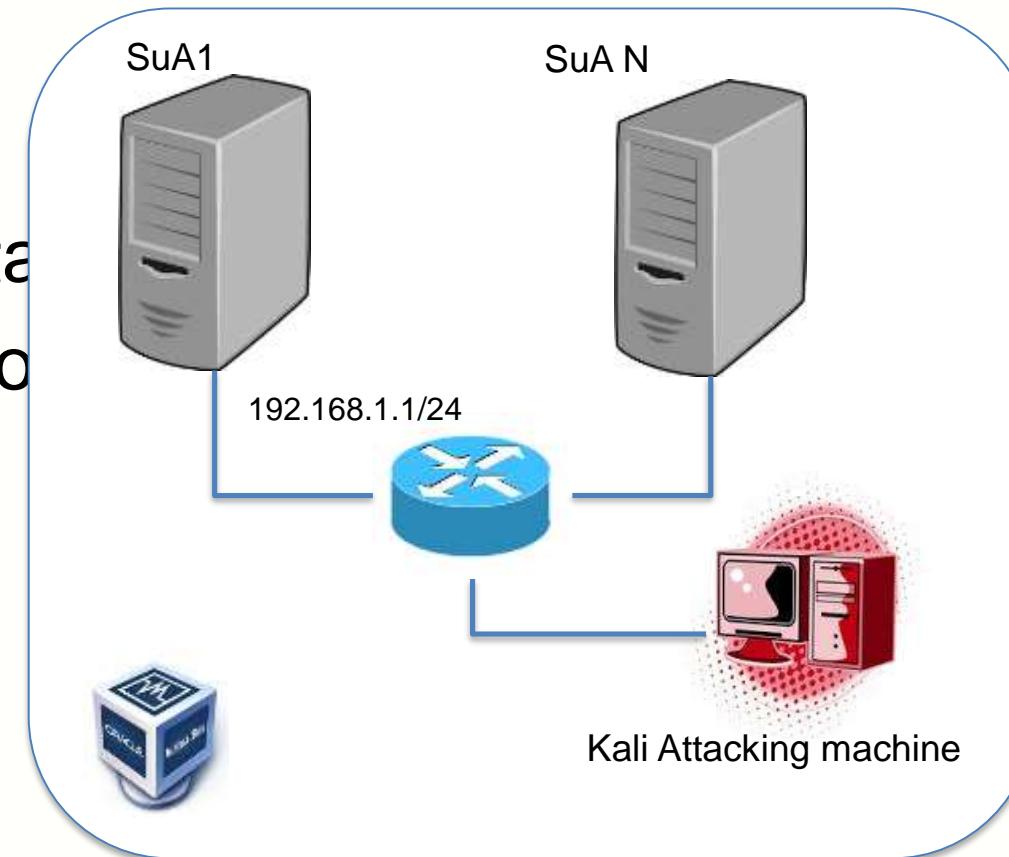
- **VirtualBox**: a creare Virtual Machines per realizzare l'esperimento
- **SuA** (System under Attack): Ubuntu 18.04 con Snort (IDS) per il rilevamento con Apache2
- **Attacker**: Kali Linux 2019.3 con tool DoS per effettuare testing



Parte Esercitativa: Ambiente di Prova

L'ambiente prevede

- Configurazione di rete isolata
- 1 o più Systemi sotto Attacco (SuA)
- 1 Macchina Attaccante



Realizzazione Ambiente test

- Installare Vbox e Guest Addition
- Configurare impostazioni di rete
 - Crea Rete con DHCP abilitato (vboxnet0)
- Creare Testbed
 - Creare VM SuA
 - Creare VM Kali
 - Verificare le impostazioni di Rete di tutte le macchine importate: rete solo host (vboxnet0)
- Far partire tutte le macchine dell'applicazione virtuale

Server Under Attack

- *Obiettivo:* Riprodurre un Sito Web
- Componenti fondamentali:
 - Configurare il Sistema Operativo (installare versioni server)
 - Installare Web Server (Configurazione Ideale, due server: Apache e NGNIX)
 - Installare un DB Server (mySQL, postgres)
 - Installare un Application Server (PHP o Servlet-based)
 - Realizzare una applicazione web minimale.
 - Solo Pagina HTML
 - Pagina HTML Dinamica con PHP/Servlet

Esercizio per casa

- Configurare il laboratorio virtuale di sicurezza
- Riprodurre i due attacchi
 - SynFlood
 - SlowLoris
- Intercettare con Wireshark gli scambi avvenuti
- Verificare che il server Web va in “Denial of Services”