DETECTING WEB ATTACKS BY P3NGU1N:

What are web attacks?

- web application like google, facebook provide web interface
- attackers attack on these web interface to steal info cause damage and harm to them
- a study found that 75 percent of attacks were at web app level

---

Why detecting web attacks important?

- normal people use web apps daily that why attacker chose to attack web apps because it contains critical data
- best way is to prevent such attacks by firewall, IPS,SIEM Rule

---

OWASP:

- open worldwide application security project
- non-profit foundation dedicatedly working on web application security

OWASP TOP TEN:

- every few years they publish a list of vulnerabilities which pose the critical security risk

The 2021 OWASP list contains these critical vulnerabilities:

- Broken Access Control

- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

🎯 QUESTION:

What is the name of the tool that OWASP has prepared to help scan web applications for vulnerabilities?

ANSWER: zap

🎯 QUESTION:

Which area does OWASP focus on?

A) Web Applications

B) Server Management

C) Wireless Security

ANSWER: A

🎯 QUESTION:

What is the name of the vulnerable web application project that OWASP wrote using Node.js for security researchers to improve themselves?

ANSWER: juice_shop

🎯 QUESTION:

What does the OWASP Top 10 list, published every few years, reveal?

A) Most critical security risks to mobile applications

B) Most critical security risks to web applications

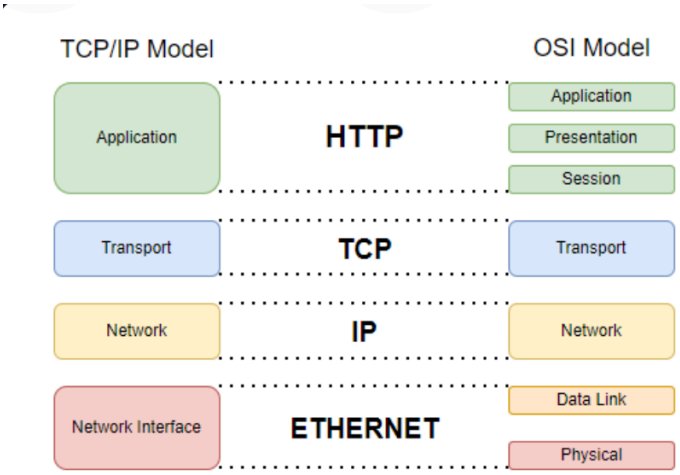
C) Most encountered web application vulnerabilities

D) Most encountered mobile application vulnerabilities

ANSWER: B

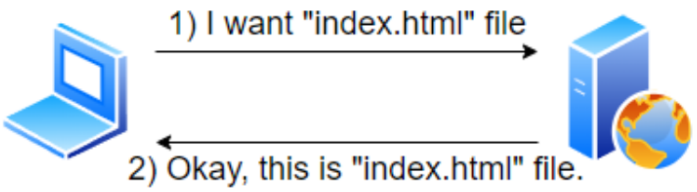
How Web Application Work:

- Applications communicate using http
- which is on application layer in OSI model
- it means that TCP, IP, SSL,TLS are used before HTTP Protocol



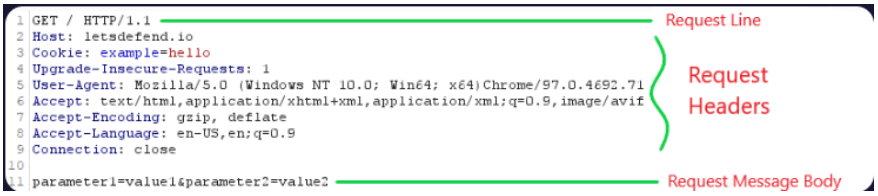
HTTP communication between server and client

- client asks for a specific resource
- server gets an https request
- server sends back an http response
- it is displayed to the client after passing through controls and processes
- it is then displayed on clients desktop in an appropriate format



HTTP REQUEST:

- request from client to server for a resource
- request must be in HTTP format so server can process it no other format will be accepted



Lets analyze the above http request:

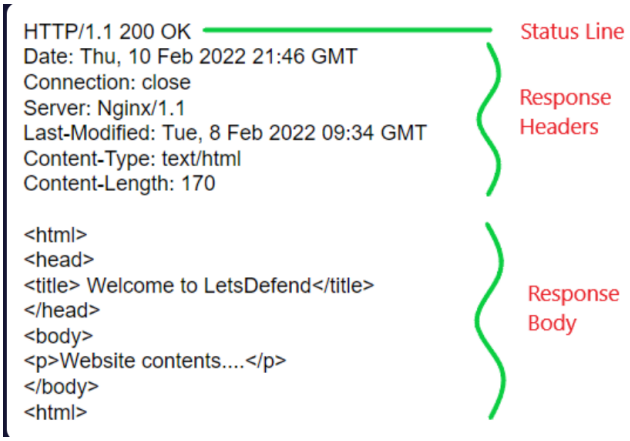
1. request line: consists of http method and resource requested from web
2. Get method "/" means that the main page of the web server is being requested
3. "Host " header to identify which domain the requested resource belongs to.
4. When a web application wants to store information on the client's device, it stores it in a "cookie" header. session info is stored here
5. "Upgrade-Insecure-Requests" header indicates that the client wants to communicate using encryption (SSL)
6. "User-Agent" header contains information about the client's browser and operating system
7. type of data requested is in the "Accept" e.g an html page
8. type of encoding accepted by the client is found in the "Accept-Encoding" header. You can usually find the names of compression algorithms
9. "Accept-Language" header contains the client's language information.
10. "Connection" header shows how the HTTP connection is made. If there is data such as "close", it means that the TCP connection will be closed after receiving the HTTP response. If you see "keep-alive", this means that the connection will be maintained.
11. empty line to create a partition
12. other data sent to site comes in request message body. for example we send credentials means http post method is used its parameters can be found in here

HTTP Responses

- server receives an HTTP request and after some processing and checks
- it gives an HTTP response
- there are no standard processing and checks depends on technology and design

It contains:

1. status line : contains status code (e.g 200 OK)
2. response header : multiple purpose
3. response message body : usually if a site is requested it will contain html code of website, contains the resource requested



Status Line:

contains HTTP response status code

used to describe the status of request

- 100-199: Informational responses
- 200-299: Successful responses
- 300-399: Redirection messages
- 400-499: Client error responses
- 500-599: Server error responses

Response Headers:

- Date: The exact time the server sent the HTTP Response to the client.
- Connection: This indicates how the connection is handled, just like the HTTP Request header.
- Server: It informs about the operating system of the server and the version of the web server.
- Last-Modified: It provides information about when the requested resource was modified. This header is used by the caching mechanism.
- Content-Type: The type of data being sent.
- Content-Length: The size of the data sent

Response Body:

The HTTP response body contains the resource sent by the server and requested by the client.

🎯 QUESTION:

What layer is HTTP on in the OSI model?

ANSWER: application

🎯 QUESTION:

Which HTTP Request header contains browser and operating system information?

ANSWER: user-agent

🎯 QUESTION:

What is the HTTP Response status code that indicates the request was successful?

ANSWER: 200

🎯 QUESTION:

Which HTTP Request Method ensures that the submitted parameters do not appear in the Request URL?

ANSWER: post

🎯 QUESTION:

Which HTTP Request header contains session tokens?

ANSWER: Cookie

Detecting SQL Injection Attacks

What is SQL Injection (SQLi)?

is a web sec vulnerability

allows an attacker to interfere with queries that application makes to its database

we have a framework that has prevention mechanism to protect from sql injection but it still happens sometimes due to improper use of framework

Types of SQL Injections:

Three types of sql injection:

In- band SQLi

- attackers sends a query
- receives response on same channel

Out-of-Band SQLi

- attacker sends a query
- receives response on some other channel like DNS

Inferential SQLi

- Attacker sends a query
- But response cannot be seen by attacker

How Does SQL Injection Work?

Web application asks user for input like in a login page

SQLi are usually common in login pages

web app takes input and makes a sql query

lets take a look at a common sql query

SELECT * FROM users WHERE username = 'USERNAME’ AND password = 'USER_PASSWORD'

it means check the user table and find user with username ”this” and password “this”

if match found in database it will authenticate user

if not found, login will fail

if my username is KIRAN and password is Kiran123 sql query will be

SELECT * FROM users WHERE username = 'KIRAN’ AND password = 'Kiran123'

if we add an apostrophe it will give error in sql query

SELECT * FROM users WHERE username = 'KIRAN’’ AND password = 'Kiran123'

ATTACKER USES A PAYLOAD

‘ OR 1=1 — -

NOW query will be

> **SELECT * FROM users WHERE username = '’ OR 1=1 — - AND password = 'Kiran123'**

In SQL, any characters after "-- -" are considered to be a comment line

so query will be

> SELECT * FROM users WHERE username = ‘’ OR 1=1

The query above now looks like this "if the username is empty or 1=1".

What Attackers Gain from SQL Injection Attacks

- Authentication bypass
- Command execution
- Exfiltration of sensitive data
- Creating/Deleting/Updating database entries

How to Prevent SQL Injections

- Use a framework

- Keep your framework up to date
- Always sanitize data received from a user
- Avoid the use of raw SQL queries

How to detect SQL injection

- check all areas that come from user
- look for special characters
- familiarize with common sql payloads
- look for sql keyword

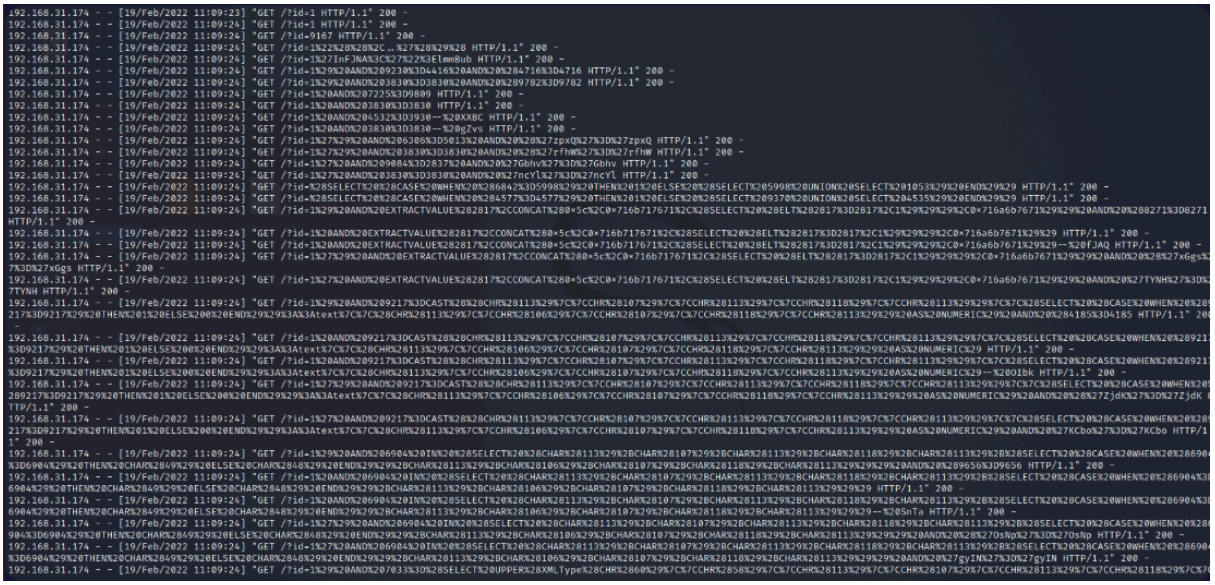
Detecting Automated SQL injection Tools:

Well known tool is SQLmap but look at bigger picture:

- look at user-agent
- Look for request per second
- if payload is complicated
- check content if payload like sqlmap’ OR 1=1

A Detection Example:

Access log from a web server contain date, ip, URL,HTTP method , user-agent useful for investigations



what the % symbols mean

- When we request a page that contains special characters
- requests are not sent directly to the web server.
- our browsers perform a URL encoding ("Percent Encoding") of the special characters
- replace each special character with a string that starts with % and contains 2 hexadecimal characters.

Character	From Windows-1252	From UTF-8
space	%20	%20
!	%21	%21
"	%22	%22
#	%23	%23
$	%24	%24
%	%25	%25
&	%26	%26
'	%27	%27
(	%28	%28
)	%29	%29

we can use an online url decoder but not not for sensitive content

we have covered that:

- There has been an SQL injection attack on the 'id' parameter on the main page of the web application.
- The requests came from IP address 192.168.31.174.
- As there were more than 50 requests per second, this attack was carried out by an automated vulnerability scanning tool.
- The complex nature of the payloads supports the assertion in #3.
- We cannot determine if the response was successful or not as we have no information on the size of the response.

Note:

Use the "/root/Desktop/QuestionFiles/SQL_Injection_Web_Attacks.rar" file for solving the questions below.

Question:

What date did the exploitation phase of SQL Injection Attack start?

File Password:

access

Answer Format:

01/Jan/2022:12:00:00

01/Mar/2022:08:35:14

```
192.168.31.167 - - [01/Mar/2022:08:35:14 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27&Submit=Submit HTTP/1.1" 200 607 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.167 - - [01/Mar/2022:08:37:10 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+--+-&Submit=Submit HTTP/1.1" 200 4559 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.167 - - [01/Mar/2022:08:38:16 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+version%28%29+--+-&Submit=Submit HTTP/1.1" 200 4809 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+--+-&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.200 - - [01/Mar/2022:08:39:39 -0800] "-" 408 - "-" "-"
192.168.31.167 - - [01/Mar/2022:08:40:26 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+user%28%29+--+-&Submit=Submit HTTP/1.1" 200 4790 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+version%28%29+--+-&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
```

What is the IP address of the attacker who performed the SQL Injection attack?

192.168.31.167


```
192.168.31.167 - - [01/Mar/2022:08:35:14 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27&Submit=Submit HTTP/1.1" 200 607 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.167 - - [01/Mar/2022:08:37:10 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+--+-&Submit=Submit HTTP/1.1" 200 4559 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.167 - - [01/Mar/2022:08:38:16 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+version%28%29+--+-&Submit=Submit HTTP/1.1" 200 4809 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+--+-&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
192.168.31.200 - - [01/Mar/2022:08:39:39 -0800] "-" 408 - "-" "-"
192.168.31.167 - - [01/Mar/2022:08:40:26 -0800] "GET /dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+user%28%29+--+-&Submit=Submit HTTP/1.1" 200 4790 "http://192.168.31.200/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+UNION+SELECT+null%2C+version%28%29+--+-&Submit=Submit" "Mozilla/5.0 (Windows NT 6.1; rv:88.0) Gecko/20100101 Firefox/88.0"
```

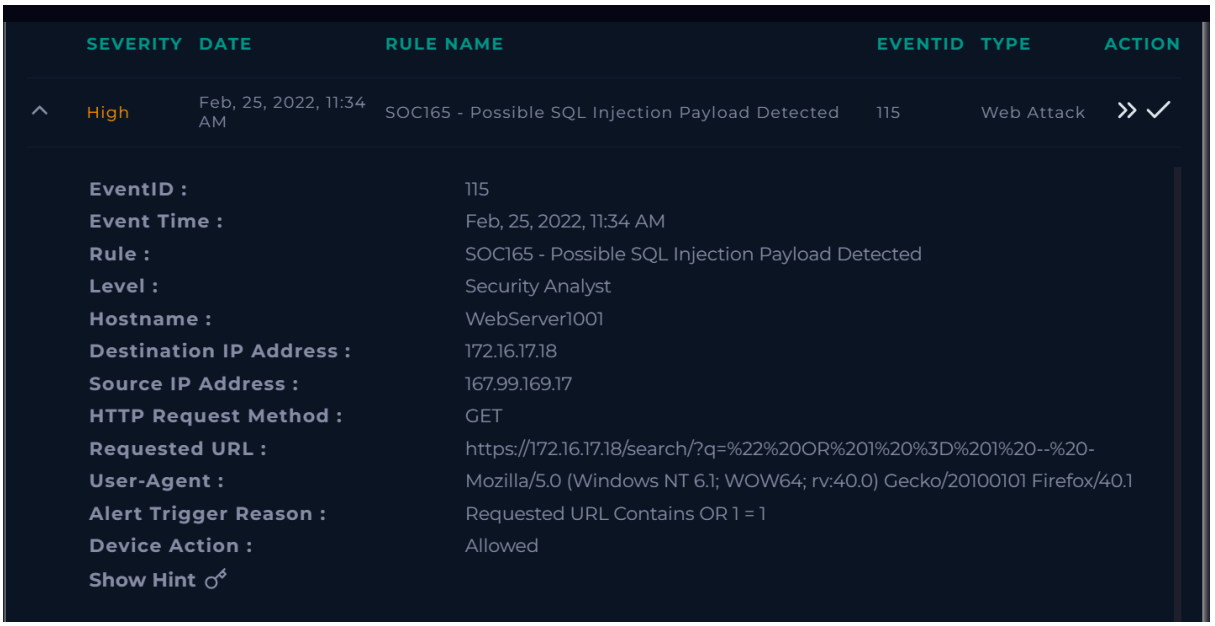

Was the SQL Injection attack successful? (Answer Format: Y/N)

Y

What is the type of SQL Injection attack? (Classic, Blind, Out-of-band

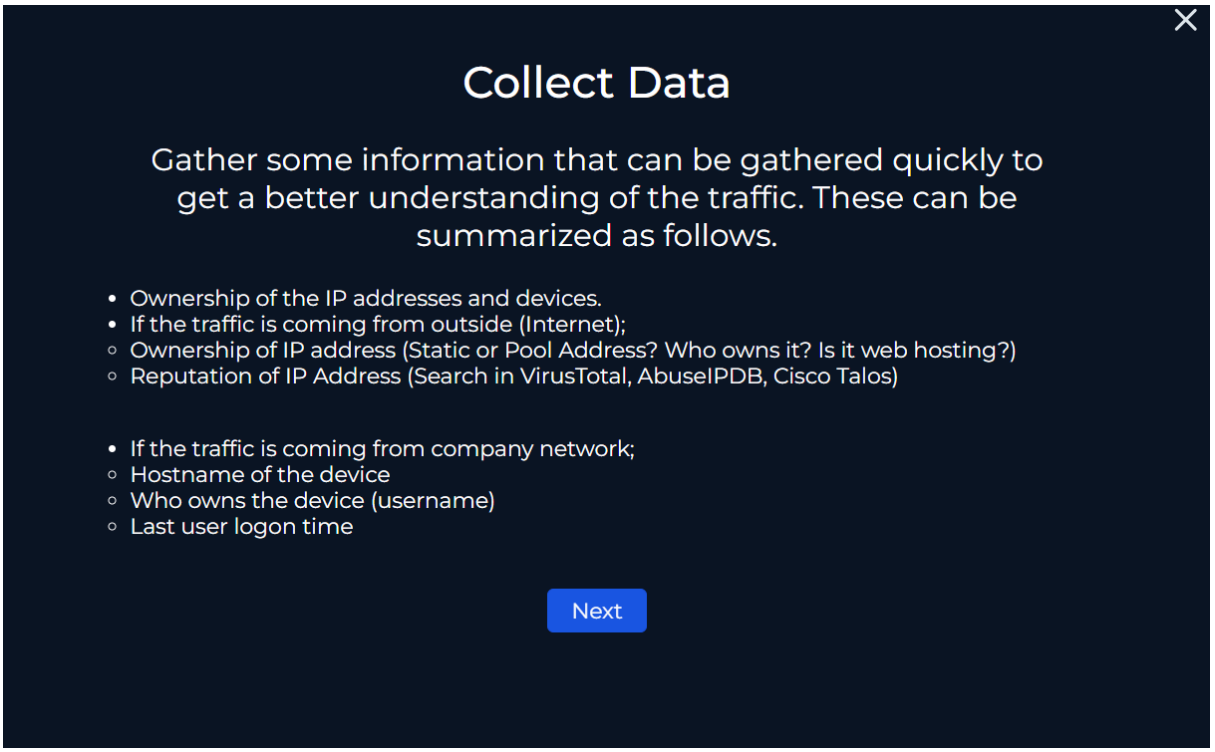
Classic

Go to monitoring > investigation channel

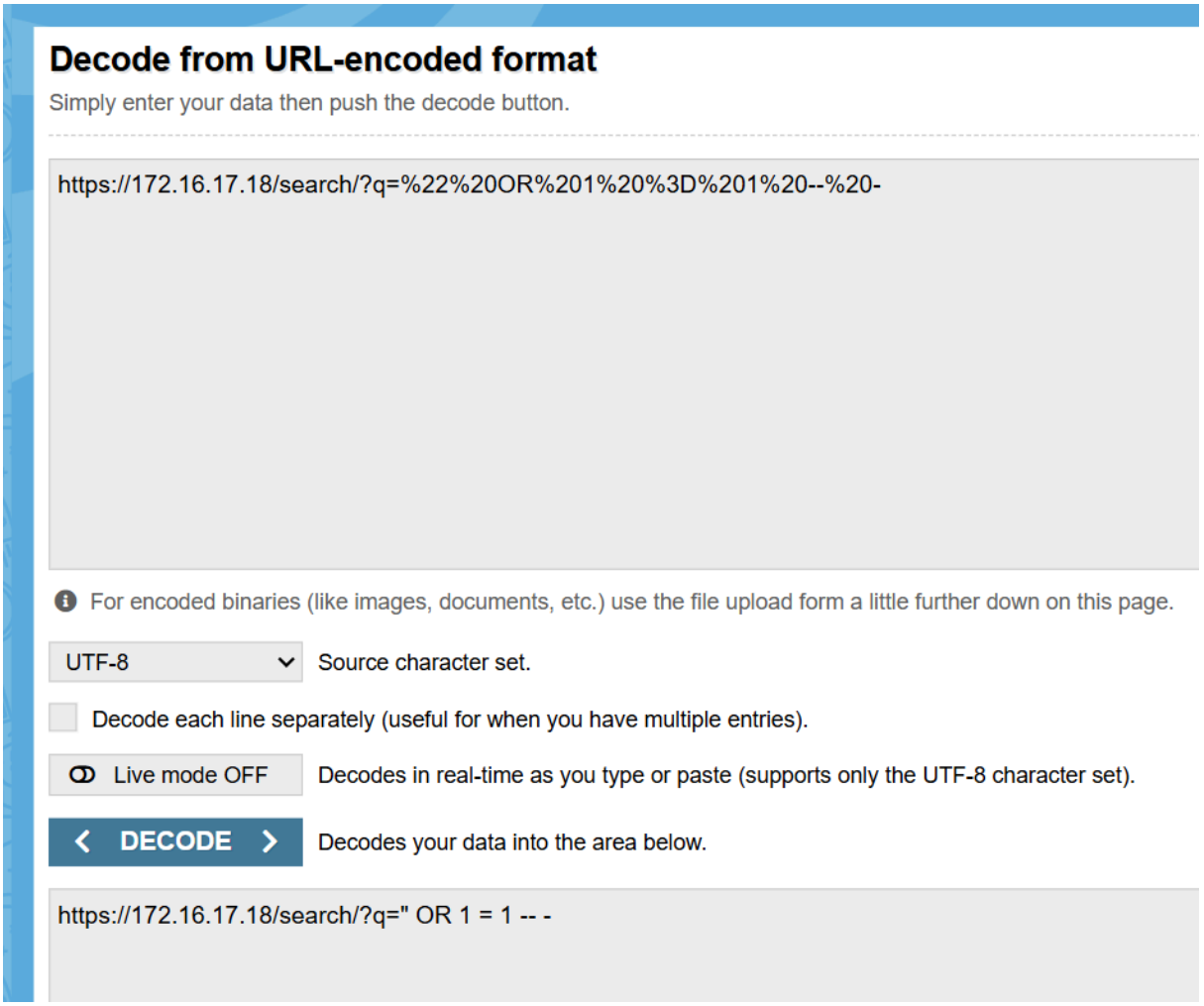


We got the answer to our why which is “Device action was allowed”

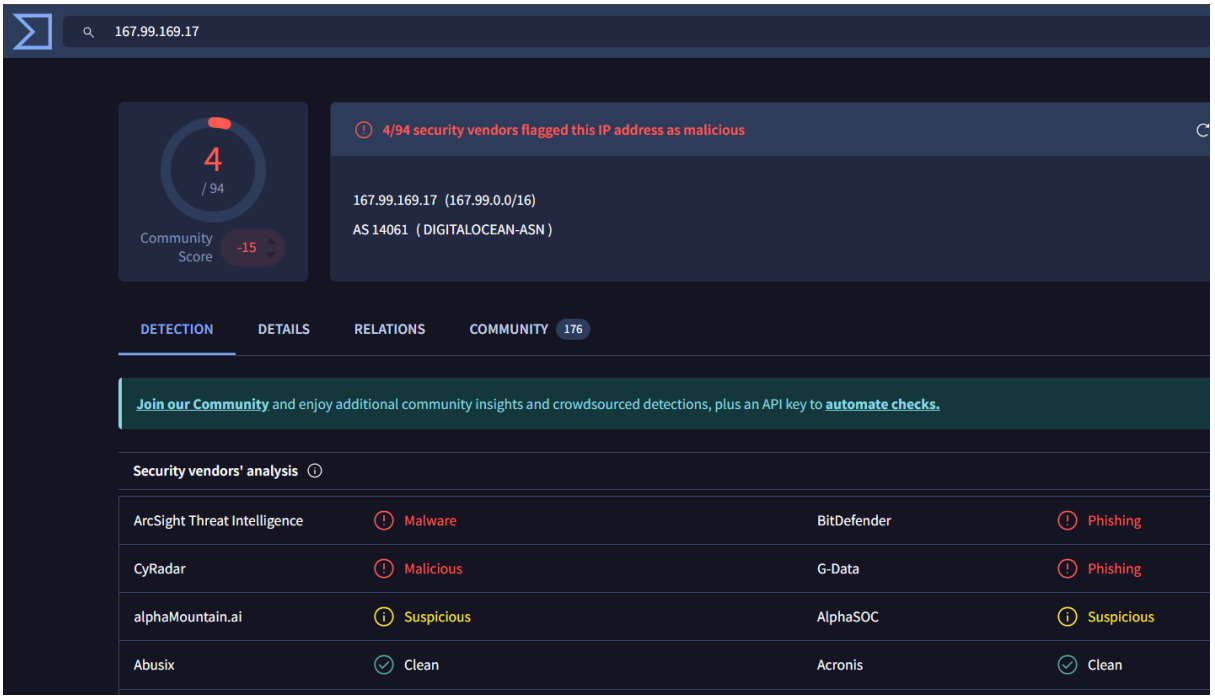
Analysis:

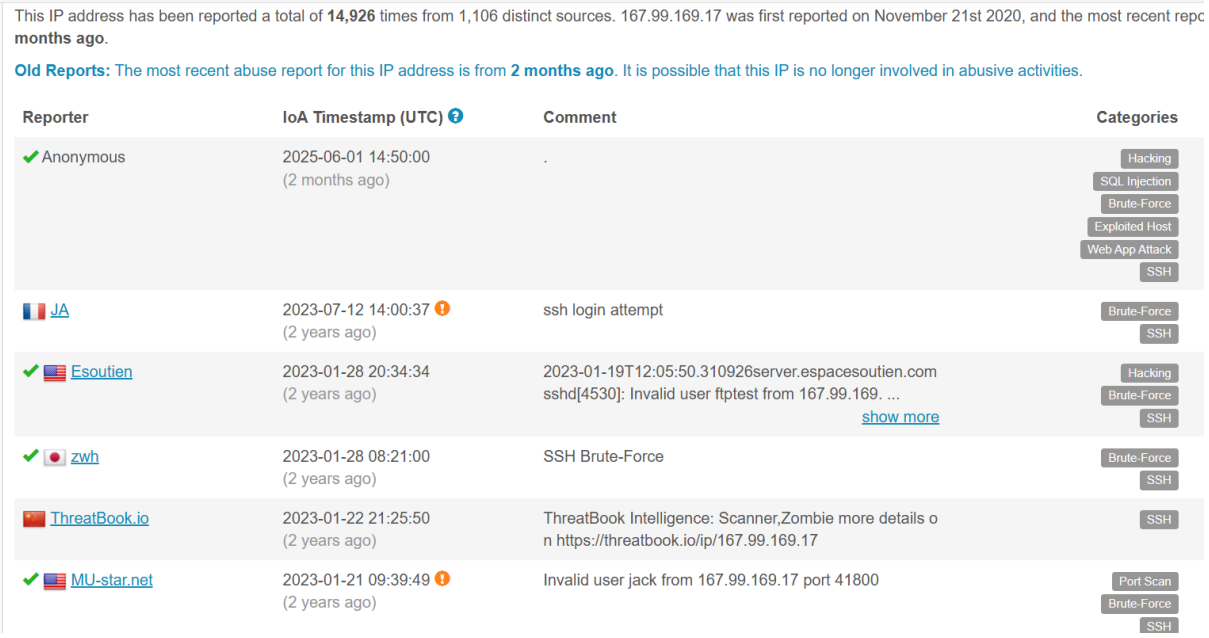


lets first decode the url to uncover Payload

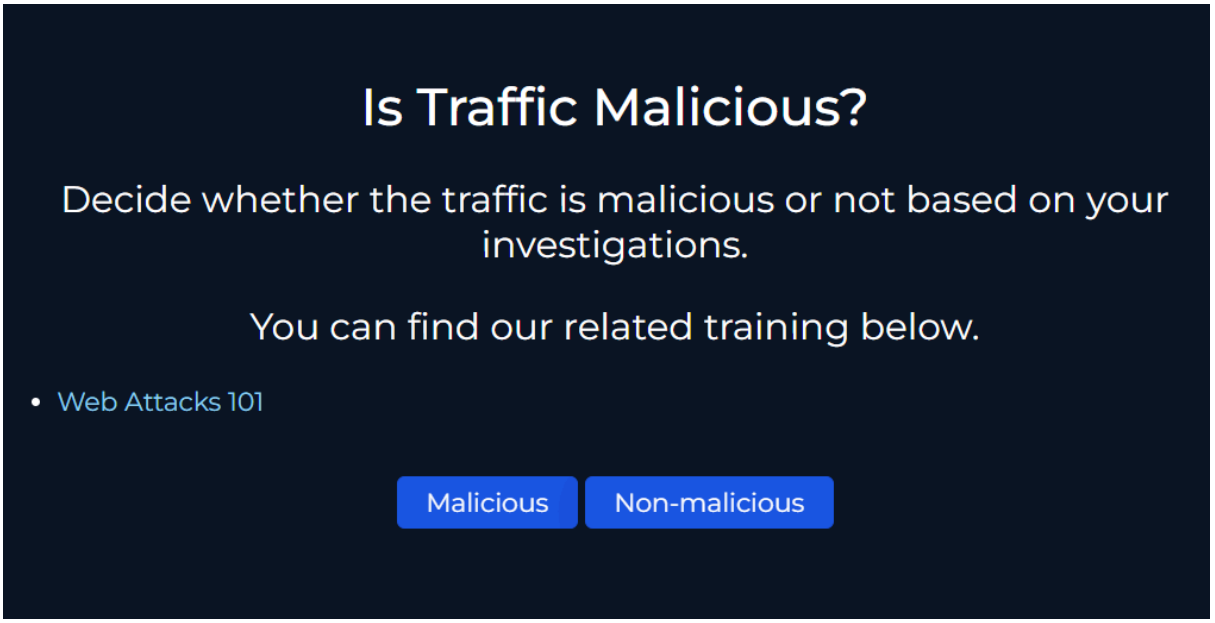


Checking source ip reputation:

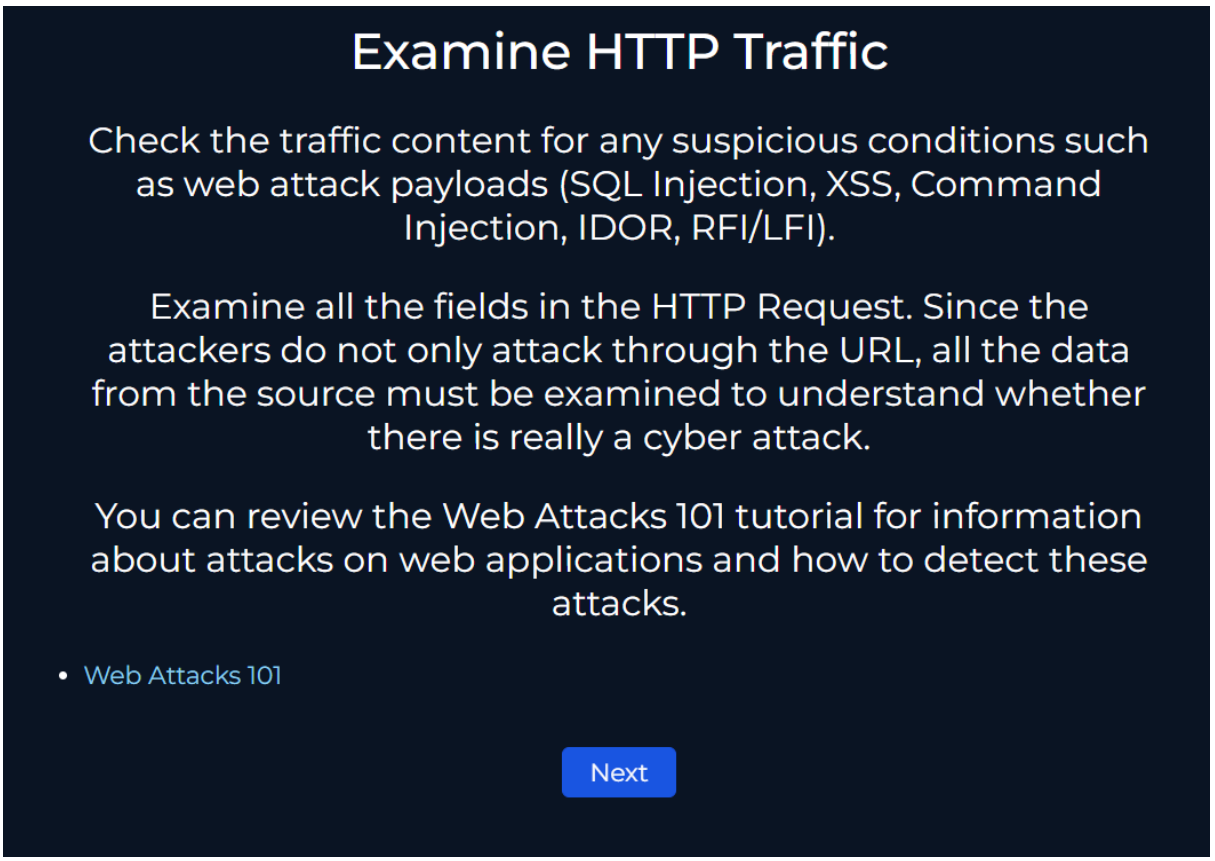




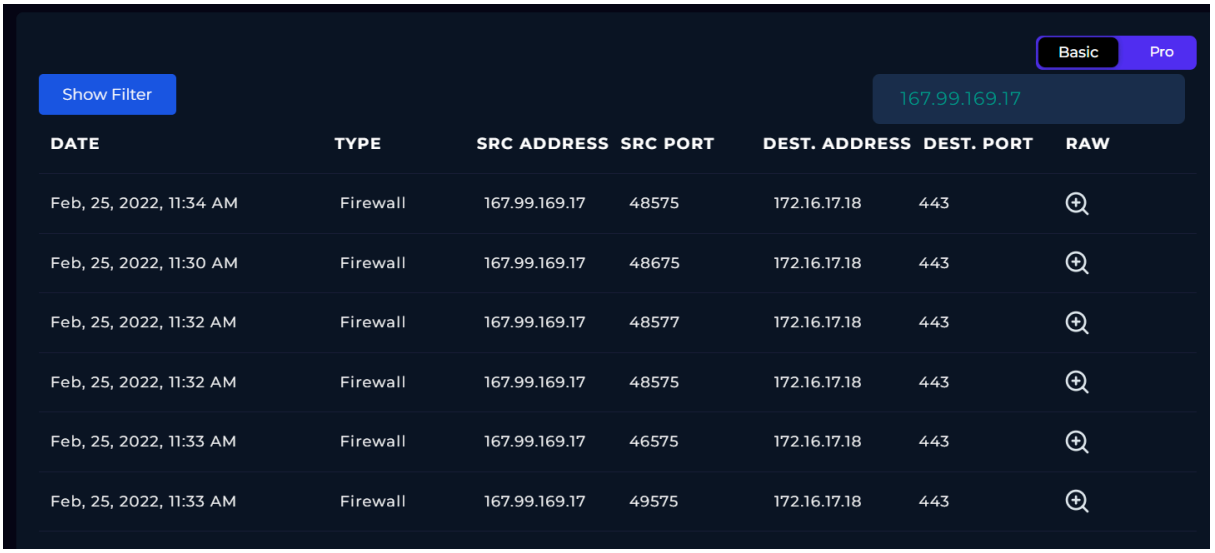
Source Ip is flagged malicious from both sites as malicious

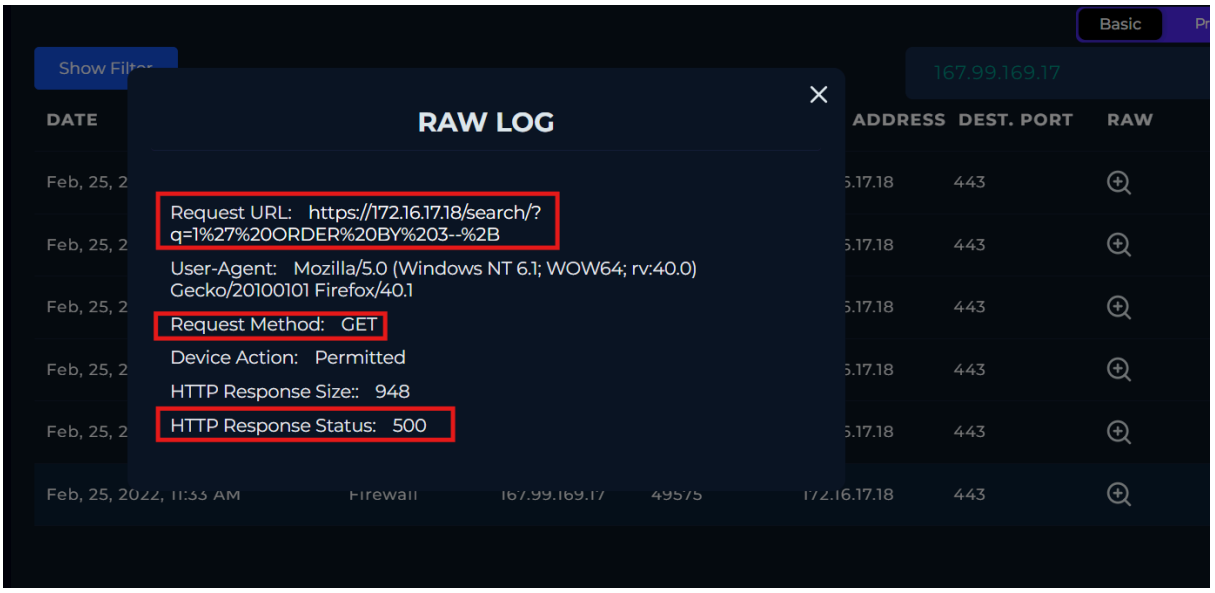


Mark as malicious



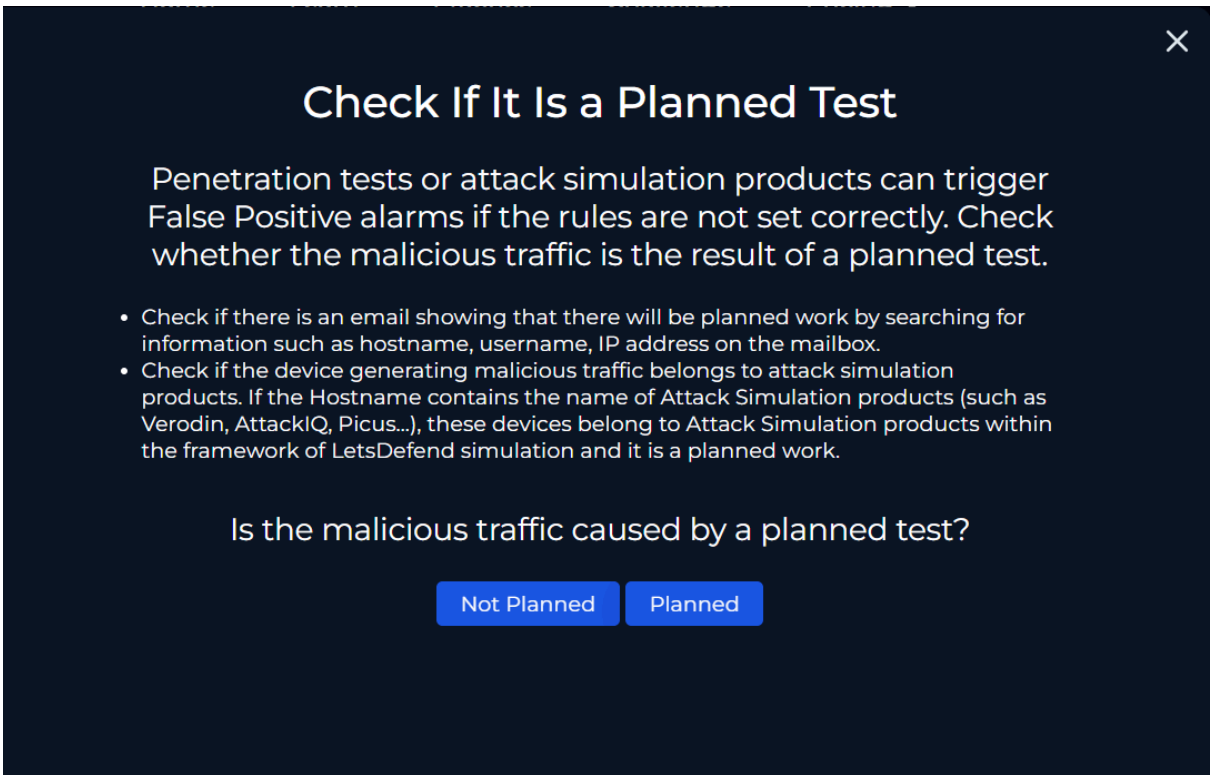
go to log management and filter using source ip which is “167.99.169.17”



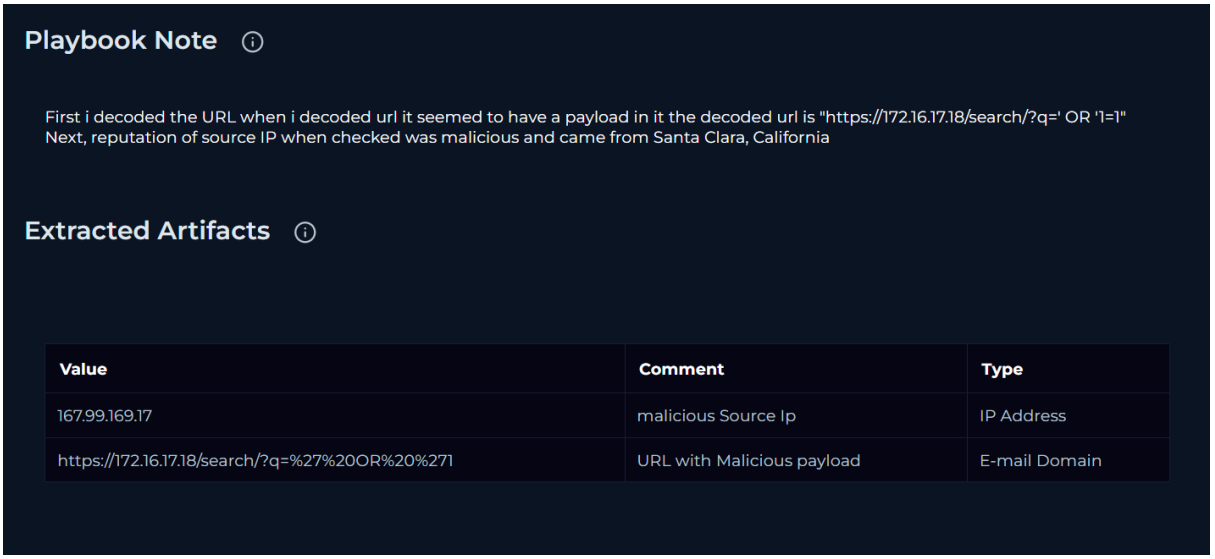


response status 500 means server gave an error and made attack unsuccessful

SQL Injection



no nothing seemed like it was planned



Detecting Cross Site Scripting (XSS) Attacks:

Cross-site scripting (XSS) is a type of injection-based web security vulnerability

can be incorporated into legitimate web applications, allowing malicious code to be executed.

frameworks have preventive measures for XSS

sometime framework is not used

or framework itself has XSS vulnerability

data coming from user is not sanitized

Types of XSS

There are 3 types of XSS. These are:

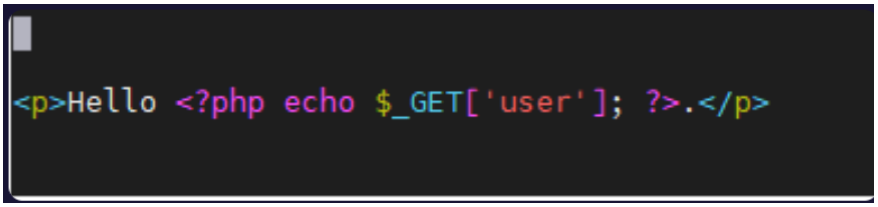
- **Reflected XSS (Non-Persistent)**: This is a non-persistent type of XSS where the XSS payload must be present in the request. It is the most common type of XSS.
- **Stored XSS (Persistent)**: This type of XSS is where the attacker can permanently upload the XSS payload to the web application. Compared to other types, Stored XSS is the most dangerous type of XSS.

- **DOM Based XSS**: DOM Based XSS is an XSS attack where the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client-side script so that the client-side code runs in an "unexpected" manner. (OWASP)

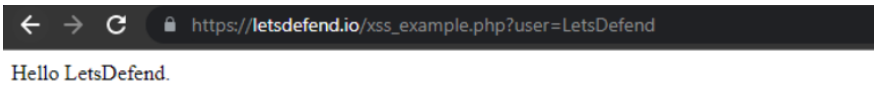
How does XSS work?

XSS is a vulnerability that is caused by a lack of data sanitization.

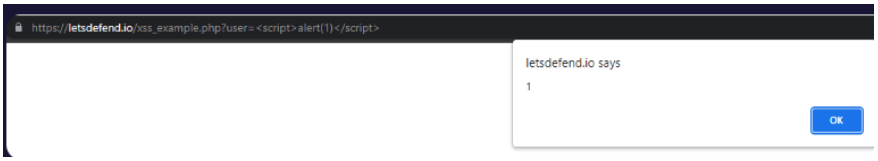
It occurs when the data received from the user is sent in the response without being sanitized.



First, we'll examine the piece of code above. What it does is actually quite simple. It simply displays whatever is entered in the 'user' parameter. If we enter "LetsDefend" as the 'user' parameter, we will see the words "Hello LetsDefend".



But, as we have already seen, there is no control mechanism for the user parameter. This means that whatever we put in the "user" parameter will be included in the HTTP response we receive back.



This is exactly how XSS works. Because the value entered by the user is not validated, the attacker can enter any javascript code

How Attackers Take Advantage of XSS Attacks

Attackers can do the following with an XSS attack:

- Steal a user's session information
- Capture credentials
- Etc.

How to Prevent a XSS Vulnerability

- Sanitize data coming from a user
- Use a framework
- Use the framework correctly:
- Keep your framework up-to-date

Detecting XSS Attacks

- Look for keywords
- Learn about commonly used XSS payloads
- Check for the use of special characters

An Example of Detection:

URL Decoder/Encoder

```
188.114.103.221 - - [19/Feb/2022:18:25:08  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:10  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:11  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:11  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:11  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:12  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:12  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:13  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.162 - - [19/Feb/2022:18:25:14  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:14  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:15  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:15  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:16  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:16  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:16  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:17  0000] "GET /blog/?s=<script>prompt(5000/200)</script> HTTP/1.1" 200 9960 "-" "python-urllib3/1.26.4"
162.158.129.162 - - [19/Feb/2022:18:25:17  0000] "GET /blog/?s=test HTTP/1.1" 200 14584 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:25  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:26  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:27  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:27  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:27  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:28  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:28  0000] "GET /blog/?s=<script>prompt(document.cookie)</script> HTTP/1.1" 200 9963 "-" "python-urllib3/1.26.4"
162.158.129.38 - - [19/Feb/2022:18:25:30  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:31  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:32  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:32  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:32  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:32  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:32  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.140 - - [19/Feb/2022:18:25:33  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
162.158.129.36 - - [19/Feb/2022:18:25:34  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:35  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:36  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:36  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:37  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:38  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:38  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.102.52 - - [19/Feb/2022:18:25:39  0000] "GET /blog/?s=<script>console.log(5000/3000)</script> HTTP/1.1" 200 9968 "-" "python-urllib3/1.26.4"
188.114.103.221 - - [19/Feb/2022:18:25:39  0000] "GET /blog/?s=test HTTP/1.1" 200 19797 "-" "python-urllib3/1.26.4"
```

As a result of our investigations:

- It is clear that the attack was aimed at the web application where the access logs came from.
- Looking at the number of requests and the user agent information, we determined that the attack was carried out by an automated vulnerability scanner.
- As the application is hosted behind Cloudflare, the source IP addresses were not found.
- We do not know if the attack was successful or not.

Note:

Use the "/root/Desktop/QuestionFiles/XSS_Web_Attacks.rar" file for solving the questions below.

Question:

What is the start date of the XSS attack?

File Password:

access

Answer Format:

01/Mar/2022:12:00:00

01/Mar/2022:08:53:20 (first payload was sent on this time)

What is the IP address of the attacker who performed the XSS attack?

192.168.31.183 (this ip sent a xss script)

Was the XSS attack successful?

Y(because 200 ok successfull)

What is the type of XSS attack? (Reflected, Stored, Dom based)

Reflected (payload present in request)

```
170 192.168.31.183 - - [01/Mar/2022:08:52:31 -0800] "GET /dvwa/ HTTP/1.1" 302 -
171 192.168.31.183 - - [01/Mar/2022:08:52:31 -0800] "GET /dvwa/dvwa/css/login.css HTTP/1.1" 200 842
172 192.168.31.183 - - [01/Mar/2022:08:52:31 -0800] "GET /dvwa/dvwa/images/login_logo.png HTTP/1.1" 200 9088
173 192.168.31.183 - - [01/Mar/2022:08:52:31 -0800] "GET /favicon.ico HTTP/1.1" 404 301
174 192.168.31.183 - - [01/Mar/2022:08:52:35 -0800] "GET /dvwa/index.php HTTP/1.1" 200 6685
175 192.168.31.183 - - [01/Mar/2022:08:52:36 -0800] "GET /dvwa/dvwa/css/main.css HTTP/1.1" 200 4026
176 192.168.31.183 - - [01/Mar/2022:08:52:36 -0800] "GET /dvwa/dvwa/js/dvwaPage.js HTTP/1.1" 200 1030
177 192.168.31.183 - - [01/Mar/2022:08:52:36 -0800] "GET /dvwa/dvwa/images/logo.png HTTP/1.1" 200 5044
178 192.168.31.183 - - [01/Mar/2022:08:52:36 -0800] "GET /dvwa/js/add_event_listeners.js HTTP/1.1" 404 301
179 192.168.31.183 - - [01/Mar/2022:08:52:36 -0800] "GET /dvwa/favicon.ico HTTP/1.1" 200 1406
180 192.168.31.183 - - [01/Mar/2022:08:52:41 -0800] "GET /dvwa/vulnerabilities/xss_r/ HTTP/1.1" 200 4224
181 192.168.31.183 - - [01/Mar/2022:08:52:41 -0800] "GET /dvwa//dvwa/js/add_event_listeners.js HTTP/1.1" 200 593
182 192.168.31.183 - - [01/Mar/2022:08:52:54 -0800] "GET /dvwa/vulnerabilities/xss_r/?name= HTTP/1.1" 200 4224
183 192.168.31.183 - - [01/Mar/2022:08:52:58 -0800] "GET /dvwa/vulnerabilities/xss_r/?name=hacker HTTP/1.1" 200 4247
184 192.168.31.183 - - [01/Mar/2022:08:53:06 -0800] "GET /dvwa/vulnerabilities/xss_r/?name=john HTTP/1.1" 200 4245
185 192.168.31.183 - - [01/Mar/2022:08:53:20 -0800] "GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%281%29%3C%2Fscript%2F HTTP/1.1" 200
    4266
186 192.168.31.183 - - [01/Mar/2022:08:54:34 -0800] "GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
    HTTP/1.1" 200 4280
187 192.168.31.183 - - [01/Mar/2022:08:55:08 -0800] "GET /dvwa/vulnerabilities/xss_r/?-
    name=%3Cscript%3Elocation.href%3D%27http%3A%2F%2Fmaliciousssite.com%27%3C%2Fscript%3E HTTP/1.1" 200 4298
```

Detecting Command Injection Attacks

What are Command Injection Attacks?

are executed when data from user is not sanitized and taken as a command and sent to OS Shell

Attacker purpose is to gain full control here so its a critical vulnerability

How does Command Injection work?

Suppose we have a basic web application that copies the user's file to the "/tmp" folder. The web application code is shown below:

```
<?php

$file_name = $_POST['file_name']
$output = shell_exec('cp $file_name /tmp/');

?>
```

Note:

Use the "/root/Desktop/QuestionFiles/Command_Injection_Web_Attacks.rar" file for solving the questions below.

Question:

What is the date the command injection attack was initiated?

File Password:

access

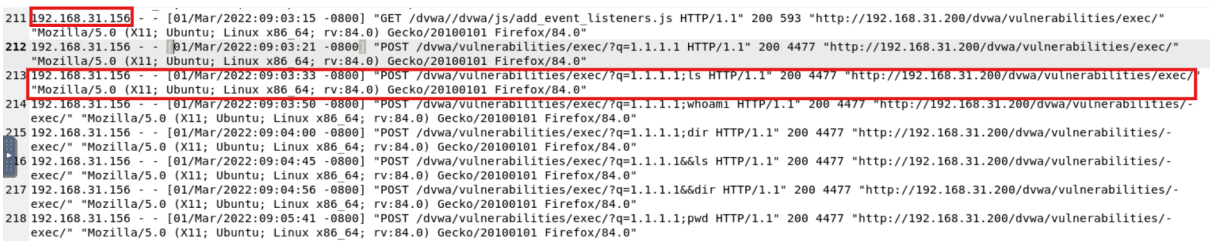
Answer Format:

01/Mar/2022:12:00:00

01/Mar/2022:09:03:33

What is the IP address of the attacker who performed the Command Injection attack?

192.168.31.156



Was the Command Injection attack successful?

N

Detecting Insecure Direct Object Reference (IDOR) Attacks

Note:

Use the "/root/Desktop/QuestionFiles/IDOR_Web_Attacks.rar" file for solving the questions below.

Question:

What is the IP address of the attacker who carried out the IDOR attack?

File Password:

access

192.168.31.174

What is the date when the attack started?

Answer Format:

01/Mar/2022:12:00:00

01/Mar/2022:11:42:32

Was the attack successful?

Y

Was the attack carried out by an automated tool?

N

Detecting RFI & LFI Attacks

Note:

Use the "/root/Desktop/QuestionFiles/File_Inclusion_Web_Attacks.rar" file for solving the questions below.

Question:

What is the attacker's IP address?

File Password: access

192.168.31.174

What is the start date of the attack?

Answer Format: 01/Mar/2022:12:00:00

01/Mar/2022:11:58:35

Was the attack successful?

Answer

N

```
178 192.168.31.174 - - [01/Mar/2022:11:58:35 -0800] "GET /dvwa/vulnerabilities/fi/?-
    page=../../ HTTP/1.1" 200 50 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
    Gecko/20100101 Firefox/78.0"
179 192.168.31.174 - - [01/Mar/2022:11:58:48 -0800] "GET /dvwa/vulnerabilities/fi/?-
    page=../../index.php HTTP/1.1" 200 50 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:-
    78.0) Gecko/20100101 Firefox/78.0"
180 192.168.31.174 - - [01/Mar/2022:11:58:54 -0800] "GET /dvwa/vulnerabilities/fi/?-
    page=../../../../../../etc/passwd HTTP/1.1" 200 50 "-" "Mozilla/5.0 (X11; Linux
    x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
181 192.168.31.174 - - [01/Mar/2022:11:59:08 -0800] "GET /dvwa/vulnerabilities/fi/?-
    page=../../text.php HTTP/1.1" 200 50 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:-
    78.0) Gecko/20100101 Firefox/78.0"
```