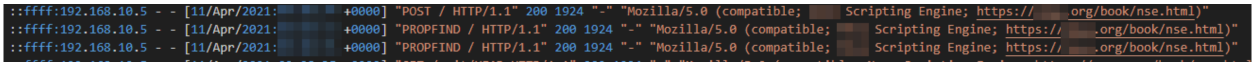
Juicy Details

Overview

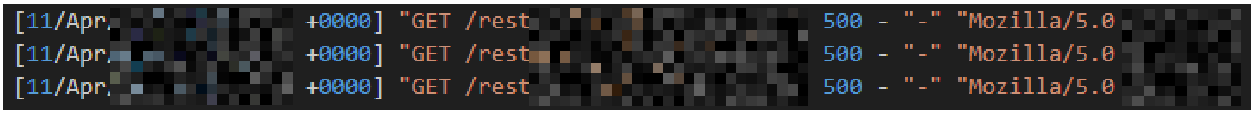
As part of a simulated Security Operations Center (SOC) engagement, I investigated a breach of a fictional global Juice Shop's network. My role was to perform forensic log analysis to identify the attacker's methods, the compromised endpoints, and the nature of the data exfiltrated. I was provided with a compressed archive containing access.log, auth.log, and vsftpd.log to conduct my analysis.

Reconnaissance and Tool Identification

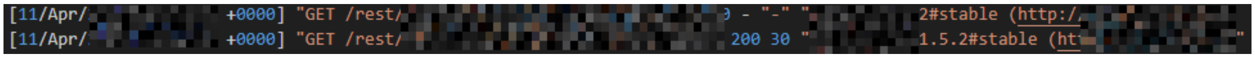
I began by analyzing the access.log file to identify potential attacker tools and behaviors. The tools used were easy to recognize, as their names appeared at the end of the request lines in the logs. By examining user-agent strings and command patterns, I observed several tools being executed in quick succession—some repeatedly and one subtly included toward the end—suggesting the use of automated reconnaissance and exploitation frameworks.



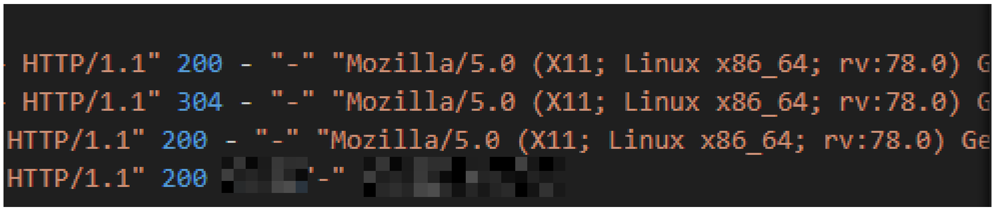
```
::ffff:192.168.10.5 - - [11/Apr/2021:[redacted] +0000] "POST / HTTP/1.1" 200 1924 "-" "Mozilla/5.0 (compatible; [redacted] Scripting Engine; https://[redacted].org/book/nse.html)"
::ffff:192.168.10.5 - - [11/Apr/2021:[redacted] +0000] "PROPFIND / HTTP/1.1" 200 1924 "-" "Mozilla/5.0 (compatible; [redacted] Scripting Engine; https://[redacted].org/book/nse.html)"
::ffff:192.168.10.5 - - [11/Apr/2021:[redacted] +0000] "PROPFIND / HTTP/1.1" 200 1924 "-" "Mozilla/5.0 (compatible; [redacted] Scripting Engine; https://[redacted].org/book/nse.html)"
```



```
[11/Apr[redacted] +0000] "GET /rest[redacted] 500 - "-" "Mozilla/5.0 [redacted]
[11/Apr[redacted] +0000] "GET /rest[redacted] 500 - "-" "Mozilla/5.0 [redacted]
[11/Apr[redacted] +0000] "GET /rest[redacted] 500 - "-" "Mozilla/5.0 [redacted]
```



```
[11/Apr/[redacted] +0000] "GET /rest/[redacted] - "-" "[redacted]2#stable (http:/[redacted]
[11/Apr/[redacted] +0000] "GET /rest/[redacted] 200 30 "[redacted]1.5.2#stable (ht[redacted]"
```

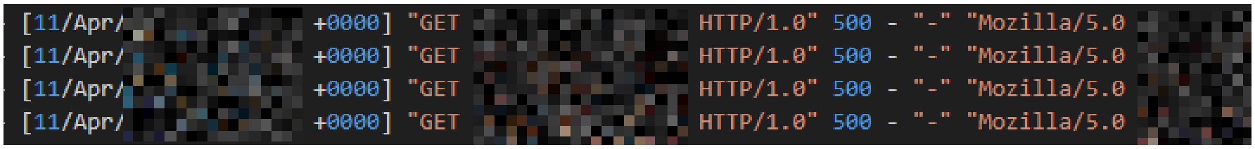


```
 HTTP/1.1" 200 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) G
 HTTP/1.1" 304 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) G
HTTP/1.1" 200 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Ge
HTTP/1.1" 200 [redacted]'-" [redacted]
```

```
[11/Apr/[redacted] +0000] "GET /[redacted]200 1924 "-" [redacted]"
[11/Apr/[redacted] +0000] "GET /[redacted]200 1924 "-" [redacted]
[11/Apr/[redacted] +0000] "GET /[redacted] 200 1924 "-"[redacted]"
```

Brute-Force Login Detection

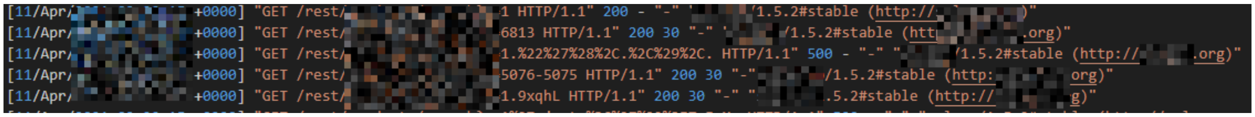
Next, I focused on identifying endpoints targeted for brute-force attacks. A high volume of automated GET and POST requests to a specific REST API login endpoint revealed an apparent brute-force attempt. The successful login was confirmed by a singular HTTP 200 OK response amidst otherwise failed login attempts — a key indicator of credential compromise.



```
[11/Apr/[redacted] +0000] "GET [redacted] HTTP/1.0" 500 - "-" "Mozilla/5.0 [redacted]
[11/Apr/[redacted] +0000] "GET [redacted] HTTP/1.0" 500 - "-" "Mozilla/5.0 [redacted]
[11/Apr/[redacted] +0000] "GET [redacted] HTTP/1.0" 500 - "-" "Mozilla/5.0 [redacted]
[11/Apr/[redacted] +0000] "GET [redacted] HTTP/1.0" 500 - "-" "Mozilla/5.0 [redacted]
```

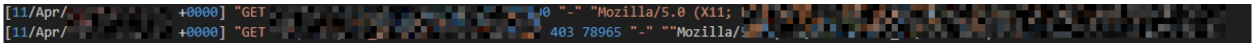
SQL Injection Discovery

Following the brute-force attack, I observed a series of suspicious queries and URL patterns, revealing a SQL injection (SQLi) attack. The frequency and structure of requests pointed to automated exploitation. I was able to identify both the vulnerable endpoint and the parameter manipulated by the attacker.



```
[11/Apr/[redacted] +0000] "GET /rest/[redacted]1 HTTP/1.1" 200 - "-" "[redacted]/1.5.2#stable (http://[redacted])"
[11/Apr/[redacted] +0000] "GET /rest/[redacted]6813 HTTP/1.1" 200 30 "-" "[redacted]/1.5.2#stable (htt[redacted].org)"
[11/Apr/[redacted] +0000] "GET /rest/[redacted]1.%22%27%28%2C.%2C%29%2C. HTTP/1.1" 500 - "-" "[redacted]/1.5.2#stable (http://[redacted].org)"
[11/Apr/[redacted] +0000] "GET /rest/[redacted]5076-5075 HTTP/1.1" 200 30 "-"[redacted]/1.5.2#stable (http:[redacted]org)"
[11/Apr/[redacted] +0000] "GET /rest/[redacted]1.9xqhL HTTP/1.1" 200 30 "-" "[redacted].5.2#stable (http://[redacted]g)"
```

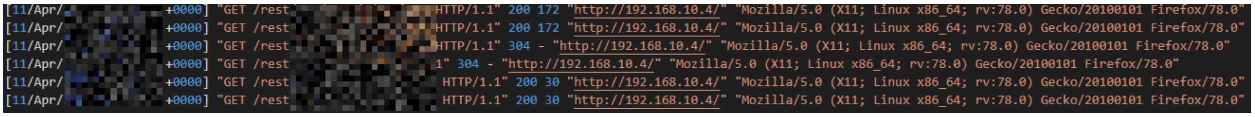
After identifying the SQL injection entry point, I proceeded to determine which endpoint was used for file retrieval. The relevant activity was visible in the logs shortly after the injection attempts, indicating a direct progression in the attack chain.



```
[11/Apr/[redacted] +0000] "GET [redacted]0 "-" "Mozilla/5.0 (X11; [redacted]
[11/Apr/[redacted] +0000] "GET [redacted] 403 78965 "-" ""Mozilla/[redacted]
```

Email Scraping Behavior

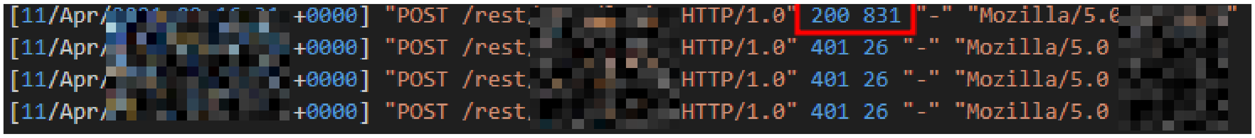
With the reconnaissance phase complete, I shifted focus to verifying key questions related to the compromised data. One objective was to determine where the attacker attempted to collect user email addresses. The logs revealed a clear pattern of multiple automated requests across a specific area of the site, strongly indicating scraping activity.



```
[11/Apr/[redacted] +0000] "GET /rest[redacted]HTTP/1.1" 200 172 "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET /rest[redacted]HTTP/1.1" 200 172 "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET /rest[redacted]HTTP/1.1" 304 - "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET /rest[redacted]1" 304 - "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET /rest[redacted] HTTP/1.1" 200 30 "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET /rest[redacted] HTTP/1.1" 200 30 "http://192.168.10.4/" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Brute-Force Login Timestamp

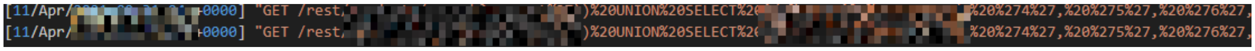
At this stage, I was asked to confirm whether the brute-force attempt succeeded and to identify the corresponding timestamp. A single successful login response stood out in the logs, clearly indicating that the attack had achieved access.



```
[11/Apr/[redacted] +0000] "POST /rest[redacted] HTTP/1.0" 200 831 "-" "Mozilla/5.[redacted]"
[11/Apr/[redacted] +0000] "POST /rest[redacted] HTTP/1.0" 401 26 "-" "Mozilla/5.0 [redacted]
[11/Apr/[redacted] +0000] "POST /rest[redacted] HTTP/1.0" 401 26 "-" "Mozilla/5.0 [redacted]
[11/Apr/[redacted] +0000] "POST /rest[redacted] HTTP/1.0" 401 26 "-" "Mozilla/5.0 [redacted]
```

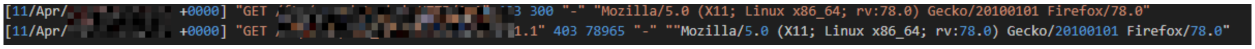
File Retrieval Activity

I was then tasked with identifying the type of information the attacker was able to access through the SQL injection vulnerability. The relevant data was exposed in the same section of the logs, confirming the impact of the exploitation.

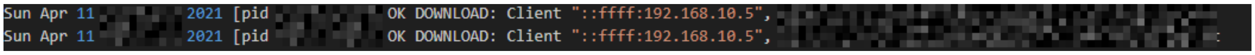


```
[11/Apr/[redacted]+0000] "GET /rest[redacted])%20UNION%20SELECT%2[redacted]%20%274%27,%20%275%27,%20%276%27,
[11/Apr/[redacted]+0000] "GET /rest[redacted])%20UNION%20SELECT%2[redacted]%20%274%27,%20%275%27,%20%276%27,
```

The files accessed and downloaded by the attacker were identifiable through entries in both the access.log and vsftpd.log, providing strong evidence of data exfiltration.

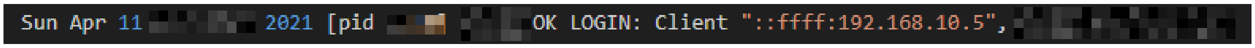


```
[11/Apr/[redacted] +0000] "GET [redacted]3 300 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[11/Apr/[redacted] +0000] "GET [redacted]1.1" 403 78965 "-" ""Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```



```
Sun Apr 11 [redacted] 2021 [pid [redacted] OK DOWNLOAD: Client "::ffff:192.168.10.5", [redacted]
Sun Apr 11 [redacted] 2021 [pid [redacted] OK DOWNLOAD: Client "::ffff:192.168.10.5", [redacted]
```

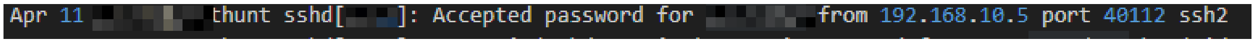
I was asked to identify the service and account used to retrieve the downloaded files. This information was confirmed through entries in the vsftpd.log, which documented the relevant login activity and file transfers.



```
Sun Apr 11 [redacted] 2021 [pid [redacted] OK LOGIN: Client "::ffff:192.168.10.5", [redacted]
```

Shell Access Confirmation

Another objective was to determine the service and username used to gain shell access to the server. This was identified by analyzing the auth.log file, which captured the successful authentication event and session activity.



```
Apr 11 [redacted] thunt sshd[[redacted]]: Accepted password for [redacted] from 192.168.10.5 port 40112 ssh2
```

Conclusion

Through comprehensive log analysis, I successfully identified the attacker’s methods, compromised endpoints, and the extent of data exposure. This investigation demonstrated my ability to extract meaningful insights from system logs, trace attacker activity, and answer key incident response questions—effectively completing the objectives outlined in the scenario.