BW3 - Esercizio 2

Analisi dei Processi e Monitoraggio di Rete in Ambiente Linux

Autore: Cybereagles

Sintesi

Il presente report documenta un'attività pratica di ***System & Network Auditing*** eseguita su un host **Linux**. L'obiettivo principale dell'analisi è l'esplorazione e il monitoraggio dei processi di sistema e delle connessioni di rete attive, al fine di identificare in modo univoco i servizi in esecuzione e le relative porte in ascolto. L'attività di indagine ha permesso di analizzare la gerarchia dei processi applicativi, mappare le connessioni e testare il comportamento dei protocolli di **Livello 4 (TCP e UDP)**.

Scopo del test e analisi dello scenario

Scenario e Obiettivi

L'ispezione dettagliata è stata condotta all'interno **dell'host stesso** per esplorare le relazioni logiche tra i processi e la rete. L'obiettivo è verificare la corretta esecuzione dei servizi, con un focus specifico sull'architettura e la sicurezza del web server ***nginx***.

- **Attacker:** Host locale (localhost) utilizzato unicamente per le procedure di auditing e la diagnostica interna.
- **Target:** Macchina bersaglio Linux (127.0.0.1) analizzata per mappare le esposizioni di rete e le configurazioni applicative.

Strumenti

- **ps**: Strumento da riga di comando essenziale utilizzato per la diagnostica profonda e la visualizzazione base, avanzata e gerarchica dei processi attivi sul sistema.
- **netstat**: Utility impiegata per ispezionare le porte in ascolto e le connessioni di rete stabilite, mappando in modo inequivocabile i *socket* ai rispettivi ***PID e applicativi***.
- **telnet**: Strumento di diagnostica impiegato per testare operativamente l'accessibilità delle **porte TCP**, interagire manualmente con i protocolli e ispezionare il ***raw output***.

Svolgimento

Fase 1: Analisi e Monitoraggio dei Processi

L'indagine è iniziata con l'analisi base dei processi attivi per poi procedere verso una diagnostica più profonda. È stato eseguito il comando in formato completo (full format, -f) e lungo (long, -l) per visualizzare tutti i processi di sistema (everything, -e), estraendo **dettagli tecnici avanzati (UID, PPID, PRI, NI)** solitamente nascosti a un utente normale.

```
$ sudo ps -elf
```

Successivamente, per comprendere le relazioni logiche tra i vari servizi operativi, è stata istruita la **shell** a mostrare graficamente l'indentazione ad albero genealogico (**Hierarchy**).

```
$ sudo ps -ejH
```

```
    802     802     802 ?        00:00:00   nginx
    803     802     802 ?        00:00:00     nginx
```

Figura 1 Output del terminale che mostra la gerarchia dei processi, evidenziando chiaramente i processi padre allineati a sinistra e i relativi processi figlio indentati verso destra.

Fase 2: Analisi delle Connessioni di Rete

Per ispezionare le configurazioni di rete, è stato impiegato il tool di monitoraggio utilizzando i privilegi di ***root***, passaggio fondamentale per associare i programmi (-p, mostra i PID) ai *socket* (-a, mostra tutti i socket). L'output è stato formattato numericamente per evitare la **risoluzione DNS** (-n) e mostrare sia il **traffico TCP che UDP** (tcp, -t; udp, -u)**.**

```
$ sudo netstat -tunap
```

Dall'analisi delle colonne prodotte, è stato possibile determinare la condizione attuale dei servizi, identificando lo stato ***LISTEN*** sulla **porta 80 TCP** e associandola immediatamente al servizio web.

```
[analyst@secOps ~]$ sudo netstat -tunap
[sudo] password for analyst:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      802/nginx: master p
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      403/sshd: /usr/bin/
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      419/vsftpd
tcp        0      0 0.0.0.0:6633            0.0.0.0:*               LISTEN      390/python3.9
tcp6       0      0 :::22                   :::*                    LISTEN      403/sshd: /usr/bin/
udp        0      0 10.0.2.15:68            0.0.0.0:*                           300/systemd-network
```

Figura 2 Dettaglio dell'output di rete che elenca i protocolli di trasporto, gli indirizzi locali, lo stato delle connessioni (LISTEN) e la colonna cruciale con i PID associati ai programmi.

Fase 3: Investigazione Architetturale del Web Server Nginx

A seguito delle evidenze di rete, si è proceduto ad analizzare il binario eseguibile responsabile della **porta 80**. È stato filtrato l'output globale dei processi per isolare l'applicativo specifico.

```
$ sudo ps -elf | grep 802
```

L'indagine ha confermato l'adozione di una **best practice** di sicurezza strutturale: il ***Master Process (PID 802)*** viene eseguito da ***root*** per aprire le porte critiche, mentre il **Worker Process (PID 803)**, delegato alla gestione del traffico, è confinato ai privilegi limitati dell'utente ***http***. Tale separazione previene la compromissione totale del sistema in caso di attacco al processo esposto.

```
[analyst@secOps ~]$ sudo ps -elf | grep 802
[sudo] password for analyst:
1 I root         346       2  0  60 -20 -     0 rescue 08:41 ?        00:00:00 [kworker/R-cfg80211]
1 S root         802       1  0  80   0 -  3723 sigsus 08:51 ?        00:00:00 nginx: master process /usr/sbin/nginx
5 S http         803     802  0  80   0 -  3827 do_epo 08:51 ?        00:00:00 nginx: worker process
0 S analyst      925     744  0  80   0 -  1615 anon_p 09:22 pts/0    00:00:00 grep 802
```

Figura 3 Finestra del terminale con l'output filtrato che dimostra la divisione dei privilegi tra l'utente root (master) e l'utente http (worker).

Fase 4: Diagnostica di Rete e Banner Grabbing

Per verificare operativamente l'accessibilità dei servizi, è stata stabilita una **connessione grezza** alla **porta 80** inviando un input testuale anomalo ("*provaprova*").

```
$ telnet 127.0.0.1 80
```

Il servizio ha restituito correttamente un **errore HTTP 400 Bad Request** formattato in HTML. Questa procedura ha consentito di estrarre il ***raw output*** e individuare un'esposizione di informazioni: la versione esatta del server (*nginx/1.28.0*) è visibile negli **header**. Successivamente, si è tentata un'interazione diagnostica con la **porta locale 68**.

```
$ telnet 127.0.0.1 68
```

Il sistema operativo ha rifiutato istantaneamente la connessione (***Connection refused***) a causa di **un'incompatibilità a Livello 4**: il servizio bersaglio operava su **protocollo UDP (*systemd-network*)**, mentre il tool diagnostico impiegato è supportato unicamente su **standard TCP**.

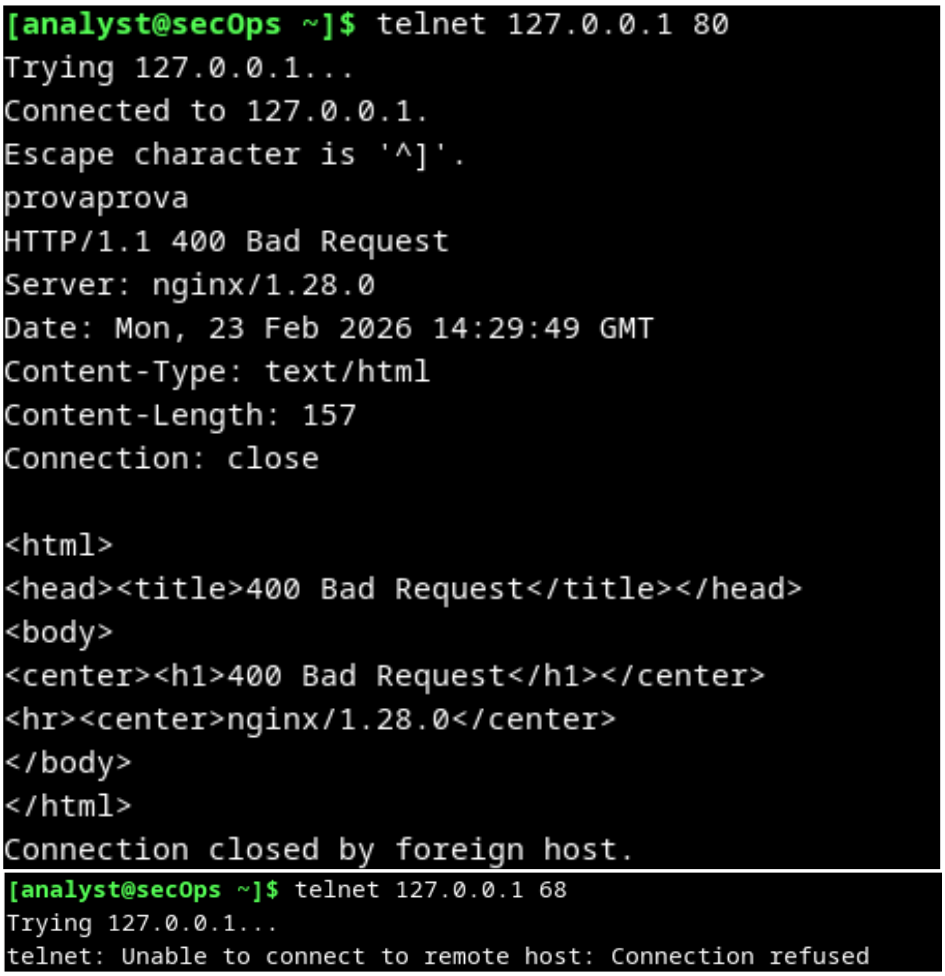

Figura 4 Schermate che documentano l'estrazione del banner HTTP con la versione in chiaro del web server e il successivo test fallito di incompatibilità sulla porta UDP.

Conclusioni

L'attività di laboratorio ha dimostrato l'importanza critica degli **strumenti a riga di comando** per l'amministrazione e la diagnostica di sicurezza in tempo reale. Si è proceduto con successo alla **mappatura inequivocabile della superficie d'attacco**, distinguendo il traffico di trasporto e analizzando l'architettura sicura a privilegi separati del web server esposto. Le tecniche di diagnostica rapida hanno inoltre permesso di ispezionare gli **header** per individuare eventuali esposizioni di versione.

Si raccomandano le seguenti azioni di mitigazione:

- **Confinamento del protocollo Telnet**: Si sottolinea che il protocollo in questione non è sicuro per le operazioni di amministrazione remota, poiché trasmette ogni dato in chiaro. Il suo utilizzo deve essere rigidamente confinato ad ambienti di audit isolati per attività di *troubleshooting* e *banner grabbing*.
- **Rimozione del Banner HTTP**: Si consiglia di disabilitare la direttiva che permette l'esposizione in chiaro della versione del software (*nginx/1.28.0*) negli header e nelle pagine di errore, al fine di mitigare le fasi di ricognizione da parte di potenziali aggressori.