Prepared By:

Gopi Gor

WEB APPLICATION EXPLOITATION

TABLE OF CONTENTS

Executive Summary

Web applications are a growing threat in the world of cybersecurity and attackers are finding new ways to exploit web vulnerabilities and attack weak and vulnerable websites. The web exploitation module explores 4 different methods used to exploit vulnerabilities mainly SQL Injection, Cross-Site Scripting, File Inclusion and File Upload - Web Shell. These 4 are the most common and widely used methods which attackers use to gain access to sensitive data, disrupt web services and provide unavailability to users or steal cookie and browser data.

These vulnerabilities happen due to insecure or poor coding practices, improper input validations, weak security measures and lack of user security awareness. This leads to large losses for organization like loss of reputation, financial losses, customer trust and legal liabilities.

This report outlines few of the web application exploitation methods, where Kali Linux VM is used mainly to exploit the vulnerabilities through the Damn Vulnerable Web Application (DVWA) and demonstrate the threats that an organization faces due to vulnerabilities in websites and webpages.

Scope & Objectives

The scope of this report is to identify the vulnerabilities in the web applications using DVWA and highlighting certain attacks by demonstrating how they are done and in what ways they can affect the web application, organization and users who may be involved in a web based exploit.

The objectives of this lab are as below:

- Discovering an SQL Injection vulnerability and demonstrating how it can used to exfiltrate a database .
- Discovering an XSS vulnerability and demonstrating how it can be used maliciously.
- Discovering a Local File Inclusion Vulnerability and demonstrating how it can be used maliciously.
- Creating a method of persistence using WebShell and using this maliciously.
- Providing recommendations to avoid these mistakes and make sure that the web applications are not vulnerable to any of these attack types.

SQL Injection

SQL Injection is a method which an attacker uses to steal important data and database information by injecting a specific code into the SQL statements through the user input on the website. This usually occurs due to improper input validation, which allows the attacker to modify or delete data.

SQL Injections can be of many types, and the one that was discovered in this report was a specific SQL query which allows unauthorized access and reveals usernames and/or passwords. The query that was typed into the input field was **‘ union all select system_user(),user() #.** This query is explained below.

1. ‘ is the string literal which is used to start the query.
2. The union all part of the statement allows to combine one more select into one result set, and here it will injecting additional data into the input.
3. The system_user() and user() fields return the value of the current user in different database types.
4. The # is used to denote the end, and anything after is a comment.

This can be used maliciously to completely exfiltrate all data of users from the database and then use it on the website as needed to steal financial or personal information.

Cross-Site Scripting

Cross-Site Scripting is when an adversary injects malicious code into the website. The user is not targeted directly but instead there can be pop-ips which look believable as if they are part of the actual legit website. This malicious script allows the attacker to input something that will be displayed when run with an injected script.

There are 3 main types of XSS - reflected, stored and DOM based, and in this assignment, we found a vulnerability for the Reflected XSS. This creates a pop-up when the user tries to click submit, and this showed that a user who is unaware of this script, will access the link or click OK on a pop-up which then gives access to the attacker to steal cookies, or sensitive information. These scripts allow the original HTML code of the website to rewritten too, and this could be a very major problem that the organization might face in terms of integrity too.

As seen in the appendix, we have first shown the input and output for a reverse cross site scripting, which further when we included the JavaScript **<script>alert(“You have been hacked!”)</script>**, it showed a popup and if the user clicks okay, he is giving away information he may not be aware of.

Local File Inclusion

The local file inclusion vulnerability allows an adversary to include a file in the target web application. This means that the attacker can read contents, or pass malicious code through the file and also make changes to the source code in certain extreme cases where the severity is low.

The LFI occurs when certain include statements are not coded properly, and password files can be viewed as well. The $file command in the source code which we noticed, tells us that the file might be visible through this webpage, which then would allow to make changes to the URL and access hidden files related to that.

In this part of the assignment, we found a vulnerability in one of the files, and then when we altered the URL on another tab, we could see the complete password file with all usernames and passwords listed. The images in the appendix justify this explanation. We did a directory traversal, where the first link of the file we opened was **127.0.0.1/DVWA/vulnerabilities/fi/?page=file2.php** and then we made a change to this url and **127.0.0.1/DVWA/vulnerabilities/fi/?page=/etc/passwd** was run to get all the passwords.

Web Shell

Web shell is a technique which is used by threat actors to compromise a website and send malicious code. This allows them to install a backdoor via a web shell to have constant access into the system for longer periods of time.

Web shells often work through vulnerabilities and weak passwords in these web servers, and once installed may go unnoticed for long periods of time. They can potentially harvest credentials and sensitive data, inject malware and also deface a website in severe cases.

In this assignment, we locate webshell php files, to then upload and establish a backdoor into the “file upload” section of DVWA. After this, we see a path that has been shown where our php backdoor file is located and we then clicked on that link to set up the web shell. Once done, we would be able to see a remote session that started on the machine.

It is important to note that web shells are dangerous and it is very important to make sure that they are well taken care of to avoid allowing a potential backdoor to be used.

Methodology

In this assignment of Web Application Exploitation, we started with understanding the different kind of web attacks that occur and our main tool used in this case was our Kali Linux VM, and the Damn Vulnerable Web Application (DVWA).

Another important factor that played a role to perform all these exploits was the Source Code that was there at the bottom of every exploit we performed. This allowed to analyze the code in detail and understand what was wrong with the code or what would allow an attacker an entry door into the application. We saw that majorly our web apps used SQL queries and database and additionally files were php files that we came across in many scenarios. These also helped us to identify the GET or POST requests which were being used on that specific website/webpage.

Additionally, we also used wget to test a CISCO website to get all the servers and IP addresses related to the web server that we were looking at. This allowed for additional recon in the case of trying to find multiple entry points.

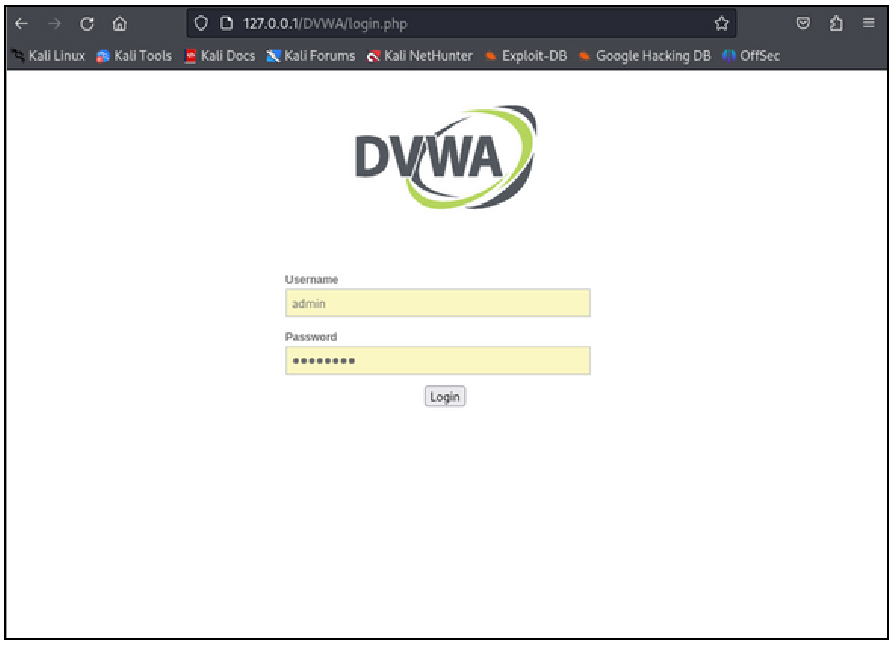
Recommendations

In terms of the web-application exploitation, there are multiple recommendations which can and should be applied to every web server, physically, locally and even on each layer.

- The first and foremost recommendation is to use very secure coding practices, with proper input validation, thereby preventing open doors for the attacker.
- The passwords used should be a strong password along with local storage of these, rather than on the unsecure web server.
- Limiting control and setting the right access control policies is very important in all senses to avoid any unauthorized port entries along with authentication.
- Using a web application firewall would be useful as well to make sure the limiting access begins from there itself.
- Penetration tests should be carried out regularly along with vulnerability monitoring on a daily or weekly basis.
- Patches for the applications, code checks, and password with MFA should be set up.

Appendix

1.SQL Injection

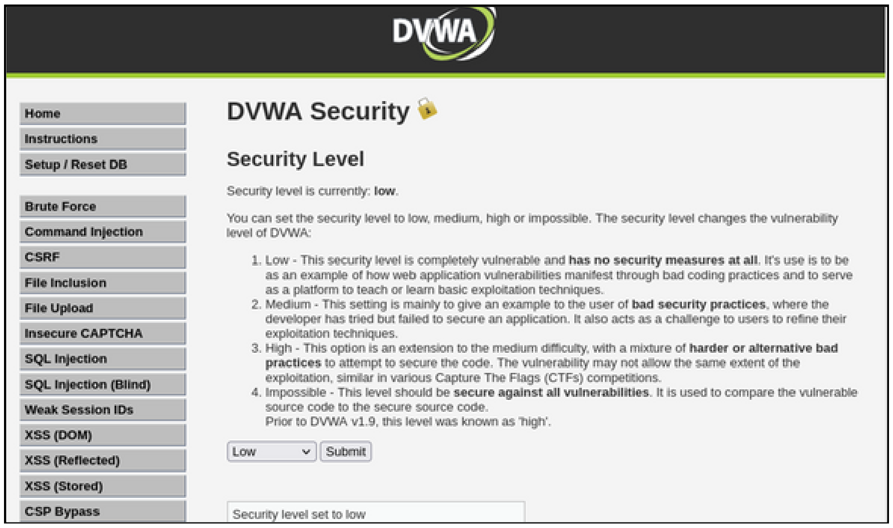


Username

admin

Password

••••••••



DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

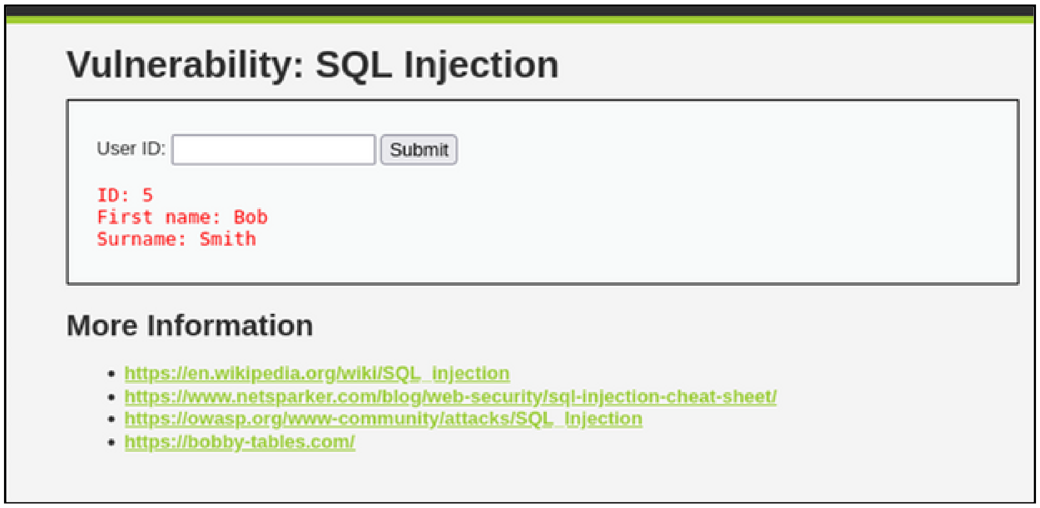
1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

Low

Security level set to low

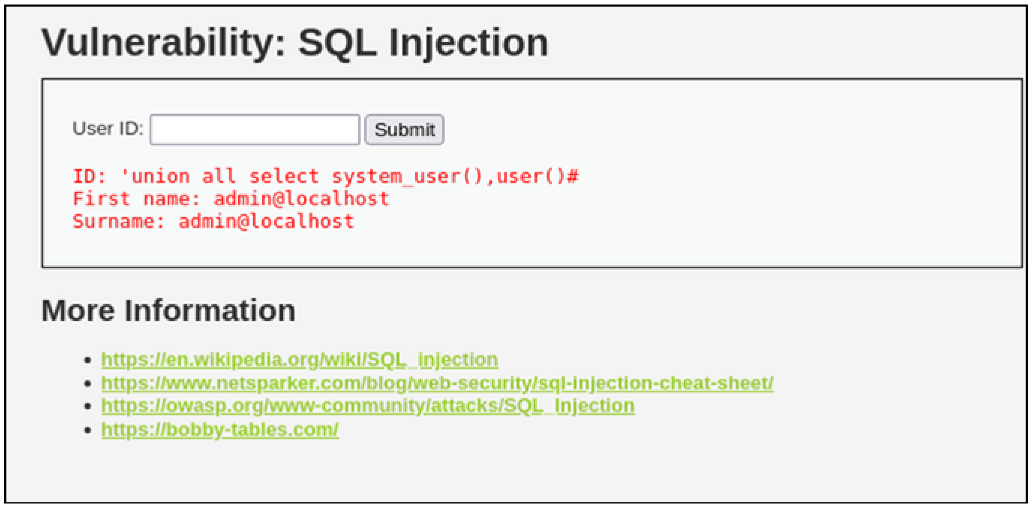
Appendix

1.SQL Injection


Vulnerability: SQL Injection
User ID:
Submit

```
ID: 5
First name: Bob
Surname: Smith
```

More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/

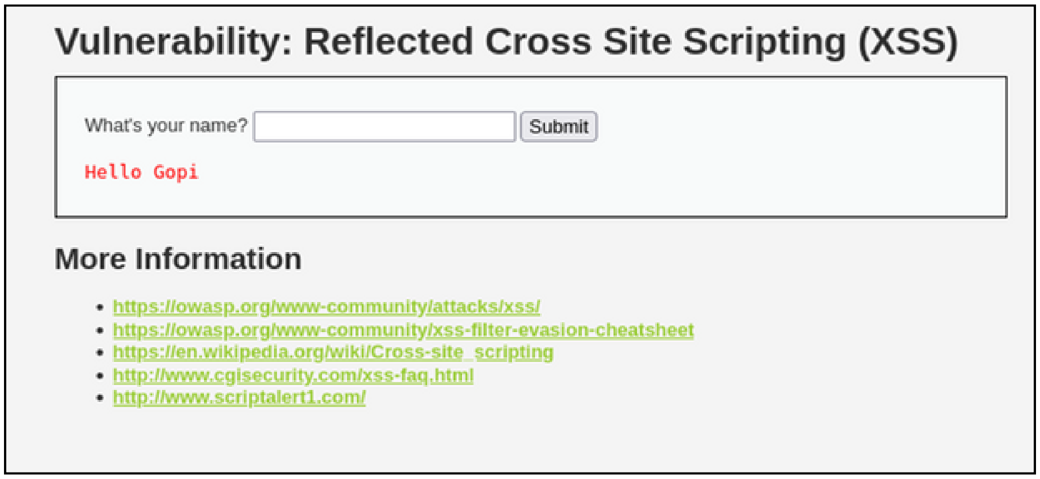

Vulnerability: SQL Injection
User ID:
Submit

```
ID: 'union all select system_user(),user()#
First name: admin@localhost
Surname: admin@localhost
```

More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/

Appendix

2.Reflected XSS



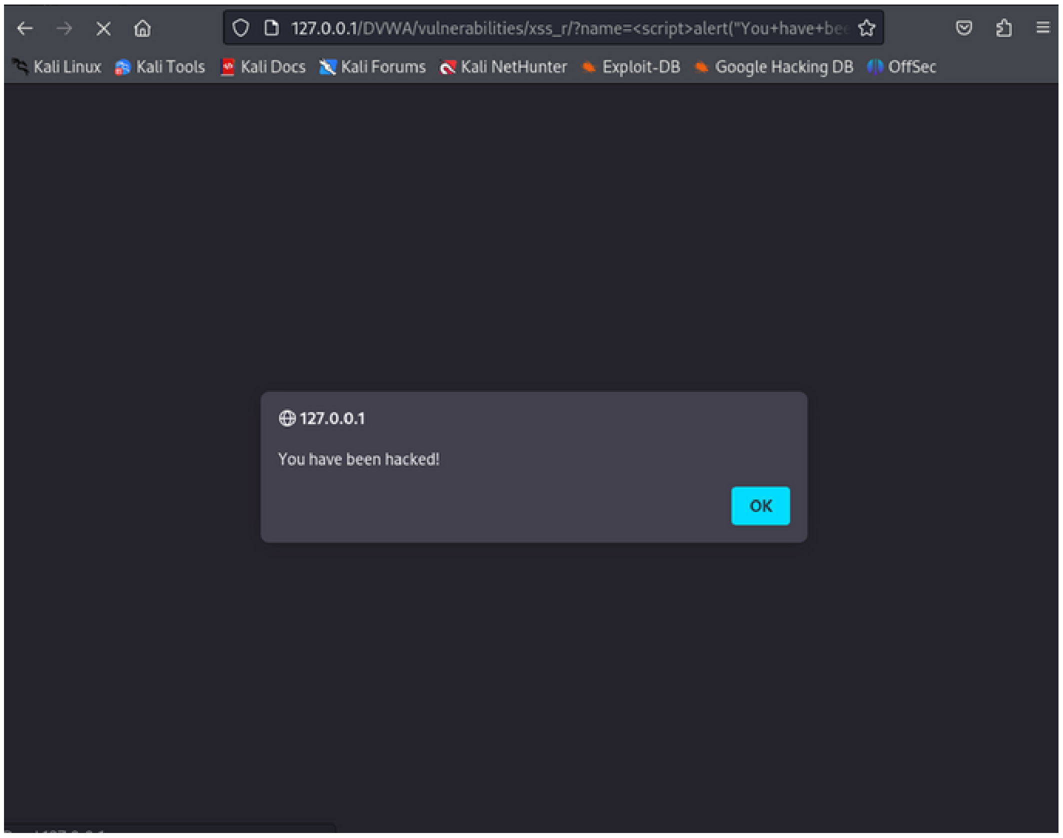
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello Gopi

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/



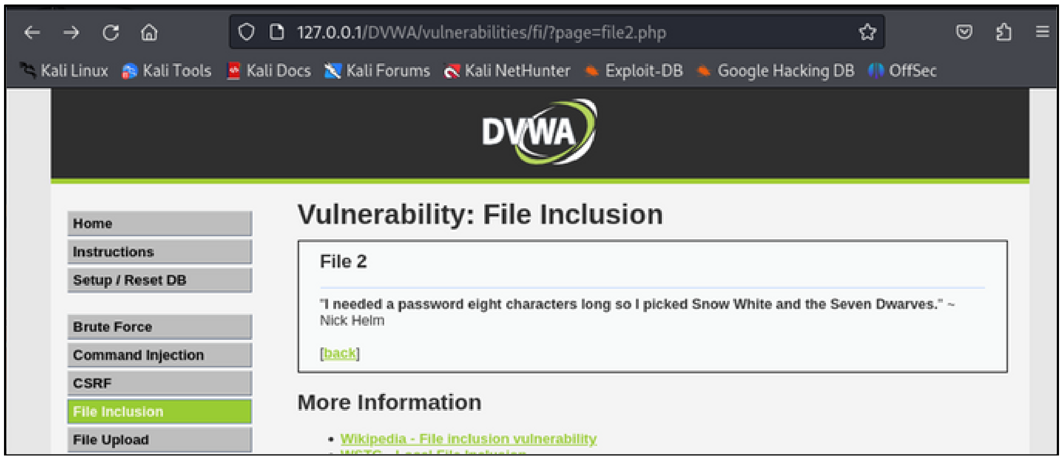
127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert("You+have+bee

127.0.0.1

You have been hacked!

Appendix

3. Local File Inclusion



127.0.0.1/DVWA/vulnerabilities/fi/?page=file2.php

Vulnerability: File Inclusion

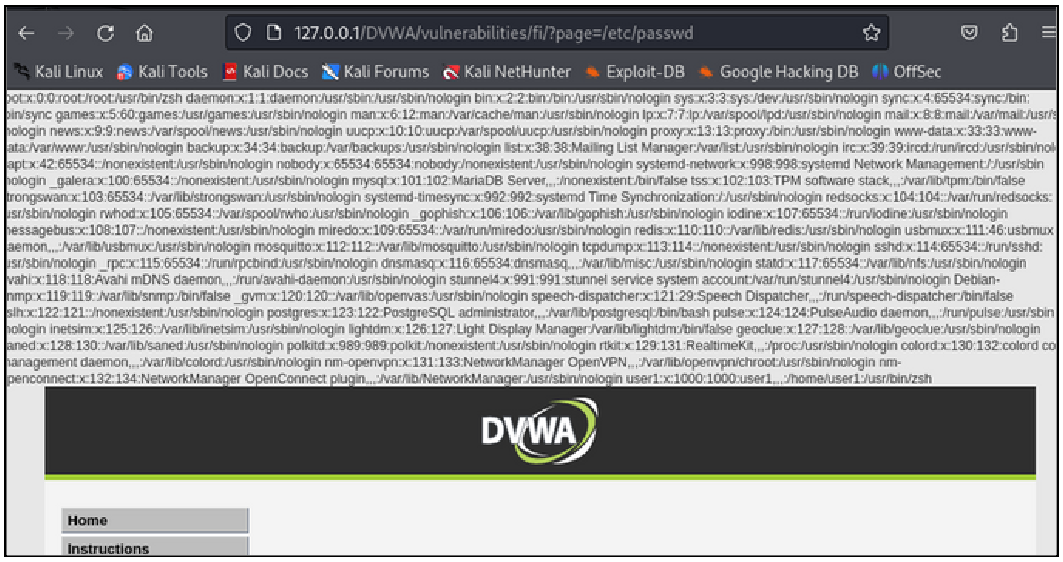
File 2

"I needed a password eight characters long so I picked Snow White and the Seven Dwarves." ~ Nick Helm

[back]

More Information

- Wikipedia - File inclusion vulnerability

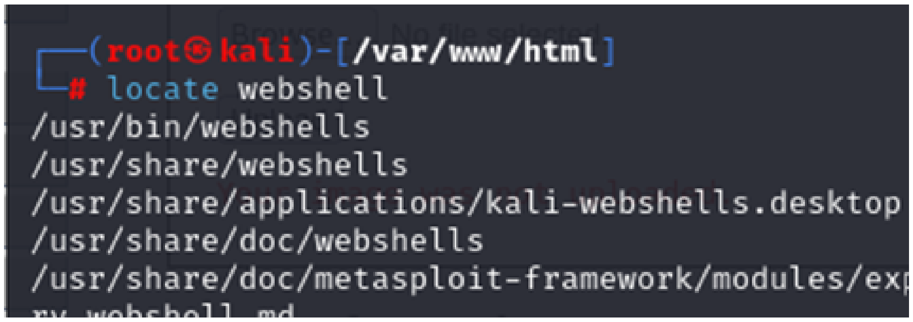


127.0.0.1/DVWA/vulnerabilities/fi/?page=/etc/passwd

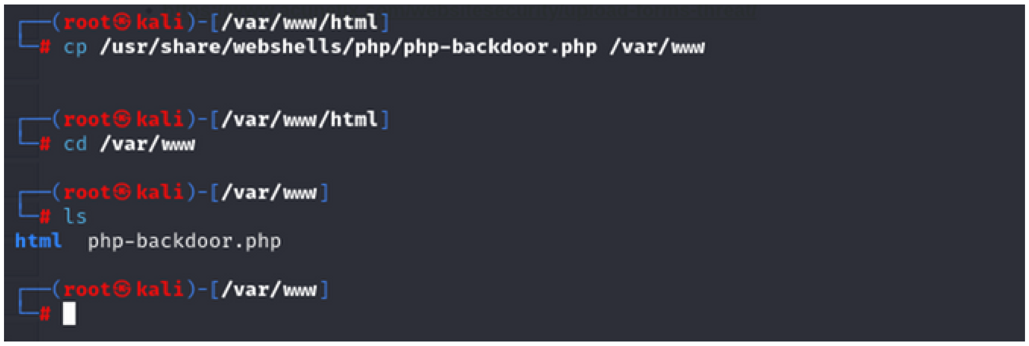
```
oot:x:0:0:root:/root:/usr/bin/zsh daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:
bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/s
ologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
ata:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nol
apt:x:42:65534::/nonexistent:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:998:998:systemd Network Management:/:/usr/sbin
ologin _galera:x:100:65534::/nonexistent:/usr/sbin/nologin mysql:x:101:102:MariaDB Server,,,:/nonexistent:/bin/false tss:x:102:103:TPM software stack,,,:/var/lib/tpm:/bin/false
trongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin redsocks:x:104:104::/var/run/redsocks:
usr/sbin/nologin rwhod:x:105:65534::/var/spool/rwho:/usr/sbin/nologin _gophish:x:106:106::/var/lib/gophish:/usr/sbin/nologin iodine:x:107:65534::/run/iodine:/usr/sbin/nologin
essagebus:x:108:107::/nonexistent:/usr/sbin/nologin miredo:x:109:65534::/var/run/miredo:/usr/sbin/nologin redis:x:110:110::/var/lib/redis:/usr/sbin/nologin usbmux:x:111:46:usbmux
aemon,,,:/var/lib/usbmux:/usr/sbin/nologin mosquitto:x:112:112::/var/lib/mosquitto:/usr/sbin/nologin tcpdump:x:113:114::/nonexistent:/usr/sbin/nologin sshd:x:114:65534::/run/sshd:
usr/sbin/nologin _rpc:x:115:65534::/run/rpcbind:/usr/sbin/nologin dnsmasq:x:116:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin statd:x:117:65534::/var/lib/nfs:/usr/sbin/nologin
vahi:x:118:118:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin stunnel4:x:991:991:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin Debian-
nmp:x:119:119::/var/lib/snmp:/bin/false _gvm:x:120:120::/var/lib/openvas:/usr/sbin/nologin speech-dispatcher:x:121:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
slh:x:122:121::/nonexistent:/usr/sbin/nologin postgres:x:123:122:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash pulse:x:124:124:PulseAudio daemon,,,:/run/pulse:/usr/sbin
ologin inetsim:x:125:126::/var/lib/inetsim:/usr/sbin/nologin lightdm:x:126:127:Light Display Manager:/var/lib/lightdm:/bin/false geoclue:x:127:128::/var/lib/geoclue:/usr/sbin/nologin
aned:x:128:130::/var/lib/saned:/usr/sbin/nologin polkitd:x:989:989:polkit:/nonexistent:/usr/sbin/nologin rtkit:x:129:131:RealtimeKit,,,:/proc:/usr/sbin/nologin colord:x:130:132:colord co
anagement daemon,,,:/var/lib/colord:/usr/sbin/nologin nm-openvpn:x:131:133:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin nm-
penconnect:x:132:134:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager:/usr/sbin/nologin user1:x:1000:1000:user1,,,:/home/user1:/usr/bin/zsh
```

Appendix

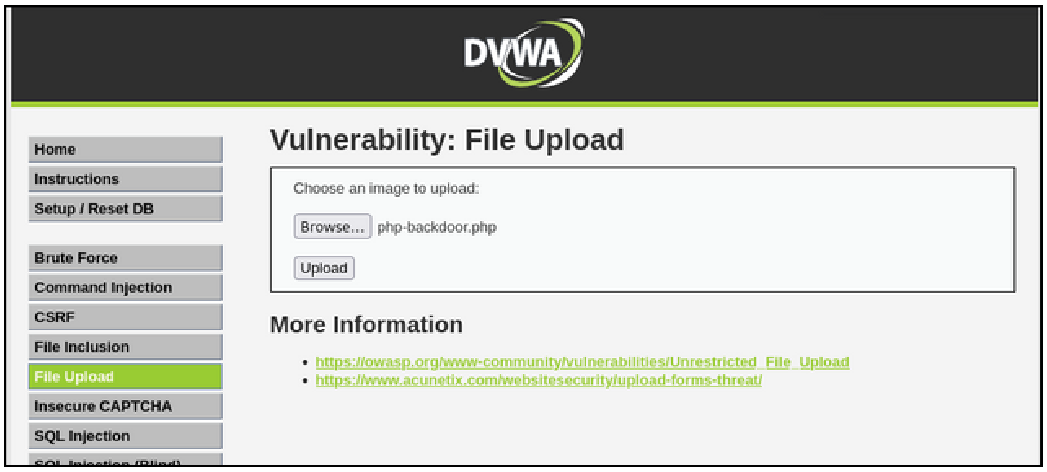
4.Web Shell



```
┌──(root㉿kali)-[/var/www/html]
└─# locate webshell
/usr/bin/webshells
/usr/share/webshells
/usr/share/applications/kali-webshells.desktop
/usr/share/doc/webshells
/usr/share/doc/metasploit-framework/modules/exp
ry_webshell.md
```



```
┌──(root㉿kali)-[/var/www/html]
└─# cp /usr/share/webshells/php/php-backdoor.php /var/www

┌──(root㉿kali)-[/var/www/html]
└─# cd /var/www

┌──(root㉿kali)-[/var/www]
└─# ls
html  php-backdoor.php

┌──(root㉿kali)-[/var/www]
└─# █
```



Vulnerability: File Upload

Choose an image to upload:

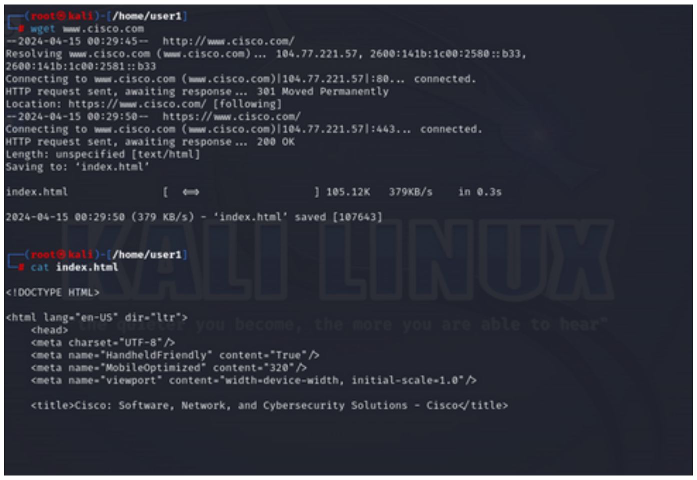
php-backdoor.php

More Information

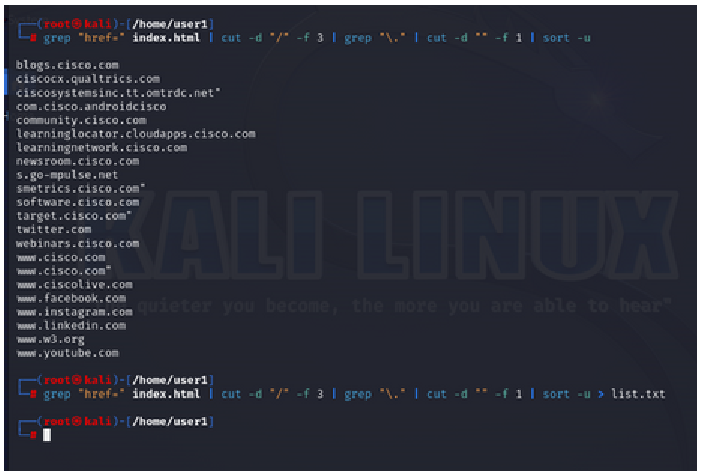
- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Appendix

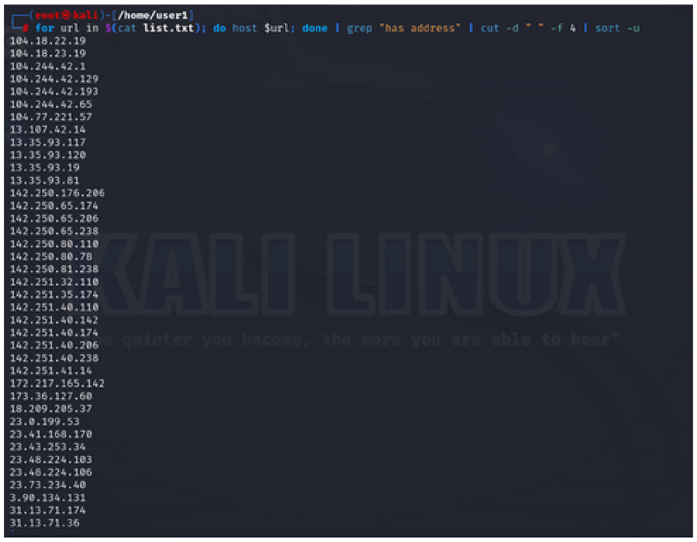
5.WGET



```
┌──(root㉿kali)-[/home/user1]
└─# wget www.cisco.com
--2024-04-15 00:29:45--  http://www.cisco.com/
Resolving www.cisco.com (www.cisco.com)... 104.77.221.57, 2600:141b:1c00:2580::b33,
2600:141b:1c00:2581::b33
Connecting to www.cisco.com (www.cisco.com)|104.77.221.57|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.cisco.com/ [following]
--2024-04-15 00:29:50--  https://www.cisco.com/
Connecting to www.cisco.com (www.cisco.com)|104.77.221.57|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

index.html              [  <=>                  ] 105.12K   379KB/s    in 0.3s

2024-04-15 00:29:50 (379 KB/s) - 'index.html' saved [107643]


┌──(root㉿kali)-[/home/user1]
└─# cat index.html

<!DOCTYPE HTML>

<html lang="en-US" dir="ltr">
    <head>
    <meta charset="UTF-8"/>
    <meta name="HandheldFriendly" content="True"/>
    <meta name="MobileOptimized" content="320"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

    <title>Cisco: Software, Network, and Cybersecurity Solutions - Cisco</title>
```



```
┌──(root㉿kali)-[/home/user1]
└─# grep "href=" index.html | cut -d "/" -f 3 | grep "\." | cut -d "" -f 1 | sort -u

blogs.cisco.com
ciscocx.qualtrics.com
ciscosystemsinc.tt.omtrdc.net"
com.cisco.androidcisco
community.cisco.com
learninglocator.cloudapps.cisco.com
learningnetwork.cisco.com
newsroom.cisco.com
s.go-mpulse.net
smetrics.cisco.com"
software.cisco.com
target.cisco.com"
twitter.com
webinars.cisco.com
www.cisco.com
www.cisco.com"
www.ciscolive.com
www.facebook.com
www.instagram.com
www.linkedin.com
www.w3.org
www.youtube.com

┌──(root㉿kali)-[/home/user1]
└─# grep "href=" index.html | cut -d "/" -f 3 | grep "\." | cut -d "" -f 1 | sort -u > list.txt

┌──(root㉿kali)-[/home/user1]
└─# 
```



```
┌──(root㉿kali)-[/home/user1]
└─# for url in $(cat list.txt); do host $url; done | grep "has address" | cut -d " " -f 4 | sort -u
104.18.22.19
104.18.23.19
104.244.42.1
104.244.42.129
104.244.42.193
104.244.42.65
104.77.221.57
13.107.42.14
13.35.93.117
13.35.93.120
13.35.93.19
13.35.93.81
142.250.176.206
142.250.65.174
142.250.65.206
142.250.65.238
142.250.80.110
142.250.80.78
142.250.81.238
142.251.32.110
142.251.35.174
142.251.40.110
142.251.40.142
142.251.40.174
142.251.40.206
142.251.40.238
142.251.41.14
172.217.165.142
173.36.127.60
18.209.205.37
23.0.199.53
23.41.168.170
23.43.253.34
23.48.224.103
23.48.224.106
23.73.234.40
3.90.134.131
31.13.71.174
31.13.71.36
```

References

1. https://www.w3schools.com/sql/sql_injection.asp#:~:text=SQL%20injection%20is%20a%20code,statements%2C%20via%20web%20page%20input.
2. https://www.hackingarticles.in/cross-site-scripting-exploitation/
3. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion#:~:text=The%20File%20Inclusion%20vulnerability%20allows,implemented%20in%20the%20target%20application.
4. https://www.imperva.com/learn/application-security/web-shell/
5. https://www.indusface.com/blog/web-application-security-challenges/