# Lab 4 — Web Attack Forensics

Web applications are an integral part of our daily lives and are used for a wide range of activities, from online shopping to banking and social media. However, their widespread use opens up a large attack surface for bad actors to exploit and gain initial foothold into the system.

In this lab, we will learn about the different types of attacks that are common against web applications, and explore various techniques used to detect these attacks by analyzing web application logs and web application firewall logs, to find the point of attack, and trace the root cause by identifying the vulnerability that was exploited.

## Environment Setup

Before delving into the topic of web attacks and forensics, let's establish an environment locally using Docker. This will allow for a more hands-on and comprehensive understanding of the material covered.

### Clone the repository

```
git clone https://github.com/vonderchild/digital-forensics-lab && cd digital-forensics-la
b/Lab\ 4/app/
```

### Install Docker

```
sudo apt-get update
sudo apt-get install -y docker.io
sudo service docker start
```

### Build & Run Docker Image

```
docker build -t app:latest .
docker run -p 9090:80 app:latest
```

# Web Attacks & Forensics

## Web Application & WAF Logs

Web application logs play an important role in digital forensics as they help track user activity, detecting potential attacks, trace the origin of an attack and determine the extent of its impact. In this lab, we'll be focusing on Apache, a widely used web server for hosting web applications. The logs generated by Apache contain access logs and error logs. Access logs contain information about incoming requests such as the client's IP address, date and time of the request, the request method (e.g. GET, POST), the requested URI, the response status code (e.g. 200, 403, 404), and the user agent. Error logs, on the other hand, contain information about errors encountered by the server, such as failed requests and unexpected events that occurred during the processing of the request. These logs can be found under `/var/log/apache2` on Linux systems.

Web application firewalls (WAFs) are an important aspect of web application security. A WAF provides a security layer for the application by blocking malicious traffic before it reaches the application. In this lab, we'll be utilizing Modsecurity as our web application firewall. The default location for its audit logs is `/var/log/apache2/modsec_audit.log`. When an error or any malicious attempt is encountered on the server, it is also logged inside `/var/log/apache2/error.log`.

The directory structure for these logs looks like as following:

```
var
└── log
    └── apache2
        ├── access.log
        ├── error.log
        ├── modsec_audit.log
        └── other_vhosts_access.log
```

## Common Web Attacks & Logs

To effectively conduct web attack forensics, it is important to have an understanding of the common types of web attacks. These attacks exploit vulnerabilities in web applications, and knowing how they work is essential in piecing together the events that led to the attack.

There are numerous vulnerabilities that can be exploited in a web application that can range from vulnerabilities that have no real impact to critical vulnerabilities that can have an enormous impact on an organization when exploited. However, in this section, we will focus on some of the critical ones, which include Path Traversal, Remote Command Execution, and SQL Injection.

To fully engage in the hands-on experience, ensure you have followed the steps in environment setup. To verify, the home page should be accessible at http://127.0.0.1:9090/.

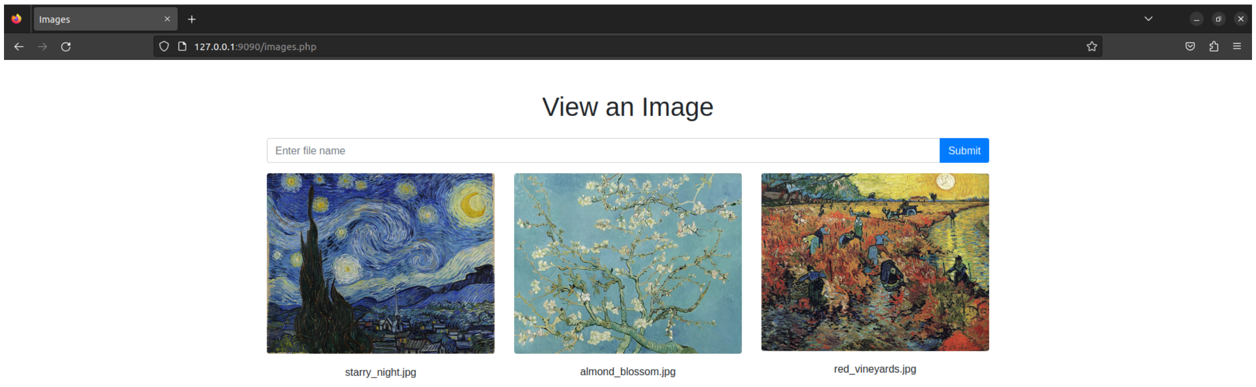
### Path Traversal

Also known as "directory traversal", this vulnerability allows attackers to access files and directories on a server that are outside of the root directory. It is often achieved by manipulating file path input fields in a web application to access files that the application has permission to access, but the attacker should not. This type of attack can result in sensitive information being leaked, such as configuration files and source code.

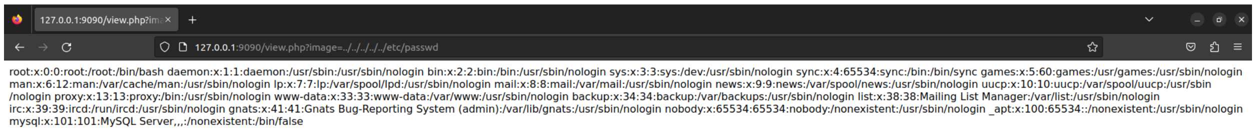
For example, let’s consider that you’re currently in the home directory `/home/kali/`. To change to a parent directory to `/home`, you would enter `cd ../`. To reach the root directory, you would enter `cd ../../`. This same concept is employed in a path traversal attack, where a website allows access to files outside the website’s root directory.

> Note: The term path traversal is often used interchangeably with Local File Inclusion (LFI), however, both are different vulnerabilities; Path traversal is limited to reading files on the server, while Local File Inclusion refers to the additional ability to execute that file on the server.

To try it out, head over to http://127.0.0.1/images.php and it should display some images along with an input field.



If we enter the correct image name, the web application will display it back to us. But what if we try to enter a file name that's not an image, like `/etc/passwd`? To our surprise, it will print out its contents. However, we just need to add some leading `../` before our file name `/etc/passwd`.


127.0.0.1:9090/view.php?ima
127.0.0.1:9090/view.php?image=../../../../../etc/passwd

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin
/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

Now that we are familiar with the method of exploiting this vulnerability, let's proceed to learn how to detect it in our logs. In order to access the logs, we must first take shell inside the docker container where our application is running. We do that by first listing the container ID using `docker ps -q` and then executing `docker exec` to spawn a shell:

```
$ docker ps -q
18c7468edbe7

$ docker exec -it 18c7468edbe7 bash
root@18c7468edbe7:/#
```

Now, let's go to the directory containing the access logs and print them:

```
root@18c7468edbe7:/# cd /var/log/apache2/
root@18c7468edbe7:/var/log/apache2# cat access.log
172.17.0.1 - - [14/Feb/2023:04:09:47 +0500] "GET /images.php HTTP/1.1" 200 1114 "-" "Mozil
la/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:04:09:53 +0500] "GET /images.php?file=starry_night.jpg HTTP/1.
1" 302 2423 "http://127.0.0.1:9090/images.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; r
v:109.0) Gecko/20100101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:04:09:53 +0500] "GET /view.php?image=starry_night.jpg HTTP/1.
1" 200 613774 "http://127.0.0.1:9090/images.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
 rv:109.0) Gecko/20100101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:04:10:02 +0500] "GET /images.php?file=..%2F..%2F..%2F..%2F..%2
Fetc%2Fpasswd HTTP/1.1" 302 2432 "http://127.0.0.1:9090/images.php" "Mozilla/5.0 (X11; Ubu
ntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:04:10:02 +0500] "GET /view.php?image=../../../../../etc/passwd
HTTP/1.1" 200 650 "http://127.0.0.1:9090/images.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_
64; rv:109.0) Gecko/20100101 Firefox/109.0"
```

As it can be seen, the logs show requests made to the server from IP address `172.17.0.1` using the Firefox browser to `/images.php` and `/view.php`. The last two logs show our attempts to exploit the path traversal vulnerability by accessing the `/etc/passwd` file.

As a next step, let's examine the logs generated by Modsecurity WAF:

```
root@18c7468edbe7:/var/log/apache2# cat modsec_audit.log
<SNIP>
--412fc70c-A--
[14/Feb/2023:04:10:02.199419 +0500] Y-rDSoUvR2SaOiwbiHSndQAAAAI 172.17.0.1 35308 172.17.0.
2 80
--412fc70c-B--
GET /view.php?image=../../../../../etc/passwd HTTP/1.1
Host: 127.0.0.1:9090
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://127.0.0.1:9090/images.php
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

--412fc70c-F--
HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 399
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

--412fc70c-E--
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false

--412fc70c-H--

<SNIP>

Message: Warning. Pattern match "(?i)(?:\\x5c|(?:%(?:c(?:0%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8s
|af))|2(?:5(?:c(?:0%25af|1%259c)|2f|5c)|%46|f)|(?:(?:f(?:8%8)?0%8|e)0%80%a|bg%q)f|%3(?:2
(?:%(?:%6|4)6|F)|5%%63)|u(?:221[56]|002f|EFC8|F025)|1u|5c)|0x(?:2f|5c)|\\/))(?:%(?:(?:f(?:
(?:c%80|8)%8)?0%8 ..." at REQUEST_URI_RAW. [file "/usr/share/modsecurity-crs/rules/REQUEST
-930-APPLICATION-ATTACK-LFI.conf"] [line "47"] [id "930100"] [msg "Path Traversal Attack
 (/../)"] [data "Matched Data: /../ found within REQUEST_URI_RAW: /view.php?image
=../../../../../etc/passwd"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "applicati
on-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia
-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/255/153/126"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Pattern match "(?:^|[\\\\\\\\/])\\\\\\\\.\\\\\\\\.(?:[\\\\\\\\/]|$)" at REQUES
T_URI. [file "/usr/share/modsecurity-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"]
 [line "71"] [id "930110"] [msg "Path Traversal Attack (/../)"] [data "Matched Data: /../
 found within REQUEST_URI: /view.php?image=../../../../../etc/passwd"] [severity "CRITICA
L"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "language-multi"] [tag "platfor
m-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/2
55/153/126"] [hostname "127.0.0.1"] [uri "/view.php"] [unique_id "Y-rDSoUvR2SaOiwbiHSndQAA
AAI"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecuri
ty-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anoma
ly Score Exceeded (Total Inbound Score: 43 - SQLI=0,XSS=0,RFI=0,LFI=35,RCE=5,PHPI=0,HTTP=
0,SESS=0): individual paranoia level scores: 43, 0, 0, 0"] [ver "OWASP_CRS/3.3.2"] [tag "e
vent-correlation"] [hostname "127.0.0.1"] [uri "/view.php"] [unique_id "Y-rDSoUvR2SaOiwbiH
SndQAAAAI"]
Apache-Handler: application/x-httpd-php
Stopwatch: 1676329802195680 3808 (- - -)
Stopwatch2: 1676329802195680 3808; combined=2356, p1=492, p2=1461, p3=41, p4=177, p5=185,
 sr=74, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.5 (http://www.modsecurity.org/); OWASP_CRS/3.3.2.
Server: Apache/2.4.52 (Ubuntu)
Engine-Mode: "DETECTION_ONLY"

--412fc70c-Z--
```

As seen, this displays a rather detailed audit of access logs containing the request as well as the response body, potentially indicating any malicious activities.

The first message indicates a warning of a pattern match related to Path Traversal. It highlights that a malicious string matching the pattern was found in the `REQUEST_URI_RAW` and it is a critical issue.

Additionally, two errors of type `Apache-Error` are displayed, one is another warning of a Path Traversal attack, but this time a slightly different pattern is matched. The other `Apache-Error` is a warning of an inbound anomaly score exceeding the set threshold, indicating a potential attack. The total inbound score, type of attacks, and paranoia level scores are all logged, providing insight into the threat level.

> Note: To make sense of log data in a forensics investigation, filtering through irrelevant details and retaining only the essential data is crucial. This is why I snipped some information from the above logs since it was just redundant information and only presented what's important to us.

## Remote Command Execution

In a RCE (Remote Command Execution) attack, an attacker is able to execute malicious commands on the server, similar to executing commands in the terminal. Under certain circumstances, this can also be referred to as a Command Injection vulnerability.

For example, if a website allows users to enter a command to search for files, an attacker could enter a command to delete all files on the server by injecting additional commands such as `; rm -rf /` into the input field. This could potentially compromise the entire system if the website has the necessary permissions to execute those commands.

> Note: The term RCE is used interchangeably between Remote Command Execution and Remote Code Execution, but, both refer to a vulnerability that allows an attacker to execute code and/or commands remotely.

To try it out, head over to http://127.0.0.1:9090/command.php. The web app allows us to enter commands, executes them, and returns the output. In absence of proper security checks, this can be exploited to compromise and wipe the whole system.

To generate logs on the server, let's execute the following commands in order:

1. `id`
2. `cat /etc/passwd`
3. `cat /etc/shadow`

Let’s now inspect the access logs:

```
root@18c7468edbe7:/var/log/apache2# cat access.log
172.17.0.1 - - [14/Feb/2023:05:20:59 +0500] "GET /command.php HTTP/1.1" 200 1020 "-" "Mozi
lla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:05:21:06 +0500] "POST /command.php HTTP/1.1" 200 1052 "http://
127.0.0.1:9090/command.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100
101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:05:21:12 +0500] "POST /command.php HTTP/1.1" 200 1395 "http://
127.0.0.1:9090/command.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100
101 Firefox/109.0"
172.17.0.1 - - [14/Feb/2023:05:21:17 +0500] "POST /command.php HTTP/1.1" 200 1020 "http://
127.0.0.1:9090/command.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100
101 Firefox/109.0"
```

As can be seen, the logs above only show incoming requests and don't reveal the commands inputted by the user as those are POST requests. This is where WAF comes into play as it displays both the request and response body. So, in order to see what commands were inputted by the user and how did the web app respond, let's examine the WAF logs.

Viewing logs for the request where command `id` was entered, we can see that it raises almost no red flags since the command is too short to match against any set of patterns.

However, for the command `cat /etc/passwd`, it detects an attempt for LFI (Local File Inclusion) and RCE (Remote Command Execution):

```
<SNIP>

Message: Warning. Matched phrase "etc/passwd" at ARGS:cmd. [file "/usr/share/modsecurity-c
rs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "97"] [id "930120"] [msg "OS File
Access Attempt"] [data "Matched Data: etc/passwd found within ARGS:cmd: cat /etc/passwd"]
 [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "language-mu
lti"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"]
[tag "capec/1000/255/153/126"] [tag "PCI/6.5.4"]
Message: Warning. Matched phrase "etc/passwd" at ARGS:cmd. [file "/usr/share/modsecurity-c
rs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "500"] [id "932160"] [msg "Remote
Command Execution: Unix Shell Code Found"] [data "Matched Data: etc/passwd found within AR
GS:cmd: cat/etc/passwd"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-m
ulti"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-leve
l/1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/88"] [tag "PCI/6.5.2"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Matched phrase "etc/passwd" at ARGS:cmd. [file "/usr/share/modsecurity-crs/rul
es/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "97"] [id "930120"] [msg "OS File Acces
s Attempt"] [data "Matched Data: etc/passwd found within ARGS:cmd: cat /etc/passwd"] [seve
rity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "language-multi"]
 [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag
"capec/1000/255/153/126"] [tag "PCI/6.5.4"] [hostname "127.0.0.1"] [uri "/command.php"] [u
nique_id "Y-rT-PsFa_lKkx8ckcXpMAAAAAQ"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Matched phrase "etc/passwd" at ARGS:cmd. [file "/usr/share/modsecurity-crs/rul
es/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "500"] [id "932160"] [msg "Remote Comma
nd Execution: Unix Shell Code Found"] [data "Matched Data: etc/passwd found within ARGS:cm
d: cat/etc/passwd"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-mult
i"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/
1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/88"] [tag "PCI/6.5.2"] [hostname "127.0.0.
1"] [uri "/command.php"] [unique_id "Y-rT-PsFa_lKkx8ckcXpMAAAAAQ"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecuri
ty-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anoma
ly Score Exceeded (Total Inbound Score: 13 - SQLI=0,XSS=0,RFI=0,LFI=5,RCE=5,PHPI=0,HTTP=0,
SESS=0): individual paranoia level scores: 13, 0, 0, 0"] [ver "OWASP_CRS/3.3.2"] [tag "eve
nt-correlation"] [hostname "127.0.0.1"] [uri "/command.php"] [unique_id "Y-rT-PsFa_lKkx8ck
cXpMAAAAAQ"]

<SNIP>
```

Similarly, for the command `cat /etc/shadow`, we can see that the server does not respond with contents of the file because the server is running as a non-privileged user `www-data` and does not have necessary access to read the `/etc/shadow` file. We can confirm this from error logs:

```
root@18c7468edbe7:/var/log/apache2# cat error.log | grep "Permission denied"
cat: /etc/shadow: Permission denied
```

However, the server still detects it as an attempt for LFI (Local File Inclusion) and RCE (Remote Command Execution):

```
<SNIP>

Message: Warning. Matched phrase "etc/shadow" at ARGS:cmd. [file "/usr/share/modsecurity-c
rs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "97"] [id "930120"] [msg "OS File
Access Attempt"] [data "Matched Data: etc/shadow found within ARGS:cmd: cat /etc/shadow"]
 [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "language-mu
lti"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"]
[tag "capec/1000/255/153/126"] [tag "PCI/6.5.4"]
```

```
Message: Warning. Matched phrase "etc/shadow" at ARGS:cmd. [file "/usr/share/modsecurity-c
rs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "500"] [id "932160"] [msg "Remote
Command Execution: Unix Shell Code Found"] [data "Matched Data: etc/shadow found within AR
GS:cmd: cat/etc/shadow"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-m
ulti"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-leve
l/1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/88"] [tag "PCI/6.5.2"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Matched phrase "etc/shadow" at ARGS:cmd. [file "/usr/share/modsecurity-crs/rul
es/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "97"] [id "930120"] [msg "OS File Acces
s Attempt"] [data "Matched Data: etc/shadow found within ARGS:cmd: cat /etc/shadow"] [seve
rity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "language-multi"]
 [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag
"capec/1000/255/153/126"] [tag "PCI/6.5.4"] [hostname "127.0.0.1"] [uri "/command.php"] [u
nique_id "Y-rT_V10ftOo0AKdI-JC2gAAAAU"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Matched phrase "etc/shadow" at ARGS:cmd. [file "/usr/share/modsecurity-crs/rul
es/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "500"] [id "932160"] [msg "Remote Comma
nd Execution: Unix Shell Code Found"] [data "Matched Data: etc/shadow found within ARGS:cm
d: cat/etc/shadow"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-mult
i"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/
1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/88"] [tag "PCI/6.5.2"] [hostname "127.0.0.
1"] [uri "/command.php"] [unique_id "Y-rT_V10ftOo0AKdI-JC2gAAAAU"]

<SNIP>

Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 172.17.0.1] ModSecurit
y: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecuri
ty-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anoma
ly Score Exceeded (Total Inbound Score: 13 - SQLI=0,XSS=0,RFI=0,LFI=5,RCE=5,PHPI=0,HTTP=0,
SESS=0): individual paranoia level scores: 13, 0, 0, 0"] [ver "OWASP_CRS/3.3.2"] [tag "eve
nt-correlation"] [hostname "127.0.0.1"] [uri "/command.php"] [unique_id "Y-rT_V10ftOo0AKdI
-JC2gAAAAU"

<SNIP>
```

> Note: The `error.log` file may contain additional information about the malicious requests processed by the server, however, I leave that for you to explore.

## SQL Injection

In an SQL Injection (SQLi) attack, attackers manipulate the website's input fields to submit malicious SQL code which is then executed on the server.

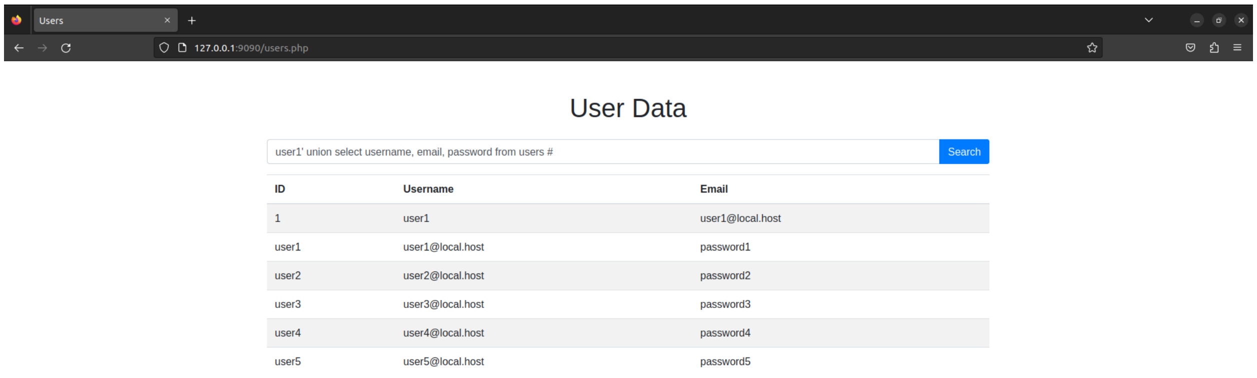
For example, consider a website with a login page that takes in a username and password. Normally, the website would compare the entered credentials against those stored in a database to determine if the user should be granted access. If the website's code is vulnerable to SQL injection, an attacker could enter `' OR 1=1--` as their username, which would essentially trick the database into returning all records, bypassing any authentication checks.

To try it out, head over to http://127.0.0.1:9090/users.php and it should display a list of users along with an input field that let's us search for users by their usernames.

To start with the attack and generate logs on the server enter the following payloads in sequence:

1. `user1' and 1=1 #`
2. `user1' union select username, email, password from users #`

We use the first query to check if the web application is vulnerable to SQL injection, and the second query to extract the users' passwords from the database.



User Data

user1' union select username, email, password from users #

| ID | Username | Email |
|---|---|---|
| 1 | user1 | user1@local.host |
| user1 | user1@local.host | password1 |
| user2 | user2@local.host | password2 |
| user3 | user3@local.host | password3 |
| user4 | user4@local.host | password4 |
| user5 | user5@local.host | password5 |

As both of these are POST requests, we can check for logs inside the `error.log` file:

```
root@18c7468edbe7:/var/log/apache2# cat error.log | grep SQLi
[Tue Feb 14 07:06:08.179011 2023] [:error] [pid 283] [client 172.17.0.1:46534] [client 17
2.17.0.1] ModSecurity: Warning. detected SQLi using libinjection with fingerprint 's&1c'
 [file "/usr/share/modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line
 "65"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched
Data: s&1c found within ARGS:search: user1' and 1=1 #"] [severity "CRITICAL"] [ver "OWASP_
CRS/3.3.2"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag
 "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/66"]
 [tag "PCI/6.5.2"] [hostname "127.0.0.1"] [uri "/users.php"] [unique_id "Y-rskPjbDGrOk3ohd
6V4HQAAAAQ"], referer: http://127.0.0.1:9090/users.php
[Tue Feb 14 07:06:30.457523 2023] [:error] [pid 284] [client 172.17.0.1:43214] [client 17
2.17.0.1] ModSecurity: Warning. detected SQLi using libinjection with fingerprint 'sUEnk'
 [file "/usr/share/modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line
 "65"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched
Data: sUEnk found within ARGS:search: user1' union select username, email, password from u
sers #"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.2"] [tag "application-multi"] [tag "lan
guage-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OW
ASP_CRS"] [tag "capec/1000/152/248/66"] [tag "PCI/6.5.2"] [hostname "127.0.0.1"] [uri "/us
ers.php"] [unique_id "Y-rspuEICGjDc-9y71EemAAAAAU"], referer: http://127.0.0.1:9090/users.
php
```

As can be seen, running `cat` command with `grep` lets us see immediately the two occurrences of SQLi detection made by WAF. The detailed audit logs for these detections can be found inside the `modsec_audit.log` file, and I'll suggest you to explore and experiment with that on your own.

# Exercises

You are a cyber security specialist who has been called upon to investigate a major cyber security breach. The company's web server has been compromised, and the attacker has attempted to exploit multiple vulnerabilities. You've been given the task of piecing together the attacker's intentions and uncovering the extent of the damage. With that in mind, your challenge is to answer the following questions:

1. What IP address does the attack seem to be originating from?
2. Which vulnerabilities do you think are being exploited, and what evidence do you have to support your findings?
3. How can we determine what web browser the attacker is using?
4. Did the attacker use any automated tools during the attack? If so, can you identify the name of the tool and its purpose?
5. Which file was the attacker trying to access but couldn't due to limited server access?
6. Did the attacker gain access to any confidential data? If yes, how much data was compromised?
7. An important secret was compromised. Can you figure it out?
   Hint: The secret you're looking for is not in a `.sql` or a `.php` file.
8. The attacker left a message for the server administrator. Find out what the message said, and also mention how you were able to find it.
9. What were some indicators that confirmed that an attack had taken place? What were your key takeaways from this attack?
10. Based on this attack, what indicators of compromise can be used to detect future attacks?

The logs can be downloaded from https://github.com/vonderchild/digital-forensics-lab/tree/main/Lab 4/files/logs.zip.