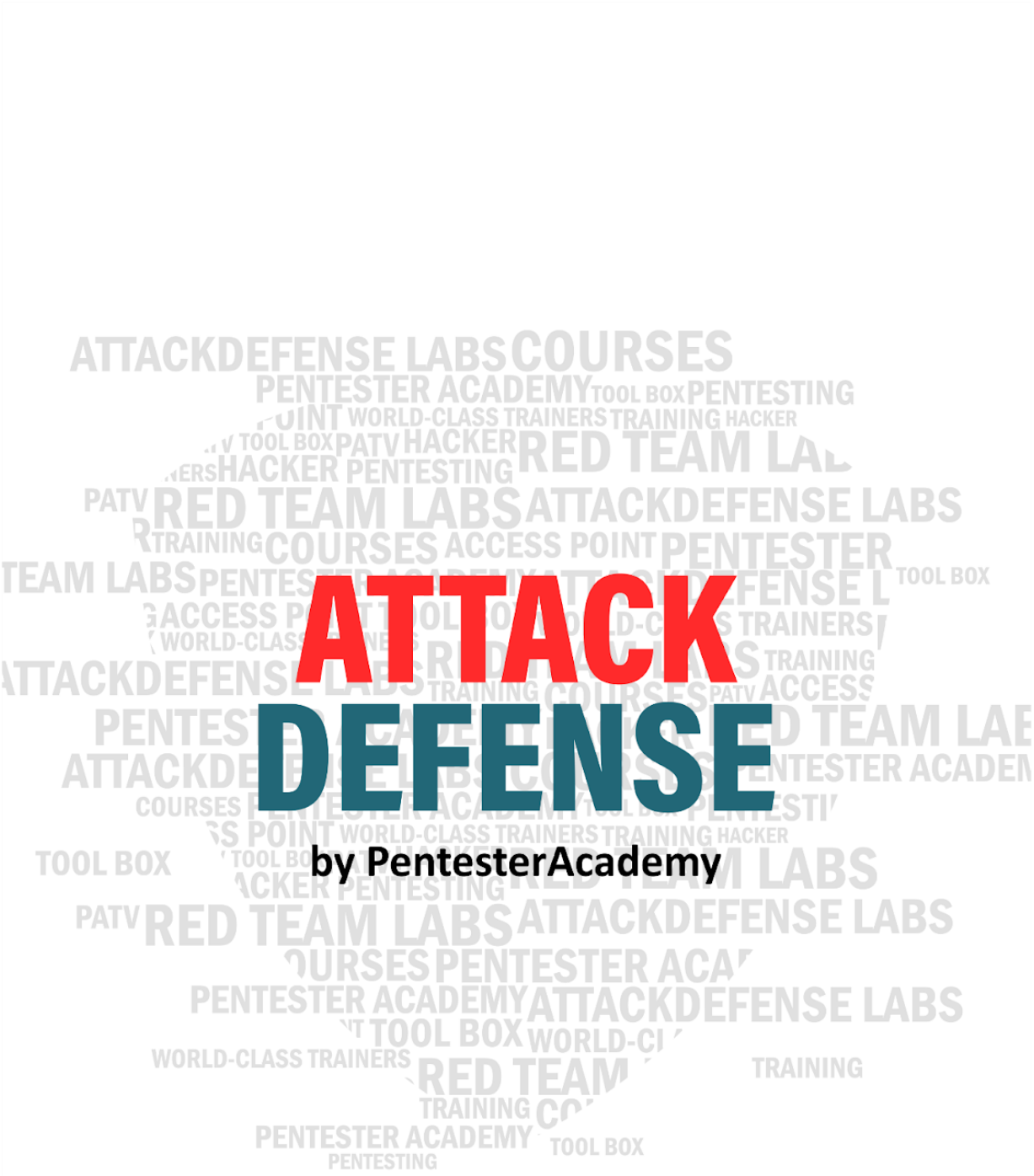
ATTACK
DEFENSE
by PentesterAcademy

Name	Angle Grinder : Apache Log Analysis
URL	https://attackdefense.com/challengedetails?cid=1183
Type	Log Analysis : Other Tools

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Q1. SQLmap was used to perform SQL injection attack against a particular webpage. Find the relative path of the vulnerable webpage.

Answer: /index.php

Solution:

Step 1: Use the following command to process only those log entries which contain sqlmap in the useragent field and extract the request, the status code and the request count.

Command: agrind -f access.json ' * | json | parse "sqlmap*" from USERAGENT as ua | count as REQ_COUNT by REQUEST, STATUS '

```
REQUEST                                                                    STATUS        REQ_COUNT
-------------------------------------------------------------------------------------------------
POST /index.php?page=login.php HTTP/1.1                                    200           32972
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20ORDER%20BY%201%23 HTTP/1.1200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20N… 200          4
POST /index.php?page=login.php%27%29%20UNION%20ALL%20SELECT%… 200          3
POST /index.php?page=login.php%20UNION%20ALL%20SELECT%20NULL… 200          3
POST /index.php?page=login.php%20UNION%20ALL%20SELECT%20NULL… 200          3
POST /index.php?page=login.php%22%20UNION%20ALL%20SELECT%20N… 200          3
POST /index.php?page=login.php%27%29%20UNION%20ALL%20SELECT%… 200          3
POST /index.php?page=login.php%20UNION%20ALL%20SELECT%20NULL… 200          3
POST /index.php?page=login.php%27%29%20UNION%20ALL%20SELECT%… 200          3
POST /index.php?page=login.php%20UNION%20ALL%20SELECT%20NULL… 200          3
```

Step 2: Use the following command to get the distinct request pages and their count from the requests where the user-agent was sqlmap.

Command: agrind -f access.json ' * | json | parse "sqlmap*" from USERAGENT as ua | parse "*?" from REQUEST as path | count as REQ_COUNT by path ';

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "sqlmap*" from USERAGENT as ua | parse "*?" fr
om REQUEST as path | count as REQ_COUNT by path ';
path                  REQ_COUNT
-------------------------------------
POST /index.php       42422
root@attackdefense:~#
```

The returned output indicates that /index.php page was attacked using sqlmap.

Q2. The most active user of the web application was suspected to be attacking it. Verify if that was actually the case. Also find out the IP address of that user.

Answer: 192.131.71.7

Solution:

Step 1: Use the following command to get the IP address of the most active client.

Command: agrind -f access.json ' * | json | count as REQ_COUNT by HOST'

```
root@attackdefense:~# agrind -f access.json ' * | json | count as REQ_COUNT by HOST'
HOST               REQ_COUNT
-----------------------------------
192.131.71.7       43399
192.131.71.6       17583
192.131.71.2       14658
192.131.71.5       6521
127.0.0.1          368
172.17.0.1         5
root@attackdefense:~#
```

The machine at IP address "192.131.71.7" was the most active client.

Step 2: Use the following command to get the requests issued by the most active client.

Command: agrind -f access.json ' * | json | where HOST=="192.131.71.7" | count as REQ_COUNT by REQUEST '

```
REQUEST                                                                                       REQ_COUNT
--------------------------------------------------------------------------------------------------------
POST /index.php?page=login.php HTTP/1.1                                                       33946
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%23 HTTP/1.1            4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20ORDER%20BY%201%23 HTTP/1.1                                4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%23 HTTP… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%23 HTTP/1.1     4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 4
POST /index.php?page=login.php%27%20UNION%20ALL%20SELECT%20NULL%23 HTTP/1.1                   4
POST /index.php?page=login.php%22%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%… 3
```

The above results show that the most active client, having IP address "192.131.71.7" was attacking the web application.

Q3. Which attacker machine was able to find out the existence of directory '/btslab/tmp/'?

Answer: 192.131.71.5

Solution:

Step 1: Use the following command to retrieve the IP address, the request and the status code of the client that was able to locate "/btslab/tmp/" directory.

Command: agrind -f access.json ' * | json | parse "/btslab/tmp/*" from REQUEST as r | count as ACCESS_COUNT by HOST, REQUEST, STATUS '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "/btslab/tmp/*" from REQUEST as r | count as ACCESS_COUNT by HOST, REQUEST, STATUS  '
HOST               REQUEST                          STATUS        ACCESS_COUNT
-------------------------------------------------------------------------------
192.131.71.5       HEAD /btslab/tmp/ HTTP/1.1       200           1
root@attackdefense:~#
```

The client at IP address "192.131.71.5" was able to find out the existence of the "/btslab/tmp/" directory.

Step 2: Use the following command to retrieve the client IP addresses and count of the requests that resulted in the status code of 404.

Command: agrind -f access.json ' * | json | where STATUS == "404" | count by HOST '

```
root@attackdefense:~# agrind -f access.json ' * | json | where STATUS == "404" | count by HOST '
HOST               _count
----------------------------------
192.131.71.6       17181
192.131.71.5       1546
192.131.71.2       703
root@attackdefense:~#
```

The client at IP address "192.131.71.5" received 404 Not Found response 1546 times, which indicates that it could be a possible attacker machine.

Step 3: Use the following command to retrieve the timestamp and the request from the logs where the client IP address was "192.131.71.5".

Command: agrind -f access.json ' * | json | where HOST == "192.131.71.5" | count by REQUEST, TIME | sort by TIME '

REQUEST	TIME	_count
GET /staff HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /administration HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /iisadmin HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /_mmDBScripts/MMHTTPDB.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /administrator HTTP/1.1	04/Aug/2019:18:19:54 +0000	2
GET /manager HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /r57eng.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /%3Cmy_tag_ddc5dc63e2371d6df9695175659fb79b/%3E HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /WEB-INF%20-%20Copy/web.xml HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin.cfm HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin-login HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /%3E%22'%3E%3Cmy_tag_ddc5dc63e2371d6df9695175659fb79b/%3…	04/Aug/2019:18:19:54 +0000	1
GET /admin HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /r57.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /nstview.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin.aspx HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /adminlogon HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin.cgi HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /install.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /rst.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /backend HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /config/database.yml HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /usuario HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /nst.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin_logon HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /.adm HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /users HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /c99shell.php HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /localstart.asp HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /ops HTTP/1.1	04/Aug/2019:18:19:54 +0000	1
GET /admin.jsp HTTP/1.1	04/Aug/2019:18:19:54 +0000	1

The client at IP address "192.131.71.5" issued a huge number of suspicious requests in a short period of time.

This indicates that the client at "192.131.71.5" was trying to attack the web application and it was also able to find the existence of "/btslab/tmp/" directory.

Q4. How many union based SQL injection attacks were attempted on the web application?

Answer: 2138

Solution:

Step 1: Use the following command to retrieve the count of all the log instances where the request query contains "union" keyword.

Command: agrind -f access.json ' * | json | parse "?*union" from REQUEST as union_query | count '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "?*union" from REQUEST as union_query | count '
_count
--------------
2138
root@attackdefense:~#
```

There were 2138 log instances where union based SQL injection attack was attempted.

Q5. How many of the requests resulted in an internal server error?

Answer: 0

Solution:

Step 1: Use the following command to retrieve the count of all the log instances where the request status was 500, i.e, internal server error.

Command: agrind -f access.json ' * | json | where STATUS=="500" | count '

```
root@attackdefense:~# agrind -f access.json ' * | json | where STATUS=="500" | count '
No data
root@attackdefense:~#
```

"No data" means that there were no records having the status code as 500.

Q6. An HTTP GET request was made to a web page to download a PDF file. The same web page was vulnerable to PATH traversal attack and was exploited to download /etc/passwd file. Find the approximate size of /etc/passwd file in KB.

Answer: 1

Solution:

Step 1: Use the following command to retrieve the distinct request paths which were queried to download PDF files.

Command: agrind -f access.json ' * | json | parse "?*.pdf" from REQUEST as p | parse "*?" from REQUEST as page | count by page '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "?*.pdf" from REQUEST as
 p | parse "*?" from REQUEST as page | count by page '
page                                                    _count
------------------------------------------------------------
GET /btslab/vulnerability/dor/download.php              146
GET /cgi-bin/c32web.exe/GetImage                        1
root@attackdefense:~#
```

A large number of requests to download PDF files were received by the page located at "/btslab/vulnerability/dor/download.php".

Also, run the following command to get the number of occurrences of the page "/cgi-bin/c32web.exe/GetImage" in other requests:

Command: agrind -f access.json ' * | json | parse " *?" from REQUEST as page | where page == "/cgi-bin/c32web.exe/GetImage" | count '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse " *?" from REQUEST as
page | where page == "/cgi-bin/c32web.exe/GetImage" | count ';
_counta
--------------
1
root@attackdefense:~#
```

The results show that the page at "/cgi-bin/c32web.exe/GetImage" has been used only once, and that request was to download the PDF file that was observed in the previous output.

Step 2: Use the following command to retrieve the response size and the count of the requests having that response size, for the page at "/btslab/vulnerability/dor/download.php".

Command: agrind -f access.json ' * | json | parse " *?" from REQUEST as page | where page == "/btslab/vulnerability/dor/download.php" | count by SIZE | sort by SIZE '

SIZE	_count
1050	2
1054	1
1057	2
1060	1
1061	1
1063	1
1064	1
1067	1
11277	6
176	51
195	1
219	1
220	1
221	7
223	1
225	8
227	3
228	3
229	10
230	6
231	1
232	8
2320	1
233	2
234	1
235	7
236	9
237	4
238	7
239	4
240	10

The results indicate that in maximum number of requests, the size of "/btslab/vulnerability/dor/download.php" page was 176 bytes.

Step 3: Use the following command to retrieve the query and response size for the logs having the request path as "/btslab/vulnerability/dor/download.php" and the query containing "/etc/passwd".

Command: agrind -f access.json ' * | json | parse " *?" from REQUEST as page | where page == "/btslab/vulnerability/dor/download.php" | parse "?*" as query | parse "*passwd" from query as p | parse "*etc" from p as e | where STATUS=="200" | count by query, SIZE | sort by SIZE '

```
query                                                                          SIZE        _count
-------------------------------------------------------------------------------------------------
file=%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.131.71.5", "REF… 1050        1
file=%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.131.71.2", "REF… 1050        1
file=%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.131.71.… 1054        1
file=%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.13… 1057        1
file=file%3A%2F%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.13… 1057        1
file=%2F..%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST": "1… 1060        1
file=file%3A%2F%2F%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST":… 1061        1
file=%2F..%2F..%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "HOST… 1063        1
file=file%3A%2F%2F%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1", "H… 1064        1
file=file%3A%2F%2F%2F..%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1… 1067        1
file=%2F..%2F..%2F..%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1"… 176         1
file=%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1", "HOST": "192.… 176         1
file=file%3A%2F%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1", "HOST": … 176         1
file=file%3A%2F%2F%2F..%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1… 176         1
file=file%3A%2F%2F%2F..%2F..%2F..%2F%2Fetc%2Fpasswd%00.pdf H… 176         1
file=%2Fetc%2Fpasswd%00.pdf HTTP/1.1", "HOST": "192.131.71.5… 176         1
file=%2F..%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1", "HOST": … 176         1
file=file%3A%2F%2F%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1", … 176         1
file=%2F..%2F..%2F..%2F%2Fetc%2Fpasswd%00.pdf HTTP/1.1", "HO… 176         1
file=%2Fbin%2Fcat%20%2Fetc%2Fpasswd HTTP/1.1", "HOST": "192.… 240         1
file=%60%20%2Fbin%2Fcat%20%2Fetc%2Fpasswd%60 HTTP/1.1", "HOS… 243         1
file=%20%3B%20%2Fbin%2Fcat%20%2Fetc%2Fpasswd%20%3B%20 HTTP/1… 245         1
```

Note: The seemingly complex query extracts the requested page in the field named "page", and the request query in the field named "query". After that, the "query" column is again parsed to check if it contains a pattern like "etc*passwd". To perform that operation, 2 passes are required. One to check the occurrence of "passwd" and one to check that it is preceded by "etc".

The above results show that the response size varies between 1050 to 1067 bytes when /etc/passwd file was downloaded. Also, as it was observed before that the download.php page had the size around 176 bytes or slightly more (depending upon the number of characters passed in the GET request).

So, the approximate size of "/etc/passwd" was 1 KB.

Q7. Which attacker machine had tried to perform the most command injection attempts using the 'exec' payload?

Answer: 192.131.71.2

Solution:

Step 1: Use the following command to retrieve client IP addresses and the request query from the logs which contain "exec" keyword.

Command: agrind -f access.json ' * | json | parse "?*" from REQUEST as query | parse "*exec" from query as exec_query | count by HOST, query '

HOST	query	_count
192.131.71.2	query=%22%3E%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E%3C HTTP/1.1	72
192.131.71.2	query=%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C HTTP/1.1	72
192.131.71.2	query=%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E HTTP/1.1	72
192.131.71.2	query=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	72
192.131.71.2	id=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	4
192.131.71.2	id=%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C HTTP/1.1	4
192.131.71.2	id=%22%3E%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E%3C HTTP/1.1	4
192.131.71.2	id=%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E HTTP/1.1	4
192.131.71.2	u=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	3
192.131.71.2	Search=Search&keyword=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	3
192.131.71.2	file=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	3
192.131.71.2	Search=Search&keyword=%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C HTTP/1.1	3
192.131.71.2	data=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E HTTP/1.1	3
192.131.71.2	data=%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E HTTP/1.1	3
192.131.71.2	Search=%22%3E%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E%3C&keyword=Search HTTP/1.1	3
192.131.71.2	Search=%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C&keyword=Search HTTP/1.1	3
192.131.71.2	Search=%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E&keyword=Search HTTP/1.1	3
192.131.71.2	Search=%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E&keyword=Search HTTP/1.1	3
192.131.71.2	Search=Search&keyword=%22%3E%3C%21--%23EXEC+cmd%3D%22ls+%2F%22--%3E%3C HTTP/1.1	3
192.131.71.2	file=%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E HTTP/1.1	3
192.131.71.2	u=%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C HTTP/1.1	3

Step 2: Use the following command to get the list of client IP addresses and the count of the log instances in which the request sent by the client contains "exec" keyword.

Command: agrind -f access.json ' * | json | parse "?*" from REQUEST as query | parse "*exec" from query as exec_query | count by HOST '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "?*" from REQUEST as query | parse "*exec" from query as exec_query | count by HOST '
HOST              _count
---------------------------------
192.131.71.2      404
192.131.71.6      5
192.131.71.7      1
root@attackdefense:~#
```

The client at IP address "192.131.71.2" tried to perform the most command injection attempts using the "exec" payload.

Q8. How many onerror based XSS attacks were attempted on the web application?

Answer: 1

Solution:

Step 1: Use the following command to get the number of log instances where the request contains "onerror" keyword.

Command: agrind -f access.json ' * | json | parse "?*onerror" from REQUEST as onerror_query | count '

```
root@attackdefense:~# agrind -f access.json ' * | json | parse "?*onerror" from REQUEST as onerror_query | count '
_counta
--------------
1
root@attackdefense:~#
```

There was only one log instance where onerror based XSS attack was attempted.

References:

1. Angle Grinder (https://github.com/rcoh/angle-grinder)