Report Esercizio 2: Analisi di Server e Servizi su Linux

A cura di Iris Canole, Federico Giannini, Daniele Castello, Luca Pani, Rosario Papa, Yari Olmi, Alessandro Salerno

Introduzione

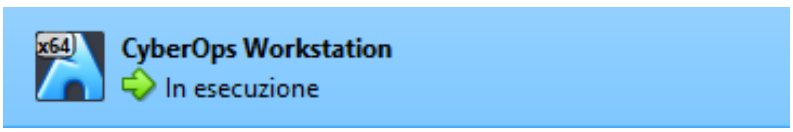
In questo report, documento le procedure che abbiamo utilizzato per identificare, analizzare e testare i server e i servizi in esecuzione su una macchina virtuale CyberOps Workstation. L'attività si è concentrata sull'impiego di strumenti a riga di comando standard dell'ambiente Linux, fondamentali per qualsiasi analista di sicurezza o amministratore di sistema. I principali strumenti che abbiamo impiegato in questa analisi sono `ps` per l'ispezione dei processi, `netstat` per l'analisi delle connessioni di rete e `telnet` per il testing diretto dei servizi TCP. L'analisi seguirà un percorso logico, partendo dall'identificazione generale dei processi attivi fino alla verifica diretta e mirata della natura dei servizi di rete esposti dal sistema.

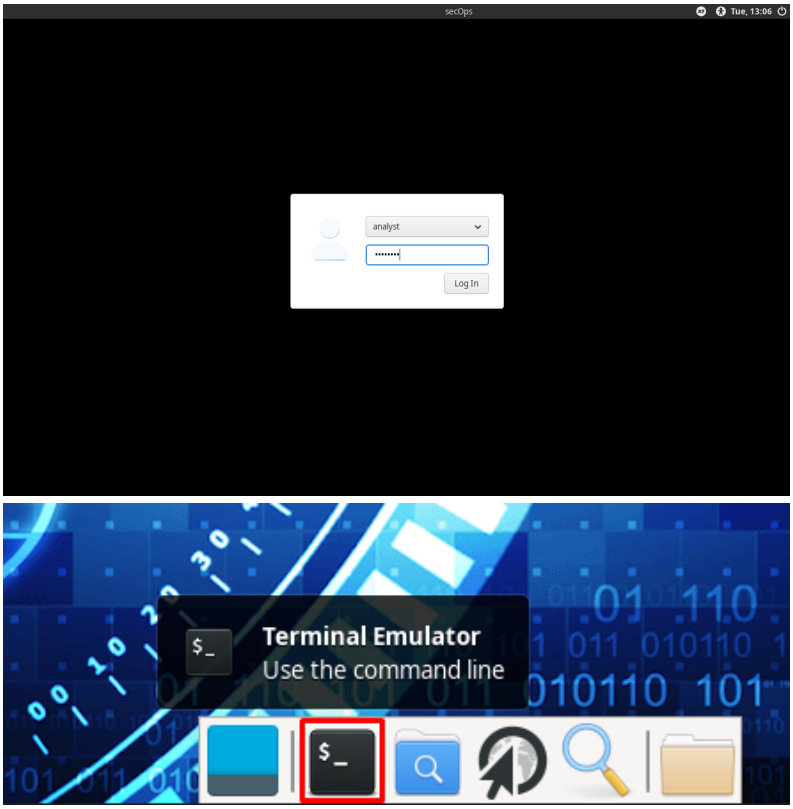
1. Parte 1: Identificazione dei Server e dei Processi

Comprendere quali processi e servizi sono in esecuzione su un sistema è un'attività fondamentale. Questa conoscenza non solo è cruciale per la gestione quotidiana e il troubleshooting, ma rappresenta anche il primo passo in qualsiasi analisi di sicurezza. Identificare i servizi attivi permette di definire la superficie d'attacco del sistema e di verificare che solo i servizi autorizzati siano in esecuzione.

1.1. Accesso al Sistema e Analisi Iniziale dei Processi

Abbiamo iniziato l'analisi con l'accesso al sistema. La procedura ha previsto l'avvio della macchina virtuale CyberOps Workstation, seguito dal login con l'utente `analyst`. Successivamente, abbiamo aperto una sessione a riga di comando tramite l'applicazione Terminal Emulator.





Una volta ottenuto l'accesso alla shell, abbiamo utilizzato il comando sudo `ps -elf` per ottenere un elenco completo di tutti i programmi (o processi) in esecuzione sul sistema.

```
[analyst@secOps ~]$ ps
    PID TTY          TIME CMD
    792 pts/0    00:00:00 bash
    829 pts/0    00:00:00 ps
[analyst@secOps ~]$ sudo ps
    PID TTY          TIME CMD
    832 pts/1    00:00:00 sudo
    833 pts/1    00:00:00 ps
[analyst@secOps ~]$ sudo ps -elf
F S UID          PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root           1       0  0  80   0 -  5534 do_epo 13:02 ?        00:00:00 /sbin/init
1 S root           2       0  0  80   0 -     0 kthrea 13:02 ?        00:00:00 [kthreadd]
1 S root           3       2  0  80   0 -     0 kthrea 13:02 ?        00:00:00 [pool_workqueue_release]
1 I root           4       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-rcu_gp]
1 I root           5       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-sync_wq]
1 I root           6       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-kvfree_rcu_reclaim]
1 I root           7       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-slub_flushwq]
1 I root           8       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-netns]
1 I root           9       2  0  80   0 -     0 worker 13:02 ?        00:00:01 [kworker/0:0-events]
1 I root          12       2  0  80   0 -     0 worker 13:02 ?        00:00:02 [kworker/u8:0-ext4-rsv-conversion]
1 I root          14       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-mm_percpu_wq]
1 S root          15       2  0  80   0 -     0 smpboo 13:02 ?        00:00:00 [ksoftirqd/0]
1 I root          16       2  0  58   - -     0 rcu_gp 13:02 ?        00:00:00 [rcu_preempt]
1 S root          17       2  0  58   - -     0 rcu_bo 13:02 ?        00:00:00 [rcub/0]
1 S root          18       2  0  80   0 -     0 kthrea 13:02 ?        00:00:00 [rcu_exp_par_gp_kthread_worker/0]
1 S root          19       2  0  80   0 -     0 kthrea 13:02 ?        00:00:00 [rcu_exp_gp_kthread_worker]
1 S root          20       2  0 -40   - -     0 smpboo 13:02 ?        00:00:00 [migration/0]
1 S root          21       2  0   9   - -     0 smpboo 13:02 ?        00:00:00 [idle_inject/0]
1 S root          22       2  0  80   0 -     0 smpboo 13:02 ?        00:00:00 [cpuhp/0]
1 S root          23       2  0  80   0 -     0 smpboo 13:02 ?        00:00:00 [cpuhp/1]
1 S root          24       2  0   9   - -     0 smpboo 13:02 ?        00:00:00 [idle_inject/1]
1 S root          25       2  0 -40   - -     0 smpboo 13:02 ?        00:00:00 [migration/1]
1 S root          26       2  0  80   0 -     0 smpboo 13:02 ?        00:00:00 [ksoftirqd/1]
1 I root          28       2  0  60 -20 -     0 worker 13:02 ?        00:00:00 [kworker/1:0H-events_highpri]
5 S root          29       2  0  80   0 -     0 devtmp 13:02 ?        00:00:00 [kdevtmpfs]
1 I root          30       2  0  60 -20 -     0 rescue 13:02 ?        00:00:00 [kworker/R-inet_frag_wq]
1 I root          31       2  0  80   0 -     0 rcu_ta 13:02 ?        00:00:00 [rcu_tasks_kthread]
1 I root          32       2  0  80   0 -     0 rcu_ta 13:02 ?        00:00:00 [rcu_tasks_rude_kthread]
1 I root          33       2  0  80   0 -     0 rcu_ta 13:02 ?        00:00:00 [rcu_tasks_trace_kthread]
```

"Perché è stato necessario eseguire ps come root (premettendo il comando con sudo)?"

Abbiamo dovuto eseguire il comando con privilegi di root tramite sudo perché un utente standard

come analyst ha una visibilità limitata ai soli processi di sua proprietà. L'utilizzo di sudo mi ha permesso di superare queste restrizioni, fornendo una visione completa e dettagliata di tutti i processi in esecuzione sull'intero sistema, inclusi i servizi di sistema critici avviati dall'utente root.

1.2. Analisi della Gerarchia dei Processi

In un sistema Linux, i processi sono organizzati in una struttura gerarchica, dove ogni processo (ad eccezione del primo, `init/systemd`) ha un processo genitore. Per visualizzare questa gerarchia, abbiamo prima avviato il web server Nginx con il comando `sudo /usr/sbin/nginx` e successivamente abbiamo eseguito sudo `ps -ejH`.

```
[analyst@secOps ~]$ sudo /usr/sbin/nginx
[analyst@secOps ~]$ sudo ps -ejH
    PID    PGID     SID TTY          TIME CMD
      2       0       0 ?        00:00:00 kthreadd
      3       0       0 ?        00:00:00   pool_workqueue_release
      4       0       0 ?        00:00:00   kworker/R-rcu_gp
      5       0       0 ?        00:00:00   kworker/R-sync_wq
      6       0       0 ?        00:00:00   kworker/R-kvfree_rcu_reclaim
      7       0       0 ?        00:00:00   kworker/R-slub_flushwq
      8       0       0 ?        00:00:00   kworker/R-netns
      9       0       0 ?        00:00:01   kworker/0:0-events
     12       0       0 ?        00:00:03   kworker/u8:0-flush-8:0
     14       0       0 ?        00:00:00   kworker/R-mm_percpu_wq
     15       0       0 ?        00:00:00   ksoftirqd/0
     16       0       0 ?        00:00:00   rcu_preempt
     17       0       0 ?        00:00:00   rcub/0
     18       0       0 ?        00:00:00   rcu_exp_par_gp_kthread_worker/0
     19       0       0 ?        00:00:00   rcu_exp_gp_kthread_worker
     20       0       0 ?        00:00:00   migration/0
     21       0       0 ?        00:00:00   idle_inject/0
     22       0       0 ?        00:00:00   cpuhp/0
     23       0       0 ?        00:00:00   cpuhp/1
     24       0       0 ?        00:00:00   idle_inject/1
     25       0       0 ?        00:00:00   migration/1
     26       0       0 ?        00:00:00   ksoftirqd/1
     28       0       0 ?        00:00:00   kworker/1:0H-events_highpri
     29       0       0 ?        00:00:00   kdevtmpfs
     30       0       0 ?        00:00:00   kworker/R-inet_frag_wq
     31       0       0 ?        00:00:00   rcu_tasks_kthread
     32       0       0 ?        00:00:00   rcu_tasks_rude_kthread
     33       0       0 ?        00:00:00   rcu_tasks_trace_kthread
     34       0       0 ?        00:00:00   kauditd
     35       0       0 ?        00:00:00   khungtaskd
     36       0       0 ?        00:00:00   oom_reaper
     38       0       0 ?        00:00:00   kworker/R-writeback
     39       0       0 ?        00:00:00   kcompactd0
     40       0       0 ?        00:00:00   ksmd
     41       0       0 ?        00:00:02   khugepaged
     42       0       0 ?        00:00:00   kworker/R-kblockd
```

"Come viene rappresentata la gerarchia dei processi da ps?"

La gerarchia viene rappresentata visivamente attraverso l'indentazione. Ogni processo figlio è rientrato (spaziato verso destra) rispetto al suo processo genitore, rendendo immediatamente riconoscibile la relazione di parentela e la struttura ad albero dei processi del sistema.

1.3. Ispezione delle Connessioni di Rete con `netstat`

Per identificare i server di rete e analizzare le loro connessioni, abbiamo utilizzato il comando `netstat`. Per ottenere un output più leggibile e informativo, l'abbiamo eseguito con le opzioni `-tunap`.

"Qual è il significato delle opzioni -t, -u, -n, -a e -p in netstat?"

Le opzioni utilizzate hanno il seguente significato:

- `-t`: Mostra unicamente le connessioni che utilizzano il protocollo **TCP**.
- `-u`: Mostra unicamente le connessioni che utilizzano il protocollo **UDP**.
- `-n`: Mostra indirizzi e porte in formato **numerico**, evitando la risoluzione dei nomi (DNS), che rende l'output più rapido e chiaro.
- `-a`: Mostra **tutte** le socket, sia quelle in stato di ascolto (LISTEN) sia quelle con connessioni già stabilite.
- `-p`: Mostra il **PID** (Process ID) e il nome del **programma** associato a ciascuna connessione, informazione cruciale per l'analisi.

"L'ordine delle opzioni è importante per netstat?"

No, l'ordine delle opzioni non è importante. Nei comandi Linux standard, le opzioni brevi possono essere raggruppate dopo un singolo trattino in qualsiasi sequenza, producendo lo stesso risultato.

1.4. Correlazione tra Servizi di Rete e Processi

L'analisi più efficace si ottiene correlando le informazioni fornite da `netstat` con quelle di `ps`. Questo permette di identificare con precisione quale programma è responsabile di una specifica porta di rete aperta.

"Basandosi sull'output di netstat, qual è il protocollo di Livello 4, lo stato della connessione e il PID del processo in esecuzione sulla porta 80?"

Dall'output si evince che il processo sulla porta 80 utilizza il protocollo **TCP**, si trova in stato **LISTEN** (in ascolto di nuove connessioni) e ha un Process ID (PID) di **897**.

"Sebbene i numeri di porta siano solo una convenzione, puoi indovinare che tipo di servizio è in esecuzione sulla porta 80 TCP?"

La porta 80 TCP è universalmente associata per convenzione al protocollo **HTTP**. È quindi molto probabile che il servizio in esecuzione sia un Web Server.

Per confermare e ottenere maggiori dettagli sul processo con PID 897, abbiamo utilizzato il comando `sudo ps -elf | grep 897` per filtrare l'output di `ps`.

```
[analyst@secOps ~]$ sudo ps -elf | grep 897
1 S root         897       1  0  80   0 -  3723 sigsus 13:27 ?        00:00:00 nginx: master process /usr/sbin/nginx
5 S http         898     897  0  80   0 -  3827 do_epo 13:27 ?        00:00:00 nginx: worker process
0 S analyst     1098     792  0  80   0 -  1615 anon_p 14:16 pts/0    00:00:00 grep 897
```

"Il processo PID 897 è nginx. Come si potrebbe concludere questo dall'output sopra?"
La conclusione è diretta: l'ultima colonna della prima riga, che corrisponde al PID 897, mostra esplicitamente il comando `nginx: master process /usr/sbin/nginx`. Questo identifica in modo inequivocabile il programma in esecuzione.

"Cos'è nginx? Qual è la sua funzione?"
Nginx è un software open-source estremamente popolare che funge primariamente da **Web Server**, servendo contenuti web (pagine HTML, immagini, etc.) ai client come i browser. È noto per le sue alte prestazioni e il basso consumo di risorse. Oltre a questo, viene comunemente impiegato come **Reverse Proxy** (per smistare e proteggere il traffico verso altri server) e come **Load Balancer** (per distribuire il carico di lavoro tra più server).

"La seconda riga mostra che il processo 898 è di proprietà di un utente chiamato http e ha il processo numero 897 come genitore. Cosa significa? È un comportamento comune?"
Sì, questo è un comportamento molto comune e rappresenta una best practice di sicurezza nota come **separazione dei privilegi**. Il processo master (PID 897) viene avviato con i privilegi di `root` per potersi legare a porte privilegiate (sotto la 1024), come la porta 80. Successivamente, genera uno o più processi "worker" (in questo caso, il figlio PID 898) che vengono eseguiti con un utente con privilegi limitati (`http`) per gestire le richieste esterne non fidate. In questo modo, se un worker venisse compromesso, l'attaccante non otterrebbe i privilegi di root, aderendo al principio del privilegio minimo e limitando notevolmente i danni potenziali al sistema.

"Perché l'ultima riga mostra grep 897?"
Questa riga appare perché, quando si esegue la pipeline `ps -elf | grep 897`, il comando ps crea un'istantanea di tutti i processi in esecuzione in quel preciso istante. In quell'istante, anche il comando `grep 897` è in esecuzione. Poiché la stringa del comando grep contiene "897", esso soddisfa il criterio di ricerca e quindi "trova se stesso" nell'elenco dei processi, comparendo nell'output finale.

Con l'identificazione statica completata, l'analisi deve ora passare a una fase dinamica: l'interrogazione diretta dei servizi per confermare la loro identità e il loro comportamento, un compito per cui telnet si rivela uno strumento diagnostico essenziale.

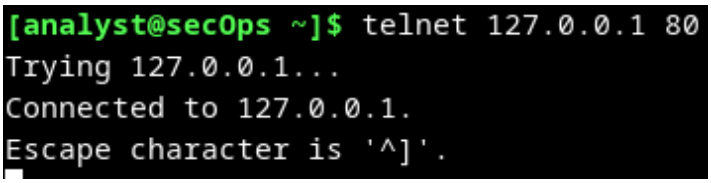
---

2. Parte 2: Usare Telnet per Testare i Servizi TCP

Mentre `ps` e `netstat` sono eccellenti per identificare i servizi in esecuzione, `telnet` offre la possibilità di interagire direttamente con essi. Sebbene telnet sia uno strumento di diagnostica versatile, è fondamentale sottolineare che è **intrinsecamente insicuro** per la gestione remota, poiché tutte le comunicazioni, incluse le credenziali, avvengono in chiaro (senza crittografia). Il suo utilizzo in questo contesto è strettamente limitato al testing e alla raccolta di informazioni.

2.1. Verifica del Web Server sulla Porta 80

Per confermare che il servizio sulla porta 80 fosse effettivamente un web server, abbiamo stabilito una connessione tramite il comando `telnet 127.0.0.1 80`.



Dopo aver stabilito la connessione, inviando una richiesta non valida secondo il protocollo HTTP (in questo caso, una semplice stringa di testo) e premendo Invio, abbiamo provocato una risposta di errore dal server.

```
fsfsfs
HTTP/1.1 400 Bad Request
Server: nginx/1.28.0
Date: Tue, 09 Dec 2025 19:51:26 GMT
Content-Type: text/html
Content-Length: 157
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.28.0</center>
</body>
</html>
Connection closed by foreign host.
```

L'output ricevuto è una risposta HTTP `400 Bad Request` formattata in HTML. Questa risposta non solo conferma in modo inequivocabile la presenza di un web server (identificato come `nginx/1.28.0`), ma dimostra anche che l'intero stack di rete del sistema, fino al Livello 7 (Applicazione), è correttamente funzionante. Dal punto di vista della sicurezza, questa pratica, nota come "banner grabbing", è una forma di raccolta di informazioni che potrebbe essere sfruttata da un utente malintenzionato per ricercare vulnerabilità note specifiche per quella versione.

"Perché l'errore è stato inviato come pagina web?"
L'errore è stato formattato come pagina web (in HTML) perché il server Nginx è progettato per comunicare secondo il protocollo HTTP, il cui client tipico è un browser web. Di conseguenza, anche i messaggi di errore sono standardizzati per essere interpretati e visualizzati correttamente da un browser.

2.2. Interrogazione di Altri Servizi TCP

Abbiamo applicato la stessa tecnica ad altri servizi. Abbiamo effettuato un test sulla porta 22, tipicamente associata al servizio SSH (Secure Shell), con il comando `telnet 127.0.0.1 22`.

```
[analyst@secOps ~]$ telnet 127.0.0.1 22
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_10.0
fsfsfs
Invalid SSH identification string.
Connection closed by foreign host.
```

La risposta del server, `SSH-2.0-OpenSSH_10.0`, è un "banner" che identifica chiaramente il servizio come un server SSH, confermando la sua natura.

"Usa Telnet per connetterti alla porta 68. Cosa succede? Spiega."
Il tentativo di connessione alla porta 68 è fallito immediatamente.

```
[analyst@secOps ~]$ telnet 127.0.0.1 68
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

Il messaggio `telnet`: Unable to connect to remote host: Connection refused indica che il sistema operativo ha rifiutato attivamente la richiesta. Questo accade perché `telnet` è un client esclusivamente **TCP**, mentre il servizio DHCP sulla porta 68 opera su **UDP**. Non essendoci alcun processo in ascolto TCP su quella porta, il kernel del sistema operativo rifiuta attivamente il tentativo di connessione, come da protocollo.

---

3. Domande di Riflessione

3.1. Vantaggi di `netstat`

"Quali sono i vantaggi dell'uso di netstat?"
`netstat` è uno strumento diagnostico fondamentale per l'analisi di rete di un sistema. I suoi principali vantaggi includono:

- **Visibilità completa:** Offre una "radiografia" dettagliata di tutte le porte aperte, le connessioni di rete attive (sia in entrata che in uscita) e le tabelle di routing.
- **Associazione Processo-Porta:** L'opzione `-p` è cruciale perché permette di collegare ogni connessione di rete a un PID specifico e al nome del programma corrispondente, rendendo immediata l'identificazione della sorgente di ogni attività di rete.
- **Diagnostica:** È indispensabile per individuare rapidamente servizi non autorizzati, porte aperte inaspettatamente o problemi di configurazione di rete

3.2. Vantaggi e Sicurezza di `Telnet`

"Quali sono i vantaggi dell'uso di Telnet? È sicuro?"
Vantaggi:

- **Test di connettività:** È uno strumento eccellente per verificare in modo rapido e semplice se una porta TCP è aperta e raggiungibile su un host locale o remoto.

- **Debug dei servizi:** Permette di "parlare" direttamente con un servizio basato su testo, inviando comandi e osservandone le risposte in tempo reale. Questo è utile per verificare versioni, banner e funzionalità base di un servizio.
- **Semplicità:** È un'utilità di base, presente su quasi tutti i sistemi operativi e non richiede configurazioni complesse per un test veloce.

Sicurezza:

NO, Telnet non è sicuro per l'amministrazione remota.

Il motivo principale è la **totale mancanza di crittografia**. Qualsiasi dato trasmesso durante una sessione Telnet, incluse username e password, viaggia in chiaro sulla rete. Chiunque sia in grado di intercettare il traffico può leggere tutte le informazioni scambiate. Per questo motivo, il suo utilizzo sicuro è limitato alla diagnostica locale (come in questo laboratorio) o su reti fidate e isolate. Per l'amministrazione remota, è stato completamente soppiantato da **SSH (Secure Shell)**, che cifra tutto il traffico, garantendo confidenzialità e integrità.

---

4. Conclusioni

Questa attività di laboratorio ha fornito una solida base metodologica per l'analisi dei servizi su un sistema Linux. L'approccio adottato ha seguito un workflow logico, passando da una visione ampia a una verifica mirata. Abbiamo iniziato con `ps` per ottenere un inventario completo dei processi a livello di sistema, stabilendo una baseline generale. Successivamente, abbiamo ristretto il campo con `netstat` per ispezionare specificamente i servizi di rete, mappando le porte in ascolto ai processi che le gestiscono e comprendendo l'architettura di sicurezza di Nginx basata sulla separazione dei privilegi. Infine, abbiamo utilizzato `telnet` per una verifica dinamica e interattiva, confermando l'identità dei servizi TCP tramite "banner grabbing" e test diretti. Questo percorso analitico, che va dal generale al particolare, dimostra la complementarietà degli strumenti e costituisce un modello efficace per future attività di troubleshooting e analisi di sicurezza.