SNORT 3 NIDS:

TUTORIAL COMPLETO DALL'INSTALLAZIONE ALL'ANALISI MALWARE

Guida pratica su Ubuntu 24.04, Regole Custom e PulledPork v3

Corso: Intelligent and Secure Networks 2025/26

Autore: Michele Tombesi

Data: Febbraio 2026

SOMMARIO:

Sezione 1: Configurazione del Laboratorio

Prima di procedere con l'installazione del software, è fondamentale predisporre correttamente l'ambiente virtuale. Per simulare uno scenario realistico di Network Intrusion Detection, non basta una singola macchina isolata: abbiamo bisogno di una rete che permetta il passaggio di traffico tra un "Attaccante" e una "Vittima/Sonda".

In questa sezione configureremo l'hypervisor (VirtualBox) e le macchine virtuali (VM) per creare un ambiente sicuro, isolato ma connesso a Internet per gli aggiornamenti.

---

1. Componenti del Laboratorio

Il nostro laboratorio si baserà su due macchine virtuali distinte:

1. **VM Snort (NIDS):** Sarà il cuore del nostro sistema. Monterà Ubuntu Server 24.04 LTS e ospiterà il motore Snort 3. Il suo compito è analizzare tutto il traffico che passa sulla rete interna.
2. **VM Client (Attaccante/Generatore di traffico):** Una seconda macchina (può essere un'altra Ubuntu, Kali Linux o anche una copia leggera della prima VM). Il suo compito è generare traffico (ping, richieste HTTP, scansioni) verso la VM Snort per testare le regole.

---

2. Topologia di Rete: La configurazione "Dual-Homed"

Per far funzionare il laboratorio, ogni VM dovrà avere **due schede di rete** distinte. Questa configurazione è essenziale per separare il traffico di gestione (aggiornamenti, SSH) dal traffico di analisi.

Ecco lo schema logico:

- **Scheda 1 (WAN/Management):** Connessa a Internet. Ci serve per scaricare i pacchetti (`apt`), clonare le repository (`git`) e aggiornare le regole (`PulledPork`).
- **Scheda 2 (LAN/Sniffing):** Connessa a una rete privata isolata. È qui che avverrà la simulazione degli attacchi. Snort ascolterà su questa interfaccia.

---

3. Configurazione passo-passo in VirtualBox

La seguente procedura deve essere applicata **sia alla VM Snort che alla VM Client**.

1. A macchina spenta, seleziona la VM in VirtualBox e clicca su **Impostazioni** > **Rete**.
2. **Configurazione Scheda 1 (Accesso Internet):**
    - Abilita scheda di rete: **Sì**.
    - Connessa a: **NAT** (Network Address Translation).
    - *Scopo:* Permette alla VM di "uscire" su Internet usando l'IP del tuo computer host.
3. **Configurazione Scheda 2 (Rete Interna):**
    - Spostati sul tab **Scheda 2**.
    - Abilita scheda di rete: **Sì**.
    - Connessa a: **Rete Interna** (Internal Network).
    - Nome: `intnet` (Assicurati che questo nome sia **identico** su entrambe le VM).
    - Avanzate > Modalità promiscua: **Permetti tutto** (Allow All).
        - *Nota:* Questo passaggio è critico per la VM Snort. La modalità promiscua permette alla scheda di leggere anche i pacchetti non destinati direttamente al suo MAC address, fondamentale per un NIDS.

---

4. Verifica della configurazione nella VM

Una volta avviate le macchine (Ubuntu Server), dobbiamo verificare che il sistema operativo veda entrambe le interfacce.

Esegui il comando:

```
ip a
```

Dovresti vedere tre voci principali:

1. `lo`: Loopback (127.0.0.1).
2. `enp0s3` (o simile): La scheda **NAT**. Dovrebbe avere un IP assegnato automaticamente (es. `10.0.2.15`).
3. `enp0s8` (o simile): La scheda **Internal**.
    - *Attenzione:* Di default su Ubuntu Server, questa seconda scheda potrebbe essere "DOWN" o senza IP. Per i test di comunicazione (Sezione 5), dovrai assegnare manualmente un IP statico a questa interfaccia (es. `192.168.10.1` per Snort e `192.168.10.2` per il Client) usando `netplan` o il comando `ip addr add`.

Esempio di assegnazione IP temporanea (per la Sezione 5):

```
# Sulla VM Snort
sudo ip addr add 192.168.10.1/24 dev enp0s8
sudo ip link set enp0s8 up
```

```
# Sulla VM Client
sudo ip addr add 192.168.10.2/24 dev enp0s8
sudo ip link set enp0s8 up
```

Ora che la rete è pronta, possiamo procedere con l'installazione del software.

Sezione 2: Installazione di Snort 3 su Ubuntu 24.04 (VM)

In questa prima sezione, getteremo le fondamenta del nostro progetto installando il motore **Snort 3**. Lavoreremo all'interno della macchina virtuale (VM) fornita per il corso "Intelligent and Secure Networks", basata su **Ubuntu Server 24.04 LTS**.

Per garantire il massimo controllo sulle funzionalità e la perfetta compatibilità con l'ambiente di laboratorio, non useremo i pacchetti pre-compilati standard, ma installeremo Snort **compilandolo direttamente dal codice sorgente**. Questo approccio ci assicura un setup robusto e riproducibile.

---

1. Verifica dell'Ambiente di Lavoro

Prima di lanciare qualsiasi comando, assicuriamoci di essere nel contesto giusto. Il progetto risiede in una cartella condivisa tra il sistema ospitante (Windows) e la macchina virtuale.

Riepilogo Configurazione:

- **Host:** Windows (il tuo PC fisico).
- **Guest:** Ubuntu Server 24.04 LTS (la VM VirtualBox).
- **Cartella di Progetto:** `/media/sf_ISNCODES/snort-nids-tutorial`

Nota Tecnica Importante: Sebbene i file del tutorial si trovino nella cartella condivisa, la compilazione vera e propria verrà eseguita dallo script in una cartella locale (`~/build`). Questa strategia è fondamentale per evitare i problemi di permessi e il drastico calo di prestazioni tipici della compilazione diretta su filesystem condivisi.

---

2. Lo Script di Installazione Automatica

L'installazione manuale di Snort 3 richiede decine di comandi e la gestione di molte dipendenze. Per semplificare e standardizzare il processo, utilizzeremo uno script dedicato: `scripts/install-snort3.sh`.

Cosa fa questo script?

1. **Prepara il sistema:** Aggiorna i repository APT e installa i tool di base (`gcc`, `make`, `cmake`).
2. **Installa le Dipendenze:** Scarica le librerie essenziali come `libpcap` (per catturare il traffico), `libpcre2` (per il pattern matching) e `libdaq` (Data Acquisition Library, il cuore dell'interfaccia di rete di Snort).

3. **Compila e Installa Snort:** Scarica il codice sorgente di Snort 3, lo compila e lo installa in `/usr/local`.

---

3. Esecuzione dell'Installazione

Procediamo ora con l'installazione effettiva. Dobbiamo posizionarci nella cartella del progetto, rendere eseguibile lo script e lanciarlo.

Comandi da eseguire:

```
# 1. Entra nella cartella del progetto (se non ci sei già)
cd /media/sf_ISNCODES/snort-nids-tutorial

# 2. Rendi lo script eseguibile
chmod +x scripts/install-snort3.sh

# 3. Avvia l'installazione
./scripts/install-snort3.sh
```

Cosa aspettarsi: Lo script impiegherà diversi minuti per completare l'operazione (a seconda delle risorse assegnate alla VM). Vedrai scorrere molte righe di testo relative al download e alla compilazione del codice C++.

---

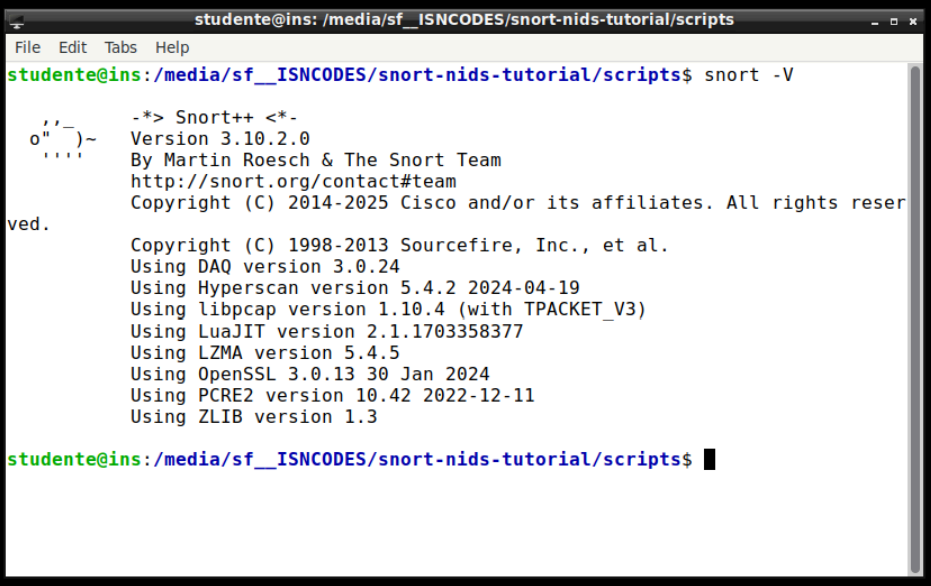
4. Verifica del Funzionamento

Al termine dello script, è fondamentale verificare che Snort sia stato installato correttamente e che il sistema riconosca il comando.

Comando di verifica:

```
snort -V
```

Se l'installazione è andata a buon fine, vedrai un output grafico (ASCII art) con il logo di Snort o, più semplicemente, le informazioni sulla versione (es. `Snort++ 3.x.x`).



---

5. Risoluzione Problemi Comuni (Troubleshooting)

Durante l'installazione potresti incontrare degli errori legati alla gestione dei pacchetti di Ubuntu. Ecco il più frequente e come risolverlo.

- **Errore "404 Not Found" durante apt update/install:**
  - *Causa:* Le liste dei pacchetti nella VM sono obsolete o corrotte, oppure i mirror di Ubuntu stanno rispondendo male. Questo impedisce allo script di scaricare le dipendenze necessarie (come `g++` o `libpcap`).
  - *Soluzione:* Esegui questi comandi per pulire la cache di apt e forzare un aggiornamento completo delle liste, poi rilancia lo script di installazione.

```
# Pulisce la cache dei pacchetti
sudo apt clean

# Rimuove le liste corrotte o vecchie
sudo rm -rf /var/lib/apt/lists/*

# Scarica le liste aggiornate
sudo apt update

# Riprova l'installazione
./scripts/install-snort3.sh
```

Sezione 3: Snort 3 - Architettura NIDS vs IPS

Dopo aver installato il motore Snort, è fondamentale comprenderne le modalità operative prima di iniziare l'analisi del traffico. Snort non è un software monolitico: il suo comportamento cambia drasticamente in base a come viene posizionato nella rete e a come viene configurato.

In questa sezione analizzeremo la differenza critica tra **NIDS** (Rilevamento) e **IPS** (Prevenzione), spiegheremo la pipeline di analisi dei pacchetti e definiremo il perimetro operativo del nostro laboratorio.

---

1. NIDS vs IPS: Due Filosofie Opposte

La distinzione più importante da fare è tra "osservare" e "intervenire". Snort può fare entrambe le cose, ma non contemporaneamente sullo stesso flusso dati senza cambiare configurazione.

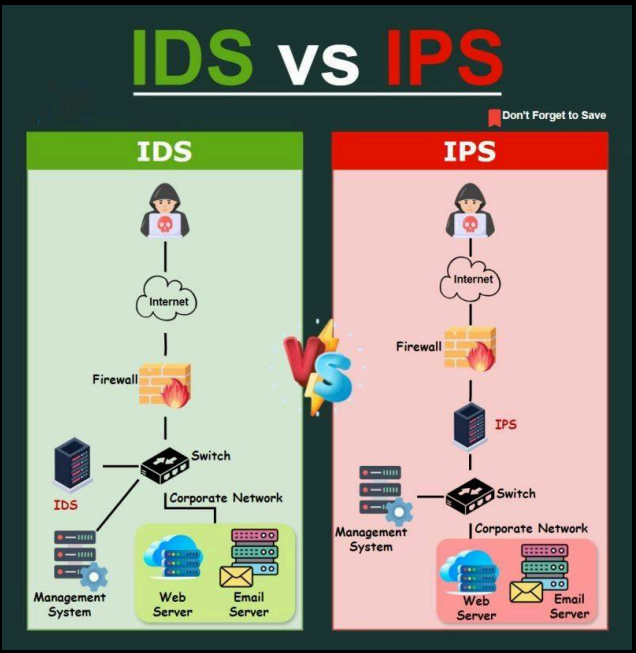
1.1 NIDS (Network Intrusion Detection System)

È la modalità che utilizzeremo in questo tutorial.

- **Ruolo:** Osservatore Passivo (come una telecamera di sorveglianza o un allarme antifurto).
- **Posizione:** Riceve una *copia* del traffico di rete (tramite port mirroring o leggendo un file PCAP).
- **Azione:** Analizza i pacchetti, cerca corrispondenze con le regole e **genera alert**.
- **Vincolo: Non può bloccare** il traffico. Anche se rileva un attacco gravissimo, il pacchetto malevolo raggiunge comunque la destinazione. Il NIDS serve a *sapere* che l'attacco è avvenuto.

1.2 IPS (Intrusion Prevention System)

- **Ruolo:** Filtro Attivo (come una guardia di sicurezza all'ingresso).
- **Posizione:** È posizionato "Inline" (in linea), ovvero il cavo di rete entra in Snort ed esce da Snort.
- **Azione:** Analizza i pacchetti e, se una regola lo prevede, può **scartarli (drop)**, bloccarli o modificarli prima che arrivino a destinazione.
- **Vincolo:** Un errore di configurazione può bloccare traffico legittimo (falso positivo), causando disservizi.

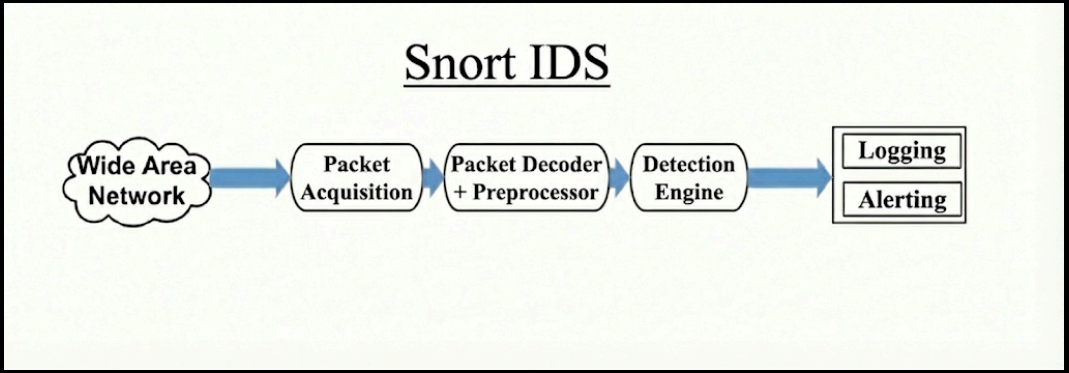


---

2. La Pipeline di Analisi: Come nasce un Alert

Quando usiamo Snort come NIDS, ogni singolo pacchetto attraversa una "catena di montaggio" rigorosa. Capire questo processo è essenziale per il debugging (ad esempio, per capire perché una regola non scatta).

1. **Packet Acquisition (DAQ):** Il modulo *Data Acquisition* preleva i pacchetti grezzi dalla scheda di rete (o dal file PCAP).
2. **Decoding:** Snort decodifica i livelli della pila ISO/OSI (Ethernet -> IP -> TCP/UDP). Qui capisce "chi parla con chi".
3. **Pre-processing / Normalization:** Il traffico viene "pulito". Ad esempio, se un attaccante prova a nascondere un attacco frammentando i pacchetti o usando codifiche URL strane (`%20` invece di spazio), il pre-processore ricostruisce il flusso originale per renderlo leggibile.
4. **Detection Engine:** È il cuore del sistema. Il traffico normalizzato viene confrontato con le **Regole**.
5. **Output / Alerting:** Se il motore trova una corrispondenza (Match), l'evento viene inviato ai plugin di output.



---

3. Modalità Operative: Live vs Forensics

Nel nostro laboratorio, e in generale nell'uso di Snort, possiamo operare su due domini temporali diversi.

3.1 Modalità Offline (Modalità Forense)

Questa è la modalità che useremo per analizzare i malware **Emotet** e **SQL Injection**.

- **Input:** Un file `.pcap` (Packet Capture).
- **Vantaggio:** Riproducibile al 100%. Possiamo analizzare un attacco avvenuto anni fa' senza rischi per la rete attuale.
- **Comando tipico:** `snort -r file.pcap ...`

3.2 Modalità Live (Tempo Reale)

- **Input:** Un'interfaccia di rete fisica (es. `eth0`, `enp0s3`).
- **Vantaggio:** Protezione (o rilevamento) istantaneo di ciò che accade ora.
- **Comando tipico:** `snort -i eth0 ...`

Nota tecnica: In modalità Live, Snort deve essere molto veloce. In modalità Offline, può prendersi tutto il tempo necessario per analizzare file enormi.

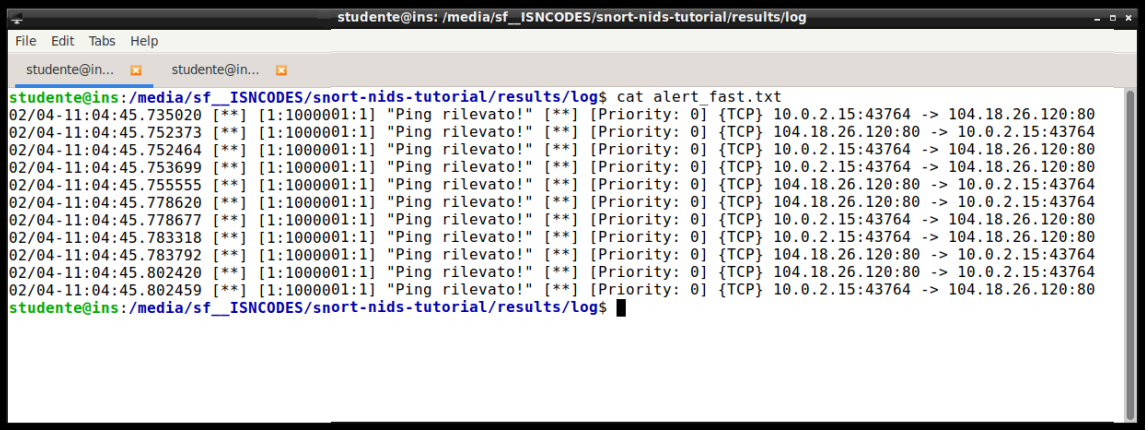
---

4. L'Output: Interpretare i Dati

In modalità NIDS, il prodotto finale del lavoro di Snort non è un pacchetto bloccato, ma un **Log**. Nel nostro progetto abbiamo configurato l'output `alert_fast`, che produce file di testo semplici e leggibili (`alert_fast.txt` o output a console).

Un tipico alert NIDS contiene la "5-Tuple" (i 5 elementi fondamentali di una connessione) più il messaggio della regola:

`[Tempo] [ID Regola] "MESSAGGIO" [IP Sorgente]:[Porta] -> [IP Dest]:[Porta]`



---

5. Limiti e Obiettivi Didattici

Per concludere questa panoramica teorica, è importante settare le aspettative corrette per l'esame:

1. **Visibilità, non Protezione:** Configurando Snort come NIDS, il nostro obiettivo è ottenere **visibilità**. Vogliamo vedere l'invisibile (es. il malware che comunica), non necessariamente fermarlo (che richiederebbe un firewall o un IPS).
2. **Falsi Positivi/Negativi:** Un NIDS è sensibile alle regole. Se una regola è scritta male, può generare falsi allarmi. Se manca una firma, il NIDS rimane muto (Falso Negativo).
3. **Il Valore dello Studio:** Usare la modalità NIDS su file PCAP ci permette di "congelare il tempo" e studiare l'anatomia di un attacco passo dopo passo, un'attività impossibile durante un attacco reale che dura pochi millisecondi.

Sezione 4: Regole Snort

In questa lezione ci focalizzeremo sul cuore pulsante del motore di detection: le **regole**. Aver installato Snort e averlo configurato come NIDS è inutile se non gli forniamo le istruzioni su cosa cercare.

Le regole sono, letteralmente, le istruzioni condizionali che dicono a Snort: *"Se vedi questo specifico pacchetto, allora scrivi questo alert"*. Prima di passare alla scrittura pratica e all'analisi del traffico, è essenziale comprendere la sintassi, perché un solo carattere sbagliato può rendere il sistema cieco o inondarlo di falsi allarmi.

---

1. Cosa sono e a cosa servono

Una regola Snort è una **descrizione formale di un evento di rete**. Immagina la regola come un "setaccio": tutto il traffico passa attraverso il motore, ma solo i pacchetti che corrispondono esattamente alla descrizione della regola rimangono nel setaccio e generano un evento.

L'obiettivo delle regole è trasformare il traffico grezzo (bit e byte) in informazioni di sicurezza leggibili, permettendo di:

- Individuare firme di attacchi noti (es. Emotet, SQL Injection).
- Rilevare tentativi di scansione (Probing).
- Monitorare violazioni di policy (es. uso di protocolli non autorizzati).

---

2. La Struttura Generale

Ogni regola Snort è divisa logicamente in due parti principali:

1. **Rule Header (Intestazione):** Definisce "Chi" parla con "Chi" e "Come" (IP, Porte, Protocolli). È il filtro iniziale.
2. **Rule Options (Opzioni):** Definisce "Cosa" cercare dentro il pacchetto (Payload, pattern, messaggi). È il cuore dell'analisi.

La sintassi generica è sempre questa:

```
#  HEADER                                                OPTIONS
#  |-----------------------------------------------|  |------------|
   azione protocollo sorgente porta -> destinazione porta ( opzioni; )
```

---

3. L'Header della Regola (Il Filtro)

L'header è la parte fissa che determina se il motore deve procedere ad analizzare il contenuto del pacchetto o scartarlo subito. È costituito da campi obbligatori:

- **Action (Azione):** Cosa fare se la regola viene soddisfatta?
  - `alert`: Genera un allarme e logga il pacchetto (la più usata in NIDS).
  - `log`: Logga il pacchetto senza generare allarme.
  - `pass`: Ignora il pacchetto (utile per whitelist).
- **Protocol (Protocollo):** Quale protocollo di trasporto analizzare?
  - `tcp`, `udp`, `icmp`, `ip`.
- **Source IP & Port:** Chi invia il pacchetto?
  - Possiamo usare indirizzi IP specifici (`192.168.1.5`), reti CIDR (`192.168.1.0/24`), variabili (`$HOME_NET`, `any`) o negazioni (`!$HOME_NET`).
- **Direction Operator:** La direzione del traffico.
  - `->`: Unidirezionale (dalla sorgente alla destinazione).
  - `<>`: Bidirezionale (utile per analizzare scambi completi).
- **Destination IP & Port:** Chi riceve il pacchetto?

Esempio di Header: `alert tcp any any -> 192.168.1.0/24 80` *(Traduzione: Allerta se vedi traffico TCP da chiunque verso la mia rete sulla porta 80)*

---

4. Le Opzioni della Regola (Il Payload)

Le opzioni sono racchiuse tra parentesi tonde `( )` e separate da un punto e virgola `;`. Qui definiamo i dettagli del match e le informazioni per l'analista.

Le opzioni principali che useremo sono:

- **msg:** Il messaggio di testo che apparirà nel log (`alert_fast.txt`). Deve essere chiaro e parlante.
  - Es: `msg:"Possibile attacco SQL Injection";`
- **content:** Il comando più importante. Cerca una stringa di testo o una sequenza esadecimale (byte) dentro il payload del pacchetto.
  - Es: `content:"GET";`
- **nocase:** Istruisce Snort a ignorare maiuscole/minuscole durante la ricerca del `content`.
  - Es: `content:"admin"; nocase;` (troverà "Admin", "ADMIN", "admin").
- **sid (Snort ID):** L'identificativo univoco della regola.
  - `< 1000000`: Riservati alle regole ufficiali (VRT/Talos).
  - `> 1000000`: Spazio riservato alle **Local Rules** (quelle che scriveremo noi).
- **rev:** Il numero di revisione. Si incrementa ogni volta che modifichi la regola.
  - Es: `rev:1;`

---

5. Esempio Pratico: Anatomia di una Regola

Passiamo alla pratica analizzando la regola per identificare un tentativo di connessione TCP sulla porta 12345.

La regola:

```
alert tcp any any -> 192.168.10.1/24 12345 (msg:"[LAB] TCP connect verso 12345 (netcat)"; flags:S; sid:1000004; rev:1;)
```

Analisi passo-passo:

- **`alert tcp`**: Snort si prepara a generare un allarme analizzando il traffico basato sul protocollo TCP.
- **`any any -> 192.168.10.1/24 12345`**: Definisce la direzione del traffico:
  - **Sorgente:** Qualsiasi indirizzo IP (`any`), da qualsiasi porta (`any`).
  - **Destinazione:** La sottorete interna specifica (`192.168.10.1/24`) sulla porta specifica **12345**.
  - *Contesto:* La porta 12345 non è standard; qui viene usata per un laboratorio (lab) per testare una backdoor o un listener Netcat.
- **`( ... )`**: Inizia l'ispezione delle opzioni della regola.
- **`msg:"[LAB] ...";`**: Se la regola scatta, scrive nel file di log questo messaggio esatto, utile per capire subito che si tratta di un test ("LAB") e non di un attacco reale.
- **`flags:S;`**: **Questa è la parte critica.** Snort controlla i flag dell'header TCP. La "S" sta per **SYN**.
  - Significa che Snort non sta guardando tutto il traffico dati, ma cerca specificamente il **tentativo di inizio connessione** (il primo pacchetto del *Three-Way Handshake*). Se qualcuno prova ad "aprire" la porta 12345, l'allarme scatta.
- **`sid:1000004; rev:1;`**:
  - `sid`: L'identificativo univoco della regola (Snort ID).
  - `rev`: La revisione della regola (versione 1).

5.1 Nota sull'uso di "flag:S"

Senza l'opzione `flags:S`, Snort genererebbe un allarme per *qualsiasi* pacchetto passante per la porta 12345 (dati, chiusura connessione, ack, ecc.), inondando i log di rumore inutile (falsi positivi). Usando `flags:S`, stai dicendo a Snort: *"Avvisami solo quando qualcuno bussa alla porta per entrare, non quando stanno già parlando."* È una tecnica fondamentale per rilevare scansioni di porte (port scanning) o tentativi di connessione a backdoor.

Sezione 5: Primo Test Operativo - Analisi Offline e Custom Rules

In questa sezione effettueremo il "battesimo del fuoco" della nostra installazione di Snort 3. Invece di collegarci subito a un'interfaccia di rete reale (con tutto il rumore di fondo che ne consegue), analizzeremo una cattura di traffico pre-registrata: un file **PCAP**.

Questo approccio "Offline" (o modalità Forense/Playback) è la prassi standard per validare nuove configurazioni. Ci permette di verificare tre aspetti critici in un ambiente controllato:

1. Il motore di detection si avvia correttamente.
2. La sintassi della nostra regola personalizzata è valida.
3. Il file di configurazione (`snort.lua`) punta correttamente al file delle regole.

---

1. Ambiente e Requisiti

Per questo test utilizzeremo file e percorsi specifici già predisposti nell'ambiente del corso. Assicurati di operare con questi riferimenti:

- **Directory di lavoro:** `/media/sf_ISNCODES/snort-nids-tutorial`
- **File di configurazione:** `snort/snort.lua`
- **File delle regole:** `snort/rules/local.rules`
- **Traffico di test:** `pcaps/test-http.pcap` (Un file contenente semplice traffico web).

---

2. Creazione della Regola "Hello World"

Le regole di Snort non perdonano errori di sintassi. Un errore comune è spezzare la regola su più righe o dimenticare il punto e virgola finale. Per questo primo test, scriveremo una regola "generica" che deve scattare su **qualsiasi pacchetto IP**. Se questa regola non scatta, significa che Snort non sta leggendo il file.

1. Crea (o apri) il file delle regole locali:

```
nano snort/rules/local.rules
```

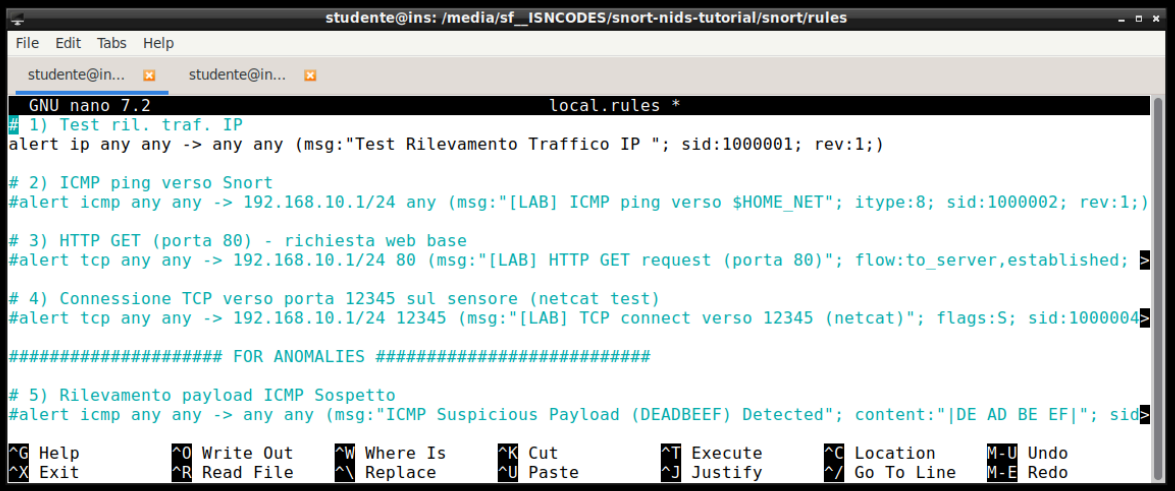
2. Inserisci questa singola riga (tutto su una riga):

```
alert ip any any -> any any (msg:"TEST Rilevamento Traffico IP";
sid:1000001; rev:1;)
```

Analisi della regola:

- **Header:** `alert ip any any -> any any`
  - *Traduzione:* "Genera un allarme su protocollo IP, da chiunque verso chiunque".
- **Options:** `(msg:"..."; sid:1000001; ...)`
  - *Traduzione:* "Scrivi il messaggio 'TEST...', usa l'ID 1000001".

3. Salva ed esci (`CTRL+O`, `Invio`, `CTRL+X`).



---

3. Collegare la Regola a Snort (snort.lua)

Ora dobbiamo dire a Snort dove trovare la nostra regola. Snort 3 utilizza **Lua** per la configurazione, il che cambia drasticamente il modo di includere i file rispetto a Snort 2.

Attenzione all'Errore Comune: Non usare il comando `include 'local.rules'` a livello globale nel file. Questo genererebbe un errore di sintassi Lua (spesso `unexpected symbol near 'ip'`), perché Snort cercherebbe di eseguire il file delle regole come se fosse uno script.

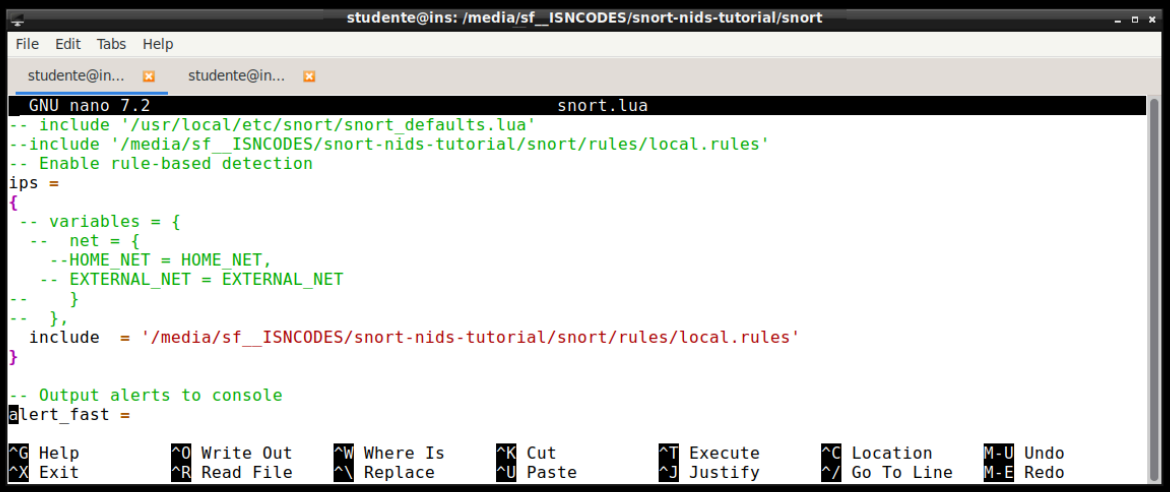
La Configurazione Corretta: L'inclusione deve avvenire all'interno del modulo **IPS**.

1. Apri il file di configurazione:

```
nano snort/snort.lua
```

2. Cerca (o crea) il blocco `ips` e configuralo così, usando il percorso assoluto per evitare ambiguità:

```
ips =
{
    -- Usa 'include' qui dentro per caricare regole testuali standard
    include =
'/media/sf_ISNCODES/snort-nids-tutorial/snort/rules/local.rules'
}
```



---

4. Esecuzione del Test

Ora che tutto è pronto, lanciamo Snort contro il file PCAP.

4.1 Il Problema dei Checksum

I file PCAP catturati da schede di rete reali spesso hanno checksum TCP/UDP matematicamente errati (a causa del "Checksum Offloading" hardware). Di default, Snort scarta questi pacchetti considerandoli corrotti. Per analizzare i PCAP, dobbiamo **obbligatoriamente** dire a Snort di ignorare i checksum errati con il flag `-k none`.

Comando di Esecuzione: Dalla root del progetto (`/media/sf_ISNCODES/snort-nids-tutorial`), esegui:

```
sudo snort -c snort/snort.lua -r pcaps/test-http.pcap -k none -A
alert_fast
```

Spiegazione dei Flag:

- `-c snort/snort.lua`: Usa questo specifico file di configurazione.
- `-r pcaps/test-http.pcap`: Leggi il traffico da questo file (Modalità Replay).
- `-k none`: **Fondamentale.** Ignora i checksum errati (non scartare i pacchetti).
- `-A alert_fast`: Stampa gli alert direttamente in console (stdout).

---

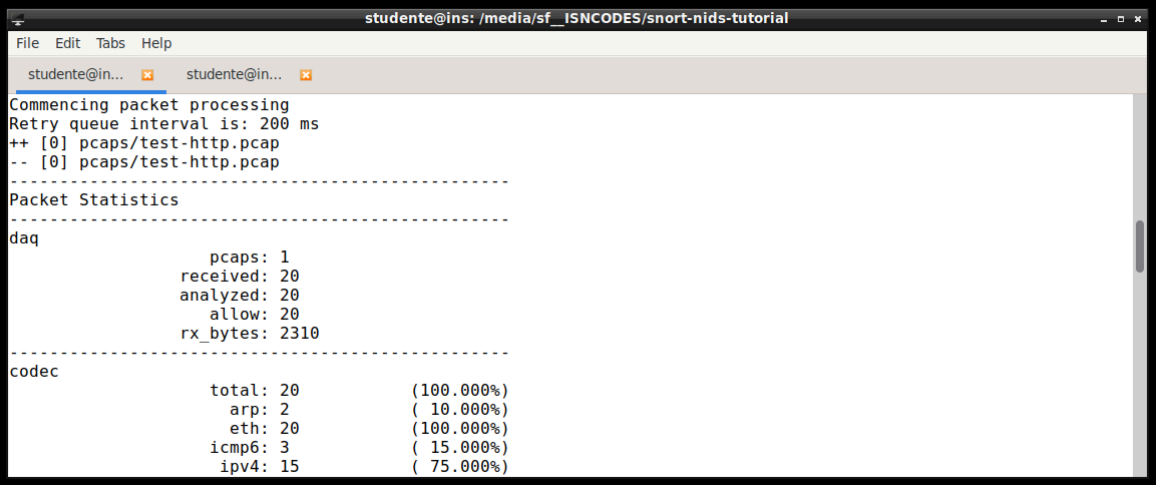
5. Analisi dell'Output

Se la configurazione è corretta, vedrai il file `alert_fast.txt` popolarsi di righe simili a questa:

```
11/14-11:04:45.783318 [**] [1:1000001:1] "TEST Rilevamento Traffico IP"
[**] [Priority: 0] {TCP} 10.0.2.15:43764 -> 104.18.26.120:80
```

Alla fine, Snort mostrerà le statistiche riassuntive. Verifica che:

- **Analyzed:** Sia > 0 (es. 20 pacchetti).
- **Allow:** Sia uguale ad Analyzed (Snort è in modalità IDS, quindi "permette" il passaggio ma segnala l'evento).



---

6. Risoluzione Problemi Comuni (Troubleshooting)

Se qualcosa va storto, ecco le cause più probabili:

- **Errore "Rules loaded: 0" o Nessun Alert:**
  - *Causa:* Snort non trova il file delle regole o il percorso in `snort.lua` è sbagliato.
  - *Test:* Prova a forzare il caricamento da riga di comando aggiungendo `-R snort/rules/local.rules`. Se così funziona, l'errore è nel file `snort.lua`.
- **Errore `"'=' expected near 'ip'"` o Syntax Error:**
  - *Causa:* Hai incluso il file delle regole fuori dal blocco `ips = { ... }`. Snort sta provando a leggere `alert ip...` come codice Lua e fallisce.
  - *Soluzione:* Sposta l'istruzione `include` dentro le parentesi graffe di `ips`.
- **Permessi Negati / File non trovato:**
  - *Causa:* Stiamo lavorando in una cartella condivisa di VirtualBox (`/media/sf_...`).
  - *Soluzione:* Usa sempre `sudo` per lanciare Snort, in modo da avere i permessi di lettura su tutti i file condivisi.

Sezione 6: Analisi "Live" - Rilevamento in Tempo Reale e Gestione Flussi

In questo capitolo passiamo dall'analisi offline (PCAP) alla modalità "Live": questo significa che Snort ascolterà in tempo reale su un'interfaccia di rete e genererà allarmi mentre il traffico scorre effettivamente attraverso il cavo (o, nel nostro caso, la scheda virtuale).

A differenza del test precedente, qui entrano in gioco variabili dinamiche come la frammentazione dei pacchetti, lo stato delle connessioni TCP (Stream) e l'interazione tra macchine diverse (Attaccante vs Vittima).

---

1. Preparazione delle Regole

Per questo test attiveremo tre regole specifiche che coprono diversi protocolli: ICMP, HTTP e TCP generico. Dobbiamo assicurarci che Snort cerchi pattern specifici (come il metodo "GET" o i flag SYN).

Apri il file delle regole sulla **VM Snort**:

```
nano snort/rules/local.rules
```

Assicurati che le seguenti regole siano presenti e **non commentate** (senza # all'inizio):

```
# 2) ICMP ping verso Snort
alert icmp any any -> 192.168.10.1/24 any (msg:"[LAB] ICMP ping verso
$HOME_NET"; itype:8; sid:1000002; rev:1;)

# 3) HTTP GET (porta 80) - richiesta web base
alert tcp any any -> 192.168.10.1/24 80 (msg:"[LAB] HTTP GET request";
flow:to_server,established; content:"GET"; http_method; sid:1000003;
rev:1;)

# 4) Connessione TCP verso porta 12345
alert tcp any any -> 192.168.10.1/24 12345 (msg:"[LAB] TCP connect verso
12345 (netcat)"; flags:S; sid:1000004; rev:1;)
```

(Nota: Assicurati che l'IP `192.168.10.1/24` corrisponda alla subnet della tua VM).

---

2. Configurazione dei Motori di Analisi

Affinché le regole complesse (come la n. 3 che usa `http_method` e `flow`) funzionino, Snort deve capire il contesto del traffico. Non basta guardare i pacchetti grezzi; deve ricostruire lo stream TCP e decodificare l'HTTP.

In Snort 3, questo si fa configurando esplicitamente i moduli e il **Binder** (il componente che lega le porte ai servizi).

Apri il file di configurazione:

```
sudo nano snort/snort.lua
```

Incolla o verifica la presenza di questa configurazione alla fine del file. È fondamentale per dire a Snort: *"Se vedi traffico sulla porta 80, trattalo come HTTP"*.

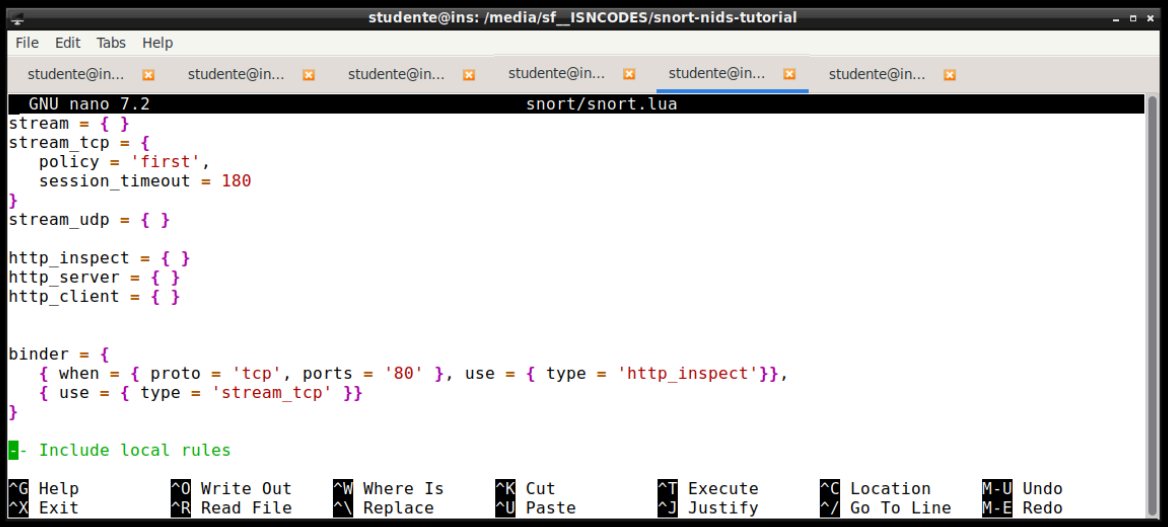
```
--------------------------------------------------------------------------------
---
-- CONFIGURAZIONE MOTORI E BINDER
--------------------------------------------------------------------------------
---

-- 1. Attivazione ispettori base (Stream e HTTP)
stream = { }
stream_tcp = { }
http_inspect = { }

-- 2. Configurazione del BINDER
```

```
-- Serve a dire a Snort quale ispettore usare su quale porta
binder = {
    -- Se è TCP su porta 80 -> Usa il decoder HTTP
    { when = { proto = 'tcp', ports = '80' }, use = { type =
'http_inspect' } },

    -- Per tutto il resto -> Traccia il flusso TCP standard
    { use = { type = 'stream_tcp' } }
}
```



---

3. Preparazione dell'Ambiente di Ascolto

Le regole 3 e 4 rilevano connessioni verso servizi specifici. Se sulla VM Snort non c'è nulla in ascolto su quelle porte, il sistema operativo invierà un pacchetto di *Reset (RST)* immediato, impedendo la creazione dello stream necessario per l'alert.

Sulla **VM Snort**, apri due terminali aggiuntivi per simulare i servizi:

1. **Simulazione Server Web (Porta 80):**

```
sudo python3 -m http.server 80
```

2. **Simulazione Servizio Vulnerabile (Porta 12345):**

```
nc -l -p 12345
```

---

4. Esecuzione di Snort in Modalità Live

Ora avviamo Snort. A differenza del test PCAP, qui dobbiamo specificare l'interfaccia di rete.

Il flag critico `-k none`: Nelle macchine virtuali, il calcolo del checksum dei pacchetti è spesso errato (offloading). Senza `-k none`, Snort scarterebbe silenziosamente i pacchetti HTTP validi considerandoli corrotti.

Esegui questo comando sulla **VM Snort**:

```
sudo snort -c snort/snort.lua -R snort/rules/local.rules -A alert_fast
-l results/log -i [NOME_INTERFACCIA] -k none
```

(Sostituisci `[NOME_INTERFACCIA]` con la tua scheda, es. `eth0` o `enp0s3`. Puoi trovarla con il comando `ip a`).

Se tutto è corretto, Snort caricherà la configurazione e rimarrà in attesa (il cursore non tornerà subito disponibile).

---

5. Generazione del Traffico

Spostiamoci ora sulla **VM Client (Attaccante)** per generare il traffico che farà scattare le regole. *(Sostituisci `192.168.10.x` con l'IP della VM Snort).*

1. **Test ICMP (Regola 2):**

```
ping -c 4 192.168.10.x
```

2. **Test HTTP (Regola 3):**

```
curl http://192.168.10.x
```

3. **Test TCP Connection (Regola 4):**

```
nc -zv 192.168.10.x 12345
```

---

6. Verifica dei Risultati

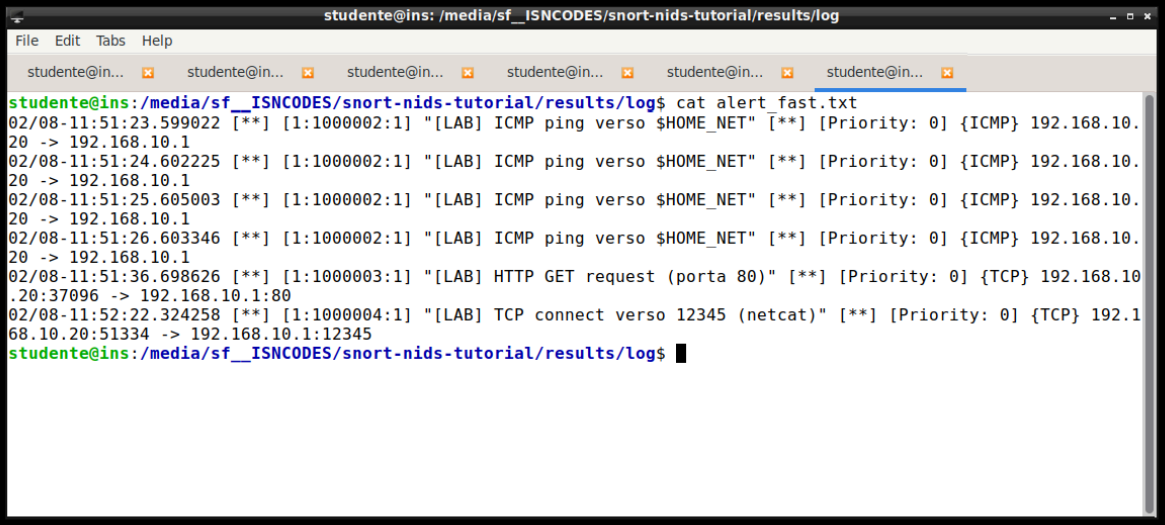
Torna sulla VM Snort. Puoi interrompere Snort con `CTRL+C` oppure aprire un nuovo terminale per leggere i log in tempo reale.

Analizziamo il file di output:

```
cat results/log/alert_fast.txt
```

Dovresti vedere una sequenza di alert che conferma il successo del test live:

```
[**] [1:1000002:1] [LAB] ICMP ping verso $HOME_NET ...
[**] [1:1000003:1] [LAB] HTTP GET request ...
[**] [1:1000004:1] [LAB] TCP connect verso 12345 (netcat) ...
```



Sezione 7: Gestione delle Regole con PulledPork v3 e Analisi di Traffico Malevolo

In questa sezione configureremo **PulledPork v3**, uno strumento essenziale per scaricare, gestire e aggiornare automaticamente le regole di Snort. Nello specifico, useremo il set di regole **LightSPD** (molto performante per Snort 3), adatteremo la configurazione per escludere le regole incompatibili con il nostro ambiente, e infine testeremo il sistema contro un attacco reale multi-stage (**Emotet + IcedID**) per dimostrare che Snort è in grado di rilevare minacce complesse.

---

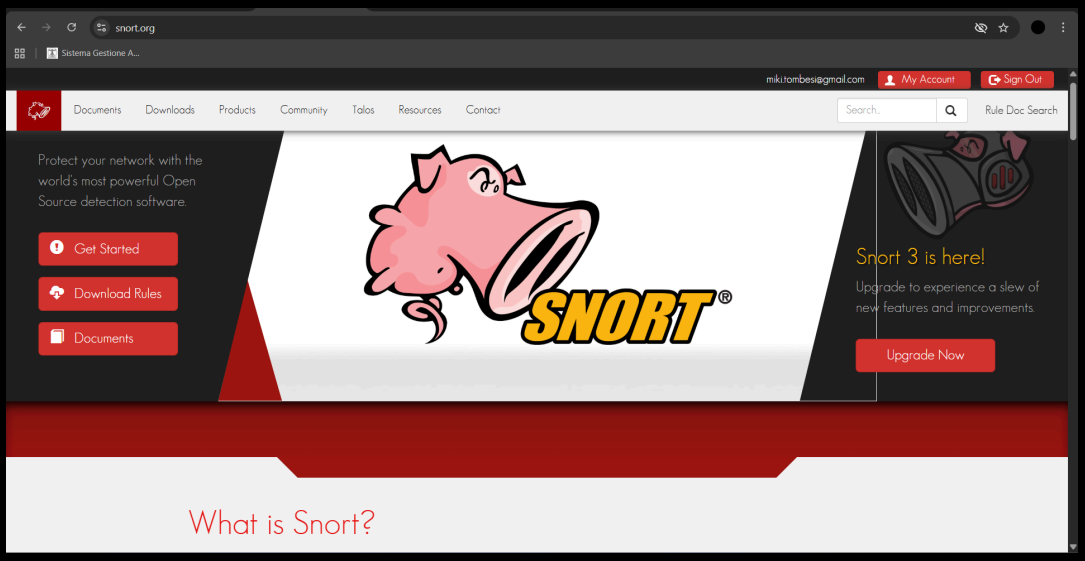
1. Prerequisiti: Oinkcode e Installazione PulledPork

Prima di iniziare, dobbiamo ottenere le credenziali per scaricare le regole ufficiali e installare il software di gestione.

1.1 Ottenere l'Oinkcode

Le regole "Registered" (come le LightSPD) sono gratuite ma richiedono registrazione.

1. Vai su Snort.org e crea un account (Sign Up).
2. Una volta loggato, clicca sul tuo indirizzo email in alto a destra e vai su **Oinkcode**.
3. Copia il codice alfanumerico lungo (es. `5a3b...`). Ti servirà nel prossimo passaggio.



1.2 Installazione di PulledPork v3

Scarichiamo l'ultima versione di PulledPork direttamente dal repository ufficiale e installiamola nel sistema.

Apri il terminale ed esegui questi comandi:

Clona il repository GitHub:

```
git clone https://github.com/shirkdog/pulledpork3.git
```

Entra nella cartella e copia l'eseguibile nella path di sistema:

```
cd pulledpork3
sudo mkdir -p /usr/local/bin
sudo cp pulledpork3.py /usr/local/bin/pulledpork3
sudo chmod +x /usr/local/bin/pulledpork3
```

Prepara le cartelle per i file di configurazione:

```
sudo mkdir -p /usr/local/etc/pulledpork3
sudo cp etc/pulledpork.conf /usr/local/etc/pulledpork3/
sudo cp etc/disablesid.conf /usr/local/etc/pulledpork3/
```

---

2. Configurazione di PulledPork (LightSPD Rules)

Ora dobbiamo configurare PulledPork per scaricare il set di regole **LightSPD**. Questo set è ottimizzato per Snort 3, ma contiene alcune regole "Shared Object" (SO) che richiedono plugin specifici non presenti nella versione open source standard. Configureremo il sistema per scaricarle ma gestirle correttamente.

Obiettivo: Inserire l'Oinkcode, attivare LightSPD e impostare i percorsi corretti.

1. Apri il file di configurazione principale:

```
sudo nano /usr/local/etc/pulledpork3/pulledpork.conf
```

2. Modifica le variabili come segue:
    - Imposta `lightspd_rules` su `true` (e disabilita community/registered).
    - Inserisci il tuo `oinkcode`.
    - Assicurati che il `rule_path` punti alla cartella di Snort.

- **Importante:** Verifica che `disablesid_path` sia attivo (senza `#` davanti) e punti al file corretto.

```
community_rules = false
registered_rules = false
lightspd_rules = true   <-- Attiviamo le regole LightSPD

oinkcode = <INCOLLA_QUI_IL_TUO_CODICE>

# Dove salvare il file finale delle regole
rule_path = /usr/local/etc/snort/rules/snort.rules

# . . .

# Percorso per il file delle esclusioni (FONDAMENTALE per evitare errori SO)
disablesid = /usr/local/etc/pulledpork3/disablesid.conf

# Commenta questo percorso per evitare conflitti con le regole compilate
# sorule_path = /usr/local/etc/so_rules/
```

3. Salva ed esci (`CTRL+O`, `Invio`, `CTRL+X`).

---

3. Gestione delle Regole SO (Shared Object)

Poiché stiamo usando Snort in versione Open Source su Linux, non possiamo caricare le regole pre-compilate (Shared Object) fornite da Cisco, altrimenti Snort darà errore (`SO rule not loaded`). Dobbiamo istruire PulledPork affinché le scarichi ma le **disabiliti automaticamente**.

Apri il file delle esclusioni:

```
sudo nano /usr/local/etc/pulledpork3/disablesid.conf
```

1. Aggiungi questa espressione regolare (PCRE). Il codice `gid:3` identifica proprio le regole Shared Object:

```
pcre:gid:3;
```

2. Salva ed esci.

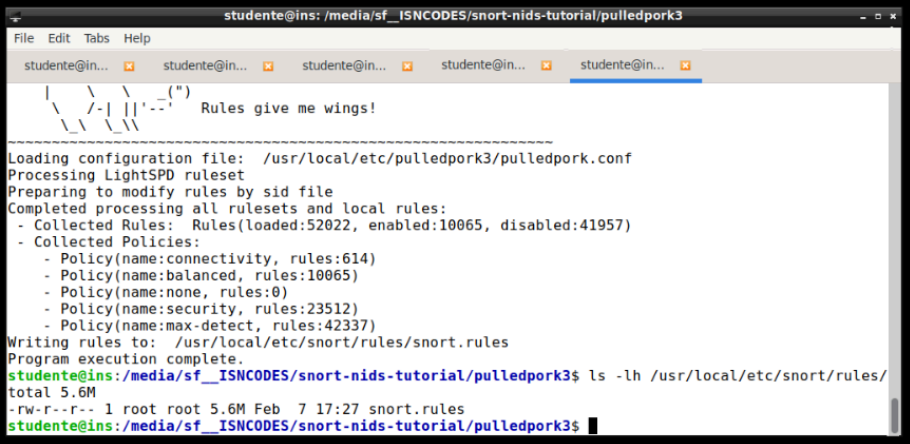
4. Generazione del File delle Regole

Ora che la configurazione è pronta, eseguiamo PulledPork. Il tool scaricherà il pacchetto LightSPD, applicherà il filtro che abbiamo appena creato nel `disablesid.conf` e genererà un unico file pulito chiamato `snort.rules`.

Comando:

```
sudo /usr/local/bin/pulledpork3 -c /usr/local/etc/pulledpork3/pulledpork.conf
```

Se la procedura va a buon fine, vedrai il messaggio: `Program execution complete`.



---

5. Preparazione di Snort (snort.lua)

Dobbiamo collegare il file delle regole appena creato a Snort e configurare la rete per l'analisi forense.

Apri la configurazione di Snort:

```
sudo nano /usr/local/etc/snort/snort.lua
```

1. **Inclusione IPS:** Cerca la sezione `ips` e assicurati che punti al file generato da PulledPork:

```
ips =
{
    include = '/usr/local/etc/snort/rules/snort.rules'
}
```

2. **Variabili di Rete:** Imposta le reti su `any`. Questo è fondamentale quando si analizzano file PCAP scaricati da internet, perché gli IP della vittima non corrispondono alla nostra rete locale.

```
HOME_NET = 'any'
EXTERNAL_NET = 'any'
```

---

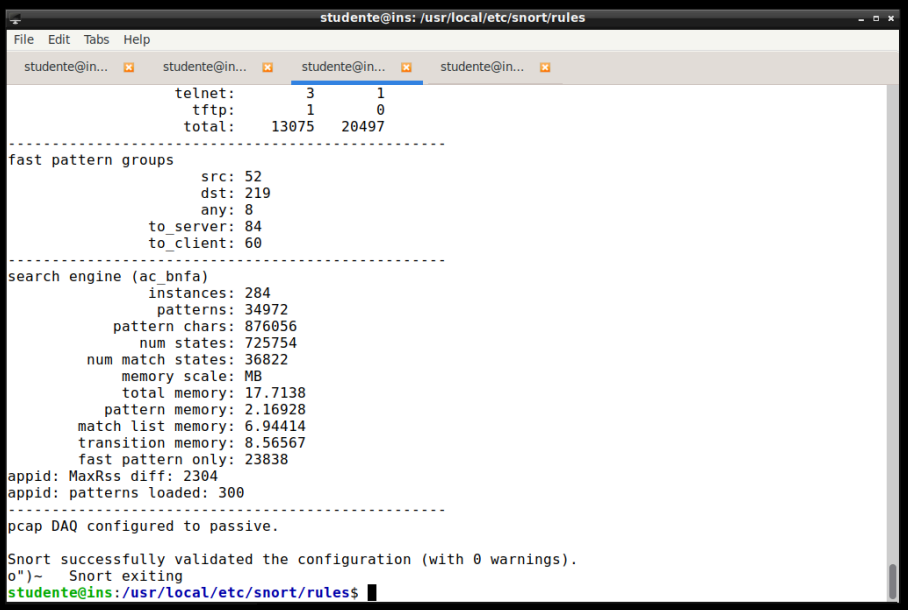
6. Validazione della Configurazione

Prima di procedere, verifichiamo che Snort riesca a leggere le regole LightSPD senza errori.

Comando:

```
sudo snort -c /usr/local/etc/snort/snort.lua
```

Dovresti vedere il messaggio: `Snort successfully validated the configuration`.



7. Analisi del Traffico Malevolo (Emotet + IcedID)

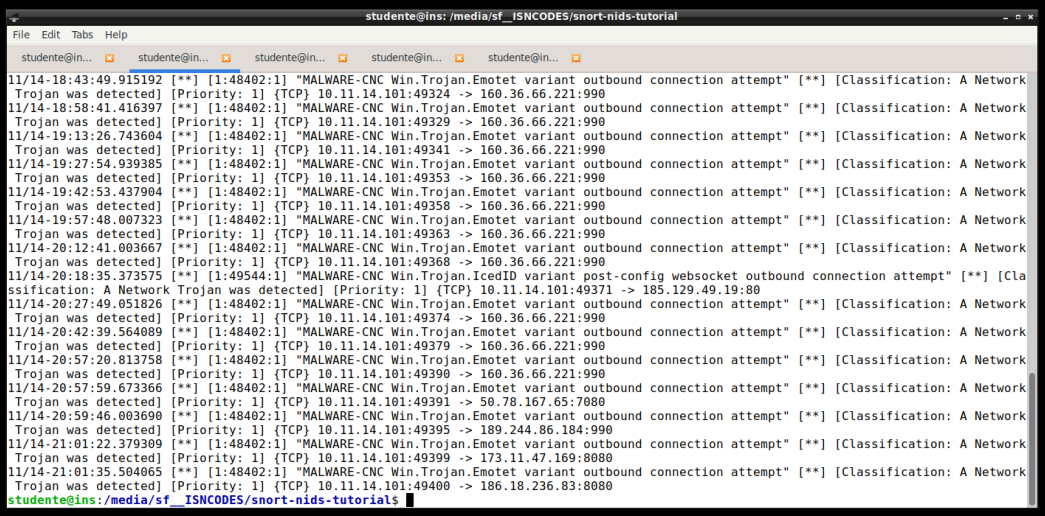
Per testare il sistema, utilizzeremo un file PCAP reale che contiene un'infezione complessa: il malware **Emotet** che scarica successivamente il banking trojan **IcedID**.

1. **Scarica il PCAP:**

```
wget
https://github.com/nipunjaswal/networkforensics/raw/master/Ch6/Emoter%20Banking%20Trojan%20Sample/2018-11-14-Emotet-infection-with-IcedID-banking-Trojan.pcap
```

2. **Esegui l'Analisi:** Lanciamo Snort forzando la lettura del file delle regole e attivando l'output degli alert in console.

```
sudo snort -c /usr/local/etc/snort/snort.lua \
-r pcaps/2018-11-14-Emotet-infection-with-IcedID-banking-Trojan.pcap \
-R /usr/local/etc/snort/rules/snort.rules \
-A alert_fast -k none
```



7.1 Analisi degli Alert Generati

Come possiamo vedere dall'immagine, Snort ha rilevato con successo l'attacco:

- **Emotet Activity:** Gli alert `MALWARE-CNC Win.Trojan.Emotet` indicano che il computer infetto sta contattando i server di comando (C2) della botnet.
- **Payload Secondario:** L'alert `Win.Trojan.IcedID` dimostra che Snort ha individuato anche il secondo malware scaricato da Emotet.

8. Risoluzione Problemi Comuni (Troubleshooting)

- **Errore 422 Unprocessable Entity:** Significa che la versione di Snort è troppo vecchia per lo snapshot richiesto. *Soluzione:* Forza una versione recente in `pulledpork.conf` (es. `snort_version = 3.1.80.0`).
- **Errori "SO rule ... not loaded":** Snort prova a caricare regole compilate senza avere i plugin. *Soluzione:* Assicurati che `disablesid.conf` contenga `pcre:gid:3;` e che il file sia correttamente collegato nel `pulledpork.conf`.
- **Nessun Alert (Solo statistiche):** Se durante l'esecuzione Snort mostra le statistiche dei pacchetti ma **non genera nessun alert** (e nessun file di log), la causa più probabile riguarda la configurazione dell'output.

  Verifica il file `snort.lua`:

  1. Apri il file: `sudo nano /usr/local/etc/snort/snort.lua`
  2. Cerca la sezione `outputs` o `alert_fast`.
  3. Assicurati che la configurazione per `alert_fast` non sia commentata (non deve avere `--` o `#` davanti). Se il modulo di output è disabilitato nel file di configurazione, Snort rileverà gli attacchi ma non potrà scriverli da nessuna parte.
  4. Una volta riattivato ("decommentato"), salva il file e rilancia il comando di analisi.

Sezione 8: Conclusioni

Siamo giunti al termine di questo percorso laboratoriale. Partendo da una macchina virtuale "pulita", abbiamo costruito da zero un **Network Intrusion Detection System (NIDS)** completo e funzionante.

Ripercorrendo le tappe fondamentali, abbiamo:

1. **Compilato Snort 3** dai sorgenti, ottenendo il massimo controllo sull'installazione.
2. **Configurato il motore** in modalità NIDS, comprendendo la differenza tra l'osservazione passiva e la prevenzione attiva (IPS).
3. **Scritto regole personalizzate**, imparando la sintassi per intercettare pattern specifici (ICMP, HTTP, TCP flags).
4. **Analizzato il traffico** in due contesti diversi:
    - **Offline (Forensics):** Ricostruendo attacchi passati (Emotet/IcedID) tramite file PCAP.
    - **Live (Real-time):** Intercettando attacchi simulati in diretta tra due macchine virtuali.
5. **Automatizzato la Threat Intelligence** con PulledPork v3, integrando le regole professionali di Cisco Talos (LightSPD).

---

1. Perché abbiamo scelto Snort 3?

Una domanda legittima che potrebbe sorgere è: *"Perché utilizzare Snort 3, che richiede una configurazione basata su Lua e una compilazione più articolata, rispetto al classico Snort 2.9?"*

La scelta non è solo una questione di "novità", ma di **architettura e prestazioni**. Ecco le differenze critiche che rendono Snort 3 (spesso chiamato Snort++) superiore:

1.1 Architettura Multi-Threaded

Questa è la differenza più impattante.

- **Snort 2:** È *single-threaded*. Un singolo processo deve gestire tutto il traffico. Se il volume di dati aumenta drasticamente, un solo core della CPU si satura e i pacchetti vengono persi (packet loss), rendendo l'IDS cieco.
- **Snort 3:** È nativamente **multi-threaded**. Può avviare più thread di elaborazione dei pacchetti che lavorano in parallelo su tutti i core disponibili della CPU. Questo garantisce un throughput (capacità di gestione del traffico) enormemente superiore e lo rende adatto alle moderne reti ad alta velocità (10Gbps+).

1.2 Configurazione in Lua

- **Snort 2:** Utilizzava un file `snort.conf` con una sintassi rigida e complessa, difficile da validare programmaticamente.
- **Snort 3:** Utilizza **Lua**, un vero linguaggio di scripting. Questo permette di usare variabili, cicli, inclusioni logiche e rende il file di configurazione (`snort.lua`) modulare e molto più leggibile per l'operatore umano. Come abbiamo visto, permette di configurare moduli complessi (come `http_inspect` o il `binder`) in modo strutturato.

1.3 Gestione HTTP Migliorata

Il web è il vettore di attacco principale oggi. Snort 2 utilizzava un preprocessore HTTP vecchio e monolitico. Snort 3 introduce il nuovo ispettore **HTTP/2** e una gestione molto più granulare dei campi HTTP (header, cookie, metodi, body), riducendo i falsi positivi e aumentando la capacità di rilevare attacchi web complessi.

1.4 Uniformità delle Regole

Snort 3 semplifica la sintassi delle regole (es. eliminando la necessità di specificare `msg` in alcuni contesti o gestendo meglio i buffer) e introduce i "Service Inspectors" che rilevano automaticamente il servizio (es. SSH sulla porta 2222) senza dover legare rigidamente le regole alle porte standard.

---

2. Prospettive Future: Cosa fare dopo?

Il sistema che abbiamo configurato è una base solida, ma nel mondo reale un NIDS è parte di un ecosistema più ampio. Ecco alcune evoluzioni possibili per questo progetto:

- **Integrazione SIEM (ELK/Splunk):** Leggere i file `alert_fast.txt` è faticoso. Il passo successivo è configurare Snort per generare output in formato **JSON** (`alert_json`) e inviarlo a una dashboard grafica (come Elastic Stack o Splunk) per visualizzare grafici e statistiche in tempo reale.
- **Passaggio a IPS:** Una volta perfezionate le regole per evitare falsi positivi, si può cambiare la modalità di Snort da NIDS a IPS (Inline), permettendogli non solo di segnalare l'attacco Emotet, ma di **bloccare la connessione** e salvare la rete dall'infezione.
- **Analisi File (File Extraction):** Snort 3 ha capacità avanzate per estrarre file dal traffico di rete. Si potrebbe configurare il sistema per salvare automaticamente tutti gli `.exe` o i `.pdf` sospetti per un'analisi antivirus successiva.

In conclusione, questo laboratorio ha fornito gli strumenti essenziali per comprendere non solo "come si installa" un IDS, ma "come ragiona" un analista di sicurezza di fronte al traffico di rete: osservare, comprendere il protocollo, definire la minaccia e creare la regola per contrastarla.